Universität der Bundeswehr München

Fakultät für Informatik

Facade Interpretation from Images

M.Sc. Kujtim Rahmani

Vollständiger Abdruck der von der Fakultät für Informatik der Universität der Bundeswehr München zur Erlangung des akademischen Grades eines

Doktors der Naturwissenschaften (Dr. rer. nat.)

genehmigten Dissertation.

Gutachter:

1. Prof. Dr.-Ing. Helmut Mayer
2. Prof. Dr.-Ing. Olaf Hellwich (TU Berlin)

Die Dissertation wurde am 12.12.2019 bei der Universität der Bundeswehr München eingereicht und durch die Fakultät für Informatik am 19.03.2020 angenommen. Die mündliche Prüfung fand am 29.06.2020 statt.

Kurzfassung

Künstliche Intelligenz (KI) hat jeden Bereich unseres Lebens erreicht. Darüber hinaus werden dreidimensionale (3D) Modelle zum Hauptthema der Technologie. Diese Arbeit widmet sich dem Einsatz von KI-Techniken zur Fassadensegmentierung, die die 3D-Rekonstruktion von Gebäudefassaden erleichtern.

Zunächst haben wir eine Pipeline für die Segmentierung der gesamten Fassade entwickelt, die die Klassen Balkon, Dach, Fenster, Geschäft, Himmel, Tür und Wand unterscheidet. Unsere Hybrid-Pipeline besteht aus traditionellen und modernen Machine Learning Ansätzen. Der Structured Random Forest basiert auf der Merkmalsextraktion ist und unser Objektdetektor verwendet einen Deep Learning Ansatz und die endgültige Teil Modellanpassungen verwendet dynamische Programmierung.

Zweitens haben wir ein System zur Fensterbegrenzung aufgebaut, das jedes Fenster in Fensterrahmen, (Mittel) Fensterpfosten und Querbalken segmentiert und uns eine vollständige Beschreibung eines Fensters gibt. Ähnlich wie beim vorherigen verwendet das System eine Pipeline verwendet, die aus einem Objektdetektor, einer semantischen Segmentierung mit Deep Learning und Modellanpassung auf der geometrischen Form und des Aussehens der Fenster basiert. Für die semantische Segmentierung wird eine Architektur gewählt, die dünne Elemente in einem Bild erfassen kann.

Zusätzlich werden auch dynamische Programmierung und Informationsfusion verwendet, um das endgültige Segmentierungsergebnis zu erzeugen. Wir sind die Ersten, die ein System zur Segmentierung von Fensterpfosten und Querbalken sowie einen hochauflösenden Datensatz zum Training und Auswertung des Detektors für Fensterpfosten und Querbalken aufgebaut haben. Wir haben die Relevanz von Merkmalen und die Optimierungsfunktionen der Knoten in den Decision Trees (Entscheidungsbäumen) bewertet.

Zusätzlich haben wir experimentell gezeigt, wie sich die Merkmalsbereinigung auf die Endergebnisse auswirkt und welche Merkmale wichtig sind.

Schließlich haben wir gezeigt, dass wir ein zuverlässiges System für die Fassadensegmentierung aufgebaut haben, das als Quelle für die Erstellung qualitativ hochwertiger 3D-Gebäudemodelle verwendet werden kann.

Abstract

Artificial Intelligence (AI) has reached every area in our life. Furthermore, three-dimensional (3D) models are becoming a main topic of technology. This thesis is devoted to using AI techniques for facade segmentation which aids the 3D reconstruction of the facades of buildings. First, we have developed a pipeline for the segmentation of whole facade distinguishing the classes balcony, door, roof, shop, sky, wall and window. Our hybrid pipeline consists of traditional and modern machine learning approaches. The Structured Random Forest is based on feature extraction, our object detector employs a deep learning approach and the final part model fitting uses dynamic programming. The pipeline is evaluated on several datasets, it performs better than or on a par with currently published methods and it is the state of the art for small datasets.

Second, we have built a system for window delineation segmenting every window in window frame, mullions and transoms, giving us a complete description of a window. Similarly to the previous one, the system uses a pipeline consisting of an object detector, semantic segmentation employing deep learning and model fitting based on the geometric shape and appearance of the windows. For semantic segmentation an architecture is chosen, which can capture thin elements in an image. Additionally, also dynamic programming and information fusion are used to create the final segmentation result. We are the first that have built a system for the segmentation of window frame, transoms and mullions as well as a high-resolution dataset for training and evaluation of the detector for transoms and mullions.

We have evaluated the relevance of features and the optimization functions of the nodes in the decision trees. Additionally, we have shown experimentally the effect of feature cleaning on the final results as well as what features are important.

Finally, we have demonstrated that we have built a reliable system for facade segmentation which can be used as a source for building high–quality 3D building models.

To my family

Acknowledgments

First and foremost I want to thank my supervisor Prof. Helmut Mayer for giving me the opportunity and trust to pursue my PhD. He followed every step of my progress and he always guided and helped me when difficulties appeared. His scientific and human qualities helped me to evolve my scientific and personal thinking.

Secondly, I want to thank my office-mate Dr. Hai Huang who supported me in my research and from whom I learnt a lot from the discussions that we had.

Special thanks go to my colleagues at the Institute for Applied Computer Science who were helpful during my three and a half years research there.

Furthermore, I wish to thank my parents and my brother who are supporting me in every step and sacrificing themselves from the first day that I came in this world. Only their support allowed me to obtain the PhD degree. FALEMINDERIT!

Finally, I want to thank my wife Ardita; her love, encouragement and motivation made this work possible. During the time of writing of this thesis my son Endrit and my daughter Diellza were a big motivation and I dedicate this thesis to them.

Contents

Chapter 1

Introduction

Facade segmentation is an important but also challenging task for urban scene understanding and 3D city modeling with applications, e.g., in city planning and the film and game industry. It is multidisciplinary, concerned with problems in artificial intelligence, computer vision, image processing, civil engineering, architecture and computer science. This thesis is focused on the computer vision and image processing part of the problem. Using techniques from artificial intelligence the facade appearance is modeled, objects on it are detected and their location is determined.

1.1 Motivation

Semantic segmentation (also: interpretation) provides the most detailed and accurate information about a facade since it labels each pixel with a class label. Thus, we know for each pixel to which object it belongs and we have comprehensive information about the image.

Nowadays, visualization and 3D reconstruction of cities are important topics in several research areas as well as in various industries and facade interpretation is an important component. 3D city models are used on our computers, cars, virtual reality devices and mobile phones as part of navigation tools and maps. Furthermore, 3D city models are important for cultural heritage, city planning and building conservation.

For all these problems facade segmentation is a tool that helps to reduce the production time and improve quality. Moreover, in change detection for facades, segmentation is the most crucial part of the problem. First, at least

two images taken at a different times are segmented and then compared to determine changes of objects.

Two of the most profitable entertainment industries, game and film, use city modeling when creating games and movies. Another usage of facade models is in civil engineering and architecture where building models are used for planning.

The high importance of facade information motivates us to concern ourselves with facade segmentation in order to aid 3D building reconstruction for creating high-quality 3D models in a short amount of time.

During 3D reconstruction, window reflections as well as other obstacles such as occlusion hamper the reconstruction and one, thus, does not obtain the desired model quality. We believe that semantic segmentation of the images will help to overcome this obstacle. Thus, our aim is to create a system for facade segmentation which supports 3D building reconstruction and produces high quality, almost noise free labeled facade images.

To achieve this goal, we go one step further than previously published work: we do not only detect the windows but also segment them in the smaller parts window frame, transoms, mullions and window area (glass), creating high-quality window information. This also helps to analyze the pattern of the windows on a wall since most of them have the same appearance.

For facade segmentation various data such as Red, Green, Blue (RGB) images and depth information from LIDAR (Light Detection And Ranging) or image matching can be used. In our work, we mainly employed already published datasets consisting of RGB channels of the image projected on the facade plane. Additionally, we have created our own dataset (Fig. 1.1) in which the depth channel is estimated by a 3D reconstruction method based on image matching (Ley and Hellwich, 2016). This is the basis for an envisioned iterative data exchange pipeline between 3D reconstruction and facade interpretation.

Usually, facade interpretation algorithms are based on the assumption of specific facade structures. Some researchers assume that facades have a specific "morphology" and approach the problem using grammars performing hierarchical segmentation dividing (big) facade regions recursively into smaller objects. These approaches are called *top-down* methods and they are based on a set of rules to segment facades.

Other, so-called *bottom-up* approaches, are based on weak assumptions about the global facade structure or hierarchical ordering. They only assume that the objects on the facade have a limited number of variations and learn their representative appearance and shapes starting from a pixel, a superpixel or a small patch finally integrating them recursively into bigger regions. Our approach is based on the latter principle since bottom-up approaches have been empirically proven to perform better both in terms of run time (speed) and accuracy especially when dealing with non-regular facade structures.

(a) Image **(b)** Depth Image **(c)** Annotated Image

Figure 1.1: Dataset for detailed facade segmentation. Left: RGB image, center: (Pseudo-) Depth map, right: Annotated image including transoms and mullions

1.2 Problem Statement and Objectives

The smallest image element is the pixel, and our goal is to provide a class label for each pixel of a facade image.

We, thus perform pixel-wise labeling of the facade image defining the boundaries of the objects on the facade. We detect balconies, doors, shops, sky, roofs, wall and windows.

Moreover, we are interested in a more detailed description of the most prominent facade object, namely windows. We want to delineate the window frame and to segment the transoms and mullions. This additionally aids the 3D reconstruction process since windows are the most complex elements due to the reflections on the glass and their small and possibly complicated components. Our aim is to build a window delineation algorithm that is accurate enough to support 3D reconstruction as well as to precisely describe the window frame.

Since no publicly available datasets exist for this problem, we have created a dataset with annotated facade images including window frame, transoms and mullions for which also depth information is available (Fig. 1.1). With the latter, we want to evaluate the influence of depth features generated by 3D reconstruction on the accuracy of the results.

Overall, the main goal of this thesis is the description of a high-quality bottom-up system for facade segmentation which produces facade objects delineated with high quality, especially windows which are an important source of information for 3D reconstruction and change detection.

1.3 Challenges

Facade segmentation is a complex computer vision problem due to the variety of architectural styles of buildings with different appearance, shape and layout. For example, in the Hausmannian architecture, almost all windows have a French balcony. Gothic style buildings have a specific type of windows and their frequency on the facade and their dimensions differ from Hausmannian buildings.

Other difficulties stem from occlusions in front of facades and weather conditions. In spring and in summer trees' leaves occlude the facade wall and flowers cover balconies and windows.

Another problem which is arising during dataset creation is how to distinguish and define the facade objects. For example, a balcony consisting of transparent glass can be confused with either the wall, a window or a "balcony door". Furthermore, the viewing angle of the camera during image acquisition

affects the pixel labels especially for objects on the upper part of the image. Particularly, in terrestrial facade images, balconies occlude the windows due to the viewing angle.

We have built several object detectors for windows, doors, balconies and shops and a segmentation algorithm that labels each pixel. In this thesis, all algorithms are suitable for images. Additionally, we have used features derived from 3D data for a particular dataset. A more detailed description of the challenges is given in Chapter 6 on experiments and evaluation.

1.4 Outline

This thesis gives a complete account of the developed building facade segmentation system. For better comprehensibility, we employ examples and use cases. The thesis is organized as follows:

Theoretical Background. This chapter introduces the theoretical background on feature extraction as well as new adapted features based on statistical properties of specific regions of facade images.

A supervised machine learning algorithm is described which employs a Structured Random Forest as the main technique for our pipeline.

Additionally, theoretical background of object detectors especially based on pyramid features is given. The latter methodology of feature extraction was found very important for window delineation. Furthermore, an introduction of the theory behind deep learning-based semantic segmentation methods is presented.

Related Work. The third chapter presents related work on facade object detectors and facade segmentation methods. It is structured according to the methodologies that are used for detection and segmentation.

Facade Segmentation Based on Structured Random Forests. This chapter gives a detailed account of the developed system for facade segmentation. The latter consists of several components including traditional machine learning algorithms as well as domain specific feature extraction, deep learning-based object detectors as well ass their integration in the segmentation algorithm and, finally, a greedy rectangular fitting approach. Moreover, we present several domain-specific methodologies which produce better results.

Window Refinement. In this chapter our deep learning-based method for detecting transoms and mullions in the window image as well as window frame refinement using the grid structure of windows is presented. Since we are one of the first pursuing window refinement, we have built a dataset and annotated transoms and mullions.

Experiments and Evaluation. The results of our pipelines are presented in the sixth chapter. Our approach consists of several components for each of which we show the quantitative and qualitative influence. Additionally, we compare our method with the currently published methods showing that our methods outperform or are on par with the state of the art.

Conclusion. Finally, we list our contributions in the field of facade segmentation, summarize our work and present potential areas of future research we have discovered or where we have been challenged during our research.

Chapter 2

Theoretical Background

This chapter presents the theoretical background for this thesis. The focus is on the two computer vision problems object detection and image segmentation. Both are solved using machine learning combined with image processing.

The success of object detection and semantic segmentation depends on the features extracted from the data and the architecture of the machine learning algorithm. Thus, first several feature extraction methods are presented as well as their influence on object detection and facade segmentation. Then, supervised machine learning algorithms, in the form of decision trees, random forests and Structured Random Forests (SRF) are introduced, and their adaptation to computer vision problems is discussed.

Finally, object detection and segmentation theory as well as approaches are described together with their development over time. The approaches are discussed in terms of their contributions in terms of quality, accuracy and runtime efficiency.

2.1 Feature Extraction

Building a machine learning algorithm based on Decision Trees or Support Vector Machines (SVM) usually requires features extracted from the image. These features express some characteristics of an image pixel, a region or the complete image. Since the pixel values are usually stored in eight bit memory per channel, the features are often mapped into the range 0 to 255.

Color channels are very basic features. They capture the distribution of the intensity of color in the image. Digital images taken by most digital cameras consist of the three color channels **R**ed, **G**reen and **B**lue, known as

RGB-channels. Each pixel contains three intensity values and uses for each of the channels eight bits of memory, resulting in 24 bits of memory overall. Additionally to the RGB representation there exist further color spaces such as the CMYK (**C**yan, **M**agenta, **Y**ellow and blac**K**) or CIELab color space. Color channel features, for example the CMYK channels (Equation 2.1) can be derived from the RGB color space. Furthermore, sensors exist that can capture special channels, such as infrared or even depth.

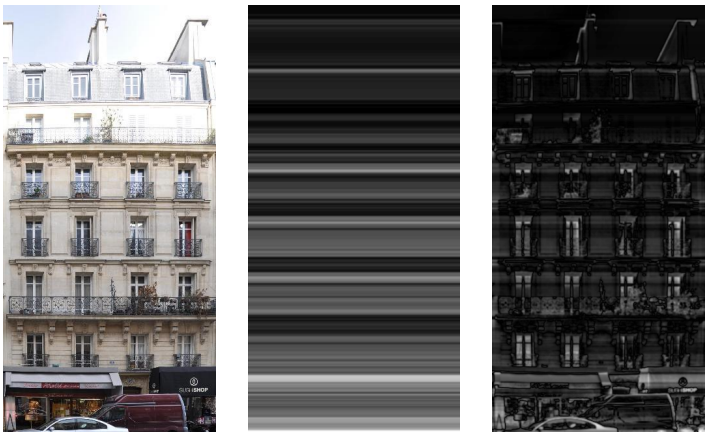
$$\begin{gathered} R' = \frac{R}{255} \qquad G' = \frac{G}{255} \qquad B' = \frac{B}{255} \\ K = 1 - max\{R', G', B'\} \\ C = \frac{1 - R' - K}{1 - K} \quad M = \frac{1 - G' - K}{1 - K} \quad Y = \frac{1 - B' - K}{1 - K} \end{gathered} \tag{2.1}$$

Image statistics features represent the distribution and statistical properties of the pixels in a predefined image region. Depending on the problem, different statistical properties based on rows, columns, rectangles or other patterns can be derived.

As we are concerned with rectified, vertically aligned facade images, we are particularly interested in statistical features for the image row. We compute the standard deviation, median and mean for the row as well as the absolute difference between the pixel value and the row mean and row median for the RGB channels. These features contain information about the (non-)existence of objects on the wall: rows that contain windows, doors or balconies have a high standard deviation while rows that just consist of plain wall exhibit a low standard deviation (Fig. 2.1).

Another important feature is the difference between the median and the pixel value. The median of the row corresponds in most cases to a wall pixel, since the wall covers more than 50% of the facade. The difference (Fig. 2.1c) provides information about which pixels are different from the wall, thus indicating the existence of a facade object.

Local Binary Patterns (LBP) features are used to identify and distinguish different image textures. They are extracted for each pixel $(x_{i,j})$ (Equation 2.2) from a single channel. LBP are built by determining a pattern in a region consisting of n pixels $(x_{i1,j1}, x_{i2,j2}, ...x_{in,jn})$, with the two argument LBP function $f(x_{i,j}, x_{ik,jk})$ with $k \in \{1, 2, ..n\}$, producing n values. Based on



(a) **(b)** **(c)**

Figure 2.1: Row statistics features. (a) Input image, (b) Standard deviation of each row of the Red channel and (c) Difference of the Red channel pixel value and the row median

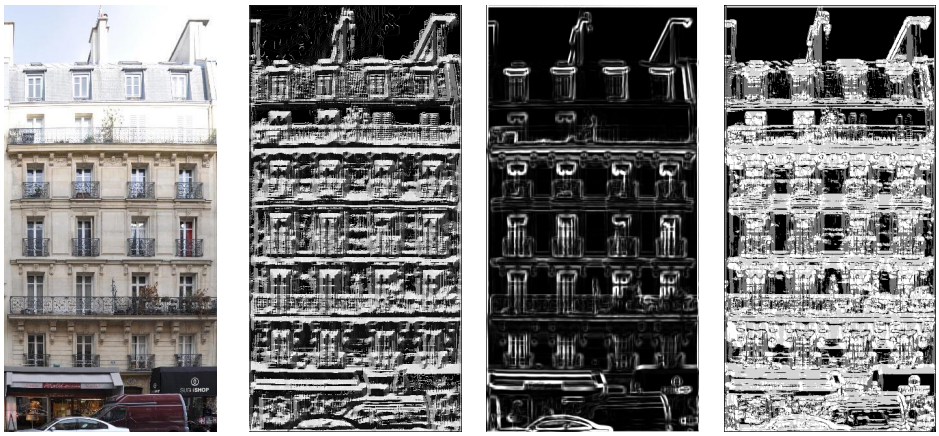
the n output values, the function ($\oplus$) is computed, deriving the LBP pixel value (Equation 2.2).

$$pixel\_value(x_{i,j}) = \oplus(f(x_{i,j}, x_{i1,j1}), f(x_{i,j}, x_{i2,j2}), ...f(x_{i,j}, x_{in,jn})) \quad (2.2)$$

Most LBP functions $f(x_{i,j}, x_{ik,jk})$ are binary functions that produce the two values zero and one, usually by comparing two pixel values. Thus, a pattern in a neighborhood consisting of eight pixels together with an $n = 8$ argument function ($\oplus$) outputs a value from zero to 255 (Equation 2.3).

$$pixel\_value(x_{i,j}) = \sum_{k=0}^{7} 2^k f(x_{i,j}, x_{ik,jk}) \quad (2.3)$$

LBP can be used to separate an object from the background and to distinguish objects that have different textures or structures. For this, different types of the LBP feature extractors have been developed. E.g., for the Center-Symmetric Local Binary Patterns (CS-LBP) (Heikkilä et al., 2006), the feature value is calculated from a binary relationship between symmetric pixels. In addition to the described and shown patterns (Fig. 2.2), more than twenty different LBP for distinguishing specific textures, object tracking and



(a) (b) (c) (d)

Figure 2.2: Different Local Binary Patterns feature extractors. (a) Input image, (b) Oriented Local Binary Patterns (Ojala et al., 2001), (c) Variance Local Binary Patterns (Ojala et al., 2002), (d) BackGround Local Binary Pattern (Davarpanah et al., 2016)

background segmentation have been developed (Silva et al., 2015). Since some of them are not suitable for facade segmentation, only ten of them are used.

TextonBoost features introduced by Shotton et al. (2006) are based on Gaussian filters. These features capture the local appearance and shape of the objects (Fig. 2.3). They are constructed from several Gaussian distributions and orders of image gradient leading to seventeen different filters.

The authors show the power of the TextonBoost features when used in a Conditional Random Field (CRF) for pattern modeling and recognition as well as for capturing spatial and contextual information. Additionally, in the presented application (Shotton et al., 2006), the TextonBoost features delineate the borders between objects very well.

Histogram of Oriented Gradients (HOG) (Dalal and Triggs, 2005) is one of the most frequently used feature extractors for object detection. Dalal and Triggs (2005) first used HOG for pedestrian detection. As they outperformed previous approaches, HOG features were widely used and adapted to other objects.

The HOG descriptor creates a histogram from the gradients of the image pixels. First, the image is divided by a grid, where each cell contains a number

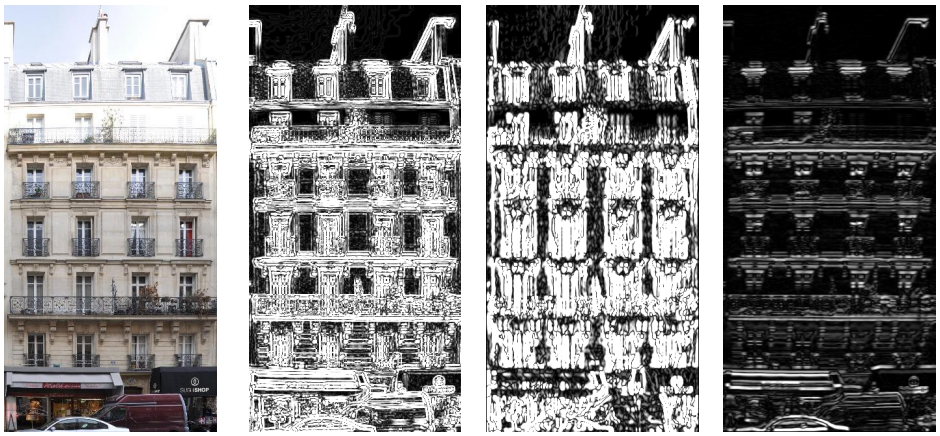


Figure 2.3: (a) Input image, (b),(c),(d) Three out of seventeen TextonBoost features

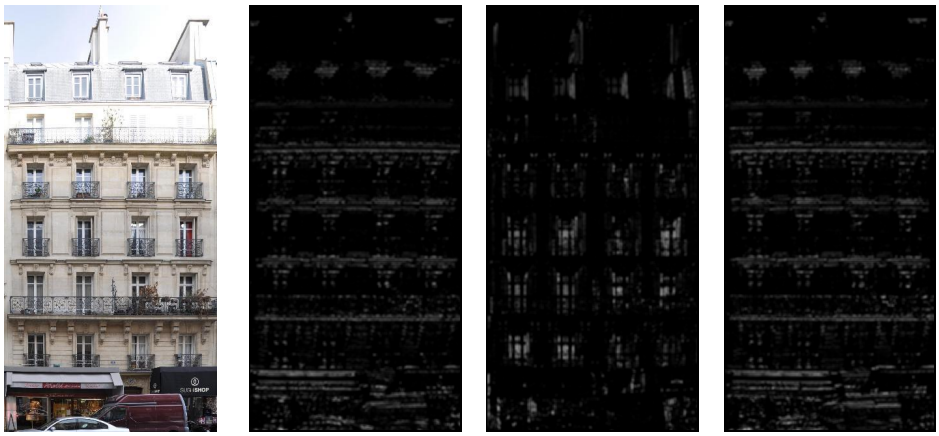
of pixels. For each pixel, the gradients in vertical and horizontal direction are computed as well as the orientations of the gradients. Gradient orientations are distributed into bins. Usually, orientations are scaled from 0-360^0, put into nine bins and converted correspondingly to nine separate image features (Fig. 2.4). HOG captures local gradient orientation, is not sensitive to color changes and can represent different textures.

Descriptive features play a crucial role in building high quality object detectors and segmentation algorithms, since they are the basic information for the machine learning algorithms to distinguish the classes.

The second necessary part for building a high quality object detector or segmentation algorithm are machine learning algorithms. They decide whether an object appears in an image or how to label a pixel or a patch based on the extracted features. Thus, in the following sections several supervised machine learning algorithms are introduced.

2.2 Machine Learning Algorithms

Supervised machine learning is used to build predictive algorithms based on labeled datasets. The algorithms learn the "relationship" between the extracted features and the target classes. Commonly used machine learning algorithms are Logistic Regression, k-Nearest Neighbors, Decision Trees, Support Vector



(a) (b) (c) (d)

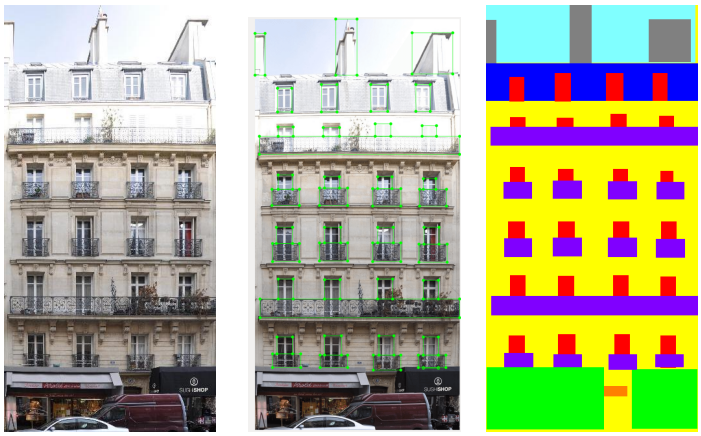
Figure 2.4: HOG features. (a) Input image, (b),(c),(d) Three out of nine HOG features

Machine (SVM) and Artificial Neural Networks. Some of them, which are useful for the application of this thesis, are described in the following sections.

For supervised machine learning, annotated (labeled) data with their ground truth class are needed. A part of the dataset, the training set, is used to learn and train the relationship between features and classes. Another part, the validation set, is employed to validate the algorithm's parameters, to fine tune them and, thus, to improve the algorithm's performance. Training and fine tuning is performed iteratively, usually in several iterations. The remaining part of the dataset is used to assess the algorithm's performance and is called test set. In machine learning competitions and benchmarks most of the time the ground truth of the test data is not released.

In computer vision, machine learning is used, e.g., for image classification, object detection, and segmentation. The task of object detection (Fig. 2.5b) is to find instances of the object classes in an image, for example windows, doors and balconies in a facade image. Semantic segmentation deals with splitting an image into a number of given object classes (Fig. 2.5c), i.e., each pixel is assigned a class label.

Formally, a dataset $S = X \times Y$ consists of n samples $X = \{x_1, x_2, ...x_n\}$ and their corresponding class labels $Y = y_1, y_2, ..., y_k$. Each sample comprises m features, $x_i = (x_i^{(1)}, x_i^{(2)}, x_i^{(3)}, ...x_i^{(m)})$. A trained machine learning algorithm



(a) (b) (c)

Figure 2.5: Machine Learning Tasks. (a) Input image, (b) Object detection, where the detector finds windows in an image (c) Semantic segmentation of a facade image, where each pixel is assigned a class label.

is a function that receives a data sample and outputs a class label $f(x_i) = y_i$. In pixel-wise facade segmentation, the samples X consist of a set of pixels of each image and extracted features such as color channels, HOG, or LBP. Their corresponding labels are $Y = \{wall, window, door, ...\}$.

2.3 Decision Trees

Decision trees (DT) are machine learning algorithms used for classification tasks. They work according to the principle "divide-and-conquer". Each node has a split function which divides the dataset, usually in two parts. The goal of the split function is, thus, to create two homogeneous sets where one part of the data is "different" from the other part. This is repeated recursively until a predefined target is achieved. During testing, each sample traverses the tree based on the split functions and the label is determined in the leaf node.

Formally, a decision tree is a function that receives a sample x and outputs a proposed label y, $f(x) = y$ where $x \in X$ and $y \in Y$. The split function (Equation 2.4) of a node j is a binary function which decides if it branches to the left or right. It has the parameters to be learned θ_j and is defined as:

$$h(x, \theta_j) \in \{0, 1\} \tag{2.4}$$

The parameters $\theta_j = \{\theta_j{}^{i_1}, \theta_j{}^{i_2}...\theta_j{}^{i_s}\}$ are learned based on a subset of features $\{x^{(i_1)}, x^{(i_2)}, ..x^{(i_s)}\}$ where $\{i_1, i_2, ...i_s\} \in \{1, 2, .., m\}$. In practice, each split function is created using a small number of features, usually not exceeding three.

The most common split function is built choosing a feature $x^{(i)}$ and a threshold τ resulting in $\theta = (i, \tau)$ and Equation 2.4 becomes:

$$h(x, \theta_j) = h(x, (i, \tau)) = \begin{cases} 1, & \text{if } x^{(i)} \geq \tau \\ 0, & \text{otherwise} \end{cases} \tag{2.5}$$

Another common split function (Equation 2.6) is defined by three parameters, $\theta = (i, j, \tau)$, consisting of two features with indices i and j and a threshold value τ which is compared with the difference of the feature values. Most of the time a feature normalization step is required when this split function is used.

$$h(x, \theta_j) = h(x, (i, j, \tau)) = \begin{cases} 1, & \text{if } x^{(i)} - x^{(j)} \geq \tau \\ 0, & \text{otherwise} \end{cases} \tag{2.6}$$

The training of the decision trees is usually based on a greedy approach where each node learns its parameters based on its subset of the dataset. During training each node has to decide which feature(s) and corresponding threshold(s) should be selected.

Assume that a subset $S_j \subseteq X \times Y$ has arrived at the tree node j. Based on this subset the parameters θ_j are learned.

If the dataset consists of a large number of features and each feature has a considerable range of values, this process is computationally expensive. Furthermore, the computational complexity grows exponentially with the linear growth of the number of parameters.

The most commonly used function for evaluating and selecting the best parameters in the split node is information gain (Equations 2.7 and 2.8).

$$IG_j = IG(S_j, S_j^L, S_j^R) \ , \tag{2.7}$$

where $S_j^L = \{(x, y) \in S_j \mid h(x, \theta_j) = 0\}$, $S_j^R = S_j \setminus S_j^L$, are the left S_j^L and the right S_J^R subset that are divided by the split function.

$$IG(S_j, S_j^L, S_j^R) = H(S_j) - \sum_{k \in \{L,R\}} \frac{\mid S_j^k \mid}{\mid S_j \mid} H(S_j^k) \tag{2.8}$$

where $H(S_j) = \sum_y p_y log(p_y)$ represents the Shannon entropy computed from the discrete probability distribution of the class labels of the subset S_j.

In each split node, a predefined number of combinations (possibly all) of the parameters, θ_j, is evaluated. Parameters, which have the highest information gain, are chosen. The same principle is conducted recursively for both the right S_j^R and the left S_j^L node. The training process and, thus, parameter selection, starts at the root node where the whole dataset is considered in the split function. It ends when a given stopping criteria is fulfilled. The two most widely used stopping criteria are a predefined depth of the tree and a minimal number of samples in the leaf node.

2.4 Structured Decision Trees

Structured machine learning (structured prediction) algorithms (Bakir et al., 2007) output a structure, for example a graph or an (un-)ordered set of labels. The difference between structured and non-structured machine learning algorithms lies in the output dimensionality. Structured machine learning algorithms can be expressed as $f_{str}(x) = (y_{i_1}, y_{i_2}, ..., y_{i_n})$ for ordered and $f_{str}(x) = \{y_{i_1}, y_{i_2}, ..., y_{i_n}\}$ for unordered output sets.

A potential problem of structured machine learning algorithms is the exponential growth of the output space. Depending on the problem, output space reduction techniques such as Principal Component Analysis (PCA) and joint probabilistic approaches are adopted (Dollár and Zitnick, 2013, Kontschieder et al., 2011a).

For image segmentation, structured machine learning algorithms return a patch (Equation 2.9). Compared to Natural Language Processing or Signal Processing, patch prediction is challenging due to the very large feature space and the highly exponential output space. Thus, the learning approach (an efficient output space reduction technique and high qualitative features to be

chosen) and the optimization function have to be carefully adopted to overcome these two problems and to produce high quality results.

$$f_{str}(x) = \begin{bmatrix} y_{11} & y_{12} & y_{13} & \cdots & y_{1n} \\ y_{21} & y_{22} & y_{23} & \cdots & y_{2n} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ y_{d1} & y_{d2} & y_{d3} & \cdots & y_{dn} \end{bmatrix} \tag{2.9}$$

Training structured decision trees is more complex than training normal decision trees, especially for segmentation, since Equations 2.7 and 2.8 have to be adapted to labels in the form of patches. The output space of a labeling problem with c classes and output patch size $n \times m$ is c^{nm}.

In facade segmentation, the number of classes is in the range of four to eight classes and the output patch is of size 11×11 up to 19×19 resulting in 10^{72} to 10^{326} combinations. Yet, in practice this number is considerably smaller since the patches often consist of large homogeneous regions. Still, the number of combinations can reach 10^6.

To reduce the complexity of the output space, different randomization, probabilistic and dimension reduction techniques have been adopted and applied in patch output space reduction. More details about challenges of the output space and the techniques used to solve them for the problems encountered in this thesis are presented in Chapter 4.

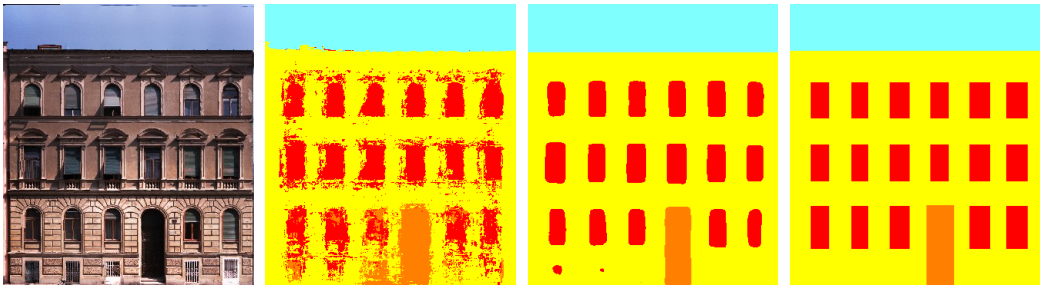
2.5 Structured Random Forests

Random forests (Ho, 1995) are ensemble learning algorithms for classification. (Standard) random forests $f_{rf}(y) = x$, consist of a finite number n of decision trees. Each tree of a random forest is trained independently.

A single decision tree is sensitive to over-fitting and the random forests reduce the over-fitting in comparison to decision trees. Since random forests consist of multiple trees, in principle, they perform better or on a par with decision trees as they comprise multiple independent models. On the other hand, random forests are more costly to compute than decision trees for both training and testing. The runtime and memory requirements increase linearly with n. To reduce them, trees in random forests usually have a lower depth than plain decision trees.

At test time, when classifying a data sample, each tree classifies the sample independently and each output class is considered as a vote. The votes from the random forest trees are collected and the final class for the sample is chosen by a predefined voting methodology, usually majority vote.

Similar to standard random forest, in Structured Random Forests (SRF) (Kontschieder et al., 2011a) each tree is trained independently and each structured output is considered as a vote. The final output is produced using a predefined voting methodology.



(a) Input **(b)** Random forest **(c)** SRF **(d)** Ground truth

Figure 2.6: Random forest and Structured Random Forest (SRF)

Structured machine learning algorithms have the ability to learn complex label distributions and local structures. They, thus, create a more homogeneous output compared to non-structured algorithms. Fig. 2.6 presents results for a standard and a structured random forest. Both algorithms are trained with the same features and training methodology and similar parameters (stopping criterion, number of trees, etc.) resulting in a much more homogeneous output for the SRF.

2.6 Artificial Neural Networks

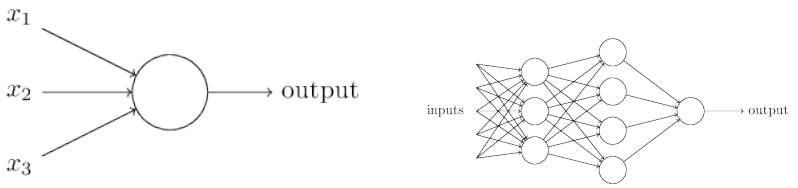
Artificial neural networks (ANN) are inspired by the working principles of the human cerebral cortex. They consist of nodes corresponding to neurons which receive information, process it and create an output.

The first the ANN-based algorithm was developed by Rosenblatt (1958) and it was called *Perceptron* (Nielsen, 2015). It was inspired by an extension of the previously published work of McCulloch and Pitts (1943).

A Perceptron (Fig. 2.7a) contains two layers, the first layer (input) consisting of neurons that receive the feature values of a data sample $(x_i^1, x_i^1, .., x_i^m)$. The number of the input neurons is equal to the number of features of the dataset. Each neuron is associated with a parameter referred to as weight w_i, which is trainable. The sum of the elementwise product of the input feature values and the weights is the input for the output neuron (Equation 2.10).

$$result = w_1x_1 + w_2x_2 + ...w_nx_n \tag{2.10}$$

The second layer receives the *result* and then post-processes it by using a predefined **activation function** f_{percep} which usually maps the *result* to the interval $[0, 1]$. This is convenient since it can be treated as a probability, $f_{percep}(result) = y,\ y \in [0, 1]$. It is worth noting that ANN research went



(a) Perceptron (Nielsen, 2015) **(b)** Multilayer Perceptrons (Nielsen, 2015)

Figure 2.7: First Artificial Neural Network Architectures (Nielsen, 2015)

through optimistic and pessimistic times, the later occurred because of the lack of computational power and appropriate learning principles.

Nowadays, advanced ANN learning algorithms and architectures are the state of the art in computer vision and speech recognition problems (Graves et al., 2013, Zhao et al., 2017) and are included in most computer vision solutions, industrial as well as academic.

In the following paragraphs we present an introduction into two neural network architectures. The first is the Multilayer Perceptron which is an extension of the Perceptron and the second, the Convolutional Neural Network. They are both particularly suitable for computer vision tasks.

For for more details and theoretical background of ANN please refer to (Nielsen, 2015).

Multilayer Perceptron

The perceptron architecture can be extended by adding more layers. This results in increased capabilities. The new layers are placed between the input and the output layer and are called "hidden layers". This architecture consisting of input, several hidden and output layers is termed "Multilayer Perceptron" (MLP) (Fig. 2.7b). Additionally, each layer has a bias neuron which is independent of the input and the previous layer. This neuron contains a learnable weight and has a static value, which usually is set to one.

Each neuron receives input from the previous layer and processes it with the activation function. In the literature, different activation functions such as threshold, sigmoid, hyperbolic tangent function (tanh) and ReLU (Nair and Hinton, 2010) have been introduced.

An MLP usually consists of a large number of parameters. Let's suppose that we have a fully connected MLP with four layers, the input layer has 100 neurons, the first hidden layer 250 neurons, the third layer 70 neurons and the output layer 10 neurons. In total, this small and simple architecture consists of $100 \times 250 + 250 \times 70 + 70 \times 10 = 43\ 200$ weights and additionally $250+70+10 = 330$ bias weights. Since MLP architectures usually have a lot of parameters, they need a larger training data set compared to other algorithms, such as decision trees.

For computer vision problems, the input layers must have a large number of neurons. For example, a very small RGB image patch with dimension 15×15 pixels results in an input layer with $3 \times 15 \times 15 = 675$ neurons. Thus, for a complete image and several layers, the number of parameters will grow and can easily reach 10^{15} parameters, which entails problems in terms of computing time and the amount of data needed for training.

Convolutional Neural Network

Convolutional Neural Networks (CNN) (LeCun et al., 1998) mostly solve the problem of a very large number of parameters for Computer Vision problems.

A convolutional layer is different from an MLP layer concerning the way the image data are treated. CNNs are designed specifically for two (2D) and three dimensional (3D) data, and the "CNN neurons" are represented as a 2D or 3D matrix of trainable weights.

The parameters of the convolutional layers are independent of the image size, since every convolutional layer acts as a sliding 2D or 3D filter through all channels and the complete image. Convolutional layers try to find, position-independent, a specific pattern or feature in a smaller or larger part of the image.

Besides the convolutional layers, the second type of layers of CNN architectures are the pooling layers. They perform down-sampling over the output of a convolutional layer by integrating the extracted convolutional features of a specific neighborhood. Furthermore, pooling layers make the result less sensitive to image rotations. The combination of the two types of layers leads to a powerful tool for automatic feature extraction in images as demonstrated, e.g., by the Imagenet dataset benchmark (Deng et al., 2009). CNNs eliminate the need for handcrafted features, since CNNs do this on their own.

Analysis of CNNs shows that the first layers learn image primitives such as edges, corners and simple geometric elements which are important for the task. In the middle layers, the pooling layers integrate image primitives and simple geometric elements into bigger ones and they detect and identify objects. Deeper layers detect complex objects and understand large parts of or even the complete image.

The last layer(s) of CNN architectures used for classification or similar tasks are fully connected MLPs. They provide a probability distribution over the classes, which is very useful for interpretation. Unfortunately, the fully connected MLP layers restrict the input size of the images to a predefined static size.

The pioneering CNN-based architectures for image classification are LeNet (LeCun et al., 1998) for OCR classification and Alexnet (Krizhevsky et al., 2012) for the ImageNet dataset (Deng et al., 2009). These architectures achieved a classification accuracy of more than 70% in a thousand class image classification problem and outperformed the previous traditional, handcrafted feature extraction based methods by a large margin.

2.7 Object Detection

Object detection is a computer vision task which finds a target object in an image and determines its location and dimension. Window and door detection

in facade images are two important object detection tasks in the scope of this thesis.

First research on object detection dates back more than 50 years (Andreopoulos and Tsotsos, 2013). The algorithms used model fitting and most of the research was related to finding cylinders in images. Until 1990, object detection (Andreopoulos and Tsotsos, 2013, Biederman, 1987) adopted different edge and corner detectors to find geometric primitives and to perform template matching based on geometric primitives.

At the beginning of the 90s, object detection based on the search for geometrical primitives was replaced by machine learning. In the remainder of this thesis we divide machine learning in two parts, the "traditional" machine learning methods, where the features are hand-crafted, and the deep learning methods, where the algorithms also extract features and the research is oriented towards Deep Neural Network architectures.

Traditional Object Detection Methods

Traditional object detection has concentrated on developing new feature extractors and time efficient machine learning algorithms for producing high quality object detectors. In the following, influential feature extraction and machine learning techniques for object detection are presented.

In 1991, Turk and Pentland (1991) introduced **Eigenfaces** (Turk and Pentland, 1991), an algorithm for face detection based on the image dimension reduction methodology Principal Component Analysis (PCA). They perform PCA on the training set and each image is represented as a vector in the corresponding eigenspace.

Viola and Jones (2001) present a real time face recognition system including a novel image representation, called **integral image representation** (IIR), which allows to compute a large number of features, for example Haar-like features, over image regions with high speed.

IIR consists of the sum of all pixel values above and to the left (Equation 2.11) for each pixel location (i, j).

$$IIR(i,j) = \sum_{ii \leq i, jj \leq j} I(ii, jj) \tag{2.11}$$

This representation allows to compute the sum of pixel values for every desired rectangle of the image in constant time ($O(1)$) (Equation 2.12).

$$R(i_1, j_1, i_2, j_2) = IIR(i_2, j_2) + IIR(i_1, j_1) - IIR(i_1, j_2) - IIR(i_2, j_1) \quad (2.12)$$

In equation 2.12, $R(i_1, j_1, i_2, j_2)$ represents the rectangle with the top-left corner at the position (i_1, j_1) and the bottom-right corner at the position (i_2, j_2).

Viola and Jones (2001) (Viola and Jones, 2001) present a cascade classifier which determines high probability regions which comprise an object and discard other areas, such as the background, from further processing.

Latter, Dollar et al. (2009) developed a real time feature extractor, called **Integral Channel Features** based on the IIR representation, which led to improved runtime and accuracy. They also showed that by using IIR (Equation 2.11), very time consuming features can be approximated with very small error, with a high efficiency and suitable for processing real time. IIR allowed to compute the features 100 times faster.

We have used this representation for our features as well as Equation 2.12 for fast computation of the optimal rectangle for rectangular fitting.

The **Implicit Shape Model** (Fig. 2.8) (Leibe et al., 2004) combines a bag-of-word method and Generalized Hough Transform (Ballard, 1981). It decomposes the object into a class-specific alphabet of parts and learns the location distribution relative to the object.

The alphabet is generated from the training set, where a large number of object patches around image points. e.g., (Förstner and Gülch, 1987), is extracted. The patches of dimension 25×25 pixels are clustered based on a similarity function and each cluster votes for the object center location. This method is one of the most influential works on object detection based on parts.

In Section 2.1, the HOG (Dalal and Triggs, 2005) feature extractor has been introduced. Dalal and Triggs (2005) built a linear SVM with HOG features for classification of the generated candidates for sliding windows over different image resolutions. (Felzenszwalb et al., 2010, 2008) present a "star model" for object detection. Detectors for the whole object are constructed from features extracted from down-sampled (lower resolution) images while detectors for the parts of the object are built from features extracted from higher resolution images. For example, when detecting humans, the features extracted from the down-sampled images help to find the whole human body while the features extracted from higher resolution allow to detect parts like

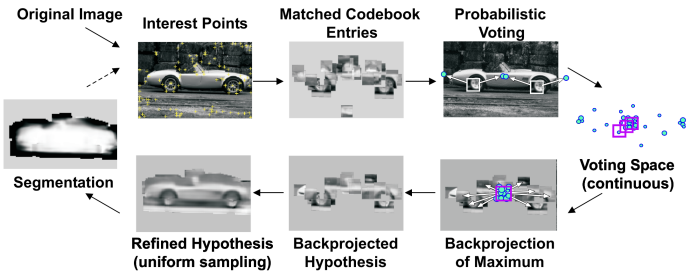


Figure 2.8: Implicit Shape Model architecture. Image taken from (Leibe, 2004)

legs or hands. Together they delineate and define the whole object as well as its parts very precisely.

Feature extraction is performed on several image resolutions leading to a feature pyramid (Fig. 2.9). For each object part, each layer of the feature pyramid defines its position and "deformation" relative to the object. All layers with their distribution of the object parts are input to a Mixture Model which defines the final object position in the image.

The algorithm by Felzenszwalb et al. (2010) is one of the most accurate models using a "traditional" machine learning algorithm. It integrates successful computer vision methodologies and is a state-of-the-art algorithm in traditional object detection.

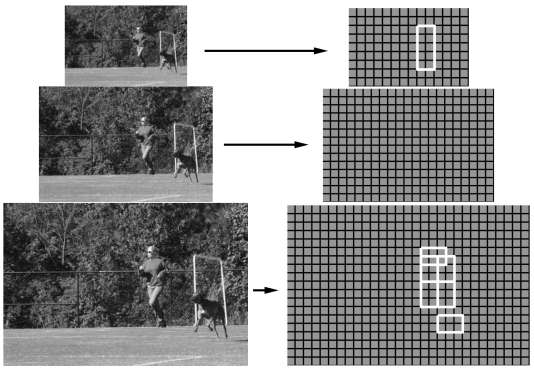


Figure 2.9: Feature pyramid. Image taken from (Girshick, 2012)

Deep Learning Based Object Detectors

In traditional object detection, the crucial parts are feature extraction and the integration in a suitable machine learning algorithm. Furthermore, for fast detection, sliding windows and search space reduction techniques were developed. Still, these methods do not scale well for large datasets and most of the techniques are problem-specific.

Deep learning (DL) methods overcome the problems of both feature extraction and search. Yet, they need a lot of data for training, a requirement only recently fulfilled by openly available datasets with millions of labeled images.

Deep learning methods for object detection have outperformed the traditional approaches not only in terms of speed but also by high margins concerning accuracy in problems with a large number of object classes. They can be divided into two types: The first type is region proposal-based, which is meant to mimic the human brain. First, an attention mechanism proposes regions of interest. Then, the algorithm focuses on these regions where a potential object might appear. Thus, the proposal component and the detection network have to share information with each other.

The second type of detectors is based on regression/classification methods that detect objects directly in the input image, without a proposal component. This type of DL approach is faster, but it has problems to detect small objects and detailed structures in the image and in general it is less accurate than region proposal-based approaches.

In this thesis, for object detection just the first type is employed. In the next section the Mask Regional Convolutional Neural Network (R-CNN) object detector is introduced.

Mask R-CNN. Region Proposal-Based Method

Girshick et al. (2014) have introduced R-CNN as an object detection architecture which solves the problem of feature extraction using a pre-trained CNN. R-CNN is a hybrid approach using traditional machine learning as well as deep learning. On each image the algorithm selects 2000 regions of interest using the selective search algorithm (Uijlings et al., 2013). Then, each region is fed in the pre-trained CNN network which generates a 4096 dimensional feature vector. The latter is used for training a linear SVM for classifying the object proposal.

While R-CNN has improved the state of the art and had a huge influence on object detection, its time and memory efficiency is not satisfying. It takes about one minute to find the potential objects on a single image and it requires more than 100 GB of RAM memory for the selective search, the CNN model, and the SVM model. Another reason for its poor performance is that the proposal and the detection components do not "communicate", leading to a large amount of redundant computation.

Fast R-CNN (Girshick, 2015) is an upgraded model, where memory requirement and runtime are reduced. As for R-CNN, selective search is used to generate the proposals' region of interest. The Fast R-CNN does not generate features for each of the proposals separately, but it extracts features using a CNN for the complete image. Each object selects features from the region that it corresponds to and transforms them to a predefined size using pooling layers. Finally, the feature vector is fed to the linear SVM.

The new architecture achieves a slightly better detection accuracy than R-CNN. Discarding redundant computation resulted in a speed up of the detection component of more than one hundred. Still, the algorithm depends on the selective search which also needs time.

The first neural-network-only object detector is Faster R-CNN (Ren et al., 2015). It improves Fast R-CNN by dropping selective search and the linear SVM. The object proposal part of Faster R-CNN is a fully convolutional network (Section 2.6) which takes as input an image and outputs the object proposals represented by their location and an objectness score. Faster R-CNN is the first end-to-end object detector, which does not need hand-crafted features as well as a searching mechanism, since this is done by the CNN. Faster R-CNN is more than three times faster than Fast R-CNN.

Mask R-CNN (He et al., 2017), an advanced architecture built on Faster R-CNN, is a CNN-based algorithm that not just detects the objects in the image, but also delineates the detected objects, i.e., segments the instances. After the region proposals, an object mask is selected in parallel with object detection (object class and box). The final object layer outputs the object and its mask. The algorithm improves the current state of the art concerning instance segmentation as well as object detection.

2.8 Semantic Image Segmentation

Semantic Image Segmentation is a crucial part of computer vision as it becomes more and more integrated in the automotive, entertainment and health industry, etc.

The first segmentation algorithms were addressing the single object segmentation problem, where each image pixel is labeled with one of the two classes background or object. For this task, several datasets were published (Fergus et al., 2003, Shotton et al., 2005).

With advances in graph algorithms and different machine learning techniques, multi-class segmentation algorithms were introduced. The first successful algorithms for segmentation with or without human interaction where graph-based and are also used in commercial products like Adobe Photoshop.

In the current Deep Learning era, segmentation algorithms achieve a very high accuracy due to sophisticated neural network architectures and post-processing methods.

The following text presents the most influential segmentation techniques. The term (image) labeling will be used synonymous with semantic (image) segmentation.

Graph Based Segmentation

In the 90s, graph based segmentation (Wu and Leahy, 1993), started to become popular, where image segmentation is transformed into a graph separation problem. Nodes represent pixel locations. From their values edge weights are derived with each node usually having four neighbors.

The segmentation problem is defined in graph theory as the division (partitioning) of the graph in disjoint subgraphs. Graph-based segmentation can be split into five groups (Peng et al., 2013) based on the used graph algorithms: Minimum Spanning Tree, Graph Cut with cost functions, Graph Cut on Markov Random Field models, Shortest-Path-based methods and other methods.

Minimum Spanning Tree (MST) based methods use clustering. For partitioning a graph into disjoint non-cyclic (trees) parts, Prims Prim (1957) and Kruskal (Kruskal, 1956) MST algorithms and variations for specif image

segmentation problems are used. Edge weights are determined by a function of the values of the pixels (node intensities) which an edge connects. The connected pixels satisfy the MST constraint that the sum of the edge weights is minimal and the subgraphs are divided based on predefined "trivial" rules, like that objects have the same color. In general, MST algorithms give results of limited quality, since they are very sensitive to color changes. Additionally, they are dependent on cluster initialization.

Graph Cut with cost functions divides the graph into disjoint sets using predefined criteria that are represented by an optimization function over the complete graph. The advantage of Graph Cut over MST is that one can use more advanced functions for partitioning. Furthermore, MST can be considered as a special case of Graph Cut. Graph Cut algorithms were in particular introduced for single object segmentation from an image based on human interaction, where a person annotates some region(s) of the object and/or background and the Graph Cut algorithm segments the object.

Graph Cut on Markov Random Field (MRF) algorithms integrate high level information. In contrast to the previously described algorithms, where just spatial knowledge is encoded, these algorithms integrate contextual information.

Shortest Path-based methods need human interaction for the segmentation of an image. They receive as input an image and a set of points around the object that is to be segmented. The shortest path (Dijkstra) algorithm is used to connect the points and segment the object. The edge weights of the graph are computed as a function of the pixel intensities around it. The most commonly used shortest path algorithm is Livewire (Mortensen and Barrett, 1995).

A more detailed theoretical overview of graph-based segmentation algorithms can be found in (Peng et al., 2013).

Conditional Random Field-Based Segmentation

Conditional Random Field - (CRF) based approaches encode knowledge about the neighborhood in the form of statistical dependencies. CRF (Lafferty et al., 2001) labeling approaches were first used for natural language processing. A CRF entails a combinatorial optimization approach with a possibly very high runtime complexity. Since sentences have a relatively small number of words,

e.g., 10 words, CRF have outperformed the previously published graph-based methods.

In computer vision, one of the first approaches that introduced a CRF is (He et al., 2004). The CRF incorporated global and local features. CRF models particularly face challenges concerning runtime efficiency during inference. To overcome these challenges, several inference techniques were proposed. Currently the most commonly used is belief propagation (Pearl, 1982).

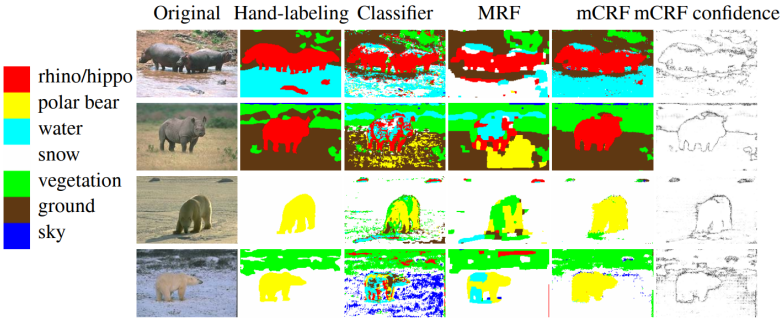


Figure 2.10: Results of mCRF (He, 2008)

In a CRF for image labeling two type of "energies" are used, the *unary potentials*, i.e., features derived from a pixel alone, and the *pairwise potentials* computed from two or more neighboring pixels, i.e., joint information. Initial methods struggled to incorporate global information in the CRF, since the CRF potentials encoded very local pixel level information.

(He, 2008, He et al., 2004) present a CRF model that integrates the global context of the image by learning the relationship of the class labels at different image scales. By this means, the label distribution in the image is learned and more homogeneous regions are generated (Fig. 2.10).

(Yang, 2011, Yang and Förstner, 2011) present a hierarchical CRF for image segmentation, particularly suitable for facades and buildings. It not only encodes spatial and contextual but also hierarchical information. For example, the algorithm learns that a road is always at the bottom of the building and a building is always under the sky. The hierarchical information improves the accuracy of the model by more than 3% for benchmark datasets.

CRF methods nowadays are used as postprocesing technique after, e.g., a CNN segmentation of an image. For more information about CRF we refer to (Liu et al., 2016, Yang, 2011).

CNN-Based Segmentation

Like in object detection, deep learning approaches outperform the traditional approaches also in semantic segmentation. This is especially true for complex datasets and problems with a large number of classes. For deep learning, the most important part is the network architecture and for special problems the optimization function. The first deep learning semantic segmentation approaches could also only label images with a fixed size.

The first breakthrough which enables processing of images with variable dimension have been the Fully Convolutional Networks (FCN) (Long et al., 2015). This architecture does not contain fully connected layers, which are transformed into convolutional layers. Furthermore, a new type of layer was introduced, the deconvolutional layer, which does the unpooling of the previous layer. This architecture improved the segmentation quality but it still produced a coarse segmentation, because the deconvolutional layers couldn't keep track of the higher resolution information.

After FCN were introduced, research focused on improving the FCN architecture, especially the deconvolutional layers, such that during the up-sampling information from the earlier layers of the architecture could be used.

(Badrinarayanan et al., 2017) introduced SegNet, where inverse pooling layers are used during deconvolution. The indices of the max-pooling layers are used to up-sample the convolved layers. Thus, the process can be described as convolutional layers in reverse order. This leads to less-memory efficient models, but it improves the quality of the segments.

(Yu and Koltun, 2016) presented a new type of layer, Dilated Convolutions, specifically suited for image segmentation. It addresses the problem that the pooling layers decrease the resolution when increasing the receptive field of the network. The Dilated Convolution (atrous convolution) preserves the spatial resolution while increasing the receptive field.

Experimentally, when the pooling layers are replaced with Dilated convolutions, the accuracy of the network increases significantly. On the other hand, these layers are computationally expensive and they need more memory, during both, training and testing. To achieve time and memory efficiency, models that contain atrous convolution layers usually downscale the images during training and testing.

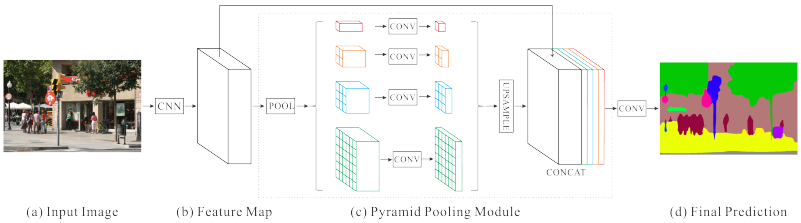


Figure 2.11: Pyramid Scene Parsing Network architecture. Image taken from hszhao.github.io/projects/pspnet/

(Zhao et al., 2017) propose the Pyramid Scene Parsing Network (Fig. 2.11) which collects information from versions of the image with different scale. First, features from the highest resolution image are extracted, which are passed to (sub-) networks for several scales of the image. In the end, the outputs of the sub-networks are concatenated and the final segmentation is produced. This architecture improved the results considerably.

CNN-based methods have achieved a very high accuracy level and have changed the corresponding part of the tech industry. Furthermore, the scientific community and big companies like Facebook and Google have developed and support Neural Network-oriented libraries such as Tensorflow, Pytorch and Caffe.

On the other hand, CNN models have millions of parameters which are very hard or impossible to be interpreted or explained. They also need special hardware for training and real time processing. An open topic in Neural Network research is to understand how they decide how to segment or find objects in an image and decide how susceptible the solutions are to perturbations of the image data.

Chapter 3

Related Work

Facade segmentation and object detection in facade images have been a topic of research over the last 20 years. Earlier publications (Dick et al., 2002, Mayer and Reznik, 2006, Reznik and Mayer, 2008, Schindler and Bauer, 2003, Wang and Hanson, 1997, Werner and Zisserman, 2002b) in this area were focusing on window detection and integration into 3D models. In the last ten years, pixel-wise facade segmentation become an active field of research with a focus on the development of grammars for parsing facade images.

Previous window and door detectors (Dick et al., 2002, Mayer and Reznik, 2006, Reznik and Mayer, 2008, Schindler and Bauer, 2003, Wang and Hanson, 1997, Werner and Zisserman, 2002b) were built using geometric and parametric models. They were pattern recognition-based and their capability for capturing the information on the facade was encoded in their mathematical models.

With the development of machine learning techniques and artificial neural networks for computer vision problems, researchers started to employ them. The development of fast graphic cards and the increased computational speed of processors advanced computer vision, machine learning and deep learning and allowed their combination for facade segmentation.

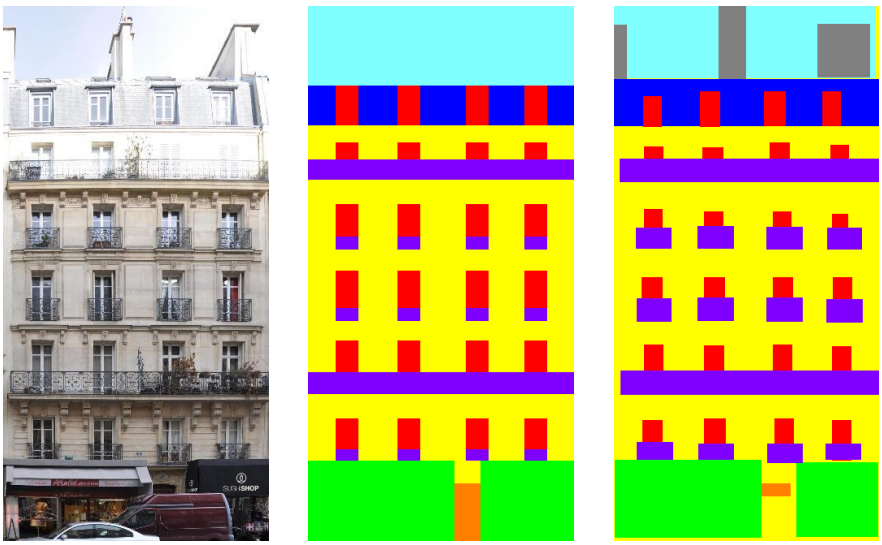
For facade images, two popular problems came in the focus of research: window detection and pixel-wise segmentation of the whole facade image.

The most prominent facade parts are usually windows. Their surface consists mostly of glass. This causes problems in 3D model reconstruction and, thus, a considerable amount of research is oriented towards window detectors and their integration into 3D-models. Only a limited amount of work is concerned

with door detectors, since they are considered less frequent and, therefore, less important facade objects.

This chapter first introduces the currently most relevant facade benchmark datasets. Furthermore, it presents recent facade object (windows and doors) detectors as well as facade segmentation techniques evaluated on the benchmark datasets.

3.1 Facade Datasets



(a) Input image **(b)** Annotated image **(c)** Updated Annotation

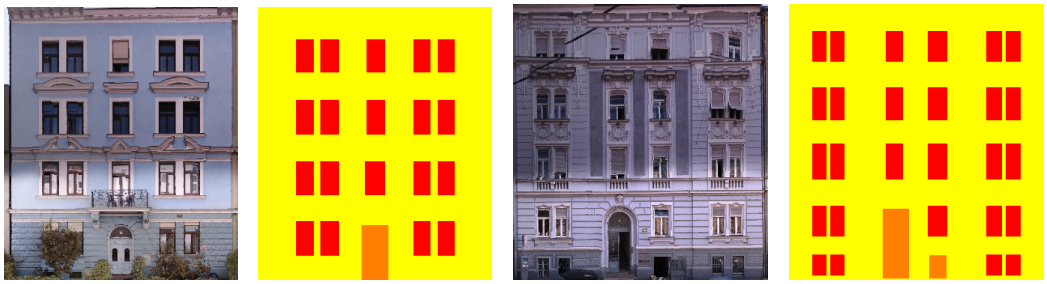
Figure 3.1: a) shows the input image. b) and c) the first and the second (updated) version of the annotation of the input image. The second annotation is of higher quality. Object classes: window, door, balcony, sky, wall, roof

The first dataset for facade segmentation, the **The Ecole Centrale Paris (ECP) 2010 dataset**, was published in (Teboul et al., 2011). It consists of 104 rectified facade images. Each image is acquired in Paris and depicts only one Haussmannian architecture style facade (Fig. 3.1). Since the initial annotations were partially not correct, a second version with corrected labels was published (Martinović et al., 2012) and has replaced the initial version.

All images are annotated with the seven classes balcony, chimney, door, roof, shop, wall and window and an additional void class for non-facade objects such as poles and traffic signs. The evaluation of the algorithms is mostly done on only six classes, omitting the chimney class.

Facade images of the ECP dataset contain occlusions, cars, pedestrians, traffic signs and vans in front of the building entrance, doors and shop windows as well as flowers and vegetation besides windows and balconies.

The images have been manually rectified and contain a considerable amount of optical (barrel) distortion. Their size is in the range of 90K to 348K pixels.



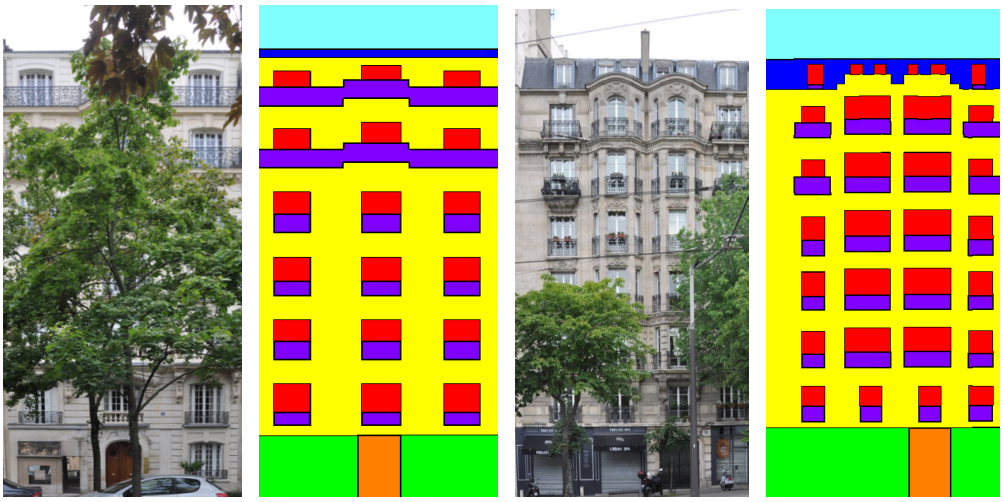
(a) Input image **(b)** Annotated Image **(c)** Input Image **(d)** Annotated Image

Figure 3.2: Graz dataset sample images and their corresponding annotations. Object classes: window, door, sky, wall

Another important dataset is the **Graz dataset** (Fig. 3.2) (Riemenschneider et al., 2012) which consists of 50 facade images. The facades are from different architectural styles (Classicism, Biedermeier, Historicism, and Art Nouveau) and each image contains only one facade.

The images have been annotated with four classes: door, sky, wall and window. They have a resolution between 76K and 238K pixels and contain a very small amount of occlusion, e.g., by vegetation, in the front of the facades.

The ParisArtDeco dataset (Gadde et al., 2016) (Fig. 3.3) comprises 80 images acquired in Paris. The images are labeled with the same classes as the ECP dataset. Also the architecture of the buildings and the size of the images is similar to the ECP dataset. Large part of the images are covered by trees, other types of vegetation and non-facade objects. This dataset is mostly used for evaluating grammar-based segmentation methods.



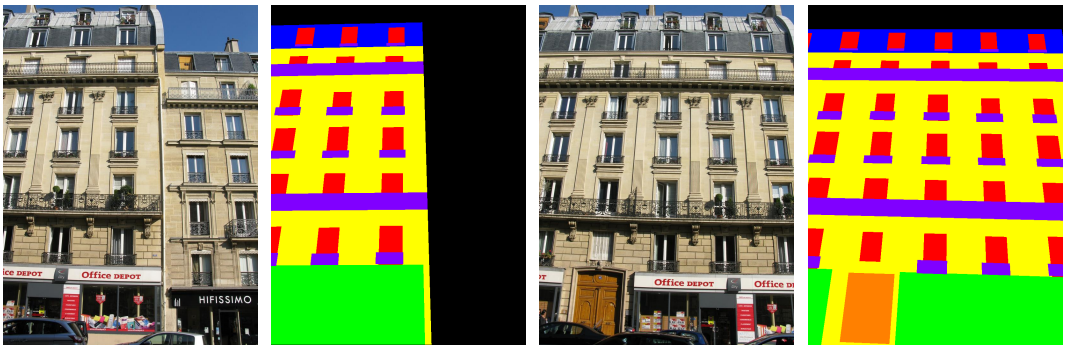
(a) (b) (c) (d)

Figure 3.3: Paris ArtDeco dataset sample images and their corresponding annotations. Object classes: - window, - door, - balcony, - sky, - wall, - roof

Riemenschneider et al. (2014) published the dataset **RueMonge** for facade segmentation and facade labeling in 2D images as well as 3D point clouds. It contains images of the same buildings as the ECP dataset but with a higher resolution of 800 × 1067 pixels. The employed classes are the same except that the chimney class is ommited. While facades of the previous datasets have been rectified, this is not the case for this dataset (Fig. 3.4). The dataset includes the ground truth for the 3D mesh and the point cloud. It consists of 428 images of which 217 have been annotated. This dataset has been used for the Varcity 3D challenge.

3.2 Window and Door Detectors

In this section we present recent window and door detectors. Most of the methods are based on parametric models of windows. A frequently used heuristic for window detection is to make use of the grid structure and symmetries of facades. Different parametric functions have been developed to model non-rectangularly shaped windows which are frequent on facades of buildings of older architectural style.



(a) (b) (c) (d)

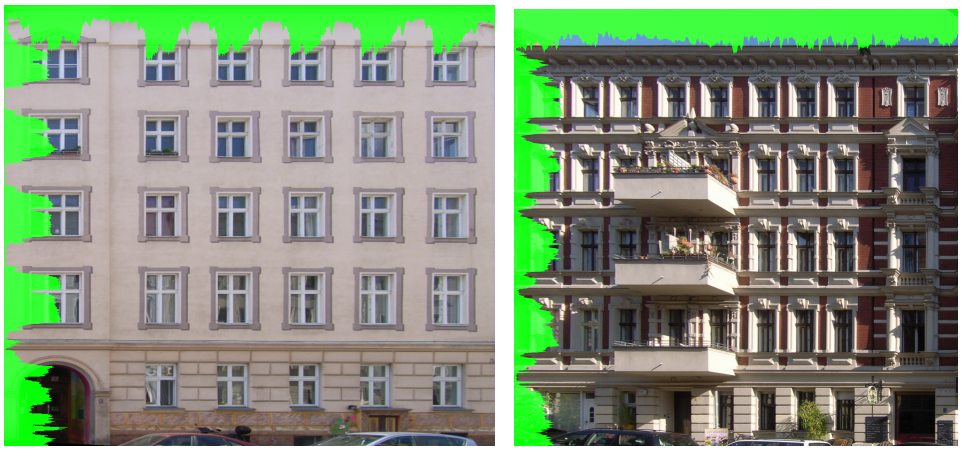
Figure 3.4: Paris RueMonge dataset sample images and their corresponding annotations. Object classes: window, door, balcony, sky, wall, roof

Image Gradients Based Window Detection

(Lee and Nevatia, 2004) is a pioneering work on window detection. The detector is based on the assumptions that facades have a grid structure and that the regions with windows contain a larger number of line segments.

Vertical and horizontal edges of a rectified facade image are projected in the vertical and horizontal direction. The projections create two profiles (histograms) which are used to define the regions in the image that contain windows (Fig. 3.5). High frequency peaks of edges in the row- and column-direction imply windows. Window candidates are generated from the grid created by the intersection of the high frequency peaks. This method produces satisfactory results for flat facades with a grid structure, a simple wall texture and without ornaments.

Recky and Leberl (2010) have extended and improved (Lee and Nevatia, 2004). They have performed unsupervised k-mean segmentation on the "a" and "b" channels of CIELab-transformed images showing that these channels provide more suitable gradients for facades. They also demonstrate that the k-means algorithm performs better on CIELab-transformed than on RGB images, since problems with shadows and occlusions are reduced.



(a) (b)

Figure 3.5: Horizontal and vertical projection of extracted lines is the basis for Lee and Nevatia (2004) to find windows arranged on a grid.

Probabilistic and Parametric Based Window Detection

(Ali et al., 2007) present a method based on Cascade classifiers built with Gentle Ada-boost. The pipeline is devised to detect windows on arbitrary (non-rectified) facade images. It works well for simple facades, but fails for modern buildings and produces a considerable amount of false positives.

(Jahangiri and Petrou, 2009) introduce a method inspired by the human vision system. The human vision system is assumed to be attracted by regions which contain some variance and texture, called "blobs". On facades, "blobs" correspond to windows which are brighter or darker and their texture differs from the wall. Thus, window detection is approached by extracting "blobs". An edge preserving algorithm is applied and contours are delineated. The detected contours are checked whether they delimit blobs. Overlapping blobs are eliminated because windows do not overlap. The rest of the blobs are assumed to correspond to windows.

A probabilistic model for window detection is given by (Tyleček and Šára, 2010). The model consists of four groups of parameters: number of windows k, shape attributes A (dimension, aspect ratio), location X (center of the window) and neighborhood relation N.

The goal is to find the most probable configuration of $\theta = (k, A, X, N)$ given an image I. Since the search space and the number of parameters is large, the joint probability distribution is decomposed and a conditional probability density function is built using problem specific priors. Different features are extracted, such as the likelihood from different color channels, the lines (segments) and the neighborhood configuration, e.g., in the form of the space between windows. The detection framework combines a weak non-specific classifier and a weak structural model. To find the best parameters for a given image, a reversible jump Markov Chain Monte Carlo (RJMCMC) (Green, 1995) framework is employed.

This model does not force windows to form a grid and does not rely on hard architectural constraints since none of the priors has a probability equal or close to one. It produces a very good recall on facades of different architectural type, but with the drawback of a high rate of false positives for the windows.

Neuhausen et al. (2018) have built a window detector based on the Soft Cascade Classifier (Bourdev and Brandt, 2005, Viola and Jones, 2001). They use boosted trees and Haar-like features to create a fast patch-based detection algorithm and have shown that to achieve high quality results only a few weak classifiers are needed to identify the non-window patches. For sliding the detector over the facade they use the approach of (Viola and Jones, 2004, 2001) and to reduce the search space they employ heuristics comprising, e.g., the range of window dimensions. The system achieves a detection rate of 85% with 2% false positives (Fig. 3.6) on the ECP dataset, although we note that the detected rectangles do not have a very precise delineation with most of them covering parts of the wall or a balcony.

Figure 3.6: Results presented in (Neuhausen et al., 2018).

Window Detection using Markov Marked Point Processes

Wenzel et al. (2008) approach window detection by searching for repeated structures on the facade. Prominent features are extracted with the SIFT (scale invariant feature transform) operator (Lowe, 1999), resulting in their location, scale and orientation. To detect symmetries, original and (vertically and horizontally) flipped versions of the features are matched. Matching structures are clustered and from the clusters the repeated structures, assumed to be windows, are derived. Results show that the method works well on facades with a grid structure and without occlusion, but has problems with facades without a grid structure.

Wenzel and Förstner (2016) present a window, balcony and door detector based on a Markov Marked Point Process (Baddeley and van Lieshout, 1993). A supervised model is formulated as a Gibbs process to learn feasible configurations of windows, balconies and doors based on points and lines in rectified facade images. For producing and optimizing the hypotheses, RJMCMC is used. Excellent results are presented for simple facade images (Fig. 3.7), but the authors state that the algorithm does not produce competitive results on the ECP dataset or other benchmarks. Similar to Wenzel et al.

Figure 3.7: Results presented by Wenzel and Förstner (2016). On the left the input image, in the middle the ground truth and on the right the results from the detector are presented.

(2008), Wang et al. (2015) model facade objects and their structure jointly in a Marked Point Process framework. Supervised learning is employed to classify each pixel whether it is a window and a probability value is assigned to it. The positions of windows are sampled from the probability map generated in the previous step and the dimensions of the windows from a predefined range. An energy function is constructed with a prior term encoding local relationships of windows, such as the space between windows, and the global layout encoding window alignment and repetition patterns. The data term corresponds to the probability map generated in the previous step. The approach produces results of high quality for specific facade types with a runtime of two to seven minutes per image on the ECP dataset.

Window Detection in 3D Point Cloud

Werner and Zisserman (2002a) present an approach using multiple images to recognize arch-shaped and rectangular windows. To obtain an arch-shaped window, they replace the top straight line by an arch.

Dick et al. (2004) introduced one of the first approaches for 3D building reconstruction using window and door detectors. In this early work, the facade objects window and door of Gothic and Classical architectural style are modeled using a Bayesian approach.

It is assumed that the architectural style is known. Since Gothic style buildings contain arch-shaped windows with sloped edges, they need more parameters for modeling: In addition to the position (x, y) of a corner and the width and the height (w, h), arch height a and slope of edge b are used.

Nguatem et al. (2014) present a fully automatic approach to detect windows and doors in 3D point clouds. Probabilistic template matching using a Monte Carlo simulation is employed to search for objects on the facade plane. By representing the templates as B-splines, the algorithm finds also arch-shaped windows (Fig. 3.8) achieving highly accurate results (Fig. 3.9). Unfortunately, this method is very computationally expensive: the runtime for detecting the windows on a single facade can be several minutes.

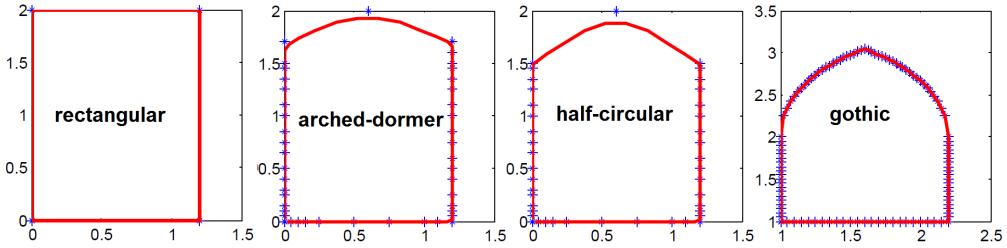


Figure 3.8: Different type of windows detected in (NGUATEM et al., 2014) using template matching.

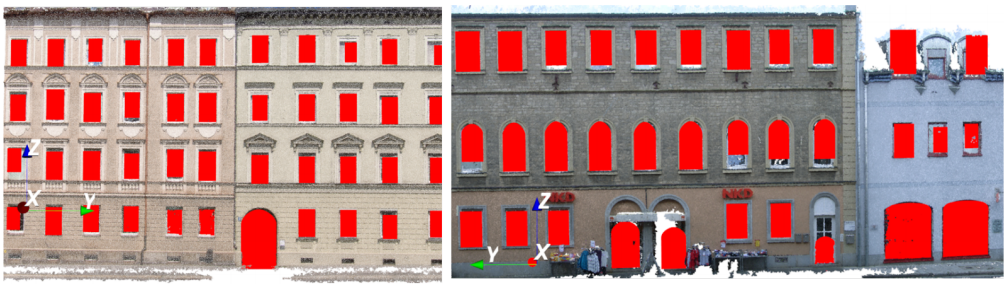


Figure 3.9: Results presented in (NGUATEM et al., 2014)

3.3 Pixel-wise Facade Segmentation

With the recent advances of semantic segmentation techniques, facade segmentation caught the attention of the computer vision and remote sensing community. Another reason motivating research on facade segmentation is that it is an important prerequisite for 3D building reconstruction and urban scene understanding.

Facade segmentation approaches are divided in two main categories: bottom-up and top-down. Bottom-up approaches directly classify pixels or larger image regions. Subsequently, "optimization" methods such as CRF or Dynamic Programming are applied to the labeled facade image. In the final phase, architectural constraints are integrated for a better delineation or to enforce a global grid structure.

On the other hand, top-down methods use shape grammars to parse the facade images. In practice, bottom-up methods are more accurate and faster than top-down approaches. Yet, top-down parsing algorithms deliver good facade segmentation results for images which contain large occlusions, since

they learn the global structure of the facade based on an initial state from the visible facade part.

In the following two sections various approaches for top-down and bottom-up segmentation are introduced in detail. Finally, Section 3.7 presents a couple of Deep Learning-based approaches.

3.4 Top-Down Facade Segmentation

Top-down facade segmentation approaches parse the facade using shape grammars. They provide a structured view of the facade elements and represent facades as hierarchies of objects or elements. Grammars stem from Natural Language Processing, and are only seldom used in computer vision, as they imply a rigid structure which does not comply with most natural objects.

Shape grammars are suitable for facade segmentation, since facades are often very regular and can be decomposed hierarchically in smaller elements, and these decompositions can then be represented as a parse tree. In this section we give a short overview of the most influential grammar-based approaches for facade segmentation.

In (Teboul, 2011, Teboul et al., 2011) shape grammars and Reinforcement Learning (RL) are used to label facade images. A random forest is trained and a probability distribution is assigned to each pixel for the classes roof, shop, wall, and window.

Since MCMC can be considered a special case of RL (Teboul et al., 2011), RL will perform better than MCMC. Particularlly, horizontal and vertical facade division rules of the binary parsing tree are learned with RL.

The first rule is that a facade is built of floors. It divides the facade horizontally in several floors and is defined by a starting position in the image. The basic vertical division rule divides the floors in windows and wall. Thus, the parse tree recursively divides each floor into facade objects.

The learning of the split function is based on a Markov Decision Process (MDP), where the next state, i.e., the next facade division, depends only on the current state.

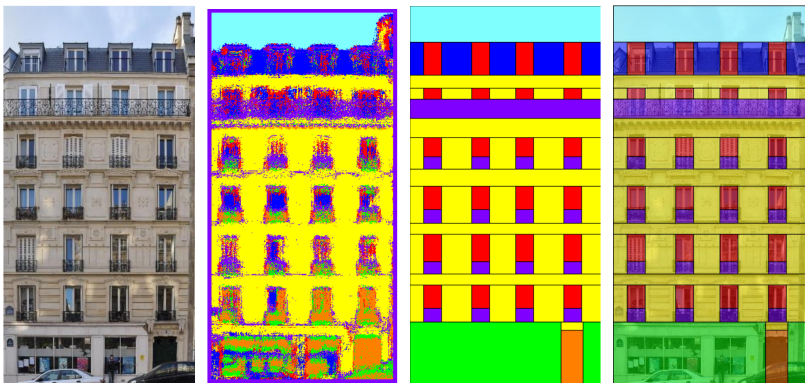


Figure 3.10: Results of (Teboul, 2011). (a) Input image, (b) Class with highest probability produced by labelling with the random forest (c) and (d) Segmentation produced by reinforcement learning and the parsing tree without and with image in the background.

The constructed grammar and the RL lead to a symmetry-based facade segmentation. Even when the image is occluded or a part of the facade is cropped, it produces a symmetric facade labeling.

The algorithm has been trained and evaluated on the first version of the ECP dataset. The results show homogeneous segments and rectangular shaped windows, since the grammar defines the shapes as rectangles (Fig. 3.10). In terms of runtime on the ECP dataset, the RL algorithm needed 30 second to parse the facade initially labeled by the random forest. The algorithm achieves an accuracy of 84% correctly classified pixels.

Riemenschneider et al. (2012) present an approach for facade segmentation based on a grammar can parse non-grid as well as non-symmetric facades. They show that for facades which can be parsed into irregular rectangular lattices, i.e., the facade is divided in rectangles with arbitrary dimension, Dynamic Programming (DP) can be applied. Consequently, their rules do not separate the whole facade with horizontal or vertical lines, but they only enforce that every column of the facade has the same number of floors. The basis is the random forest labeling approach by (Teboul et al., 2011). On top of it an object detector is added during grammar construction.

The evaluation of this method shows relatively good results achieving an accuracy of 80% for the ECP dataset and 78% for the Graz Dataset.

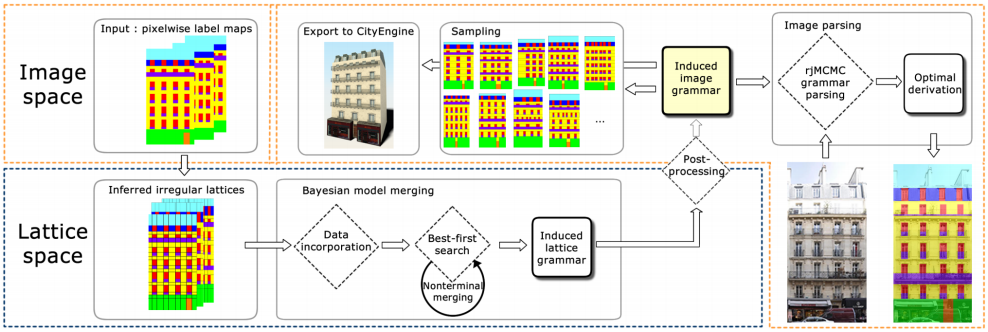


Figure 3.11: Approach presented in (Martinović, 2015)

In (Martinović, 2015, Martinovic and Van Gool, 2013) a faster parsing method based on the Earley-Stolcke parser (Stolcke, 1994) is proposed. Initially, this parser has been built for string parsing in NLP and Martinovic and Van Gool (2013) have been the first to adapt this method for 2D input data and to apply it to facade segmentation (Fig. 3.11).

The parsing algorithm is built based on (Riemenschneider et al., 2012), first generating rules from each image. This results in a very large set of rules. To reduce the number of rules, iterative merging based on a Bayesian model is applied. This creates a small set of rules which is more effective in terms of speed and accuracy, because it is more general and less sensitive to over-fitting. This approach achieves an accuracy of 74.8% on the ECP dataset.

Figure 3.12: Parse tree in (Kozinski, 2015)

A facade parsing method based on user-defined shape priors and hierarchical partitioning into the grid is presented in (Kozinski, 2015, Kozinski et al., 2015)(Fig. 3.12). It allows a simultaneous alignment in both image directions.

Each cell of the grid is classified into pseudo semantic classes (V, H), where V represents the label of the row and H the label of the column of the grid and

Figure 3.13: Results for (Kozinski, 2015)

$V, H \in \{window, wall, balcony, ...\}$. Additionally, rules are defined for pairs of neighboring cells defining in what combinations pseudo semantic classes can occur.

To learn the hierarchical parsing based on hierarchical adjacency patterns, Maximum Posterior Probability (MAP) estimation is used for a Markov Random Field (MRF).

The algorithm parses an image in about 30 seconds and achieves a classification accuracy of 92.5% on the Graz Dataset and of 91.3% on the ECP dataset (Fig. 3.13). The results demonstrate that this method defines the state of the art of the top-down methods.

3.5 Bottom-Up Facade Segmentation

In this section we are going to present methods that use same principle as our method. Most of the Bottom-up methods use a pipeline architecture which consists of an initial labeling or segmentation method and then improving the result step by step by using different optimization approaches.

Facade Segmentation using a three layer architecture

In (Martinović, 2015, Martinović et al., 2012), a pipeline is presented that consists of superpixel segmentation and a Recursive Neural Network (RNN) (Socher et al., 2011) which incrementally creates larger and larger semantic segments. Additionally, window and door detectors integrated into an MRF

improve the delineation of the segments. Finally the labels are optimized based on hard architectural and dataset-specific constraints.

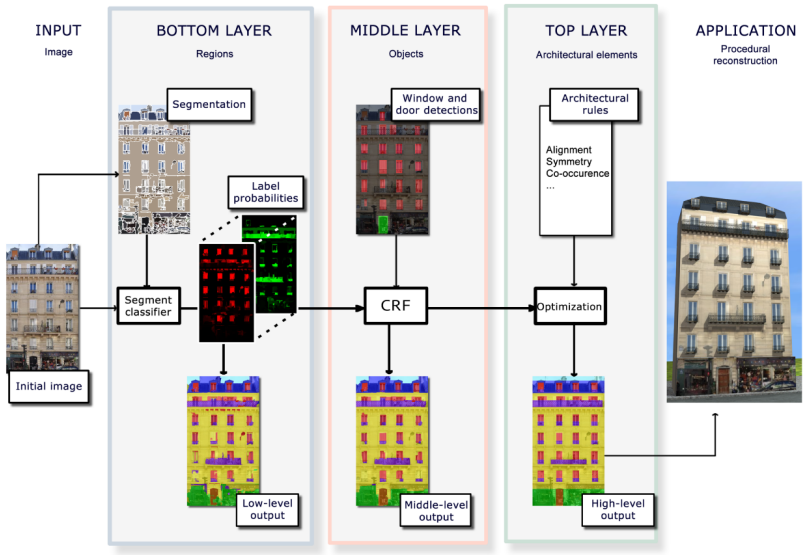


Figure 3.14: Architecture of the proposed system by Martinović (2015)

The mean-shift algorithm (Comaniciu and Meer, 2002) is used to segment the facade images and to create super-pixels. On top of this, an RNN is trained, which can detect repetitive (recursive) natural structures. For super-pixel merging, a bottom-up binary parsing tree is built. In order to produce a high quality segmentation of the facades, several feature extractors are employed on more than 600 of the initial merged super-pixels, where each super-pixel may cover a complete facade object or a part of it. The features are fed to the RNN input layer and based on the features the RNN incrementally merges the neighboring segments.

Since the boundaries of the RNN output are of low quality, a second layer with window and door detectors is employed. The detectors act as a second information source for delineation and help to reduce noise. The Integral Feature Channel (Dollar et al., 2009) is used for constructing the detectors. The detector scores and the RNN outputs are integrated into an MRF. The MRF energy minimization is based on four-pixel neighborhood pairwise potentials using the Potts model.

On the top layer, weak as well as hard architectural constraints are introduced. The weak architectural constraints are rules that are valid for each

architectural style, like the roof is on the top or the building entrance door is on the first floor. Opposed to this, hard architectural constraints follow a specific architectural style like windows have the same dimensions or the entrance door is always in the middle of the first floor. Furthermore, dataset specific constraints are used. These are rules that are just valid for a dataset (city or part of it) but not necessarily for an architectural style.

In the optimization function, the following constraints are encoded: vertical and horizontal (non-) alignment, similarity of different facade objects, facade symmetry, co-occurrence of elements, equal width/height of facade objects in a row or column, door hypothesis is on the first floor and touching the ground, vertical order of facade object layers: {shop*, facade^{+},roof*,sky*}, and a running balcony is on the 2nd and 5th floor.

Vertical and horizontal (non-) alignment is implied by the fact that most facades have a grid window structure. This constraint is formulated as energy minimization for the object and its four neighbors.

The similarity of different windows on the same facade is determined based on an Implicit Shape Model-like (Section 2.7) voting schema. For incorporation of the facade, vertical symmetry, symmetry lines are determined.

The constraint that facade objects have an equal width and height along a row or column is applied if found "beneficial" in the validation set. For example, if both the RNN and the door detector have not found a door, a door hypothesis is generated based on the probability distribution of the gradients and the size of the other facade objects.

Additionally, the authors have added data-set specific constraints for the ECP data-set which for the Haussmanian-style buildings propose a (long) running balcony on the second and the fifth row of windows.

The complete pipeline (Fig. 3.15) achieves a pixel-wise accuracy of 85% on the ECP dataset. It needs 180 seconds on average to segment a single image. The results demonstrate a high-quality delineation of the objects. Yet, these high-quality delineations are only achieved when also the last layer is used.

An extended version of the three layer architecture of (Martinović et al., 2012) was presented by Mathias et al. (2016) (Fig. 3.15) with the ATLAS framework where different superpixel segmentation, classification and energy minimization algorithms have been evaluated.

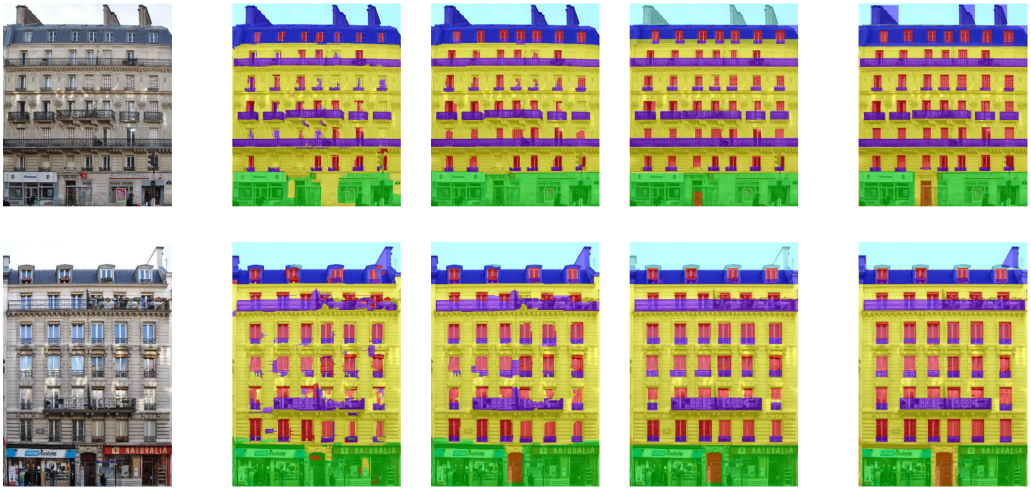


Figure 3.15: Results presented by (Mathias et al., 2016). The layers architecture is the similar for Martinović et al. (2012) expect in the second layer instead of CRF, a MRF is used. Image taken from Martinović (2015).

For superpixel segmentation they have used three types of segmentation algorithm, Mean-Shift, SEEDS (Van den Bergh et al., 2012) and gPb (Arbelaez et al., 2011).

They evaluated the performance of Multiclass Logistic Regression, CRF, Multilayer Perceptron, Multiclass Support Vector Machine (SVM) and RNN for superpixel classification. The experiments show that Multiclass SVM performs best both concerning pixel-wise as well as average class accuracy in combination with any of the three superpixel segmentation approaches. For the bottom layer they, thus, chose the combination of Mean-shift and Multiclass SVM.

For the second layer, a new dataset consisting of Belgian facade images with more than 3900 windows, 440 doors and 850 cars from different viewing angles was introduced to create better detectors. It was used to evaluate the Deformable Part-based Model detector (DPM) (Felzenszwalb et al., 2010) and the *Very Fast* detector (Benenson et al., 2012). The *Very Fast* detector outperforms DPM, because the doors and windows do not have "moving" parts and they always have a rectangular shape. Yet, for car detection the DPM shows a better performance.

The object detector and Multiclass SVM outputs are fed to the CRF to delineate the objects. The last layer mostly remains the same, just the dataset-based constrains such as that the second and fifth row contain a long running balcony are omitted.

The extended model achieves 88.1% accuracy on the ECP dataset which is better by around 4% than the previous model. The time required on average for one image is more than 190 seconds.

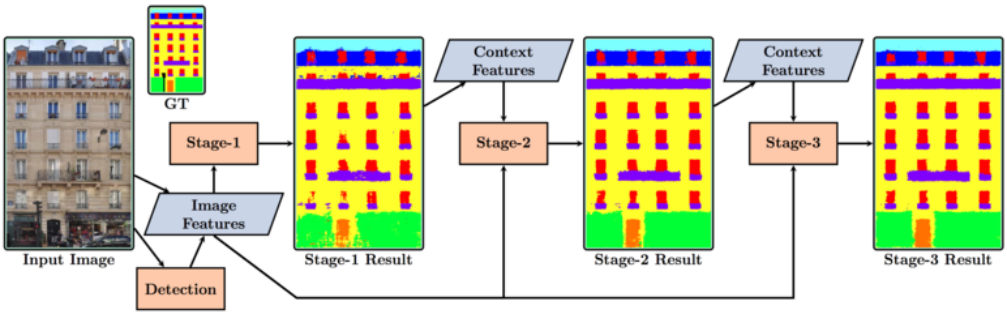


Figure 3.16: Architecture of the proposed system by Martinović (2015) in the works (Martinović, 2015, Martinović et al., 2012)

3.6 Auto-Context for Facade Segmentation

(Jampani et al., 2015) present a facade parsing method based on Auto-context (Tu, 2008). Their pipeline consists of an object detector for windows and doors, boosted decision trees used in three iterations of Auto-context and a CRF for creating homogeneous segments and an improved object delineation.

Auto-context introduced by Tu (2008) is a multi-stage iterative segmentation method. In the first iteration a machine learning algorithm segments an image labeling each pixel with the probability distribution over the classes. In the next iteration another algorithm extracts additional features from the output of the previous iteration.

Jampani et al. (2015) have trained a door and a window detector using Integral Channel Features (Dollar et al., 2009) based on the publicly available toolbox (Dollár, 2017). They output the objects' location and confidence (probability) scores. For the two detectors separate image channels are created and the corresponding confidence scores are assigned to each pixel which lies

in the detected rectangle. Pixels that are not part of any rectangle have value zero. The detector can output overlapping rectangles and each pixel obtains as score the sum of the confidence scores.

The Auto-context classifiers are trained together with boosted trees by means of stacked generalization (Wolpert, 1992). In the first iteration, features such as 17 TextonBoost filter responses, dense Histogram of Oriented Gradients (HOG), Local Binary Pattern features and row and column average filters are extracted. In total, 761 low-level and two detector features are used to built the boosted decision trees.

The boosted decision trees output for each pixel the probability distribution over the classes. For the next Auto-context iterations, several pixelwise, regional and global features are extracted from the previous image labeling. They consist of the pixel class probability distribution, pixel entropy, the percentage of each class per row and per column as well as the average probability score of each pixel per row and column.

Additionally, several regional features are computed consisting of the Euclidean and Manhattan distance to the nearest pixel of each class, the log likelihood based on the RGB channels for each class, bounding box features and neighborhood statistics. These features are added to each pixel and a new classifier is built. For Auto-context three iterations are used, since after three iterations the result does not improve any more.

For optimizing the output and, particularly, for reducing the noise of the segmentation, a CRF is employed with eight neighbors and pairwise Potts potentials. This step takes at least 24 seconds per image. This is rather long compared to all previous steps that do not take more than six seconds.

The method is evaluated on the five datasets ECP and Graz (Section 3.1), eTrims (Korc and Förstner, 2009), CMP (Tyleček and Šára, 2013) and LabelMeFacade (Fröhlich et al., 2010). The presented results show that the magnitude of the Auto-context segmentation improvement is correlated with the number of classes. For example, the Graz dataset consists of four classes, and after three iterations Auto-context improves the results by 1.5% compared to the initial labelling. The ECP dataset has seven classes and the improvement after three iterations is 2%. The CMP dataset consists of twelve classes and an improvement of more than 6% is achieved.

Even after three iterations, the results contain a considerable amount of noise (Fig. 3.17). This is because boosted decision trees are not capable to

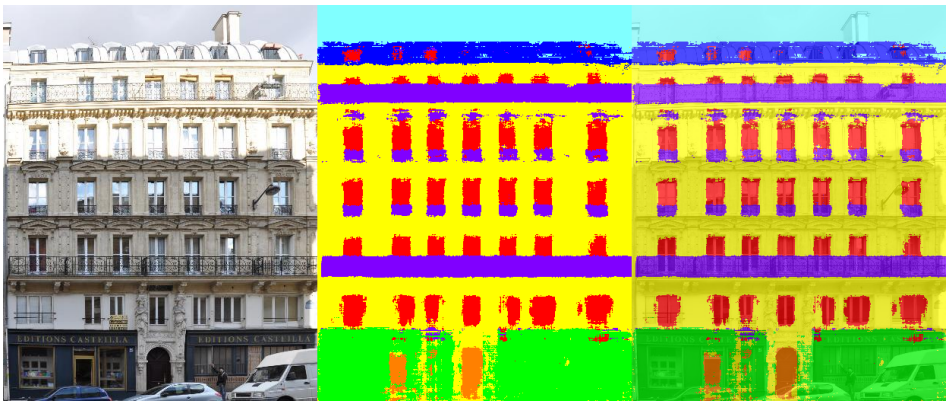


Figure 3.17: Results after three iterations of Auto-context (Jampani et al., 2015).

model the local structure and arrangement of the objects and are, thus, not able to produce noise-reduced images. Furthermore, boosted decision trees classify each pixel independently. To overcome these problems, i.e., to produce homogeneous segments and reduce noise, a CRF was added. In each of the seven datasets the CRF improves the result by less than 2% (0.5-2%).

(Gadde et al., 2017) evaluated Auto-Context for Point Cloud Labeling using the RueMonge2014 dataset (Section 3.1). The evaluation is conducted under several conditions: just using 2D data, point cloud only, or both together. The results show that when just using 2D data the overall pixel-wise classification accuracy is lowest but the IoU is better than when employing 3D data only. When using both, the pixel-wise accuracy increases by 1% and the IoU by 2%.

Dynamic Programming for Facade Segmentation

(Cohen et al., 2014) present a Dynamic Programming (DP) approach for facade segmentation. The main idea is to encode predefined problem specific constraints via DP.

The facade is labeled using a supervised learning approach based on boosted decision trees with features similar to (Martinović et al., 2012). It outputs a probability distribution over the classes for each pixel.

Several soft and hard architectural constraints are defined and encoded via DP. The hard constraints consist of: all facade elements are represented as rectangular boxes, balconies must be located below windows and must have

the same width and all pairs of window and balcony on the same floor have the same height. Soft constraints comprise, e.g., that chimneys are above the roof, the roof covers the entire width of the facade and shops are located on the entrance floor.

The DP approach is based on the incremental adding of objects to the optimization function. It finds the "best" rows with windows. Formally, given the pixel class distribution, the goal is to find a set of rectangular boxes $(x_{k,1}, x_{k,2}, y_1, y_2)$ that are on the same row and have a high score concerning the above conditions. After the window rows are found, the balcony rows are added and optimized together with the window rows.

Similarly, optimal rectangles for the shop and for doors are determined. Finally, a joint optimization of roof, sky and chimney is performed with a worst case time complexity of $O(N^5)$, where N is the height or width of the image. Although, the worst case complexity is huge, in practice the algorithm is very fast since there is only a small number of possible arrangements of the objects. The DP takes on average 2.8 seconds per image and the approach achieves a classification accuracy of 90.8% on the ECP dataset and 89,6% on the Graz dataset.

(Cohen et al., 2017) have extended their previous work by adding symmetry constraints to the facade segmentation. They extract a set of key-points with SIFT (Lowe, 2004). For each key-point a symmetric descriptor is computed for the mirrored image. By matching similar key points are found in the original image and the mirrored image. Each matched pair votes for a vertical symmetry line. The Hough-transform is used to find the most dominant vertical symmetry line. An additional post-processing step excludes non-symetric parts of the facade like the entrance floor or chimneys.

The symmetric features improve the accuracy by 1% for the ECP and 2% for the Graz dataset. On average, the symmetry features entail an additional processing time of 0.6 seconds per image.

3.7 Facade Segmentation Based on Deep Learning

The first deep learning approach for facade segmentation has been published by Schmitz and Mayer (2016). The architecture of the Convolutional Network

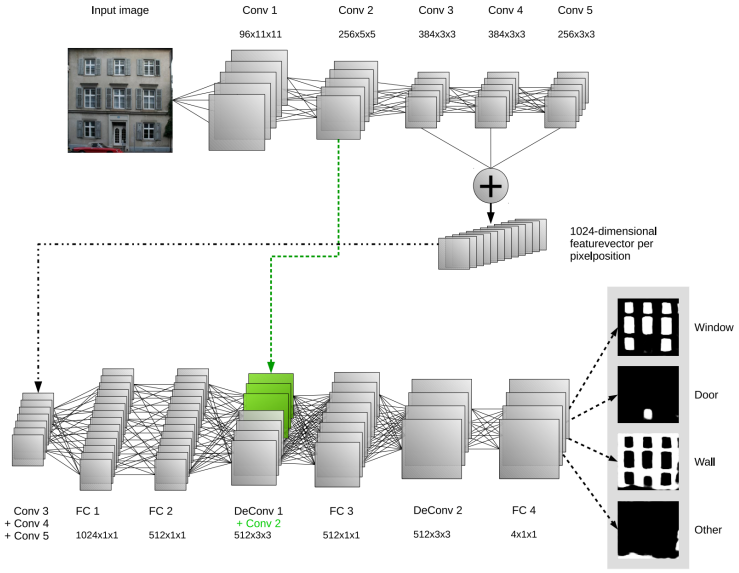


Figure 3.18: The CNN architecture presented by Schmitz and Mayer (2016) (Schmitz and Mayer, 2016)

is based on AlexNet (Krizhevsky et al., 2012). Inspired by U-net (Ronneberger et al., 2015) they have modified the architecture by concatenating three layers (layers three, four and five) with the same dimensions (Fig. 3.18). Additionally, they have added four Fully Connected layers and two Deconvolutional layers. The architecture is trained and tested with four classes consisting of window, door, wall and other. It achieves an accuracy of 85% on the test dataset.

(Liu et al., 2017) have been the first to use an end-to-end Fully Convolutional deep learning architecture for facade segmentation. They employ the VGG (Simonyan and Zisserman, 2014) Fully Convolutional pre-trained network. The end-to-end approach gives pretty good results but it also leads to some noisy homogeneous regions.

In this work also a novel facade-specific loss function has been proposed. Instead of the general cross entropy loss function, they encode in the loss function the symmetric structure of the facades (Equation 3.1).

$$L_s(x) = \sum_p (Var[X_c] + Var[Y_c]) \tag{3.1}$$

In equation 3.1 p represents a facade object (door, shop, window), and X_c and Y_c are random variables representing the positions of horizontal and vertical lines of the object p. This loss function minimizes the difference between the number of pixels on both sides of the lines enforcing that the facade objects have a rectangular shape (Fig. 3.19).

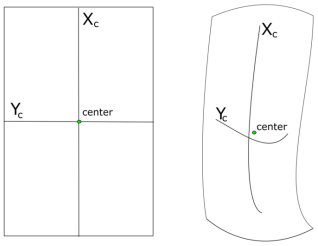


Figure 3.19: The symmetric loss proposed in (Liu et al., 2017). In the left image, the symmetric loss has value zero. Both symmetry lines intersect the object center. The right image has a considerable amount of loss and its symmetry lines do not pass through the object center.

To achieve better performance, just some of the layers of the VGG pretrained Faster R-CNN (Ren et al., 2015) architecture are used for detecting facade objects.

Finally the Faster R-CNN-based object detector and the FCN-based segmentation architecture are integrated by summing class-wise scores and labeling the pixels according to the highest score (Equation 3.2).

$$y = argmax(fcn_{out} + R\_CNN_{out}) \tag{3.2}$$

The approach achieves a very high quality in the form of an almost noise-free segmentation with an accuracy of 95.4% on the ECP dataset, which is the best current result for this dataset.

Chapter 4

Facade Segmentation Based on Structured Random Forests

This chapter presents a novel hybrid method with several components consisting of a "traditional" machine learning algorithm and its optimization method, deep learning for object detection and a fast greedy rectangular fitting algorithm. The components of the developed algorithm have been integrated into a pipeline architecture.

The previous chapter has shown that bottom-up approaches were found to be more efficient and accurate compared to top-down approaches. Thus, we have developed a bottom-up approach.

From the previous chapter we also concluded that all successful hybrid methods use a pixel-wise labeling approach in the initial step. The output contains a considerable amount of noise with *salt-and-pepper noise* most dominant. Additionally, there are often segments (super-pixels) that do not fulfill the soft architectural constraints of facades.

Our proposed segmentation algorithm essentially removes the salt-and-pepper noise completely and reduces other noise. In most cases it completely removes implausible segments on the facade image such as very small segments that appear in large regions of other classes, i.e., a very small segment of class door which is located in the middle of a large shop segment. Furthermore, the developed system implicitly fulfills all soft architectural constraints.

We, particularly, present a patch-based segmentation method which learns based on extracted features. Patch-based methods have the advantage over traditional pixel-wise approaches that they encode local dependencies of the objects.

Our system is built as a pipeline where each step improves the labeling produced by the previous step. The pipeline consists of the four parts feature extraction, object proposal, patch-based labeling and rectangular fitting. The first and the second produce features which are input to the classificator which uses patches to label the pixels. The last step transforms the labeled image into a suitable descriptive format.

As we employ decision trees, it is important to determine the best features that encode the distribution of the classes. Furthermore, the number of features grows fast, which effects the learning speed and accuracy of the decision trees. We, thus, employ feature cleaning to achieve fast training and inference as well as a high accuracy.

In the last step, model fitting is used to delineate the facade objects. It acts as regularizer and corrects and transforms the data in a shape similar to the ground truth.

Overall, this pipeline architecture allows to achieve our goal namely the creation of a segmentation algorithm that, due to its high accuracy, can be used as source of information for tasks such as the production of high quality 3D building models.

4.1 German Facade Dataset

For evaluating our approach on different architectural styles, we have built a dataset acquired in the three German cities Berlin, Munich and Mühldorf. It consists of images of facades from the XV-XIX century (Fig. 1.1). Image rectification is performed based on the 3D reconstruction method presented by Ley and Hellwich (2016). This is an advantage compared to the previously published datasets (Section 3.1), which have problems with the rectification due to geometrical distortion. We have annotated more than 30 buildings.

4.2 Architecture of the Model

The proposed pipeline consisting of four components is summarized in Fig. 4.1. Features are extracted from the image and in parallel objects are detected by a deep convolutional network. From the latter several features are constructed which act as object proposals for the next step. Both types of features are

used when building the Structured Random Forest (SRF) which produces the initial labeling of the images.

The SRF outputs a pixel-wise labeled image. While it could be modified to output a probability distribution over the classes for each pixel, we did not use this, because we empirically found that it did not result in any improvements for the next step.

An iterative optimization is applied to the SRF result leading to the "processed" labeled image. The final result is produced by greedy rectangular fitting.

Rectangular fitting creates a descriptive output for all facade objects. Each facade object is presented by its position (center) and dimensions. This representation is similar to that of grammar-based methods where the terminals usually have a rectangular shape. Our method makes use of advantages of both the top-down and the bottom-up paradigm. From the first it adopts the output representation which is suitable for our problem, and from the latter the fast and accurate labeling methodology.

The chosen architecture allows the separate evaluation and improvement of each component as well as the quantification of the influence of even small sub-components on the overall system.

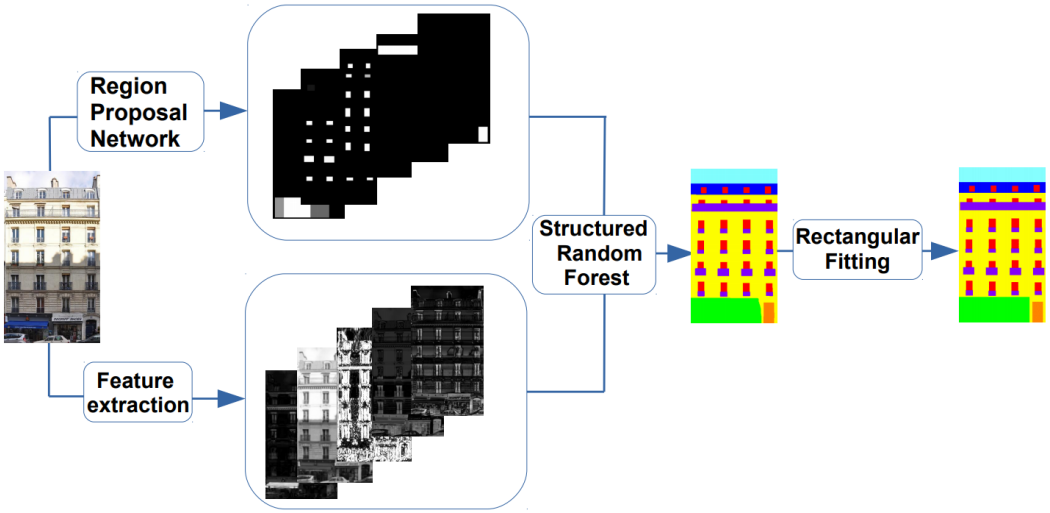


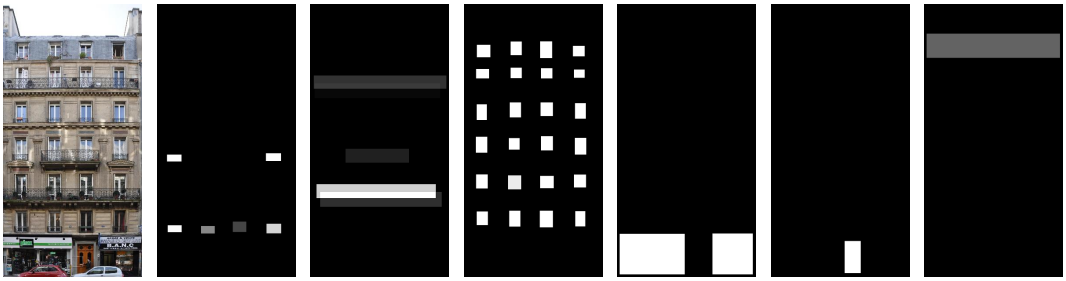
Figure 4.1: The architecture of the proposed pipeline for facade segmentation consisting of Feature Extraction, Object Detector (RPN), Structured Random Forest as well as its optimization and Rectangular Fitting

4.3 Facade Object Detector

The object detector finds objects in the image and determines their probability. Its output is used to produce features for each pixel, describing the probability of the pixel being part of an object.

Our object detector is built based on a pretrained CNN architecture. We have evaluated several pretrained networks based on Single Shot Detectors (SSD) and the Faster R-CNN architecture (Section 2.7) and found that the accuracy of the final result does not differ significantly.

Object detectors for balcony, running balcony, door, shop, roof and window have been developed. For their training published datasets (Section 3.1) and several images taken from Google Maps are used.



(a) (b) (c) (d) (e) (f) (g)

Figure 4.2: Object detector. (a) image, (b) balconies, (c) running balconies, (d) windows, (e) shops, (f) doors and (g) roof. The intensity represents the confidence of the network in the existence of an object at that position. The brighter the region, the higher is the probability of the existence of the object.

Because their appearance and dimensions are different, we decided to use two different classes for balcony, namely balcony and running balcony. We performed this split since CNN-based object detectors perform best if the object's scale relative to the whole image and the object's width and height ratio has a small variance. Furthermore, we also tested the algorithm with all balconies considered as one class and the detector did not produce useful results.

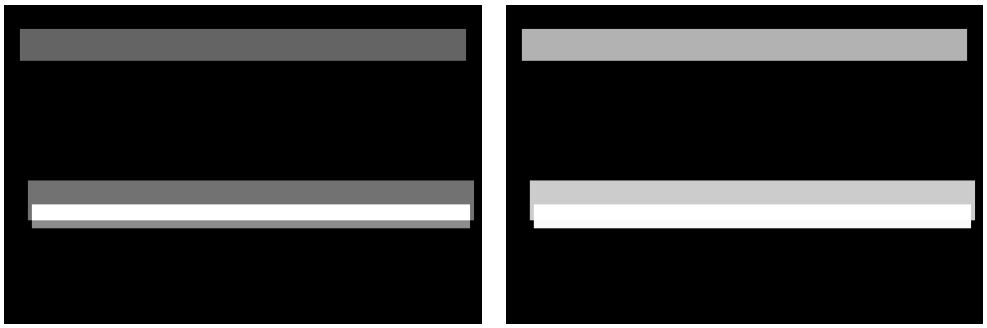
The most challenging objects for detection are entrance doors, since they are not as frequent as other objects, their appearance varies widely and they

might be partly covered by occlusions, e.g., by cars or humans. Furthermore, the detector has to distinguish shop doors from entrance doors.

To make it easier for the classifier, we redefine the object "door" by incorporating a small part of the wall around the entrance door. Since shops doors are usually not surrounded by wall, this additional information will allow the classifier for an easier distinction of shop and building entrance doors.

The output of the object detector is input to the SRF. We integrate them differently from all previous approaches presented in Section 3.3

The object detector D applied to an image I with width w and height h outputs a set of n rectangular regions $R(x_{iR}, y_{iR}, w_{iR}, h_{iR}, c_{iR}, p_{iR})$. Each rectangle is defined by the top left corner (x_{iR}, y_{iR}) and its width and height (w_{iR}, h_{iR}) with $1 \leq iR \leq n$, $x_{iR} + w_{iR} < w$ and $y_{iR} + h_{ir} < h$, the class label $c_{iR} \in \{balcony, long-balcony, door, roof, shop, window\}$, i.e., $c_{iR} < 7$ and the probability $p_{iR} \in [0, 1]$ of the corresponding class.



(a) **(b)**

Figure 4.3: Effect of the normalization techniques. Suppose that there are three long balcony proposals, with probabilities (starting from the top) 0.7, 0.8, 0.98. The left image shows the min-max normalization used by other methods. Here, the upper region has a probability of less than 0.4, i.e., it becomes a region with low probability. The right image depicts our proposed normalization method. None of the regions obtains a low probability

For using the output of the object detectors as a feature for the classification, a separate channel is created for each object and the detected rectangles are mapped to the channel. For the ECP dataset (Section 3.1) we created six feature channels $\{ch_1, ch_2, ..ch_6\}$ (Fig. 4.2) with the same dimensions as the input image I. The pixels are initialized with zero. All detected objects of the

class c_i are added in the channel ch_i, adding to each pixel inside the rectangle $R_j(x_j, y_j, w_j, h_j)$ the value p_j (Equation 4.1):

$$ch_i(x_j : x_j + w_j, y_j : y_j + h_j) \rightarrow ch_i(x_j : x_j + w_j, y_j : y_j + h_j) + p_j \quad (4.1)$$

Creating features in this way has problems with overlapping region proposals. It is common for most detectors to produce overlapping rectangles also for the same class (Fig. 4.3a). Simply adding probabilities for overlapping regions can result in probabilities higher than 1.0. This causes problems, because they lie outside of the range of probabilities.

Previous methods of (Jampani et al., 2015, Martinović et al., 2012) perform a min-max normalization in each channel. This may lead to that regions with a high probability become regions with a low probability. E.g., they might obtain a probability of less than 0.4 in case that somewhere in the image there are more than two overlapping high-probability regions.

Another problem with this normalization is that it takes into account only one image, thus, it is not treating all detection probabilities the same.

We have solved these problems, by max thresholding after all regions have been added to the channel. We have chosen the threshold 1.0 (Equation 4.2). Thus, pixel values can be interpreted as probabilities, for all non-intersecting regions the probabilities remain the same and the normalization is the same for the complete dataset (including test and validation set). This method produces features with a higher quality and does not punish non-overlapping regions (Fig. 4.3b).

$$ch_m(i, j) = max(1.0, ch_m(i, j)), i \leq w, j \leq h \quad (4.2)$$

Our feature normalization solves all the problems that we had with the min-max normalization.

The same principle is used for the Graz dataset and the German building dataset. In the Graz dataset, doors, windows and sky are detected.

4.4 Feature Extraction

In this thesis several feature extractors are used and we have additionally developed domain specific features.

Particularly, we extract RGB and CIELab, HOG, several LBP and TextonBoost features (Section 2.1) among others, initially resulting in more than 60 features per pixel.

Additionally, we have developed location-based features. Considering that facades have a specific vertical structure, we have created a feature for encoding the vertical pixel position (row). For an image I with height h, the vertical position feature is produced by assigning each pixel in the row r the value $\frac{r}{h}$ (Fig. 4.4.b). Furthermore, under the assumption that doors and long balconies are usually at the center of the facade, we encode the horizontal pixel position (column) in a similar way (Fig. 4.4.c).

(a) **(b)** **(c)**

Figure 4.4: Location based features. (a) Input image, (b) Vertical position feature and (c) Horizontal position feature

New features have also been engineered based on image rows statistics. This type of features has been introduced in Section 2.1. Similar statistical features have been used in previous work by (Jampani et al., 2015, Mathias et al., 2016), yet, none of them uses the standard deviation over the row or the difference of the pixel value to the median and the mean of the row (Equation 4.3).

$$pixel\_value(i,j) = |I(i,j) - \frac{1}{m}\sum_{k=0}^{w} I(i,k)| \tag{4.3}$$

4.5 Structured Random Forest for Facade Segmentation

The features described in Sections 2.1, 4.3 and 4.4 describe a region, row or single pixel and are used to build a Structured Random Forest (SRF) (Section 2.4) for pixel-wise facade labeling. In this section we present our SRF adaptation for facade segmentation as well as the split function used to train them. We follow the conventions for SRFs introduced in Section 2.4.

For training the SRF, the Extremely Randomized Trees (Forest) methodology of Geurts et al. (2006) is used. It selects the best split using random sampling of the split parameters. Each split node of the tree receives a number of tries n as parameter. In each attempt it randomly selects parameters θ which consist of several pairs (usually up to 3) of features and their corresponding thresholds. From the n tries the best split is selected.

In the split node we use eight different split functions. In each try one function is selected randomly for evaluation. Equations 4.4, 4.5, 4.6 and 4.7 are examples of the employed split functions and the remaining are similar to the presented functions.

$$h(x(i,j),\theta_1) = h(x(i,j),(k,\tau)) = \begin{cases} 1, & \text{if } x(i,j)_k \geq \tau \\ 0, & \text{otherwise} \end{cases} \tag{4.4}$$

$$h(x(i,j),\theta_2) = h(x(i,j),(k,l,s,t,\tau)) = \begin{cases} 1, & \text{if } x(i+s,j+t)_k - x(i,j)_l \geq \tau \\ 0, & \text{otherwise} \end{cases} \tag{4.5}$$

$$h(x(i,j),\theta_3) = h(x(i,j),(k,l,s,t,u,v,\tau)) = \begin{cases} 1, & \text{if } x(i+s,j+t)_k - x(i+u,j+v)_l \geq \tau \\ 0, & \text{otherwise} \end{cases} \tag{4.6}$$

$$h(x(i,j),\theta_1) = h(x(i,j),(k,l,m,s,t,u,v,a,b,\tau)) = \begin{cases} 1, & \text{if } 2 * x(i+a,j+b)_k - x(i+s,j+t)_l - x(i+u,j+v)_m \geq \tau \\ 0, & \text{otherwise} \end{cases} \tag{4.7}$$

The split function in Equation 4.4 contains two parameters and it is built just based on a single feature value of the central pixel in the channel k and its threshold. Equation 4.5 consists of five parameters which encode the relation between the value in channel k of the central pixel and the value in channel l of a pixel in the neighborhood as well as the corresponding threshold. Equation 4.6 presents a split function built from the values of two (not necessarily) different channels of two pixels in the neighborhood of the central pixel and their corresponding threshold. In the last equation 4.7 a split function is shown built from the values in three (not necessarily different) channels of three pixels around the central pixel and their thresholds. Equation 4.7 has ten parameters and a large number of random samples is required to find a satisfying solution.

Each feature parameter of the split functions is sampled randomly without a heuristic from the range 0-255 corresponding to the range of the pixel values. The distance parameters range from -25 to 25.

SRFs have a problem with the combinatorial explosion of the output space, which directly effects information gain computation. We, therfore, employ a similar technique as in KONTSCHIEDER et al. (2011a,b) for the reduction of the output dimension: We just use the pixels for which the information gain is computed in the split node. For example, in equation 4.5 we use two pixel values to compute the information gain. New pseudo classes are created as ordered pairs (or triplets) of the pixels' classes and used for the entropy computation of the left split node as follows:

$$H(S^L) = \sum_k \sum_l P(x_{i+s,j+t} = c_k, x_{i,j} = c_l)\ log(P(x_{i+s,j+t} = c_k, x_{i,j} = c_l)) \tag{4.8}$$

where c_k and c_l are the pseudo classes. In the same way the entropy of the right split S^R is computed.

Because the information gain computation is computationally very expensive, we use only up to two million patches which are represented by their center pixel for building the training set. Thus, e.g. in the ECP dataset there are less patches than the number of pixels of four images. Since the distribution of the classes in the images is not uniform and the variability of the facade objects is high, we build the dataset using all training images, but select just a small subset (several thousands patches) from each image.

For creating the dataset, a couple of thousand pixels are selected and patches are created from them. The number of selected pixels from each image is only "almost" the same, since pixels from some classes (e.g., doors) are less frequent

and their number does not exceed a predefined *threshold*. This means that, e.g., facades that do not contain shops or other less frequent objects, have fewer "representative" pixels in the training set. The number of pixels selected from each image is:

$$\sum_{c_i}^{c_i \in Y} min(\text{threshold}, \sum_j \sum_k [I(j,k) = c_i]) \tag{4.9}$$

where [ ] represent the indicator function.

An important parameter of SRFs, is the patch dimension. In this thesis, the patch is always represented as a square with an odd number of pixels in the range of 11×11 to 21×21 pixels. Larger patch dimensions increase the memory as well as the computational complexity.

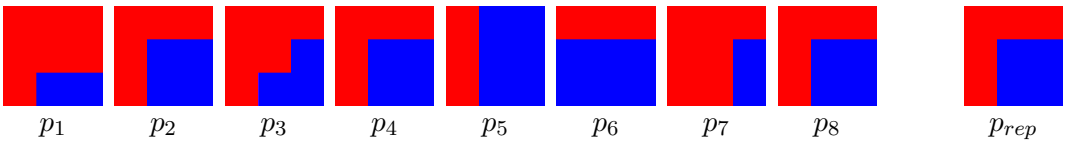


Figure 4.5: Patches in a leaf node p_1, p_2,...,p_8. p_{rep} is the representative patch of the leaf.

In general, decision trees overfit easily. To ensure that our model does not overfit substantially, the stopping criterion of a minimum number of samples per leaf node is applied.

Patches in the leaf node node might have different labelings (Fig. 4.5) and the leaf must output a single patch. Thus, a representative patch for the leaf node has to be selected. We have adopted the technique by KONTSCHIEDER et al. (2011a) where the representative patch is chosen by pixel-wise majority voting (Equation 4.10). The class of each pixel is determined independently from the other pixels of the patch. Just the representative patch is saved in the leaf node.

$$p_{rep}(i,j) = \max_{c_l} \sum_{k=1}^{k \leq n} [p_k(i,j) = c_l] \quad [\,] \text{ represents the indicator function} \tag{4.10}$$

In Figure 4.5, we show an example with two classes $\{red, blue\}$ and patches with dimension 3×3 pixels. Suppose that the eight patches $\{p_1, p_2, ..., p_8\}$ have reached the leaf node. Then, for each position (i,j) $(1 \leq i,j \leq 3)$ the class is chosen as the class with the most votes (Equation 4.10).

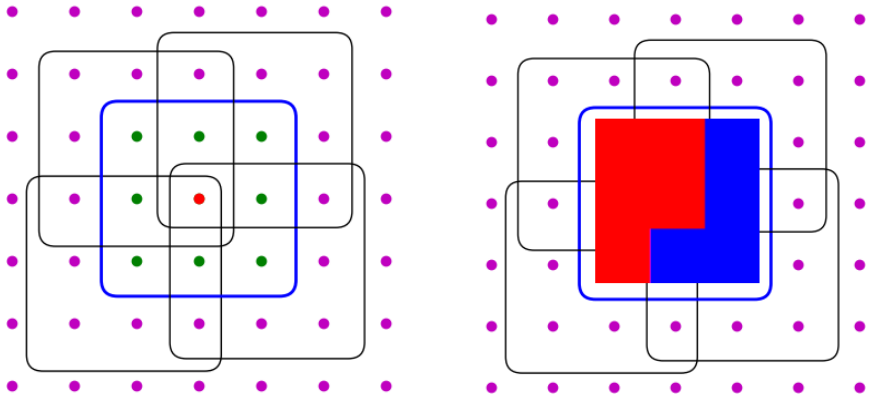


Figure 4.6: a) Each pixel is labeled by multiple patches. In this example each pixel is labeled by nine patches, but to improve interpretability we show just five of them. b) The proposal of the central pixel patch which annotates the neighboring pixels

During testing each central pixel outputs a patch which labels the area around it. This leads to multiple labelings of a pixel (Fig. 4.6). Suppose that the output patch has dimensions $n_p \times n_p$, then each pixel is labeled n_p^2 times. We choose the class label of the pixel by majority voting from all n_p^2 patches.

For selecting a pixel's label the neighborhood has an important role. Furthermore, the patches tend to consist of homogeneous regions leading to a more or less *salt-and-pepper* noise free segmentation. Additionally, we found that SRFs have the capability to learn the local arrangement of the facade objects. E.g., they learn that balconies are always under windows, doors are surrounded by wall, etc.

On the other hand, SRFs suffer from that they cannot capture or learn the global arrangement of the objects on the image, such as a grid of windows. This is because they do not integrate the features from the whole image like CNN architectures integrate in the deep layers.

While results of SRFs are promising, there is still room for improvement by means of optimization, which is discussed in the following section.

4.6 Structured Random Forest Optimization

In this section we describe the SRF optimization method, introduced by KONTSCHIEDER et al. (2011b) which we adopted for our work.

Using the information that each pixel is labeled by several patches and the pixel label is chosen by majority voting, we can check the correctness of the forest output as well as each tree separately. Particularly, we can compare the forest output patch with the final labeling after all patches have voted for each pixel. Since we use an SRF, we can improve the forest by selecting the "best" output in the form of the most appropriate tree for each patch labeling and omitting the others. This is a combinatorial problem, as the best patch decision effects the neighbors. Thus, the best patch selection can be regarded as an optimization problem constrained by the neighbors' "best" patch selection.

An iterative process for this optimization problem is proposed by KONTSCHIEDER et al. (2011b) converging to the "best" patches. The best patch selection is based on the agreement score of two patches (Equation 4.11), which is computed as the amount of pixels with the same class at the corresponding locations.

$$Agree(P_1, P_2) = \sum_i \sum_j [P_1(i,j) = P_2(i,j)] \tag{4.11}$$

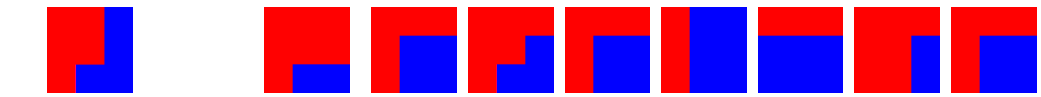


Figure 4.7: Agreement score between the *Labeled patch* and the tree proposals. The third patch has the highest agreement score and is selected as representative patch of the forest.

Suppose we have an SRF F consisting of eight trees t_1, t_2,...,t_8. Each of the trees outputs a patch p_i, $t_i(x) = p_i$ for an input x. In the initial labeling the output for the sample x is created as function of all the forest trees $F(x) = \oplus\{t_1(x), t_2(x), ...t_8(x)\}$, while in the next iteration just the patch with the highest agreement score is selected (Fig. 4.7).

In the example in Figure 4.7 the output of the third tree has the highest agreement score and is selected as representative patch, thus $F(x) = t_3(x) =$

p_3. The same procedure is applied for each pixel and the labeling is updated (Fig. 4.8).

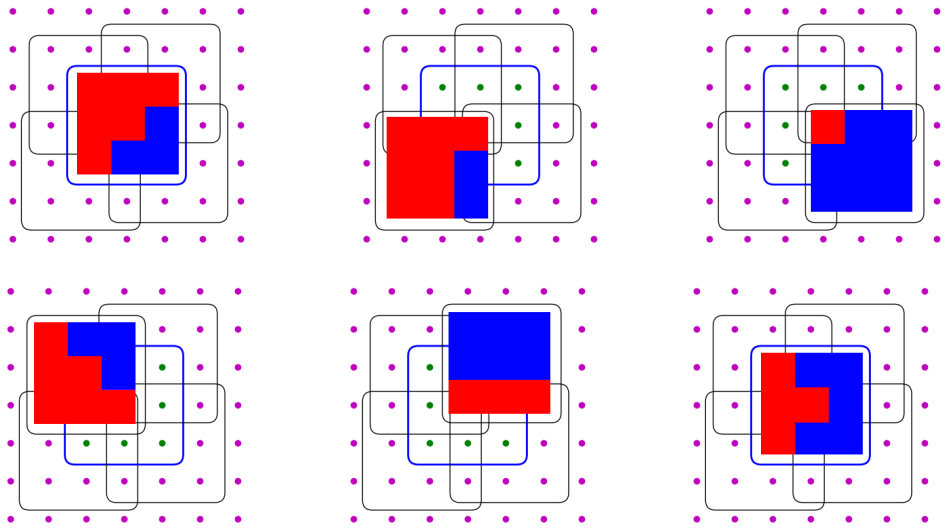


Figure 4.8: The labeling for the image. The input is shown in the upper left and the output in the lower right. The other four graphs show the diagonal elements, while the horizontal and vertical elements have been omitted for the sake of clarity.

In the next iterations (Fig. 4.9), the agreement score is always computed between the new labeling and the trees' output.

As proven by KONTSCHIEDER et al. (2011b), the process converges and the agreement score in each iteration in the complete image does not decrease. Yet, there is a problem with memory consumption, because the tree outputs for each pixel have to be saved. Furthermore, it has a run time complexity of $O(n_p^2 whn_t)$, where w and h are the dimensions of the image, and n_t is the number of trees of the forest.

4.7 Rectangular Fitting

Facade objects such as windows, doors and shops, mostly have rectangular shapes. Thus, we apply rectangular fitting to the optimized SRF labels.

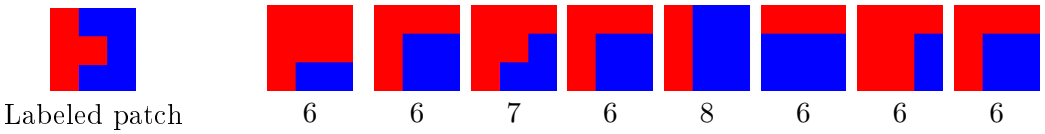


Figure 4.9: Agreement score computation between the labeled patch (left) and the tree proposals. In the second iteration, the patch with the highest agreement score is the fifth patch which is different from the patch of the first iteration (Fig. 4.7) and the result of the labelling of the image (Fig. 4.8).

Our goal for rectangular fitting is to maximize the number of object pixels inside the rectangle and at the same time minimize the number of other pixels in it (Fig. 4.10). In contrast to the method by Cohen et al. (2014) which performs a dynamic programming over all elements using soft and hard architectural constraints, we employ a greedy rectangular fitting without any constraint. The rectangular fitting method is formulated as follows:

$$\underset{x_1,x_2,y_1,y_2}{\arg\min} \sum_{i=x_1}^{x_2} \sum_{j=y_1}^{y_2} I[L(i,j) \neq o_c] + \sum_{i=x_i-k}^{x_2+k} \sum_{j=y_1-k}^{y_2+k} I[L(i,j) = o_c \wedge (i,j) \notin R_{x_1,y_1,x_2,y_2}] \tag{4.12}$$

where R_{x_1,y_1,x_2,y_2} is a rectangle with top left corner at position (x_1, y_1) and the bottom right corner at position (x_2, y_2)

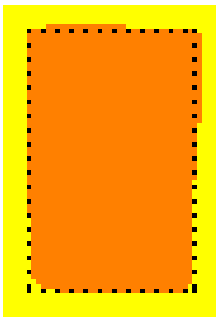


Figure 4.10: Rectangular Fitting

For optimization, we first identify potential rows and columns that may contain the starting or ending edge of the intended object class. For a more concrete explanation, let the intended object be a window (Fig. 4.11). In each direction we identify all potential starting and ending rows as well as columns

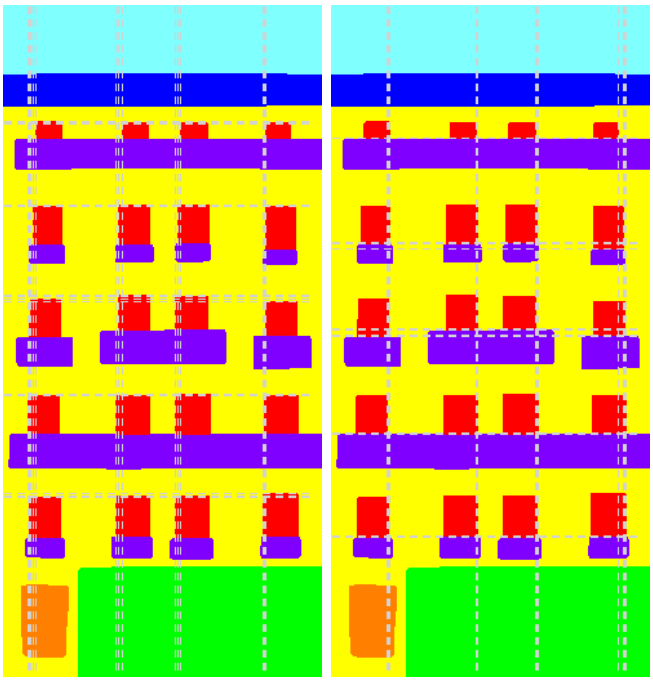


Figure 4.11: Initial candidate generation

by counting the transitions from window to other objects, e.g., wall, balcony, roof. We compute the following equations: 4.13 for start row transition (SRT), 4.14 for end row transition (ERT), 4.15 for start column transition (SCT) and 4.16 for end column transition (ECT).

Suppose we have an annotated image I with r-rows and c-columns and $I(i,j)$ denotes the pixel label at the position (i,j), then the number of start and end transitions for each row and column are computed as follows:

$$SRT(i, class) = \sum_{j=1}^{j \leq c} ([I(i,j) = class \text{ and } I(i,j-1) \neq class]) \tag{4.13}$$

$$ERT(i, class) = \sum_{j=1}^{j \leq c} ([I(i,j) \neq class \text{ and } I(i,j-1) = class]) \tag{4.14}$$

$$SCT(j, class) = \sum_{i=1}^{i \leq r} ([I(i,j) = class \text{ and } I(i-1,j) \neq class]) \tag{4.15}$$

$$ECT(j, class) = \sum_{i=1}^{i \leq r} ([I(i,j) \neq class \text{ and } I(i-1,j) = class]) \tag{4.16}$$

All rows and columns that have transitions are further considered for generating the potential windows.

First we determine the intersections of the SRTs and SCRs from which we "search" for the top-left window corners (Fig. 4.11). We discard all intersection points that do not have any transition points in their right and bottom segments, because they cannot lead to an optimal rectangle. In a similar way we find the potential bottom-right corner by "searching" for the end row and column intersection.

After locating potential top-left and bottom-right corners, we optimize the resulting rectangles (Equation 4.12). Finally, we generate candidates for windows. The range of window widths and heights is known from the training set and each pair of top-left and bottom-right corners is generated based on this range. In the same way, we determine optimal rectangles for other rectangular facade objects (balconies, shops, doors and roof).

This algorithm can be performed in real time. Its worst case complexity is $O(w^2h^2)$, but there is not a large number of candidates for each object and all "checking" computations are performed in $O(1)$ time complexity due to memoization (an optimization paradigm which saves computation results and uses them in further computations). Thus, the average complexity is $O(wh)$, with w width of the image and h the height of the image.

Chapter 5

Window Refinement

In this chapter, we present our work on the refined delineation of windows as an extension of the previous chapter. Our system is extended with a component which refines and segments a window into transoms / horizontal parts of the frame, mullions / vertical parts of the frame and the window area (glass). For processing, we term the horizontal parts of the window frame as well as the transoms as "horizontal bars" and the vertical parts of the window frame as well as the mullions as "vertical bars" (Fig. 5.1). They are beside the glass area the most important components of windows.

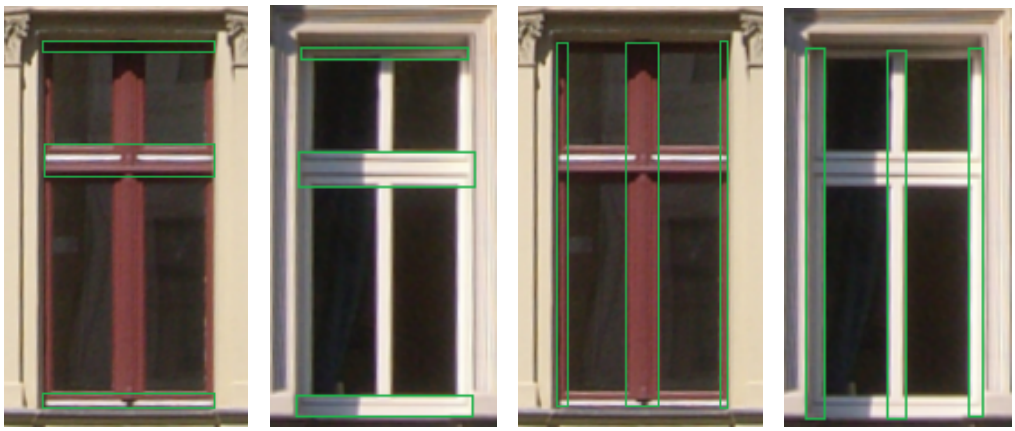


Figure 5.1: Horizontal and vertical bars of the windows. In the first two images on the left the horizontal bars and in the two images on the right the vertical bars of the windows are shown.

In the first steps of our system, we do not distinguish the vertical and the horizontal bars, since the intersections belong to both classes. Furthermore, using just one class makes it easier for the classifier, because horizontal and

vertical structures have the same texture and color. Thus, we decided to treat both structures as a single class and we name it the (window) frame class, during the initial segmentation using deep learning.

Because there is no published dataset for this task, we have built our own with hundreds of high resolution annotated images of windows.

In further steps, we use the assumption that windows are arranged in grid structures to find the correct position of the frames. In the end, we fuse this information and the output of the segmentation network to produce the final labeling for the window. The complete pipeline of the system is presented in Figure 5.3. The output consists of a description of the number and the position of the vertical and the horizontal structures for each window.

Although there is no other system to compare with, we can at least state that the pipeline produces results of high quality. They do not depend on the arrangement of the windows on a grid, although the pipeline uses it if it exists.

The chapter is organized as follows: We first introduce the dataset that we have built and then describe each component of the system. This starts with the object detector as well as the semantic segmentation component and is followed by optimization and model fitting.

5.1 Window Refinement Dataset

There does not exist any dataset with labeled transoms / horizontal bars and mullions / vertical bars. Since current publicly available datasets (Section 3.1) have a low resolution, we could not use them for the annotation of window frame, transoms and mullions. We, thus, have built our own (Section 4.1) and labeled windows, the wall, transoms and mullions as well as all frame parts. We annotated more than 50 images containing 1478 windows for training and four images containing 164 windows for testing (Fig. 5.2). Our images, have a high resolution and are rectified by mapping them on the 3D model derived from them.

Particularly, our images have a dimension of larger than 1500×1500 pixels. They stem from the German cities Munich, Mühldorf and Berlin. We have used the approach of Ley and Hellwich (2016) for facade image rectification and all images are annotated manually without any automation.

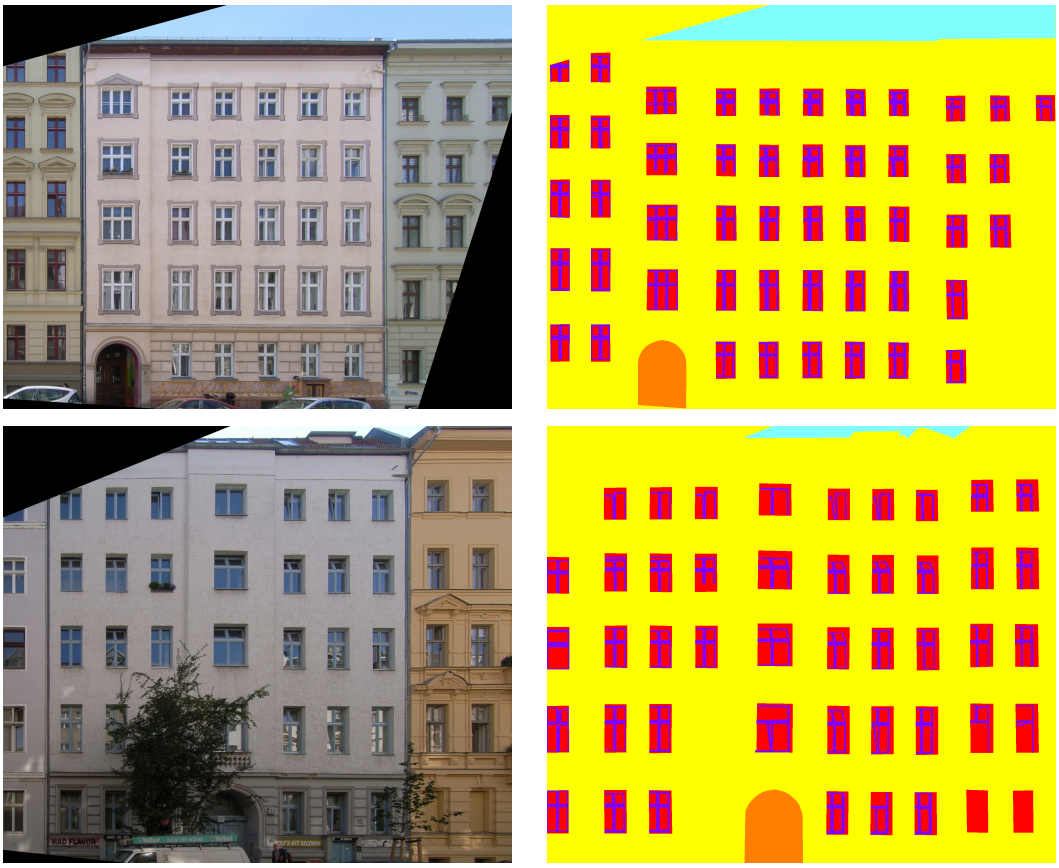


Figure 5.2: Images from the dataset with annotated vertical and horizontal bars

5.2 Detection of Horizontal and Vertical Bars

In this thesis, we have limited the detection of vertical and horizontal bars to rectified facades. To simplify window delineation, we employ an object detector for windows and then just segment the window objects (Figure 5.3). By this means, we make facade segmentation independent of window delineation.

For the delineation of windows, we have created a hybrid approach which consists of deep learning and optimization components. We use two deep neural networks for window detection as well as for semantic segmentation of the detected windows. The latter labels each detected window with three classes: (vertical and horizontal) bars, window area (glass) and wall.

In order to correct and improve the segmentation, we use clustering. The segmented windows are clustered based on their location and dimension and each window is assigned to a horizontal and a vertical cluster. Clustering helps to improve the position of vertical and horizontal bars.

After we have determined the position of the vertical and horizontal bars, we derive their exact dimension using dynamic programming and model fitting. Finally we label the window.

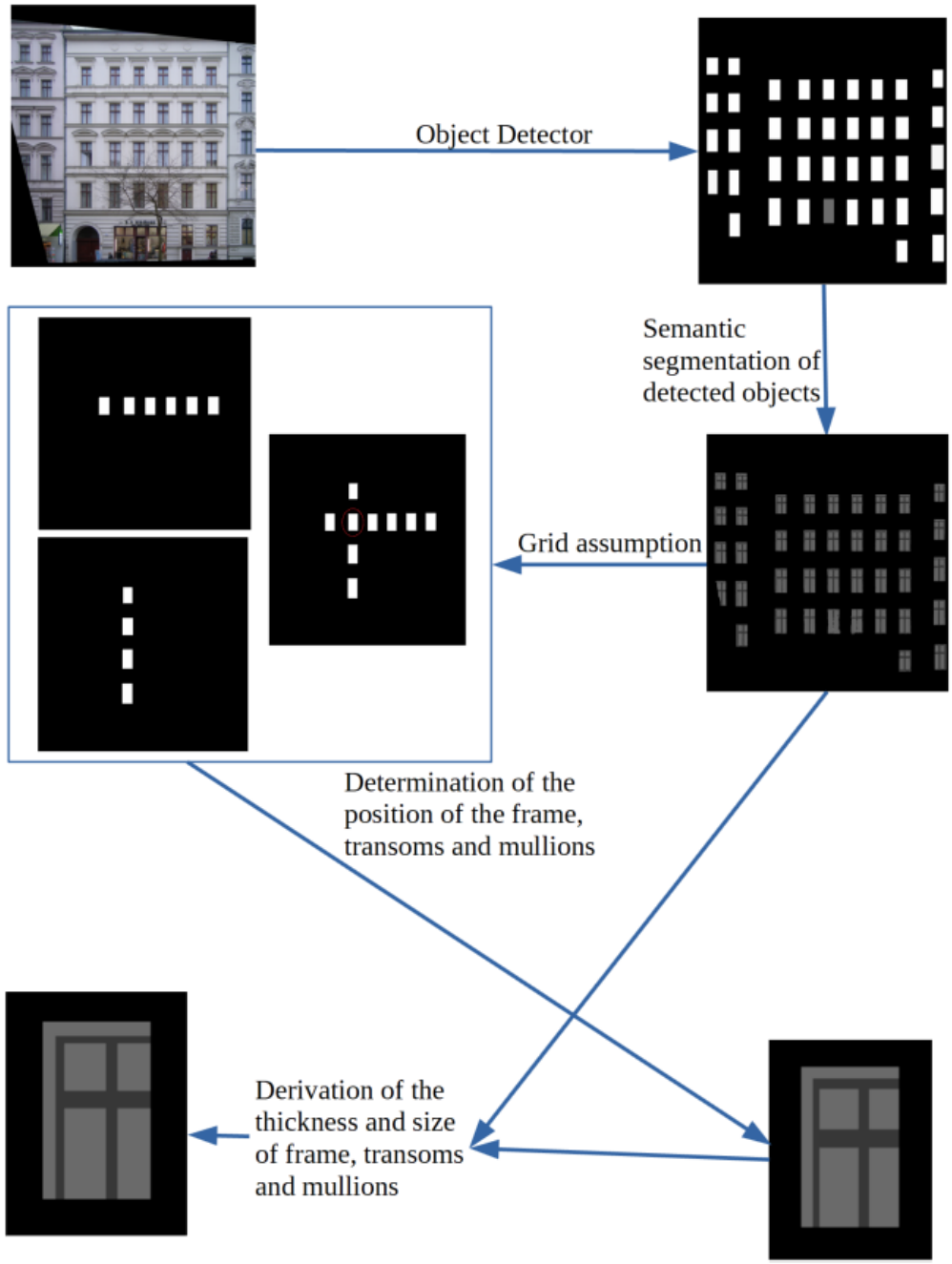


Figure 5.3: The architecture of the proposed pipeline for window segmentation consisting of object detector, semantic segmentation, determination of the number of horizontal and vertical bars of each window as well as their location based on the grid, and finally derivation of the exact position of the vertical and horizontal bars.

Semantic Segmentation of the Windows

We have created a window detector to simplify the segmentation. Thus, the network only has to learn how to segment windows, and it always receives the same structure as input. Furthermore, the network does not have to consider other classes like door, shop or sky. Because of its high detection rate, the object detector has the same architecture as in the facade segmentation task (Chapter 4).

We, use a pretrained Pyramid Scene Parsing Network (PSPNet) by Zhao et al. (2017) for window segmentation. This architecture has been chosen since it is one of the best performing methods in benchmarks and it is suitable for our problem. A particular reason is that some layers of the network do not use pooling. This fits for our task since the network can learn the structure of the window and at the same time capture thin vertical and horizontal bars. Furthermore, the chosen network allows to acquire information about the hierarchy.

We use pretrained weights from Resnet-50 by He et al. (2016) for the segmentation and the object detector network. For training and inference of the semantic segmentation network, we add a margin around the windows, such that the network sees the complete window as well as a part of the wall around it. By this means, all images have the same dimension in both regimes, training and testing. For training, we use two types of images: In one type there are several facades in a single image and in the other type there is just one facade. The windows in the second type are around two times bigger in terms of pixel dimension than the windows in the first type. Thus, the first type of windows are re-sized such that they are in the range of size of the second type.

Adding a part of the wall around the windows also allows to perform data augmentation. Furthermore, because of it the network correctly captures the border between windows and wall (Fig. 5.4). We can randomly choose the offset on each side, resulting in a dataset with a huge number of samples which reduces over-fitting. Similarly, we always add a part of the wall around a window also during inference such that the window images look similar to the training images.

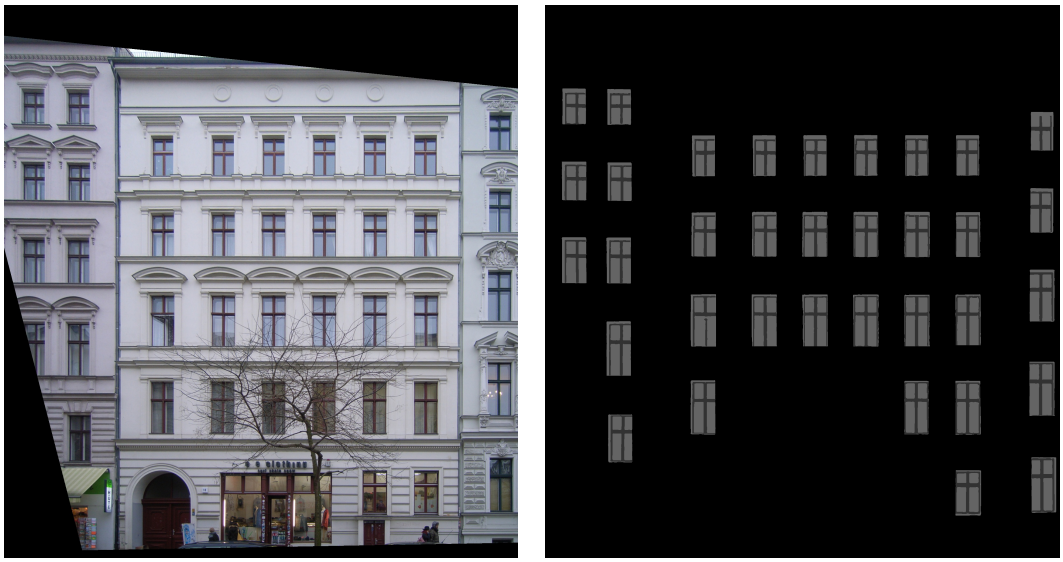


Figure 5.4: Results of Pyramid Scene Parsing Network (PSPNet) for window segmentation. Left: input image, right: semantic segmentation of the detected windows

Horizontal and Vertical Clustering

Since windows are often arranged on a grid, we can use this information for a better delineation of the horizontal and vertical bars. For this, we use two separate clustering principles: According to the first, windows in the same cluster should have the same width and be on the same column. For the other type, the windows should be located on the same row and have the same height (Fig. 5.5). For images taken from the ground, the horizontal frame parts' and transoms' height (thickness) and visibility are the same for each row regarding the viewing angle and we call aligned windows horizontal clusters. The same idea is applied for windows on the same column and with the same width to find the positions of the vertical frame parts and mullions for the vertical clusters. Each window is assigned to no more than one vertical and one horizontal cluster. Since the object detector does not give very precise locations of a window, we use a tolerance of 15 pixels (images have dimension 1500×1500 pixels) in terms of object dimension and location.

Because we employ rectified images, we can align the detected objects of each horizontal and vertical cluster such that all are on the same horizontal or

vertical line, respectively. By this means, we correct errors made by the object detector.

From the object detector we get the position of each detected window and we use them for the clustering. We perform a greedy creation of the clusters. E.g. for the vertical clustering, we start without any cluster. The first detected object creates the first cluster and the cluster is represented by the vertical position of the detected object. Then, the other windows are introduced one by one to the clustering algorithm. If the vertical position of a window differs from all current clusters by more than the predefined threshold of 15 pixels, a new cluster is added. Else, the window is added to the cluster for which it has the smallest difference for the vertical position. The same methodology is applied for the creation of horizontal clusters.

We regularize all windows in the vertical clusters to the same width and all windows in the horizontal clusters to the same height.

Figure 5.5: Window clustering. Left: Vertical clusters, windows are on the same column and have the same width. Right: Horizontal clusters, windows appear on the same row and have the same height.

Each window of a horizontal or a vertical cluster influences the position of the vertical bars and horizontal bars, respectively, thus, it votes for the mullions or transoms representative for the clusters. For simplicity, we just

describe the procedure for the horizontal clusters: Each window produces two types of votes, one for the transoms and the horizontal parts of the frame and one for determination of the horizontal borders between window and wall (the height of the window).

Formally, the determination of the position of the transoms and the horizontal parts of the frame is based on the horizontal cluster $C = \{w_1, w_2, ..w_k\}$ with k-window samples with the same dimension $m \times n$ (Fig. 5.6). Furthermore, the position is determined in a "window" W_{pro} with always the same dimension, in our application 200×150 pixels. A pixel at position x, y of window i is denoted as $w_i(x, y)$. It can only have one of two values: 1 if the pixel is labeled as horizontal bar and 0 otherwise. Each pixel x, y of window $w_i \in C$ casts a vote for the position $W_{pro}(x, y)$ if $w_i(x, y) = 1$. The final proposal has the value:

$$W_{pro}(x, y) = \sum_{i \in C} w_i(x, y), \quad \text{where } 0 \leq x < 200 \quad \text{and } 0 \leq y < 150 \tag{5.1}$$

After this, we apply a threshold changing each pixel value of W_{pro} smaller than $0.5k$ to 0.

$$W_{pro}(i, j) = \begin{cases} 0, & \text{if } W_{pro}(i, j) \leq \frac{k}{2} \\ 1, & \text{otherwise} \end{cases} \quad \text{where } k = |C| \tag{5.2}$$

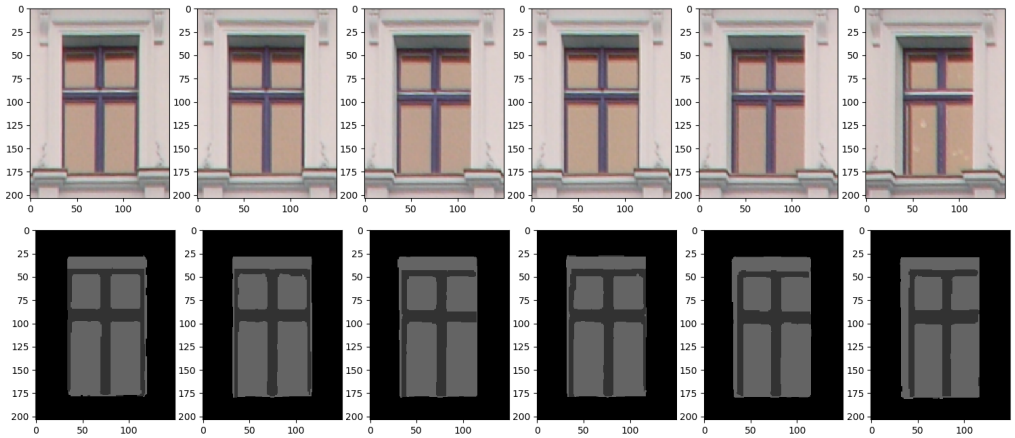


Figure 5.6: Windows of a horizontal cluster. Top: Images, bottom: segmentation by PSPNet.

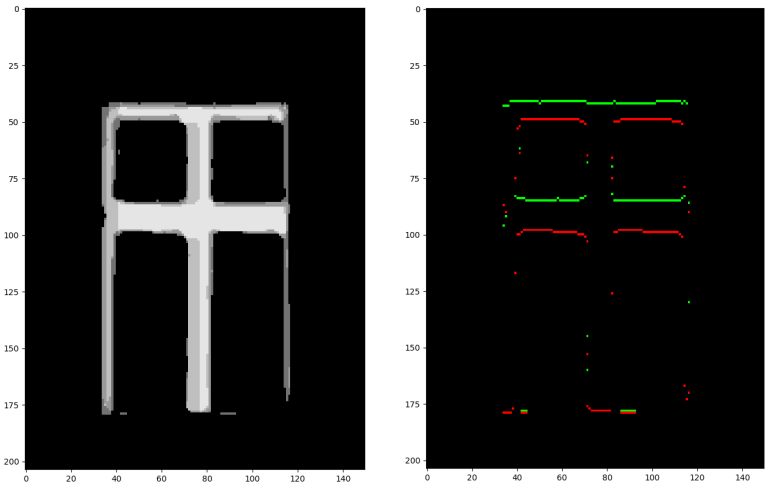


Figure 5.7: The left image presents the voting of each window of the horizontal cluster given in Figure 5.6. Brighter pixels mean more votes. The right image shows the gradient of the left image. Green pixels represent the transition from a non-horizontal bar to a horizontal bar pixel, indicating the start of a horizontal bar. The green pixels are pixels with a transition from a horizontal bar to a non-horizontal bar pixel, which represent the end of a horizontal bar. The gradients help to reduce the search space for finding plausible horizontal bars.

From W_{pro} we find the final position of the transoms and horizontal parts of the window frame. In a similar way, we post-process the proposal for the height of the window.

After we have found the transoms and the horizontal part of the window frame from the cluster, we use Dynamic Programming (DP) similarly to rectangular fitting (Section 4.7) to determine the position and dimension of the transoms and horizontal parts. We search for two parallel lines which fit as many transom pixels in their area as possible while containing as few as possible non-transom pixels. We perform this in a greedy way starting from the bottom of the window and going upwards. To reduce the search space for finding the optimal bars, we use gradients of the window image (Fig. 5.7). The determination of the horizontal part of the frame is mapped to the derivation of the height of the window. This step is easier than finding transoms since there is just one window in each sample $w_i \in C$.

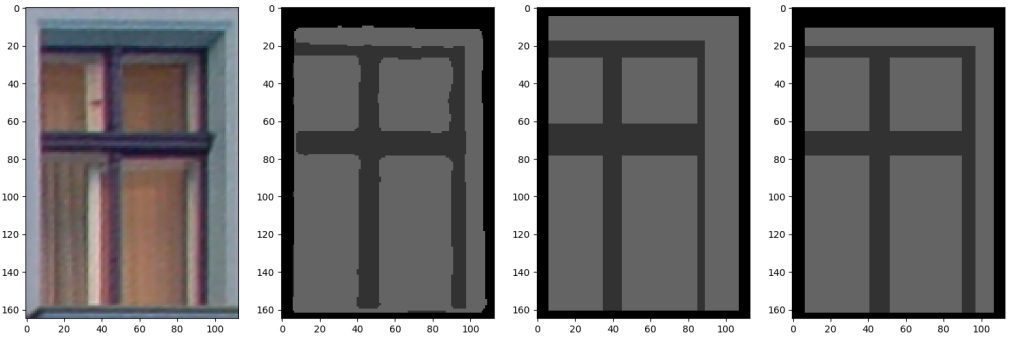
Because the images are taken from the ground, horizontal bars in the upper part of the window are mostly visible. Yet, some of the clusters have just one window and sometimes the semantic segmentation module does not detect a horizontal bar in the upper part of the window. In this case, we add a horizontal bar using the window size and the ending position of the vertical bars (Fig. 5.8).

Finally, for each window the clustering outputs a set of values relative to the object, which is mapped back to the facade image. For example, for the window in Fig. 5.9 the following output is obtained: The window starts at the position $(4, 5)$ and ends at the position $(160, 106)$. It contains horizontal bars at rows $[18, 26]$ and $[62, 78]$ as well as vertical bars at columns $[35, 44]$ and $[84, 88]$.

Figure 5.8: Correction of the upper bar when PSPNet fails to find it.

Delineation of Transoms and Mullions

After detecting windows, transoms / horizontal parts of the frame and mullions / vertical parts of the frame, we first localize the window and then the other parts taking into account the output of the Semantic Segmentation network. Clustering proposes the general structure of the windows in each row. Yet, each window slightly varies from the candidate image derived in the previous phase. Additionally, due to errors of the rectification, windows are not perfectly aligned. In this final step, we, thus, mostly thin or extend mullions, transoms and horizontal / vertical parts of the frame produced by clustering, to match the output of segmentation.



a) b) c) d)

Figure 5.9: Results of window delineation pipeline. a) Input image, b) Segmentation produced by PSPNet c) Structure proposed by clustering, d) Segmentation produced by combining b) and c)

Clustering captures the skeleton of the window and gives us an approximate location of the bars (Fig. 5.9c). During delineation this information is used to find the best start and end position of the window and the bars by searching the positions in the neighborhood that lead to an optimal rectangular fitting. For vertical and horizontal bars we search for the optimal positions in the sense that they contain as many as bars pixels.

For each of the proposed bars from clustering we search for the optimal bar in the PSPNet-segmented window by using DP under the constraints that it fits as many as possible bar pixels, excludes other pixels and has a thickness of at least a predefined number of pixels.

Finally, the window area should fulfill two constraints: It should include all horizontal and vertical bars and should have the optimal position computed by rectangular fitting.

For the example presented above, the final values (Fig. 5.9) of the start and end position of the window become $(10, 6)$ and $(161, 106)$, respectively, since for this values rectangular fitting achieves the optimal value. The optimal horizontal bars are at rows $[20, 25]$ and $[65, 78]$ and the vertical bars at columns $[40, 51]$ and $[89, 95]$.

Chapter 6

Experiments and Evaluation

This chapter presents the results for our pipelines for semantic segmentation and window refinement, which are described in Chapters 4 and 5, respectively. We show that our pipeline for facade segmentation outperforms all currently published methods for small datasets and performs better than or on a par with the current published methods for larger datasets. We have evaluated our approach for the three datasets ECP Paris, Graz (Section 3.1) and the dataset generated by us with German buildings (Section 4.1). To demonstrate the importance of each component of the pipeline, we have evaluated the pipeline algorithms incrementally.

For evaluation we have used classification accuracy as measurement score:

$$\text{classification accuracy } = \frac{tp + tn}{tp + tn + fp + fn} 100\% \tag{6.1}$$

with tp – true positive, tn – true negative, fp – false positive, fn – false negative.

In image segmentation, classification accuracy represents the ratio of correctly labeled pixels to the numbers of all pixels in the image. We present classification accuracy as percentage. As all current publications on facade segmentation have used this evaluation methodology, this allows us to compare results with them.

For window refinement and quantification of the effect of the pseudo depth map there is only the German dataset available and we evaluate our pipeline just on this dataset. Particularly, we use only the window regions to evaluate our algorithm on window refinement.

The evaluation of the facade segmentation pipeline is performed on the ECP Paris and Graz dataset.

6.1 Facade Segmentation

The two datasets, ECP Paris and Graz are separated in two subsets, the training set and the testing set. We have decided to use 80% of the images for training and 20% for testing.

The ECP Paris dataset is annotated with seven classes: balcony, door, roof, shop, sky, wall and window as well as the void pixels which do not correspond to any of the previous classes. The Graz dataset is annotated with four classes: door, sky, wall and window. In both datasets, approximately half of pixels is from the wall class. The distribution of the classes of the datasets is given in Table 6.1.

Class	ECP	Graz
Door	5.9%	13.1 %
Shop	8.7%	/
Balcony	11.3%	/
Window	20.2%	28.4%
Wall	44.0%	55.1%
Sky	5.7%	34.4%
Roof	4.2%	/

Table 6.1: Distribution of the classes in the datasets. Approximately half of the surface of the facade images is covered by wall.

Structured Random Forest

In this part we evaluate the construction of the Structured Random Forest (SRF). We use several features consisting of color channels, Histogram of Oriented Gradients (HOG), statistical features and location (Sections 2.1 and 4.4), object detectors based on integral channel feature detector Dollár (2017), Dollar et al. (2009) and Local Binary Patterns.

From the quantitative results (Table 6.2) we can see that for the ECP Paris dataset SRF achieves an accuracy of 88.9% mostly due to the low performance for doors and windows. For the Graz dataset already SRF alone achieves an accuracy of almost 92%. In the Graz dataset, the evaluation is low for the window class with a classification accuracy of less than 80%. The most infor-

mative features are vertical position, followed by HOG features and difference between pixel value and the median (mean) of the row.

Class	Results
Door	**56.6**
Shop	95.3
Balcony	86.4
Window	**70.4**
Wall	91.4
Sky	95.3
Roof	93.2
Overall	88.9

ECP dataset

Class	Results
Sky	86
Window	**79**
Door	91.8
Wall	96
Overall	91.9

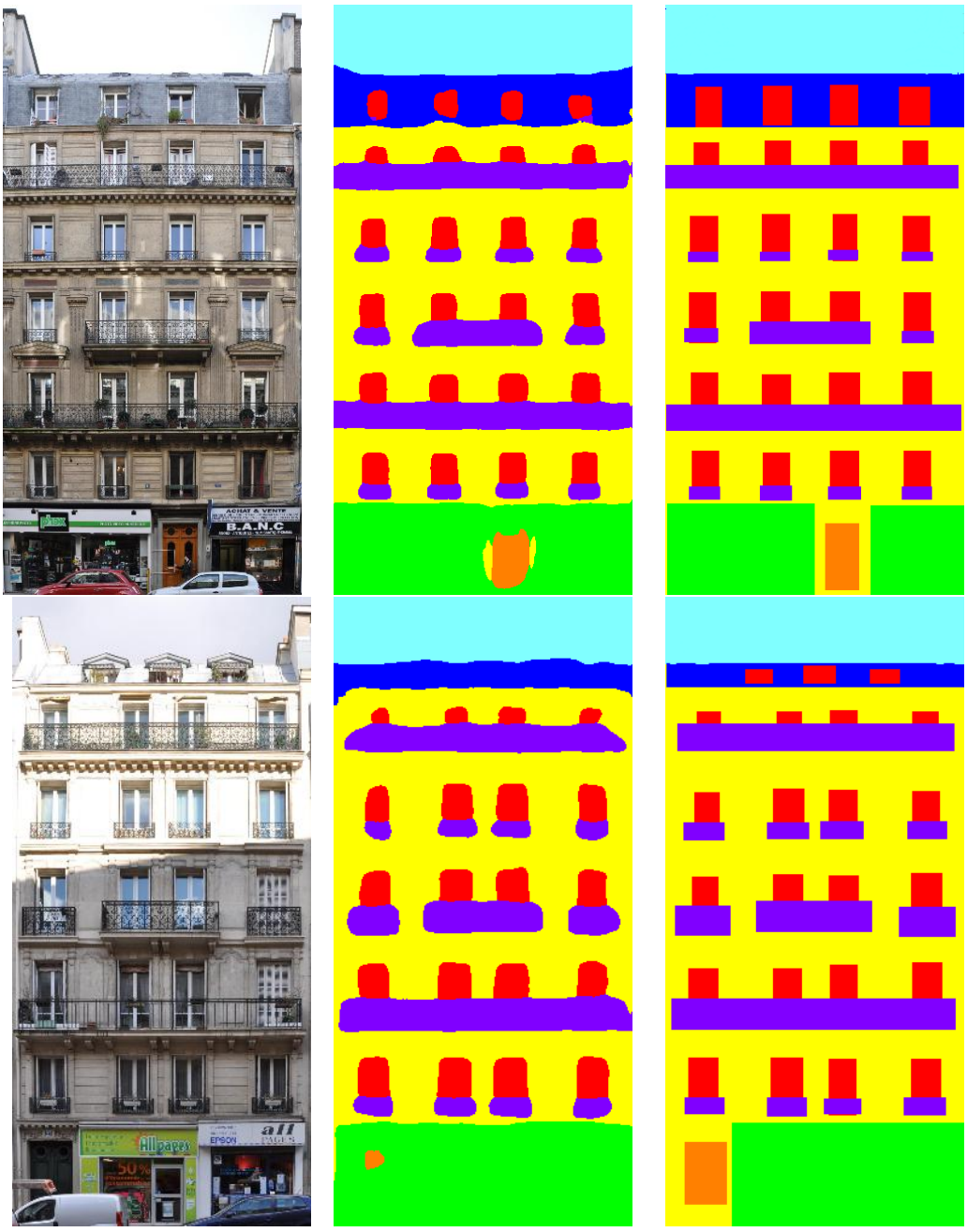
Graz dataset

Table 6.2: Accuracy for SRF only. Left: results of the SRF for the ECP Paris dataset. Right: results of the SRF for the Graz dataset.

Qualitatively, we can see (Figs. 6.1 and 6.2) that the SRF has created homogeneous regions and each window and balcony on the wall are detected. Furthermore, large regions like shop, sky and roof are very well defined. Yet, for the ECP Paris dataset doors are not detected and the area around them is not labeled as wall, but as shop. Further analysis showed that this kind of structure is never seen in the training set and, thus, the algorithm had problems learning the structures relating door, wall and shop.

Also, the corners of the objects are not sharp, which is due to the voting process in the leaf node and its neighborhood (Section 4.5) leading to blurred corner.

Figure 6.1: Results for the Graz dataset using basic features



Input Image | Result for SRF | Ground truth

Figure 6.2: Results for ECP Paris dataset using basic features

Results presented in the tables and images above are after feature cleaning. Initially, we had also used several Local Binary Pattern (LBP) features. Yet we empirically found that they have a low information gain compared to other features. This has been particularly problematic as we use Extremely Random Trees (Geurts et al., 2006), which are sensitive to the number of features, since they choose the features in each split node randomly, resulting in a lower probability for features with high information gain when there are many features. This means that if we have a high number of features with small information gain and some with high information gain, the accuracy of the algorithm drops compared to the algorithm which uses only high information gain features.

Because of all this, we have performed feature cleaning where we have removed the Local Binary Patterns. Results without cleaned features were published in Rahmani et al. (2017).

Inspecting the trees we found that the most frequently used split function in the first three (tree) layers is:

$$h(x(0,0),\theta_1) = h(x(0,0),(k,\tau)) = \begin{cases} 1, & \text{if } x(0,0)_k \geq \tau \\ 0, & \text{otherwise} \end{cases} \tag{6.2}$$

where $x(0,0)$ is the center of the patch. This function covers more than 50% of the nodes in the first layers. Going deeper, the distribution of the functions tends to become uniform, but still the functions with fewer parameters tend to be used more often. Since the search space for a function with a higher number of parameters is larger and finding the optimal parameters is, thus, harder, it needs more tries to find a good solution.

Similarly, in the first layers of the tree the split nodes chose functions with less pixels since they have higher information gain, because as we have just 7 classes (4 classes in the Graz dataset). If we use two pixels, there are 49 combined classes (16 combinations for the Graz dataset) and it is harder to find a combination that has a higher information gain than split functions with fewer parameters.

Furthermore, during training the first layers the dataset is split mostly based on the center pixel, while in the deeper layers the functions with more arguments (pixels) are more frequent. We interpret this, that in the deeper layers the SRF finds the borders as well as the corners of the facade objects.

The *vertical position* is the most used feature covering 65% of the first two layers of the trees. It is followed by HOG features representing more than 20% of the nodes. The *vertical position* feature in the root node distinguishes the lower part from the upper part, since the lower part consists of shops and doors and very few windows and wall pixels while the upper part contains the other classes. This split, therefore, results in the highest information gain.

The Textonboost features do not appear in the root and very rarely in the second layer of the trees. In the deeper layers we see a higher usage of all the features with a tendency towards a uniform distribution with slightly higher usage of a few features with a very high information gain.

Regional Proposal Network

We use object detectors to find regions with specified object classes. For the ECP Paris dataset we have built detectors for balconies, long balconies, windows, doors, roof and shop. For the Graz dataset we have devised detectors for windows, doors and sky. The features created from the object detectors (Section 4.3) are stacked on top of the features described in the previous section.

Class	Basic SRF	RPN
Door	56.6	89.1
Shop	95.3	95.5
Balcony	86.4	89
Window	70.4	77.3
Wall	91.4	92.9
Sky	95.3	97.1
Roof	93.2	94.9
Overall	88.9	91.5

ECP dataset

Class	Basic SRF	RPN
Sky	86	89.9
Window	79	80.6
Door	91.8	93.7
Wall	96	96.1
Overall	91.9	92.6

Graz dataset

Table 6.3: Accuracy for SRF using Regional proposal networks (RPN)

The additional features improve the overall results for the Paris dataset by more than 2.5% and for the Graz dataset by 0.7%. The large improvement for the Paris dataset is a result of the huge improvement for specific classes, especially for door (+**32.5%**) and window (+**6.9%**) (Table 6.3). Additionally,

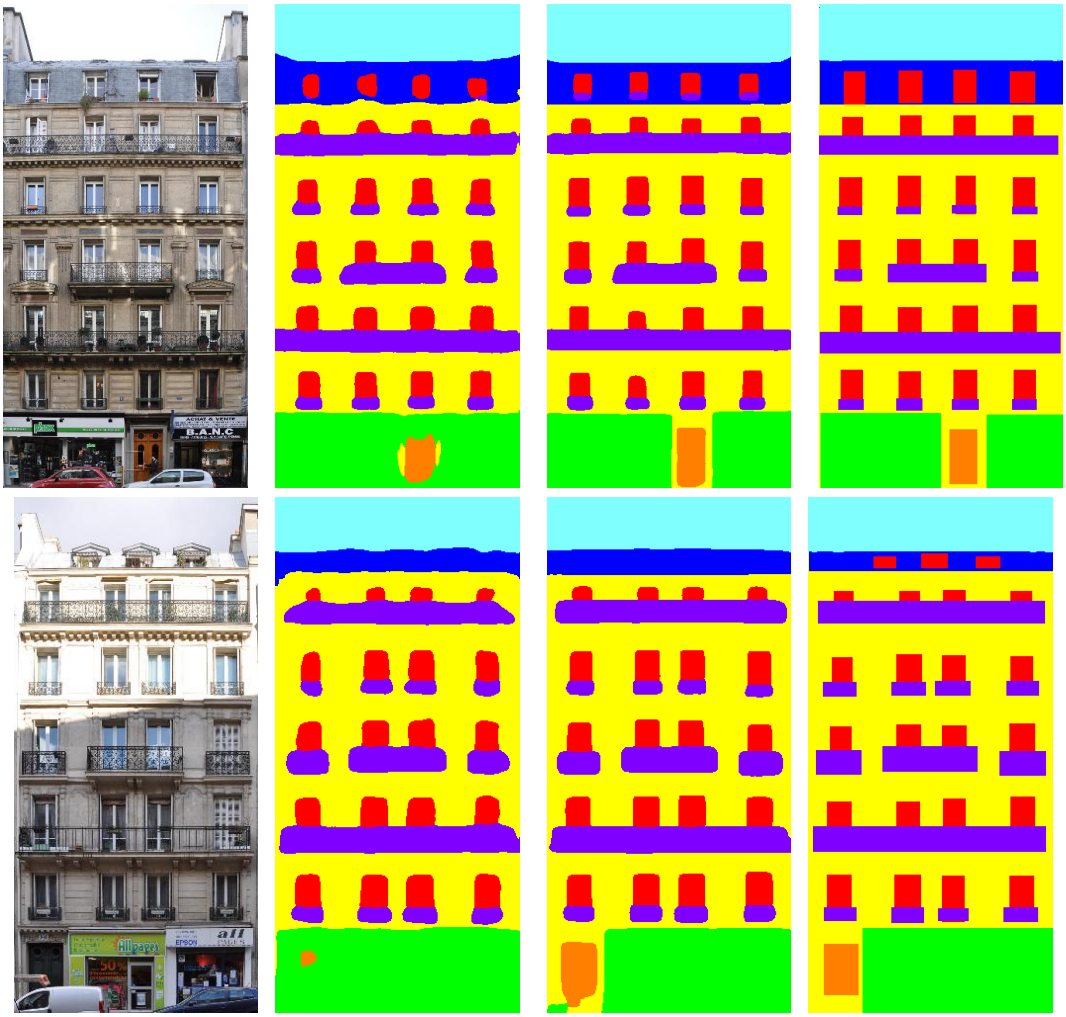


Figure 6.3: Results for the ECP Paris dataset using RPN.

Figure 6.4: Results for Graz dataset using RPN.

we see an improvement in the door class for the Graz Dataset achieving a classification accuracy of nearly 94% (Table 6.3).

Also qualitatively, the results have improved compared to SRF without RPN (Figs. 6.3 and 6.4). The edges and corners are sharper and the images are almost noise free. The structure of door-wall-shop in the Paris dataset could now be learned. Similar results like for the ECP Paris dataset were obtained for the Graz dataset.

As without detectors, in the upper tree layers the *vertical position* is used for more than 65% and HOG features for 17% of the split functions. Similar like in the previous section, the root node divides the image in the lower part and the upper part. In the next layer, the object proposal features are immediately employed in one split node, to distinguish shop and doors in the lower part of the image. In the other node, where the pixels above the shop are traversed, the parameters for the best split dividing sky and roof are determined.

For the lower layers there is a uniform usage of all features, except the RPN features, which are used two times more frequently than other features. The most frequently used feature is window proposal followed by shops, roof and balconies. In contrast to SRF without RPN described above, where the *vertical position* is the most frequently used feature in all layers, each of the region proposal features is used more often than it.

We can, thus, empirically see the importance of the region proposal features.

Optimization with SRF

Quantitatively, optimization brings only small improvements to the pipeline (Table 6.4). The overall accuracy for the ECP Paris dataset is increased by 0.1% and for the Graz dataset around 0.2%. In our initial results published in (Rahmani et al., 2017) for which we did not use the RPN and also without feature cleaning, the optimization improved the overall accuracy by more than 0.5%.

Looking at the results for each class, we can observe a drop of accuracy for some classes in the ECP dataset. The optimization lowers the accuracy of the classes door, wall and roof. On the other hand, for the Graz dataset we have an improvement for every class.

Optimization is the step that consumes the most computation power and memory. If we do not want to store the output of the trees, we must traverse each tree which needs computations (time) and makes the process slower. In our implementation we, thus, store the patches for each pixel trading runtime against memory.

Class	Base	RPN	OPT
Door	56.6	89.1	89.0
Shop	95.3	95.5	96.0
Balcony	86.4	89	90.0
Window	70.4	77.3	78.0
Wall	91.4	92.9	92.6
Sky	95.3	97.1	97.2
Roof	93.2	94.9	94.8
Overall	88.9	91.5	91.6

ECP Paris dataset

Class	Base	RPN	OPT
Sky	86	89.9	90.0
Window	79	80.6	80.9
Door	91.8	93.7	94.1
Wall	96	96.1	96.3
Overall	91.9	92.6	92.8

Graz dataset

Table 6.4: Accuracy after optimization

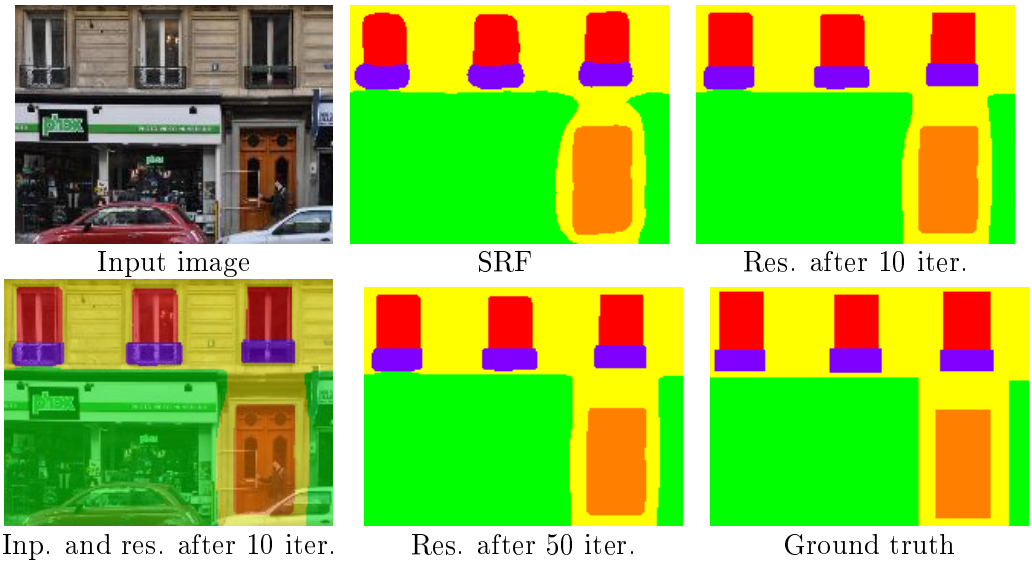


Figure 6.5: Effect of optimization

Qualitatively, the optimization removes the noise from the images and creates more homogeneous regions (Figures 6.6 and 6.7). Additionally, it produces

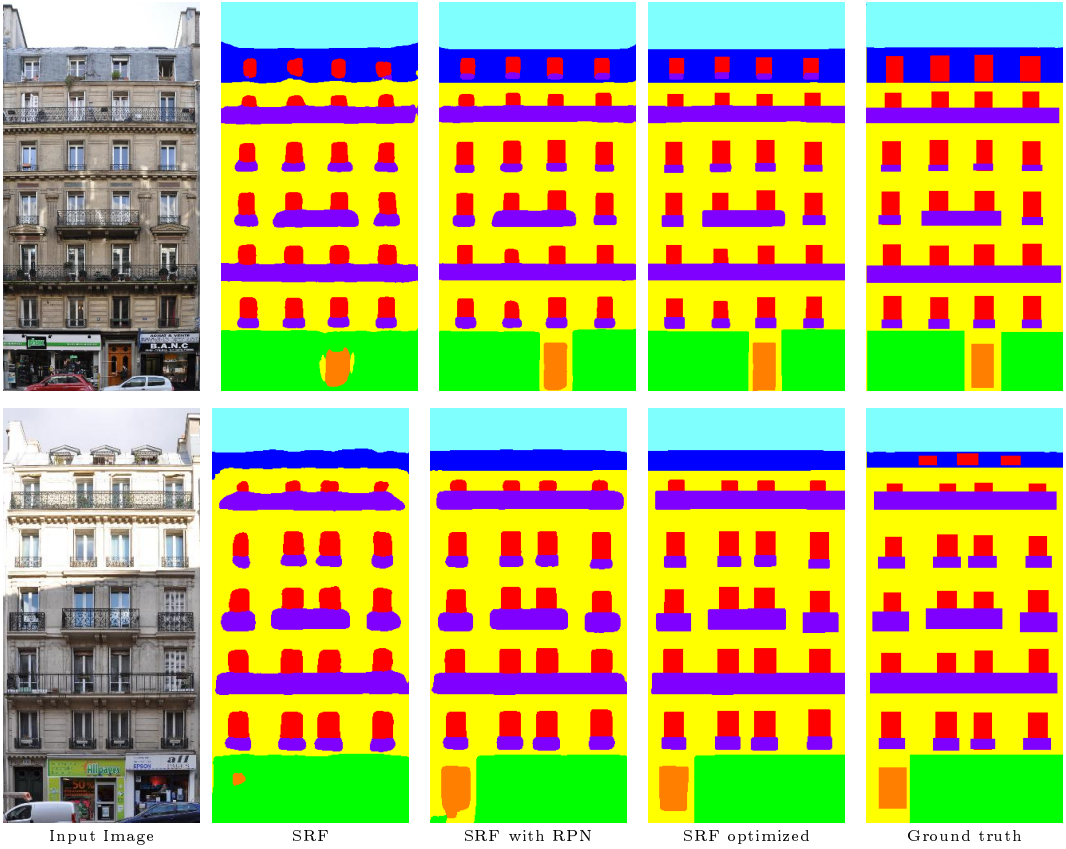


Figure 6.6: Results for the ECP Paris dataset after optimization.

sharper edges and corners (Fig. 6.5). These improvements are due to the selection of the patches from the Random Forest. The representative patch is the output of a single tree and the patch looks very similar to a patch of the training set (Section 2.4.

The inherent object generalization transforms arc-shaped windows into rectangular windows, since most of the windows are of rectangular shape. Furthermore, after the optimization small regions representing noise mostly disappear.

Noise cleaning and edge sharpening are beneficial for rectangular fitting since they reduce the number of potential grid positions (see Section 4.7)

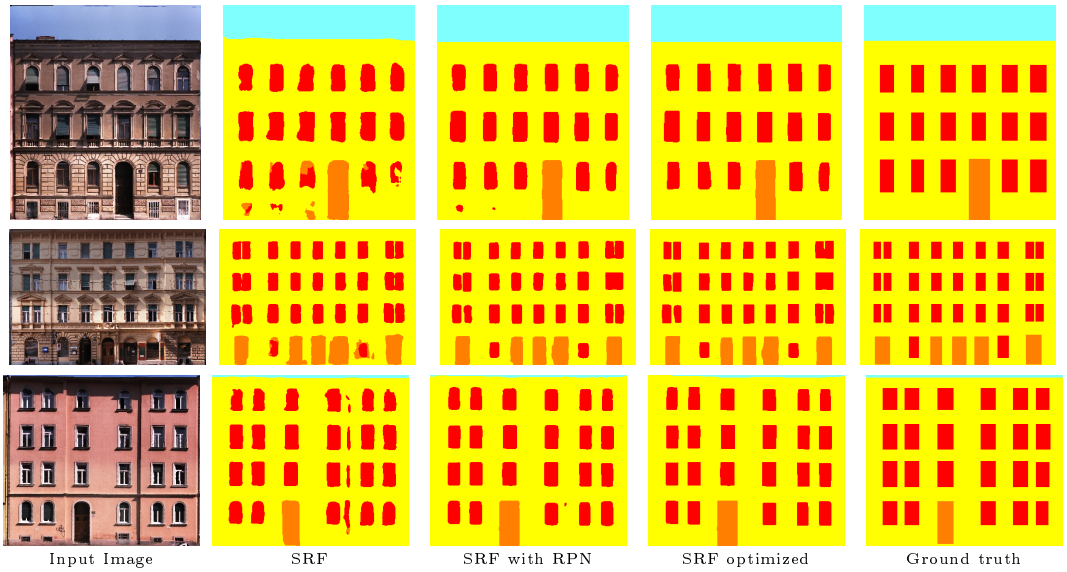


Figure 6.7: Results for the Graz dataset after optimization.

Rectangular Fitting

The final step of our pipeline is rectangular fitting. Since all objects in our dataset have a rectangular shape, this step improves the overall accuracy by more than 0.6% for the ECP Paris and 0.8% for the Graz dataset (Table 6.5). The accuracy of the sky remains the same, since we do not perform a straight line delineation between sky and roof, as roofs are not always straight. On the other hand, we delineate the boundary of wall with roof or sky with a straight line, since each image contains only one facade and roof and wall are always delineated by a straight line.

In the ECP Paris dataset, we see a drop in accuracy for the roof and balcony classes by 1.8% and 0.8%, respectively. The reason for this is the geometric (radial) distortion of several images which results in thinner roofs than in the ground truth annotation after rectangular fitting (Fig. 6.8).

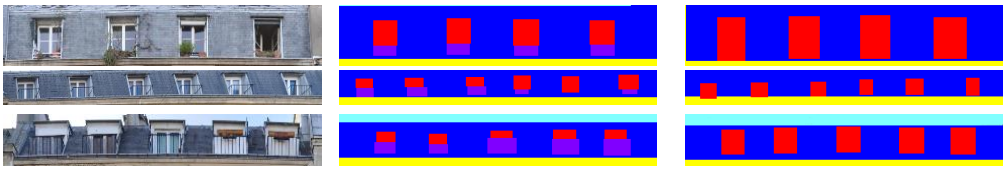


Figure 6.8: Problems with windows and balconies

Yet, there are significant improvements for the wall and window classes by 0.9% and 0.6% and small improvements for door and shop by 0.2% and 0.3%, respectively. One can see that rectangular fitting improved the accuracy of the classes which are frequent in the images, while the classes roof and balcony, where the delineation is actually not really straight, dropped.

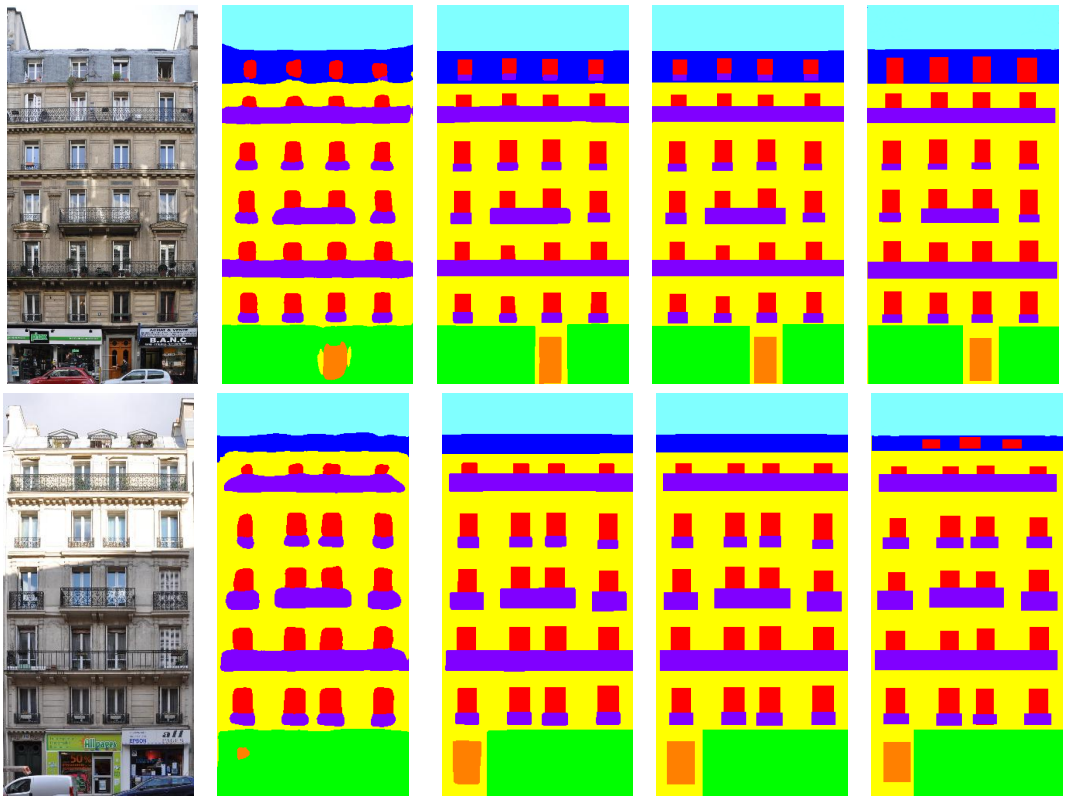


Figure 6.9: Results for the ECP Paris dataset after Rectangular Fitting

For the Graz dataset, the accuracy of doors drops by 0.3% mostly due to the arc-shaped structure of the doors for which rectangular fitting has problems to determine the upper boundary. The improvement of the accuracy of windows is 2.8% because of the rectification. The accuracy of sky is improved by more than 1.3% due to the straight delineation of sky and wall.

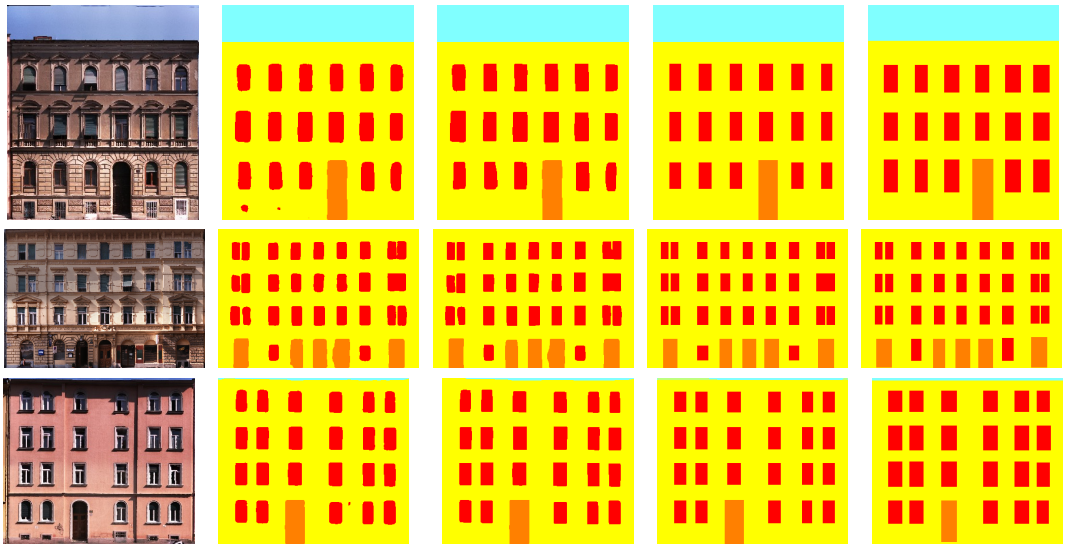


Figure 6.10: Results for the Graz dataset after Rectangular Fitting

Qualitatively, rectangular fitting improves the segmentation significantly (Figures 6.9 and 6.10), because each object is represented as a axis parallel rectangle. The output of rectangular fitting can,thus, be easily transformed to an output similar to grammar based methods. Additionally, with rectangular fitting we prepare our output for being used in 3D reconstructions pipelines.

Class	Base	RPN	OPT	RF
Door	56.6	89.1	89.0	89.2
Shop	95.3	95.5	96.0	96.3
Balcony	86.4	89	90.0	89.2
Window	70.4	77.3	78.0	78.6
Wall	91.4	92.9	92.6	93.5
Sky	95.3	97.1	97.2	97.2
Roof	93.2	94.9	94.8	93.0
Overall	88.9	91.5	91.6	92.2

ECP Paris results

Class	Base	RPN	OPT	RF
Sky	86	89.9	90.0	91.3
Window	79	80.6	80.9	83.7
Door	91.8	93.7	94.1	93.8
Wall	96	96.1	96.3	96.5
Overall	91.9	92.6	92.8	93.6

Results Graz

Table 6.5: Accuracy after Rectangular Fitting(RF)

Comparison with other methods

We have compared our method with several other methods (Cohen et al., 2017, Gadde et al., 2017, Mathias et al., 2016, Riemenschneider et al., 2012) based on pipeline architectures. Our method performs best and is, thus, the state of the art for bottom-up approaches using components.

Class	Results						
	I	II	III	Base	RPN	OPT	RF
Door	81.3	79	71	56.6	89.1	89.0	**89.2**
Shop	93.2	96	95	95.3	95.5	96.0	**96.3**
Balcony	89.3	**92**	87	86.4	89	90.0	89.2
Window	82.3	**87**	78	70.4	77.3	78.0	78.6
Wall	92.9	91	89	91.4	92.9	92.6	**93.5**
Sky	**98.2**	97	96	95.3	97.1	97.2	97.2
Roof	89.2	91	79	93.2	**94.9**	94.8	93.0
Average	89.5	-	85.2	84.1	90.8	**91.1**	91.0
Overall	91.4	91.8	88.0	88.9	91.5	91.6	**92.2**

Table 6.6: Results for dataset ECP Paris for different methods. I – Jampani et al. (2015), II – Cohen et al. (2017), III – Mathias et al. (2016). Base – Performance of Structured Random Forest (SRF) without Region Proposal Network (RPN), RPN – SRF with RPN, OPT – optimized SRF with RPN and RF – Rectangular Fitting. Best results given in bold.

Our method is not competitive with end-to-end deep learning approaches Liu et al. (2017) on large datasets, but for small dataset like Graz, there is no end-to-end network architecture publication to compare to.

On the ECP Paris dataset (Table 6.6), our pipeline outperforms the current methods by 0.4%. Yet, particularly for the door class, we outdo all other methods by more than 7.9%, Furthermore, we can see that our method performs best if only by a small margin for the classes shop and wall. From this we conclude that our pipeline works significantly better than all other published methods for the lower part of the images containing the classes shop, door and wall.

Our method also performs better than others by a large margin (more than 3%) for the class roof. For the class balcony we have weaker results than Cohen et al. (2017) by about 3%. Their results are better due to the symmetry constraints for facade objects.

Class	Results							
	I	II	III	IV	Base	RPN	OPT	RF
Sky	91	88	88	90.6	86	89.9	90.0	**91.3**
Window	60	76	85	80.9	79	80.6	80.9	**83.7**
Door	41	52	64	63	91.8	93.7	**94.1**	93.8
Wall	84	95	96	95.8	96	96.1	96.3	**96.5**
Overall	78.0	89.6	91.6	91.68	91.9	92.6	92.8	**93.6**

Table 6.7: Results for Graz dataset for different methods. I – Riemenschneider et al. (2012), II – Cohen et al. (2014), III – Cohen et al. (2017), IV – Jampani et al. (2015). Base – Performance of Structured Random Forest (SRF) without Region Proposal Network (RPN), RPN – SRF with RPN, OPT – optimized SRF with RPN and RF – Rectangular Fitting. Best results given in bold.

For the Graz dataset (Table 6.7), our baseline method performs better than all other methods Cohen et al. (2017), Jampani et al. (2015), Riemenschneider et al. (2012). Our complete pipeline has a better accuracy by 2.5% than all other methods and is, thus, the state of the art for the Graz dataset. Our method is better than all published methods for all classes, especially the door class where our method is more than 30% more accurate. Since the door class covers only 13% of overall pixels (Table 6.1), it does not significantly effect the overall accuracy.

Problems with Windows in the ECP Paris Dataset

The worst accuracies are obtained for the class window (Table 6.6) due to problems with windows on the roof. Our algorithm struggles to identify windows because of occlusions and balconies that cover almost the complete window due to the viewing angle (Fig. 6.8). The object detector proposes at the same location a window as well as a balcony.

Trust of SRF in RPN

Even though object detectors are very accurate, they still make mistakes. The problem of our pipeline is that if an object detector makes an error in the form that it proposes a non-existent object, an SRF cannot back-track from this problem (Fig. 6.11). Object proposal features have a very high information gain and, thus, the SRF trusts them.

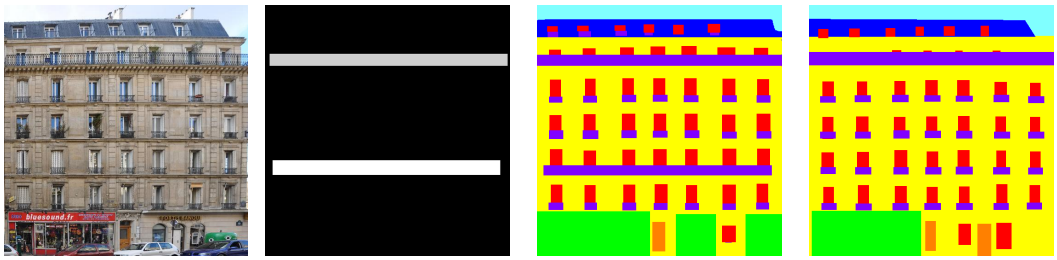


Figure 6.11: Effect of a false positive object proposal for long balcony on final pipeline results.

6.2 Mullions, Transoms and Frame Segmentation

In this section we present the evaluation of our transom and mullion segmentation pipeline based on the output of the two components of the pipeline: The Pyramid Scene Parsing Network (PSPNet) and the final output of the pipeline.

We have chosen a different measure to determine the accuracy of the segmentation of transoms, mullions and window frame. We use the Jaccard index which is also known as Intersection over Union (IoU) (Equation 6.3). It is particularly suitable for segmentation tasks where there are void classes or other classes that are not relevant for evaluation. For each class a score is computed as ratio between the intersection of the ground truth with the prediction and the union of both.

$$IoU(Prediction, Ground\_truth) = \frac{|Prediction \cap Ground\_truth|}{|Prediction \cup Ground\_truth|} \tag{6.3}$$

In our case, we just want to evaluate the classes window and frame the latter including mullions and transoms, omitting other classes, particularly wall.

We evaluate the output of PSPNet as well as after using the joint information of the grid and PSPNet. In our evaluation dataset 28% of the pixels are from the frame and the remaining are window pixels. All other pixels are not part of the evaluation and are considered void.

Pyramid Scene Parsing Network (PSPNet)

The trained PSPNet labels each pixel with one of the three classes window, frame and wall. The basic architecture of the network is not changed and we have used a pre-trained network for parameter initialization. The network is trained with window patches found by the object detector and we have used images of different sizes.

Quantitatively, the IoU score for the class window is 81.27%. It is higher than the IoU for the class frame (Table 6.8). The overall score is 78.01%.

Class	IoU
Frame	70.08
Window	81.27
Overall	78.01

Table 6.8: The IoU score for segmentation by the PSPNet for classes window and frame

Qualitatively, we can observe (Fig. 6.13) that for windows without occlusions, with frontal view and with high (color) contrast between frame and window, the segmentation network performs very well: Under these conditions, in most cases PSPNet detects the complete window and segments thin structures such as transoms and mullions very well.

On the other hand, at the margin of the images or when a part of a window is occluded by vegetation, street lights or similar objects, the PSPNet results are not as desired. In most cases the system classifies the occluding objects as wall, since their color does not correspond to the frame or the window.

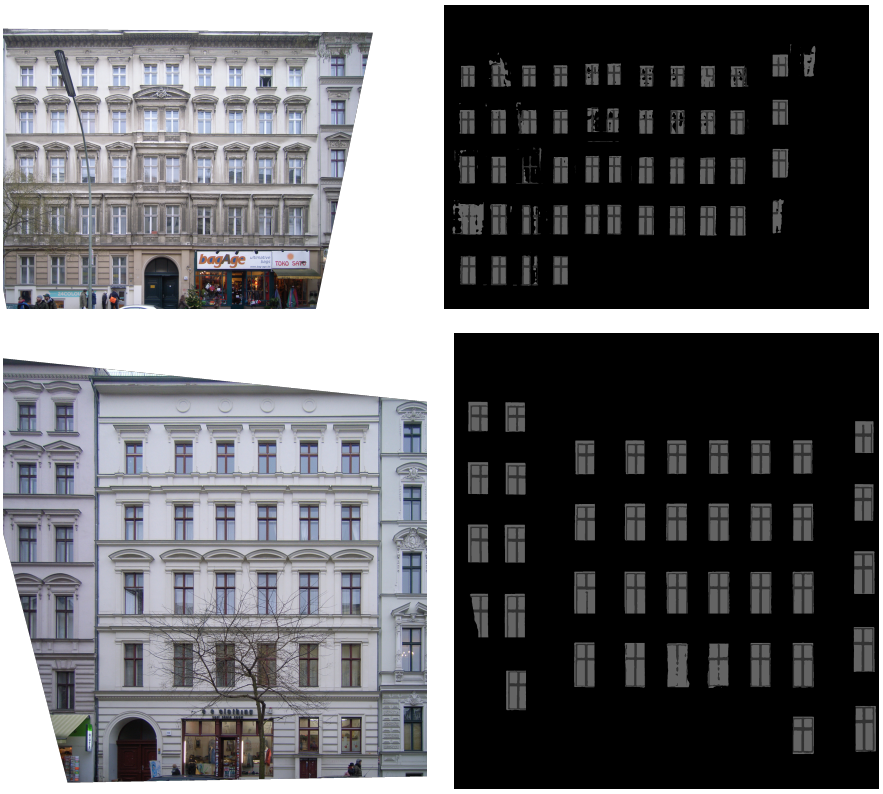


Figure 6.12: Results of PSPNET. Left: Input image. Right: Output of PSPNet (light gray: window, dark gray: frame, black: wall).

Improvement by Using the Global Structure

The second step of our pipeline uses the global grid structure of the facade for an improved delineation of the windows and the frame (Section 5.2). At the same time, we create a structured output for the facade consisting of the windows and their coordinates. Each window is described with two pairs of coordinates of two diagonally opposite corners and two sets for the transoms and mullions. Also the latter two are represented as rectangles and by the coordinates of their corners. This step improves the overall IoU by 1.46% (Table 6.9).

Class	IoU	
	PSPNet	Grid
Frame	70.08	71.33
Window	81.27	82.75
Overall	78.01	79.47

Table 6.9: Improvements by using the grid structure of the facade.

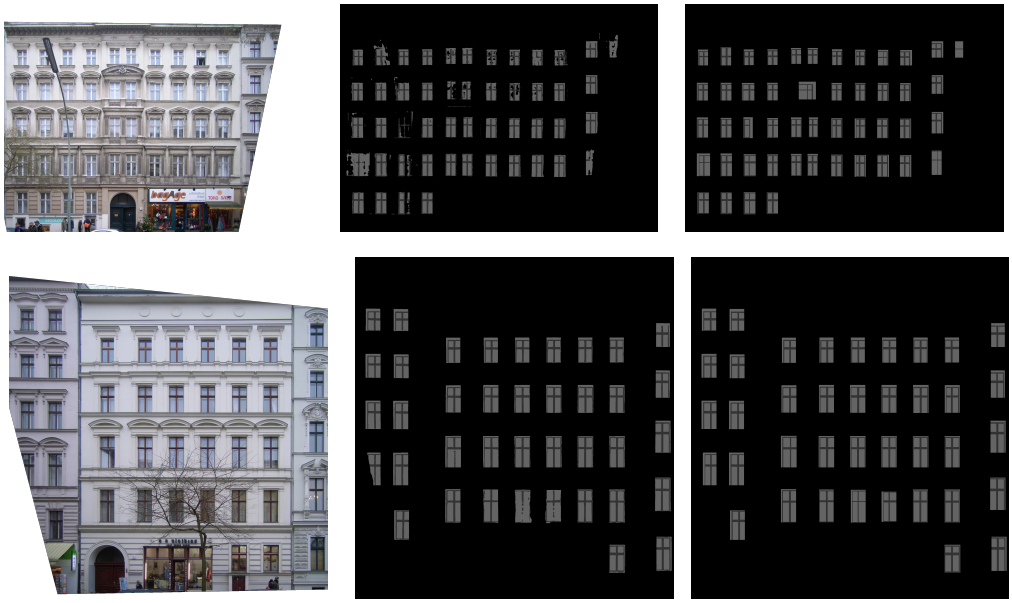


Figure 6.13: Results of transom, mullion and frame segmentation after using the joint information of the grid and PSPSNet. Left: input image, center: output of PSPNet, right: output using the global (grid) information and the PSPNet output.

6.3 Effect of Pseudo Depth Map on Facade Segmentation

This thesis is the outcome of a project in cooperation with a partner whose goal has been to estimate 3D primitives, particularly 3D planes, of the facade (Ley and Hellwich, 2016).

One of the hypothesis of this project has been, that there is a positive effect by using a depth map generated by 3D reconstruction on the segmentation accuracy. To check this hypothesis, we have evaluated the effect of the (pseudo) depth map generated from the 3D primitives on the accuracy of the segmentation results obtained by the Structured Random Forest (SRF). To this end, we trained several SRF with and without the depth map and found that the depth does not significantly effect the accuracy of the model. The main reason for this is the significantly higher information gain of the object detector features compared to the depth map features and, thus, mainly the former are used. Empirically, we found that parts of the depth-map have a poor quality compared to the quality of the object detector features, and therefore, can negatively affect the accuracy.

When we do not use object detector features, the classification accuracy is 87.13%. By adding the depth information the accuracy improves by more than 1.2% and the system achieves an accuracy of 88.4%. When the object detectors are added, the average accuracy using depth is 91.21% and without 91.18%. These quantitative results tell us that the depth map has not significantly improved the results. The main problem is that the pseudo depth map is not entirely correct. Particularly, there are missing windows, e.g., due to occlusions by vegetation (Figure 6.14).

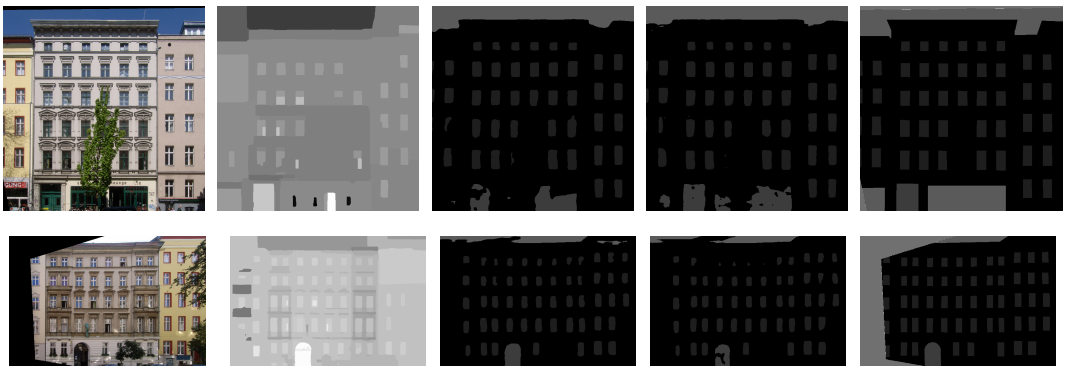


Figure 6.14: Result with and without depth map. a) Input images, b) Depth map, c) Results using depth map, d) Results without depth. e) Ground truth. There are only very small differences between results with and without depth map.

Chapter 7

Conclusion

This thesis is concerned with facade segmentation, particularlly, the accurate delineation of facade windows. We have used several different paradigms to produce high-quality results. Furthermore, we have gone one step further than former work by delineating substructures of windows in the form of window frame, transoms and mullions. We represent the facade elements as geometric shapes which can be used for 3D modeling.

7.1 Contribution

This thesis describes the following contributions:

- Our pipeline for facade segmentation based on Structured Random Forest (SRF), object detection and model fitting has empirically been proven to be a state-of-the-art approach for facade segmentation. The hybrid approach is based on traditional machine learning, deep learning and shape optimization. In our experiments we have shown that we can deal with occlusions, geometric distortion as well as windows mostly covered by balconies on the upper floors of the building.
- For the delineation of the windows we have created the first dataset for facade segmentation including labels for window frames as well as transoms and mullions. It contains more than 30 facades in more than 100 images from different German cities.
- Our pipeline is the first system for the delineation of window substructures using deep learning techniques for object detection and segmentation together with clustering and optimization approaches for post-

processing and model fitting. We have shown that the Pyramid Scene Parsing Network (PSPNet) is suitable for this problem.

- The evaluation of the effect of a (pseudo) depth map on facade segmentation has shown that it only leads to improvements of the results when the depth map is correct and when there are no features that have a higher information gain than it. Particularly, the object detector features shadow the effect of the depth map, since they contain more accurate information.

7.2 Summary

In this thesis, we have presented our work on facade segmentation starting with the introduction of the problem and its challenges. For the latter, most are similar for all types of facades, but some are only valid for specific types of architecture or datasets. The objective to build a highly accurate as well as descriptive facade segmentation algorithm has been achieved. It supports the detailed 3D reconstruction of cities, which has been the main motivation of this thesis.

The second chapter presents the **Theoretical Background**. It starts with feature extraction algorithms, some of them specifically developed for facade segmentation. This is followed by an introduction to the theory behind Structured Machine Learning algorithms and, particularly, the Structured Random Forest.

Deep Learning and Artificial Neural Networks are important components of our pipeline and, thus, we introduce architectures that we have used, also introducing their strengths. We give an overview on object detection and segmentation algorithms since they are employed for facade segmentation. Finally, the Pyramid Scene Parsing Network (PSPNet) architecture, which we employ to delineate windows and segment window frame, mullions, transom and glass, encodes some of the important principles of traditional machine learning object detection algorithms.

The third chapter on **Related Work** gives an overview on window detectors and facade segmentation algorithms. Several window detection algorithms are described which are all based on model fitting and traditional machine learning algorithms and employ 2D as well as 3D data.

We have divided the facade segmentation approaches into two types: top-down grammar-based approaches and bottom-up approaches. The latter are based on pixel or super-pixel classification and outperform the former in terms of accuracy and speed. For the bottom-up approaches, we distinguish those using deep learning (end-to-end training) and traditional machine learning.

In the fourth chapter on **Facade Segmentation Based on Structured Random Forests** our novel pipeline for facade segmentation is presented. It combines deep learning and traditional machine learning in a hybrid approach. By means of object delineation and model fitting it creates descriptive results of the facade in a way similar to top-down approaches.

Several highly specific domain dependent features have been conceived and an object detector has been built using a deep neural network. For the latter, we make the detection task easier by adapting the object classes.

We have introduced the power of the Structured Random Forest (SRF) for facade segmentation. It can encode local structures and it produces an almost noise free segmentation. Its optimization reduces the noise even more and creates local shapes very similar to the shapes seen in the training set.

The last step of the pipeline is rectangular fitting. It corrects and improves the segmentation and at the same time represents the facade objects' rectangular geometric shapes similar to the top-down approaches for facade segmentation.

In the fifth chapter **Window Refinement**, first the dataset with window frame, transoms, mullions as well as depth information generated from images of German cities is presented. The main part is dedicated to our novel method for segmenting the window frame as well as transoms and mullions using a Deep Neural Network based on the Pyramid Scene Parsing Network (PSPNet). For post-processing, clustering based on row and column position is used to determine a representative window position. Finally, each window is delineated individually employing a deterministic algorithm using information derived by clustering as well as the output of the semantic segmentation.

The sixth chapter presents **Experiments and Evaluation**. We have used several datasets with different architectural styles. Our facade segmentation system is compared with the currently published systems showing that for some of the datasets our algorithm obtains state-of-the-art results. No other methods exist for the segmentation of the window frame as well as transoms and mullions, for which we obtain high quality results.

While the goal that has been set for this thesis has been achieved, there is still, room for improvement as discussed in the next section.

7.3 Future Work

Using window frames, transoms and mullions, one could generate a 3D reconstruction from a single image assuming that windows have the same shape, dimensions and depth on the facade.

A feature extractor that encodes aspects of the global structure of the facade would be useful as additional input for the Structured Random Forest. Methods using Auto-Context can generate this information, but the runtime grows linearly with the number of Auto-context iterations. Another methodology to acquire the global structure of the facade could be to extract features from the detector's output, e.g., counting the number of windows or forcing all similar windows to strictly align in a grid.

For the Structured Random Forest, specific or generic dimension reduction techniques could be used. This would help to reduce the output space and at the same time lead to better splits during traversal. Another open challenge is how to integrate the knowledge of more than three pixels in a split function and at the same time limit the complexity. A practical problem is memory usage during inference and training for the SRF. Reducing it, would probably also lead to runtime improvement.

Improving traditional machine learning methods requires either new features or adding other components to the pipeline, which would both increase runtime and memory usage. Thus, the desired solution for any semantic segmentation system is an end-to-end convolutional neural network. Nowadays, pure deep learning approaches have been demonstrated to be very successful and, thus, arguable the ideal pipeline for this topic. The feature extractors can be taken from a pre-trained Deep Neural Network. Also a new architecture could be built, but this would require a lot of training data. Finally, also a hybrid solution as in this thesis could be a topic for the future of facade segmentation if the neural network solution should still show deficits.

In the future, the dataset with window frame, transoms and mullions should be extended with more buildings from different architectural styles.

We plan to publish our software in the near future with open access so that other researchers can use it.

Bibliography

Ali, H., Seifert, C., Jindal, N., Paletta, L. and Paar, G. (2007): Window detection in facades, *International Conference on Image Analysis and Processing*, 837–842.

Andreopoulos, A. and Tsotsos, J. K. (2013): 50 years of object recognition: Directions forward, *Computer Vision and Image Understanding* **117**(8): 827–891.

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

Baddeley, A. and van Lieshout, M.-C. (1993): Stochastic geometry models in high-level vision, *Journal of Applied Statistics* **20**(5-6): 231–256.

Badrinarayanan, V., Kendall, A. and Cipolla, R. (2017): Segnet: A deep convolutional encoder-decoder architecture for image segmentation, *IEEE transactions on pattern analysis and machine intelligence* **39**(12): 2481–2495.

Bakir, G., Hofmann, T., Schölkopf, B., Smola, A. J., Taskar, B. and Vishwanathan, S. (2007): *Predicting Structured Data*, MIT press.

Ballard, D. H. (1981): Generalizing the Hough transform to detect arbitrary shapes, *Pattern Recognition* **13**(2): 111–122.

Benenson, R., Mathias, M., Timofte, R. and Van Gool, L. (2012): Pedestrian detection at 100 frames per second, *Computer Vision and Pattern Recognition*, 2903–2910.

Biederman, I. (1987): Recognition-by-components: A theory of human image understanding, *Psychological Review* **94**(2): 115.

Bourdev, L. and Brandt, J. (2005): Robust object detection via soft cascade, *Computer Vision and Pattern Recognition*, Volume 2, 236–243.

Cohen, A., Oswald, M. R., Liu, Y. and Pollefeys, M. (2017): Symmetry-aware façade parsing with occlusions, *International Conference on 3D Vision (3DV)*, 393–401.

Cohen, A., Schwing, A. G. and Pollefeys, M. (2014): Efficient structured parsing of facades using dynamic programming, *Computer Vision and Pattern Recognition*, 3206–3213.

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

Dalal, N. and Triggs, B. (2005): Histograms of oriented gradients for human detection, *Computer Vision and Pattern Recognition*, Volume 1, 886–893.

Davarpanah, S. H., Khalid, F., Abdullah, L. N. and Golchin, M. (2016): A texture descriptor: Background local binary pattern (bglbp), *Multimedia Tools and Applications* **75**(11): 6549–6568.

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K. and Fei-Fei, L. (2009): Imagenet: A large-scale hierarchical image database, *Computer Vision and Pattern Recognition*, 248–255.

Dick, A. R., Torr, P. H. and Cipolla, R. (2002): A bayesian estimation of building shape using MCMC, *European Conference on Computer Vision*, 852–866.

Dick, A. R., Torr, P. H. and Cipolla, R. (2004): Modelling and interpretation of architecture from several images, *International Journal of Computer Vision* **60**(2): 111–134.

Dollár, P. (2017): Piotr's computer vision matlab toolbox (pmt), `https://github.com/pdollar/toolbox`.

Dollár, P. and Zitnick, C. L. (2013): Structured forests for fast edge detection, *International Conference on Computer Vision*, 1841–1848.

Dollar, P., Tu, Z., Perona, P. and Belongie, S. (2009): Integral channel features, *British Machine Vision Conference*, 1–11.

Felzenszwalb, P. F., Girshick, R. B., McAllester, D. and Ramanan, D. (2010): Object detection with discriminatively trained part-based models, *IEEE Transactions on Pattern Analysis and Machine Intelligence* **32**(9): 1627–1645.

Felzenszwalb, P., McAllester, D. and Ramanan, D. (2008): A discriminatively trained, multiscale, deformable part model, *Computer Vision and Pattern Recognition*, 1–8.

Fergus, R., Perona, P. and Zisserman, A. (2003): Object class recognition by unsupervised scale-invariant learning, *Computer Vision and Pattern Recognition*, Volume 2.

Förstner, W. and Gülch, E. (1987): A fast operator for detection and precise location of distinct points, corners and centres of circular features, *ISPRS Intercommission Conference on Fast Processing of Photogrammetric Data*, Interlaken, 281–305.

Fröhlich, B., Rodner, E. and Denzler, J. (2010): A fast approach for pixelwise labeling of facade images, *International Conference on Pattern Recognition*, 3029–3032.

Gadde, R., Jampani, V., Marlet, R. and Gehler, P. (2017): Efficient 2D and 3D facade segmentation using auto-context, *IEEE transactions on pattern analysis and machine intelligence* **40**(5): 1273–1280.

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

Geurts, P., Ernst, D. and Wehenkel, L. (2006): Extremely randomized trees, *Machine learning* **63**(1): 3–42.

Girshick, R. B. (2012): *From rigid templates to grammars: Object detection with structured models*, The University of Chicago.

Girshick, R. (2015): Fast R–CNN, *International Conference on Computer Vision*, 1440–1448.

Girshick, R., Donahue, J., Darrell, T. and Malik, J. (2014): Rich feature hierarchies for accurate object detection and semantic segmentation, *Computer Vision and Pattern Recognition*, 580–587.

Graves, A., Mohamed, A.-r. and Hinton, G. (2013): Speech recognition with deep recurrent neural networks, *International Conference on Acoustics, Speech and Signal Processing*, 6645–6649.

Green, P. J. (1995): Reversible Jump Markov Chain Monte Carlo computation and Bayesian model determination, *Biometrika* **82**(4): 711–732.

He, K., Gkioxari, G., Dollár, P. and Girshick, R. (2017): Mask R-CNN, *International Conference on Computer Vision*, 2980–2988.

He, K., Zhang, X., Ren, S. and Sun, J. (2016): Deep residual learning for image recognition, *Computer Vision and Pattern Recognition*, 770–778.

He, X. (2008): *Learning Structured Prediction Models for Image Labeling*, University of Toronto.

He, X., Zemel, R. S. and Carreira-Perpiñán, M. Á. (2004): Multiscale conditional random fields for image labeling, *Computer Vision and Pattern Recognition*, Volume 2, II–II.

Heikkilä, M., Pietikäinen, M. and Schmid, C. (2006): Description of interest regions with center-symmetric local binary patterns, *Computer Vision, Graphics and Image Processing*, 58–69.

Ho, T. K. (1995): Random decision forests, *International Conference on Document Analysis and Recognition*, Volume 1, 278–282.

Jahangiri, M. and Petrou, M. (2009): An attention model for extracting components that merit identification, *International Conference on Image Processing*, 965–968.

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

Kontschieder, P., Bulo, S. R., Bischof, H. and Pelillo, M. (2011a): Structured class-labels in random forests for semantic image labelling, *International Conference on Computer Vision*, 2190–2197.

Kontschieder, P., Bulò, S. R., Donoser, M., Pelillo, M. and Bischof, H. (2011b): Semantic image labelling as a label puzzle game, *British Machine Vision Conference*, 1–12.

Korc, F. and Förstner, W. (2009): eTRIMS image database for interpreting images of man-made scenes, *Department of Photogrammetry, University of Bonn, Tech. Rep. TR-IGG-P-2009-01*.

Kozinski, M. (2015): *Segmentation of facade images with shape priors*, PhD thesis, Universite Paris-Est.

Kozinski, M., Gadde, R., Zagoruyko, S., Obozinski, G. and Marlet, R. (2015): A MRF shape prior for facade parsing with occlusions, *Computer Vision and Pattern Recognition*, 2820–2828.

Krizhevsky, A., Sutskever, I. and Hinton, G. E. (2012): Imagenet classification with deep convolutional neural networks, *Advances in Neural Information Processing Systems*, 1097–1105.

Kruskal, J. B. (1956): On the shortest spanning subtree of a graph and the traveling salesman problem, *Proceedings of the American Mathematical society* **7**(1): 48–50.

Lafferty, J., McCallum, A. and Pereira, F. C. (2001): Conditional random fields: Probabilistic models for segmenting and labeling sequence data, 282–289.

LeCun, Y., Bottou, L., Bengio, Y. and Haffner, P. (1998): Gradient-based learning applied to document recognition, *Proceedings of the IEEE* **86**(11): 2278–2324.

Lee, S. C. and Nevatia, R. (2004): Extraction and integration of window in a 3D building model from ground view images, *Computer Vision and Pattern Recognition*, Volume 2, II–II.

Leibe, B. (2004): *Interleaved Object Categorization and Segmentation*, PhD thesis, KU Leuven.

Leibe, B., Leonardis, A. and Schiele, B. (2004): Combined object categorization and segmentation with an implicit shape model, *Workshop on Statistical Learning in Computer Vision, European Conference on Computer Vision*, Volume 2, 7.

Ley, A. and Hellwich, O. (2016): Depth map based facade abstraction from noisy multi-view stereo point clouds, *German Conference on Pattern Recognition*, 155–165.

Liu, H., Zhang, J., Zhu, J. and Hoi, S. C. H. (2017): Deepfacade: A deep learning approach to facade parsing, *International Joint Conference on Artificial Intelligence*, 2301–2307.

Liu, T., Huang, X. and Ma, J. (2016): Conditional random fields for image labeling, *Mathematical Problems in Engineering*.

Long, J., Shelhamer, E. and Darrell, T. (2015): Fully convolutional networks for semantic segmentation, *Computer Vision and Pattern Recognition*, 3431–3440.

Lowe, D. G. (1999): Object recognition from local scale-invariant features, *International Conference on Computer Vision*, Volume 2, 1150–1157.

Lowe, D. G. (2004): Distinctive image features from scale-invariant keypoints, *International Journal of Computer Vision* **60**(2): 91–110.

Martinović, A. (2015): *Inverse procedural modeling of buildings*, PhD thesis, KU Leuven.

Martinovic, A. and Van Gool, L. (2013): Bayesian grammar learning for inverse procedural modeling, *Computer Vision and Pattern Recognition*, 201–208.

Martinović, A., Mathias, M., Weissenberg, J. and Van Gool, L. (2012): A three-layered approach to facade parsing, *European Conference on Computer Vision*, 416–429.

Mathias, M., Martinović, A. and Van Gool, L. (2016): ATLAS: A three-layered approach to facade parsing, *International Journal of Computer Vision* **118**(1): 22–48.

Mayer, H. and Reznik, S. (2006): MCMC linked with implicit shape models and plane sweeping for 3d building facade interpretation in image sequences, *Photogrammetric Computer Vision* **6**: 130–135.

McCulloch, W. S. and Pitts, W. (1943): A logical calculus of the ideas immanent in nervous activity, *The Bulletin of Mathematical Biophysics* **5**(4): 115–133.

Mortensen, E. N. and Barrett, W. A. (1995): Intelligent scissors for image composition, *Conference on Computer Graphics and Interactive Techniques*, 191–198.

Nair, V. and Hinton, G. E. (2010): Rectified linear units improve restricted Boltzmann machines, *International Conference on Machine Learning*, 807–814.

Neuhausen, M., Obel, M., Martin, A., Mark, P. and König, M. (2018): Window detection in facade images for risk assessment in tunneling, *Visualization in Engineering* **6**(1): 1.

Nguatem, W., Drauschke, M. and Mayer, H. (2014): Localization of windows and doors in 3D point clouds of facades, *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences* **2**(3): 87.

Nielsen, M. A. (2015): *Neural networks and deep learning*, Volume 25, Determination Press USA.

Ojala, T., Pietikäinen, M. and Mäenpää, T. (2001): A generalized local binary pattern operator for multiresolution gray scale and rotation invariant texture classification, *International Conference on Advances in Pattern Recognition*, 399–408.

Ojala, T., Pietikainen, M. and Maenpaa, T. (2002): Multiresolution gray-scale and rotation invariant texture classification with local binary patterns, *IEEE Transactions on Pattern Analysis and Machine Intelligence* **24**(7): 971–987.

Pearl, J. (1982): *Reverend Bayes on inference engines: A distributed hierarchical approach*, Cognitive Systems Laboratory, School of Engineering and Applied Science, University of California, Los Angeles.

Peng, B., Zhang, L. and Zhang, D. (2013): A survey of graph theoretical approaches to image segmentation, *Pattern Recognition* **46**(3): 1020–1038.

Prim, R. C. (1957): Shortest connection networks and some generalizations, *The Bell System Technical Journal* **36**(6): 1389–1401.

Rahmani, K., Huang, H. and Mayer, H. (2017): Facade segmentation with a structured random forest, *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences* **4**: 175–181.

Recky, M. and Leberl, F. (2010): Windows detection using k-means in CIE-Lab color space, *International Conference on Pattern Recognition*, 356–359.

Ren, S., He, K., Girshick, R. and Sun, J. (2015): Faster R-CNN: Towards real-time object detection with region proposal networks, *Advances in Neural Information Processing Systems*, 91–99.

Reznik, S. and Mayer, H. (2008): Implicit shape models, self-diagnosis, and model selection for 3D façade interpretation, *Photogrammetrie Fernerkundung Geoinformation* (3): 187–196.

Riemenschneider, H., Bódis-Szomorú, A., Weissenberg, J. and Van Gool, L. (2014): Learning where to classify in multi-view semantic segmentation, *European Conference on Computer Vision*, 516–532.

Riemenschneider, H., Krispel, U., Thaller, W., Donoser, M., Havemann, S., Fellner, D. and Bischof, H. (2012): Irregular lattices for complex shape grammar facade parsing, *Computer Vision and Pattern Recognition*, 1640–1647.

Ronneberger, O., Fischer, P. and Brox, T. (2015): U-net: Convolutional networks for biomedical image segmentation, *International Conference on Medical image computing and computer-assisted intervention*, 234–241.

Rosenblatt, F. (1958): The perceptron: A probabilistic model for information storage and organization in the brain., *Psychological Review* **65**(6): 386.

Schindler, K. and Bauer, J. (2003): A model-based method for building reconstruction, *International Workshop on Higher-Level Knowledge in 3D Modeling and Motion Analysis*, 74–82.

Schmitz, M. and Mayer, H. (2016): A convolutional network for semantic facade segmentation and interpretation, *International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences* **XLI**(133): 709–715.

Shotton, J., Blake, A. and Cipolla, R. (2005): Contour-vased learning for object detection, *International Conference on Computer Vision*, Volume 1, 503–510.

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

Silva, C., Bouwmans, T. and Frelicot, C. (2015): An extended center symmetric local binary pattern for background modeling and subtraction in videos, *International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications*, 395–402.

Simonyan, K. and Zisserman, A. (2014): Very deep convolutional networks for large-scale image recognition, *IEEE Transactions on Pattern Analysis and Machine Intelligence* **39**(12): 2481–2495.

Socher, R., Lin, C. C., Manning, C. and Ng, A. Y. (2011): Parsing natural scenes and natural language with recursive neural networks, *International Conference on Machine Learning*, 129–136.

Stolcke, A. (1994): *Bayesian learning of probabilistic language models*, PhD thesis, University of California, Berkeley.

Teboul, O. (2011): *Shape Grammar Parsing: Application to Image-Based Modeling*, PhD thesis, Ecole Centrale Paris.

Teboul, O., Kokkinos, I., Simon, L., Koutsourakis, P. and Paragios, N. (2011): Shape grammar parsing via reinforcement learning, 2273–2280.

Tu, Z. (2008): Auto-context and its application to high-level vision tasks, *Computer Vision and Pattern Recognition*, 1–8.

Turk, M. and Pentland, A. (1991): Eigenfaces for recognition, *Journal of Cognitive Neuroscience* **3**(1): 71–86.

Tyleček, R. and Šára, R. (2010): A weak structure model for regular pattern recognition applied to facade images, *Asian Conference on Computer Vision*, 450–463.

Tyleček, R. and Šára, R. (2013): Spatial pattern templates for recognition of objects with regular structure, *German Conference on Pattern Recognition*, 364–374.

Uijlings, J. R., Van De Sande, K. E., Gevers, T. and Smeulders, A. W. (2013): Selective search for object recognition, *International Journal of Computer Vision* **104**(2): 154–171.

Van den Bergh, M., Boix, X., Roig, G., de Capitani, B. and Van Gool, L. (2012): Seeds: Superpixels extracted via energy-driven sampling, *European Conference on Computer Vision*, 13–26.

VIOLA, P. and JONES, M. J. (2004): Robust real-time face detection, *International Journal of Computer Vision* **57**(2): 137–154.

VIOLA, P. and JONES, M. (2001): Rapid object detection using a boosted cascade of simple features, *Computer Vision and Pattern Recognition*, Volume 1, 1–9.

WANG, J., LIU, C., SHEN, T. and QUAN, L. (2015): Structure-driven facade parsing with irregular patterns, *Asian Conference on Pattern Recognition*, 041–045.

WANG, X. and HANSON, A. R. (1997): Extracting surface textures and microstructures from multiple aerial images, *Computer Vision and Pattern Recognition*, 301–306.

WENZEL, S. and FÖRSTNER, W. (2016): Facade interpretation using a marked point process, *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences* **3**: 363.

WENZEL, S., DRAUSCHKE, M. and FÖRSTNER, W. (2008): Detection of repeated structures in facade images, *Pattern Recognition and Image Analysis* **18**(3): 406–411.

WERNER, T. and ZISSERMAN, A. (2002a): Model selection for automated architectural reconstruction from multiple views, *British Machine Vision Conference*, 53–62.

WERNER, T. and ZISSERMAN, A. (2002b): New techniques for automated architectural reconstruction from photographs, *European Conference on Computer Vision*, 541–555.

WOLPERT, D. H. (1992): Stacked generalization, *Neural Networks* **5**(2): 241–259.

WU, Z. and LEAHY, R. (1993): An optimal graph theoretic approach to data clustering: Theory and its application to image segmentation, *IEEE Transactions on Pattern Analysis and Machine Intelligence* **15**(11): 1101–1113.

YANG, M. Y. (2011): *Hierarchical and spatial structures for interpreting images of man-made scenes using graphical models*, University of Bonn.

Yang, M. Y. and Förstner, W. (2011): A hierarchical conditional random field model for labeling and classifying images of man-made scenes, *International Conference on Computer Vision Workshops*, 196–203.

Yu, F. and Koltun, V. (2016): Multi-scale context aggregation by dilated convolutions, *International Conference on Learning Representations*.

Zhao, H., Shi, J., Qi, X., Wang, X. and Jia, J. (2017): Pyramid scene parsing network, *Computer Vision and Pattern Recognition*, 2881–2890.