Inverse Procedural Modelling and Applications

A dissertation submitted to

ETH ZURICH

for the degree of

Doctor of Sciences (Dr. sc. ETH Zürich)

presented by

Julien Weissenberg

MSc. Advanced Computing, Imperial College London
Ingénieur diplômé ENSIIE, Evry, France
born July 11, 1986
citizen of France

accepted on the recommendation of

Prof. Dr. Luc Van Gool, examiner
Prof. Dr. Nikos Paragios, co-examiner

2014
TO MY FAMILY.
Abstract

Architects, urban planners, the film and gaming industry as well as map makers are all very much interested in detailed, semantic city models. While generating a synthetic city has become much easier thanks to procedural modelling, automatically modelling an existing city remains a challenge. With procedural modelling, a building is described as a procedure, wherein an initial shape is successively refined according to a formal grammar set of rules. In this work, we present a set of methods to automatically instantiate and generate such sets of rules, from pictures.

First, we present the applications and challenges behind city modelling. Shape grammars have greatly contributed to making city modelling easier. From there, a large spectrum of applications became possible, such as generating realistic large-scale virtual cities or fine-grained urban planning simulations. We introduce the shape grammar framework and explain how it is used to represent buildings. Nevertheless, when it comes to modelling an existing city, much of the work remains a tedious, manual task. The first steps to an automatic pipeline include facade extraction and style detection. Next, we distinguish between two cases: the first, where a pre-defined set of rules is available and can be used to support detections of architectural elements; the second, where the set of rules is automatically inferred from a facade layout.

In the first case, the set of rules is used as a powerful tool to support 3D reconstruction. In more detail, the goal is to find the parameters of a grammar for a specific building structure. The rules are used to constraint the search and infer hard-to-detect parameters from initial detections. We demonstrate the use of rules for reconstruction of Doric style temples. We show how the initial Structure-from-Motion reconstruction and detections can be leveraged by using rules that make it possible to automatically restore the ruins.

However, it is the exception that a pre-defined set of rules is available. In fact, architects systematically transgress the rules they establish. Furthermore, we must account for the wide spectrum of architectural styles and the very large number of buildings to be reconstructed. Therefore, automation of large-scale city reconstruction implies inferring these sets of rules automatically. We present a method to automatically infer such set of rules. In addition to reconstruction, we show how to use the inferred rules for compression, comparison and virtual facade synthesis.

More than tools to reconstruct cities, this work tells about the logic of architecture. Finally, it is a unique opportunity to analyze the relations between language and meaning.
Résumé

Les architectes, tout comme l’industrie du cinéma et du jeu vidéo, ou encore les cartographes et urbanistes, ont besoin de moèles de villes sémantiques et détaillés. Si la modélisation procédurale a rendu la synthèse d’une ville virtuelle bien plus facile, la modélisation d’une ville existante reste un défi. En modélisation procédurale, un bâtiment est défini par une procédure, qui affine progressivement un bloc de départ. Nous traitons dans cette thèse d’un ensemble de méthodes pour automatiquement déterminer les règles et leurs paramètres partir de photos.

Tout d’abord, nous présentons les applications et défis liés la modélisation urbaine. Les grammaires formelles ont ouvert la voie à la synthèse de mégapoles virtuelles ou encore de simulations urbaines précises. Nous présentons les grammaires de formes et expliquons comment les utiliser pour décrire un bâtiment. Néanmoins, lorsqu’il s’agit de représenter des villes existantes, le travail reste en grande partie manuel. La détection du style et l’extraction des façades constituent les premières étapes vers une automatisation. Ensuite, il faut distinguer deux cas. Dans le premier, un ensemble de règles est pré-défini et sert améliorer les détectes des éléments architecturaux. Dans le second, les règles sont déterminées automatiquement à partir d’images de façades. Dans le premier cas, les règles sont un outil puissant pour la reconstruction 3D. Plus précisément, le but est de trouver les paramètres d’une grammaire pour une structure de bâtiment donnée. Les règles permettent de restreindre l’optimisation et ainsi de déterminer des paramètres difficiles à détecter à partir des observations. Une étude de cas sur les temples doriques montre comment les règles permettent de tirer partie d’une reconstruction 3D et des détections pour restaurer automatiquement les ruines.

Néanmoins, la disponibilité de règles relève de l’exception. En fait, les architectes transgressent systématiquement les règles qu’ils établissent. De plus, il faut tenir compte du panel de styles et du grand nombre de bâtiments à reconstruire. Par conséquent, l’automatisation pour la reconstruction à grande échelle passe par la création automatique de telles règles. Nous présentons une méthode pour ce faire. De plus, nous montrons l’utilité de ces règles afin de compresser, comparer et synthétiser des façades.

Au delà de la création d’outils pour la reconstruction de villes, cette thèse porte sur la logique même de l’architecture. Enfin, il s’agit d’une opportunité d’analyser les relations entre sens et langage.
Acknowledgements

Thank you first of all to Professor Luc Van Gool for offering me the opportunity of doing research at the Computer Vision Lab.

His genuine passion and great knowledge about the topic made the PhD a delight. He also gave me the unvaluable freedom to explore my own research interests.

Also, I would like to warmly thank Hayko Riemenschneider for making my contribution stronger, for his valuable guidance and human qualities.

No others deserve more thank than everyone at CVL and from 3DCOFORM and VarCity projects, from Zurich to Leuven I will keep very joyful memories.

Keeping in mind the importance of encouragements, I would especially like to thank my office colleagues for making C108 the funniest office, of my friends and family for always being there and being who they are.

Starting with the Computer Vision community I would like to finish with the architecture one. Thank you very much to everyone at Information Architecture whose knowledge was very helpful.
Contents

List of Figures xiv

List of Tables xx

1 City Modelling: applications, methods and challenges 1
  1.1 Architecture and Urbanism .............................. 1
  1.2 Digital fabrication ..................................... 3
  1.3 Maps .................................................. 4
  1.4 Film and video games .................................. 6
  1.5 Cultural heritage ...................................... 7
  1.6 Challenges in City Modelling ......................... 8
  1.7 Contributions ......................................... 9
  1.8 Overview ............................................. 10

2 A language for architecture 11
  2.1 3D data acquisition .................................... 11
      2.1.1 LiDAR vs. Structure-from-Motion ................. 11
  2.2 Airborne vs. Ground imagery ......................... 13
      2.2.1 Picture acquisition ............................... 13
  2.3 From visual realism to abstraction .................. 13
  2.4 A language for architecture ......................... 14
      2.4.1 Architectural theory .............................. 14
      2.4.2 Computational design ............................. 16
      2.4.3 Shape grammars .................................. 17
      2.4.4 L-systems ........................................ 19
      2.4.5 Split grammars .................................. 20
  2.5 CGA shape grammars .................................. 21
      2.5.1 Properties ........................................ 22
      2.5.2 Syntax ............................................ 22
      2.5.3 Shape ............................................. 22
      2.5.4 Operations ....................................... 22
      2.5.5 Derivation tree .................................. 23
## Contents

2.5.6 CGA example ........................................... 23
2.6 GML ....................................................... 24

### 3 Approach

3.1 Pre-processing steps ...................................... 25
3.1.1 Style detection ....................................... 25
3.1.2 Facade extraction ..................................... 25
3.2 Top-down vs. bottom-up .................................. 25
3.3 Pipeline .................................................. 27

### 4 Style recognition

4.1 Introduction ............................................. 29
4.1.1 Related work .......................................... 30
4.2 System overview ........................................ 31
4.3 Scene classification ...................................... 31
4.3.1 Scene classes ......................................... 32
4.3.2 Feature extraction and classification ............... 32
4.3.3 Results ............................................... 33
4.4 Image rectification ....................................... 33
4.5 Facade splitting .......................................... 35
4.5.1 Line segment detection and grouping ............... 35
4.5.2 Vertical line sweeping ............................... 36
4.5.3 Results ............................................... 36
4.6 Style classification ...................................... 37
4.6.1 NBNN algorithm ...................................... 38
4.6.2 Results ............................................... 38
4.7 Conclusion and future work ............................ 39

### 5 Facade segmentation

5.1 Introduction ............................................. 41
5.2 Related Work ............................................ 43
5.3 Datasets description ..................................... 44
5.4 Bottom Layer: Recursive Neural Network for Semantic Segmentation ........................................ 45
5.5 Middle Layer: Introducing Objects Through Detectors ..................................................... 48
5.6 Top Layer: Weak Architectural Principles ............. 50
5.6.1 Parameters ............................................ 52
5.7 Results ................................................... 52
5.8 Conclusion and future work ............................ 54

### 6 Learning Where To Classify In Multi-View Semantic Segmentation

6.1 Introduction ............................................. 55
6.2 Related Work ............................................ 57
6.3 3D Surface and Semantic Classification .................. 60
6.3.1 Multi-View Surface Reconstruction .......................... 60
6.3.2 Heavy vs. Light Features for Semantic Labelling .............. 60
6.3.3 Multi-View Optimization For 3D Surface Labelling ............ 61

6.4 Multi-view Observation Importance ............................ 63
6.4.1 Ranking Observations by Importance .......................... 63
6.4.2 Reducing View Redundancy and Scene Coverage ............... 64

6.5 Experimental Evaluation ......................................... 65
6.5.1 Single Discriminative Views - Zero Redundancy ................. 67
6.5.2 Reduction of Scene Coverage .................................. 69

6.6 Conclusions ......................................................... 69

7 Rules for reconstructions ............................................ 71
7.1 Introduction ......................................................... 71
7.2 Related work ...................................................... 73
7.3 Main system components .......................................... 74
7.3.1 Grammar Interpreter .......................................... 75
7.3.2 Asset detector .................................................. 78
7.3.3 3D reconstruction module ...................................... 78
7.3.4 Vision module .................................................. 79
7.4 Grammar attribute estimation ...................................... 83

7.5 Case Study - Doric Temples ........................................ 84
7.5.1 Asset Detectors .................................................. 84
7.5.2 Temple Grammar ............................................... 84
7.5.3 Results ......................................................... 85

7.6 Conclusion and Future Work ....................................... 85

8 Grammatical inference ............................................... 87
8.1 Introduction ......................................................... 87
8.2 Related work ...................................................... 89
8.3 Approach .......................................................... 90
8.3.1 Shape grammars for architecture .............................. 90
8.3.2 What makes a good grammar? .................................. 91
8.3.3 Fâade parse tree generation ................................... 92
8.4 Optimization of Shape Grammars ................................ 94
8.4.1 Grammatical inference ......................................... 94
8.4.2 Compression ..................................................... 96
8.4.3 Comparing fâades ............................................... 96
8.4.4 Virtual fâade synthesis ......................................... 97

8.5 Evaluation .......................................................... 97
8.5.1 Experimental setup .............................................. 97
8.5.2 Losslessness and computational cost .......................... 98
8.5.3 Compression ..................................................... 99
8.5.4 Façade comparison ........................................ 99
8.5.5 Virtual façade synthesis ................................. 100
8.6 Conclusion ..................................................... 101

9 Next generation navigation ............................... 103
  9.1 Introduction .................................................. 103
  9.2 Related work ................................................ 105
  9.3 Approach Overview ........................................ 107
  9.4 What makes a facade unique? ............................ 108
    9.4.1 Notations .............................................. 109
    9.4.2 Features ............................................... 109
    9.4.3 Outlier detections .................................... 111
    9.4.4 Mapping to the human perceptual space ........... 112
    9.4.5 Clustering ............................................. 114
  9.5 Experimental Evaluation ................................. 114
    9.5.1 Special facade mining ................................. 116
    9.5.2 Perceptual study for characterization ............... 117
    9.5.3 Facade characterization ............................... 117
  9.6 Conclusion .................................................. 118

10 Conclusion .................................................. 121
  10.1 The logic of architecture ............................... 121
  10.2 Grammars as compression-based models ............... 122
  10.3 Outlook .................................................... 122
List of Figures

1.1 Duomo of Florence, Santa Maria del Fiore, designed by the Renaissance architect Brunelleschi in 1418. Drawing by Cigoli (left), wood model of the Dome by Brunelleschi (centre), current view of the dome (right). ................................................................. 2
1.2 Left: Pipe network of the BIM model of Cathedral Hill Hospital, designed by HerreroBoldt on Autodesk. Right: Robot constructing a wall for the Domoterra Lounge Basel, Gramazio and Kohler, Architecture and Digital Fabrication, ETH Zurich 2007 ......................... 3
1.3 Top: Detail from a hand-drawn map of Hoorn, by Johannes Blaeu, 1649. Bottom: Map of London and North Eastern Railway by George Dow, 1929. ................................................................. 4
1.4 Google bird’s-eye view (left), map (bottom-right) and street view (top-right) of Muensterhof, Zurich. ................................................................. 5
1.5 Cityscapes in Ratatouille, in the style of Paris (top), Total Recall 2012, in the style of a futuristic London (bottom). ............................. 6
1.6 The skyline of Los Santos, an imaginary city in the game Grand Theft Auto V based on Los Angeles. ................................................................. 7
1.7 Blended image of an original picture and the reconstruction of the temple of Hera, Paestum, Italy. ................................................................. 7

2.1 From top to bottom: Sparse SfM point cloud, Dense mesh, Dense textured mesh of Seilergraben, Zurich obtained with the CMPMVS pipeline [Jancosek & Pajdla 2011a]. ................................................................. 12
2.2 LiDAR scan detail of the town hall in Klodzko, Poland, obtained with a ZF imaginer 5003 laser scanner (image courtesy of 3deling). ....c 12
2.3 The Vitruvian Man, Leonardo da Vinci, ca. 1490, was inspired by [Vitruvius 1914]. The figure depicts idealized body proportions, which should in turn be taken into account in architectural designs. 15
2.4 From regularity to uniqueness. From left to right: Haussmanian, Gruenderzeit, Baroque buildings, Eiffel Tower, Kunsthaus Graz [from H. Riemenschneider]. ................................................................. 16
<table>
<thead>
<tr>
<th>FIGURE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.5</td>
<td>The first three iterations of the construction of the Koch snowflake (image courtesy of Wikipedia).</td>
</tr>
<tr>
<td>2.6</td>
<td>Shape grammar example and derivations, from [Stiny 1975].</td>
</tr>
<tr>
<td>2.7</td>
<td>Left: Sample tree and Right: Synthetic landscape with self-organizing trees, from [Palubicki et al. 2009].</td>
</tr>
<tr>
<td>2.8</td>
<td>CGA shape grammar shape definition, from [Müller et al. 2006b]. P corresponds to the pivot, S to the scope (outlined by the grey box) while a sample shape is shown within the scope.</td>
</tr>
<tr>
<td>2.9</td>
<td>Example of a facade decomposition, from [Müller et al. 2007].</td>
</tr>
<tr>
<td>3.1</td>
<td>Inverse procedural modelling pipeline. First, images are fed into an SfM/PMVS pipeline. Then, facades are extracted. Once the style has been determined (see Chapter 4), there are two possibilities: if a style grammar is available (see Chapter 7), it can be used to parse the facade (top-down). Otherwise (see Chapters 5 and 6), the facade can be labelled using weaker priors (bottom-up). In the latter case, a set of procedural rules can then be inferred and is useful for virtual facade synthesis, facade comparison, compression (see Chapter 8).</td>
</tr>
<tr>
<td>4.1</td>
<td>Style classification system overview.</td>
</tr>
<tr>
<td>4.2</td>
<td>Rectification process: (a) input image with dominant lines, (b) projective distortion removal (c) affine distortion removal (d) similarity transformation.</td>
</tr>
<tr>
<td>4.3</td>
<td>Facade splitting algorithm.</td>
</tr>
<tr>
<td>4.4</td>
<td>Style detection: a) Neoclassical style (features in red), b) Haussmannian style (features in blue), c) Renaissance style (features in purple) and d) Unknown style (features in green).</td>
</tr>
<tr>
<td>5.1</td>
<td>The proposed three-layered approach to facade parsing.</td>
</tr>
<tr>
<td>5.2</td>
<td>Basic RNN structure. Two input segments are transformed into a semantic space and merged into a supersegment. The supersegment’s semantic vector can be recursively combined with other semantic vectors by repeating the same network structure $W$, $W^{score}$ and $W^{label}$.</td>
</tr>
<tr>
<td>5.3</td>
<td>Left: Precision-recall curve of our window detector and an example image with detector output. Right: Statistics calculated for the window detections on a validation set of the ECP dataset. Red color indicates window class, green wall, blue balcony. Other labels have negligible influence. As we increase the number of selected window detections (threshold), we introduce more false positives. This reflects in the reduction of window class probability.</td>
</tr>
</tbody>
</table>
5.4 Examples of top-layer output on various buildings from the ECP dataset. 

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

6.1 View overlap is ignored by existing work in semantic scene labelling, and features in all views for all surface parts are extracted redundantly and expensively (top left). In turn, we propose a fine-grained view selection (top right), as well as to reduce scene coverage (bottom left) by only classifying regions essential in terms of classification accuracy. The labels of the classified regions are then spread into all regions (bottom right). This sparsity increases efficiency by orders of magnitude, while also increasing the accuracy of the final result (bottom right vs. top left).

6.2 Dataset overview - most are coarsely labelled at low resolution. We use a pixel accurate labelling with fine details at 1-3 megapixel resolution. (rightmost).

6.3 Features like color and gradient filters are expensive since they are densely calculated in the entire image. Geometry-based are more light-weight. Extra features like denseSIFT should improve the baseline, yet are even heavier to calculate.

6.4 Geometric link between 3D model and 2D image space. Contrary to related work in view clustering, we look for the best view \( c^* (f_i) \) per mesh triangle \( f_i \). For small viewing angles the texture is visually pleasing but not best for semantic classification.

6.5 Removing View Redundancy: showing accuracy for the single k-th ranked feature on x-axis (e.g. 1st largest area, 10th smallest angle, 4th learned importance) and average feature value (red dash). The smaller the area or the larger the angle, the worse performance gets. Our learned performance captures the combination of area and angle better. This is CamVid, other datasets are in supplemental material.

6.6 Reducing Scene Coverage: showing accuracy over percentage of selected triangles within graph optimization. Dashed lines are accuracy at full coverage (allviews, maxarea, minangle, importance). On average 30% are sufficient to label the entire scene as correctly as 100% coverage! Last rows show classwise results (see text for details).
6.7 Overview of results - top left is full street, view redundancy as heatmap (more redundancy, the greener), ground truth (zoomed for two parts of street), and results for full redundancy, single best view and best score for coverage (at stable 30%). Overall, the accuracy are the same after all our speedups. Middle column shows failure cases (\(\times\)), where the initial classifier already fails and gracefully further smoothes the results. ................................. 70

7.1 Reconstruction of the Temple of Poseidon in Paestum, Italy    72
7.2 The proposed system. .................................................. 73
7.3 An example CGA grammar is shown on the left. The resulting shape tree is in the middle, and the rendered model with the default values on the right. .................................................. 77
7.4 Comparison of the general detector and the retrained specialized one. .................................................. 78
7.5 Plane estimation process: The first image shows the entire point cloud, then the planes are estimated in the reduced point cloud and shown in the cleaned complete point cloud. ......................... 80
7.6 Determining the assets size: the red and green arrows indicate the estimated height and width respectively. ......................... 81
7.7 Similarity voting space for a single detection (red rectangle). . . . 82
7.8 Reconstruction of the Parthenon replica in Nashville and the Temple of Athena. .................................................. 84

8.1 Pure mesh-based or semantic labelled models suffer from a limited field of uses. In this work we propose methods to automatically build procedural façade models in milliseconds for compression, comparison and new virtual façade creation. ................................. 88
8.2 Our inverse procedural modelling pipeline. First, labellings of rectified façade pictures are the input to our method. A split tree is inferred for each façade. Those trees are combined and constitute a set of rules that describe the façade style. These inferred rules are then used for comparison, editing, virtual façade synthesis, and rendering. .................................................. 90
8.3 Distance matrix for the ECP2011 and Graz2012 datasets, re-ordered according to its dendrogram (log scale). Some of the associated labellings are shown on the right. ................................. 98
8.4 Cumulative Match Characteristics (CMC) for ECP2011 for semantic façade retrieval. The powersets distance better captures the structural similarities over common rules distance. ................................. 98
8.5 Computational time with respect to the number of Haussmannian façades given as input (left); number of rules with respect to the number of Haussmannian (centre) and Gruenderzeit (right) façades given as input, using no compression, n-ary compression and n-ary and rule inference compression (log scale).

8.6 Rows correspond to different façade structure type (i.e. different starting rules), columns correspond to different $\sigma$ values which influence parameter variations.

9.1 Our automatically mined local landmark buildings maps (top) vs. classical separated Street View (middle) vs. Street network (bottom) maps and street view imagery are respectively very abstract or hard to browse. If major landmarks are sometimes annotated on tourist maps, these are too sparse. We automatically mine atypical buildings and highlight them at a given scale.

9.2 Overview of our approach: Features are extracted from facades and their labelings, Outliers are detected and clustered. The results are re-ranked according to human perception. New applications such as relative text description, facade search, speech for navigation and the unusualness map of a city arise.

9.3 Different types of pediments, from [Larousse 1898-1904]. Although the architectural variety is very large, a non-specialist does not have the required knowledge to distinguish between all styles.

9.4 Unusualness (green) vs. Noise (red) - the construction work (Left) and car and pedestrian (Right) are distinctive image features but are temporary occluding the facade. In contrast, the black shutters (Left) and brown door (Right) help distinguish between the two facades.

9.5 Description features - instead of image patches (left: discriminant vs. SIFT patches [Doersch et al. 2012]), we extract size and color attributes (right) which are easily phrased into natural language.

9.6 Interface for the Amazon Mechanical Turk facade unusualness collection. As instruction, participants were asked to “select the words that best describe what is special about this building compared to two others” so that a friend could find it.

9.7 Heatmap representation of measured unusualness. Left: facade image, Middle: using LOF results, Right: after mapping according to human perception. Mapping places the emphasis on unusual colours (the shutters are bright blue) and assets which are cited more in the human perception study, e.g. shops. Note the very large difference between the two heatmaps: statistical unusualness needs to be re-mapped to match with human perception of unusualness.
9.8 Map highlighting the most unusual building around an itinerary in Graz. .......................................................... 115
9.9 The most salient (top) and least salient (bottom) facades in Graz50 as predicted by us. ................................. 118
9.10 Our automatic description of the facade in the middle: “The facade with tall blue shop and wide brown shop.” On right, the heatmap of the most unusual features. .............................................. 118
9.11 Our automatic description of the facade in the middle: “The facade with green wall and wide red windows.” On right, the heatmap again. 119
9.12 Zurich20 Analysis. Columns: Asset, feature type, colour and size usage. The vertical axis in the histograms refers to the number of times an asset or attribute was cited. Rows: Top: study results, Middle: LOF, Bottom: After regression. The regression helps to give descriptions closer the human-given annotations. Note that the material and shape are not used in the automatic unusualness inference. ............................................................. 119
9.13 TopK scores obtained for facade mining with our method (blue) against a randomness-based baseline (red) for Graz50 (left), ECP2011 (middle) and Zurich20 (right). The TopK score quantifies the agreement between the human and automatic rankings for the top K items. A perfect agreement gives a score of 1. .......................... 120
9.14 TopK scores obtained for facade characterization with our method (blue) against a randomness-based baseline (red) for Graz50 (left), ECP2011 (middle) and Zurich20 (right). ................................. 120
List of Tables

<table>
<thead>
<tr>
<th>Table</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.1</td>
<td>Comparision of bottom-up and top-down approach applicability.</td>
<td>27</td>
</tr>
<tr>
<td>4.1</td>
<td>Confusion matrix for the scene classification algorithm. The value in i-th row and j-th column represents the percentage the i-th class was labeled as j-th class.</td>
<td>33</td>
</tr>
<tr>
<td>4.2</td>
<td>Confusion matrix for the style classification algorithm. The value in i-th row and j-th column represents the percentage the i-th class was labeled as j-th class.</td>
<td>38</td>
</tr>
<tr>
<td>5.1</td>
<td>Weak architectural principles used to complement the segmentation results of the first 2 layers. A &quot;x&quot; in the &quot;alter&quot; column denotes that the principle adjusts element borders. The principle may also remove or add new elements. Last two columns indicate which principles are used for each of the datasets.</td>
<td>50</td>
</tr>
<tr>
<td>5.2</td>
<td>Results on ECP (left) and eTrims (right) dataset, for each of the layers of our system. Class accuracies are shown in percent, as well as the total pixel accuracy.</td>
<td>52</td>
</tr>
<tr>
<td>6.1</td>
<td>Summary of all results (details in supplemental). Semantic Segmentation accuracy (PASCAL IOU in %) for Full428, Sub28 and CamVid102 datasets. By reducing redundancy to zero and also scene coverage to 1/6th, we speedup by 2 orders of magnitude. Ranking by area is better than angle yet the 1st ranks are not best (bold).</td>
<td>66</td>
</tr>
<tr>
<td>7.1</td>
<td>Size comparison for the Temple of Poseidon.</td>
<td>86</td>
</tr>
<tr>
<td>9.1</td>
<td>Overview of the features used to describe facades. The features are listed in the first row and the second row gives an example for each feature. All these features were used to describe facades in the free text experiment, while we selected a subset for the guided labelling experiment and for automatic labelling.</td>
<td>110</td>
</tr>
</tbody>
</table>
9.2 Summary of the facade mining study results. RF: Random Forest, Corr: Pearson correlation. A correlation of 1.0 would mean that we can perfectly predict the distribution of the responses of humans for each facade.

9.3 Summary of the facade characterization study results. Corr: correlation, LOF: Local Outlier Factor, RF: Random Forest
City Modelling: applications, methods and challenges

Cities are among some of the most complex created systems [Allen & Sanglier 1981]. Since they are man-made, cities can be considered as simple systems in the first place. Complexity emerges from the interaction of human agents with the city and between themselves [Portugali 2000]. In particular, Portugali argues that the the behaviour of urban agents is affected by their urban environment. The architecture of a city affects its social and economic dynamics, as well as the development of its infrastructure and growth [Bettencourt 2013].

The demand for accurate city models historically stems from various fields, including architecture, urban planning, archaeology, map making, and the film and video game industry. The umbrella term city modelling therefore covers very different kinds of models. These models reflect the information which is needed to produce them and their intended usage. In the rest of this work, we will only deal with exterior and building models. Other architectural models comprise interior models and construction models, describing the structure of a building. Urban models additionally encompass the terrain elevation, street network model, vegetation, street furniture and other urban elements related to the city life: pedestrians, cars, stalls but also tower cranes, scaffolding and construction sites. We now present some of the traditional and pre-existing methods to procedural modelling associated with each application. Although some of them are distant from computer science, understanding traditional methods allows putting the requirements related to city modelling into perspective.

1.1 Architecture and Urbanism

Before the advent of computers, architects and urban planners would solely rely on drawings and wood models. These two tools complement each other very well. In particular,
wood models [Knoll & Hechinger 2007] (see Fig. 1.1) are used for visualization of complex shapes, as they let the viewer change perspective. Wood models are used at different time points of a project. They are crucial tools to test different ideas, sell a project and explain complex designs to builders. However, wood models are unsuitable for rapid prototyping. They suffer from a very long building time, making it impossible to alter them swiftly, for example in the course of a discussion with a client.

![Fig. 1.1: Duomo of Florence, Santa Maria del Fiore, designed by the Renaissance architect Brunelleschi in 1418. Drawing by Cigoli (left), wood model of the Dome by Brunelleschi (centre), current view of the dome (right).](image)

Also, a recent trend is to reduce the manual work involved in woods model by using 3D printed models instead.

In contrast to wood models, drawings [Schaller 1997] can be used to both quickly sketch an idea and detail its implementation. However, drawings do not allow perspective changes. Hence, technical drawings (see Fig. 1.1) are mostly suited for trained professionals.

Software has been developed to replicate the habits of architects, while overcoming some of the shortcomings of physical models. For instance, 3D modelling and rendering enables to create photo-realistic views of a design.

Beyond pure visualization, Building Information Models (BIM) allows the storage of all the information pertaining to a building in a single model. BIM was invented by Van Nederveen & Tolman [1992] and has been increasingly used since 2000. For a given project, the BIM files are used by all stakeholders: architects, engineers, contractors as well as building owners and managers. The benefits of using BIM are threefold: First, the building design is optimized for the final users. Second, the construction and deconstructions processes are optimized. Last, all the information is preserved. It should be noted that currently, a large part of the construction process is rework, accounting for 3% to 23% of the construction costs [Love & Li 2000, Love & Edwards 2004]. The possibility to build a BIM model for existing buildings would be beneficial for building maintenance,
1.2 Digital fabrication

A step beyond architectural plans, being able to automatically construct a physical model or an actual building has been a desired goal. A new field, known as Construction Robotics, aims at empowering robots to automatically build buildings or some of its parts. We refer the reader to the books from Gramazio et al. [2014], Bock [2007]. Fig. 1.2 shows a robot which is able to automatically build a brick wall of any given shape (needless to say, provided it can stand). Here, a mesh is not enough and semantics are needed to know the materials to select to build a part as well as which parts should be connected or disjoint. For example, a window should contain glass and be articulated. A semantic understanding of a set of buildings would not only enable to reproduce the main functions and appearances of these buildings, but also to automatically design new buildings in the same style.
1.3 Maps

Maps serve multiple purposes: First, they are a primary tool for navigation. Second, they are analysis tools to represent any data across a geographical area. For the first, city maps typically consist of the street network. Sometimes, they feature additional layers, such as public transportation lines, points of interest or buildings.

In addition to tourist maps, buildings are increasingly represented in navigation services such as GPS guidance systems or street view services. Traditionally, buildings had to be hand drawn. There, the focus was less on the accuracy of the drawing, but rather an abstraction indicating the discriminative features of a building.

![Figure 1.3: Top: Detail from a hand-drawn map of Hoorn, by Johannes Blaeu, 1649. Bottom: Map of London and North Eastern Railway by George Dow, 1929.](image)

Even though the result looks convincing (see Fig. 1.3), manual work does not scale up to the size of whole cities. As a result, birds-eye views, 3D mesh reconstruction and street view services have been recently developed. Fig. 9.8 shows a comparison for Muensterhof...
1.3. Maps

in Zurich. These offer the advantage of realism, but have several drawbacks for the purpose of map-making. First, they are harder to read than hand-drawn maps since they do not exhibit the discriminative features of a building. Further, they suffer from occlusions, e.g. by vegetation.

![Figure 1.4](image)

**Figure 1.4:** Google bird’s-eye view (left), map (bottom-right) and street view (top-right) of Muensterhof, Zurich.

Street view services are also excessively uncomfortable to browse, for they are restricted to a set of sparse bubbles. Second, they raise privacy issues. In particular, faces and car plates are often recognizable. In addition, unusual capture angles might allow to see private grounds. Furthermore, street view services and 3D mesh reconstructions are impractical to use on mobile devices as they require large amounts of data.

It is worth noting that, even for navigation purposes, maps do not have to be at scale. For example, public transportation maps are topological maps, which are easier to read. Figure 1.3 shows what is recognized as the first topological transit map. The inventor, George Dow, inspired Harry Beck for his famous 1933 London Tube Map.

Beyond navigation, maps are used as tools for showing and simulating data which may pertain not only to the street layout but also to the infrastructure or the buildings themselves. A whole field, referred to as GIS (Geographic Information System) deals with creating and exploiting geographical databases. The applications are broad, and range from public transport optimization to mapping crime or pandemics. Note that the use of GIS can be integrated with urban planning and replanning. Also, collecting information about buildings is a tedious work and has therefore not been a primary focus, although this would be very beneficial for a more fine-grained analysis. For example, information needed for civil protection includes building use type and construction year, and could benefit from additional information such as window and door access. The position of doors and windows can be used in particular for flood prevention and to identify possible access points.

In summary, the focus of building modelling has mostly been on realism, leaving out the semantic part and readability of maps, which are important for both navigation and GIS.
1.4 Film and video games

The aspiration of graphics designers for films and video games in matters of urban mod-elling is primarily to deliver an impression and a feel. There, the styles of the buildings prime over the perfect match to the actual buildings; the same applies to the street network. For example, in the film Ratatouille, the layout of landmarks does not match the actual layout of the city of Paris at all: the landmarks are denser than in reality, even though the overall style of the city is preserved (see Fig. 1.5). As a result, it is also important that the rendering is coherent with the one of characters: a cartoon animation would not match with a photo-realistic city. Interestingly, a higher level of realism does not always result in an increased comfort. In fact, it appears that near-realism may become uncanny as hypothesised by Mori [1970]. Therefore, in the film and video game industries, fully controlling the rendering is a requirement. Such a level of control can only be achieved when using a 3D model of a city, as opposed to editing film footage.

![Cityscapes in Ratatouille, in the style of Paris (top), Total Recall 2012, in the style of a futuristic London (bottom).](image)

Like the film industry, the video game studios are eager to produce visually pleasing models. Figure 1.6 shows state-of-the-art rendering. Note the importance of other urban elements than buildings which contribute to the realism: fog, railings etc.
It is interesting to note that in addition to showing a pre-defined city model, it is also desirable to be able to generate new buildings in real-time.

**Figure 1.6:** The skyline of Los Santos, a imaginary city in the game Grand Theft Auto V based on Los Angeles.

### 1.5 Cultural heritage

Like architects, archaeologists have used wood models to represent reconstructions of remains. In addition to documenting a site, archaeologists use models as tools to communicate with the general public. Therefore, realism and immersion play very important

**Figure 1.7:** Blended image of an original picture and the reconstruction of the temple of Hera, Paestum, Italy.
1.6 Challenges in City Modelling

We have sketched a picture of the needs related to architectural modelling. A summary makes this picture diverse and full of contrasts. We draw a non-exhaustive list of desirable qualities of a city model: compactness, accuracy, allowing for fast alterations, with adjustable degrees of realism and simplicity, semantic, abstract.

**Compactness** is important not only to reduce data consumption (in particular for data transfer on mobile devices) but also to reduce memory consumption at render time.

**Accuracy** should be as high as possible and aimed for photorealism.

**Rapid prototyping** means the model should be interactive. This can only be achieved if the model bears an abstract understanding of the underlying contents, e.g. floors.

**Adjustable Level-of-Detail (LOD)** is relevant at render time, when distant buildings can be simplified to save computational resources.

**Semantic** tells about the meaning and functions of architectural elements.

**Abstraction** not only gives an understanding, but is also key to making compact, flexible and useful models.

Finally, the noise and wide appearance spectrum in the input data are the classical challenges in computer vision.

Not only is this list partial, it ignores the main issue behind city modelling: **time**. Modelling has been essentially manual for a large number of applications. Building a model of a building takes two or three days. Building a model of a city, years. In his thesis [Müller 2010], Mueller reports that for films such as “The Day after Tomorrow”, “Spiderman 2” or “Batman begins”, which all take place in very large exiting or non-existing cities, it took several man-years of work to design the 3D models of the cities. And this is for a single application.

As an additional motivation, we think that the modelling of cities will go far beyond the automation of manual tasks. For other complex systems, automation led to giant’s leaps. For example, the Internet used to be accessible only through the use of web directories
which were manually designed. Not only did the advent of search engines supersede the manual work, it also led to an outstanding improvement for accessing information.

The tremendous potential of analysis and exploitation of data based on urban models, the complexity of the task and the immense amount of manual work calls for efficient automated methods.

In this work, we introduce new tools to simplify the urban modelling process and bring facade analysis forward for various applications. In particular, we create a full pipeline, propose methods which exploit 3D information and use a bottom-up approach where a grammar set of rules is inferred, which results in an understanding of the building suitable for different tasks.

1.7 Contributions

The main contributions of this thesis are as follows:

- we present a method to automatically rectify and separate facades in a picture and find the style of each facade.

- we propose an approach to automatically label a facade image using detectors, segmentation and weak architectural principles.

- we introduce a method for efficiently labelling Multi-view Stereo (MVS) meshes by only using a fraction of the input pictures, while maintaining good labelling performance.

- we present a pipeline using detectors and Structure-from-Motion (SfM) to instantiate a set of grammar rules so as to match a building instance. We demonstrate the method on antique temple reconstruction.

- we design a method to automatically infer a set of grammar rules from multiple facades. This helps for compression, facade retrieval and comparison of facades as well as virtual facade synthesis.

- we introduce an approach to mine facades which are perceived as important and discover the important elements within a facade for the creation of tourist maps.

- more fundamentally, we show that Shape Grammars can capture the essence of an architectural style. Further, we show that different grammatical descriptions are required for different applications.
1.8 Overview

The rest of this work is articulated as follows:

- **Chapter 2** gives an overview of the type of models which can be used to represent buildings and of their historical developments. In particular, we present Procedural Modelling and CGA shape grammars.

- **Chapter 3** introduces an approach to inverse procedural modelling which takes into account different scenarios.

- **Chapter 4** deals with the automatic recognition of architectural styles of facades in a picture. This step is of importance to kick-start the reconstruction process using appropriate detectors and/or style grammars.

- **Chapter 5** introduces a method for the labelling of facade images which solely relies on weak architectural priors as opposed to full style grammars. Semantic segmentation results are first combined with generic architectural element detectors before the weak architectural priors are enforced.

- **Chapter 6** introduces a method for the efficient labelling of Multi-view Stereo (MVS) meshes. A light-feature vector allows for a quick selection of the best view and a subset of the mesh faces to be classified. Computational power is largely reduced while maintaining classification accuracy.

- **Chapter 7** presents a method for the robust inference of the parameters of a grammar given a set of pictures of a building. Detections on multiple images are aggregated into the 3D space and the grammar parameters are automatically inferred. The method is demonstrated on Doric temples.

- **Chapter 8** introduces a novel method to infer set of rules and parameters for a set of facade labellings. The applications are numerous: compression, facade synthesis, facade comparison and retrieval.

- **Chapter 9** introduces a new method to automatically determine the importance of facades and, within a facade, which features are perceived as important. We pave the road to new methods of navigation.

We conclude this thesis with remarks about grammars, the logic of architecture and future applications.
In this chapter, we present the two main methods to acquire 3D data and the different kinds of models that can be used to represent buildings.

### 2.1 3D data acquisition

3D data can be obtained from two main sources: LiDAR (Light Detection and Ranging) and 2D images. **LiDAR** uses a laser scanner to measure the distance between objects points and the laser. Although LiDAR is very precise, it is also very slow. As a result, dense point clouds cannot be obtained at a large scale. Also, LiDAR does not capture colour information.

**Structure-from-Motion** (SfM) uses multiple 2D images to triangulate the 3D position of points in the images. SfM only works when points can be matched across several images. As a result, SfM works best when the picture coverage has enough overlap and when feature points can be extracted and matched. In practice, several dozens of pictures are needed to reconstruct an object. For urban modelling, feature point matching is challenging when surfaces are reflective (e.g. glass or metal) or when they are smooth (e.g. blank wall). We refer the reader to Chapter 7 of the book by Szeliski [2010] for a survey and more details about SfM.

Fig. 2.1 shows the different stages of an SfM/MVS (Multi-View Stereo) pipeline. In this example, the major architectural elements are captured. The whole street (about 20 buildings) took a few hours to capture and reconstruct. Fig. 2.2 shows a LiDAR reconstruction, where the fine-details are clearly visible. The accuracy is in the order of a millimeter. Please note that the LiDAR reconstruction took 5 days of scanning.

#### 2.1.1 LiDAR vs. Structure-from-Motion

We now compare the results given by LiDAR to the ones of SfM pipelines.
2.1. 3D DATA ACQUISITION

**Figure 2.1:** From top to bottom: Sparse SfM point cloud, Dense mesh, Dense textured mesh of Seilergraben, Zurich obtained with the CMPMVS pipeline [Jancosek & Pajdla 2011a].

**Figure 2.2:** LiDAR scan detail of the town hall in Klodzko, Poland, obtained with a ZF imaginier 5003 laser scanner (image courtesy of 3deling).

**Accuracy** LiDAR gives the best accuracy (typically less than a cm error), whereas SfM has been improving: in [Goesele et al. 2007], 90% of the reconstructed points were within 0.128 m of the model of a 51 meter high building.

**Completeness** LiDAR also provides complete models, where SfM is dependant on features. In practice, completeness of SfM models largely depends on the object at hand.

**Acquisition time** SfM is much more efficient than LiDAR to obtain semi-dense point clouds. While pictures can easily be captured at the framerate of a video (25 fps), a LiDAR capture takes from several minutes to a few hours.

**Processing time** LiDAR does not need processing after capture, apart from geo-registration. Conversely, SfM typically requires dozen of CPU hours for dense reconstruction of a block. A large fraction of this time is attributable to feature matching, whose time footprint is a current research topic.

**Cost** When it comes to cost, SfM has a clear advantage as a LiDAR scanner costs dozens of thousands of dollars. It should also be noted that for popular landmarks, images can be downloaded from the Internet (see Section 2.2.1), even removing the need to physically send a person for a capture.
From these points, we see that SfM and LiDAR have different applications: LiDAR is much more accurate but does not scale up as easily as SfM techniques. Therefore, LiDAR has been so far used for construction work measurements, archaeology and aerial sensing, while SfM has been used for small to large-scale reconstruction. Note that the two can be combined [Lin et al. 2011].

2.2 Airborne vs. Ground imagery

While airborne imagery captures wider areas and allows the reconstruction of rooftops, ground imagery provides a much better perspective for facade reconstruction. Therefore, the two are complementary and should eventually be combined to reconstruct a full urban model [Fruh & Zakhor 2003, Hu et al. 2006].

2.2.1 Picture acquisition

Various approaches have been developed to collect pictures.

Using a special van is the most straightforward way of capturing high-density data with a large coverage. A typical van or car used for streetview and urban modelling capture has a set of cameras covering a 360 degree view, a GPS and often features a LiDAR scanner. To make the scanning process more efficient, the scanner is usually pointing to the side and only scans a blade.

Aerial images can be obtained by using a conventional plane or satellite, as well as Unmanned Aerial Vehicle (UAV), i.e. drones. These can automatically cover a predefined area.

Finally, landmarks are densely covered by photographies found on the web. Approaches for automatic mining of images have been successful [Goesele et al. 2007, Quack et al. 2008]. On the one hand, the capture over extensive periods of time filters out any non-static element in the reconstruction. On the other hand, the method is limited to famous landmarks and the reconstructions are often incomplete due to uneven coverage. As a matter of fact, tourists tend to adopt the same set of viewpoints.

2.3 From visual realism to abstraction

While the focus had been on achieving visual realism, we see from the challenges described in Chapter 1 that more is needed from urban models. In this section, we present the spectrum of abstraction in architectural models.
2.4 A LANGUAGE FOR ARCHITECTURE

• Fig: Pictures, point cloud, mesh, primitives, semantic models, procedural models.

• different types of applications: visual realism vs. understanding and abstraction

Finally, it should be noted that the higher the level of abstraction, the higher the complexity and the number of steps required to reach the desired abstraction level. As a result, it should be noted that higher levels of abstraction are more difficult to produce using automatic systems.

In the rest of this work, we focus on automatic modelling methods which are generated from ground pictures.

2.4 A language for architecture

As developed in the introduction, urban models are required for a variety of applications, while, as we just saw, computer models are also of very diverse natures. Using only a unified model for all applications is a Holy Grail in computer science. Such a unified model should by definition simultaneously offer a high-level abstraction, suitable for analysis and compression, and the possibility to easily derive a low-level description, appropriate for graphics rendering.

In his thesis [Müller 2010], Pascal Mueller proposed to use Shape Grammars as a high-level model that could best capture the human architectural concepts.

Pascal Mueller’s work mainly focused on the design of such a model and on the derivation of low-level representations suitable for graphics rendering. In our work, we follow the inverse path: starting from low-level data, we aim at inferring high-level models.

We first give an overview of architectural theory. Then, how architecture can be described using formal grammars.

2.4.1 Architectural theory

This section presents a short overview of the field. For more details, we refer the reader to the comprehensive book [Evers & Thoenes 2003]. Architectural theory has originally been strongly linked with classical and neo-classical architecture. In De architectura [Vitruvius 1914], Vitruvius gives a comprehensive guide to Roman architecture, from designs to construction methods. The work, comprising ten books, also defines the classical orders of architecture. It was not until the Renaissance that Vitruvius’s work was rediscovered. Among others, Alberti [Alberti & Bartoli 1986], Palladio [Palladio & Ware 1965] and Brunelleschi [King 2013] founded their theory by expanding the work of Vitruvius. For
all, the importance of proportions became a founding principle. One of the most famous work by Leonardo da Vinci, the Vitruvian Man (see Fig. 2.3), is part of a comment on Vitruvius’s work. It is important to note that, while the figure focuses on the idealized proportions of a human body, Vitruvius emphasizes the correlations between human proportions and architectural designs. As a toy example, a single door should be taller than wider based on human proportions.

Figure 2.3: The Vitruvian Man, Leonardo da Vinci, ca. 1490, was inspired by [Vitruvius 1914]. The figure depicts idealized body proportions, which should in turn be taken into account in architectural designs.

Later, Semper, who was both a theoretician and a renowned architect, established architectural principles based on anthropology in [Semper et al. 1989]. In the 20th century, Le Corbusier revisited the Vitruvian Man by introducing the Modulor, which sought to expand the foundations for the design of harmonious and functional buildings based on the human body. Following Vitruvius, Le Corbusier makes use of many standards regarding sizes and proportions: the height of the ceiling is such that the average person cannot touch it when raising his arms, the width of windows is such that one can lean out of a window, the width of staircases is such that two people can pass. Furthermore, it is worth noting that the formal character of architecture spans across different cultures. For example, Li Jie wrote an architectural treaty [Jie 1103] formalizing the Chinese architecture. The book was published by the Huizong of Song in order to standardize architecture throughout the Empire.

Meanwhile, the styles evolved tremendously. Here, we do not seek to give even an overview concerning the history of styles and refer the reader to [Harbison 2009]. Nevertheless, we generally observe an alternation between structured and organic architectures.
For example, Art Deco came as a reaction to Art Nouveau. Recently, the interest for organic architecture was renewed with architects such as Calatrava and automatic design and construction. In 2.4, we give examples on the spectrum of the regularity of buildings.

![Figure 2.4: From regularity to uniqueness. From left to right: Haussmanian, Gründerzeit, Baroque buildings, Eiffel Tower, Kunsthaus Graz [from H. Riemenschneider].](image)

In 1914, Frankl & Dauer [1981] presents the fundamental principles of the history of architecture, where he places purpose, function and intention at the centre of architectural creation. The analysis of architectural theory led to formal architecture. In [Tzonis & Lefaivre 1986], Tzonis and Lefaivre formalize classical architecture and writes its rules of compositions. Formal architecture is a turning point in Architecture Theory: for the first time, architecture is described with a high level of abstraction, and by tools such as formal grammars, as proposed by Mitchell [1990].

Finally, we conclude this section with philosophical remarks. We carve out the commonality between these theories: architecture is an expression of the human mind and body constrained by the laws of physics and nature.

In practice, this means that a language to describe architecture is constrained: First, by physical and functional constraints. For example, doors need to be located on the ground floor or accessible through stairways. Second, by human values: repetition and proportions are at the core of architecture. We can only speculate about their origins in architecture and draw parallels with other arts such as painting and music, where simplicity and elegance also play a vital role.

### 2.4.2 Computational design

Independently of the formalization of architecture, computers have evolved as tools for design. Computational design refers to the field of computer science which helps for the design of objects. In 1962, Pierre Bezier invented the Bezier curve, a kind of parametric curve which was used to model car bodies [Bézier 1968], easing the design of car parts using computers. In order to describe and generate new complex shapes, shape grammars were introduced.
2.4.3 Shape grammars

The Swedish mathematician Helge von Koch constructed one of the first known fractal shape, the Koch snowflake [Von Koch 1904].

![Figure 2.5: The first three iterations of the construction of the Koch snowflake (image courtesy of Wikipedia).](image)

The Koch snowflake is also one of the first shapes which is described as a procedure. Furthermore, it became possible to describe a family of shapes.

The construction procedure is as follows:

1. the line segment is divided into three segments of equal length.
2. the middle segment obtained at step 1 is replaced by an equilateral triangle pointing outwards.
3. the base of the triangle from step 2 (corresponding to the line segment of step 1) is removed.

Independently, research on language made a breakthrough with the discovery of formal grammars [Chomsky 1956]. Formal grammars aim at representing natural languages as a re-writing system. A formal grammar is defined by the tuple \((N, \Sigma, R, S)\), where

- \(N\) is a finite set of non-terminal symbols distinct from \(\Sigma\),
- \(\Sigma\) is a finite set of terminal symbols, disjoint from \(N\),
- \(R\) is a finite set of production rules of the form: \((\Sigma \cup N)^* N (\Sigma \cup N)^* \rightarrow (\Sigma \cup N)^*
- \(S\) is a starting symbol (axiom).

For example, these sentences from the English language: “A player catches the ball”, “The man chases the taxi”

could be formed by the following production rules, where \(S\) is an arbitrary symbol:

- \(S \rightarrow <\text{nounphrase}> <\text{verbphrase}>\)
• \(<\text{nounphrase}\>\rightarrow<\text{determiner}><\text{noun}\>)

• \(<\text{verbphrase}\>\rightarrow<\text{verb}><\text{nounphrase}\>)

Chomsky established a containment hierarchy of formal grammars, known as the **Chomsky hierarchy**. Type 3 grammars are included in type 2 grammars, which are included in type 1 grammar, while type 0 grammar include all types.

**Unrestricted grammars (type 0)** include all formal grammars.

A grammar is **context-sensitive (type 1)** if all the rules in \(\mathcal{R}\) are of the form:

\[
\alpha A \beta \rightarrow \alpha \gamma \beta
\]

where \(\alpha, \beta \in (\mathcal{N} \cup \Sigma)^*, \gamma \in (\mathcal{N} \cup \Sigma)^+\) and \(A \in \mathcal{N}\). In other words, \(A\) can be replaced by \(\gamma\) depending on its context \((\alpha\) and \(\beta)\).

A grammar is **context-free (type 2)** if all the rules in \(\mathcal{R}\) are of the form:

\[
A \rightarrow \gamma
\]

where \(\gamma \in (\mathcal{N} \cup \Sigma)^+\) and \(A \in \mathcal{N}\).

A grammar is **regular (type 3)** if all the rules in \(\mathcal{R}\) are of the form:

\[
\begin{align*}
A & \rightarrow a \\
A & \rightarrow aB \\
A & \rightarrow \epsilon
\end{align*}
\]

where \(A, B \in \mathcal{N}, a \in \Sigma\) and \(\epsilon\) designates the empty string.

A **parse tree** is a tree which represents a context-free grammar, and where each node corresponds to a symbol. The root is the starting symbol \(S\), and the leaves the terminal symbols \(\Sigma\).

Shape grammars are the combinations of computational geometry and formal grammars.

Stiny was the first to introduce shape grammars [Stiny 1975], originally to generate paintings and sculptures [Stiny & Gips 1971]. The idea is to use a formal grammar [Chomsky 1956] which operates on shapes instead of words.

Shape grammars are production rule systems where shapes, which are arrangements of lines, are successively transformed into other shapes using a set of rules. A shape grammar is defined by:

• \(S\) is a finite set of shapes,
2.4. A LANGUAGE FOR ARCHITECTURE

Figure 2.6: Shape grammar example and derivations, from [Stiny 1975]

- $\mathcal{L}$ is a finite set of symbols,
- $\mathcal{R}$ is a finite set of shape rules of the form: $(\mathcal{S}, \mathcal{L})^+ \rightarrow (\mathcal{S}, \mathcal{L})^*$,
- $I$ is a labelled shape in $(\mathcal{S}, \mathcal{L})^+$ called the initial shape.

Note that the matching is done directly at the shape level, not at a symbolic level. This results in problems when matching is ambiguous, plus it requires an increasing level of computations as the shape complexity grows.

Shape grammars are nondeterministic. In other words, several rules may be applicable to the same symbol, yielding several possible derivations.

In computer graphics, procedural modelling was initially introduced for texture and cloud generation independently by Perlin [1985], Gardner [1985]. Numerous works followed, including L-systems, created by Prusinkiewicz and Lindenmayer to generate models of plants.

2.4.4 L-systems

While Lindenmayer was a biologist who was at first studying the growth patterns of plankton, L-systems proved to also be adequate for describing the growth patterns of whole plants, trees and flowers.

An L-system is a parallel rewriting system defined by:

- $\mathcal{V}$ (variables), a set of symbols which can be replaced,
- $\omega$ (axiom), a string of symbols from $\mathcal{V}$,
• $\mathcal{P}$ is a finite set of production rules.

Further, an L-system can be context-sensitive (type 1 grammar), stochastic or parametric. **Context-sensitivity** is similar to the one of formal grammars.

In a **stochastic** L-system, rules are associated with and triggered according to a given probability.

A **parametric** L-system associates a parameter list to each rule, e.g. corresponding to a set of sizes.

Note that, in an L-system, the shape is defined by a string. This solves the graphical matching problems mentioned when working with Stiny’s shape grammars. The graphics can be generated from the resulting string, by using for example *turtle graphics*. Turtle graphics use a relative cursor, referred to as a turtle, which is moved in a Cartesian plane according to a sequence of instructions. Historically, turtle graphics were first generated by a robot called “turtle”, created in the late 1940s by Grey Walter, which would draw using a pen on a sheet of paper. Later, the Logo language was created by Seymour Papert to control the turtle.

For example, the von Koch curve can be simply described in Logo Turtle language as:

- $\omega = F + +F + +F$

- a single rule: $F \rightarrow FF + +FF$

where $F$ means “draw a unit line segment forward”, - means “rotate left with an angle of $\alpha$” and + means “rotate left with an angle of $\alpha$”. For the von Koch curve, $\alpha = 60^\circ$.

The outcome of L-systems for plant modelling was fruitful: Fig.2.7 shows state-of-the-art results from Palubicki et al. [2009].

### 2.4.5 Split grammars

Shape grammars were quickly applied to the description of architectural rules: in 1978, Stiny and Mitchell [Stiny et al. 1978] wrote the “Palladian grammar”. Other work followed for various styles, including Frank Lloyd Wright’s prairie houses [Koning & Eizenberg 1981], Alvaro Siza’s patio houses [Duarte 2005] and Maya buildings from Xkipché [Müller et al. 2006a].

In 2003, Wonka combines two ideas to create *split grammars*: First, the idea that building can be represented by successivly splitting shapes, carving out a manhattan-world, lego-like volume. Second, the idea originating from Stiny to use symbols for the grammar so as to avoid the problems related to shape matching.
The description of a building using split grammars is intuitive. For example, a regular facade can be split into floors, which are in turn split into series of windows and wall pieces. Note that the terminal symbols can correspond to any arbitrary shape, and not only block-like shapes.

We briefly present two of the most used shape grammars for architecture, CGA (Computer Generated Architecture) shape grammars and GML (Generative Modelling Language).

### 2.5 CGA shape grammars

CGA (Computer Generated Architecture) shape grammars are a more versatile successor to split grammars, introduced in [Müller et al. 2006b]. Here we only give a short introduction and invite the reader to study [Müller et al. 2006b]. Formally, CGA shape grammars are set grammars [Stiny 1982]. Like split grammars, the derivations are operated on symbols to prevent from using visual shape matching.

A CGA shape grammar is defined by:

- $S$, a finite set of shapes,
- $O$, a finite set of shape operations,
- $R$, a finite set of shape production rules of the form $S^+ \rightarrow (S^* \cup O^*)+$ where $(S^* \cup O^*)+$ is a sequence of symbols or shape operations,
- $I \in S^+$, the initial shape.
2.5. CGA SHAPE GRAMMARS

2.5.1 Properties

The CGA shape grammar is context-free, stochastic, conditional and parametric.

Note that, even though the symbol matching is context-free, CGA shape grammar can apply the rule by examining the spatial context. The conditional property lets the interpreter apply rules depending on a logical expression.

2.5.2 Syntax

All rules, except for conditional or stochastic rules, are written as \textit{predecessor} \rightarrow \textit{successor}. We refer the reader to [Müller 2010] for a comprehensive overview of the syntax.

2.5.3 Shape

A shape is defined by a symbol, a set of parameters and attributes.

The main attributes of a shape are:

- the pivot, defined by an origin and orientation vector,
- the scope, corresponding to a bounding box and given by a translation, a rotation and a size,
- the geometry (if the symbol is terminal), i.e. an asset, for example a mesh or a texture.

Figure 2.8 shows the main attribute of a CGA shape.

2.5.4 Operations

Shape operations are the main innovation of CGA shape grammars over L-systems and shape grammars. In this section, we present the main CGA operations:

- \textbf{insertion} inserts a geometry and/or a texture. The inserted object is referred to as an \textit{asset}. For buildings, these correspond to architectural elements, such as a window, a door, or a column part. Note that simple geometrical shapes, such as a box, are built-in.

- \textbf{scope transformations: translation, rotation and scale}

- \textbf{subdivision split} splits the current shape into multiple shapes along an axis. In architecture, splits are omnipresent. A special case of subdivision splits is the repeat splits, which tiles a scope with a given rule.
Component split divides a shape into components of lesser dimensions, i.e. faces, lines or points. In architecture, this is for instance useful to separate a facade from the building mass model. A mass model is a rough model which gives the basic structure of a building, leaving out details.

2.5.5 Derivation tree

A sequence of rule applications can be represented by a tree, which we refer to as the derivation tree. The terms derivation tree and parse tree can be used interchangeably. Often, the term parse tree is used when the tree is inferred from data, and the term derivation tree when the tree is derived from a set of rules, although they refer to the same tree. In this tree, each rule is represented by a node whose children are its successors. Note that the rules can be derived sequentially or in parallel, which is useful for implementation.

2.5.6 CGA example

Figure 2.9: Example of a facade decomposition, from [Müller et al. 2007].

Figure 2.9 shows a simple example of the decomposition of a facade. This decomposition is very easily captured by a CGA shape grammar.
2.6. GML

Here is a set of rules describing Fig. 2.9:

- **Facade** $\rightarrow$ split(y)\{1.3: GroundFloor | {1 : Floor}*\}
  splits the facade vertically into the ground floor and a repetition of upper floors,

- **GroundFloor** $\rightarrow$ split(x)\{0.8: DoorTile | {0.8 : WindowTile}* \}
  splits the ground floor horizontally into a the door tile and a repetition of window tiles,

- **Floor** $\rightarrow$ split(x)\{0.8 : WindowTile}* 
  splits the other floors horizontally into a repetition of window tiles.

2.6 GML

GML (Generative Modelling Language) [Havemann 2005] was developed shortly after CGA. GML and CGA provide the same functionality and use a different syntax [Hohmann et al. 2010]. The main difference with CGA is that GML uses function-based descriptions. In particular, when in CGA the terminal symbols often consisted of pre-defined 3D models, GML describes the shapes using parametric functions. For example, while in CGA, a window is encouraged to be defined as a 3D asset, it is in GML described as a set of splits. As a result, GML potentially offers a higher level of procedurally modelled details. Note that, in many cases it would also be possible to model assets procedurally in CGA, but it is usually not done because of the additional work it requires.

In the rest of this thesis, we will use CGA shape grammars to describe buildings. The main advantages of CGA are its simplicity and the fact that a commercial software, CityEngine, is available for parsing the rules into 3D models. The next chapter gives an overview of our approach to inferring parameters or rule sets of CGA shape grammars from pictures.
3

Approach

In this chapter, we present a global approach to inverse procedural modelling and give some insights about the choices to be made.

3.1 Pre-processing steps

Given a point cloud and initial images of a building, its style needs to be detected and its facades extracted in order to kick-start the reconstruction process.

3.1.1 Style detection

A method for style detection is detailed in Chapter 4. First, the input image is classified as containing no building, a building part, a street or facades. If it contains facades, the image is rectified to correct perspective effects. Then, facades are separated and each facade is classified according to its style.

3.1.2 Facade extraction

A method for facade extraction is given in Chapter 7. If several images of a building are provided, a 3D point cloud is computed using Structure-from-Motion algorithms such as the ones presented in Section 2.1. The point cloud is cleaned and facades as well as the vertical direction are extracted.

3.2 Top-down vs. bottom-up

Two lines of approaches are possible when it comes to model inference: top-down and bottom-up.
The top-down approach consists in starting from a given parametric model, and inferring an instantiation which matches the building at hand. Works which use a top-down approach include [Teboul et al. 2010b; 2011a]. The bottom-up approach starts from the building data and infers both a parametric model and its parameters. This line of approach for inverse procedural modelling is one of the contributions of this work. Previous work, such as [Stava et al. 2010, Bokeloh et al. 2010b] use various cues, such as symmetries and partial symmetries to infer the model.

When specifically applied to facade images [Muller et al. 2007, Riemenschneider et al. 2012] these bottom-up approaches were restricted to inference on a single facade. As a result, some of the advantages of procedural models, such as compactness and building synthesis were not fully used.

The main virtues of a top-down approach are the possibility to enforce complex prior knowledge using little evidence and the certainty to obtain a plausible model. Top-down approaches typically use Conditional Random Fields (CRF) and Markov Random Fields (MRF) for parameter inference. There, the difficulty lies in the formulation of the MRF and was successfully tackled by [Dick et al. 2004, Čech & Šára 2009, Alegre & Dellaert 2004, Ripperda & Brenner 2006a, Teboul et al. 2010a].

However, a top-down approach is not always suitable. First, since a pre-defined, manually designed style grammar is required. Producing such grammars is a tedious work: as seen in Section 2.4.5, full papers are dedicated to the creation of one such grammar. Also, buildings very often combine elements of different styles. Further, the less regular the style, the more complex the grammar and the larger the search space. Finally, irregularities cannot be captured by a style grammar. These irregularities constitute the signature of the building, so discarding them may not hurt the quantitative measurements of accuracy but will greatly hurt the human perception of the building model.

In comparison, a bottom-up approach does not have these drawbacks, as the grammar rules are inferred from data. Nevertheless, the evidence should be strong to compensate for weaker priors: when a style grammar could help “hallucinate” whole facade parts, a bottom-up approach relies more strongly on the observations.

In a bottom-up approach, the observations constitute the terminal symbols, which will be combined into a parse tree. These terminal symbols can either be visually similar primitives or semantic architectural elements, such as window, door etc. While the first was more popular in earlier works [Muller et al. 2007, Bokeloh et al. 2010b, Xiao et al. 2009a], the latter is semantically richer and less ambiguous. Semantic labellings can be obtained by combining state-of-the-art detectors and segmentation (see Chapter 5), as well as selecting the best view and polygon to be classified in a 3D scenario (see Chapter 6).

It should be noted that the smallest grammar problem is an NP-hard problem. We invite the reader to read [Carrascosa et al. 2010] for an overview about the topic.
Table 3.1: Comparision of bottom-up and top-down approach applicability.

<table>
<thead>
<tr>
<th></th>
<th>Prior</th>
<th>Evidence</th>
<th>Suitable styles</th>
<th>Challenges</th>
</tr>
</thead>
<tbody>
<tr>
<td>Top-down approach</td>
<td>strong</td>
<td>little</td>
<td>regular</td>
<td>v. large search space</td>
</tr>
<tr>
<td>Bottom-up approach</td>
<td>little</td>
<td>strong</td>
<td>unrestricted</td>
<td>sub-tree matching</td>
</tr>
</tbody>
</table>

Figure 3.1: Inverse procedural modelling pipeline. First, images are fed into an SfM/PMVS pipeline. Then, facades are extracted. Once the style has been determined (see Chapter 4), there are two possibilities: if a style grammar is available (see Chapter 7), it can be used to parse the facade (top-down). Otherwise (see Chapters 5 and 6), the facade can be labelled using weaker priors (bottom-up). In the latter case, a set of procedural rules can then be inferred and is useful for virtual facade synthesis, facade comparison, compression (see Chapter 8).

Table 3.1 compares the requirements and suitability of top-down and bottom-up approaches depending on the problem at hand.

In this work, we explore both approaches: in Chapter 7, the reconstruction uses an predefined style grammar, while in Chapter 8, the set of rules and its parameters are automatically inferred.

Eventually, a combination of top-down and bottom-up approaches is needed to yield optimal results. The labelling inference in Chapter 5 does not rely on any grammar, but uses weak architectural priors instead.

3.3 Pipeline

A full inverse procedural modelling pipeline is given in Figure 3.1. First, an input picture is classified according to its architectural style. Then, the architectural elements of the facade, such as doors, windows and wall, are labelled. We will refer to architectural elements as assets (computer graphics terminology). If a generic grammar for that style exists, it can be used to improve labelling and instanciated. In most cases however, a
generic style grammar does not exist due to high variance and style mixtures. In that case, a facade-specific grammar is inferred. This grammar can later be used for rendering, editing, compression, and used to generate a more generic style grammar which is suitable to generate new buildings in that style. This pipeline can be extended to multiple facade buildings with techniques used in Chapter 7. We demonstrate our work both on regular facades (chapter 8) as well as on landmarks (Chapter 7).
Style recognition

Procedural modeling has proven to be a very valuable tool in the field of architecture. In the last few years, research has soared to automatically create procedural models from images. However, current algorithms for this process of inverse procedural modeling rely on the assumption that the building style is known. So far, the determination of the building style has remained a manual task. In this paper, we propose an algorithm which automates this process through classification of architectural styles from facade images. Our classifier first identifies the images containing buildings, then separates individual facades within an image and determines the building style. This information could then be used to initialize the building reconstruction process. We have trained our classifier to distinguish between several distinct architectural styles, namely Flemish Renaissance, Haussmannian and Neoclassical. Finally, we demonstrate our approach on various street-side images.

4.1 Introduction

Procedural modeling of architecture describes a building as a series of rules. Starting from a mere footprint or polyhedral approximation, finer detail is added when going from rule to rule. Procedural modeling is quite different from the traditional production of textured meshes. Procedural models are compact, semantically structured and can be easily altered and used to generate photorealistic renderings. Furthermore, they support a wide range of applications, from detailed landmark or building modeling to full-size megalopolis simulations.

Whereas meshes or point clouds can be generated through dedicated mobile mapping campaigns, more precise and visually pleasing models have so far been made manually. It takes several man-years to accurately model an existing city such as New York or Paris (e.g. 15 man years for the NY model in the King Kong movie). An alternative could come from inverse procedural modeling. This process aims to reconstruct a detailed procedural
model of a building from a set of images, or even from a single image. Buildings are modeled as an instantiation of a more generic grammar.

Considering the vast diversity of buildings and their appearances in images, the underlying optimization problem easily becomes intractable if the search space of all possible building styles had to be explored. Thus, all currently available inverse procedural modeling algorithms narrow down their search by implicitly assuming an architectural style. Muller et al. [2007] developed a method based on the identification of repetitive patterns of regular facades. Teboul et al. [2010c] optimize the parameters of an Haussmannian style description. Vanegas et al. [2010] reconstruct the building mass model using a Manhattan-World grammar. In all cases, the style grammar is considered a given. Whereas for landmarks this may be derived from Wikipedia page coming with their images [Quack et al. 2008], street-side imagery typically does not come with style information, however.

We propose a four-stage method for automatic building classification based on the architectural style. The style information can then be used to select the appropriate procedural grammar for the task of building reconstruction. In this paper, we demonstrate our approach on three distinct architectural styles: Flemish Renaissance, Haussmannian, and Neoclassical. Please note that we use a loose interpretation of these architectural terms, as our focus is on the categorization of building appearance, not actual provenance. For example, our Flemish Renaissance dataset also contains buildings from the Flemish Renaissance Revival style, which both have similar visual features. We also created a publicly available dataset of facade images spanning the three presented styles, taken from the cities of Leuven, Antwerp and Brussels, in Belgium.

4.1.1 Related work

Very little research has been carried out in the field of architectural style identification. Romer & Plumer [2010] aims at classifying buildings belonging to Wilhelminian style from a simplified 3D city model. However, their approach is based on a few coarse features (building footprint and height), with no image support.

Available image classification systems such as the one of Bosch et al. [2008] often distinguish between images whose appearances are very different. Much focus has been on distinguishing indoor from outdoor scenes [Payne & Singh 2005, Szummer & Picard 2002]. Conversely, facade pictures share many common features no matter their styles. For instance, colour or edges cannot be used as cues to classify Haussmannian against Neoclassical buildings.

To our best knowledge, we are the first to tackle the problem of image-based architectural style identification. Our system provides a systematic and comprehensive way of estimating the building style from a single street-side image, incorporating steps of scene classification, image rectification, facade splitting and style classification.
4.2 System overview

The overall goal of our work is to model cities from images, taken with cameras on a mobile mapping van. We therefore look at the broader problem of selecting images that are useful for the modeling of buildings. It is likely that a significant number will not even contain buildings, but trees or only a part of a building. Figure 4.1 gives an overview over our system. The first step in the approach is to determine if the image actually contains building facades (Section 4.3). If this condition is met, we attempt to rectify the image (Section 4.4), as the images of buildings taken from the street usually contain significant projective distortions. After the image has been rectified, we still face the problem of identifying individual buildings in the image. Urban spaces often consist of long, unbroken building blocks, but the architectural styles may vary from facade to facade. In Section 4.5) we use edge information to find individual building separators. Finally, we extract features from the individual facades, and use a Naive-Bayes Nearest-Neighbor classifier to determine the architectural style of the facade (Section 4.6). The obtained results are summarized in Section 4.7.

4.3 Scene classification

Mobile mapping images come with different content and quality. There are typically several cameras mounted on a van, with different viewing directions. Therefore, the first step in the process of building classification consists of winnowing all the collected images into a set of images that actually contain objects of interest. We want this step to be as fast as possible, due to the fact that it will have to deal with all images taken. On the other hand, the algorithm is desired to have good generalization to robustly deal with novel scenes. It has been shown by Oliva & Torralba [2006] that humans have the capability
of determine scene type in less than 200ms. This abstract representation of the scene is called *gist*, and has served as a starting point for the development of numerous algorithms for fast scene classification [Siagian & Itti 2007, Oliva & Torralba 2001a]. These holistic algorithms attempt to capture the global scene properties through various low-level image features. The suitability of different gist-based approaches for scene categorization is discussed in [Siagian & Itti 2008]. Therefore, we opt for a gist-based scene classification.

### 4.3.1 Scene classes

We want to distinguish between the four most common scene types in street-side imagery (see Figure 4.1)

- **No buildings** - images not containing any buildings. Typical examples in urban scenarios are parks, gardens and waterfronts.
- **Street** - images containing facades captured at a high angle to the facade planes, occurring when camera orientation coincides with street direction.
- **Facades** - images containing one or more whole facades.
- **Building part** - images containing only a small part of a facade, not enough for a complete reconstruction.

Among the listed scene classes, only the “facades” class enables us to attempt a complete facade reconstruction. The appearance of the “No building” class in collected images tells us there’s a gap in the building block, and that no buildings should be reconstructed. Similarly, if the image is classified as "Street", we can deduce the existence of a street crossing. Finally, the "building part" class informs us that the building is too large (or the street too narrow) to be captured in a single image.

### 4.3.2 Feature extraction and classification

In our implementation, we use a similar approach to [Torralba et al. 2003]. We use a steerable pyramid of Gabor filters, tuned to 4 scales and 8 orientations. Filter outputs are then averaged on the 4x4 grid. This produces a feature vector comprising of 512 features. Classification is performed using a Support Vector Machine (SVM) with a Gaussian radial basis kernel function. The SVM is trained using a one-versus-all approach.

The scene classification dataset contains 1616 images in total, split into 4 classes of 404 images. Apart from using our own images from Leuven and Antwerp, we extracted additional images from publicly available datasets [Shao & Van Gool 2003, Torralba 2010, ...]}
4.4. IMAGE RECTIFICATION

Teboul 2010]. The images were then resized to a common size of 256x256 pixels and sorted into appropriate classes. Training and test sets are extracted randomly from the complete dataset, by taking 354 images of each class for training and 50 for testing.

4.3.3 Results

<table>
<thead>
<tr>
<th>Buildings</th>
<th>None</th>
<th>Part</th>
<th>Street</th>
<th>Facades</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td>100</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Part</td>
<td>2.8</td>
<td>85.6</td>
<td>2.4</td>
<td>9.2</td>
</tr>
<tr>
<td>Street</td>
<td>0.8</td>
<td>1.2</td>
<td>98</td>
<td>0</td>
</tr>
<tr>
<td>Facades</td>
<td>0</td>
<td>7.2</td>
<td>0.4</td>
<td>92.4</td>
</tr>
</tbody>
</table>

Table 4.1: Confusion matrix for the scene classification algorithm. The value in i-th row and j-th column represents the percentage the i-th class was labeled as j-th class.

The process of training and validation is repeated five times with different splits in training and test sets, to eliminate possible biases in the choice of the training set. Results were then averaged, and a confusion matrix was generated (See Table 4.1).

We can see that the most distinct classes are easily separated from the others. Utilizing the vocabulary from [Oliva & Torralba 2001a], we can deduce that the images from the ’No building’ class usually have a high degree of naturalness, while the ’Street’ class, characterized by long vanishing lines, has a high degree of expansion. The misclassification mostly occurs between classes ’Building part’ and ’Facades’. This behavior is expected, because the scenes are visually quite similar.

4.4 Image rectification

Facade images are often taken in narrow streets. Sideways looking cameras have a low chance of capturing most of a facade, as opposed to cameras looking obliquely forward, upward or backward. The images taken by these cameras are projectively distorted. The prior rectification of the images to a fronto-parallel view is a prerequisite to not only our facade splitting algorithm but also further processing steps. In our implementation we followed the approach from [Liebowitz & Zisserman 1998]. After the scene classification from Section 4.3 we assume to look onto a planar surface containing two dominant perpendicular directions, which is a sensible assumption for man made scenes.

The relation between points of the image plane \( x \) and points in the world plane \( x' \) can be expressed by the projective transformation matrix \( H \) as \( x' = Hx \), where \( x \) and \( x' \) are homogeneous 3-vectors. The rectification follows a step-wise process (see Figure 4.4)
by estimating the parameters of the projective $P$, affine $A$ and similarity $S$ part of the transformation $H$, which can be (uniquely) decomposed into:

$$H = SAP$$  \hspace{1cm} (4.1)$$

The projective transformation matrix has the form

$$P = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ l_1 & l_2 & l_3 \end{bmatrix}$$  \hspace{1cm} (4.2)$$

where $l_\infty = (l_1, l_2, l_3)^T$ denotes the vanishing line of the plane. Parallel lines in the world plane intersect in the distorted image at vanishing points. All vanishing points lie on $l_\infty$. To find these vanishing points we detect lines in the image using the publicly available implementation of the state-of-the-art line detector from [Barinova et al. 2010]. Then we use RANSAC to detect the two vanishing points of the image [Fischler & Bolles 1981].

The affine transformation:

$$A = \begin{bmatrix} \frac{1}{\beta} & \frac{-\alpha}{\beta^2} & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$  \hspace{1cm} (4.3)$$

has two degrees of freedom represented by the parameters $\alpha$ and $\beta$. The knowledge of the perpendicular intersection of the dominant lines $l_a$ and $l_b$ is the only constraint we can impose, as we have no further knowledge about other angles or length ratios in the image.

\textbf{Figure 4.2}: Rectification process: (a) input image with dominant lines, (b) projective distortion removal (c) affine distortion removal (d) similarity transformation
4.5. Facade splitting

Urban environments often consist of continuous building blocks with little or no space between individual buildings. Additionally, each building in the block may have a different architectural style. Therefore, the style recognition system needs to be able to separate different facades in order to properly classify them. As man-made structures are usually characterized by strong horizontal and vertical lines, we choose to exploit them as the main cue for building separation. We assume that the individual facades can be separated using vertical lines. The following heuristic, similar to [Xiao et al. 2009b] is used: horizontal line segments on the building usually span only one facade. Vertical lines which cross a large number of horizontal lines have less chance of being a valid facade separator.

4.5.1 Line segment detection and grouping

After the rectification step, we know that the vertical lines in the image correspond to the global direction of gravity. We use a line segment detector [Grompone von Gioi et al. 2010] to find salient edges. Then, line segments are grouped in three clusters. The first cluster contains horizontal line segments (with a tolerance of +/- 10 degrees in orientation). Similarly, the second contains vertical line segments, while the third contains all other detected line segments. The last cluster will typically have a smaller number of elements, due to the predominance of two perpendicular orientations in urban scenery.
4.5.2 Vertical line sweeping

Next, we sweep a vertical line over the image. At each position of the line, we calculate two values: support and penalty.

Support is defined as the number of vertical line segments that coincide with the sweeping line (or reside in its close vicinity). Every vertical line segment is additionally weighted with its length: longer vertical line segments provide more support for the line. The support from neighboring vertical lines is reduced linearly with the distance to the line.

Penalty is calculated through the number of horizontal lines that the sweeping line crosses. Every horizontal line segment is weighted with its length: the longer the crossed segment is, the more penalty it generates. Relative position of the crossing point to the center of the horizontal line segment is also evaluated. Vertical lines that cross horizontal segments near the edges will receive less penalty than those who cut the segments through the middle.

After the line sweeping process we have two vectors of the same size, equal to the image width: support vector and penalty vector. We want to find the positions of the vertical line which correspond to local minima in the penalty vector and local maxima in the support vector. In order to calculate this, we first use the penalty vector to threshold the support vector. All of the line positions which have more than 3% of the maximum penalty value are discarded. Then, positions which have less then 20% of the maximum support value are eliminated as well. We set the appropriate values in the support vector to zero. Finally, we perform local non-maxima suppression on the support vector through the use of a sliding window (9% of the image width). The resulting local maxima then coincide with the desired separator positions. We use these values to cut the building block into individual facades. The process of estimating facade separators from line segments is illustrated in Figure 4.3.

4.5.3 Results

We tested our facade splitting algorithm on a dataset consisting of 178 facade images from Brussels. We achieved a detection rate of 77%, with 29.4% false positive rate. The cases where system failed to detect a boundary between facades were generally buildings which had strong horizontal features on the splitting lines. Highly protruding eaves from the neighboring roofs and shops with awnings which span multiple facades are typical examples. False positives generally appear on background buildings and non-planar facades.
4.6 Style classification

The style classification is an important step in order to select an appropriate grammar for the given building. To differentiate between the different styles, namely "Flemish renaissance", "Haussmann", "Neoclassical" and "Unknown", we got convincing results using the Naive-Bayes Nearest-Neighbor (NBNN) classifier proposed by [Boiman et al. 2008]. Despite its simplicity, it has many advantages. This non-parametric classifier does not need time consuming offline learning and it can handle a huge amount of classes by design. This means that new styles can easily be added. Furthermore it avoids over-fitting, which is a serious issue for learning-based approaches.

**Figure 4.3:** Facade splitting algorithm.
4.6. STYLE CLASSIFICATION

4.6.1 NBNN algorithm

The algorithm is summarized as follows [Boiman et al. 2008]:

\begin{center}
\begin{tabular}{|l|}
\hline
**NBNN Algorithm:** \\
1. Compute descriptors $d_1, \ldots, d_n$ of the query image $Q$. \\
2. $\forall d_i \forall C$ compute NN of $d_i$ in $C$: $NN_C(d_i)$. \\
3. $\hat{C} = \arg\min_C \sum_{i=1}^n ||d_i - NN_C(d_i)||^2$. \\
\hline
\end{tabular}
\end{center}

First we calculate all descriptors for our training images and sort them into the different classes. Then, for every descriptor $d_i$ of the query image the nearest neighbor distances to each class is approximated using the FLANN library [Muja & Lowe 2009]. The sum over the Euclidean distances of each query descriptor $d_i$ denotes the image-to-class distance. The class with the least distance is chosen as the winner class $\hat{C}$.

4.6.2 Results

We cross-validated our style detector using SIFT [Lowe 2004b] and SSIM [Shechtman & Irani 2007a] feature descriptors. Our dataset contains 949 images: 318 background facades (i.e. facades belonging to none of the trained styles), 286 images for Neoclassical, 180 for Haussmann and 165 for Flemish Renaissance. We have taken these images ourselves, except for the Haussmannian style images that come from [Teboul 2010]. Table 4.2 shows the confusion matrix after cross-validation for the SIFT descriptor which was performing best throughout our experiments. While the Haussmannian style is clearly separated from other classes, many buildings of the Renaissance type are classified as "Unknown". While we have the least number of images for the Renaissance style, our definition for the class is very loose, resulting in a great diversity of the facades of that class. The mean detection rate of the SIFT features was 84% while for the self similarity descriptor (SSIM) it reached only 78%. The Figure 4.4 shows the regions of the sift interest points colored in different colors. The colors indicate to which style the given feature

<table>
<thead>
<tr>
<th>Style</th>
<th>Haussman</th>
<th>Neoclassical</th>
<th>Renaissance</th>
<th>Unknown</th>
</tr>
</thead>
<tbody>
<tr>
<td>Haussman</td>
<td>0.98</td>
<td>0</td>
<td>0</td>
<td>0.02</td>
</tr>
<tr>
<td>Neoclassical</td>
<td>0.02</td>
<td>0.76</td>
<td>0</td>
<td>0.22</td>
</tr>
<tr>
<td>Renaissance</td>
<td>0</td>
<td>0</td>
<td>0.59</td>
<td>0.41</td>
</tr>
<tr>
<td>Unknown</td>
<td>0.03</td>
<td>0.005</td>
<td>0.005</td>
<td>0.96</td>
</tr>
</tbody>
</table>

\textbf{Table 4.2:} Confusion matrix for the style classification algorithm. The value in i-th row and j-th column represents the percentage the i-th class was labeled as j-th class.
had the minimum distance. The colors associated with the styles clearly dominate the images. The features that respond accordingly to the style are mostly attached to architectural elements that are typical for that style, e.g. the features responding to the capitals in a neoclassical building.

![Figure 4.4: Style detection: a) Neoclassical style (features in red), b) Haussmannian style (features in blue), c) Renaissance style (features in purple) and d) Unknown style (features in green)](image)

### 4.7 Conclusion and future work

We presented a system for automatic architectural style recognition. The output from our system can directly be used to initialize an inverse procedural modeling reconstruction.

In case the system doesn’t recognize the style, the inverse procedural modeling system can try several possibilities or use default values. However, it comes with the cost of a more complicated optimization problem.

Furthermore, knowing the style of a building implies we know the kind of elements to look for during the reconstruction and their typical appearances. Moreover, it will be able to select accordingly the corresponding typical 3D models (or even textures) of the architectural elements to be used for rendering.

In our future work, we will use feedback from the procedural modeling system to perform online learning. For example, if the modeling is successful, the style database can be updated with the current building. If not, we can select another style and retry the reconstruction.
Facade segmentation

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

5.1 Introduction

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

The state-of-the-art methods for automated facade parsing assume that an appropriate shape grammar is available from the outset [Teboul et al. 2010b]. This assumes that one has strong prior knowledge about the structure of the facade, e.g. that it follows the Haussmann style and therefore a grammar restricted to that style can be invoked. Here, we make no such assumptions, yet get better results. The proposed approach can deal with a wide gamut of styles. Yet, as more restrictive knowledge should be used when available, we show that our method outperforms the state-of-the-art by a still larger margin in the presence of style information. Moreover, we demonstrate how procedural rules and thus
shape grammars can be derived based on the segmentation, rather than vice-versa. This is an important step forward, avoiding the need for the prior, manual construction of style-specific grammars.

Our proposed facade parsing method consists of three distinct layers. In the first layer, a supervised training method learns the labeling of facade elements based on an initial oversegmentation. For this purpose we utilize the recently developed Recursive Neural Networks (RNN) [Socher et al. 2011]. In the middle layer we introduce the knowledge about distinct facade elements, such as doors and windows. The raw RNN output is then combined with the information from object detectors trained to detect architectural elements (see Fig. 9.2). We pose the merging of RNN and detector output as a pixel labeling problem, modeled as a 2D Markov Random Field over the pixels. The multi-label MRF is then solved using graph cuts. Finally, the top layer introduces architectural concepts which encourage the facade reconstruction to assume a more structured configuration. This knowledge is encoded as a set of rules directly observable in the images, in contrast to shape grammar approaches, where some concepts (e.g. vertical alignment) are implicit or hidden in the grammar derivation. Our approach also enables modeling of irregular facades, as we use the architectural concepts as guidelines, not as hard constraints.
Our main contributions are as follows: (1) a novel three-layer approach for facade parsing, utilizing low-level information coming from the semantic segmentation, middle-level detector information about objects in the facade, and top-level architectural knowledge; (2) a rigorous evaluation on two different datasets which shows that we outperform the state-of-the-art in facade parsing by a significant margin; (3) the concept of weak architectural rules, which introduce the high-level knowledge needed for making the final reconstruction architecturally plausible; (4) updated annotations for the facade dataset of [Teboul 2010], which are closer to the ground truth.

5.2 Related Work

There already exists a significant body of work on facade extraction and parsing from ground imagery. Zhao et al. [2010] presented an algorithm that parses such imagery into buildings, grass and sky, followed by the partitioning of buildings into individual facades. This facade splitting problem was also studied by Wendel et al. [2010], Recky et al. [2011], where repetitive patterns provide cues to find the correct boundary between facades. Another approach was presented in [Mathias et al. 2011a], where a scene classification step identifies input images that contain facades. After automated image rectification, buildings are split into individual facades. We demonstrate on the eTRIMS [Korc & Forstner 2009] dataset that our approach can extract facades even in cluttered scenes. However our focus is the semantic segmentation of already isolated and rectified facades.

Xiao et al. [2008a; 2009c] target realistic visualization without much semantic encoding in the reconstruction. Other authors attempt to infer structure starting from a set of basic elements, such as rectangles [Korah & Rasmussen 2008] or windows [Mayer & Reznik 2006]. These approaches, however, also depend on the strong assumption of element repetition. Probabilistic approaches to building reconstruction started with the work of Dick et al. [2004], where a building is assumed to be a 'lego' set of parameterized primitives. An expert is needed to set the model parameters and prior probabilities for full Bayesian inference.

For facade parsing, many approaches employ a procedural grammar, explicitly or implicitly. Muller et al. [2007] detect symmetries and repetitions using Mutual Information to generate an instance of a procedural model. The approach was extended in [Van Gool et al. 2007], where images with strong perspective distortions are used to infer vanishing points and 3D information from a single image. Although these approaches produce good results in facade parsing, they assume strong priors on the input facades, i.e. that they consist of a rather regular window grid. Other grammar-based approaches include stochastic grammars [Alegre & Dellaert 2004], rjMCMC for the construction of a grammar tree [Ripperda & Brenner 2006b], and hybrid bottom-up/top-down approaches [Han & Zhu 2009]. Good results were reported by [Teboul et al. 2010b], where facade reconstruction was postulated as a problem of finding the correct parameters of a pre-specified
shape grammar. A random-walk algorithm was used to find the optimal values of the parameters. Recently, the approach was enhanced [Teboul et al. 2011a] with a novel optimization scheme, based on Reinforcement Learning. In this paper, we evaluate our system on the dataset from [Teboul et al. 2010b; 2011a], for which we also provide a more precise set of annotations.

The benefit of relying on shape grammars is that they strongly restrict the search space during parsing. Yet, the grammar may not be expressive enough to cover the variance in real world data. Furthermore, an expert is needed to write the grammars for the relevant styles. Human intervention is also required to pre-select the grammar appropriate for each specific building. The latter requirement can be mitigated by applying style classifiers [Mathias et al. 2011a] that automatically recognize the building style from low-level image features. Still, using a style grammar would imply it needs to be available beforehand, which at least for the moment is a limiting issue. It is easier to generate style classifiers than style grammars. Therefore, we chose not to assume there is such predefined grammar. In fact, our guiding principle is to derive procedural grammars based on automatically parsed facades, rather than vice-versa. Some interactive work in that vein has already appeared. Aliaga et al. [2007a] infer simple grammatical rules from a user-given subdivision of a building. Bokeloh et al. [2010b] presented a framework applied on synthetic 3D data.

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

5.3 Datasets description

We evaluate our approach on two datasets, the “Ecole Centrale Paris Facades Database“ [Teboul 2010] and the eTRIMS database [Korc & Forstner 2009]. Since we are primarily interested in accurate modeling of building facades, we focus more on the ECP dataset, as it provides labels for multiple facade elements. To validate our approach, we show that we also outperform the state-of-the-art results reported on the eTRIMS dataset.

ECP Database

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

**eTRIMS Database**

contains 60 images, along with accurate pixel-wise annotations. In contrast to the ECP dataset, images are not rectified and the facades are not cropped. In order to compare our approach with the reported result from Yang & Förstner [2011] we automatically rectify the input images using the algorithm of Liebowitz & Zisserman [1998]. Furthermore, the labels of this dataset are quite different compared to the former dataset: building, car, door, pavement, road, sky, vegetation and window. For evaluation, we perform a 5-fold cross-validation as in [Yang & Förstner 2011] with random subsampling of 40 images for training and 20 for testing. Note that we do not create a validation set due to the limited number of images in the dataset.

### 5.4 Bottom Layer: Recursive Neural Network for Semantic Segmentation

**Principles of RNN.**

The Recursive Neural Network [Socher et al. 2011] is a parsing algorithm designed to capture the recursive structure commonly found in natural and man-made scenes. Starting from an initial oversegmentation of an image, we can use the network to create a binary parse tree of the whole image. This bottom-up approach is performed by recursively combining segments into supersegments until all the segments have been merged. See Fig. 5.2 for an illustration.

The starting point of our bottom layer is an oversegmentation of the image into small regions which represent objects or object parts. Afterwards, we extract features from these regions, and present them to the input layer of the RNN. This layer serves the purpose of transforming the input feature space to a semantic space of given dimensionality. The representation is computed by:
5.4. BOTTOM LAYER: RECURSIVE NEURAL NETWORK FOR SEMANTIC SEGMENTATION

Figure 5.2: Basic RNN structure. Two input segments are transformed into a semantic space and merged into a supersegment. The supersegment’s semantic vector can be recursively combined with other semantic vectors by repeating the same network structure $W$, $W^{score}$ and $W^{label}$.

$$s_i = f(W^{sem} F_i + b^{sem})$$  \hspace{1cm} (5.1)

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

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

$$s_{(i,j)} = f(W[s_i; s_j] + b)$$ \hspace{1cm} (5.2)

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

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

$$score_{(i,j)} = W^{score} s_{(i,j)}$$  \hspace{1cm} (5.3)
where \( \text{score}(i,j) \) is the score of the supersegment and \( W^{\text{score}} \) the scoring layer of RNN.

In addition to computing the scores, we compute the class label of each segment. More precisely, a softmax layer is added on top of the semantic representation, which produces a probability distribution of the labels for the given segment. The multinomial distribution for every pixel \( x \) in a segment \( s_i \) is calculated by:

\[
p(l | \text{RNN}(x)) = \text{label}_i = \text{softmax}(W^{\text{label}}s_i), \quad x \in s_i
\]  

where \( W^{\text{label}} \) represents the class prediction layer of the network, and \( \text{label}_i \) is a vector of size \( L \), the total number of label classes in the image.

**Implementation Details.**

We set the length of vectors in the semantic space to 50. We do not observe any significant improvement in the results if we use larger vectors, while the training time becomes much longer. However, using shorter vectors leads to a noticeable drop in performance. For the preparation of the data, we follow the approach of [Socher et al. 2011], with some modifications. First, the input image is oversegmented into regions using the mean-shift segmentation algorithm of [Comaniciu & Meer 2002a]. We prefer to have a more fine-grained segmentation, so as not to combine different facade elements in a single region. On average, we obtain 643 regions per image (average image size is 600*400 pixels). Next, the appearance (color and texture), geometry, and location features are extracted for each region using the procedure of [Gould et al. 2009a]. We use the default parameters from the implementation in STAIR Vision Library [Gould et al. 2009b], which results in feature vectors of size 225, and these are used as the input of the RNN.

**Training.**

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

**Interpreting the Results.**

After the training has finished, a query image is presented to the network. The trained RNN builds a parse tree for this image, assigning a score to each merger of the segments and a multinomial label distribution to each segment. We can then simply read out the probabilities in the leaves of the tree, and label the superpixels by assigning the most probable label to each image region. However, we also keep the label distributions \( p(l | \text{RNN}(x)) \) and propagate them to the higher levels of our pipeline, as they contain more information than the maximum-likelihood estimate.
5.5 Middle Layer: Introducing Objects Through Detectors

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

Window and Door Detection.

For the detection of windows and doors we use our own GPU-based implementation of the Dollar’s Integral Channel Features detector [Dollar et al. 2009]. This detector provides state-of-the-art results for pedestrian detection and proves to be equally suited for the task of window and door detection (see Fig. 5.3). Following the settings in the original paper we use depth-2 decision trees boosted by discrete AdaBoost. For feature evaluation we use 6 gradient orientation channels, 1 gradient magnitude channel and the 3 LSV color channels. The training is performed on rectified images.

Detector Output.

For every pixel \( x_i \) of a test image, we wish to estimate the probability distribution of the labels \( l_j \) given a detection \( D_k(x_i) \) with the detection score \( r \) covering that pixel: \( p(l_j | D_k(x_i)) \), where \( k = 1 \) denotes the window detector, and \( k = 2 \) denotes the door detector. A detection score of 0 denotes that the pixel is not covered by a detection. Basically, pixels inside a high-scoring detection should receive a label distribution more peaked towards the window class, while all the pixels not covered by a detection should have a uniform label distribution. We use the validation set to empirically estimate this distribution. We obtain \( t \) distributions over the labels \( l_j \) for each possible detection score in our validation set: \( p_t(l_j) \). Fig. 5.3 illustrates an example validation set detection statistics.

For a new detection in a test image, we select the distribution from the statistics with the score closest to the one of the new detection. The selected distribution is assigned to all of the pixels within the detection window.

Incorporating Detector Knowledge With Markov Random Fields.

To merge the information coming from the lower level of the pipeline and the middle level knowledge introduced by the detectors, we formulate a labeling problem by placing a 2D Markov Random Field over the image pixels. We seek to minimize the total energy,
5.5. Middle Layer: Introducing Objects Through Detectors

Figure 5.3: Left: Precision-recall curve of our window detector and an example image with detector output. Right: Statistics calculated for the window detections on a validation set of the ECP dataset. Red color indicates window class, green wall, blue balcony. Other labels have negligible influence. As we increase the number of selected window detections (threshold), we introduce more false positives. This reflects in the reduction of window class probability.

defined as the sum of unary potentials for each node, and the sum of all pairwise potentials between neighboring pixels:

\[ E(l) = \sum_{x_i} \phi_s(l_i | x_i) + \lambda \sum_{x_i \sim x_i} \sum_{x_j} \phi_p(l_i, l_j | x_i, x_j) \]

(5.5)

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

\[ \phi_p(l_i, l_j | x_i, x_j) = \begin{cases} 0, & \text{if } l_i = l_j \\ 1, & \text{otherwise} \end{cases} \]

(5.6)

The unary potential of a pixel is a weighted sum of the low-level information (RNN labeling) and detector potentials. The weighting parameters \( \alpha_k \) are estimated on the validation set, along with the smoothing parameter \( \lambda \).

\[ \phi_s(l_i | x_i) = -\log p(l_i | RNN(x_i)) - \sum_k \alpha_k \log p(l_i | D_k(x_i)) \]

(5.7)

We solve this labeling problem using graph cuts [Boykov et al. 2001a] and obtain a solution depicted in Fig. 9.2. The output after the second level is again a probability distribution over labels \( P_l \).
5.6 Top Layer: Weak Architectural Principles

Table 5.1: Weak architectural principles used to complement the segmentation results of the first 2 layers. A "x" in the "Alter" column denotes that the principle adjusts element borders. The principle may also remove or add new elements. Last two columns indicate which principles are used for each of the datasets.

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

Parameters.

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

There were only 3 parameters to be selected for the merging of the bottom layer probability maps and detector outputs: the smoothing factor $\lambda$ and the weights for the two detectors $\alpha_{\text{win}}$ and $\alpha_{\text{door}}$. We estimated these parameters on a validation set for the ECP dataset. We set them to $\lambda = 6.5$, $\alpha_{\text{win}} = 0.75$ and $\alpha_{\text{door}} = 7$.

5.6 Top Layer: Weak Architectural Principles

In the first two layers we have not used any information about the facade structure. This reflects in results which, although convincing quantitatively, suffer from errors such as missing or misplaced facade elements, which makes it difficult to derive convincing models. To combat this problem, we introduce the concept of weak architectural principles, summarized in Table 5.1.

The principles listed above are used to encode high-level architectural knowledge, and they can be directly evaluated in facade images. Some of them can be applied on a vast amount of different facades, while others may apply specifically for a certain architectural
5.6. TOP LAYER: WEAK ARCHITECTURAL PRINCIPLES

style. A relative weight is assigned to every principle, so that we can decide on which principles to use given the facade style. Furthermore, we can use the ground-truth labeling of the validation set to automatically deduce which principles should hold.

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

The (non-)alignment principle is based on the observation that many facade elements are either exactly aligned or clearly off-center (see Fig. 5.5 left). We formulate this principle as an energy optimization problem where we estimate a locally optimal solution using the BFGS Quasi-Newton method. The energy for an object class is defined as:

$$E = \sum_{r_1, r_2 \in R} \left( \omega_t \rho_t (x^{(r_1)}_1 - x^{(r_2)}_1) + \omega_t \rho_t (x^{(r_1)}_2 - x^{(r_2)}_2) + \omega_r \rho_r (y^{(r_1)}_1 - y^{(r_2)}_1) + \omega_r \rho_r (y^{(r_1)}_2 - y^{(r_2)}_2) \right)$$

(5.8)

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

(5.10)

For each pair of rectangles the bounded influence function (5.10) rates their weighted top, bottom, left and right ($\omega_t$, bottom $\omega_b$, left $\omega_l$ and right $\omega_r$) alignment. The function has a constant value as soon as the distance between boundaries exceeds $\tau_i$.

The similarity principle is applied similar to [Mathias et al. 2011c]. Every detected element votes for similar elements using an ISM-like voting scheme. Self similarity features [Shechtman & Irani 2007b] are calculated at Harris corner points. For all features inside the window bounding boxes, a vote vector to the center of the box is cast from the positions of the $n$ nearest neighbors into a global voting space. The maxima of that voting space belong to both the initial detections and new detection hypotheses.

Harris corners are also used as a simple measure for vertical symmetry. The interest points are mirrored about a symmetry line hypothesis. A match is an interest point that has a mirrored counterpart. The maximum of the matches divided by the points under consideration defines the symmetry line and the symmetry score. If symmetry is detected (symmetry score $> \tau_{sym}$), symmetric elements are mirrored, overlapping ones are removed and for the remaining ones we have 3 possibilities: add a new mirrored element, remove or keep the existing one. The decision between adding or removing an element is based on the confidence score of each element, computed from $P_l$. If an element has a confidence higher than $\tau_{keep}$, we simply keep it. We employ the same procedure for the 3 possibilities when we observe only one of two co-occurring elements.
If the principle of **Equal width or height** of elements along a row or column holds in the validation set, this property will also be enforced in the testing set. If there has not been a door detection in the image, a **Door Hypothesis** is generated based on gradients in the probability map, average probability and relative estimated sizes to other facade elements.

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

### 5.6.1 Parameters

In Equations 5.8 and 5.9 the parameter $\tau_w$ is set to half the median of the objects’ width, and $\tau_h$ to half the median of the objects’ height. With these settings, completely misaligned windows are not shifted by the minimization. The threshold $\tau_{sym}$, for considering a facade as being symmetric, is set conservatively to the highest score of an actually non-symmetric facade in the validation set. The number $n$ of the nearest neighbors for similarity voting is set to 10. $\tau_{keep}$ is also estimated from the validation set.

### 5.7 Results

We compare our results with the state-of-the-art algorithms reported on the two evaluated datasets. Table 5.2 shows the results that we obtain on each layer of the pipeline. Due to limited space, we report only the class accuracies, while the full confusion matrices may be found in the supplementary material. Figure 5.4 shows several examples of the final output of our system.

**ECP Database.**

On the left side of Table 5.2, our results are compared with the approach of Teboul et al. [2011a]. The baseline is run as described in [Teboul 2011]. We perform the 5-fold cross validation on the same dataset.

<table>
<thead>
<tr>
<th>Class</th>
<th>Baseline [Teboul 2011]</th>
<th>Layer 1</th>
<th>Layer 2</th>
<th>Layer 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>window</td>
<td>62</td>
<td>62</td>
<td>69</td>
<td>75</td>
</tr>
<tr>
<td>wall</td>
<td>82</td>
<td>91</td>
<td>93</td>
<td>88</td>
</tr>
<tr>
<td>balcony</td>
<td>58</td>
<td>74</td>
<td>71</td>
<td>70</td>
</tr>
<tr>
<td>door</td>
<td>47</td>
<td>43</td>
<td>60</td>
<td>67</td>
</tr>
<tr>
<td>roof</td>
<td>66</td>
<td>70</td>
<td>73</td>
<td>74</td>
</tr>
<tr>
<td>sky</td>
<td>95</td>
<td>91</td>
<td>91</td>
<td>97</td>
</tr>
<tr>
<td>shop</td>
<td>88</td>
<td>79</td>
<td>86</td>
<td>93</td>
</tr>
<tr>
<td>Pixel acc</td>
<td>74.71</td>
<td>82.63</td>
<td>85.06</td>
<td>84.17</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Class</th>
<th>Baseline [Yang &amp; Förstner 2011]</th>
<th>Layer 1</th>
<th>Layer 2</th>
<th>Layer 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>building</td>
<td>71</td>
<td>89</td>
<td>90</td>
<td>86</td>
</tr>
<tr>
<td>car</td>
<td>35</td>
<td>67</td>
<td>66</td>
<td>67</td>
</tr>
<tr>
<td>door</td>
<td>16</td>
<td>24</td>
<td>20</td>
<td>18</td>
</tr>
<tr>
<td>pavement</td>
<td>22</td>
<td>35</td>
<td>35</td>
<td>35</td>
</tr>
<tr>
<td>road</td>
<td>35</td>
<td>47</td>
<td>47</td>
<td>47</td>
</tr>
<tr>
<td>sky</td>
<td>78</td>
<td>91</td>
<td>90</td>
<td>91</td>
</tr>
<tr>
<td>vegetation</td>
<td>66</td>
<td>82</td>
<td>83</td>
<td>81</td>
</tr>
<tr>
<td>window</td>
<td>75</td>
<td>72</td>
<td>75</td>
<td>80</td>
</tr>
<tr>
<td>Pixel acc</td>
<td>65.8</td>
<td>81.11</td>
<td>81.94</td>
<td>80.81</td>
</tr>
</tbody>
</table>
5.7. Results

We see a clear performance boost already at the first layer. This improvement is in part due to better bottom-up features that we use, compared to the pixel level classifier used in [Teboul 2011]. Another improvement comes from the classifier itself: if we replace the RNN region classifier with an SVM, the results drop sharply by 13%. The middle layer increases the accuracies of window and door classes, as expected due to the introduction of object detectors. The drop in balcony class comes from the fact that they are often transparent, and the window detector finds the whole window structure partially occluded by the balcony. Furthermore, the usage of the smoothness term in the MRF slightly improves some of the other classes. By introducing high-level knowledge through the top layer, we obtain even better results in almost all of the classes (Table 5.2 left). However, when parsing the image on the highest level, pixel accuracy becomes an unreliable measure of performance, as slight displacements of elements might produce high error rates. Therefore, a qualitative evaluation would be more appropriate.

eTRIMS Database.

The results we show for this dataset are obtained without using the door detector in the middle layer, as there was insufficient training data. After the 5-fold cross-validation, we obtain the results on the right of Table 5.2. Example output of the top layer may be seen in Fig. 5.5. It is clear that the bottom layer of our approach already outperforms the baseline method [Yang & Förstner 2011], by a margin of 15%. The only underperforming class is window. By introducing the window detector in the middle layer we increase the performance of the window class, and slightly decrease the accuracy of doors, due to the lack of a door detector and the fact that windows are often detected on door regions. The results of the top layer require some discussion. First, it is important to note that the buildings in this dataset come from different styles and have a huge variety of appearances. They also contain a significant amount of clutter and occlusions. Also, the only existing facade elements in this dataset are windows and doors. As we introduce additional windows through our architectural principles, it happens that we reconstruct a window occluded by vegetation, which explains the small drop in detection of vegetation class. Similarly, in some cases windows may be introduced or enlarged in the wall area, which accounts for the small decrease in building and door class accuracy, however still above the state-of-the-art.

Application: Image-based Procedural Modeling.

The output of the top layer is used in a straightforward procedural modeling scenario. We define the elements of the facade to be the terminal symbols of a procedural grammar: window, balcony and door. As our top-layer segmentation output has a rather strong structure with straight boundaries, it is a simple task to subdivide the facade in a recursive way, following the borders between elements in the output. The subdivision is stopped
at the terminal symbols. This subdivision is encoded as a set of splitting CGA rules in CityEngine[Procedural 2010]. A rendered model can be seen to the far right in Fig.9.2.

![Figure 5.4: Examples of top-layer output on various buildings from the ECP dataset.](image)

![Figure 5.5: Examples of top-layer output on various buildings from the eTrims dataset.](image)

### 5.8 Conclusion and future work

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

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

There is an increasing interest in semantically annotated 3D models, e.g. of cities. The typical approaches start with the semantic labelling of all the images used for the 3D model. Such labelling tends to be very time consuming though. The inherent redundancy among the overlapping images calls for more efficient solutions. This paper proposes an alternative approach that exploits the geometry of a 3D mesh model obtained from multi-view reconstruction. Instead of clustering similar views, we predict the best view before the actual labelling. For this we find the single image part that bests supports the correct semantic labelling of each face of the underlying 3D mesh. Moreover, our single-image approach may surprise because it tends to increase the accuracy of the model labelling when compared to approaches that fuse the labels from multiple images. As a matter of fact, we even go a step further, and only explicitly label a subset of faces (e.g. 10%), to subsequently fill in the labels of the remaining faces. This leads to a further reduction of computation time, again combined with a gain in accuracy. Compared to a process that starts from the semantic labelling of the images, our method to semantically label 3D models yields accelerations of about 2 orders of magnitude. We tested our multi-view semantic labelling on a variety of street scenes.

6.1 Introduction

Multi-view 3D reconstructions are common these days. Not only have tourist data become ubiquitous [Gammeter et al. 2010, Li et al. 2008] but the images also often result from deliberate mobile mapping campaigns [Ladicky et al. 2012, Sengupta et al. 2012, Browstow et al. 2008, Geiger et al. 2012]. The images have to exhibit sufficient redundancy – overlap – in order to be suited for Structure-from-Motion (SfM) and Multi-View Stereo (MVS) reconstruction. In the meantime, solutions have been worked out to keep the number of images within bounds, primarily for making the reconstruction pipelines
Figure 6.1: View overlap is ignored by existing work in semantic scene labelling, and features in all views for all surface parts are extracted redundantly and expensively (top left). In turn, we propose a fine-grained view selection (top right), as well as to reduce scene coverage (bottom left) by only classifying regions essential in terms of classification accuracy. The labels of the classified regions are then spread into all regions (bottom right). This sparsity increases efficiency by orders of magnitude, while also increasing the accuracy of the final result (bottom right vs. top left).

applicable to larger scenes. For instance, the redundancy can be captured by measuring visual similarity between images, and the scene can be summarized, e.g. by constructing a graph of iconic views [Li et al. 2008].

In the aftermath of SfM/MVS reconstruction processes arise recent efforts to make these 3D models widely applicable. An important step in that direction is to augment the models with semantic labels, i.e. to identify parts of the 3D data to belong to certain object classes (e.g. building, tree, car, etc), or object part classes (e.g. door, window, wheel, etc). Typically, the semantic labelling is carried out in all the overlapping images used for 3D reconstruction [Ladicky et al. 2010, Tighe & Lazebnik 2012]. This implies that many parts of the scene get labeled multiple times, resulting in a large computational overhead in the order of the redundancy of the image set. The runtime of semantic classification pipelines still lies between 10 s and 300 s per image [Tighe & Lazebnik 2012]. Worse, these speeds are reported for moderately sized images of $320 \times 240$ pixels, and not for the high-resolution megapixel-sized images common for SfM. The bottleneck of redundant labelling is not in the classification step [Koehler & Reid 2013, Sengupta et al. 2013, Roig et al. 2013], but rather in feature extraction and description. Also, an extra step is needed after labelling the images, namely, to fuse the different labels of the same 3D patch in order to obtain a consistently labelled model.

We propose an alternative strategy to semantically label the 3D model. We start by producing the mesh model and then determine for each of its faces which single image is
best suited to well capture the true semantic assignment of the face. Not only do we avoid to needlessly process a multitude of images for the same mesh face, but we also have the advantage that we can exploit both geometry (3D model) and appearance (image). Moreover, the accuracy of the semantic labelling will be shown to improve over that of multi-view labelling.

A somewhat similar problem is known from texture mapping or image-based rendering. There decisions have to be made about which image to use to render the local appearance of the model. As to avoid the texture getting blurred, it is also quite usual to look for the best source image among a set of possibilities. Most methods use criteria that are related to the size of the model patch in the image and the degree to which the view is orthogonal to the patch. One may expect to find the same criteria to dominate the choice in segmentation as well, but that intuition is misleading for our application, as we will also show.

On top of selecting a single view to get each face’s label from, we speed the process up further by not providing explicit classification for all the faces. We will demonstrate that it suffices to do this for about 30% of the faces, whereas all remaining labels can be inferred from those that were extracted. Moreover, this second parsimony again increases the accuracy of labelling.

We demonstrate our semantic labelling approach for different street scenes. Yet the core of our method is general and can be applied to different types of scenes and objects. In keeping with the central goals of the paper, we achieve a speedup with about two orders of magnitude while improving the label accuracy. In summary our contributions are the following.

1. An alternative approach is proposed for multi-view semantic labelling, efficiently combining the geometry of the 3D model and the appearance of a single, appropriately chosen view - denoted as reducing view redundancy.

2. We show the beneficial effect of reducing the initial labelling to a well-chosen subset of discriminative surface parts, and then using these labels to infer the labels of the remaining surface. This is denoted as scene coverage.

3. As a result, we accelerate the labelling by two orders of magnitude and make a finer-grained labelling of large models (e.g. of cities) practically feasible.

4. Finally, we provide a new 3D dataset of densely labelled images.

6.2 Related Work

The research in the field of semantic segmentation has enjoyed much attention and success in the last years (+17% in 5 years on PASCAL [Everingham et al. 2012]). Yet most
semantic segmentation approaches still rely on redundant independent 2D analysis. Only recently some dived into the 3D realm and exploit joining the domains.


None of the above focus on the scalability issue of large scenes and only operate on individual images. Pure 2D scalable semantic classification was addressed in [Tighe & Lazebnik 2012], which reduces by nearest neighbor searching for images and superpixels.

**For the 2D domain in streetside**, where surfaces are more structured than in arbitrary scenes, fewer works have been carried out. Berg et al. [2007] pioneered the feel for architectural scene segmentation. Xiao & Quan [2009] carried out 2D classification with a generic image height prior. Riemenschneider et al. [2012], Martinović et al. [2012] both used streetside object detectors on top of local features to improve the classification performance. Yet classification is performed on 2D images. 3D is introduced only at a procedural level [Teboul et al. 2010a, Simon et al. 2012, Müller et al. 2006c].

Floros & Leibe [2012] exploit temporal smoothness on highway scenes. The idea is that redundant time-adjacent frames should be consistently labeled, where assumption is that between frames the motion is not too strong (always forward looking and high-frame rate) and scene content is redundantly present.

**For the 3D domain in streetside**, [Browstow et al. 2008] were the first to combine sparse SfM and semantic classification. Ladicky et al. [2012] interleaved 2.5D depth estimation and semantic labelling. In these lines Zhang et al. [2010] used dense 2.5D depth images for classification and Gallup et al. [2010a] used semantic segmentation for deciding where to use 2.5D depth for plane fitting. Munoz et al. [2012] again worked only on sparse 3D data and yet provides a method for linking these different densities of the full 2D image and sparse 3D domain. Sengupta et al. [2012] classified 2D images and then aggregated their labels to provide an overhead map of the scene. This uses a homography assumption to aggregate the birdseye map of the scenes. Most accuracy problems arise because of occlusions and averaging of multiple views. Recently, Haene et al. [2013], Kim et al. [2013] combined the creation of geometry with the semantic labelling implicitly evaluating all data redundantly.

Most related to our baseline are the works [Sengupta et al. 2013, Koehler & Reid 2013] who used 3D meshes to directly label 3D scenes. This has the benefit of using 3D features and operating in one place to fuse the classification yet still requires description and
6.2. RELATED WORK

Figure 6.2: Dataset overview - most are coarsely labelled at low resolution. We use a pixel accurate labelling with fine details at 1-3 megapixel resolution. (rightmost).

classification. Sengupta et al. [2013] showed how a common 3D classification can speed up the labelling over redundant 2D classification. Koehler & Reid [2013] introduced decision tree fields for 3D labelling to learn which pairwise connections are important for efficient inference.

Yet in summary, all of the 3D semantic research uses all data redundantly. All images are fully analyzed, described and all its features classified.

Related work for the view selection has only been carried out on an image level. Before SfM, the visual graphs are analyzed and clustered for iconic scenes [Li et al. 2008, Gammeter et al. 2010] to split the data into coherent scene parts. After SfM, camera and geometry information are used to select clusters and non-redundant views - again only at the image level [Furukawa et al. 2010, Mauro et al. 2013].

Our work is inspired by the related world of 3D model texturing, where the goal is to find an optimal single texture file for a 3D model [Debevec et al. 1998, Laveau & Faugeras 1994, Williams & Chen 1993]. Usually, for finding the single best texture, the largest projection in terms of area size or most fronto-parallel view is used in addition to lighting constancy constraints.

We propose to change this paradigm and only analyze the most discriminative views of the data. To the best of our knowledge, we are the first to actively exploit this redundancy in a multi-view semantic labeling. Further, we propose a novel view to select the best such view by selecting the best view according to its ability to classify the scene correctly.

A further note on 3D datasets, most related work only shows examples on small outdoor scenes of single coarse buildings or small indoor scenes like the NYU 3D scenes [Silberman et al. 2012]. The datasets for semantic streetside labeling consist of very few coarse labels (building, vegetation, road) and do not focus on the details of the scenes. For example, datasets like Leuven [Ladicky et al. 2012], Yotta [Sengupta et al. 2012], CamVid [Browstow et al. 2008] and the KITTI [Geiger et al. 2012] labelled for semantics by [Sengupta et al. 2013] only contain these coarse scenes labels, see Figure 6.2. Except for CamVid where there exist 700 accurately labelled ground truth images, the other
datasets only contain coarsely labelled images (in order of user strokes) from 70 to 89 images for training and testing.

In this work, we move to finely detailed ground truth labels including building detail such as windows, doors, balconies, etc. Further, the dataset is used for SfM with high resolution images of 1-3 megapixels and pixel-accurate dense labels.

### 6.3 3D Surface and Semantic Classification

Our final goal is to label each part of the scene – a 3D mesh surface – by detailed semantic labels (wall, window, door, sky, road, etc). We briefly describe the multi-view reconstruction methods to obtain the surface, the cues for semantic scene labelling, and then dive into the multi-view scene labelling problem.

#### 6.3.1 Multi-View Surface Reconstruction

Our input is a set of images which are initially fed to standard SfM/MVS algorithms to produce a mesh. SIFT features [Lowe 2004a] are extracted and matched across the images, and reconstructed along with the cameras by using incremental bundle adjustment [Wu 2013]. The estimated views are clustered and used to compute depth maps via dense MVS. Volumetric fusion is performed by tetrahedral partitioning of space over the obtained dense 3D point cloud, and by exploiting point-wise visibility information in a voting scheme [Labatut et al. 2007, Hiep et al. 2012]. The final surface is recovered using a robust volumetric graph cuts optimization [Jancosek & Pajdla 2011b].

The output of the reconstruction procedure is the set of cameras $\mathcal{C} = \{c_j\}$ and a surface mesh $\mathcal{M}$, which consists of a set of 3D vertices, a set of face edges and a set of triangular faces $\mathcal{F} = \{f_i\}$. Since we are about to assign semantic labels to faces $f_i$, we will represent this mesh as a graph, where nodes correspond to mesh faces and edges correspond to face adjacencies.

#### 6.3.2 Heavy vs. Light Features for Semantic Labelling

For semantic labelling, we extract simple 2D image and geometric features. The typical approach is to extract features for every location of every single image in the dataset. We deviate from this dense computational scheme to a sparse computation, which is a main contribution of this paper.

In contrast to related work [Sengupta et al. 2013, Koehler & Reid 2013], we split the features into two sets. The first set consists of features that will take longer time to compute:

$$\mathcal{X}^{heavy} = (L^*, a^*, b^*, t, h, d, n), \quad (6.1)$$
Figure 6.3: Features like color and gradient filters are expensive since they are densely calculated in the entire image. Geometry-based are more lightweight. Extra features like denseSIFT should improve the baseline, yet are even heavier to calculate.

This is a 16-dimensional feature vector containing the CIELAB Lab\* color components, 8 responses of the MR8 filter bank [Varma & Zisserman 2005, Geusebroek et al. 2003] in vector t, the height h defined as the distance from the ground plane, the depth d w.r.t. the dominant plane (e.g. facade plane), and the surface normal n, shown in Figure 6.3. One could use additional features here, e.g. dense SIFT, etc. See [Tighe & Lazebnik 2012, Sengupta et al. 2013, Ladicky et al. 2009] for inspiration.

To aggregate features over the projection of a face $f \in F$ in any observing camera $c$, we use Sigma Points [Kluckner et al. 2009a], which efficiently capture the first two statistical moments of the feature vectors.

The second set contains only lightweight features:

$$X^{light} = (A_{2D}, A_{3D}, A_{2D}/A_{3D}, \alpha),$$

(6.2)

where $A_{3D}$ is the area of a mesh face $f \in F$, $A_{2D}$ is the area of its 2D projection in a specific camera $c \in C$, and $\alpha$ is the angle of observation of the face from $c$.

It should be emphasized that $X^{heavy}$ relies on image content, whereas $X^{light}$ relies on geometric information only. In practice, calculation of $X^{light}$ takes only a fraction of the time (120 seconds for all 1.8 million faces and 428 camera views vs. 21+ hours needed to calculate $X^{heavy}$ for the Full428 dataset).

6.3.3 Multi-View Optimization For 3D Surface Labelling

We define a mesh graph $G_M = (F, E)$, where the nodes represent the triangular faces $F = \{f_i\}$ of the surface mesh $M$, and $E$ is the set of graph edges, which encode 3D adjacencies between the faces. We aim to assign a label $x_i$ from the set of possible semantic labels $L = \{l_1, l_2, \ldots, l_L\}$ to each of the $n$ faces $f_i$. A possible complete labelling of the mesh is denoted by $x = (x_1, x_2, \ldots, x_n)$. 


A Conditional Random Field (CRF) is defined over this graph and we aim to find the Maximum-A-Posteriori (MAP) labelling $x^*$ of the surface mesh $\mathcal{M}$. This is equivalent to an energy minimization problem of the general form

$$x^* = \arg\min_{x \in \mathcal{L}^n} E(x),$$

which we solve by efficient multi-label optimization, namely, the alpha-expansion graph-cuts [Boykov et al. 2001b, Boykov & Kolmogorov 2004, Kolmogorov & R.Zabih 2004]. Our energy consists of unary data terms for every face $f_i$, and pairwise regularity terms for every pair of adjacent faces $(f_i, f_j)$.

$$E(x) = \sum_{f_i \in \mathcal{F}} \sum_{c_j \in \mathcal{C}} \Theta(f_i, c_j, x_i) + \lambda \cdot \sum_{(f_i, f_j) \in \mathcal{E}} \Psi(f_i, f_j, x_i, x_j)$$

where $\sum_{c_j} \Theta(f_i, c_j, x_i)$ is the potential (penalty) for face $f_i$ obtaining label $x_i$. $\Theta(f_i, c_j, x_i)$ is a per-view subterm, which relies on the single specific projection (an observation) of face $f_i$ into view $c_j$. It can be written as the log-likelihood

$$\Theta(f_i, c_j, l) = -\log p(l \mid X_{ij}),$$

where $X_{ij} = \mathcal{X}(f_i, c_j)$ denotes the feature vector associated to the projection of face $f_i$ into camera $c_j$, and $p(l \mid X_{ij})$ is the likelihood of label $l \in \mathcal{L}$ for this particular projection of the face. In our scenario, the likelihoods $p(l \mid \mathcal{X})$ are provided by a random forest classifier trained on ground truth labels using the features described in Section 6.3.2.

The pairwise potential $\Psi(f_i, f_j, x_i, x_j)$ in Eq. 6.3 enforces spatially smooth labelling solutions over the mesh faces by penalizing occurrences of adjacent faces $f_i$ and $f_j$ obtaining different labels ($x_i \neq x_j$). We use a Potts model

$$\Psi(x_i, x_j) = \begin{cases} 0 & \text{if } x_i = x_j \\ \nabla & \text{if } x_i \neq x_j \end{cases},$$

where $\nabla = 1$ is a constant penalty. In the future, we plan to weight $\nabla$ in function of the dihedral angles or plane distances between neighboring faces.

The coefficient $\lambda$ in Eq. 6.3 controls the balance between unary and pairwise, data and smoothness terms. A grid search showed that $\lambda = 0.5$ works best.

Now for the fun part, it should be emphasized that each triangle $f_i$ is typically observed from multiple cameras $c_j$. This redundant set of observations poses a computational challenge when extracting the feature vectors $\mathcal{X}(f_i, c_j)$ over all views $c_j$ and for each face $f_i$. In the classical formulation, every view is considered and the final unary potential is aggregated over all views (see the second sum over the camera set $\mathcal{C}$ in the unary term of Eq. 6.3). In our findings, this is unnecessary. In the following section we describe our model of view importance and how it can be used to reduce the redundant set of views to the single most discriminative view for a more efficient semantic scene classification.
6.4 Multi-view Observation Importance

In a multi-view scenario redundancy is inherent due to the view overlaps needed for SfM/MVS. Prior work ignored the relationship between these views. In turn, we start by defining two characteristics of the computational burden.

First, **view redundancy** $R_i$ is the number of redundant camera views a mesh face $f_i$ is observed in. See the top of Figure 6.7 and Table 6.1 for some typical average view redundancy values. Each triangle of the scene is visible in up to 50 cameras ($\bar{R} = 49$) on average! We aim for zero view redundancy ($R_i \equiv 0, \forall f_i$).

Second, we define (prior) **scene coverage** $S$ as the percentage of mesh faces used for feature extraction and semantic classification. Traditionally, the entire scene is classified ($S = 100\%$). However, small areas or parts of homogeneous areas may not need to be classified individually, as the graphcut optimization in Section 6.3 is capable of spreading the correct labelling into these regions from “covered” regions, i.e. regions where the unaries in Eq. 6.3 are actually evaluated.

Our method aims at reducing both the view redundancy and the scene coverage for an efficient classification, while also improving accuracy. An initial idea could be to use a single global texture by fusing all images, and to only use this texture for extracting and classifying the heavy features. However, as we will show, the visually best texture is not always the best for semantic classification. Hence, we avoid using a fused texture, and rather keep the rich multi-view environment to decide which views are discriminative, yet before classification.

6.4.1 Ranking Observations by Importance

In this section, we are looking for the most discriminative view per mesh face in terms of semantic classification. Since SfM also delivers the exact camera models $C = \{c_j\}$, we can accurately relate each 3D surface element $f_i$ (triangular mesh face) to each of the views $c_j$, as shown in Figure 6.4. For efficiency, we aim to eliminate observations which are redundant or less important.

For this, we introduce the term **observation importance** $I$, which deviates from the existing paradigms of pairwise view clustering and ranking. In our work, we require a relationship to the 3D scene, and define $I_{ij}$ per observation of a mesh face $f_i$ in any camera $c_j$. Furthermore, our observation importance ranks according to usefulness for final semantic scene classification rather than for camera clustering or texturing.

Inspired by its success in texture mapping, we will rank the views by the simple texture features such as area and angle. However generally, we define a ranking function that
6.4. Multi-view Observation Importance

Figure 6.4: Geometric link between 3D model and 2D image space. Contrary to related work in view clustering, we look for the best view $c^*(f_i)$ per mesh triangle $f_i$. For small viewing angles the texture is visually pleasing but not best for semantic classification.

weights the cheap geometric cues for predicting the likelihood of the final classifier performance. The goal is to rank each triangle projection without the heavy feature set. Our importance rank is defined as

$$I_{ij} = p(f_i \text{ is classified correctly in } c_j|\lambda^{\text{light}}_{ij}).$$  \hspace{1cm} (6.6)

We learn to regress these probabilities, by requiring that $I_{ij}$ correlates with view and face-wise classification accuracies resulted from the classical scenario, i.e. when all views and all faces are used to extract all features. A view $c_j$ is reliable for classifying face $f_i$ if the semantic label $x^*_i = \arg\min_{l \in \mathcal{L}} \Theta(f_i, c_j, l)$ equals the ground truth label. Hence, for the training set, we extract all features and classify all observations of every mesh face. This provides binary labels for reliability (correct/incorrect). We use these and the features $\lambda^{\text{light}}$ (including, e.g. area $A^{2D}_{ij}$, observation angle $\alpha_{ij}$) to train a meta-classifier. For this, we use random forests again and, according to Eq. 6.6, we use the final leaf probability, i.e. classifier confidence, as a measure of the importance $I_{ij}$. Intuitively, views $c_j$ with small apparent area $A^{2D}_{ij}$ of face $f_i$, or views observing the face from a sharper angle $\alpha_{ij}$ should be less reliable. For completeness, we also experimented using individual features, such as area $A^{2D}_{ij}$, angle $\alpha_{ij}$, class likelihood $\Theta$ defined in Eq. 6.4, or its entropy $H[\Theta]$, to replace the importance $I_{ij}$.

6.4.2 Reducing View Redundancy and Scene Coverage

For both characteristics – view redundancy and scene coverage – we use the observation ranking in Eq. 6.6 to remove redundant views.

For view redundancy, we optimize for the best observation $c^*(f_i)$ of each face $f_i$ over all views $c_j \in \mathcal{C}$. This simplifies the energy function in Eq. 6.3 to

$$E_R(x) = \sum_{f_i \in \mathcal{F}} \Theta(f_i, c^*(f_i), x_i) + \ldots, \quad \text{with} \quad c^*(f_i) = \arg\max_{c_j \in \mathcal{C}} (I_{ij}),$$  \hspace{1cm} (6.7)
where we select only the maximally informative view per triangle instead of merging unary potentials from all observations. Thus, \( X^{\text{heavy}} \) only needs to be extracted, described and classified in these most informative views.

For \textbf{scene coverage}, we only classify a subset of all triangles that are present in the surface mesh. We choose for each face \( f_i \) the most informative view \( c^*(f_i) \) having importance \( I_{is} \). We then rank faces according to their values \( I_{is} \) and only use the set of top \( k \) faces \( F_k \subset F \) for further heavy feature extraction, rather than the full set \( F \). This further simplifies the energy to

\[
E_S(x) = \sum_{f_i \in F_k} \Theta(f_i, c^*(f_i), x_i) + \lambda \cdot \sum_{(f_i, f_j) \in E} \Psi(f_i, f_j, x_i, x_j)
\]

which contains unary potentials for only the top \( k \) mesh faces, i.e. we set the unaries of all remaining faces to zero. The smoothness term will take care of propagating labels into these areas. An optimal labelling over the complete face set \( F \) defines our final labelling solution (see bottom right of Figure 6.1).

This is where we again deviate from existing approaches, which evaluate all potentials as they have no means to rank them. Only a recent work [Roig et al. 2013] introduced the so-called Expected Label Change (ELC) ranking after sampling where to evaluate \( \Theta \) and running full optimization multiple times. In a multi-view scenario, our methods avoids such a redundant graphcut optimization to estimate the ranking, as we propose the light geometric features to directly estimate the ranking.

\section*{6.5 Experimental Evaluation}

In this section we analyze the effect of eliminating view redundancy and reducing scene coverage at the classification stage. As shown below our method considerably reduces computational burden, while showing that we not only maintain but can also improve the final classification accuracy.

We divide our experiments into two investigations summarized in Figure 6.2 and Table 6.1. First, we evaluate various importance measures as detailed in Section 6.4.1 to find the most discriminative view per mesh face. Second, we evaluate the effect of reducing the scene coverage at the classification stage.

Our datasets consist of three outdoor urban scenes annotated with ground truth labels, such as road, wall, window, door, street sign, balcony, door, sky, sidewalk, etc. CamVid [Browstow et al. 2008] is a public dataset. We use its sequence 0016E5, which contains the most buildings and frames. Note that SfM/MVS was only stable for a subset sequence of 102 of its 300 frames. We introduce the larger ETHZ RueMonge 2014 dataset (short: Full428) showing 60 buildings in 428 images covering 700 meters along Rue Monge street in Paris.
### Table 6.1: Summary of all results (details in supplemental). Semantic Segmentation accuracy (PASCAL IOU in %) for Full428, Sub28 and CamVid102 datasets. By reducing redundancy to zero and also scene coverage to 1/6th, we speedup by 2 orders of magnitude. Ranking by area is better than angle yet the 1st ranks are not best (bold).

<table>
<thead>
<tr>
<th></th>
<th>Full428</th>
<th>Sub28</th>
<th>CamVid102</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Stats</td>
<td>1794k</td>
<td>185k</td>
<td>46k</td>
<td># Triangles</td>
</tr>
<tr>
<td></td>
<td>428 (8)</td>
<td>28 (8)</td>
<td>102 (11)</td>
<td># Images (# Categories)</td>
</tr>
<tr>
<td></td>
<td>9 ± 3</td>
<td>8 ± 2</td>
<td>50 ± 27</td>
<td>Redundancy</td>
</tr>
<tr>
<td>Baseline</td>
<td>35.77</td>
<td>26.05</td>
<td>42.61</td>
<td>MAP SUMALL (λ = 0)</td>
</tr>
<tr>
<td>Eq. (6.3)</td>
<td>35.25</td>
<td>25.13</td>
<td>29.25</td>
<td>MAP MINENTROPY (λ = 0)</td>
</tr>
<tr>
<td></td>
<td>35.57</td>
<td>25.19</td>
<td>33.21</td>
<td>MAP BESTPROB (λ = 0)</td>
</tr>
<tr>
<td></td>
<td>37.33</td>
<td>26.63</td>
<td>50.80</td>
<td>GC SUMALL (baseline)</td>
</tr>
<tr>
<td></td>
<td>37.82</td>
<td><strong>26.93</strong></td>
<td>36.73</td>
<td>GC MINENTROPY ∀C_j</td>
</tr>
<tr>
<td></td>
<td><strong>38.27</strong></td>
<td>25.42</td>
<td>37.31</td>
<td>GC MAXPROB ∀C_j</td>
</tr>
<tr>
<td>SingleView</td>
<td>37.38 (8px)</td>
<td>26.09 (18px)</td>
<td>52.19 (135px)</td>
<td>Ranked 1st GC AREA (avg)</td>
</tr>
<tr>
<td>Eq. (6.7)</td>
<td>37.38 (8px)</td>
<td>26.60 (15px)</td>
<td>54.60 (62px)</td>
<td>Ranked 4th GC AREA (avg)</td>
</tr>
<tr>
<td></td>
<td>35.73 (9°)</td>
<td>25.64 (8°)</td>
<td>47.84 (37°)</td>
<td>Ranked 1st GC ANGLE (avg)</td>
</tr>
<tr>
<td></td>
<td>36.06 (15°)</td>
<td>26.34 (24°)</td>
<td>50.04 (41°)</td>
<td>Ranked 4th GC ANGLE (avg)</td>
</tr>
<tr>
<td></td>
<td>37.04 (0.19)</td>
<td>26.19 (0.49)</td>
<td>52.62 (0.70)</td>
<td>Ranked 1st GC LEARN (avg)</td>
</tr>
<tr>
<td></td>
<td><strong>37.64</strong> (0.18)</td>
<td><strong>26.86</strong> (0.47)</td>
<td><strong>56.01</strong> (0.63)</td>
<td>Ranked 4th GC LEARN (avg)</td>
</tr>
<tr>
<td>Coverage</td>
<td>38.37 (15%)</td>
<td><strong>28.28</strong> (27%)</td>
<td><strong>61.07</strong> (35%)</td>
<td>Best Accuracy (AREA)</td>
</tr>
<tr>
<td>Eq. (6.8)</td>
<td>37.68 (14%)</td>
<td>26.39 (12%)</td>
<td>57.08 (20%)</td>
<td>1st as Baseline (AREA)</td>
</tr>
<tr>
<td></td>
<td>35.73 (35%)</td>
<td>26.83 (74%)</td>
<td>54.37 (16%)</td>
<td>Best Accuracy (ANGLE)</td>
</tr>
<tr>
<td></td>
<td>35.67 (35%)</td>
<td>25.76 (22%)</td>
<td>52.20 (13%)</td>
<td>1st as Baseline (ANGLE)</td>
</tr>
<tr>
<td></td>
<td>37.08 (35%)</td>
<td>27.97 (40%)</td>
<td>60.57 (31%)</td>
<td>Best Accuracy (LEARN)</td>
</tr>
<tr>
<td></td>
<td>36.15 (33%)</td>
<td>25.96 (34%)</td>
<td>52.98 (13%)</td>
<td>1st as Baseline (LEARN)</td>
</tr>
<tr>
<td>Timing</td>
<td>1280min</td>
<td>88min</td>
<td>184min</td>
<td>TIME Full View Redundancy</td>
</tr>
<tr>
<td></td>
<td>11.9x</td>
<td>8.6x</td>
<td>52.6x</td>
<td>SPEEDUP Zero Redundancy</td>
</tr>
<tr>
<td></td>
<td>108min</td>
<td>10.2min</td>
<td>3.5min</td>
<td>TIME Zero Redundancy</td>
</tr>
<tr>
<td></td>
<td>7.1x</td>
<td>8.3x</td>
<td>5.0x</td>
<td>SPEEDUP 1st Coverage as Eq. (6.3)</td>
</tr>
<tr>
<td></td>
<td>15min</td>
<td>1.2min</td>
<td>0.7min</td>
<td>TIME 1st Coverage as Eq. (6.3)</td>
</tr>
<tr>
<td></td>
<td>85x</td>
<td>72x</td>
<td>262x</td>
<td>SPEEDUP Overall</td>
</tr>
<tr>
<td></td>
<td>+1.04%</td>
<td>+1.65%</td>
<td>+11.81%</td>
<td>GAIN Overall (absolute)</td>
</tr>
<tr>
<td></td>
<td>103%</td>
<td>106%</td>
<td>124%</td>
<td>GAIN Overall (relative)</td>
</tr>
</tbody>
</table>

It has dense and accurate ground-truth labels (Figure 6.2). Sub28 is a smaller set of 28 images showing four buildings. The CamVid dataset is taken from a car driving forward on a road (with an average viewing angle of 40°) while in the other two datasets the human camera man points more or less towards the buildings (avg. angle ≈ 10°).
6.5. Experimental Evaluation

Figure 6.5: Removing View Redundancy: showing accuracy for the single k-th ranked feature on x-axis (e.g. 1st largest area, 10th smallest angle, 4th learned importance) and average feature value (red dash). The smaller the area or the larger the angle, the worse performance gets. Our learned performance captures the combination of area and angle better. This is CamVid, other datasets are in supplemental material.

We split each dataset into independent training and testing buildings of roughly 50% of the images and train using all observations of all triangles of the training set. We train both classifiers using a random forest [Amit et al. 1996, Breiman 2001] because of its inherent abilities to handle multiple classes, label noise and non-linearity of the features. The number of trees is optimized to 10 and depth to 20 levels.

Please note that our method to reduce view redundancy and scene coverage is general and the speedup generalizes to other semantic classification pipelines. Hence, to study the exact differences, we use the graphcut optimization explained in Eq. (6.3) over all views as our main baseline (see Table 6.1).

6.5.1 Single Discriminative Views - Zero Redundancy

In this first experiment, we determine the most discriminative measure for observation importance. We evaluate the measures in terms of semantic scene classification using PASCAL IOU accuracy averaged over all classes. Table 6.1 is a summary of our findings. Please look in the supplemental material and website for more detailed results.

As one would expect, exploiting all of the view redundancy and averaging the classifier confidence from each observation (SUMALL) provides stable results. However, these approaches do not provide any speedup and require all the heavy features to be extracted over all observations.

Yet calculating all potentials is the time consuming task, hence we focus on how to find the best observation from cheap geometric features only. The measures to rank are apparent face area $A^{2D}$ (AREA), viewing angle $\alpha$ (ANGLE), and our importance in Eq. 6.6 (LEARN).

From the evaluations, we have three conclusions. First, on average using the 2D projection area works better than the viewing angle. This is likely due to more robust statistics
6.5. Experimental Evaluation

Figure 6.6: Reducing Scene Coverage: showing accuracy over percentage of selected triangles within graph optimization. Dashed lines are accuracy at full coverage (allviews, maxarea, minangle, importance). On average 30% are sufficient to label the entire scene as correctly as 100% coverage! Last rows show classwise results (see text for details).

of larger areas and implicit preference for closer views, as the viewing angle is scale-invariant. Despite the challenging datasets of hugely varying appearance (training to testing performance drops roughly by 30%), other experiments show that the view invariance of the classifier is inherently quite high, which could further explain why the minimum angle is not as useful.

Second and surprisingly, our findings show that neither the largest 2D area nor the most fronto-parallel view deliver the best performance. Rows 10-14 in Table 6.1 show the average area/angle to change several units for slightly better results. This gain is higher for CamVid because of the steep forward-looking camera and also because of the different semantic classes. For more detail over the class-averaged measures in Figure 6.5, we also looked at classwise results for area. For all datasets, the classes captured by changing thin 3D surfaces (pole, fence, door, window, sign/pole, etc) experience a gain in accuracy with less frontal projections. These findings suggest that for these classes slanted views better capture the 3D structure.

Overall, our learned combination of the light features works best, since it can balance the distortion of the area and the extreme viewing angles.
6.5.2 Reduction of Scene Coverage

In the second experiment, we investigate how many total mesh faces are really essential for good performance semantic classification in multi-view scenarios. Going one step further, we reduce the scene coverage and only select the top $k$ triangles after selecting the most discriminative view per triangle.

Here our baselines are a) using all redundancies and the zero redundancy of b) area and c) angle - all at full coverage. The results are shown as average over all classes (top) and as classwise results (bottom) in Figure 6.6. First conclusion is that the area is usually better at selecting the important triangles for coverage. Its curve climbs faster and overall its accuracy is higher, except for steep-angled CamVid dataset. Here the angle measure works better, and overall our learned importance combining the two is best.

Second conclusion may surprise again, we can even get better than the baselines at full coverage (dashed lines)! This is explained by the smaller classes (which occur less frequently and cover less space). Not sampling these early, removes competition for the large classes, which perform much better here. Hence, it is the size of the area that matters. As the importance measure is less good at the early coverage (below 10% coverage), we visualized the three measures and learned that the area is spread across the scene where our learned ranking focuses more on high confidence classes like building and road.

Third and most important conclusion, for large classes it is enough to use 10% of the scene coverage to reach the baselines. Overall, around 30% scene coverage stable results are obtained for all classes. This means that 70% of the potentials usually calculated for semantic scene segmentation are not necessary. The same accuracy can be achieved by using our proposed observation importance and optimization over the graph neighborhood.

6.6 Conclusions

In this work we investigated methods for reducing the inherent data overlap of multi-view semantic segmentation. As the speeds for other parts have been improved, the bottleneck is the redundant feature extraction and classification.

By exploiting the geometry and introducing single discriminative views per detailed scene part (a triangle), we avoid the redundancy and only classify a single time. This provides a speedup in the order of the data redundancy.

Further, we showed that simple features used for texture mapping are not best when the goal is semantic scene classification. Our learned importance better combines the features like area and viewing angle and improves the ranking.
6.6. Conclusions

a. Full428 (zooms below)  b. Sub28 (failure ¼)  c. CamVid (SfM part)

Figure 6.7: Overview of results - top left is full street, view redundancy as heatmap (more redundancy, the greener), ground truth (zoomed for two parts of street), and results for full redundancy, single best view and best score for coverage (at stable 30%). Overall, the accuracy are the same after all our speedups. Middle column shows failure cases (¼), where the initial classifier already fails and gracefully further smooths the results.

Lastly, we proposed further efficiency by reducing the scene coverage and classifying only 30% of the scene and still obtain accurate labels for the entire scene. All in all, after reducing the redundancy and coverage we even increase the overall accuracy.

For future work we noticed that the overall accuracy of the scene classification depends on the resolution of this mesh as too large triangles cover semantic units and small triangles are not reliable for classification. Hence we plan to find the best resolution and rank even features in terms of their computational effort.
7

Rules for reconstructions

We propose a novel grammar-driven approach for reconstruction of buildings and landmarks. Our approach complements Structure-from-Motion and image-based analysis with a ‘inverse’ procedural modeling strategy. So far, procedural modeling has mostly been used for creation of virtual buildings, while the inverse approaches typically focus on reconstruction of single facades. In our work, we reconstruct complete buildings as procedural models using template shape grammars. In the reconstruction process, we let the grammar interpreter automatically decide on which step to take next. The process can be seen as instantiating the template by determining the correct grammar parameters. As an example, we have chosen the reconstruction of Greek Doric temples. This process significantly differs from single facade segmentation due to the immediate need for 3D reconstruction.

7.1 Introduction

Over the last years, the efficient creation of 3D models, ranging from single landmarks up to whole cities, has received increasing interest. Hereby landmarks play a particularly important role. Structure-from-Motion (SfM) approaches are popular methods to build such models from image sequences. They do not require expensive hardware and the images can be re-used for model texturing. Yet, SfM has problems which have proven to be difficult to solve [Cornelis et al. 2008]. It is doubtful whether further refinements to the typical SfM pipelines, i.e. better bottom-up processing, can overcome these issues. Therefore, it is worth exploiting prior knowledge about the scene. In the case of buildings, such knowledge can be provided through shape grammars, which describe the structure of buildings. Procedural modeling, based on such grammars, is a powerful method to efficiently produce 3D building models [Müller et al. 2006d]. It offers a lightweight semantically meaningful representations instead of huge mesh files. However, procedural modeling has mainly been used for the creation of new virtual buildings, not for reconstruction of existing ones. The process of ‘inverse’ procedural modeling has so far essentially been limited to the analysis of facades.
7.1. Introduction

We propose to create 3D models of buildings, by combining SfM, building element (‘asset’) detection, and inverse procedural modeling. The latter incorporates a shape grammar interpreter which drives the process. The usage of asset detectors avoids fragile segmentation processes and leverages recent progress in object class recognition. As grammars are specific for a particular building style, like our Doric temples illustration, they need to be pre-selected correctly. This is not critical however, as one can automatically mine images and information from Wikipedia pages of landmark buildings [Quack et al. 2008]. The Wikipedia pages typically specify the building style. This is also the case for the examples shown in this paper. It is also important to note that the mined images often do not allow for a complete SfM reconstruction.

The remainder of the paper is organized as follows: Sec. 7.2 discusses the related work. Sec. 7.3 describes the four main components of our system, namely the grammar interpreter (Sec. 7.3.1), the asset detector (Sec. 7.3.2), the 3D reconstruction module (Sec. 7.3.3) and the vision module (Sec. 7.3.4). Sec. 7.4 describes how these work together to instantiate a procedural 3D model. The validity of our approach is shown in a case study in Sec. 7.5. Finally, Sec. 7.6 concludes the paper.
7.2 Related work

In the field of urban architecture modeling, numerous approaches are available. In the following we give an overview of representative works.

Grammar-based architectural modeling has a long history. In 1975, Stiny [1975] introduced the idea of shape grammars. These were successfully used in architecture, but due to their over-expressiveness the applicability for automated generation of buildings was limited. A breakthrough came with the introduction of split grammars [Wonka et al. 2003b], with mechanisms to enable automatic rule derivation. Further development of this idea led to the introduction of CGA Shape [Müller et al. 2006d], a shape grammar designed for the procedural generation of large-scale urban models [Procedural 2010, Kimberly et al. 2009].

A framework for interactive grammar editing has been introduced by Lipp et al. [2008], making procedural modeling more accessible. Approaches to derive grammar rules from input images have been presented by Aliaga et al. [2007b], Muller et al. [2007], Van Gool et al. [2007]. While the first approach uses an interactive method to construct a grammar from the images, the others try to automate this process, relying on regular and repetitive facade patterns. In their recent work, Vanegas et al. [2010] use a rewriting grammar to describe the building geometry, based on aerial imagery. Cornelis et al. [2008] present an approach for stereo-based, real-time, but simplified 3D scene modeling. Gallup et al. [2010b] has demonstrated a way to also handle non-planar surfaces. An approach for automatic partitioning of buildings into individual facades has been described by Zhao et al. [2010], while Xiao et al. [2009b] uses the concept of facades to reconstruct street-side models of buildings.

Probabilistic approaches have gained ground since the influential work of Dick et al. [2004], where a model-based, Bayesian approach with Markov Chain Monte Carlo (MCMC) optimization is used. Alegre & Dellaert [2004] use a stochastic context-free grammar and MCMC methods to deduce semantic information from building facades, relying on color constancy and rectangular shapes of facade elements. Ripperda & Brenner [2006b] apply reversible jump MCMC methods for facade reconstruction, together with a formal
grammar for facade description. Recently, Teboul et al. [2010c] have used a coarse probabilistic interpretation of the facade to match the instantiated grammar model to the observed image. In order to find the grammar parameters, they kick-start the process with a pixel-wise segmentation and labeling step and then employ an algorithm for random walk exploration of the grammar space.

Our approach combines the robustness of a top-down grammar-based approach with the flexibility of the bottom-up image-based approach. Our main contributions are: (1) The reconstruction process is guided by the grammar. Instead of the developers having style-specific guidelines in mind when producing the system, a grammar interpreter tool renders the process more generic. It is the grammar that decides on what to do when. Moreover, structures that may not even be visible can be filled in. (2) Rather than relying on fragile segmentation processes to kick-start the semantic analysis, the grammar chooses the matching available detectors to assign initial semantic labels to image regions. (3) The system learns from its previous results. For instance, asset detectors self-improve by using earlier results as additional training material. This also allows us to start with rather generic asset detectors, which have not been developed uniquely for the targeted style.

7.3 Main system components

Our system is composed of four main components:

- **Grammar interpreter**: Analyzes the input shape grammar, extracts semantic information and leads the reconstruction process.

- **3D reconstruction module**: Generates a 3D point cloud from the input images through uncalibrated SfM.

- **Asset detectors**: Extracts bounding boxes of ‘assets’ (building substructures) in the images.

- **Vision module**: Improves the detections by using 3D and the semantic information coming from the grammar.

We will use ‘shape symbol’ to refer to a string or name in the grammar, which refers to a class of shapes. In case a detector is available for that class of shapes, that type of shape is referred to as an ‘asset’. Windows, doors, or pillar shafts are examples of assets in our system.

The input to our system is thus a set of images of the building to reconstruct, a database of asset detectors and a style grammar whose class of possible derivations includes the building of interest. The above components are generic and have each been elaborated to
the point where they support the Doric temple showcase. For instance, we have trained
detectors of capitals and pillar shafts, but would not have detectors for important elements
in other styles yet (except for very general classes like windows and doors). Similarly,
CGA shape grammars come with a gamut of rules, of which our system handles Repeat
and Split rules. A simple CGA grammar is given in Fig. 7.3. The Doric temple CGA
grammar underlying this paper is provided as supplementary material.

Fig. 7.2 shows how the parts of the system interact. First (1), the grammar interpreter
initializes the vision module with a list of shape symbols automatically extracted from
the grammar. They are then compared with the list of symbols that represent trained asset
detectors from our database. The matching symbols (assets) are identified, reported to
the grammar interpreter (2) and the detection (Sec. 7.3.2) process is initialized for those
assets resulting in detection bounding boxes in all input images (3-4).

The images are fed into the 3D reconstruction module ARC3D [Vergauwen & Van Gool
2006] to obtain a sparse 3D point cloud and the camera parameters from the building
(5-6). For the matched symbols (detectable assets) the grammar interpreter parses the
grammar to find structural information e.g. spatial configuration or repetitions of these
symbols (step 7).

The vision module (Sec. 7.3.4) uses a plane fitting algorithm to extract the dominant
planes of the building. The detections from all images are projected into 3D and re-
weighted based on consensus in 3D and the structural information. The output of this
vision module are the sizes of the detected assets and their color, the footprint for the
building and the parameters for the structural configurations (step 8). Finally the building
can be instantiated by the grammar interpreter by using the extracted parameters.

7.3.1 Grammar Interpreter

In this paper we use CGA Shape grammar for the description of our procedural models.
CGA Shape has been successfully employed in various urban reconstruction applications
[Müller et al. 2006d, Kimberly et al. 2009], it has a standardized description with powerful
shape operations while remaining readable to humans and a commercial tool (CityEngine)
exists for rendering 3D models from CGA Shape rules [Procedural 2010]. The CGA
Shape grammar supports a very large number of different operations and functions in its
rules, which are outside the scope of this paper. Therefore, we only present the essence
of the grammar.

Grammar definition

The grammar is defined by four components [Müller et al. 2006d]:
7.3. Main system components

- A finite set of shapes $\mathbb{S} = \{S_1 \ldots S_n\}$
- A finite set of attributes $\mathbb{A} = \{A_1 \ldots A_m\}$
- A finite set of shape operations $\mathbb{O} = \{O_1 \ldots O_k\}$
- A finite set of production rules $\mathbb{R} = \{R_1 \ldots R_l\}$:

  $$\text{pred} \rightarrow (\text{cond}) \text{ succ},$$

  where the pred(ecessor) shape is replaced by the succ(essor) shape, if the cond(ition) evaluates to true.

Model production starts from an initial shape, which is most commonly the building footprint. This shape is gradually refined as rules are successively applied. We now concisely describe these concepts. For more details see [Müller et al. 2006d].

**Shapes**

Each shape consists of a symbol, geometric and numeric attributes. The symbol is the identifier of a shape, and is usually just a string. Geometric attributes correspond to the scope, an oriented bounding box in space, which is defined by the starting point, three main direction vectors and a size vector. Each shape can be either a terminal or a non-terminal. The latter is replaced by other shapes using the production rules. Terminal shapes correspond to simple geometric primitives, like cubes, planes, etc. or more complex full 3D meshes.

**Shape Operations**

- **Scope operations** modify the scope of a given shape and include translation, rotation, and resizing.
- **Split operations** split the scope along a given axis, with split sizes as attributes.
- **Repeat operations** indicated with a '*' repeat a shape in a given direction as long as there is enough space. In CGA Shape they are written as a part of a split rule. The actual number of instantiated shapes depends on the rule attributes and the scope size of the predecessor shape. E.g. a window tile gets repeated over the whole length of a floor.
- **Component split operation** splits 3D scopes into shapes of lesser dimension, e.g. faces, edges, or vertices.
Automatic Extraction of Semantic Information

In order to get the semantics of the building from a given shape grammar, we automatically construct a tree-like structure. Its nodes represent shapes, split, component split and repeat operations, capturing the structure of the building. The process begins with the extraction of shape symbols, and their classification as terminal or non-terminal shape symbols. In the next step, the rule set is analyzed, creating the tree structure. The interpreter also extracts the attributes from the grammar and assigns them to the appropriate nodes in the tree. An example grammar and its tree structure are shown in Fig. 7.3.

After the interpreter has analyzed the input grammar, the extracted symbols are fed to the vision module, which then returns the list of detectable assets. The assets constrain the grammar interpreter to extract only structure and composition information pertaining to assets. For each asset, it queries the semantic shape tree to retrieve the number and direction of repeat configurations the shape symbol appears in. This information is then sent to the vision module to re-weight the existing detections (see Sec. 7.3.4).

In the next step, we perform queries on all pairs of assets, determining their possible mutual composition. Assets typically correspond to multiple shapes in the shape hierarchy. Therefore, we check if all of the instances of one asset are in the same configuration with instances of a second asset. The configurations we can extract from the shape tree are:

- One shape is part of another shape
- One shape is on top/left/bottom/right of the other, relative to parent scope

For the Doric temple example, the system notices that capitals are on top of shafts, and such coupling information is passed on to the vision module. Similarly, it would notice that windows are parts of facades, but not always on top of doors.
7.3. MAIN SYSTEM COMPONENTS

7.3.2 Asset detector

An important part of the strategy is to keep available a large set of asset detectors. We have used Felzenszwalb’s [Felzenszwalb et al. 2010], trained on a few hundred hand-labeled examples for each asset. The detectors output bounding boxes of image regions where the asset was found, together with a score. Of course, there are the usual false positives and false negatives. The exploitation of the grammar helps the vision module in re-weighting or pruning those or just reducing their weight.

Another important aspect of our system is its ability to improve the detectors based on its previous ‘experience’. For instance, the capital and shaft detectors that are activated to handle the Doric temples in this paper have been trained on a diverse set of examples, including Ionic and Corinthian style in addition to Doric ones. As the system arrives at high confidence detections during the re-weighting process, it can then collect specific training examples to specialize the current general detector to one better suited for Doric temples as shown in the example Fig. 7.4. This online learning increases the chances of success to reconstruct the next Doric temple as demonstrated in 7.5.1.

7.3.3 3D reconstruction module

For the creation of a 3D point cloud from the images of a building, we use the publicly available, online web service ARC3D [Vergauwen & Van Gool 2006]. It employs a SfM approach which also estimates the camera positions and calibrations. The meshed surfaces provided by ARC3D are not used, as its 3D information is only needed to support our system and not to deliver parts of the output model. One can imagine that one might
eventually want to use part of the ARC3D meshes for ornamental structures, if they were not available as assets.

### 7.3.4 Vision module

While the grammar interpreter guides the reconstruction process, the vision module gathers the information from the 3D data, the detectors and is responsible for substantiating the semantic information extracted from the grammar. It consists of four main components.

1. The plane estimator, that extracts the dominant planes from the sparse 3D point cloud.
2. The module for 3D reasoning is responsible for projecting the 2D object bounding boxes from the images into 3D and to estimate the assets sizes.
3. The spatial relations between different assets is utilized to re-weight matching assets by the module for spatial configurations.
4. Eventually, detections for assets that appear in a repeat rule of the grammar are enhanced by similarity detection.

The modules for 3D reasoning, spatial configuration and similarity detection implement a re-weighting scheme \( w_{3D}, w_{sp} \) and \( w_{sim} \) of the detection score \( S_{det}^i \) of the \( i \)-th detection. The final score \( S_{final}^i \) is calculated as:

\[
S_{final}^i = S_{det}^i \cdot w_{3D}^i \cdot w_{sp}^i \cdot w_{sim}^i
\]  

(7.1)

**Plane estimator**

We apply RANSAC [Fischler & Bolles 1981] as the basic algorithm to extract facades from the point cloud. To improve the quality of the detected planes we reduce the point cloud to points that project into detection bounding boxes in the images. This leads to planes going through the assets of interest. We set the inlier threshold proportional to the size of the point cloud. We stop extracting planes when the number of inliers of the final estimate is less than a given fraction of the total number of points. Furthermore, as soon as we have more than two planes detected, we calculate the gravity vector and the ground plane through the vector product of the plane normals, under the assumption of vertical planes. The footprint of the temple is extracted from the intersection of the ground plane with the facade planes. Fig. 5 illustrates the process of finding the planes.

**3D reasoning module**

The planes detected in the sparse 3D model are used to back-project the bounding boxes from all views into 3D. In the planes in 3D overlapping detections are clustered. The clusters \( C_j \) are found in a greedy fashion. The detections get sorted by their score and
starting from the best scoring detection as a cluster center, all overlapping detections are added to that cluster. A detection that does not overlap with any previous defines a new cluster. The weight $w_{3D}$ accounts for the size of the cluster and the ‘rectangularity’ of the detection. The latter is defined by the ratio $A_p / A_{br}$, the area of the polygon $A_p$ obtained by back-projecting an image detection bounding box onto the 3D asset plane and the area of the corresponding polygon’s minimum bounding rectangle $A_{br}$. This rectangularity ratio decreases the influence of detections that come from cameras with a non-perpendicular angle to the plane, as detections coming from an angle oblique to the plane produce spread polygons in 3D. Every detection belongs to a cluster $C_j$, represented by the cluster center (the detection of the cluster with the highest score). The ratio of the size of the cluster $|C_j|$ and the number of times $n$ the cluster center can be seen by cameras rates the size of the cluster.

$$w_i^{3D} = \frac{A_p}{A_{br}} \cdot \frac{|C_j|}{n}$$ (7.2)

After thresholding by detection scores, cluster size and rectangularity, the remaining detections belonging to each cluster are used to find the spatial extent of the detected assets (see Fig. 7.6). The intersection area of these back-projected detections is orthogonally projected to the x and y axes (the y axis being aligned with the gravity vector) to find the asset’s dimensions (red and green arrow).

**Module for spatial configurations**

This module uses semantic information coming from the grammar interpreter. The re-weighting $w_{sp}$ is based on the spatial configurations of detections. The grammar interpreter informs the vision module about the possible spatial relations ($c_1 \ldots c_k$) between two elements. For these elements the following relations are possible: “left of”, “right
Figure 7.6: Determining the assets size: the red and green arrows indicate the estimated height and width respectively.

The score of every detection that appears in a given relation is boosted by a constant multiplier $\alpha$ (in our experiments $\alpha = 1.1$).

**Similarity detection**

When the grammar interpreter informs the vision module that an asset appears in a repeat rule, it expects as an answer the repeat distance. The similarity detection not only extracts this parameter but calculates a new weight $w_{\text{sim}}^{i,t}$ for the detections of the repeat rule. For detections with no information about repetition $w_{\text{sim}}^{i,t}$ is set to 1. The presence of a repeat rule implies directly that the asset included in that rule will appear several times along a given axis. This module searches for this periodicity. It helps the detection performance in three ways. Assets that have not been detected so far can be inferred by similarity to a detected one. As the repeat is defined along an axis, the repeated assets are expected to
lie on a certain line, the similarity line. The relative distance of the repeated assets to that similarity line defines $w_{\text{sim}}^{t}$. The parameter for the repeat distance is found as a byproduct of the repetition detection.

Fig. 7.7 shows the structures deemed similar for the detection marked with a red rectangle.

**Similarity voting** For every image a global voting space (accumulator) is created. The similarity voting is based on local image features. Every feature $F_{t}$ is described by its position $\tilde{p}_{t}$, its scale $s_{t}$ and the feature descriptor $\tilde{d}_{t}$, i.e. $F_{t} = (\tilde{p}_{t}, s_{t}, \tilde{d}_{t})$. The algorithm iterates over all asset detections in an image and uses an ISM-like voting scheme [Leibe et al. 2004] to find similar detections.

Starting with one detection, the set of indices $I$ denotes all features $F_{i}$ inside that detection bounding box. They get assigned a vote vector $\bar{v}_{i}$ towards the center of the box. The indices $J$ refer to the remaining features $F_{j}$ outside. For each feature $F_{j}$ we determine the vote vector by a nearest neighbor search as follows:

$$\bar{v}_{j} = \frac{s_{j}}{s_{k}} \cdot \bar{v}_{k} \text{ with } k = \arg\min_{i \in I} \{||\tilde{d}_{i} - \tilde{d}_{j}||\}$$  \hspace{1cm} (7.4)

Now, every image feature $F_{t}$ is associated with a vector $\bar{v}_{t}$ and cast a vote by adding a Gaussian Kernel with sigma $\sigma$ centered at $\tilde{p}_{t} + \bar{v}_{t}$ to the accumulator. The process is repeated for all detections of the image resulting in one voting space per image.

In our implementation we use Hessian Affine interest points [Mikolajczyk & Schmid 2004] and SIFT feature descriptors [Lowe 2004b]. The value for $\sigma$ is dependent of the mean detection bounding box size of the current image.
**Similarity Line Extraction**  Similarity lines are detected separately for each image that contains detections. The best similarity lines are found by the lines through the maxima of that voting space using RANSAC [Fischler & Bolles 1981]. The number of similarity lines searched per image depends on the number of planes seen in the image and on the given grammar. E.g. a split-rule immediately in front of a repeat rule implies to search for more than one line per plane. Maxima of the voting space that lie on that line but do not correspond to any detection in the image are used to infer new detections.

**Finding the re-weighting factor**  The similarity lines found in all images are back-projected onto the planes of the model. The back-projection lines all vote in a Hough space to find the globally best similarity lines. Detections not corresponding to these lines are considered as outliers and are re-weighted based on their distance to the lines.

\[
 w^{d}_{\text{sim}} = e^{-\frac{d^2}{2\sigma^2}}, \text{where } \sigma = \frac{\Delta}{4\sqrt{2\log 2}} \tag{7.5}
\]

The value for \( \sigma \) is calculated according to the mean detection bounding box extent \( \Delta \) along the axis perpendicular to the similarity line.

**Repeat distance**  Even if a detection is not perfectly centered at the detected asset, the maxima in the voting space of this detection are shifted equally from the center, resulting in equal distances between the maxima of the voting space. For a fronto-parallel view, an extra voting space is generated tracking these distances for all detections in all images. The maximum of that voting space is the parameter used as the repeat distance (period). By using frontal views, the distances are less error prone to errors in the plane detection process.

### 7.4 Grammar attribute estimation

At the end of the recognition stage, we have the estimated values of asset sizes and the spacing of assets in repeat configurations. We also have the estimated size of the building footprint. The grammar interpreter then translates these parameters into the appropriate grammar attributes. In a typical scenario, the grammar will have additional attributes that we cannot estimate using the asset detectors alone (e.g. ornaments). For these attributes we use the default values present in the grammar. This approach enables us to “give an educated guess” for objects not visible in the images, but which have to be there due to the structural constraints imposed by the grammar.
7.5 Case Study - Doric Temples

Classical temples conform to strict architectural rules, which have been thoroughly analyzed in literature [Summerson 1996]. These rules have been converted into a shape grammar representation. We show the reconstructions of three Greek Doric temples: The Parthenon in Nashville, a full-scale replica of the original Parthenon in Athens. The Temple of Athena (also known as Temple of Ceres) and the Temple of Poseidon, both archaic Doric temples in the ancient city of Paestum. The results are summarized in this section. Fig. 7.1 shows steps of the reconstruction process for the Nashville temple.

7.5.1 Asset Detectors

To train our asset detectors we use the publicly available implementation of Felzenszwalb’s detector [Felzenszwalb et al. 2010]. To cope with the higher variability in different types of capitals we have trained this detector as a two component detector, whereas the shaft detector consists of only one. For our first detector we hand-labeled a few temple images of different styles, resulting in 189 annotations for capitals and 204 for shafts.

The reconstruction of the Temple of Poseidon resulted in 188 + 124 (capitals+shaft) newly gathered samples that we added to the training set, now specialized in Greek Doric temples. Keeping the false positive rate fixed at 2.2% for capitals and 5.4% for shafts we increased our detection rate by 7.31% and 14.89% respectively.

7.5.2 Temple Grammar

A very simplified version of a grammar that describes classical temples is described in this section. We focus on the colonnade (the sequence of columns) as this is the most relevant part which contains our detectable assets\(^1\).

\(^{1}\)The full grammar for our experiments is available in the supplementary material.
The colonnade is first split in the $x$ direction into columns. Note that the repeat (marked with the *) does not include all columns. The side columns are handled separately resulting in a different spacing between the repeated columns and the the spacing to the left and right column. This fact is directly captured in the grammar rules, but cannot easily be inferred from the images alone. A column is further divided into capital and shaft. Due to this rule, the grammar interpreter informs the vision module about the relation “capital on top of shaft”. The insert rule replaces the 3D volume by an asset from the database. As seen in the excerpt above, the parameters for column spacing, capital height and shaft height appear directly in these rules. The remaining parameters, namely the column width, shaft width and temple color are extracted from the full derivation tree. The lot size is estimated from the point cloud and not directly encoded in the grammar. Further parameters are either dependent on the estimated attributes or set to default values (e.g. the roof angle).

### 7.5.3 Results

Fig. 7.8 shows instantiations of the Parthenon replica in Nashville and Temple of Athena, respectively. Properties like the number of columns can easily be found from the detections. These are not grammar attributes but can be inferred through the instantiation process. In table 7.1, we compare the dimensions of Temple of Poseidon with our estimations. All parameters are scaled to the size of the temple width. Sizes measured in the images are marked (*), while the groundtruth sizes are taken from [TUFTS University 2010]. Column height is the size of capital height + shaft height. The large error for the capital height can be explained by the view angles at which the pictures were taken (ground imagery). This results in the capital appearing taller than it really is.

### 7.6 Conclusion and Future Work

This paper has introduced a novel way of 3D building reconstruction using shape grammars. As opposed to previous approaches, the grammar drives the reconstruction process. Also, detectors provide a good starting point for estimation of the grammar parameters. Furthermore, the system improves itself by automatically specializing the applied detectors. The validity of our approach is shown on a case study of classical Doric temples.
### Table 7.1: Size comparison for the Temple of Poseidon.

<table>
<thead>
<tr>
<th></th>
<th>Reconstruction</th>
<th>Original</th>
<th>Ratio</th>
</tr>
</thead>
<tbody>
<tr>
<td>temple width</td>
<td>24.26</td>
<td>24.26</td>
<td>1.0</td>
</tr>
<tr>
<td>temple length</td>
<td>58.51</td>
<td>59.98</td>
<td>0.98</td>
</tr>
<tr>
<td>shaft width</td>
<td>2.13</td>
<td>2.11</td>
<td>1.01</td>
</tr>
<tr>
<td>column height</td>
<td>9.28</td>
<td>8.88</td>
<td>1.04</td>
</tr>
<tr>
<td>capital width</td>
<td>2.56*</td>
<td>2.72</td>
<td>0.94</td>
</tr>
<tr>
<td>capital height</td>
<td>1.37*</td>
<td>1.04</td>
<td>1.32</td>
</tr>
<tr>
<td>inter-column distance</td>
<td>4.45</td>
<td>4.48</td>
<td>0.99</td>
</tr>
</tbody>
</table>

As future work, we will extend the set of CGA rules to extract information from. Furthermore, a matching phase between the estimated model and the original images will be added to verify and fine-tune the estimations of the parameters. We plan to also use this matching for the inclusion of extra shapes via the ARC3D meshes (like ornamentation) and to add effects of destruction and erosion as parts to be displaced or taken off the original model.
8

Grammatical inference

Urban models are key to navigation, architecture and entertainment. Apart from visualizing façades, a number of tedious tasks remain largely manual (e.g. compression, generating new façade designs and structurally comparing façades for classification, retrieval and clustering).

We propose a novel procedural modelling method to automatically learn a grammar from a set of façades, generate new façade instances and compare façades. To deal with the difficulty of grammatical inference, we reformulate the problem. Instead of inferring a compromising, one-size-fits-all, single grammar for all tasks, we infer a model whose successive refinements are production rules tailored for each task. We demonstrate our automatic rule inference on datasets of two different architectural styles. Our method supercedes manual expert work and cuts the time required to build a procedural model of a façade from several days to a few milliseconds.

8.1 Introduction

Mitchell [1990] claims that architecture is structured by a certain logic which can be captured by formal grammars. Procedural modelling has extensively been used by architects, urban planners, the film and game industries, map makers and cultural heritage specialists to generate large-scale models of cities [Watson et al. 2008]. The introduction of procedural modelling has cut down the required amount of work to synthesize convincing models of cities, which used to take several man-years [Müller 2010]. Procedural models are semantic and highly structured, and are very well-suited for simulations [Aschwanden et al. 2009] and planning [Halatsch et al. 2008] compared to conventional 3D models. The main features of procedural models are that they are compact, editable, readable, semantic and advantageous for retrieval and fast graphics generation [Haegler et al. 2010].

While whole virtual cities can be generated in minutes thanks to procedural modelling, it takes a lot more effort when it comes to constructing a model of an existing city. Existing
Figure 8.1: Pure mesh-based or semantic labelled models suffer from a limited field of uses. In this work we propose methods to automatically build procedural façade models in milliseconds for compression, comparison and new virtual façade creation.

Inverse procedural modelling pipelines for example [Teboul et al. 2010a; 2011b, Mathias et al. 2011b] require an expert to manually design a set of style-specific rules. Architects estimate that manual modelling takes them up to two days to model a single building — several years to model a city. In addition to the colossal amount of work involved, it also makes any update process very slow.

The quality of 3D city modelling for visualization has dramatically improved over the past few years. Using Structure-from-Motion pipelines, works such as [Xiao et al. 2009a] have achieved a very high degree of realism. Yet, the resulting meshes are only good for visualisation. In parallel, semantic labelling has shown very encouraging results [Martinoović et al. 2012, Dai et al. 2012]. However, the possibilities given by a set of labelled façades is limited compared to the set of applications offered by a procedural model (see Fig. 8.1). In this work, we transform a set of labelled façades into a procedural model, automatically. Additionally, we use the procedural models to generate new façade instances and compare façades layouts.

In a procedural model, rules are the backbone of the semantic information. They describe how architectural elements are grouped together and organised. The rules created for inverse procedural modelling should not only be able to generate a set of exemplar buildings in a style, but specifically only the possible buildings from a style. This poses the problem of finding a principled way of inferring such a rule set. As opposed to automatic inference, manual work takes a considerable amount of time and does not ensure consistency between the different models. Finally, the resulting rule set may contain a very large set of parameters, making the optimization intractable. To circumvent these problems, we propose a new formulation where the grammar is directly inferred from the labelling of a façade or a set of façades (see Fig. 8.2).

Our main contributions are: (1) an inverse procedural modelling pipeline where both rules and parameters are automatically inferred, (2) a method to measure the structural distance between façades for retrieval and clustering, (3) a method to synthesise new, non-existent façades from a model describing a set of façades in the same style, (4) a compression of
8.2 Related work

Facade modelling in form of image-based architectural modelling [Akbarzadeh et al. 2006, Xiao et al. 2009a] and semantic segmentation [Berg et al. 2007] is not relevant to our problem due to its restrictions to pure visualization and reporting. In fact, the work performed in image-base city modelling is a pre-processing step to our pipeline. Our goal is to use the procedural structure of facades for compression, comparison and virtual layout generation by exploiting shape grammars.

Shape grammars were initially proposed by Stiny & Gips [1972] as a generation tool for geometric paintings and sculptures. Later, Wonka et al. [2003a] and Müller et al. [2007] proposed to use split shape grammars to describe architecture. The idea is to describe a building as a sequence of formal rules which gradually refine and add details to the model. Next we detail two main approaches that have been pursued to tackle architectural inverse procedural modelling using shape grammars, and then present related work in the field of grammatical inference.

The first inverse procedural modelling approaches assume that a grammar is given as input and infer the appropriate parameters to represent a given building [Ripperda & Brenner 2006a, Teboul et al. 2011b; 2010a]. These methods offer the advantage of systematically yielding architecturally sound buildings, but the actual correctness depends on whether the grammar can create the building at hand as an instantiation. Recent work relaxes the restrictions and use rather general architectural principles [Martinović et al. 2012] or symmetries and repetitions [Riemenschneider et al. 2012] to infer the parameters.

In the second approach, both the rule set and the parameters are inferred. Inverse procedural modelling of L-systems has been tackled by Šťava et al. [2010]. Bokeloh et al. [2010a] produce procedural models of a mesh assuming it contains partial symmetries. Grammatical inference is also of interest in the analysis of natural language, music and genetics. De La Higuera [2005] present a comprehensive survey about grammatical inference. Charikar et al. [2005] examine the smallest context-free grammar problem. They show that it is an NP-complete problem and compare grammar-based compression algorithms, such as [Nevill-Manning & Witten 1997].

However, to the best of our knowledge, grammatical inference has never been performed for more than one facade (except for the simultaneous work [Martinović & Van Gool 2013]) at a time and without any architectural style nor grid layout restrictions.
8.3. Approach

Our goal is to automatically derive a set of procedural rules to describe a set of façades of the same style, generate new ones and compare them for classification and retrieval. One or several Manhattan-world segmentations of façades are given as input, which can be obtained manually or automatically [Berg et al. 2007, Dai et al. 2012, Martinović et al. 2012, Riemenschneider et al. 2012].

Our weakly supervised algorithm for inverse procedural modelling provides as output, sets of style rules and for each façade, a set of parameters. We employ different representations for each different application.

8.3.1 Shape grammars for architecture

A shape grammar $\mathcal{G}$ is a context-free grammar consisting of a tuple $(\mathcal{N}, \Sigma, \mathcal{R}, S)$, where $\mathcal{N}$ is a finite set of non-terminal shapes, $\Sigma$ is a finite set of terminal shapes, $\mathcal{R}$ is a finite set of production rules and $S$ is a starting shape (axiom). A shape is enclosed in a 3D bounding box called scope. A terminal symbol shape can be a 3D model or a texture.
which we will refer to as asset (as in computer graphics terminology). An asset is hence an architectural element, such as a door, a window or a piece of wall. A production rule \( r \in \mathcal{R} \) consists of a sequence of operations. An operation transforms a shape into other shapes. We consider *insert*, *horizontal* and *vertical* split operations. Insert operations insert an asset in the current scope, while split operations divide a shape into \( n \) newly created shapes. We distinguish between two kinds of split operations: binary and n-ary. We call a binary split a split between two elements such as

\[
r_{\text{binary}} \rightarrow \text{split}(d)\{s_1 : r_1 | s_2 : r_2\}
\]  

(8.1)

where \( d \) is the splitting direction (horizontal or vertical), \( s_1 \) and \( s_2 \) are the sizes of the newly created shapes, and \( r_1 \) and \( r_2 \) their associated rules.

With shape grammars, a façade is described as a recursive splitting procedure from the root node (corresponding to the axiom \( S \)) to the leaf nodes (each a single asset). Semantically higher level operations are compact notations for a series of lower level operations. Higher-level operations provide an *interpretation*, an *understanding* of the building. A number of implementations have been developed such as CGA [Müller et al. 2007], GML [Havemann 2005] and Teboul et al. [2010a]’s.

Operations, and therefore rules, are parametric. Hence, a rule set plus a set of parameters generate a single façade. Altering one or the other will result in a different façade. We refer to a grammar instance as a set of shape grammar production rules plus a set of associated parameters.

### 8.3.2 What makes a good grammar?

The same façade can be represented by many different sequences of operations, therefore rule sets. Hence, we pay special attention to defining the properties a “good” grammar should fulfill. *The value of a grammar depends on its intended usage.* In the case of façade modelling, we consider visualization of an existing city, novel building generation, comparison between façades, reporting and compression. Bearing in mind the targeted applications, we propose to adopt the following criteria for the grammar generation. The grammar inference should be: consistent, fast, online and should produce an accurate, generative (within class), specific (between class) and compact grammar. Here, consistency means that two similar façade labellings will be described by two similar rule sets. By online algorithm, we mean that new façade instances can be added iteratively. This is a very important property, as in practice a city model needs to be constantly updated. Often, optimizing for one criteria will degrade the others. For instance, achieving Minimum Description Length (MDL) is likely to come at the cost of longer computations and the need to run an optimization over the whole dataset. Consequently, we propose to use different representations for different applications. The generative model, which creates new, virtual building instances for visualization, should be conservative and specific,
as any error will be caught by the human eye. The analytic model, which can be used to compare buildings, should generalize such that it encompasses all façades from the style.

### 8.3.3 Façade parse tree generation

In this section, we describe the parsing algorithm which, starting from a labelling, encodes a façade as a binary split tree whose nodes correspond to façade regions, operations and parameters. The collection of trees is the starting point for all subsequent processing in Section 9.4. The parsing algorithm recursively splits the façade into smaller façade regions until all of them consist of a single asset. This can be seen as a top-down tree clustering of the façade.

Generating the split trees boils down to defining an energy function to select the split lines. The structure of the parse tree will solely be affected by the order of the split, determined according to this energy function. The rest of this section describes the energy function given in Eq. 8.3. Intuitively, the energy function aims at grouping assets which occur frequently next to each other and are not separated by a long edge. As the parsing is performed on labelled images, an edge designates a straight line that separates regions with different labels. The set of asset labels is $\mathcal{L}$.

In the rest of this section, $d$ designates a direction, $h$ means horizontal and $v$ vertical. For each scope $s$, we start from a set of horizontal and vertical split line proposals $\mathcal{P}$ defined as

$$\mathcal{P} = \mathcal{P}_h \cup \mathcal{P}_v \quad (8.2)$$

where $\mathcal{P}_h$ is the set of horizontal split line proposals and $\mathcal{P}_v$ the set of vertical split line proposals. A line is a split proposal $p \in \mathcal{P}$ if a) it contains at least an edge collinear to the direction of $p$ and b) it does not intersect with any edge perpendicular to $p$. The latter condition prevents from splitting across an asset. For simplicity, we only write the equations for horizontal split proposals, which can be easily be extended to vertical proposals. $(x, y)$ refer to the coordinates of a pixel on a proposal line. For each split proposal $p_y \in \mathcal{P}$, we compute an energy function $f(p_y)$ using an edge length term $e_x$ and an asset affinity $v_d$ such that

$$f(p_y) = b^{(h)}b^{(a)} \left( \alpha \frac{1}{W} \sum_{x=0}^{W} e_x + (1 - \alpha) \frac{1}{W} \sum_{x=0}^{W} v_d(x, y) \right) \quad (8.3)$$

where $b^{(h)}$ is a horizontal vs. vertical bias, $b^{(a)}$ is a parental bias, $\alpha \in [0, 1]$ is a weight between the edge support term and the affinity term, $W$ is the size of the façade, $e_x = 0$ if a pixel $(x, y)$ is an edge, 1 otherwise. The edge length term $e_x$ rewards splitting along
edges, while the affinity term $v_d$ penalizes splitting between assets with stronger affinity, i.e. those which frequently occur as neighbors in the data. The biases are

\[
 b^{(h)} \begin{cases} 
 \in [0, 1] & \text{if } p \text{ is horizontal} \\
 1 & \text{if } p \text{ is vertical}
\end{cases}
\]  

(8.4)

\[
 b^{(a)} \begin{cases} 
 \in [0, 1] & \text{if } d(p) = d(a) \\
 1 & \text{otherwise}
\end{cases}
\]  

(8.5)

where $d(a)$ is the direction of the last accepted proposal (of the parent scope) and $d(p)$ the direction of $p$.

The second term sums over the affinity $v_d(x, y)$ between the asset at the edge pixel $(x, y)$ and the asset at the nearest facing edge pixels $(x, \bar{y}^*)$, which is located according to

\[
 \bar{y}^* = \arg \min_{\bar{y} \in \mathcal{P}_{\bar{A}}(\Delta_{y, \bar{g}})}
\]

(8.6)

where $\mathcal{P}_{\bar{A}}$ is the set of split proposals which do not belong to the same asset. $\Delta_{y, \bar{g}}$ is the distance between proposals $p_y$ and $p_{\bar{g}}$. The search space in Eq. 8.6 can be reduced by only considering edges located in the direction of the normal vector to the edge. The affinity $v_d(x, y)$ is defined as:

\[
v_d(x, y) = c_d(l_{x,y}, l_{x,\bar{y}^*})
\]

(8.7)

where $l_{x,y}$ is the asset label at position $(x, y)$ and $l_{x,\bar{y}^*}$ the asset label at the nearest facing edge. The co-occurrence $c_d$ between asset pairs, where $d$ is the direction, is computed across all façades and stored in two affinity matrices $C_h$ and $C_v$. These matrices $C_d$ are normalized weighted co-occurrence matrix of size $|\mathcal{L}| \times |\mathcal{L}|$, where $|\mathcal{L}|$ is the number of asset labels. Each value in $C_d$ is defined by

\[
c_d(l_{x,y}, l_{x,\bar{y}^*}) = \sum_{i=1}^{\nu} \left(1 - \frac{|y_i - \bar{y}_i^*|}{W}\right)
\]

(8.8)

where $\nu$ is the total number of pixels belonging to split proposal lines, and the proximity measure is the normalized distance between a pixel belonging to a split proposal $p_y$ and its nearest facing asset edge located along $p_{\bar{g}}$. In order to reduce the number of computations, the affinity coefficients are pre-computed for each pixel over the façade in both horizontal and vertical directions.

At the end of this parsing step, we obtain a binary tree of split operations that describes the façade. This tree can be represented as set of binary split rules (see Eq. 8.1). The
whole process is exactly lossless, i.e. the original labelling can be re-generated from the tree of rules.

In the next section we show how to use these parse trees to optimize for compression, retrieval and virtual generation of new façades.

8.4 Optimization of Shape Grammars

Given the general method to construct a parse tree of the façade layouts, it is our goal to optimize the grammar with respect to its production rules for the each of following specific applications, separately.

1. **Compression** reduces the size of the grammar by examining redundancies within and between parse trees.

2. **Comparison** defines structural features to create a retrieval metric capturing differences in façade layout.

3. **Virtual façade synthesis** analysis examples of façade types and creates a new set of consistent parameters to instantiate a façade of the same style.

8.4.1 Grammatical inference

In the previous section, we presented a way to infer a parse tree depicting a given façade. However, the inferred tree consists only of binary *split* operations, i.e. single operation rules. The grammatical inference phase is a succession of steps to group those single operation rules to infer a shorter representation. Note that the inferred grammar provides us with an understanding of the building as lower level operations are combined into fewer number of higher level operations. For instance, repetitions of the same rule across the building are detected. In addition, rendering performance benefits from using a compact grammar as fewer rules need to be evaluated at render time. The grammatical inference is a two step process. First, the binary split nodes in each parse trees are converted into n-ary split nodes. Then, more complex production rules are inferred by comparing n-ary split nodes over all parse trees.

**Transformation to n-ary split nodes**

Nested binary splits in the same direction (horizontal or vertical) are re-written as n-ary splits following
8.4. OPTIMIZATION OF SHAPE GRAMMARS

\[
\begin{align*}
\{ r_i & \rightarrow \text{split}(d)\{s_j : r_j | s_k : r_k\} \\
\{ r_j & \rightarrow \text{split}(d)\{s_{j1} : r_{j1} | s_{j2} : r_{j2}\} \\
& \Leftrightarrow \\
\{ r_c & \rightarrow \text{split}(d)\{s_{j1} : r_{j1} | s_{j2} : r_{j2} | s_k : r_k\} \\
\end{align*}
\] (8.9)

Since the position and size of each scope remains identical, this transformation is guaranteed to be lossless. Each such transformation reduces the total number of rules |R| by 1 (two rules were turned into one). It is performed multiple times by recursively exploring the parse tree until no more occurrences can be re-written.

**Production rule inference**

The production rule inference is based on a similar principle as [Busatto et al. 2004, Nevill-Manning & Witten 1997]. We recursively replace repeating structures with a rule. When two or more similar nodes are found, those are re-written as a single parametric rule.

To perform this transformation, each pairs of the nodes are first compared. Two nodes are considered similar if they have the same operation and similar children, regardless of their numerical parameters (i.e. size values). Formally, as shown in Eq. 8.10, \( r_\alpha \) and \( r_\beta \) can be re-written as a parametric rule \( r_\phi \) such that

\[
\begin{align*}
\{ r_\alpha & \rightarrow \text{split}(d)\{s_{1}^\alpha : r_1 | ... | s_{k}^\alpha : r_k\} \\
r_\beta & \rightarrow \text{split}(d)\{s_{1}^\beta : r_1 | ... | s_{k}^\beta : r_k\} \\
& \Leftrightarrow \\
r_\phi(x) & \rightarrow \text{split}(d)\{x_1 : r_1 | ... | x_k : r_k\} \\
\end{align*}
\] (8.10)

Now a single \( \text{split} \) rule is to be invoked by changing the values of the parameter vector \( x = (x_1, x_2, ..., x_k) \). This transformation is guaranteed to losslessly preserve the layout of the scopes.

After each transformation, the number of rules is changed by the reduction of repeated rules as

\[ |R|_t = |R|_{t-1} + 1 - \theta(\psi + 1) \] (8.11)

where \( |R|_t \) is the total number of rules at the iteration \( t \), \( |R|_{t-1} \) the total number of rules before the transformation, \( \theta \) the number of occurrences of the rule and \( \psi \) the number of child rules. From Eq. 8.11, we note that the more similar rules exist, the more efficient the rule inference will be at reducing the total number of rules. In order to produce similar
rules, the split tree should be as consistent as possible. Also, the more child rules $\psi$, the smaller the total number of rules $|\mathcal{R}|_t$ with respect to $|\mathcal{R}|_{t-1}$.

Comparing two nodes in the tree implies comparing their children. In order to avoid traversing the tree multiple times, the nodes are compared in a bottom-up fashion. First, only nodes whose children are *insert* operation are compared and replaced by production rules if possible. Later, nodes whose children are *insert* or production rules are considered until it is not possible to create any new production rule. Note that the parameters of the child rules are carried over the parameter vector of the newly inferred rule.

### 8.4.2 Compression

The rule set $\mathcal{R}$ is optimized for the smallest number of rules following the Minimum Description Length (MDL) principle by solving for

$$\arg \min_{\alpha, b_p, b_h} (|\mathcal{R}|)$$

where $\mathcal{R}$ is a rule set describing the input facades inferred by the method given in Sect. 8.3.3 to Sect. 8.4.2, $\alpha$, $b_p$, and $b_h$ are the parameters defined in Eq 8.3.

### 8.4.3 Comparing façades

Retrieval and clustering are two application examples for comparing façades. More formally, comparing façades means finding an adequate distance function $\delta$ as

$$\delta : \mathcal{R} \times \mathcal{R} \rightarrow \mathbb{R}^+$$

As all façades of the same style are similar and share visual features, using a feature-based distance would be inappropriate. We propose two different distance measurements. The first is based on an MDL paradigm, while the second is a powerset-based distance derived from the parse trees.

The MDL-based approach measures the similarity of façades in terms of their common rules. As [Cilibrasi & Vitányi 2005] we define that two façades $A$ and $B$ are more similar than $A$ and $C$ if

$$\frac{|\mathcal{R}_A \cup \mathcal{R}_B|}{|\mathcal{R}_A| + |\mathcal{R}_B|} < \frac{|\mathcal{R}_A \cup \mathcal{R}_C|}{|\mathcal{R}_A| + |\mathcal{R}_C|}$$

where $|\mathcal{R}_A|$ is the number of rules used to describe façade $A$ after compression. We create a histogram of the frequency of rules shared by two façades and use a $\chi^2$ as an appropriate distance. We refer to this distance as $\delta_{\text{common}}$. 

In the second approach, we compare the binary split trees. To this end, we consider the powerset $\mathcal{P}\mathcal{S}(O \cup L)$ of the set of operations $O$ and asset labels $L$. For each façade parse tree, we draw a histogram which reflects the number of the elements of $\mathcal{P}\mathcal{S}(O \cup L)$ and use a $\chi^2$ as a distance measure. We refer to this distance as $\delta_{\text{powersets}}$.

### 8.4.4 Virtual façade synthesis

In this section, we show how to create new, non-existent façades from a set of real building façades in the same style. To build virtual cities, experts manually model a few typical buildings and relax their parameters by assigning ranges from which the parameters are randomly drawn. This approach is inspiring since it produces valid buildings as the structures are not altered, and yet it delivers a good illusion of diversity.

At the end of the production rule generation, we have a set of parametric rules, each dependent on one or more parameter vectors $x$. Each façade of the input set is identified by a starting rule and a parameter vector. The starting rules correspond to rules that split the whole façade, generally into floors and balcony layout. Each of these starting rules will then call the hierarchy of rules describing the structure of the façade. The parameter vectors specify all the sizes used in the rules. To generate a new façade, we instantiate the starting rule with a new set of parameters. By doing so, we sample the parameter space while preserving the structure. We have to make sure the new parameters will be consistent, i.e. it is important to preserve the correlations between the different vector variables. Assuming these parameters follow Gaussian distributions, the correlations are discovered by applying a PCA on the parameter vector set.

### 8.5 Evaluation

In this section, we evaluate the compression, façade retrieval and virtual façade generation. Further we discuss the losslessness and computational cost of the rule inference.

#### 8.5.1 Experimental setup

In our setup we evaluate on two different datasets. First, the ECP2011 façades dataset [Teboul et al. 2011b] comprises 104 labelled images taken in rue Monge, a street in Paris. The architecture is representative for Haussmannian style, which is also found in other cities like Buenos Aires or Brussels. Second, a subset of Graz2012 [Riemenschneider et al. 2012] is selected, which consists of 30 annotated façades in Gruenderzeit style, which is widespread in Germany and Austria.
8.5. Evaluation

Figure 8.3: Distance matrix for the ECP2011 and Graz2012 datasets, re-ordered according to its dendrogram (log scale). Some of the associated labellings are shown on the right.

Figure 8.4: Cumulative Match Characteristics (CMC) for ECP2011 for semantic façade retrieval. The powersets distance better captures the structural similarities over common rules distance.

8.5.2 Losslessness and computational cost

Each step of the grammatical inference algorithm is perfectly lossless. In fact, one can regenerate the original labelled image by replacing the assets by colour patches.

As shown in Fig. 8.5, the computational cost of the inference algorithm (i.e. parsing, n-ary split compression, rule inference and data collection for statistics) is linear with respect to the number of input façades. This is a very important property, as it shows the inference algorithm scales up to the size of whole cities. Our implementation of the inference algorithm takes about 32 ms per façade on a single core of an Intel Core i7 930. Our method works online and is applicable in practice to model whole cites. A new façade can be added to the dataset at a linear cost. Finally, all steps in the inference algorithm can be parallelized as well.
Figure 8.5: Computational time with respect to the number of Haussmannian façades given as input (left); number of rules with respect to the number of Haussmannian (centre) and Gruenderzeit (right) façades given as input, using no compression, n-ary compression and n-ary and rule inference compression (log scale).

8.5.3 Compression

Our findings indicate that the parameters are robust for a large range. For example, for ECP2011 the minimal number of inferred rules is 68, and is found for all values of $\alpha \in [0.25, 1.0]$, $b_p \in [0.0625, 0.5]$ and $b_h \in [0.125, 1.0]$. In the results shown in this section, we use parameters $\alpha = 0.6$, $b_p = 0.25$ and $b_h = 0.5$.

The growth of the number of inferred rules with respect to the number of input façades is shown in Fig. 8.5 for ECP2011 and Graz2012. We can reduce the number of rules using compression by two orders of magnitude (notice the logarithmic scale). Especially, the more regular ECP2011 dataset shows a clear drop in rule growth. This shows that the core logic principles of the Haussmannian style can be explained after examining about 20 façades.

However, we also see that in addition to core principles within a style, exceptions are the rule. This translates to a continual growth of the number of rules in Fig. 8.5 when new façade samples are added. In the Graz2012 dataset, a large number of rules are only used once. Further investigations would elucidate whether this stems from architectural diversity (needed for all applications) or annotation noise.

8.5.4 Façade comparison

The goal of façade retrieval is to compare a query façade to the set of known façades and determine the most similar ones. In our scenario we are not comparing appearance or the sizes of architectural elements, but the procedural layout of the façade. It is our goal to group façades which have the same layout in terms of floors, window columns, balconies and door placement.
8.5. Evaluation

Following the two distance measures $\delta_{\text{common}}$ and $\delta_{\text{powersets}}$ defined in Sect. 8.4.3, we evaluated a façade retrieval and clustering on the datasets\(^1\). Creating a ground truth for measuring distances is a tedious task due to the number of pairs to evaluate ($104^2$ for the ECP2011 dataset). We defined a gold standard distance function as the number of architectural changes in the number of floors, window columns, door placement and location of running balconies. We manually annotated each façade with its related data\(^2\).

For retrieval, we evaluate the distance measures $\delta_{\text{common}}$ and $\delta_{\text{powersets}}$ and count if the ground truth façades with no architectural changes (deemed identical) are retrieved in the top-K ranking. This measure is typically known in identification retrieval as the Cumulative Match Characteristics (CMC) [Moon & Phillips 2001] and shows how many retrieved results to look at before finding the desired result.

Optimizing the parameters in Eq. 8.3 in order to maximize this CMC retrieval score at rank $k = 20$ for the ECP2011 dataset gives $\alpha = 0.5$, $b_p = 0.0625$ and $b_h = 0.125$.

In Fig. 8.4, we compare the two methods detailed in Sect. 8.4.3. Using the powerset method, we see that retrieving the exact instance within $k = 1$ has a mean expectation accuracy of 87% whereas within $k = 2$ all the correct façades are retrieved. The common rule approach does not yield such good results in comparison (56% at $k = 2$). This strongly supports our claim that different representations are suitable for different applications.

For clustering, we use the distance measure $\delta_{\text{powersets}}$ and can show distinct groups between and within each of the façade datasets for Haussmannian and Gruenderzeit styles, as indicated by the dendrogram and the linked heat maps as shown in Fig. 8.3. We can see that the distance measurement effectively accounts for structural changes. The distinction between the two styles is clear due to the differences in the frequency of asset types. For instance, shops and balconies are more common in Haussmannian. Consequently, the two logics can be automatically separated and hence two style grammars can be inferred.

8.5.5 Virtual façade synthesis

For virtual façade synthesis, the quality of the sampling improves when the number of parameter vector instances associated with each rule increases. Hence, the optimization of the grammatical inference for virtual façade synthesis follows the same objective function as for compression (see Eq. 8.12). Examples of virtual façades are shown in Fig. 8.6. Notice the greater variance of the door position, heights of the balconies and roof, in contrast to the variance of the width of windows. Here textures and colours were not randomized to emphasize the structural changes only. Stiny & Gips [1972], Mitchell [1990], and Wonka et al. [2003a] have developed an ontology for architecture. We were

---

\(^1\)Examples of retrieval are shown in the supplementary material.

\(^2\)The ground truth annotation is available on the author’s website.
Figure 8.6: Rows correspond to different façade structure type (i.e. different starting rules), columns correspond to different \( \sigma \) values which influence parameter variations.

able to judge how well shape grammars are suited for capturing a set of real examples from a style by evaluating compression, comparison and synthesis. The generated rule set for Haussmannian summarizes the main features of the style. **The most frequent rules correspond to:** 7 floors (including the ground and roof floor), 4 window columns, running balconies on the 2\textsuperscript{nd} and 5\textsuperscript{th} floors and shops on the ground floor.

### 8.6 Conclusion

In this work we show that procedural models provide a much larger flexibility than pure mesh-based or semantic labelled representations by enabling compression, façade comparison and new virtual façade synthesis. Our method starts by a binary split procedure on labelled image to create parse trees and consequent procedural rule sets. The final grammar models are optimized on the requirements for compression and virtual synthesis (minimum number of rules inspired by MDL) and retrieval (best ranking performance inspired by bag-of-words models).

Our evaluations confirm that a single grammar model is not enough. The optimization results for compression and retrieval produce different models with different performances. In case of retrieval the performance nearly doubles with a more tailored grammar model. In all, our method removes the need for manual expert work and cuts time to build a procedural façade model from days to milliseconds.
The benefits of our procedural knowledge can be used to highlight the atypical parts in a façade and automatically complete occluded areas. Also the generated grammar rules could be translated to human language to teach architectural principles to humans.

Future work entails lifting the process to 3D and include depth as well as entire buildings into the grammar models. We will also investigate improving noisy semantic image labelling methods with the inferred grammar rules and build joint labelling and grammar inference methods. Finally, the answer to the title is yes.
Next generation navigation

Navigation has been greatly improved by positioning systems, but their visualization still relies on maps. Yet because they only represent an abstract street network, maps are sometimes difficult to read. Conversely, Tourist Maps, which are enriched with landmark drawings, have been shown to be much more intuitive to understand. However, outside of a city centre, major landmarks are too sparse to be helpful. In this work, we present a method to automatically augment maps with drawings of the most locally prominent landmarks, at every scale. Further, we generate a characterization which helps emphasize the special attributes of these buildings. Descriptive features are extracted from facades, analyzed and re-ranked to match human perception. To do so, we collected a total number of over 5900 human annotations to characterize 117 facades across 3 different cities. Finally, the characterizations are also used to produce natural language descriptions of the facades.

9.1 Introduction

During the last decade, navigation aids have undergone a genuine revolution. Rather than having to trace their trajectories on maps, people get real-time information about where they are and what to do next to reach their goal. Nonetheless, much of this novel technology is still grounded in the use of traditional maps. Unfortunately, many people find maps difficult to use [Blades & Spencer 1987]. As perception research has shown [Tom & Denis 2003], people tend to remember trajectories on the basis of decisions relative to landmarks, rather than guidelines like ‘first take the second street on the left, then third on the right’. With online navigation we are still pretty much following the latter paradigm. Before starting their journey, people therefore often try to get acquainted with the appearance of the goal location, using tools like Google street view. What makes navigation still difficult? First, once on the road, street view navigation is too cumbersome to continuously use. Second, maps rely on street names and house numbers, which are not always visible. Last, navigational aids would be most needed where the language or even alphabet are different, precisely when maps are the most difficult to use.
9.1. Introduction

Figure 9.1: Our automatically mined local landmark buildings maps (top) vs. classical separated Street View (middle) vs. Street network (bottom) maps and street view imagery are respectively very abstract or hard to browse. If major landmarks are sometimes annotated on tourist maps, these are too sparse. We automatically mine atypical buildings and highlight them at a given scale.

Figure 9.2: Overview of our approach: Features are extracted from facades and their labellings, Outliers are detected and clustered. The results are re-ranked according to human perception. New applications such as relative text description, facade search, speech for navigation and the unusualness map of a city arise.

As a matter of fact, traditional tourist maps offer a good compromise for this. They combine an - often metrically incorrect - map with drawings of major landmarks. The creation
of such maps was not a coincidence, as it offers people the appearance-based reference they would not find on traditional maps. Such maps can be generated automatically by mining information about landmarks from the web [Grabler et al. 2008]. Nevertheless, traditional tourist maps are only useful in city centres, where landmarks are dense. The visual aid quickly starts to lose its effect once the user enters the maze of smaller streets, where there are no tourist landmarks such as an Eiffel tower or a Big Ben.

In this work, we propose to use an adaptive tourist map that automatically shows the most helpful buildings to navigate in any city area. The destination and all important buildings on the user’s path are highlighted, at every scale. Since they form the landscape which is visible from street level, we work directly at the facade level. We analyze each facade and produce a semantic description of its main architectural characteristics. This description is based on other buildings in the neighbourhood and highlights the differences. In particular, our work learns the saliency of different facade attributes, to emphasize what is important in the eyes of humans.

Last but not least, our method also is capable to produce language-based descriptions of the facades. Therefore, even when it is dangerous to take one’s eyes off from the street when walking or driving, a spoken voice can provide descriptions of the landmarks of interest. Our contributions are:

- A method for discovering atypical buildings in an area.
- A method to produce a visual characterization and describe a facade in natural language relative to its surroundings.
- We study the relevance of different features as perceived by humans, provide a statistical analysis and adapt our tool to match human perception.
- We introduce a new facade dataset for the city of Zurich, and a dataset of 5904 human annotations for unusualness of 117 facades across Graz, Paris and Zurich.

9.2 Related work

The topic most related to our work is facade saliency for navigation. In Raubal & Winter [2002], Nothegger et al. [2004], the idea of selecting local landmarks is defined and features covering many aspects such as building visibility and function are used. The main differences to our work is that we focus on the visual and perceptual aspect. We use extra visual features and a novel outlier detection pipeline to match our building selection scores with the human perception of saliency. In Winter et al. [2005] simple linear weights between different saliency features are established experimentally. Raubal [2004]
focuses on evaluating facade saliency depending on the user context. Finally, Sadeghian & Kantardzic [2008] presents challenges in automatic facade mining.

Concerning maps, the two works most related to ours are Grabler et al. [2008], Doersch et al. [2012]. In the first, Grabler et al. augment a map with visual abstractions of landmarks. The position and pictures of landmarks are mined from the internet. In our work, we also augment a map with unique buildings displayed for reference, which are mined following a unusualness analysis for facades. The seminal work “What makes Paris look like Paris?” [Doersch et al. 2012] solves our inverse problem, i.e. identifying the patterns that are typical for a city. In contrast, we find the oddities of facades within a style.

In recent times, much effort has been devoted to augmenting maps with 3D models. The level of detail which has been reached in reconstructions allows for detailed visualization of entire cityscapes. These are however very large and still prone to reconstruction errors. As a result, semantic labelling has been used as a way to address these two issues. For example, image-based modelling [Xiao et al. 2008b] and procedural modelling [Müller et al. 2006b, Teboul et al. 2011b, Riemenschneider et al. 2012, Martinović & Van Gool 2013, Weissenberg et al. 2013] allow to enhance the reconstruction result and reduce the amount of data. Yet, the resulting maps are only more accurate and visually pleasing, and not a fundamentally better tool for navigation. In this work, we aim at filtering out the excessive amount of information pertaining to buildings to help users navigate. Please note that here, we do not detail the pre-processing steps which are required to obtain an initial semantic 3D model of a city. We refer the reader to [Akbarzadeh et al. 2006, Xiao et al. 2009a, Jancosek & Pajdla 2011b, Zhou et al. 2013] about urban and dense 3D reconstruction and [Martinović et al. 2012, Teboul et al. 2011b, Riemenschneider et al. 2012] about automatic labelling, for details about implementing them.

Unusualness can be defined from different angles. In general, an event is usual if it can be explained by prior knowledge, and unusual if this contradicts. From a statistical perspective, we call an unusual event an outlier. As there is no unique and formal definition of an outlier, we use the one of Barnett & Lewis [1994]: “An observation (or subset of observations) which appears to be inconsistent with the remainder of that set of data. The methods for unusualness detection rely on a wide palette of techniques including classification, nearest neighbour, clustering, statistical models and entropy-based methods. We refer the reader to the comprehensive survey by Chandola et al. [2009]. We use in particular the Local Outlier Factor (LOF) [Breunig et al. 2000], which gives a measure of the relative degree of isolation of a sample.

Despite all this, what is unusual may not necessarily match the human perception of what is important in an image. In particular, the works of Turakhia & Parikh. [2013], Berg et al. [2012], Sadovnik et al. [2013] focus on determining the human-centric importance of attributes. Compared to [Turakhia & Parikh. 2013], we are able to find the most unusual attribute for a single instance. Kulkarni et al. [2011] focuses on the relations between
9.3 Approach Overview

Our goal is to discover atypical facades and what attributes make them special. Fig. 9.2 gives an overview of our approach. In this work, we start with ortho-rectified facades and semantic labellings of each architectural asset. Here we assume such labelling is available, as automated solutions exist [Teboul et al. 2011b, Martinović et al. 2012] and have steadily and rapidly improved in the last few years.

We denote the following terms: Asset is the main architectural elements of a facade, e.g. windows, doors, balconies. Attribute types are the properties of the assets, e.g. size, color, material, shape. Attribute values are the numerical or fuzzy values, e.g. tall, wide, large, blue, green.

First, features are computed for each facade and for each asset within the facades. An outlier score is estimated for each value with respect to the neighbourhood. When a feature value is an outlier with respect to the distribution of feature values for other facades in the neighbourhood, it is potentially perceived as an unusual feature. In this work, we
9.4 What makes a facade unique?

Figure 9.4: Unusualness (green) vs. Noise (red) - the construction work (Left) and car and pedestrian (Right) are distinctive image features but are temporary occluding the facade. In contrast, the black shutters (Left) and brown door (Right) help distinguish between the two facades.

tackle two questions: mining unusual facades and characterizing a facade by describing its unusual features.

To spot what makes a facade unique, we need to identify its characteristic features, i.e. these which help discriminate between different facades. These discriminative features correspond to the most unusual ones (see Section 9.2 for the definition of unusualness).

For our purpose, the challenges are numerous. First, abnormality is a contextual notion. For instance, a balcony on a facade may be perceived as unusual or very usual depending on the style, the street and the neighbouring facades. Then, the boundary between usualness and unusualness is fuzzy. This makes it difficult to obtain ground truth data, which will also be fuzzy in nature. Also, unusualness must be distinguishable from noise in the data, see Figure 9.4.

In addition, an unusualness detection system must, by its nature, monitor a very wide gamut. Finally, in our case, what is measured as statistically unusual must be matched with the human perception of unusualness.

In particular, some features may be perceived by humans as more important than others, or easier to perceptually distinguish. Therefore, the list of statistically most unusual elements needs to be re-ranked such that the resulting list matches with what humans perceive as unusual.

In the rest of this section, we first introduce the notations we will be using and present the method both to determine which facades are the most unusual and what is unusual about them. After that, the outlier detection algorithm is detailed in Section 9.4.3. Then,
as outliers are not necessarily the most discriminative parts of a facade, we present a method to learn the link between outliers and their discriminative perception. Finally, Section 9.4.5 explains how the facade elements are clustered.

9.4.1 Notations

We are given a set of rectified facade images $\mathcal{I}$ and their associated labellings $\mathcal{L}$. Each labelling consists of a set of bounding boxes $\mathcal{B}$. Each bounding box $b \in \mathcal{B}$ is associated with a set of features $\mathcal{A}$. Let $a_m \in \mathcal{A}$ denote a feature. As a result, each facade is associated with a matrix $\mathbf{F}$ comprising $|\mathcal{A}|$ columns and $|\mathcal{B}|$ lines, where $|\mathcal{A}|$ and $|\mathcal{B}|$ designate the number of elements in $\mathcal{A}$ and $\mathcal{B}$ respectively.

9.4.2 Features

Each feature needs to comply with the following condition:

1. If an attribute value is an outlier with respect to its distribution, it must be perceived by humans as an outlier.

   In addition, for the natural language description:

   2. The feature must be visually abstractable and translatable in words, such that it is immediately understandable by a human with no training.

To determine the set of features which are suitable for the task, we set up the following user studies. Ten participants were asked to tell what is special about a facade with respect to two other facades in a maximum of 40 characters, for 19 neighbouring triplets of the Zurich dataset. The Zurich dataset is described in details in Section 9.5. In Table 9.1, the free text line presents an overview of the analysis of the 190 descriptions. Note that structural differences are more difficult to grasp instantly and described in words, and thus were therefore very seldom used.

Alternatively, one possible way to tackle our problem would have been to find out image patches which are discriminative between facades. However, this approach comes with issues, as illustrated in Figure 9.5. First, very unusual patches are likely to be caused by occlusions (e.g. cars) or ephemeral (e.g. content of a shop window) rather than stable parts of the facades. Second, image patches can largely vary, while containing the same semantic element, which makes it difficult to estimate if something is usual or special from just a few examples. Finally, it is difficult to put an image patch into words. As a consequence we rely on simple descriptive features, that are more robust to change, can be semantically abstracted and transcribed into natural language.
9.4. What makes a facade unique?

Figure 9.5: Description features - instead of image patches (left: discriminant vs. SIFT patches [Doersch et al. 2012]), we extract size and color attributes (right) which are easily phrased into natural language.

Table 9.1: Overview of the features used to describe facades. The features are listed in the first row and the second row gives an example for each feature. All these features were used to describe facades in the free text experiment, while we selected a subset for the guided labelling experiment and for automatic labelling.

<table>
<thead>
<tr>
<th>Feature</th>
<th>element</th>
<th>structures</th>
<th>colour</th>
<th>material</th>
<th>shape</th>
<th>sizes</th>
<th>position</th>
<th>count</th>
<th>style</th>
<th>use type</th>
<th>text</th>
</tr>
</thead>
<tbody>
<tr>
<td>Example</td>
<td>“door”</td>
<td>“floor”</td>
<td>“red”</td>
<td>“brick”</td>
<td>“pointy”</td>
<td>“tall”</td>
<td>“top right”</td>
<td>“many”</td>
<td>“Gothic”</td>
<td>“office”</td>
<td>“Cafe”</td>
</tr>
<tr>
<td>Free text</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Guided</td>
<td>✓</td>
<td>-</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>-</td>
</tr>
<tr>
<td>Automatic</td>
<td>✓</td>
<td>-</td>
<td>✓</td>
<td>-</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>-</td>
</tr>
</tbody>
</table>

All the features mentioned in the free text line of Table 9.1 comply with the two previously mentioned conditions. However, these features are not all easy to extract from the data we are given. In particular, extracting the material, shape and style would require specific tools, which are not all available or reliable.

We now give more details concerning the features that we implemented in the automatic mining (M) characterization (C) tools.

GIST (M) [Oliva & Torralba 2001b] is a robust descriptor which captures the overall appearance and structure of a scene.

Colour (M, C) Our goal is to produce descriptive words, hence we use the method of van de Weijer et al. [2007] to classify each pixel in the facade image as one of the basic colours. For each facade asset, a histogram of these basic colours constitutes which is used as a feature.
9.4. What makes a facade unique?

Sizes and positions (C) Each bounding box is associated with its height, width, position and area, as well as the logarithm of its aspect ratio. Taking the logarithm of the aspect ratio allows to subtract aspect ratios, which is needed for the outlier score computation.

Number of elements (C) When an architectural element type appears infrequently in a facade, its number of occurrences is a robust discriminative feature.

Asset type (C) was encoded by using a different position within the vector for each type (Wall, door, etc.).

Formally, each facade is associated with a vector $q_M$:

$$q_M = (g, c)$$ (9.1)

where $g$ is the GIST descriptor and $c$ is the colour histogram over the whole facade.

Each asset is associated with a feature vector $q_C$:

$$q_C = (a, w, h, x, y, w \cdot h, ar, c)$$ (9.2)

where $a$ is the asset type, $w$ and $h$ are the width and height of the bounding box, $x$ and $y$ its position, $w \cdot h$ its area, $ar = \log(w) - \log(h)$ the logarithm of its aspect ratio and $c$ its colour histogram.

9.4.3 Outlier detections

Unusualness is formally characterized by outlier detection. The goal is to find the facades and the architectural elements whose feature values are outliers to the distribution of values in other facades in the neighbourhood of the building. Two lines of approaches have been developed when it comes to outlier detection. First, the most straightforward approach is to estimate a probability that a point has been sampled from a distribution. The second approach consists of formulating outlier detection as a clustering problem. More precisely, we want to discover the points which are the most distant from other points in their clusters. A number of tools have been developed for outlier detection [Chandola et al. 2009]. From these, the Local Outlier Factor (LOF) [Breunig et al. 2000] offers the advantage of generality. The Local Outlier Factor (LOF) compares the distance of a point to its nearest neighbours with the average distance of its nearest neighbours between them.

In our case, the amount of data might be very limited when we only consider a local neighbourhood (e.g. many buildings only have one door per facade). This prevents a robust estimate of the parameters of a statistical model. Therefore, we resort to the second approach.
9.4. What makes a facade unique?

Figure 9.6: Interface for the Amazon Mechanical Turk facade unusualness collection. As instruction, participants were asked to “select the words that best describe what is special about this building compared to two others” so that a friend could find it.

For each facade we compute its unusulness as a whole and the unusualness of its assets with respect its neighbours. For this, we compute the distribution of feature values for the whole facade and for each feature \( a_m \) of each asset’s bounding box \( b \) in the neighbourhood of the facade at hand. Consequentially, the LOF scores are computed based on the distribution \( d \):

$$ o = \text{LOF}(q, d) $$

(9.3)

where \( \text{LOF}(q, d) \) is the LOF score for each raw value in \( q \) of the facade or the asset with respect to the value distribution \( d \). The distribution \( d \) is a vector of raw values from the neighbouring facades. \( q \) is the feature vector, \( q_M \) for mining and \( q_C \) for characterization.

9.4.4 Mapping to the human perceptual space

The outlier detection yields the statistically most unusual facades and elements in a given facade. However, “what pops out” [Turakhia & Parikh. 2013], i.e. the humans-perceived unusualness, may differ from the measurement of the LOF score. In other words, different features are of different importance. For elements within a facade, this is the case on two levels: First, between different assets. For instance, doors may be more often used than windows to characterize a facade. Second, between different features. For example,
colours may be perceived as more important than sizes when it comes to describing a facade wall. Our goal is to establish a relation between a LOF score for a facade or a given asset, and the probability that it will be cited. For example, a high LOF score does not mean that the attribute needs to be cited. If small shutters are hardly ever cited, even though they are statistically unusual, we want to learn that they should not be cited.

We now detail how we learn the link between our outlier detection and the perceived importance of features. To obtain the perceived importance of facades and features, we set up two user studies, whose results are discussed in Section 9.5:

**Which facade is the most special?** We collected ground truth data using Amazon Mechanical Turk (AMT). For each dataset, we showed a random subset of 5 facades and asked “which building is the most special?”.

**What is special about a facade?** In this study, participants were asked to say what is special about a facade with respect to its two neighbouring facades using a set of predefined features. For each of the datasets (Zurich, Paris and Graz) we obtained descriptions from more than 50 people, for each facade (See Table 9.3). The used interface is shown in Fig. 9.6. The features used for the guided description are summarized in Table 9.1.

Two different lines of approaches can be used to match the LOF detected outliers with the perceived outliers. First, structured learning approaches, such as Ranking SVM [Joachims 2002] learn how to re-rank the results based on the order given by humans. This method has been used to improve and in particular to personalize search engine results to a user’s preference. Second, a regression analysis would allow to predict a perceived importance score from the LOF score and feature. Regression analysis can also be used for ranking with minor performance difference [Sculley 2010], and offers the advantage of associating a continuous value (which can then be thresholded) and not only a rank. Because of this, we opt for a regression analysis which will allow us to associate a score to each part of the facade.

In essence, we want to predict the unusualness probability $u_{predict}(q_M)$ for mining and $u_{predict}(q_C)$ for characterization, where $q_M$ and $q_C$ are the feature vectors defined in Eq. (9.4.2). We refer to $p_M$ and $p_C$ as the perceived importances, which are used to re-rank the scores from the LOF outliers, as described in the following section. The regression, which is learnt per dataset, helps moving from statistical outlier detection to perceived unusualness.

**Regression training** The regression $r$ is trained using the Decision Tree regression from Breiman [2001], Amit et al. [1996], Liaw & Wiener [2002] to predict the importance $p$:

$$u_{predict}(q, o) = r(q, o)$$ (9.4)

We now detail the vector design and the importance matching.

**Vector design** The tuple $(q, o)$ is the vector containing the LOF score $o$ per facade or asset concatenated with raw feature values $q$. 
Importance matching

We define the perceived importance \( p \), which we want to regress:

\[
p = \frac{n_q}{N}
\]

(9.5)

where \( n_q \) is the number of times a facade image or a feature value/element type combination was selected, \( N \) is the total number of times a facade image was shown or the total number of cited feature value/element type combinations.

A grid search indicated that performance is robust with respect to the number of trees and highest with a moderate number of trees. We keep the number of trees fixed to 200 trees for both regressions and all datasets. For training, we employ a leave-one-out training, where we train on F-1 facades, where F is the number of facades in the dataset, and test on a single facade. Since we are interested in retrieving the most salient facades, the regression is also trained such that it gives more emphasis to the training examples with a high importance. To do so, we applied a linear weight equal to \( p \) to minimize the error of the of the facade images or features and assets which were perceived as most important.

Fig. 9.7 shows the effect of the mapping within the facade, which improves our result. We notice that statistical unusualness needs to be re-mapped in order to account for what humans perceive as important. The large difference is to be expected, as the statistical degree of different attributes cannot be directly compared without re-weighing according to human perception. In the example, the window at the top is statistically the most unusual element, but is not perceived so by humans.

9.4.5 Clustering

Clustering is needed for natural language characterization, in order to group elements when referring to them. For example, if all windows are large, they should be referred to as a single entity (“the large windows”) rather than picking one of them (“the large window at the top-left”). Therefore, the facade elements are clustered according to their colours, widths and heights using Mean-shift [Fukunaga & Hostetler 1975, Comaniciu & Meer 2002b]. The advantage of Mean-shift is that the number of clusters or variances does not need to be known beforehand, and only the bandwidth needs to be set. In our experiments, we set the bandwidth to 17, after visual inspection of a few examples.

9.5 Experimental Evaluation

For the experiments, we use three different datasets, comprising 50, 47 and 20 facade images respectively. Each dataset corresponds to a single street or area, meaning the styles are expected to be similar within each dataset. Each ortho-normal image is associated with
9.5. Experimental Evaluation

Figure 9.7: Heatmap representation of measured unusualness. Left: facade image, Middle: using LOF results, Right: after mapping according to human perception. Mapping places the emphasis on unusual colours (the shutters are bright blue) and assets which are cited more in the human perception study, e.g. shops. Note the very large difference between the two heatmaps: statistical unusualness needs to be re-mapped to match with human perception of unusualness.

Figure 9.8: Map highlighting the most unusual building around an itinerary in Graz.

a ground truth labelling. Labels are Door, Wall, Sky, Window, Shop, Balcony, Roof and sometimes Shutters.
Table 9.2: Summary of the facade mining study results. RF: Random Forest, Corr: Pearson correlation. A correlation of 1.0 would mean that we can perfectly predict the distribution of the responses of humans for each facade.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Zurich20</th>
<th>Graz50</th>
<th>ECP2011</th>
</tr>
</thead>
<tbody>
<tr>
<td>#facades</td>
<td>20</td>
<td>50</td>
<td>47</td>
</tr>
<tr>
<td>#subjects</td>
<td>20</td>
<td>55</td>
<td>53</td>
</tr>
<tr>
<td>#annotations</td>
<td>400</td>
<td>1100</td>
<td>1060</td>
</tr>
<tr>
<td>Corr. (re-ranked RF)</td>
<td>0.60</td>
<td>0.40</td>
<td>0.17</td>
</tr>
</tbody>
</table>

The Graz50 and ECP2011 datasets have originally been designed to assess the quality of facade parsing. In addition, we introduce the Zurich20 dataset. The Graz50 dataset [Riemenschneider et al. 2012] consists of 50 labelled rectified pictures from Graz. The Zurich20 dataset comprises 20 labelled rectified pictures from Zurich. The ECP2011 dataset [Teboul et al. 2011b] comprises 104 labelled rectified pictures from rue Monge in Paris. Because of the very large effort to label facade oddities, we kept all the even numbered ones (47 facades, a complete side of the rue Monge) in the dataset. In all experiments, the data was split into training and testing in a leave-one-out fashion.

Overall, 423 participants took part in the AMT experiments. 62 % and 38 % of the participants were female and male subjects respectively. Their ages ranged from 18 to 65, while the average age was 31.9.

9.5.1 Special facade mining

The unusual facades are mined using (9.4) and the method presented in Section 9.4. From there, we generate a map showing the most unusual buildings. We gathered ground truth data for Graz, Paris and Zurich. Table 9.2 gives a summary of the collected data and the performance of the regression. Note that the regression does not need to be very precise for low scored facades, as in the end the top results are displayed. Therefore, we also report the TopK scores for each dataset in Fig. 9.13. The TopK score quantifies how well the computer-ranked top K images agree with the human ranking [Grabner et al. 2013]. The score ranges from 0 to 1, where a perfect agreement of the two rankings leads to a score equal to 1.

As can be seen, the performance of our mining method is good for the Graz50 and Zurich20 datasets, while performance is less on the ECP2011 dataset. The ECP2011 difficult is more difficult, as the standard deviation of the building importance scores change less for ECP2011 (0.0816) compared to Graz50 and Zurich20 (0.1266 and 0.1650). In other words, the very regular ECP2011 dataset exhibits a smaller range of importances (similar facades) as compared to the other datasets. Consequently, it is not surprising that the ECP2011 dataset makes it a very challenging task to extract special buildings. Fig. 9.9
9.5. Experimental Evaluation

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Zurich20</th>
<th>Graz50</th>
<th>ECP2011</th>
</tr>
</thead>
<tbody>
<tr>
<td>#facades</td>
<td>20</td>
<td>50</td>
<td>47</td>
</tr>
<tr>
<td># subjects</td>
<td>87</td>
<td>53</td>
<td>51</td>
</tr>
<tr>
<td># annotations</td>
<td>1473</td>
<td>2331</td>
<td>2100</td>
</tr>
<tr>
<td>Corr. (LOF)</td>
<td>0.25</td>
<td>0.13</td>
<td>0.07</td>
</tr>
<tr>
<td>Corr. (re-ranked RF)</td>
<td>0.52</td>
<td>0.72</td>
<td>0.57</td>
</tr>
</tbody>
</table>

Table 9.3: Summary of the facade characterization study results. Corr: correlation, LOF: Local Outlier Factor, RF: Random Forest

shows the most and the least special buildings for the Graz50 dataset \(^1\). Fig. 9.8 shows an example of the resulting map.

9.5.2 Perceptual study for characterization

**Experimental setup** The experiment was presented as shown in Figure 9.6. Participants were instructed: “In the following page, you will be presented three pictures of buildings. Imagine that you are meeting with a friend in a year’s time in front of the building highlighted in red. Your task is to help him find that building. To do so, please select the words that best describe what is special about this building compared to the two others.”.

**Results** Fig. 9.12 presents statistics about the Zurich dataset. In our experiments, we considered the two neighbouring facades. We notice that the colour plays an important role in identifying unusual elements to discriminate between facades. In contrast, the Local Outlier Factor (LOF) gives more importance to the size differences. The regression reestablishes the dominance of colour over size. Also, sizes are not used equally by humans: large, tall and wide elements constitute a large majority of the sizes which are stated. This can be explained by the fact that it is easier to identify a large element than a smaller one. Finally, rare architectural elements are relatively more important. For example, although doors are much rarer than windows, they are cited relatively frequently.

Table 9.3 gives a summary of our performance results. The correlation refers to the correlation between the predicted importance scores and the importance scores measured in the experiments. The Pearson correlation is much higher using after re-ranking than using the normalized LOF scores. This explains the large difference in the heatmaps in Fig. 9.7.

9.5.3 Facade characterization

For each facade, we obtain both an importance map highlighting the main characteristics and a natural language description. These could be used as input to methods such as [Cole et al. 2006, Kopf et al. 2013] to emphasize the most discriminative parts.

\(^1\)all results are in the supplementary material.
The natural language description can be turned into speech and are useful in case one does not want to look at a screen or a map, for instance when driving a car. Figures 9.10 and 9.11 show some exemplar results of our generated sentences.

**Figure 9.9:** The most salient (top) and least salient (bottom) facades in Graz50 as predicted by us.

**Figure 9.10:** Our automatic description of the facade in the middle: “The facade with tall blue shop and wide brown shop.” On right, the heatmap of the most unusual features.

### 9.6 Conclusion

Facades are at the heart of the urban landscape. The next step for digital mapping has arrived: understanding a facade like a human. In this work, we have introduced a method to analyse facades by mining their atypical features. This solves two fundamental problems in digital mapping. First, we can produce more readable maps even if no famous landmark is present in the area. Second, the descriptions are valuable for other media, such as
9.6. Conclusion

Figure 9.11: Our automatic description of the facade in the middle: “The facade with green wall and wide red windows.” On right, the heatmap again.

audio or text. Last but not least, this work illustrates a generic method to automatically learn where to place emphasis, which is a crucial issue in visualization. The resulting data tells us which buildings are considered as important to be shown on a map and what is important within these buildings. This constitutes a valuable input to methods such as the ones of Cole et al. [2006], Kopf et al. [2013] which need this information for attracting attention or rescaling. For future work, we plan to add more features related to material and shape, as well as using city-scale datasets.

Figure 9.12: Zurich20 Analysis. Columns: Asset, feature type, colour and size usage. The vertical axis in the histograms refers to the number of times an asset or attribute was cited. Rows: Top: study results, Middle: LOF, Bottom: After regression. The regression helps to give descriptions closer to the human-given annotations. Note that the material and shape are not used in the automatic unusualness inference.
Figure 9.13: TopK scores obtained for facade mining with our method (blue) against a randomness-based baseline (red) for Graz50 (left), ECP2011 (middle) and Zurich20 (right). The TopK score quantifies the agreement between the human and automatic rankings for the top K items. A perfect agreement gives a score of 1.

Figure 9.14: TopK scores obtained for facade characterization with our method (blue) against a randomness-based baseline (red) for Graz50 (left), ECP2011 (middle) and Zurich20 (right).
Conclusion

In this work, we presented methods to construct semantically-rich city models from pictures. To begin, we examined the diverse needs associated with city models. Then, we presented procedural modelling as suitable models for architecture.

From there, we introduced an approach to generating procedural models from pictures. This starts with determining the styles of the buildings to be dealt with. After that, we presented a 2D method based on weak architectural principles for facade segmentation, as well as a method to dramatically speedup and improve 3D facade labelling. Then, we distinguished between two cases: when a style grammar is available or not. In the former case, we demonstrated a parameter inference method on Doric temples. For the latter, we introduced a novel approach to grammatical inference from a set of facade labellings. The resulting grammars are used for different applications: compression, facade synthesis, comparison and retrieval. Finally, we show that new applications emerge from semantically-rich models: we introduce a novel method to enhance navigation, by mining special facades and characterizing them.

We conclude this thesis by some remarks and suggestions for future directions.

10.1 The logic of architecture

An interesting finding from Chapters 8 and 9 is that architecture is both made from redundancies and exceptions. From a computer modelling point of view, both should be taken into consideration. Finding and exploiting redundancies results in the capacity to build an abstract representation of a building. Abstraction yields two benefits: an understanding of architecture and data compression. Exceptions are the signature of buildings. They give a unique identity to the places where we live and work. Although one could think irregularities only serve the purpose of art, exceptions are also useful to help us navigate. Therefore, it is important to preserve these in a computer model.
10.2 Grammars as compression-based models

As we have seen in Chapter 8, grammars can be used as means of compression. Grammar-based compression algorithms generate a set of rules which represent the string at hand. Language is primarily seen as a carrier of meaning. It is interesting to relate meaning to compression. In the case of Haussmanian and Gründerzeit architectures, we realized that the minimum description length is also a good capture of a logic, i.e. meaning. Language is an efficient way to reduce the amount of data to be processed while not always requiring decompression to be processed. Therefore, using language as an efficient compression method could decrease memory and processing power requirements. Words give abstractions, which are in fact replacing multiple instances by a single entity. For example, a door is an abstract concept and is mostly characterized as being a moveable object which can allow or restrict access to an aperture. The memory and processing reduction could not only be valid for a computer, but also for a human brain. Beyond Occam’s razor (as proposed by Carrascosa et al. [2010]), we can hypothesize that, this could be the reason why what we perceive as most meaningful coincides with a minimum description length.

10.3 Outlook

First and foremost, the most obvious continuation of this work and others in the field of urban modelling is to combine them into a hollistic approach. In particular, semantic labelling and grammatical inference could be carried out in the 3D space from start to finish. Next, it is of prime importance to collect large-scale datasets. Current public datasets are of limited size and mostly 2D, making it hard to predict generalizability of training and use 3D information.

On the model side, using an alternative to grammars could prove very useful for model fitting. As a matter of fact, the very many possibilities to describe a building are reduced to a subset of grammar rules. Using these for model fitting then constraints the search space, which is a benefit, but makes exceptions difficult to handle. It would be desirable to allow for the description of more sophisticated relations between architectural elements in a facade. Inverse procedural modelling involves dealing with gigantic search spaces. Methods for efficient search are a bottleneck of any top-down approach. The discovery and the development of efficient search methods could greatly benefit the field.

Finally, we think grammars could be used for other fields involving custom design and fitting. Digital fabrication recently made a leap forward, e.g. with the democratization of 3D printing. Nevertheless, design remains a difficult, manual task. In parallel, acquisition of measurements via cameras is also a hot topic. Automating the design of objects, from clothes to furnitures, is a promising application of research on grammars.
Bibliography


Chomsky, N. (1956). Three models for the description of language. *Information Theory, IRE Transactions on*. 2.4.3, 2.4.3


Evers, B., & Thoenes, C. (2003). *Architectural theory: from the Renaissance to the present: 89 essays on 117 treatises*. Taschen. 2.4.1


Gould, S., Russakovsky, O., Goodfellow, I., Baumstarck, P., Ng, A. Y., & Koller, D. (2009b). The stair vision library (v2.2). http://ai.stanford.edu/~sgould/svl. 5.4


Jie, L. (1103). Yingzao Fashi (Treatise on Architectural Methods or State Building Standards). 2.4.1

Joachims, T. (2002). Optimizing search engines using clickthrough data. In ACM SIGKDD int. conf. on Knowledge discovery and data mining. 9.4.4


Mikolajczyk, K., & Schmid, C. (2004). Scale & affine invariant interest point detectors. *IJCV.* 7.3.4


Müller, P. (2010). *Procedural modeling of buildings*. PhD Thesis, ETH Zurich. 1.6, 2.4, 2.5.2, 8.1


Palladio, A., & Ware, I. (1965). *The four books of architecture*. Dover publications New York. 2.4.1


Quack, T., Leibe, B., & Van Gool, L. (2008). World-scale mining of objects and events from community photo collections. In *CIVR*, CIVR ’08. 2.2.1, 4.1, 7.1


Semper, G., Mallgrave, H. F., Herrmann, W., Rykwert, J., & Semper, G. (1989). *The four elements of architecture and other writings*. Cambridge University Press Cambridge. 2.4.1


Stiny, G., & Gips, J. (1971). Shape grammars and the generative specification of painting and sculpture. In *IFIP Congress (2)*. 2.4.3


Szeliski, R. (2010). *Computer vision: algorithms and applications*. Springer. 2.1


Tzonis, A., & Lefaivre, L. (1986). *Classical architecture: The poetics of order*. MIT Press. 2.4.1


Zhao, P., Fang, T., Xiao, J., Zhang, H., Zhao, Q., & Quan, L. (2010). Rectilinear parsing of architecture in urban environment. In CVPR. 5.2, 7.2

List of Publications

Refereed Conference Proceedings


Curriculum Vitae

Personal Data

Name Julien Weissenberg
Date of birth 11th July 1986
Place of birth Paris, France
Citizenship French

Education

2010 – 2014 ETH Zurich, Computer Vision Laboratory, Switzerland
Doctoral studies
2008 – 2009 Imperial College London, London
MSc in Advanced Computing
2006 – 2009 ENSIE, Evry, France
MSc in Computer Engineering, ENSIE, Evry, France
Graduation with the degree Dipl.-Ing.

Work Experience

June – August 2008 Medicasoft, Paris, France
Internship. Design and implementation of a very efficient reporting software for echo-cardiologists.
June – August 2007 Niji, Rennes, France
Internship. Design and implementation of a Web TV player. Feasibility study and architecture of a web Video-on-Demand platform.