Institut für Geodäsie und Geoinformation
Professur für Photogrammetrie

High-Level Facade Image Interpretation using Marked Point Processes

Dissertation
zur
Erlangung des Grades
Doktor-Ingenieur
(Dr.-Ing.)
der
Landwirtschaftlichen Fakultät
der
Rheinischen Friedrich-Wilhelms-Universität
Bonn

vorgelegt von
Susanne Wenzel
aus Berlin

Bonn 2016

Referent: Prof. Dr.-Ing. Dr. h.c. mult. Wolfgang Förstner

Koreferent: Prof. Dr. Lutz Plümer

Tag der mündlichen Prüfung: 10. Juni 2016

Diese Dissertation ist auf dem Hochschulschriftenserver der ULB Bonn elektronisch publiziert (http://hss.ulb.uni-bonn.de/diss_online).

Für meine Familie
Jörg, Tobias und Pauline,
meine Eltern

Zusammenfassung

High-Level Interpretation von Fassaden-Bildern unter Benutzung von Markierten Punktprozessen

Das Thema dieser Arbeit ist die Interpretation von Fassadenbildern als wesentlicher Beitrag zur Erstellung hoch detaillierter, semantisch reichhaltiger dreidimensionaler Stadtmodelle. In rektifizierten Einzelaufnahmen von Fassaden detektieren wir relevante Objekte wie Fenster, Türen und Balkone, um daraus eine Bildinterpretation in Form von präzisen Positionen und Größen dieser Objekte abzuleiten.

Die digitale dreidimensionale Rekonstruktion urbaner Regionen ist ein aktives Forschungsfeld mit zahlreichen Anwendungen, beispielsweise der Herstellung digitaler Kartenwerke für Navigation, Stadtplanung, Notfallmanagement, Katastrophenschutz oder die Unterhaltungsindustrie. Detaillierte Gebäudemodelle, die nicht nur als geometrische Objekte repräsentiert und durch eine geeignete Textur visuell ansprechend dargestellt werden, erlauben semantische Anfragen, wie beispielsweise nach der Anzahl der Geschosse oder der Position der Balkone oder Eingänge. Die semantische Interpretation von Fassadenbildern ist ein wesentlicher Schritt für die Erzeugung solcher Modelle.

In der vorliegenden Arbeit lösen wir diese Aufgabe, indem wir aus Daten abgeleitete Evidenz für das Vorkommen einzelner Objekte mit Vorwissen kombinieren, das die Analyse der gesamten Bildinterpretation steuert. Wir präsentieren dafür ein dreistufiges Verfahren: Wir erzeugen Bildmerkmale, die für die Beschreibung der relevanten Objekte geeignet sind. Wir lernen, auf Basis abgeleiteter Merkmale, eine Repräsentation dieser Objekte. Schließlich realisieren wir die Bildinterpretation basierend auf der zuvor gelernten Repräsentation und dem Vorwissen über typische Konfigurationen von Fassadenobjekten, welches wir aus Trainingsdaten ableiten.

Wir leisten dazu die folgenden wissenschaftlichen Beiträge: Wir schlagen eine neuartige Methode zur Interpretation von Fassadenbildern vor, die einen sogenannten markierten Punktprozess verwendet. Dafür entwickeln wir ein Modell zur Beschreibung typischer Konfigurationen von Fassadenobjekten und entwickeln ein Bildinterpretationssystem, welches aus Daten abgeleitete Evidenz und a priori Wissen über typische Fassadenkonfigurationen kombiniert. Für die Erzeugung der Evidenz stellen wir Merkmale vor, die wir Shapelets nennen und die skaleninvariant und durch eine ausgesprochene Distinktivität im Bezug auf Fassadenobjekte gekennzeichnet sind. Als Basismerkmale für die Erzeugung der Shapelets dienen Linien-, Kreis- und Ellipsensegmente. Dafür stellen wir eine neuartige Methode zur Vereinfachung von Liniensegmenten vor, die eine Pixelkette durch eine Sequenz von geraden Linienstücken und elliptischen Bogensegmenten approximiert. Diese basiert unter anderem auf einer Adaption des Douglas-Peucker Algorithmus, die anstelle gerader Linienstücke, Bogensegmente als geometrische Basiselemente verwendet.

Wir evaluieren jeden dieser drei Teilschritte separat. Wir zeigen Ergebnisse der Liniensegmentierung anhand verschiedener Bilder und weisen dabei vergleichbare und teilweise verbesserte Ergebnisse im Vergleich zu bestehende Verfahren nach. Für die vorgeschlagenen Shapelets weisen wir in der Evaluation ihre diskriminativen Eigenschaften im Bezug auf Fassadenobjekte nach. Wir erzeugen auf einem anspruchsvollen Datensatz von skalenvariablen Fassadenobjekten, mit starker Variabilität der Erscheinung innerhalb der Klassen, vielversprechende Klassifikationsergebnisse, die die Verwendbarkeit der gelernten Shapelets für die weitere Interpretation belegen. Schließlich zeigen wir Ergebnisse der Interpretation der Fassadenstruktur anhand verschiedener Datensätze. Die qualitative Evaluation demonstriert die Fähigkeit des vorgeschlagenen Lösungsansatzes zur vollständigen und präzisen Detektion der genannten Fassadenobjekte.

Abstract

In this thesis, we address facade image interpretation as one essential ingredient for the generation of high-detailed, semantic meaningful, three-dimensional city-models. Given a single rectified facade image, we detect relevant facade objects such as windows, entrances, and balconies, which yield a description of the image in terms of accurate position and size of these objects.

Urban digital three-dimensional reconstruction and documentation is an active area of research with several potential applications, e.g., in the area of digital mapping for navigation, urban planing, emergency management, disaster control or the entertainment industry. A detailed building model which is not just a geometric object enriched with texture, allows for semantic requests as the number of floors or the location of balconies and entrances. Facade image interpretation is one essential step in order to yield such models.

In this thesis, we propose the interpretation of facade images by combining evidence for the occurrence of individual object classes which we derive from data, and prior knowledge which guides the image interpretation in its entirety. We present a three-step procedure which generates features that are suited to describe relevant objects, learns a representation that is suited for object detection, and that enables the image interpretation using the results of object detection while incorporating prior knowledge about typical configurations of facade objects, which we learn from training data.

According to these three sub-tasks, our major achievements are:

We propose a novel method for facade image interpretation based on a marked point process. Therefor, we develop a model for the description of typical configurations of facade objects and propose an image interpretation system which combines evidence derived from data and prior knowledge about typical configurations of facade objects. In order to generate evidence from data, we propose a feature type which we call shapelets. They are scale invariant and provide large distinctiveness for facade objects. Segments of lines, arcs, and ellipses serve as basic features for the generation of shapelets. Therefor, we propose a novel line simplification approach which approximates given pixel-chains by a sequence of lines, circular, and elliptical arcs. Among others, it is based on an adaption to Douglas-Peucker's algorithm, which is based on circles as basic geometric elements.

We evaluate each step separately. We show the effects of polyline segmentation and simplification on several images with comparable good or even better results, referring to a state-of-the-art algorithm, which proves their large distinctiveness for facade objects. Using shapelets we provide a reasonable classification performance on a challenging dataset, including intra-class variations, clutter, and scale changes. Finally, we show promising results for the facade interpretation system on several datasets and provide a qualitative evaluation which demonstrates the capability of complete and accurate detection of facade objects.

Danksagung

Das Projekt Promotion war weit mehr als diese Dissertation – der Start in die wissenschaftliche Arbeit, Themenfindung, Lehre, viele angeregte Diskussionen, gemeinsame Reisen und schließlich die Konzentration auf dieses Thema. Begleitet wurde ich dabei von vielen Menschen, ohne deren Unterstützung diese Arbeit nicht entstanden, das Projekt Promotion nicht beendet worden wäre.

In erster Linie bedanke ich mich bei meinem Betreuer Wolfgang Förstner, der mich für die Photogrammetrie begeistert und in sein Arbeitsgruppe aufgenommen hat. Seine Faszination für das wissenschaftliche Denken und eine präzise Argumentation haben mich inspiriert und meine Forschung geprägt. Ich danke Lutz Plümer für das Koreferat, seine Unterstützung und die konstruktive Diskussion über die vorliegende Arbeit.

Besonders Danken möchte ich Cyrill Stachniss, der mich als Nachfolger von Wolfgang Förstner in sein Team aufgenommen und mir viel Zeit und Freiraum für meine Forschung und damit die erfolgreiche Beendigung dieser Arbeit gegeben hat. In der Entstehung dieser Arbeit haben mich viele Kollegen begleitet und mit spannenden Diskussionen und manchmal auch verrückten Ideen inspiriert. Die Zusammenarbeit und vor allem das gemeinsame Denken mit Timo Dickscheid, Martin Drauschke, Filip Korč, Ribana Roscher, Jan Siegemund, Falko Schindler, Johannes Schneider und Michael Yang haben diese Arbeit auf den Weg und vor allem voran gebracht und meine Zeit am IPB zu einer ganz besonderen gemacht.

Für die Unterstützung, insbesondere in der letzten Phase der Promotion, danke ich Cyrill und Ribana, für das Korrekturlesen und die Hilfestellungen zur Beendigung der Arbeit. Falko hat mit seinem ausgeklügelten Latex-Design den Grundstock für das Layout dieser Arbeit geliefert. Ich danke Thomas Läbe für seine Unterstützung in allen Fragen zur Hard-und Software. Heidi Hollander und Birgit Klein waren mir immer eine große Hilfe in allen organisatorischen Fragen, bei der Vorbereitung von Konferenzen und der Buchung der entsprechenden Reisen.

Ich widme diese Arbeit meiner Familie, ohne deren beständige Unterstützung ich die Promotion nicht hätte abschließen können. Meine Eltern haben mir den Weg geebnet, mein Studium ermöglicht und mich immer darin bestärkt diesen Weg zu gehen.

Jörg hat mir immer den Rücken frei gehalten, alle eigenen Projekte immer und immer wieder zurück gestellt und mich beständig ermuntert das Projekt Promotion zu beenden. Meine Kinder haben den Weg zum Hut neugierig begleitet und geduldig akzeptiert, dass ich oft nicht viel Zeit für sie hatte. Das war Teamarbeit!

Notation

General

$\mathcal{A}$	name of a set
$\{\dots\}$	a set of elements, may be unsorted, no element is the same, elements may have arbitrary type
$(\dots)$	a list of elements, sorted, elements may have arbitrary type
$[\dots]$	vectors or matrices, the elements of matrices are stacked column wise and of fixed dimension, in case of vectors their elements are scalars
$\ln(\cdot), \mathrm{lb}(\cdot)$	natural logarithm and binary logarithm

Statistics

$\underline{\cdot}$	random variable
$\overline{\cdot}$	mean value
$\mathbb{P}(\cdot)$	Probability of an event
$A \perp\!\!\!\perp B$	independence of events A and B
$\mathbb{E}(\cdot)$	Expectation
$P_x(\cdot), P(\cdot)$	a (cumulative) distribution function,
$p_x(\cdot), p(\cdot)$	a density function
$\mathcal{N}(\cdot)$	Normal density function
$\mathcal{U}(a, b)$	Uniform distribution in interval $[a, b]$
$\mathcal{M}(\alpha)$	Multinomial distribution with parameter α
$\mathcal{B}(a, b)$	Beta distribution, with parameters $[a, b]$
$\mathcal{D}(\boldsymbol{\alpha})$	Dirichlet distribution with parameter vector α

Geometry

$\mathcal{X}, \mathcal{l}, \mathcal{C}$	names of geometric objects, regardless their mathematical representation.
$\mathcal{K}$	a circle
$\mathcal{S}$	a circle segment
$\mathcal{C}$	a conic
$\boldsymbol{x}, \boldsymbol{y}$	2D euclidean vectors
$\mathbf{x}, \mathbf{l}$	2D homogeneous vectors
$(\cdot)^s$	spherical normalised homogeneous vector
$(\cdot)^e$	euclidean normalised homogeneous vector
$\mathsf{N}(\cdot)$	operator for the spherical normalisation of a homogeneous entity
$\mathsf{N}^{\mathsf{e}}(\cdot)$	operator for the euclidean normalisation of a homogeneous entity
$\Sigma_{\boldsymbol{xx}}$	covariance matrix of $\boldsymbol{x}$
R, T	euclidean matrices
M, H	homogeneous matrices
$\mathcal{X} = \{x_i\}$	A set of points $\mathcal{x}_i, i \in 1 \dots I$
K	homogeneous matrix of a circle
C	homogeneous matrix of a conic
$\mathcal{C}^S$	a conic segment, in our case an ellipse segment, given by $(\mathsf{C}, \phi_1, \phi_2, \Sigma_{cc})$

Algebra

I_D	identity matrix of dimension D
$\lvert\cdot\rvert$	absolute value of a scalar, euclidean norm of a vector, determinant of a matrix
$\lvert\cdot\rvert_L$	L-norm
$\text{null}(\cdot)$	null space operator, yields a set of column vectors spanning the null space

Pairs of Adjacent Line Segments and Shapelets

$\boldsymbol{t}$	a PAS-descriptor
$D(\boldsymbol{t}^a, \boldsymbol{t}^b)$	distance of two PAS-descriptors
N_t	minimal number of PAS to form a cluster of the shapelets-codebook
$\boldsymbol{s}$	a shapelet
$\boldsymbol{S}$	codebook of shapelets
T_X, T_Y	number of tiles horizontally and vertically for building the spatial tiling in BoW-representation
$\boldsymbol{h}$	histogram of shapelets
I_Σ	Integral image
H_Σ	Integral histogram

Point Processes and Markov Chains

$\mathcal{S}$	bounded Borel set, usually $\mathcal{S} \subset \mathbb{R}^d$
$\mathcal{M}$	space of marks
$\underline{\mathcal{X}}$	a point process on $\mathcal{S}$ or marked point process on $\mathcal{S} \times \mathcal{M}$
$\mathcal{X}$	a configuration of points in $\mathcal{S}$, $\mathcal{X} = \{\boldsymbol{x}_1, \dots \boldsymbol{x}_{N(\mathcal{X})}\}$
$N(\mathcal{X})$	number of points in $\mathcal{X}$
Ω	space of possible configurations on $\mathcal{S}$
N	the set of all counting measures on $\mathcal{S}$
$\mathcal{N}$	σ-algebra on N, given by the numbers of points in configurations of Ω
$\lambda_d(\cdot)$	Lebesgue measure on $\mathbb{R}^d$
$\mu(\cdot)$	intensity measure of a reference Poisson process, a measure on $\mathcal{S}$
π_μ	distribution of the finite Poisson process with intensity measure μ
$\boldsymbol{x}$	an object in image space, represented by $[x, y, w, h, c]$, position, width, height, and class
$f(\underline{\mathcal{X}})$	density of a point process
$F(\underline{\mathcal{X}})$	distribution of a point process
$\sim$	symmetric relationship to denote neighbouring objects
$\text{Ne}(\boldsymbol{x})$	neighbourhood of $\boldsymbol{x}$ with respect to $\sim$
$U(\mathcal{X})$	Gibbs energy of a configuration $\mathcal{X}$
$U_{\text{data}}(\mathcal{X})$	data term of the Gibbs energy, unary energy
$U_{\text{geom}}(\mathcal{X})$	geometric prior, unary energy
$U_{\text{conf}}(\mathcal{X})$	configuration prior, pairwise energy
$U_{\text{num}}(\mathcal{X})$	prior on number of objects per class
$\lambda_1, \lambda_2, \lambda_3$	weights of individual energy terms
$\Phi(\cdot)$	reward function, used in $U_{\text{data}}(\mathcal{X})$
$\Psi(\cdot)$	penalty function used in the prior energy terms
$\Psi_\cap(\cdot, \cdot)$	penalty function for overlapping objects
$\Psi_d(\cdot, \cdot)$	penalty function for the distance of two objects

$d_{\cap}(\cdot,\cdot)$	intersection area of two objects
$d_{d}(\cdot,\cdot)$	minimal distance of two objects
$d_{\leftrightarrow}(\cdot,\cdot)$	alignment distance horizontal of two objects
$d_{\updownarrow}(\cdot,\cdot)$	alignment distance vertical of two objects
$d_{\Delta h}(\cdot,\cdot)$	size difference in height of two objects
$d_{\Delta w}(\cdot,\cdot)$	size difference in width of two objects
$t_d(c_i, c_j)$	minimal distance of two object classes c_i and c_j stored during training
$g_k(\cdot)$	function representing learned configuration statistics, $k \in \{x, y, w, h, \leftrightarrow, \updownarrow, \Delta h, \Delta w\}$
T	temperature parametre of simulated annealing
$q(\cdot \mid x)$	proposal distribution when at state x
$A(\cdot \mid \cdot)$	acceptance probability of the Metropolis-Hastings algorithm
$R(\cdot \mid \cdot)$	acceptance ratio of the Metropolis-Hastings-Green algorithm
$\mathcal{Q}(\cdot \mid x)$	transition kernel when at state x

Abbreviations

AIC (Akaike's) An Information Criterion

BIC (Schwarz's) Bayesian Information Criterion

BoW bag of words

cdf (cumulative) probability distribution function

CNN convolutional neural network

CRF conditional random field

DLCE detecting line, circle, and ellipse segments

ELSD elliptical line segment detector

FAG feature adjacency graph

HOG histogram of oriented gradients

i.i.d. independent and identically distributed

ISM implicit shape model

IVM import vector machine

kAS k-adjacent line segments

KLR kernel logistic regression

MCMC Markov chain Monte Carlo

MDL minimal description length

MPP marked point process

MRF Markov random field

OA overall accuracy

PAS pairs of adjacent line segments

pdf probability density function

pmf probability mass function

RBF radial basis function

RF random forest

rjMCMC reversible jump Markov chain Monte Carlo

SVM support vector machine

TP true positive

Contents

1 Introduction **19**
- 1.1 Motivation . . . 20
- 1.2 Goal and Achievements . . . 20
- 1.3 Related Work on Facade Image Interpretation . . . 22
- 1.4 General Concept and Organisation of the Thesis . . . 39
- 1.5 Summary . . . 44

2 Theoretical Background **45**
- 2.1 Geometric Entities in 2D . . . 47
- 2.2 Statistical Background . . . 58
- 2.3 Model Selection . . . 60
- 2.4 Bag of Words . . . 61
- 2.5 Marked Point Processes . . . 71
- 2.6 Optimisation Using Reversible Jump Markov Chain Monte Carlo Sampling and Simulated Annealing . . . 77

3 Detecting Line, Circle and Ellipse Segments in Images **91**
- 3.1 Introduction of the DLCE algorithm . . . 92
- 3.2 Circle Peucker . . . 93
- 3.3 Merging Line Primitives . . . 94
- 3.4 Related Work . . . 96
- 3.5 Experiments . . . 97
- 3.6 Summary . . . 103

4 Using Shapelets for Object Classification **107**
- 4.1 Introduction of Shapelets and their Use for Object Categorisation . . . 108
- 4.2 Pairs of Adjacent Line Segments . . . 109
- 4.3 Learning Shapelets . . . 113
- 4.4 Using Shapelets for Classification . . . 114
- 4.5 Related Work . . . 116
- 4.6 Experiments . . . 118
- 4.7 Summary . . . 127

5 Top-Down Facade Image Interpretation by Marked Point Processes **129**
- 5.1 Using Bottom-Up Evidence for Top-Down Facade Interpretation . . . 130
- 5.2 An Energy for Facade Image Interpretation . . . 133
- 5.3 Optimisation . . . 141
- 5.4 Related Work . . . 143
- 5.5 Experiments . . . 145
- 5.6 Summary . . . 156

6 Conclusion **157**
6.1 Summary . . . 157
6.2 Discussion . . . 158
6.3 Outlook . . . 159

A Matrix Calculus **161**

B Uncertain Line Primitives **163**
B.1 Points and Straight Lines . . . 163
B.2 Representation of Ellipses . . . 167
B.3 Representation of Circles . . . 169

C Optimisation **175**
C.1 Golden section search . . . 175
C.2 Gauss-Helmert model . . . 177
C.3 Derivation of Green's Ratio . . . 178

D Further Results **183**
D.1 More Results on DLCE . . . 183
D.2 More Results on shapelets . . . 191

CHAPTER 1

Introduction

The objective of this theses is the interpretation of images showing facades of urban buildings. Given a single rectified facade image, our system detects and distinguishes relevant facade objects as windows, entrances, and balconies and results in a description of the image in terms of accurate position and size of these objects.
In this chapter, we motivate this task, describe the goals of the thesis, and point out our achievements. We review the related work in the field of facade image interpretation. Afterwards we are prepared to give an overview of the systems architecture and the according structure of the thesis.

1.1 Motivation **20**
1.2 Goal and Achievements **20**
1.3 Related Work on Facade Image Interpretation **22**
1.3.1 Tasks Addressed in Facade Image Interpretation and Reconstruction 22
1.3.2 Methods Used for Facade Image Interpretation and Reconstruction 31
1.3.3 What we Learned, Tasks Solved, and Open Issues 38
1.4 General Concept and Organisation of the Thesis **39**
1.4.1 The Big Picture 40
1.4.2 Organisation of the Thesis 40
1.5 Summary **44**

1.1 Motivation

The digital three-dimensional reconstruction and documentation of cities has received a growing interest over the last years. Thereby, urban reconstruction is an active area of research with several potential applications. First of all, digital mapping for navigation with cars, smart phones, tablets or on PCs requires two and three-dimensional urban models, as given by Google Earth, Bing Maps, OpenStreetMap, or classical map providing companies as Falk or land surveying offices. Further, urban planning relies on urban reconstruction as basic information about the environment or for visualization. Security relevant services, such as emergency management, disaster control, or security training need accurate, detailed, and up-to-date urban models. Finally, the entertainment industry is probably the largest market. Movies and computer games take place in real cities, where at least parts of the models are obtained by urban reconstruction (Musialski et al., 2012). From an economic point of view, there is an enormous benefit of being able to generate high-quality digital models in short time and fully automatic (Musialski et al., 2012).

Thereby, the urban area consist of many objects: streets, buildings, cars or street furniture. In this thesis, we focus on buildings, especially on facades, which contain relevant information for detailed reconstruction itself. A detailed building model which is not just a geometric object enriched with texture, allows for semantic requests as the number of floors or the location of balconies and entrances. For example, in case of a flood, this allows the emergency management to estimate the potential height of water until the first floor will be flooded. Or, in case of fire, the fire service will be aware of potential places were people try to get out of the house. The taxi driver will be able to stop his car exactly in front of the entrance. The architect will get detailed information about the structure of facades in the immediate vicinity of his planing area.

Various input data are used for the task of building reconstruction: LIDAR from ground or airborne and single or multiview images taken from ground or airborne. In this thesis, we use terrestrial, single view images. They are easy to obtain, to store, to exchange and a huge amount of images are available for free in the cloud. Nevertheless, for learning a model which is able to solve the task, in most cases, we need annotated training data which is not available for free in general.

There are still many unsolved problems on the way to fully automatic algorithms. Object detection from top-down expect well-defined object models, which clearly depend on context (Alegre and Dellaert, 2004; Müller et al., 2007; Ripperda, 2008). On the other hand, estimating the objects from bottom-up is context free, but assumes the data to fit the underlying assumptions (Lee and Nevatia, 2004; Čech and Šára, 2008).

Further, there is a trade off between quality and scalability of approaches. The increase of automation often is related to loss of quality. High quality models often are generated pure manually, or at least with manual control. Obviously, human interaction does not scale with huge data (Musialski et al., 2012). The intermediate solution is given by medium user interaction, e.g., Müller et al. (2007).

Finally, we deal with constraints given by the images. Illumination, shadows, occlusions and noise due to unwanted objects, such as cars, trees or decorations are problems we take into account.

1.2 Goal and Achievements

The goal of this theses is the interpretation of facade images by combining evidence for the occurrence of individual object classes which we derive from data, and prior knowledge which guides the image interpretation in its entirety. The former we denote as bottom-up evidence, while the latter we denote by top-down image interpretation. Given single rectified facade images, we aim at the accurate detection and description of relevant facade objects such as windows, entrances, and balconies.

We believe that image interpretation of complex scenes needs an interaction between low-level bottom-up feature detection and high-level inference from top-down. A top-down approach uses results of a bottom-up detection step as evidence for high-level inference of scene interpretation. Therefor, we present a three-step procedure which

- generates features from bottom-up that are suited to describe relevant objects;
- learns a representation of relevant objects at mid-level, based on learned features, that is suited for object detection; and
- enables high-level image interpretation using the results of object detection and incorporating prior knowledge about typical configurations of facade objects, which we learn from training data.

Further, we propose to apply the theory of marked point processes to the task of facade image interpretation and verify their potential as flexible model for the modelling of configurations of facade objects.

Achievements. According to these sub-tasks, these are our major scientific contributions, starting top-down:

- We propose a novel method for facade image interpretation based on a marked point process for which we define a probability density function over configurations of facade objects.
- We propose a feature type which we call shapelets, for which we learn pairs of adjacent line segments, including curved line segments, which are representatives for objects we aim at.
- We propose a novel approach for line simplification which approximates given pixel-chains by a sequence of lines and elliptical arcs. For this, to the best of our knowledge, we are the first that introduce an adaption to Douglas-Peucker's algorithm, which is based on circles as basic geometric elements.

Thereby, the first item contains the main contribution of this work. The application of Marked point processes to the interpretation of facade images is a new field and the definition of a probability density over configurations of facade objects a particular challenge. But, as we aim at the combination of data-driven bottom-up evidence for the occurrence of target object classes with prior knowledge from top-down, we also develop an appropriate classification system from bottom-up. This results in further subordinate contributions.

The contents of this thesis have been partly published in the following articles and conference proceedings:

Wenzel and Förstner (2012) Learning a Compositional Representation for Facade Objects. In *ISPRS Annals of Photogrammetry, Remote Sensing and the Spatial Information Sciences; Proc. of 22nd Congress of the International Society for Photogrammetry and Remote Sensing (ISPRS)*, Volume I-3, pp. 197-202.

Wenzel and Förstner (2013) Finding Poly-Curves of Straight Line and Ellipse Segments in Images. *Photogrammetrie, Fernerkundung, Geoinformation (PFG)*, 4, pp. 297-308.

Wenzel and Förstner (2016) Facade Interpretation Using a Marked Point Process. In *ISPRS Annals of Photogrammetry, Remote Sensing and the Spatial Information Sciences; Proc. of 23nd Congress of the International Society for Photogrammetry and Remote Sensing (ISPRS).*

Out of scope in this thesis, but done prior to this, the following articles and conference proceedings have been published

Wenzel, Drauschke, and Förstner (2007) Detection and Description of Repeated Structures in Rectified Facade Images. *Photogrammetrie, Fernerkundung, Geoinformation (PFG)*, 7, pp. 481–490.

Wenzel, Drauschke, and Förstner (2008) Detection of repeated structures in facade images. *Pattern Recognition and Image Analysis*, 18, pp. 406–411.

Wenzel and Förstner (2008) Semi-supervised incremental learning of hierarchical appearance models. *21st Congress of the International Society for Photogrammetry and Remote Sensing (ISPRS)*, Part B3b-2, pp. 399–404.

Wenzel and Hotz (2010) The Role of Sequences for Incremental Learning. *ICAART 2010 - Proceedings of the International Conference on Agents and Artificial Intelligence*, 1, pp. 434–439.

1.3 Related Work on Facade Image Interpretation

Research in facade reconstruction started in the nineties and dealt with topics of three-dimensional reconstruction, object detection and semantic interpretation, which are addressed in the following review and which motivates the approach proposed in this thesis. Here, we give a broad view of facade image interpretation to identify trends, constraints and restrictions. This prepares us to argue about our proposed solution.

We divide the review into two parts, the first following tasks that were addressed, and afterwards following methods that were used to solve the different tasks and subtasks. In some cases, we visualize their results or even meaningful figures from their approaches to illustrate the tasks they solve, and their method, as well as their performance.

The subsequent chapters will contain their own reviews of related work dealing with the special methods, used in the corresponding part of the work to solve the according subtasks.

1.3.1 Tasks Addressed in Facade Image Interpretation and Reconstruction

This section gives an overview of different task that were addressed in the context of facade image interpretation. The final goal of urban reconstruction are highly detailed, semantically meaningful 3D reconstructed buildings. We start with early approaches in 3D building reconstruction which motivates the need of facade image interpretation. Further, many approaches, dealing with facade image interpretation, also aim at 3D reconstruction, from which we we also partially review those methods related to 3D facade reconstruction, focusing on the facade image interpretation part.

There is a long history in 3D city modelling, either from aerial images or airborne laser scans or even from scanning maps Gruen (1997). Later, terrestrial images, as well as terrestrial laser scans, were used, too. Early work focused on three-dimensional reconstruction of buildings from aerial digital images, possibly deriving a digital elevation model as intermediate result, by modelling the objects using parametrized volume primitives together with application-specific constraints (Braun et al., 1995; Weidner and Förstner, 1995). Gruen et al. (1995) give an overview of these works. There is no explicit facade modelling included in theses works. But, they became more detailed. For example, Werner and Zisserman (2002) aim at the fully automatic reconstruction of buildings from multiple uncalibrated images. They generate a coarse piecewise planar model of the principal scene planes and their delineations and model indentations and protrusions, such as windows and doors, but without recognising their semantics, cf. Figure 1.1. Thereby the facades are modelled implicitly and the visualisation is enhanced by adding texture to the geometric model.

1.3.1.1 Window Detection

One of the earliest approaches which directly aim at the detection and reconstruction of windows from images is shown in Lee and Nevatia (2004). They state that the detection of details of building facades is needed for high quality visualisation as well as semantic modelling, and identify windows as key elements for a detailed facade reconstruction, cf. Figure 1.2. In the following, window detection was addressed in many works, either as main focus or intermediate result. Ali et al. (2007) perform

window detection through pure classification, cf. Figure 1.3. Most approaches include object specific prior knowledge, either assuming a typical grid structure of windows within facades (Lee and Nevatia, 2004; Recky and Leberl, 2010; Wenzel et al., 2008) or a typical geometry and appearance of windows (Čech and Šára, 2008), cf. Figures 1.4 to 1.6. Different from that, Jahangiri and Petrou (2009) aim at detecting salient regions which they assume to be meaningful parts of the facade and which turn out to be windows in most cases, cf. Figure 1.7. Most flexible in terms of structure of the window grid and appearance of single windows, Tyleček and Šára (2010) aim at detecting windows, in single rectified facade images, by modelling shape, appearance, location, and neighbourhood relations of windows, cf. Figure 1.8.

Different from the approaches before, Mayer and Reznik (2005) and Reznik and Mayer (2008) aim at the detection of windows from image sequences. Deriving depth of found window hypotheses, they reconstruct these parts of facades which are dominated by windows, cf. Figure 1.9.

A task, opposite from window detection, is addressed by Burochin et al. (2014). They use opening detection as indicator for the identification of blind facades, thus, facades without any openings, such as windows or doors, which is important for urban planing.

1.3.1.2 Semantic Interpretation through Pixelwise Labelling

Aiming at more than one specific facade object, next group of approaches perform a pixelwise labelling or facade segmentation. Berg et al. (2007) address image parsing using single, low resolutional, uncalibrated images as input. They first identify coarse regions, such as sky, foliage, buildings, and street, and at a finer level of detail, they further identify the positions of windows, doors, rooflines, and the spatial extent of particular buildings, cf. Figure 1.10. Similarly, Fröhlich et al. (2010) model eight classes for pixelwise classification, whereby, they do not take any structural meta knowledge about facades into account, cf. Figure 1.11.

Using high-level interpretation methods, cf. Section 1.3.2, Teboul et al. (2010); Martinović et al. (2012); Cohen et al. (2014) segment the images into 9 classes, which are window, wall, balcony, door, roof, chimney, sky, and shop, cf. Figure 1.12. While Teboul et al. (2010) and Cohen et al. (2014) rely on strong architectural constraints, Martinović et al. (2012) and Koziński et al. (2015) avoid strong prior assumption about the structure of facades and introduce weak architectural knowledge, which enforces the final reconstruction to be architecturally plausible and consistent, cf. Figure 1.13. Koziński et al. (2015) even model occlusions as extra class and are able to reconstruct the facades structure even if it is partly occluded, cf. Figure 1.14.

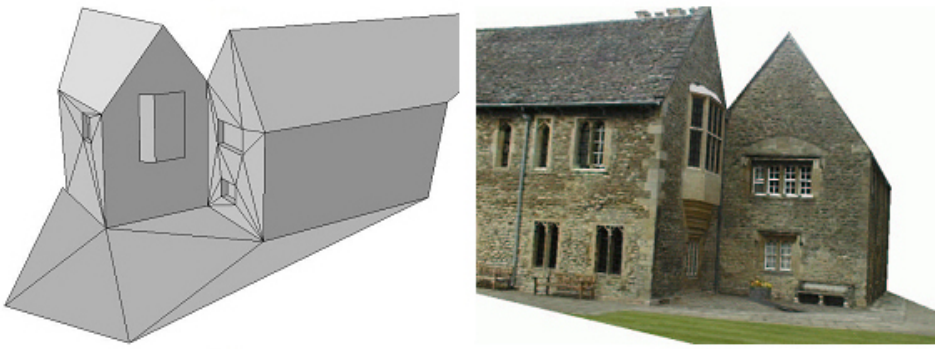


Figure 1.1: The 3D building reconstruction, as shown in Werner and Zisserman (2002), is given as piecewise planar model including indentations and protrusions, which we may identify as windows. The geometric representation is attached with texture, given by the images, but no semantic information is extracted.

Figure 1.2: Window detection and reconstruction from single view rectified images, as shown in Lee and Nevatia (2004). **Left:** Gradient projection. **Right:** Results after grouping similar elements and adding depth by plane sweep.

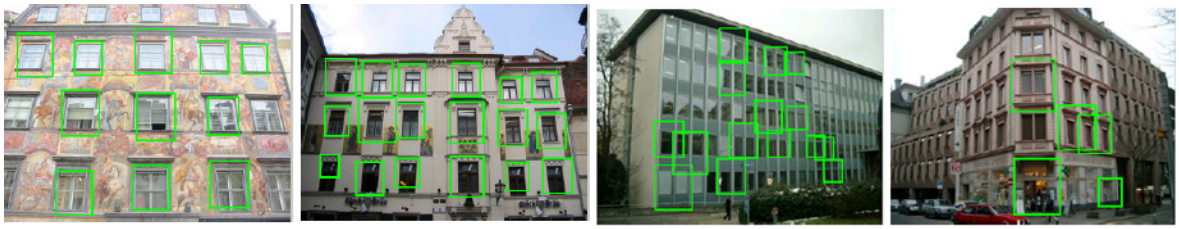


Figure 1.3: Results of window detection using an AdaBoost classifier, as shown in Ali et al. (2007).

Figure 1.4: Results of window detection based on gradient projection, as shown in Recky and Leberl (2010).

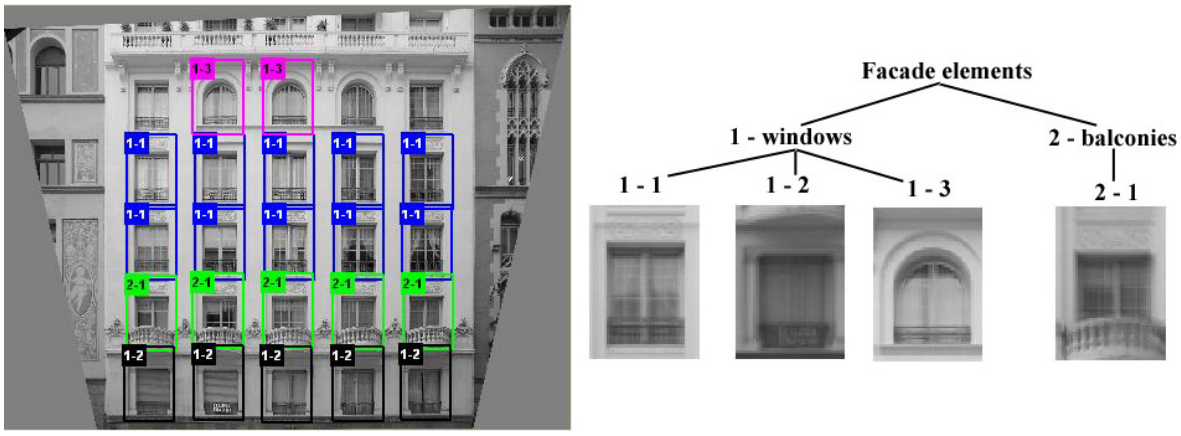


Figure 1.5: Results of semi-supervised window detection, as shown in Wenzel and Förstner (2008). **Left:** detected image patches. **Right:** derived class hierarchy.

Figure 1.6: Results of window pane detection, as shown in Čech and Šára (2008). **Left:** detected window panes. **Right:** labelled image patch and allowed configuration of labels (E - external, I - internal windowpane label, (L,R,T,B) - (left, right, top, bottom edge), (TL, TR, BL, BR) - corners).

Figure 1.7: Results of blob detection, as shown in Jahangiri and Petrou (2009).

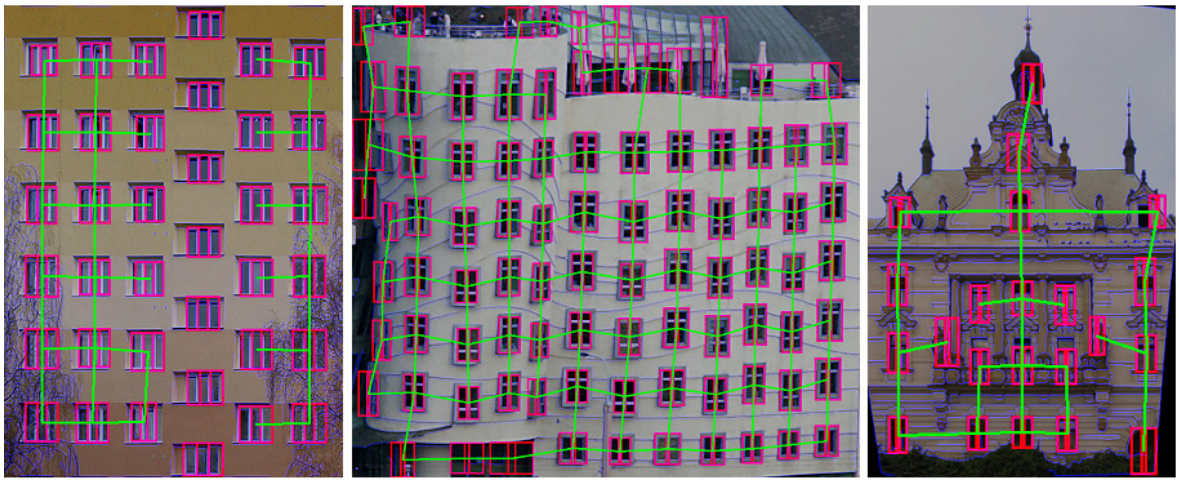


Figure 1.8: Results of window detection, using a hierarchical generative model, as shown in Tyleček and Šára (2010). They show results for modern regular, irregular, and sparse facades.

Figure 1.9: Result of window detection and 3D reconstruction, as shown in Reznik and Mayer (2008). Found window hypotheses are marked as red boxes.

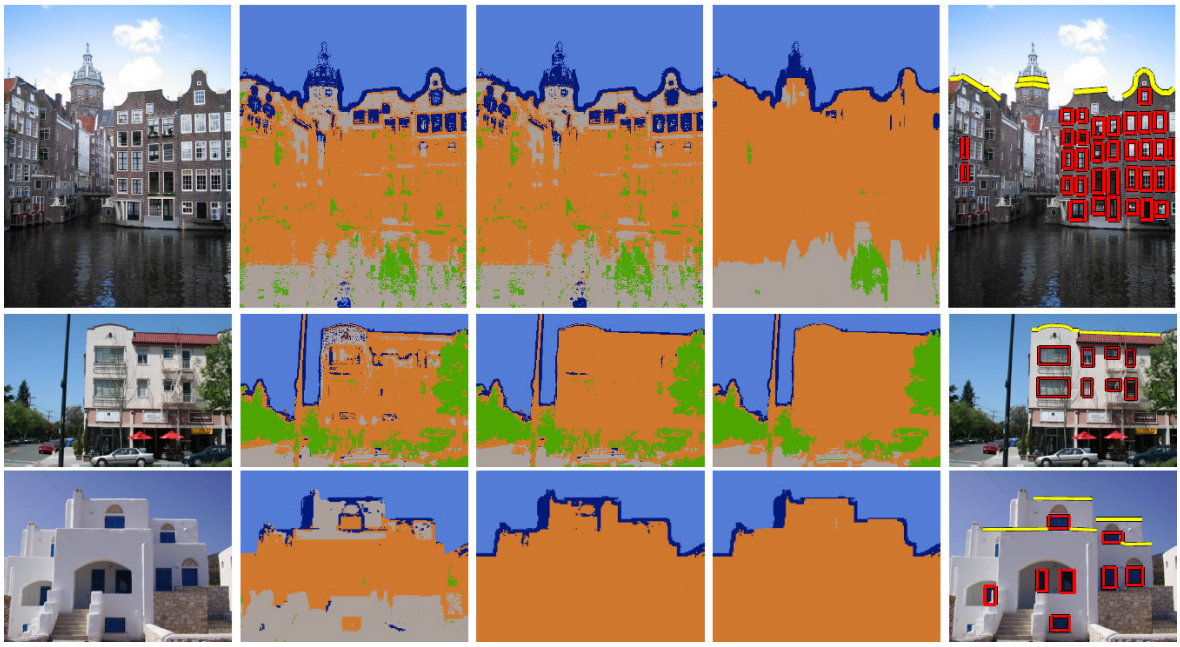


Figure 1.10: Facade image parsing, as shown in Berg et al. (2007). **From left to right:** input image, three intermediate results of the parsing procedure, final result (red: found windows, yellow: roofline).

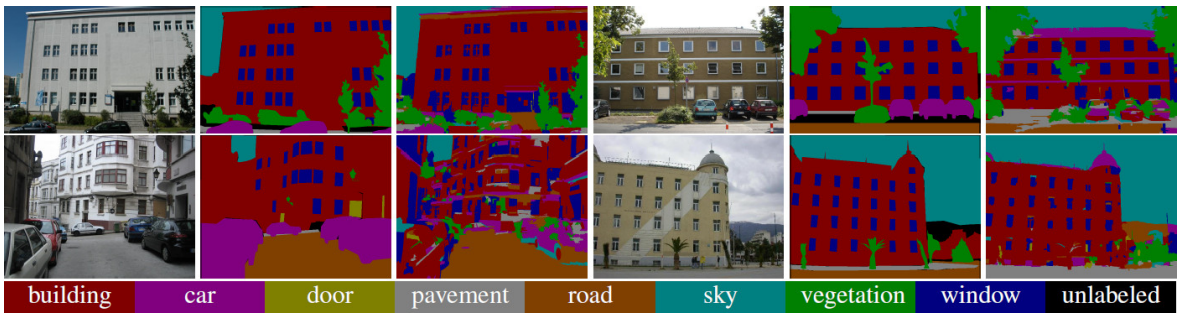


Figure 1.11: Pixelwise labelling of facade images, as shown in Fröhlich et al. (2010). **Columns 1 and 4:** original image. **Columns 2 and 5:** ground truth. **Columns 3 and 6:** result.

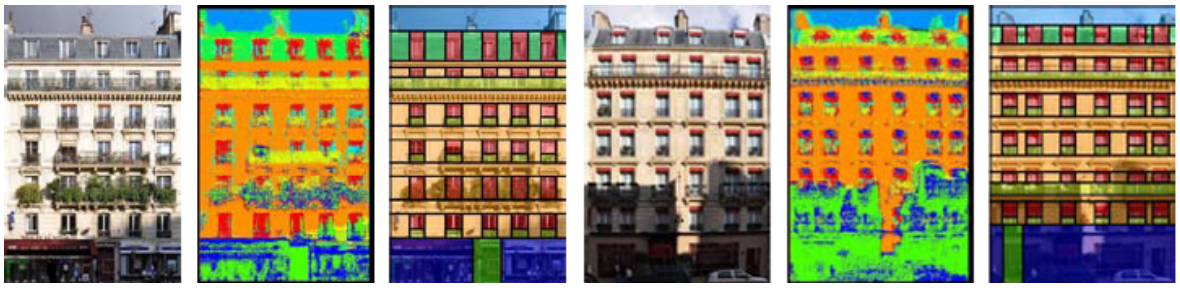


Figure 1.12: Facade image segmentation on Haussmannian buildings, as shown in Teboul et al. (2010). **Columns 1 and 4:** original images. **Columns 2 and 5:** pixelwise classification. **Columns 3 and 6:** segmentation with procedural shape prior.

Figure 1.13: Results of facade image segmentation, as shown in Martinović et al. (2012). They show results on complete building images, and furthermore, on facades showing less structure.

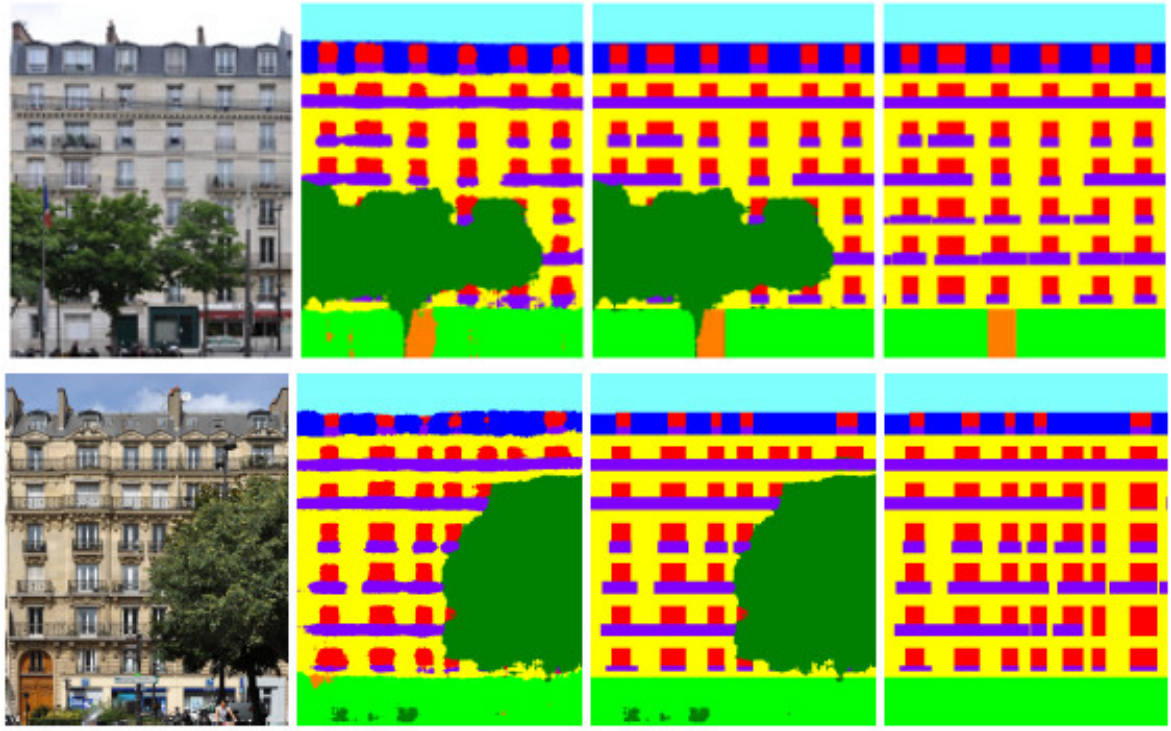


Figure 1.14: Results of facade image segmentation, as shown in Koziński et al. (2015). They explicitly model occlusions, and in the sequel, they predict the facades structure even it is partly occluded. **From left to right:** original image, pixelwise classification, segmentation with occlusion, extracted facade structure.

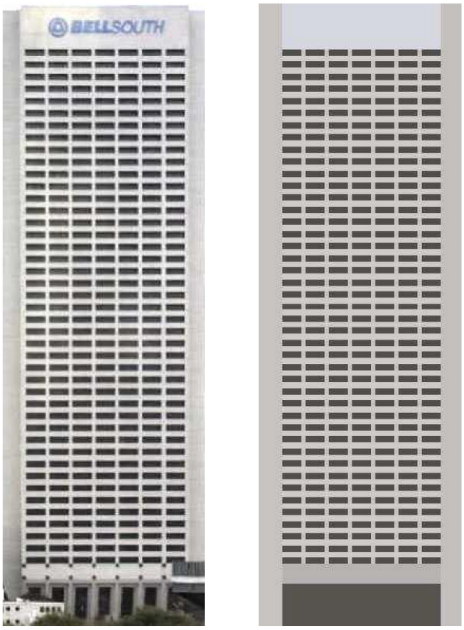


Figure 1.15: Result as shown in Alegre and Dellaert (2004). **Left:** original input image. **Right:** inferred model.

1.3.1.3 Exploiting Repetitive Structures

Some of the aforementioned approaches implicitly exploit the repetitive character of facade elements in terms of similarity of facade objects (Lee and Nevatia, 2004; Wenzel and Förstner, 2008). In order to provide a compact description of facades or to predict hidden or occluded facade objects, another group of works aim at exploiting the repetitive structures within facade images. Using the knowledge about structure of repeated objects, they are able to predict missing detections (Spinello et al., 2010; Musialski et al., 2009). They are partly able to deal with different lattices within one image (Wendel et al., 2010; Spinello et al., 2010; Park et al., 2010), but except Wenzel et al. (2007), there are no nested grids modelled.

Alegre and Dellaert (2004) aim at a semantic interpretation of building facades and a compact representation of a hierarchical regular structure of a facade in terms of exact metric and semantic information, in order to provide highly structured descriptions of buildings. They use rectified images from modern office blocks, thus, showing regular grids of windows, cf. Figure 1.15. They do not claim to detect repeated structures, but they use them for modelling, see the description of methods, cf. Section 1.3.2.

Wenzel et al. (2007) and Musialski et al. (2009) aim at the detection of translational and reflective symmetries from single rectified facade images. Wenzel et al. (2007) derive hierarchically nested repeated translational symmetries, in order to provide a compact image description, cf. Figure 1.16. Musialski et al. (2009) aim at grids with simple structure and propagate the found symmetries over the whole image to remove unwanted image content, e.g., noise or occlusions, cf. Figure 1.17. Also Spinello et al. (2010) and Park et al. (2010) exploit repetitive object patterns for model compression and completion, but are not restricted to rectified images. While Spinello et al. (2010) allow for more complex window grids compared to Musialski et al. (2009), cf. Figure 1.18, Park et al. (2010) deal with different lattices on different object planes, i.e. facades, within one image, cf. Figure 1.19. Recently, Teeravech et al. (2014) aim at decomposing facade images into floors and tiles by discovering the repetitive patterns of the dominant structures, such as windows and balconies.

Different from the approaches before, Wendel et al. (2010) aim at the segmentation of facades. Using single, not necessarily rectified images showing a frontage consisting of different buildings, they use the results of detecting repetitive structures as a feature to split the frontage into facades of

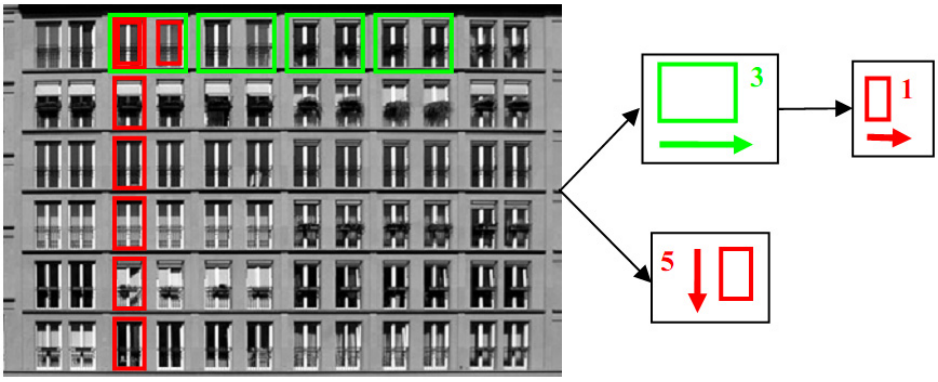


Figure 1.16: Result of nested repetitive structure detection, as shown in Wenzel et al. (2007). The graph on the right visualises the compact description in terms of found repetitions. Vertically, there is one object repeated five times. Horizontally, there is a nested repetition: a larger group which is repeated 3 times and contains another group of two windows, which corresponds to one repetition.

Figure 1.17: Result of propagating found repetitive structure for image restoration, as shown in Musialski et al. (2009).

Figure 1.18: Detecting and completing repetitive structure for image compression, as shown in Spinello et al. (2010). **From left to right:** extracted self-similar objects, conditional random field lattice and inferred position of objects, reconstruction of the image based on model compression.

Figure 1.19: Translational-symmetry-based perceptual grouping, as shown in Park et al. (2010). Different colours mean different groups.

individual buildings. Thereby, they close the gap between real-data and assumptions made for several algorithms, e.g., images showing a single facade.

1.3.1.4 Semantic 3D Reconstruction

Finally, the overall goal is the reconstruction of highly detailed, semantically meaningful, three dimensional models. To the best of our knowledge, Dick et al. (2004) first describe an automatic acquisition of three dimensional semantic architectural models, cf. Figure 1.20. They use short image sequences as input and describe buildings as a set of walls together with what they call a Lego kit of parametrised primitives, such as doors or windows.

Brenner and Ripperda (2006) and Ripperda (2008) address the extraction of building facades in terms of a structural description in the form of a derivation tree which they use to derive a 3D reconstruction, cf. Figure 1.21. They use images as well as LIDAR data as input. Similarly, Schmittwilken and Plümer (2010) aim at the reconstruction and classification of facade parts from 3D point clouds.

Van Gool et al. (2007) and Müller et al. (2007) semi-automatically derive 3D models of high visual quality, including a meaningful hierarchical facade subdivision, from single rectified facade images of arbitrary resolutions, cf. Figure 1.22.

1.3.2 Methods Used for Facade Image Interpretation and Reconstruction

This section reviews the methods used to solve the aforementioned tasks. They partly deal with three-dimensional reconstruction using methods from multi-view geometry, including estimating vanishing points (Werner and Zisserman, 2002; Zhao et al., 2010) or plane sweep (Werner and Zisserman, 2002; Lee and Nevatia, 2004; Mayer and Reznik, 2005; Reznik and Mayer, 2008). We are dealing with single images solely and do not estimate nor use three-dimensional information. Therefore, we do not go into detail of these methods. We are interested in these methods related to single view facade image interpretation.

1.3.2.1 Detecting Regions of Interest

In order to detect regions of interest in the image, such as windows, there are two main trends: either using gradient projection to find aligned edges or using a classificator that detects regions of interest

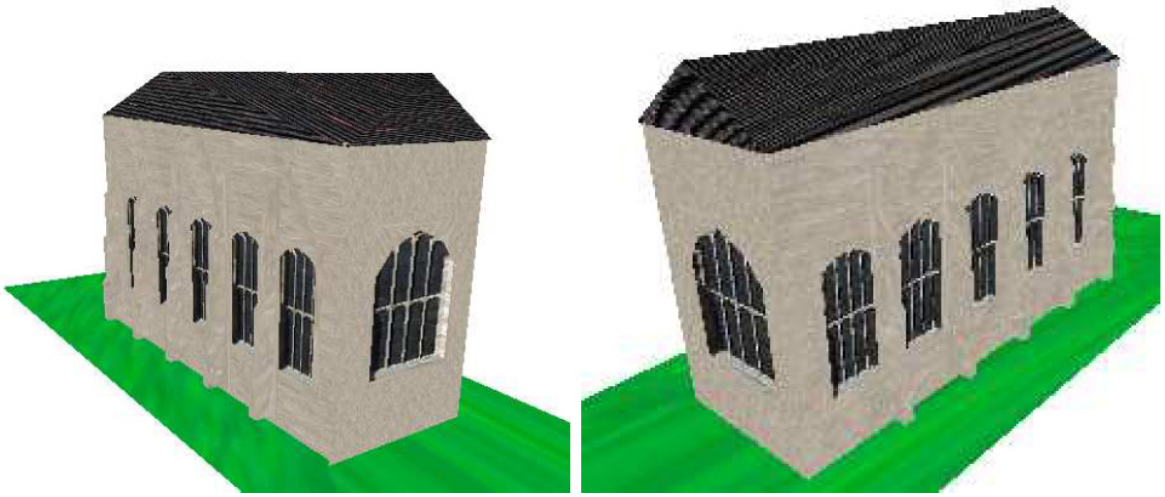


Figure 1.20: Result of semantic 3D reconstruction with added generic texture, as shown in Dick et al. (2004).

Figure 1.21: Grammar based facade interpretation. **Top:** Semantic partitioning of a facade and the according derivation tree, as shown in Brenner and Ripperda (2006). **Bottom:** reconstruction results, as shown in Ripperda (2008)

.

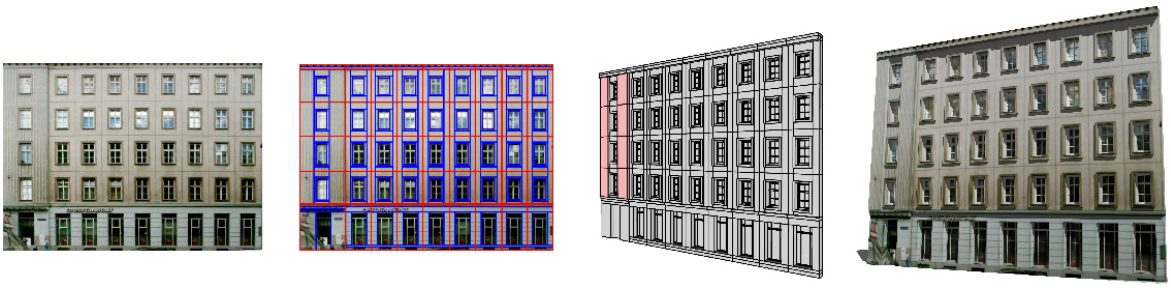


Figure 1.22: Results as shown in Müller et al. (2007). **From left to right:** rectified facade image as input, facade automatically subdivided and encoded as shape tree, resulting polygonal model, rendering of final reconstruction.

searching over the image. The former is restricted to facade types whose windows fulfil an alignment assumption, while the latter do not take any alignment or structure assumption, respectively, into account.

Gradient Projection. Gradient projection was one of the first methods used for window detection, based on the observation that facades are often characterised by grids of windows, aligned horizontally as well as vertically. It was introduced in the pioneering work of Lee and Nevatia (2004).

Defining windows as homogeneous rectangles in the images, window detection is based on horizontal and vertical gradient projection. While Lee and Nevatia (2004) and Teeravech et al. (2014) use rectified images, which allow for gradient projection in order to directly detect outlines of windows, Recky and Leberl (2010) do not assume rectified images and use vertical gradient projection to identify splits within the facades, thus, floors. This also prevents strong assumptions of vertical alignment of facade objects, as they perform horizontal gradient projection within these stripes. The latter is also done by Teeravech et al. (2014). However, using rectified or non rectified images is not essential, as rectification, using vanishing point estimation, is straight forward using facade images (Schaffalitzky and Zisserman, 2000).

In the following, Lee and Nevatia (2004) group found candidates to get different window classes and use them to adjust the window grid. Grouping and classification, respectively, is based on normalised cross-correlation of image content and a dimensional similarity depending on geometric features of each candidate. Furthermore, they detect arc-shape windows within found candidates in a Hough voting scheme. Recky and Leberl (2010) analyse and label resulting image blocks with a greedy procedure based on the gradient content and a k-means clustering of colors in Lab-colorspace, respectively. As already mentioned above, Teeravech et al. (2014) do not aim at window detection, but at decomposing facade images into floors and tiles by discovering the repetitive patterns of the dominant structures, such as windows and balconies. They interpret the accumulated horizontal and vertical histogram profiles of gradients as a series of noisy wave cycles. Assuming them to represent the approximated position and dimensions of repetitive elements, they discover repetitive pattern from the dominant frequency by iteratively fitting candidate sine waves using a RANSAC-styled algorithm.

Obviously, all of these approaches rely on strong alignment assumption. While Lee and Nevatia (2004) presume strong alignments of facade elements in horizontal and vertical direction, Recky and Leberl (2010) and Teeravech et al. (2014), nevertheless, still require strong horizontal alignment to achieve high-quality results. As a sequel Lee and Nevatia (2004), as well as Teeravech et al. (2014), show results on rectified images showing large repetitive structures. The latter even use images containing the facade, solely.

Object Detection Detection of regions of interest, mainly windows, from bottom-up, is done in different ways.

Ali et al. (2007) perform window detection using an AdaBoost classifier. They do not exploit any prior knowledge about regularities or alignments, cf. Figure 1.3. Jahangiri and Petrou (2009) aim at detecting salient regions which they assume to be meaningful parts of the facade. They define "blobs" as regions containing some variation, i.e. local texture, and that are roughly convex. This is thought as input for further analysis by a higher level interpretation module. Their extraction of blobs is based on the use of Gaussian kernels and mathematical morphology and is independent of scale and color. Their final output are bounding boxes of detected blobs, cf. Figure 1.7.

Wenzel and Förstner (2008) propose an incremental learning scheme, based on repeated template matching using normalised cross-correlation, to get a class hierarchy of repetitive structure, cf. Figure 1.5. Given one object of interest, manually labelled by the user, they automatically identify prototypes of image patches and use them to build a hierarchical appearance based model, in a supervised manner.

Compared to gradient projection methods, these approaches yield inferior results in terms of completeness (Ali et al., 2007) and distinctiveness (Jahangiri and Petrou, 2009). On the other hand they are more flexible due to perspective distortion (Ali et al., 2007) and loose structure of facade elements (Jahangiri and Petrou, 2009). Due to repeated template matching, using normalised cross-correlation, Wenzel and Förstner (2008) need rectified facade images again, but do not use any alignment assumptions.

1.3.2.2 Bottom-up Pixelwise Labelling

In contrast to the detection of regions of interest, another group of approaches perform a pixelwise labelling or facade segmentation. They either combine a strong pixelwise classification, e.g., random forest, with an unsupervised segmentation (Fröhlich et al., 2010; Teboul et al., 2010) or perform a hierarchical segmentation. Therfor, they first parse the image into a coarse set of classes which they further parse into more detailed classes using meta knowledge from coarse level of detail (Berg et al., 2007; Zhao et al., 2010).

For example, purely bottom-up, thus, without taking any structural meta knowledge about facades into account, Fröhlich et al. (2010) introduce the combination of a strong pixelwise classification with the result of an unsupervised segmentation. They apply a random forest classifier to each pixel to get classwise probabilities and assign labels to each segment, given by the maximum average probability taken from the pixelwise classification, cf. Figure 1.11.

But, purely bottom-up, these results lack in accuracy. Berg et al. (2007) and Teboul et al. (2010) combine results of pixelwise classification with methods from top-down, see below.

Recently, Jampani et al. (2015) proposed to use a sequence of boosted decision trees with what they call auto-context features. Thereby, they make use of typical architectural structure principals, while avoiding to fix them trough strong constraints. By learning the features in the surrounding of predictions from the layer before, they are able to learn the structure from data. The principle is similar to the idea from deep learning using a convolutional neural network.

1.3.2.3 Detecting Repetitive Structures

Exploiting repetitive structures, in the domain of facade images, usually deal with a large number of observations, trying to find these groups of observations that indicate the dominant translation. Therefor, these approaches either use Hough-like voting schemes (Wenzel et al., 2007; Musialski et al., 2009; Wendel et al., 2010; Spinello et al., 2010) or obtain a robust estimate using RANSAC (Park et al., 2010; Teeravech et al., 2014).

Wenzel et al. (2007) detect reflecting and translational symmetries based on matching SIFT features within a single rectified image. Therefrom, they derive a compact description of hierarchically nested repeated translations from the Hough-votes of found translations. They further evaluate identified translational symmetries based on minimal description length (MDL) as model selection criterion, cf. Figure 1.16. Similarly, Wendel et al. (2010) detect and match interest points within the image, but instead of matching them directly, they use color intensity profiles, between all of them, to get

corresponding profiles. Musialski et al. (2009) use normalised cross-correlation to match regions of small size, within the image, over several scales, similar to Wenzel and Förstner (2008).

Different from these approaches, Park et al. (2010) detect a complementary set of features and group them by mean-shift into potential groups of similar features that possibly form a lattice. Different lattices are evaluated by RANSAC.

All these approaches yield reasonable results and might be used as mid-level tool in order to guide a high-level interpretation module, e.g., to guide the splitting procedure in grammar based approaches, see below.

1.3.2.4 High-Level Facade Interpretation

In order to derive highly detailed, semantic meaningful, partly even three-dimensional models, we need high-level interpretation. Most of these approaches use generative stochastic modelling, grammars or Markov random fields. Also marked point processes are used for facade image interpretation, which belong to generative stochastic modelling, too. But, we discuss these methods in a separate paragraph, as the modelling of prior knowledge from top-down is different from those we mention before.

Generative Stochastic Modelling. We summarise a large group of approaches under the term generative stochastic modelling that use generative models, thus models which are suited to generate synthetic samples of objects they modelled, and which are optimised by stochastic sampling, usually Markov chain Monte Carlo (MCMC), cf. Section 2.6.

Dick et al. (2004) describe the automatic acquisition of three dimensional architectural models from short image sequences. Top-down, they aim at organising a given image into semantically meaningful parts by stochastically varying models. They describe a building as a set of walls together with what they call a Lego kit of parametrised primitives, such as doors or windows. Priors on wall layout, and on the parameters of each primitive are defined by experts and learnt from training data, respectively. Changing the parameters (as width, aspect ratio, brightness, etc.) using MCMC sampling, they try to generate the image which is most similar to the given image. With this work, they prove the great potential and applicability of generative stochastic modelling.

In order to detect windows in image sequences, Mayer and Reznik (2005) train a window classifier and cope with structure using MCMC to estimate parameters of windows, such that they are of same size and aligned. Therefor, they first determine facade planes by robust least squares matching between images of the sequence. The detection of window hypotheses is done by learning an implicit shape model. Changing the parameters of the model, as width, aspect ratio, or brightness within MCMC they try to generate the image which is most similar to the given image. In Reznik and Mayer (2008) they continue this work. Based on plane sweep, similar to Werner and Zisserman (2002), they derive depth information of facade objects, cf. Figure 1.9. Some heuristics and again a MCMC process are used for self diagnosis of found window hypotheses. However, they only address windows as facade elements.

Also for the detection of windows, but in single rectified facade images, Tyleček and Šára (2010) propose a complex generative model, where they include object geometry as well as neighbourhood relations. This way, they have a flexible configuration model for the window positions. They formulate a generative model with parameters for complexity, shape attributes, location attributes, and element neighbourhood relation. The recognition task is formulated as optimisation task of the joint density over the parameters. The joint density is decomposed hierarchically into several prior densities for each parameter group. They apply reversible jump Markov chain Monte Carlo (rjMCMC) sampling, cf. Section 2.6.4, to find the optimal parameters. They show results even on irregular facade structure, cf. Figure 1.8, but they do not show images with occlusions or high variability in window types.

Grammars. A shape grammar is a set of rules to generate geometric shapes. It consists of rules about the shapes itself and about the process how to combine these rules. Wonka et al. (2003) introduced shape grammars into the field of computer graphics for the generation of realistic building models. These grammars are context-dependent and therefore, difficult to obtain efficiently. Alegre and Dellaert

(2004) first demonstrate the potential of stochastic context-free grammars to represent a hierarchical regular structure of a facade. They use rectified images from modern office blocks, thus, showing regular grids of windows which they recursively split into parts using the rules of the grammar and rjMCMC for optimization, cf. Figure 1.15.

Later Brenner and Ripperda (2006) and Ripperda (2008), in their pioneering work to the use of formal grammars for semantic 3D reconstruction, use a formal grammar to derive a structural facade description in the form of a derivation tree. They add variables and constraint equations to the symbols of the grammar. Thus, the derivation tree automatically leads to a constraint equation system. This equation system can be used to optimally fit the entire facade description to given measurement data. The application of derivation steps during the construction of the tree is done by rjMCMC, again. In Ripperda (2008) they further derive prior facade information from a set of facade images in order to support the stochastic modelling process, cf. Figure 1.21. However, they assume 3D data, e.g., from LIDAR, and additionally images as input.

Van Gool et al. (2007) and Müller et al. (2007) combine procedural modelling which is a rule-based modelling technique known from computer graphics, with shape grammars and image analysis to derive a meaningful hierarchical facade subdivision, cf. Figure 1.22. Splitting is done recursively, evaluating horizontal and vertical image content. Based on mutual information they search for irreducible tiles, e.g., vertical stripes which do not contain repetitions. The splitting is represented by a tree which allows the derivation of a set of rules for a grammar. The depth of found basic elements is generated semi-automatic by a manual user input.

Different from that, Teboul et al. (2010) do not aim at semantic 3D reconstruction, but at the segmentation of building facades using procedural shape priors, cf. Figure 1.12. They formulate what they call a constrained generic shape grammar to express special types of buildings and train a random forest classificator to determine a relationship between semantic elements of the grammar and the observed image support. Thus, a pixelwise classification is used as low-level input for the grammar. For model selection they formulate an energy. Thus, segmenting the facade is viewed as choosing those rules which generate a facade that best fits the image in terms of the energy given by the probabilistic output of the classifier. In their preceding work (Teboul et al., 2013), they include reinforcement learning. In this work, in contrast to Teboul et al. (2010), they avoid strong prior assumption about the structure of facades. They show how the procedural rules of a shape grammars can be derived based on the segmentation, rather than vice-versa.

We see that parameters of the grammar are estimated either using MCMC sampling (Dick et al., 2004; Alegre and Dellaert, 2004; Brenner and Ripperda, 2006; Ripperda, 2008) or directly determined during a recursive splitting and merging procedure (Van Gool et al., 2007; Müller et al., 2007). All of them deal with more or less long pipelines including feature detection, 3D reconstruction, maybe a region or pixelwise classification, and high-level reasoning in terms of grammar parsing. Nevertheless, all approaches are restricted either to the type of facades the grammar was designed for or to facades with simple structure.

In the context of 3D building reconstruction from point clouds, Becker (2009, 2010) propose to infer production rules of a formal grammar automatically and fully bottom-up from observed parts of facades. For these parts, they assume point clouds and images to be given. They aim at the geometric refinement of planar building facades, therfor they apply learned rules top-down to generate facade structure for the non observed parts of the building. Also Dehbi and Plümer (2011) proposed to learn the rules of a grammar. Their approach is based on Inductive Logic Programming. They extend context-free grammars by attributes and semantic rules, and link them with logic programs. Recently, Dehbi et al. (2016) continue this work to the automatic learning of the rules of a weighted attributed context-free grammar. They learn parameters and constraints of the grammar using Markov Logic Networks which combine probabilistic graphical models with logic and are strongly related to MRFs. Similar to our work, they are able to model facades with an a priori unknown number objects and learn the prior knowledge that is needed for top-down interpretation, from data.

Dynamic Programming. Dynamic programming is a method for optimization using the divide and conquer principle. It breaks down the whole problem into simpler subproblems which probably occur several times, solves these small problems, stores their solutions and uses them whenever they are needed for solving the whole task.

Recently, Cohen et al. (2014) propose to use dynamic programming for efficiently parsing facade images. They also introduce strong architectural constraints in order to obtain a reliable and consistent interpretation result. But, instead of parsing the facade using a predefined and fixed set of rules in terms of a grammar, they sequentially parse the individual classes according given constraints, which make the process more flexible to different types of facades. However, they still rely on strong architectural structure. Constraints, as well as according parameters, to apply for each type of facade, are predefined by the user.

Markov Random Fields. MRFs are graphical models which express the joint density of random variables within an undirected graph. Thereby, nodes and edges of the graph represent the random variables, i.e. objects, and their neighbourhood relations. They could be labels of pixels in a regular grid or complex objects whose neighbourhood relations are fixed by the graph.

Thus, by introducing neighbourhood relations, Berg et al. (2007) propose the two-step corse-to-fine parsing by first parsing the image into coarse regions, such as sky, foliage, buildings, and street, and second, into a finer level of detail, identifying the positions of windows, doors, rooflines, and the spatial extent of particular buildings, cf. Figure 1.10. Both steps are based on a conditional random field (CRF) formulation, whereas the approach uses unary potentials in the first, while unary and pairwise potentials in the second step. Thus, the detailed parsing considers neighbourhood relations based on the underlying coarse parsing.

Through pixelwise labelling and incorporating, in this case pixelwise, neighbourhood relations, Čech and Šára (2008) detect windows. They identify window panes as characteristic part of windows, which implicitly lead to the detection of windows. Using single orthographic rectified images of building facades, they define window panes as axis-parallel rectangles of relatively low variability in appearance. The detection of window panes is formulated as a task of maximum a posteriori labelling in a Potts model, thus a Markov random field (MRF). Every image pixel has one out of 10 possible labels, and the adjacent pixels are interconnected via links which define allowed label configuration, such that the labels are forced to form a set of non-overlapping rectangles, cf. Figure 1.6. Thus, they get a strongly structured model, forcing the labels to form rectangles. Their approach allow for low variability in appearance of windows, and obviously they are restricted to the special type of windows they modelled.

Yang and Förstner (2011) combine a pixelwise classifier from bottom-up with a hierarchical CRF for top-down interpretation. They use the probabilistic outputs of a random forest classifier as unary potentials and the spatial and hierarchical structures of the regions, they segmented before, within the pairwise potentials. Thus, they encode within the hierarchical model inter-class location information in terms of high-order potentials.

Similarly, Martinović et al. (2012) propose a three-layered approach for semantic segmentation of building facades, cf. Figure 1.13. They start from an oversegmentation of a facade and produce probabilistic interpretations for each segment, using Recursive Neural Networks. The initial labelling is merged with the output of a specialised facade component detectors formulating a MRF. They also introduce weak architectural knowledge, which enforces the final reconstruction to be architecturally plausible and consistent.

Koziński et al. (2015) propose to express architectural prior knowledge into a set of hierarchical rules over different semantic classes that specify which pairs of classes can be assigned to pairs of vertically- and horizontally-adjacent pixels. They transfer those rules into the structure of a MRF. They handle occlusions and therefore, are able to recover partly occluded facades and to infer their structure.

Marked Point Processes. Point Processes are stochastic processes which allow to handle configurations of objects, without specifying their number. They are most flexible in defining the neighbourhood

relations of objects and can be seen as generalization of MRFs, cf. Section 2.5. They are used in image processing for the detection of dense object configurations, as tree crowns on agricultural farms or birds from remotely sensed images or cars in urban environment. Recently they were introduced in the field of facade image interpretation.

In order to detect openings in facades as indicator for the classification of blind facades, Burochin et al. (2014) formulate a stochastic process to sample rectangles in image polygons, formerly identified as facades, from oblique aerial images. Due to perspective distortions the resolution of rectified images is low in vertical direction, images are noise and contain shadows. Given these conditions they define openings as dark rectangles with respect to the wall background. As the detection results are dedicated as indicator for the classification of blind facades, beside other features, they obviously do not require complete and precise detections. But, to the best of our knowledge this is the first approach dealing with marked point processes in the context of facade image interpretation. Their energy model consist, as usual, of a data term and a prior term, weighted by a regularisation factor. The data term depends on the gradient information on the edges of proposed rectangles, while the prior term solely penalizes overlapping rectangles and does not consider any other configuration constraints. Optimization is done by rjMCMC with simulated annealing.

Recently Wang et al. (2015) propose structure-driven facade parsing using a marked point process. In fact they solely deal with windows as facade objects for which they derive multiple grids within the image. They propose an energy formulation whose data term depends on a probability map given by a pixelwise classification. The prior term consist on a repulsive term scoring interacting rectangles in terms of their horizontal and vertical distance. Interestingly, they do not model the assumed grid structure of windows within the prior energy, but through a structure-driven sampling during a multiple birth and death optimisation. Their results are not convincing compared to state-of-the-art results, such as Teboul et al. (2011); Martinović et al. (2012); Koziński et al. (2015). But, the concept of sparse and data-driven sampling via marked point process seems to be promising and worth being continued.

1.3.3 What we Learned, Tasks Solved, and Open Issues

Early approaches deal with the detection of particular facade objects, thereby, either assume large regular structures, at least in one direction (Lee and Nevatia, 2004; Recky and Leberl, 2010), or they allow for low variability in appearance of these objects, or even are restricted to the special type of windows they have modelled (Čech and Šára, 2008).

Pure bottom-up approaches lack in accuracy, as no prior knowledge is used to adjust the results (Fröhlich et al., 2010). Most approaches that deal with pixel- or region-wise bottom-up classification or object detection, respectively, rely on standard features, e.g., colour intensities, gradients, or thereof derived features. These sets of features are defined beforehand and fixed for the approach and the given data. We have seen that learning such features, e.g., by an implicit shape model which is related to bag of words (Mayer and Reznik, 2005), or by boosted classification trees (Jampani et al., 2015) yield reliable results and is highly flexible according the given data.

For the task of semantic interpretation, incorporating prior knowledge through grammars yield state-of-the-art results. However, they rely on fixed, predefined, often hand crafted rules (Brenner and Ripperda, 2006; Ripperda, 2008; Teboul et al., 2010). Their is no generalisation to other facade layouts. In order to produce highly detailed 3D models, procedural modelling combined with shape grammars is still semi automatic (Müller et al., 2007). However, there is clearly a need to learn the rules of a grammar from data (Teboul et al., 2013; Dehbi et al., 2016).

Using MRFs, the structure of the underlying graph is fixed, either predefined in terms of fixed neighbourhood relations pixel- or region-wise (Čech and Šára, 2008; Koziński et al., 2015) or derived from the results of a segmentation (Yang and Förstner, 2011; Martinović et al., 2012).

Most flexible in terms of structure and number of objects are marked point processes. But, their usage in the context of facade image interpretation is still at the very beginning. Recent results are not convincing. There is clearly a need for strong data terms related to the image content and for concepts how to incorporates our prior knowledge about facades structure into the energy. Further, there is a need for the automatic derivation of according rules through learning from data.

We learned that the incorporation of prior knowledge from top-down, assuming evidence from bottom-up is essential in order to obtain reliable and accurate results. Thereby, the top-down approach uses the results of the bottom-up detection step or classification result, respectively, as evidence for the occurrence of objects. Grammars are used for that task most frequently, but learning their rules automatically from data is still in the beginning. We believe that marked point processes could serve as competitive approach to grammars, as they are most flexible in terms of their structure and number of objects.

In order to obtain bottom-up evidence, we believe that learning the representation of objects we aim at, is essential in order to avoid pre-defined, manually chosen and fixed sets of features for the classification.

1.4 General Concept and Organisation of the Thesis

We are now prepared to describe our concept for facade image interpretation. This section provides an overview of the systems architecture and the structure of the thesis. We will motivate all tasks and subtasks and assign them to their chapters and sections.

In the last section, we have seen, and most aforementioned approaches make use of it, that man-made objects, especially facades are well structured. We observe floors, symmetries, and repetitive elements. These structural concepts impose constraints on an image interpretation system. We aim at an image interpretation system which combines evidence from bottom-up and prior knowledge about these constraints from top-down. Thereby, we use evidence from bottom-up given by an object classifier which uses aggregates of line segments as features to describe the image content. We incorporate prior knowledge from top-down through a configuration model that describes typical configurations of facade objects that we learned from training data. Thereby, the term configuration depicts size and location of individual objects and their pairwise relations in terms of alignment, distance, and size differences.

Main focus of this work is the image interpretation from top-down. But, in order to provide evidence for the occurrence of objects in the image, we propose the according framework for representation and classification of image section.

In the following, in order to avoid confusion with the term line, we denote by *straight line segment* the shortest path, between two points on the euclidean plan. Vice versa, the terms line and line segment inherently include curved paths such as circular and elliptical line segments.

Line segments reveal a large invariance to shadows and changes in illumination. Especially windows show a large variety in appearance, in particular due to the mirroring of other objects in the window panes. The overall characteristics of facade objects, inherently man-made, let line segments appear as promising image features. Due to decorative elements and varying shapes of facade objects, in particular windows and doors with arc-type scuncheon, curved lines segments further enrich the expressive power of line features. Further, aggregates of line segments show a large distinctiveness for certain object categories of facades. This motivates our approach for the extraction of our particular features from bottom-up and our representation at mid-level.

Input data are training images and their according annotations which label each pixel as member of one of the target classes. Please note that we represented each object in the image by its bounding box, for learning as well as detection. As target classes we aim at windows, balconies, and entrances which, from our point of view, are the most dominant and most important facade objects. The set of target classes can easily be extended by adding the according annotations to the training data.

We use terrestrial, single view and rectified images. Rectification is done in a preprocessing step using a semi-automatic algorithm which uses vanishing points to estimate to homography to transform the image to an ortho-rectified version. Due to the typical structure of facade images showing lots of

perpendicular edges in the horizontal and vertical direction, this step might be done fully-automatic using an algorithm that estimates the vanishing points, e.g., Schaffalitzky and Zisserman (2000).

We assume the resolution of images to be large enough to enable the detection of a sufficient number of low-level image features. Images we use have image sizes of 1000 to 2000 pixel on their longest image dimension. We obtain image scales by assigning one length in object space to the according distance in the image, e.g., the height of an entrance to its mapping in the image.

1.4.1 The Big Picture

We first present the overview over the image interpretation system, roughly sketching the main steps. In Section 1.4.2 we go into detail of each part to motivate the subsequent tasks.

Figure 1.23 represents the overview of the whole system. We start bottom-up. From each training image we extract line features which are segments of straight lines, circular arcs, and ellipses.

At mid-level, we build pairs of adjacent line segments and learn, which of them are representative for given images and target classes, respectively. We call these representatives shapelets and use them as features for the compact description of image patches showing one object of the target classes each, to train a classifier for object categorisation. Thus, the classifier is able to assign a class label to new, previously unseen image patches.

From top-down, we learn a configuration model from the annotations of given training images. For high-level image interpretation of a new, previously unseen test image, we determine a configuration of class specific image segments, which fits the image content and the structure of the configuration model. Thereby, the fitness of each proposed image segment to its given class label is defined by the output of the classifier and the fitness to the configuration model is given by size and location of each individual image patch and their pairwise relations as alignment, distance, and size differences.

1.4.2 Organisation of the Thesis

According to the system overview, given in Figure 1.23, the thesis is split into three main parts, addressing the low-level image processing in Chapter 3, learning mid-level features in Chapter 4, and the high-level image interpretation in Chapter 5. Theoretical background, thus, methods we need to solve the according problems, are given in Chapter 2. Each of the chapters 3 to 5 contain their own review of related work and an experimental section which evaluates the proposed method.

To motivate our solutions of subtasks of the proposed image interpretation system and further to motivate the organisation of the thesis, we will go into detail of each part of the system. Figure 1.24 schematically shows the organisation of the thesis and the dependencies between subsections of different chapters.

Most image interpretation systems, which use image features, rather than pure pixel information, are based on key point or edge detection. We believe that directly identifying circular and elliptical structure gives rise to much more informative image features from bottom-up. Thus, we aim at the extraction of image features from bottom-up which are more informative than using intensity values solely of single pixels or even line segments. However, this process of inferring low-level features is not automated in a way that we identify which of them are relevant. We rather pre-define their structure, otherwise there would be a step low-level feature learning.

In Chapter 3 we propose a two-step polygon simplification algorithm that approximates a given set of ordered points in 2D, i.e. pixel-chains given by edge extraction, by a sequence of straight line, circular arc, and ellipse segments. These should serve as low-level features for further image interpretation in the subsequent chapters.

Therefor, we propose an adaption of Douglas-Peucker's polyline simplification algorithm, using circle segments instead of straight line segments. It is robust and decreases the complexity of given polygons better than the original algorithm. We call this line simplification algorithm `circlePeucker`. In a second step, we further simplify the poly-curve by merging neighbouring segments to straight line,

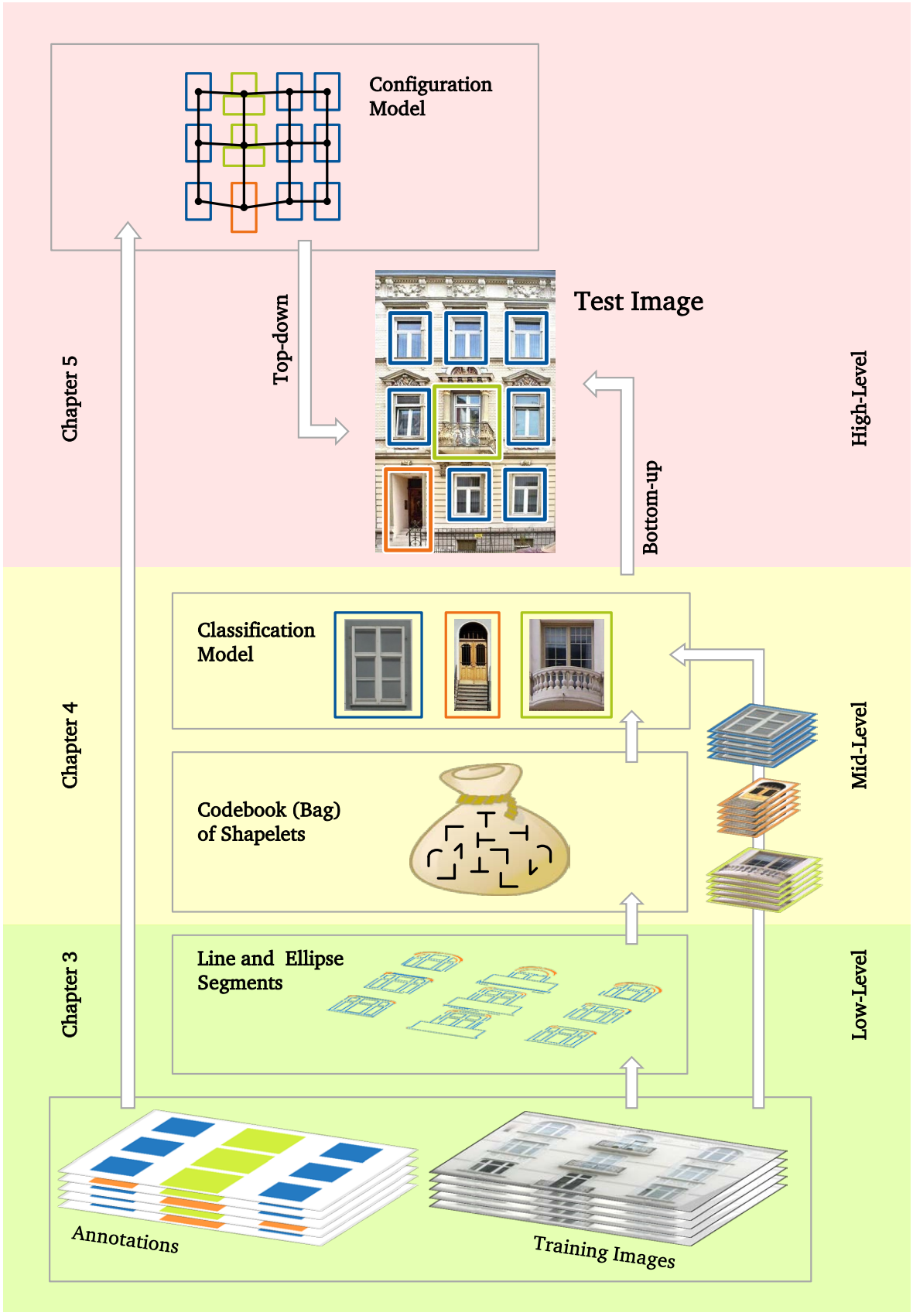


Figure 1.23: System overview.

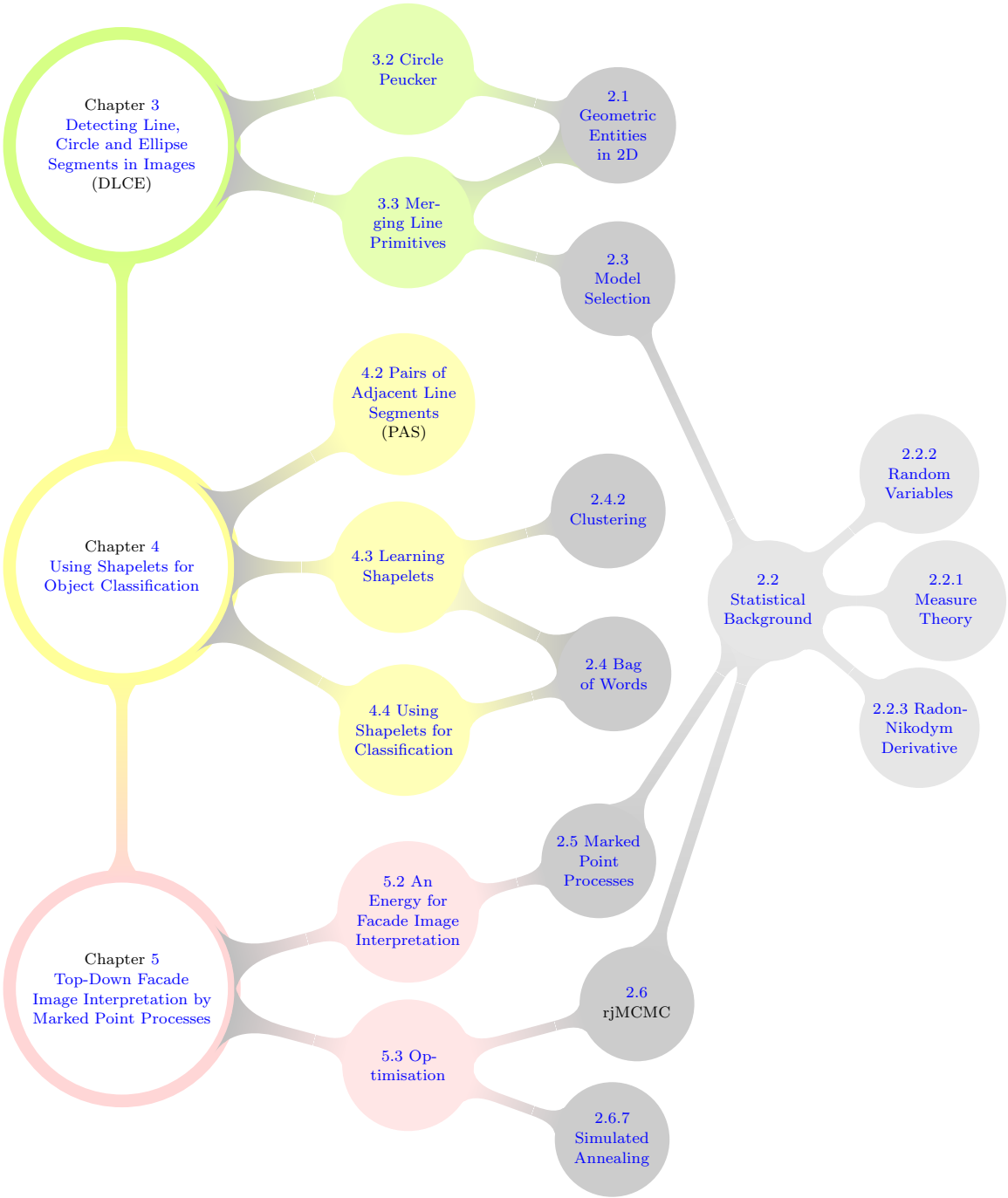


Figure 1.24: Organisation of the thesis. Coloured are the main chapters, according to the system overview in Figure 1.23. Gray are subsections from Chapter 2, collecting all the theoretical background we need to solve our tasks.

circular arc, and ellipse segments, based on the concept of model selection. The whole procedure is called DLCE standing for detecting line, circle, and ellipse segments.

We introduce the mathematical concepts and methods, to represent and to estimate needed geometric entities in 2D, in Section 2.1. Further, we need the concepts of model selection which we review in Section 2.3.

At mid-level, Chapter 4 aims at the categorisation of objects in a facade, using aggregated line primitives as features. This should serve the top-down image interpretation module as a link to the image data.

Widely used object categorisation methods either use a number of object specific features (Fergus et al., 2003) or learn a huge codebook of local image patches (Leibe et al., 2004) which result in a huge search space for matching image features within this codebook. We believe that learning which features best describe the target object classes is more natural than pre-choosing a fixed set of features in advance and learning models based on these features. Further, learning the representation is essential in order to avoid humanly modified models for classification or object detections, as shown by several authors (Amit et al., 2004; Fidler et al., 2006; Gangaputra and Geman, 2006). We learn the parts that represent individual object classes, therefore, avoid fixed and pre-selected features.

The scope of Chapter 4 is to classify subsections of images, based on the histogram of relevant aggregates of line segments in a bag of words (BoW) approach, which we introduce in Section 2.4.

We learn generic parts of object structure which are pairs of adjacent line segments (PAS), from image sections showing individual objects of the target classes. We identify which PAS are relevant to serve as words in a BoW model, thus, are suited to be part of a compact and discriminative image description. We call them shapelets. Finally, we use learned shapelets to classify new, previously unseen image segments of learned target classes using the histogram of shapelets.

Finally, in Chapter 5 the task is to combine evidence from bottom-up for each single object proposal with prior knowledge from top-down. Grammars and MRFs are widely used in image processing for such tasks. Instead, we use marked point processes which can be seen as natural extension of MRFs, not only due to the freedom of the underlying graph structure and its dimensionality, but also due to the way they handle parametric objects. They have shown competitive results in several object detection problems (Lafarge and Gimel'farb, 2008; Lafarge et al., 2010; Börcs and Benedek, 2012; Bredif et al., 2013; Ortner et al., 2007, 2008; Tournaire et al., 2007, 2010; Verdie and Lafarge, 2013). While MRFs are restricted to labelling problems in static graphs, marked point processes handle parametric objects within dynamic graphs. Therefore, we are able to incorporate complex spatial interactions. Typically this is designed manually (Ortner et al., 2007, 2008; Lafarge et al., 2010; Börcs and Benedek, 2012; Verdie and Lafarge, 2013). We introduce to learn typical spatial interactions from training images. This way, we are able to combine the initial belief from bottom-up image categorisation and the prior knowledge we have about typical configurations of facade objects.

Therefor, Chapter 5 presents a novel method for facade image interpretation based on marked point processes, combining bottom-up evidence given by the object classifier, as proposed in Chapter 4, and prior knowledge about typical configurations of facade objects, from top-down. We represent facade objects by a simplified rectangular object model, representing their bounding boxes in the image, and present an energy model which evaluates the agreement of a proposed configuration with the given image and the statistics about typical configurations which we learned from training data. We use the concept of marked point processes to optimise the agreement of configurations of objects, randomly thrown from an appropriate search space, with the given image and learned statistics about typical configurations, in order to identify the configuration that best fits the image content.

For optimisation we use reversible jump Markov chain Monte Carlo sampling combined with simulated annealing, which is a sampling algorithm that allows us to find the global optimum of complex functions with varying dimension. The latter is of special importance, as the number of objects in the image is unknown beforehand.

Theoretical background about these concepts is given in Section 2.5 for marked point processes

and Section 2.6 for optimization using rjMCMC. A particular challenge is the representation of a probability density over a space of configurations with varying number of objects. Thus, we aim at a representation which allows us to model the probability of a configuration which is a collection of parametrised objects whose number is unknown. Both the parameters of the configuration and the number of objects in the configuration are random variables, which make the concept different from classical geodetic applications dealing with probability densities. In order to solve that task, we will need the concept of probabilities based on measure theory. Therefor, we introduce the background in measure theory in Section 2.2.1 and our representation of random variables in Section 2.2.2. Finally, the Radon-Nikodym Derivative, cf. Section 2.2.3, allows us to represent a probability density function in terms of a probability measure.

1.5 Summary

In this section, we motivated the task of automatic, accurate, and detailed facade image interpretation as valuable ingredient for the production of highly detailed, visual attractive and semantic meaningful 3D city models and showed several applications of 3D city modelling. We formulated the goal of the thesis, which is the accurate detection of facade objects from single-view images incorporating prior knowledge from top-down.

We reviewed the existing approaches in the field of facade image interpretation and reconstruction. State of the art methods use combined bottom-up – top-down approaches, in order to incorporate prior knowledge about facades typical structure. Further, at the level of object classification, learning a representation that is suited to describe target objects yield state-of-the-art results. We believe that both learning learning a representation from bottom-up and learning prior knowledge from top-down from data, rather than manually modelling by hand, will overcome typical limitations of existing approaches.

We proposed a three-step system which in the first step, extracts line features from bottom-up, which are segments of straight lines, circular arcs, and ellipses. Secondly, at mid-level, we learn what we call shapelets from pairs of adjacent line segments, which are representatives for the description of facade objects. Finally, at high-level, we learn typical configurations of facade objects from training data and use them within a marked point process, such that they fit the image content as well as the typical configuration of facade objects.

To guide the reader through the following chapters, we introduced our main concept, motivated the according subtasks, and assign them to the chapters and sections of this thesis.

CHAPTER 2

Theoretical Background

This chapter collects the mathematical and technical concepts we will use throughout this thesis. It contains the representation and estimation of geometric entities, as circle and ellipse segments, statistical terms and notation, and model selection. We introduce the bag of words (BoW) approach as model for compact image representation, such that we are able to use this representation as feature for image classification. For classification we provide the basic concepts of support and import vector machines. Further, we review the theory of spatial and marked point processes as statistical model in stochastic geometry for the analysis of objects in images concerning both their geometry and their interactions. Finally, we review the theory of Markov Chain Monte Carlo sampling and simulated annealing in order to optimise complex probability density functions.

2.1 Geometric Entities in 2D . . . 47
2.1.1 Notation of Geometric Entities . . . 47
2.1.2 Approximation of Curved Paths . . . 47
2.1.3 Ellipses and Ellipse Segments . . . 47
2.1.4 Circles and Circle Segments . . . 51
2.1.5 Conditioning . . . 56
2.2 Statistical Background . . . 58
2.2.1 Measure Theory . . . 58
2.2.2 Random Variables . . . 58
2.2.3 Radon-Nikodym Derivative . . . 60
2.3 Model Selection . . . 60
2.4 Bag of Words . . . 61
2.4.1 Features Used as Words . . . 64
2.4.2 Clustering . . . 64
2.4.3 Normalising the BoW histogram . . . 66
2.4.4 Classification Methods . . . 66
2.5 Marked Point Processes . . . 71
2.5.1 Background Theory on Marked Point Processes . . . 73
2.5.2 Stability of the Point Process . . . 75
2.5.3 Markov Point Processes . . . 76
2.5.4 Relation to Markov Random Fields . . . 76

2.6 Optimisation Using Reversible Jump Markov Chain Monte Carlo Sampling and Simulated Annealing 77
2.6.1 Markov Chains 80
2.6.2 Metropolis-Hastings Algorithm 81
2.6.3 Mixtures of Kernels 83
2.6.4 Reversible Jump Markov Chain Monte Carlo Sampling 83
2.6.5 Green's Birth and Death Sampler 84
2.6.6 A Marked Point Process Sampler 86
2.6.7 Simulated Annealing 87

2.1 Geometric Entities in 2D

Given pixel-chains from edge detection, Chapter 3 proposes a method to segment them into straight line, circular arc, and ellipse segments. Found line segments should serve as low-level features for further image interpretation, as already stated in Section 1.4.

This section presents the concepts and methods, we need to represent and to estimate needed geometric entities.

2.1.1 Notation of Geometric Entities

We denote names of geometric objects regardless their mathematical representation calligraphic, euclidean vectors and matrices cursive bold and homogeneous vectors and matrices upright bold. All geometric elements we use are 2-dimensional. They are written with small letters. Sets are written with capital letters. For example, a point is called $\mathcal{x}$, its euclidean vector is written as $\boldsymbol{x}$ and his homogeneous representation is $\mathbf{x}$. A set of points is collected by the set $\mathcal{X} = \{\mathcal{x}_i\}, i \in 1 \dots I$ which might be unsorted, but no element is the same. Their geometric representations are either $\boldsymbol{X} = [\boldsymbol{x}_i]_{i \in 1 \dots I}$ or $\mathbf{X} = [\mathbf{x}_i]_{i \in 1 \dots I}$ which are matrices whose elements are stacked column wise, or vectors in case the elements are scalar. We use round brackets $(\cdot)$ for lists, thus, collections of elements which may have arbitrary type.
A list of typical geometric elements, we use throughout the thesis, is given on page 11.

2.1.2 Approximation of Curved Paths

The term *path* denotes an arbitrary connection between two points. This could be curved or not. The shortest path, between two points on the euclidean plan, we denote by *straight line segment*. We deal with curved paths in terms of circles and ellipses, whereby ellipses include circles and straight lines. We approximate paths as collections of straight lines, circle and ellipse segments which we represent as homogeneous elements.
Please note that we exclude parabolas and hyperbolas which are results of mapping a 3D circle to a conic in general, too. But, assuming a straight line preserving mapping, we observe ellipses in general and circles and straight lines as special cases.
A detailed discussion of uncertain homogeneous points and lines can be found in Förstner and Wrobel (2016) and Meidow et al. (2009). We summarise the results, we will need throughout this thesis, in Appendix B.

In the following, we give some more details about circle and ellipse segments including its parametrisation and estimation procedure which will be used in Chapter 3 to segment pixel-chains, extracted from the image, into segments of lines, circles, and ellipses. We start with the most general curved element which includes circles and lines.

2.1.3 Ellipses and Ellipse Segments

Regular conics are ellipses, hyperbolas or parabolas. Singular conics, among others, might be two straight lines. In this section, we are interested in ellipses, solely.

Depending on the application we choose different representations for an ellipse which are given in Appendix B.2.

Mainly, we use the homogeneous representation of conics to express the parameters of the ellipse. Thereby, we represent conics with the symmetric 3×3-matrix

$$\mathsf{C} = \left[\begin{array}{cc|c} c_{11} & c_{12} & c_{13} \\ c_{12} & c_{22} & c_{23} \\ \hline c_{13} & c_{23} & c_{33} \end{array}\right] = \left[\begin{array}{cc} \mathsf{C}_{hh} & \boldsymbol{c}_{h0} \\ \boldsymbol{c}_{0h}^{\mathsf{T}} & c_{00} \end{array}\right] . \tag{2.1}$$

Any point $\mathbf{x}$ on the conic fulfils

$$\mathbf{x}^{\mathsf{T}} \mathsf{C} \mathbf{x} = 0 . \tag{2.2}$$

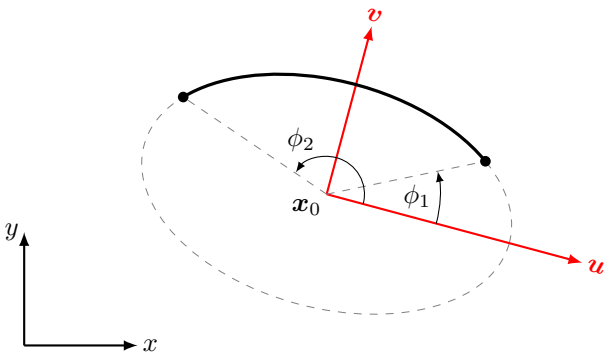


Figure 2.1: An ellipse segment is given by the parameters of the full ellipse and the angles ϕ_1 and ϕ_2 of its start- and end-point.

From Equation (2.1) we see that C is described by 6 free parameters. Due to the homogeneous representation, this results in 5 degrees of freedom.

To ensure the conic to be an ellipse, the homogeneous part of the conic must fulfil

$$|\mathsf{C}_{hh}| > 0\,. \tag{2.3}$$

Thus, when forcing the conic to be an ellipse, we might constrain

$$|\mathsf{C}_{hh}| = 1\,, \tag{2.4}$$

which is a valid choice, as the conic representation is homogeneous.

2.1.3.1 Representation of Ellipse Segments

Ellipse segments are assumed to be uncertain, due to the uncertainty of the original image data. We represent them by an uncertain ellipse and two fixed angles ϕ_1 and ϕ_2 indicating their start and end point, cf. Figure 2.1. Thus, an ellipse segment $\mathcal{C}^S$ is given by its conic and two angles

$$\mathcal{C}^S = (\mathsf{C}, \Sigma_{cc}, \phi_1, \phi_2)\ , \tag{2.5}$$

with covariance matrix Σ_{cc} referring to the elements of the ellipse, given by conic C.

2.1.3.2 Algebraic Solution for Ellipse Fitting

For fitting an ellipse trough a set of given points, we aim at a maximum likelihood estimation, following a Gauss-Helmert model with the constraint (2.4). As this is obviously a non linear problem, we will need initial parameters which we will estimate using the direct method of Fitzgibbon (Fitzgibbon and Fisher, 1999). As a result, we not only obtain the ellipse parameters, but also the estimated variance $\widehat{\sigma}_0^2$ of the data and the covariance matrix $\Sigma_{\widehat{c}\widehat{c}}$ of the parameters which we use for subsequent tests.

First, we will summarise the direct algebraic solution for fitting an ellipse through a set of 2D points of Fitzgibbon and Fisher (1999). We start with the implicit polynomial representation of the conic given in (B.42)

$$\boldsymbol{c}^\mathsf{T}\,\boldsymbol{y} = 0, \tag{2.6}$$

collecting the parameters of the conic within vector

$$\boldsymbol{c} \;=\; [c_{11},\, c_{12},\, c_{22},\, c_{13},\, c_{23},\, c_{33}]^\mathsf{T} \tag{2.7}$$

and the observations within

$$\boldsymbol{y} \;=\; [x^2,\, 2xy,\, y^2,\, 2x,\, 2y,\, 1]^\mathsf{T}\ , \tag{2.8}$$

given by an observed point $\boldsymbol{x} = [x, y]^\mathsf{T}$.

Both, (2.2) and (2.6), define an algebraic distance of a point χ to the conic. When fitting a conic through a set of points we try to minimise this algebraic distance.

According to (2.3), we force the conic to be an ellipse when restricting its parameters to

$$c_{11}c_{22} - 2c_{12}^2 > 1\,. \tag{2.9}$$

Due to the homogeneous character of the conic representation we are free to scale the parameters; thus, we can change the inequality constrain by the equality constraint

$$c_{11}c_{22} - 2c_{12}^2 = 1 \tag{2.10}$$

as supposed by Fitzgibbon and Fisher (1999).

Other constrains where suggested, e.g., $|\boldsymbol{c}|^2 = 1$, $c_{11} + c_{22} = 1$ or $c_{33} = 1$, c. f. (Rosin, 1993; Gander et al., 1994). But, these constrains are meant to avoid the trivial solution $\boldsymbol{c} = \mathbf{0}_6$, but do not restrict the conic to an ellipse.

The constraint (2.10) can be expressed in matrix form as

$$\boldsymbol{c}^\mathsf{T} F \boldsymbol{c} = \boldsymbol{c}^\mathsf{T} \begin{bmatrix} 0 & 0 & 1/2 & 0 & 0 & 0 \\ 0 & -2 & 0 & 0 & 0 & 0 \\ 1/2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \boldsymbol{c} = 1\,. \tag{2.11}$$

In the following, Fitzgibbon and Fisher (1999) solve the optimisation problem

$$\text{minimise} \quad |Y\boldsymbol{c}|^2 \tag{2.12}$$

$$\text{subject to} \quad \boldsymbol{c}^\mathsf{T} F \boldsymbol{c} = 1\,, \tag{2.13}$$

using the design matrix $Y = [\boldsymbol{y}_1, \ldots, \boldsymbol{y}_I]$, build from I points $\boldsymbol{x}_i$. The corresponding Lagrangian function is given by

$$L(\boldsymbol{c}, \lambda) = |Y\boldsymbol{c}|^2 + \lambda(\boldsymbol{c}^\mathsf{T} F \boldsymbol{c} - 1)\,. \tag{2.14}$$

Differentiating with respect to $\boldsymbol{c}$ leads to [1]

$$\frac{\partial L}{\partial \boldsymbol{c}} = 2Y^\mathsf{T} Y\boldsymbol{c} + \lambda(F + F^\mathsf{T})\boldsymbol{c} = 2Y^\mathsf{T} Y\boldsymbol{c} + 2\lambda F\boldsymbol{c}\,. \tag{2.15}$$

And differentiating with respect to λ leads to

$$\frac{\partial L}{\partial \lambda} = \boldsymbol{c}^\mathsf{T} F \boldsymbol{c} - 1\,. \tag{2.16}$$

Minimising both leads to the system

$$Y^\mathsf{T} Y\boldsymbol{c} = \lambda F\boldsymbol{c} \tag{2.17}$$

$$\boldsymbol{c}^\mathsf{T} F \boldsymbol{c} = 1\,, \tag{2.18}$$

where we skipped the sign in front of λ, due to the interpretation as eigenvalue equation. We solve (2.17) for its generalised eigenvalues which leads to at most six pairs $(\lambda_i, \boldsymbol{u}_i)$ of eigenvalues and eigenvectors. Due to the homogeneous character of the equations any multiple α of $\boldsymbol{u}_i$ is a solution, too. Setting

$$\boldsymbol{c} = \alpha \boldsymbol{u}_i\,, \tag{2.19}$$

we obtain from (2.18)

$$\alpha_i = \sqrt{\frac{1}{\boldsymbol{u}_i^\mathsf{T} F \boldsymbol{u}_i}}\,. \tag{2.20}$$

Finally, we take this solution out of six with positive eigenvalue, to guaranty the existence of square root (2.20). Fitzgibbon and Fisher (1999) proved that there exist exactly one positive eigenvalue which gives an unique solution.

[1] Here we used the identities (A.1),(A.2),(A.3).

2.1.3.3 Gauss-Helmert-Model for Ellipse Fitting

The direct solution, given in last section, provides initial values for a statistically best fit in terms of least squares. In the following, we will formulate a Gauss-Helmert model which enables not only the estimation of the parameters, but also of the according covariance information which we need for further statistical testing.

The classical solution requires six implicit equations given by (2.1) or (2.6) for example. An additional constraint is needed due to rank deficiency caused by homogeneity. The according estimate has been shown to be unstable, in case of flat or short ellipse segments, as the elements c_{11}, c_{23} and c_{33} become very large. The approach, given in this section, does not suffer from these problems; thus, we are able to fit ellipses even in cases the segments are extremely flat.

To avoid the use of constraints within the Gauss-Helmert model for parameter fitting, we re-parametrise the ellipse equation, which results in a five-parameter vector.

Re-parametrising the Ellipse. We seek for a change of the parameter-vector from

$$\mathbf{c} = [c_{11},\, c_{12},\, c_{22},\, c_{13},\, c_{23},\, c_{33}]^{\mathsf{T}} \qquad \text{to} \qquad \boldsymbol{u} = [b,\, c,\, d,\, e,\, f]^{\mathsf{T}} \tag{2.21}$$

and solve a classical Gauss-Helmert model without constraints.

We start with the constraint given in (2.4)

$$\begin{vmatrix} c_{11} & c_{12} \\ c_{12} & c_{22} \end{vmatrix} = 1\,. \tag{2.22}$$

These are 3 parameters which we wish to reduce to 2 parameters b and c. Therefor, we choose $c_{12} = b$, $c_{11} = x - c$ and $c_{22} = x + c$ and obtain the new constraint

$$\begin{vmatrix} \Delta - c & b \\ b & \Delta + c \end{vmatrix} = 1\,. \tag{2.23}$$

To make this plausible, assume $b = 0$ and $c = 0$. then

$$1 = \begin{vmatrix} c_{11} & c_{12} \\ c_{12} & c_{22} \end{vmatrix} = \begin{vmatrix} 1 & 0 \\ 0 & 1 \end{vmatrix}\,, \tag{2.24}$$

which is a circle. Solving (2.23) for Δ results in

$$\Delta = \pm\sqrt{1 + b^2 + c^2} \tag{2.25}$$

and leads to the re-parametrised conic

$$\mathsf{C} = \begin{bmatrix} \sqrt{1 + b^2 + c^2} - c & b & d \\ b & \sqrt{1 + b^2 + c^2} + c & e \\ d & e & f \end{bmatrix} \tag{2.26}$$

where we choose the positive solution of (2.25) to avoid a negative $\mathsf{C}_{11}-$element.

The transformed parameter vector, which we wish to estimate in the following, is given by

$$\begin{aligned} \boldsymbol{u}(\mathbf{c}) &= [b,\, c,\, d,\, e,\, f]^{\mathsf{T}} && (2.27)\\ &= \left[c_{12},\, \frac{(1 - c_{11}^2 + c_{12}^2)}{2c_{11}},\, c_{13},\, c_{23},\, c_{33}\right]^{\mathsf{T}}\,, && (2.28) \end{aligned}$$

and vice versa

$$\begin{aligned} \mathbf{c}(\boldsymbol{u}) &= [c_{11},\, c_{12},\, c_{22},\, c_{13},\, c_{23},\, c_{33}]^{\mathsf{T}} && (2.29)\\ &= \left[\sqrt{1 + b^2 + c^2} - c,\, b,\, \sqrt{1 + b^2 + c^2} + c,\, d,\, e,\, f\right]^{\mathsf{T}}\,. && (2.30) \end{aligned}$$

Gauss-Helmert model. Given N observed points $\mathsf{X} = [\boldsymbol{x}_n]\,, n = 1 \ldots N$, using the reduced parameter set $\boldsymbol{u} = [b,\, c,\, d,\, e,\, f]^\mathsf{T}$, and collecting all observed coordinates within the $(2N) \times 1$ vector of observations $\boldsymbol{l}$, with $\boldsymbol{l}_n = [x,\, y]_n^\mathsf{T}$, we represent the ellipse by its implicit polynomial representation, cf. (2.6),

$$\begin{aligned} &g_n(\boldsymbol{l}_n, \boldsymbol{u}) : \\ &\left(\sqrt{1+b^2+c^2} - c\right) x_n^2 + 2b\; x_n y_n + \left(\sqrt{1+b^2+c^2} + c\right) y_n^2 + 2d\; x_n + 2e\; y_n + f = 0\;, \end{aligned} \tag{2.31}$$

which fits the typical Gauss-Helmert constraint

$$\boldsymbol{g}(\boldsymbol{l} + \widehat{\boldsymbol{v}}, \widehat{\boldsymbol{u}}) = 0\;. \tag{2.32}$$

We aim at adjusted parameter $\widehat{\boldsymbol{u}}$, as well as adjusted observations $\widehat{\boldsymbol{l}} = \boldsymbol{l} + \widehat{\boldsymbol{v}}$, given by their residuals $\widehat{\boldsymbol{v}}$.

The solution for this model is given in Appendix C.2. As result, we obtain the adjusted parameter vector $\widehat{\boldsymbol{u}}$ together with its covariance matrix $\boldsymbol{\Sigma}_{\widehat{u}\widehat{u}}$, the adjusted observations $\widehat{\boldsymbol{l}}$ and the estimated variance coefficient $\widehat{\sigma}_0^2$.

At the end, we obtain the original parameters $\widehat{\boldsymbol{c}}$ of the conic using (2.30) and $\boldsymbol{\Sigma}_{\widehat{c}\widehat{c}}$ by variance propagation

$$\underset{6\times6}{\boldsymbol{\Sigma}_{\widehat{c}\widehat{c}}} = \underset{6\times5}{J_{uc}} \underset{5\times5}{\boldsymbol{\Sigma}_{\widehat{u}\widehat{u}}} \underset{5\times6}{J_{uc}^\mathsf{T}} \tag{2.33}$$

using the Jacobian

$$J_{uc} = \begin{bmatrix} b/k & c/k & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ b/k & c/k & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \quad \text{with} \quad k = \sqrt{1+b^2+c^2}\;. \tag{2.34}$$

2.1.4 Circles and Circle Segments

A circle is a special regular conic for which $\mathsf{C}_{hh} \propto I_2$. In this section, we introduce our representation of circles and circle segments and the estimation of their parameters.

Instead of using the over-parametrised conic representation, we represent circles by their implicit homogeneous equation

$$A(x^2 + y^2) + Bx + Cy + D = 0\;, \tag{2.35}$$

which is introduced in Appendix B.3.2. Every point $\mathbf{x} = [x\; y\; 1]^\mathsf{T}$, sitting on the circle, fulfils (2.35). Collecting the coordinates $\boldsymbol{x} = [x\; y]^\mathsf{T}$ in a vector $\mathbf{y}(\boldsymbol{x})$

$$\mathbf{y}(\boldsymbol{x}) = \begin{bmatrix} x^2 + y^2 \\ x \\ y \\ 1 \end{bmatrix} \tag{2.36}$$

and the parameter within vector $\mathbf{p}$

$$\mathbf{p} = \begin{bmatrix} A \\ B \\ C \\ D \end{bmatrix} \tag{2.37}$$

we obtain the implicit, linear, homogeneous form

$$\mathbf{y}(\boldsymbol{x})^\mathsf{T} \mathbf{p} = 0\,. \tag{2.38}$$

Note that setting $A = 0$ allows us to represent circles with infinite radius, thus, lines.

2.1.4.1 Direct Solution for Estimating the Homogeneous Circle Elements

Given a set of points $\boldsymbol{x}_n = [x_n\ y_n]^\mathsf{T}$, $n = 1 \dots N$, supposed to sit on a circle, with covariance matrices $\boldsymbol{\Sigma}_{l_n l_n}$ which we assume to be $\sigma_n^2 \boldsymbol{I}_2$, we aim at estimating the circles parameter vector $\mathbf{p} = [A\ B\ C\ D]^\mathsf{T}$. A direct solution is given in Förstner and Wrobel (2016, Section 3.6.2.5) and Al-Sharadqah and Chernov (2009), which we summarise here shortly. A more detailed derivation of the equations is given in Appendix B.3.3.

Using the constraints

$$g_n(\boldsymbol{x}_n, \mathbf{p}) = A(x_n^2 + y_n^2) + Bx_n + Cy_n + D = \mathbf{y}(\boldsymbol{x}_n)^\mathsf{T}\mathbf{p}\ , \tag{2.39}$$

as already seen in (2.35) and (2.38), we wish to minimise the squared sum of weighted algebraic distances

$$\Omega^2(\widehat{\mathbf{p}}) = \sum_n \frac{g_n^2(\widehat{\boldsymbol{x}}_n, \widehat{\mathbf{p}})}{\sigma_{g_n}^2} = \frac{\widehat{\mathbf{p}}^\mathsf{T} \boldsymbol{M}\ \widehat{\mathbf{p}}}{\widehat{\mathbf{p}}^\mathsf{T} \boldsymbol{N}\ \widehat{\mathbf{p}}}\ , \tag{2.40}$$

using

$$\boldsymbol{M} = \sum_n \frac{\widehat{\mathbf{y}}_n \widehat{\mathbf{y}}_n^\mathsf{T}}{\sigma_n^2} = \begin{bmatrix} \overline{z^2} & \overline{xz} & \overline{yz} & \overline{z} \\ \overline{xz} & \overline{x^2} & \overline{xy} & \overline{x} \\ \overline{yz} & \overline{xy} & \overline{y^2} & \overline{y} \\ \overline{z} & \overline{x} & \overline{y} & 1 \end{bmatrix}\ , \tag{2.41}$$

where we introduced the weighted mean values

$$\overline{z} = \frac{1}{N}\sum_n \frac{\widehat{x}_n^2 + \widehat{y}_n^2}{\sigma_n^2} \qquad \overline{x} = \frac{1}{N}\sum_n \frac{\widehat{x}_n}{\sigma_n^2} \qquad \overline{y} = \frac{1}{N}\sum_n \frac{\widehat{y}_n}{\sigma_n^2}\ . \tag{2.42}$$

For matrix $\boldsymbol{N}$ Al-Sharadqah and Chernov (2009) proposed to used

$$\boldsymbol{N} = 2\boldsymbol{N}_T - \boldsymbol{N}_P\ , \tag{2.43}$$

with

$$\boldsymbol{N}_P = \begin{bmatrix} 0 & 0 & 0 & -2 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -2 & 0 & 0 & 0 \end{bmatrix} \quad \text{and} \quad \boldsymbol{N}_T = \begin{bmatrix} 4\overline{z} & 2\overline{x} & 2\overline{y} & 0 \\ 2\overline{x} & 1 & 0 & 0 \\ 2\overline{y} & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \tag{2.44}$$

and showed that this choice leads to an unbiased solution.

Finally, the optimal parameters are given by minimising (2.40), which leads to the generalised eigenvalue problem

$$\boldsymbol{M}\mathbf{p} = \lambda \boldsymbol{N}\mathbf{p}\,. \tag{2.45}$$

The estimated parameters $\widehat{\mathbf{p}} = \left[\widehat{A}\ \widehat{B}\ \widehat{C}\ \widehat{D}\right]^\mathsf{T}$ are given by the eigenvector belonging to the smallest eigenvector.

The solution, given in this section, is based on an algebraic minimisation. We do not estimate adjusted observations nor the Jacobians we usually use within the adjustment procedure. Therefore, we are not able to give the covariance of the estimated parameters directly. Instead we follow Förstner and Wrobel (2016, Section 3.6.2.5) and derive the covariance matrix of the circles parameters $[x_0, y_0, r]$ directly from observed points. To do so, we assume the observations without outlier, thus, accurate enough to assume the estimated parameters to be a good approximation of the true values.

First, we obtain centre $(\widehat{x}_0, \widehat{y}_0)$ and radius $\widehat{r}$ by

$$\widehat{x}_0 = -\frac{\widehat{B}}{2\widehat{A}} \qquad \widehat{y}_0 = -\frac{\widehat{C}}{2\widehat{A}} \qquad \widehat{r} = \sqrt{\frac{\widehat{B}^2 + \widehat{C}^2 - 4\widehat{A}\widehat{D}}{4\widehat{A}^2}}\,. \tag{2.46}$$

using (B.62). Given the implicit circle equations

$$g_n(\boldsymbol{x}_n, \widehat{\boldsymbol{p}}) = (x_n - \widehat{x}_0)^2 + (y_n - \widehat{y}_0)^2 - \widehat{r}^2 = 0\,, \tag{2.47}$$

using the classical circle parameters $\widehat{\boldsymbol{p}} = [\widehat{x}_0\ \widehat{y}_0\ \widehat{r}]$, we obtain the covariance of estimated parameters by implicit variance propagation, cf. Appendix B.3.3,

$$\Sigma_{pp} = \begin{bmatrix} \overline{c^2} & \overline{cs} & \overline{c} \\ \overline{cs} & \overline{s^2} & \overline{s} \\ \overline{c} & \overline{s} & \overline{w} \end{bmatrix}^{-1}\,, \tag{2.48}$$

with abbreviations

$$\begin{aligned} \overline{s} &= \sum_n \frac{s_n}{\sigma_n^2} & \overline{c} &= \sum_n \frac{c_n}{\sigma_n^2} & \overline{cs} &= \sum_n \frac{c_n s_n}{\sigma_n^2} \\ \overline{c^2} &= \sum_n \frac{c_n^2}{\sigma_n^2} & \overline{s^2} &= \sum_n \frac{s_n^2}{\sigma_n^2} & \overline{w} &= \sum_n \frac{1}{\sigma_n^2} \\ s_n &= \frac{x_n - \widehat{x}_0}{\widehat{r}} & c_n &= \frac{y_n - \widehat{y}_0}{\widehat{r}}\,. \end{aligned} \tag{2.49}$$

From the residuals

$$\widehat{v}_n = \widehat{r} - |\boldsymbol{x}_n - \widehat{\boldsymbol{x}}_0|\,, \tag{2.50}$$

we obtain the estimated variance factor

$$\widehat{\sigma_0}^2 = \frac{1}{N-3} \sum_n \frac{\widehat{v}_n}{\sigma_n^2}\,. \tag{2.51}$$

2.1.4.2 Circle Segments

In Section 3.2, we will propose an adaption of the Douglas-Peucker algorithm, which partitions an ordered sequence of points into a sequence of circle segments. In order to enforce continuity, we restrict the end-points of the segments to points of the given sequence. Therefor, this section describes the parametrisation and estimation of circle segments which are restricted to their endpoints. We concentrate on a geometric solution which we solve using an appropriate optimisation algorithm. As our line simplification mentioned above is meant as pre-segmentation, thus, as splitting the sequence of points into smaller sequences we are, at the moment, neither interested in an estimate in Least Squares sense nor in full covariance information.

Parametrisation of Circle Segments. A circle has three degrees of freedom, but by restricting the arc to two points there is only one degree of freedom left. We parametrise the arc segment by its height h which could be positive and negative.

We call the first and last point of an arc segment $\mathcal{X}_s$ and $\mathcal{X}_e$, respectively. This way, the chord $\mathcal{s}$ is oriented from $\mathcal{X}_s$ to $\mathcal{X}_e$. Everything left of $\mathcal{s}$ has positive sign, everything right of $\mathcal{s}$ has negative sign.

A circle segment $\mathcal{S}$ is then parametrised by

$$\mathcal{S} : \{\boldsymbol{x}_s, \boldsymbol{x}_e, h\} \tag{2.52}$$

whereby, we assume its start- and end-point fixed and the height h to be the single free parameter.

We define the orientation of the circle segment clockwise, if the points defining the segment are on the left of the chord, thus, $h > 0$ and counter clockwise otherwise, cf. Figure 2.2.

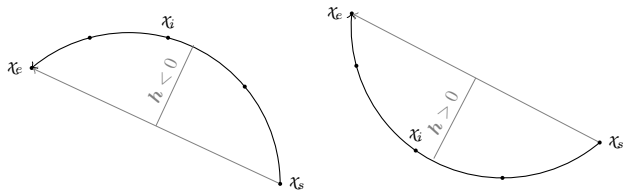
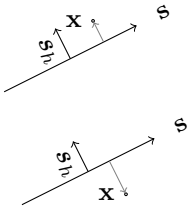


Figure 2.2: Orientation of circle segment. **Left:** counter clockwise. **Right:** clockwise.

To test the orientation, we need to determine the perpendicular distance of a point $\mathbf{x}$ to $\mathbf{s}$ which is given by

$$d(\mathbf{x}, \mathbf{s}) = \frac{\mathbf{x}^\mathsf{T}\mathbf{s}}{|x_h s_h|} = \begin{cases} > 0 & \text{if } \mathbf{x} \text{ left of } \mathbf{s} \\ < 0 & \text{if } \mathbf{x} \text{ right of } \mathbf{s} \end{cases} \tag{2.53}$$



Thereby, the chord $\boldsymbol{s}$ is given by

$$\mathbf{s} = \mathsf{N}^\mathsf{e}(\mathbf{x}_\mathsf{s} \times \mathbf{x}_\mathsf{e}) \,, \tag{2.54}$$

with euclidean normalisation $\mathsf{N}^\mathsf{e}(\cdot)$, and assuming the third coordinate of the two end points to be positive. This way, the normal $\boldsymbol{s}_h$ of $\mathbf{s}$ is directed to its left.

Estimating the Height of a Circle Segment. Given a set of points $\mathcal{X} = \{\mathcal{x}_i\}$, we aim at fitting an arc segment $\mathcal{S}$, such that the distances of points to the segment are minimised. Thereby, we determine the distances $\boldsymbol{d} = [d(\mathcal{x}_i, \mathcal{S})]$, of involved points $\mathcal{x}_i$ to the arc segment $\mathcal{S}$, and not to the whole circle. To be precise, the distance of a point to a segment is the minimum of the distance to the footpoint on the segment or the distance to the start or endpoint, which is visualised in Figure 2.3.

Finally, we obtain an estimate of height h by solving the following optimisation problem

$$\hat{h} = \text{argmin}_h \, |\boldsymbol{d}(\mathcal{X}, \mathcal{S}(\mathcal{x}_s, \mathcal{x}_e, h))|_L \,. \tag{2.55}$$

The choice of norm depends on the problem. When choosing $L = 2$ we obtain a Least Squares estimate. For a robust estimate we might choose the L_1-norm; thus, we optimise h such that the sum of absolute distances of all points to the arc segment is minimised. Using the L_∞-norm we optimise h such that

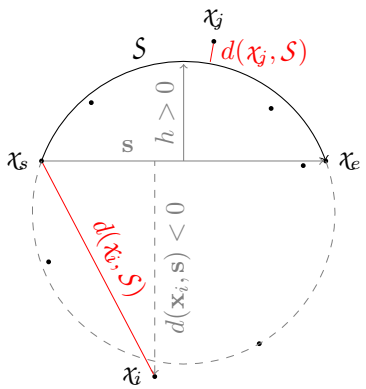


Figure 2.3: Distance measure of points $\mathcal{x}$ to a given arc $\mathcal{S}$. For points, not belonging to the arc, e.g., $\mathcal{x}_i$, its distance to the arc is given by its shortest distance to the arcs end points instead of shortest distance to the circle.

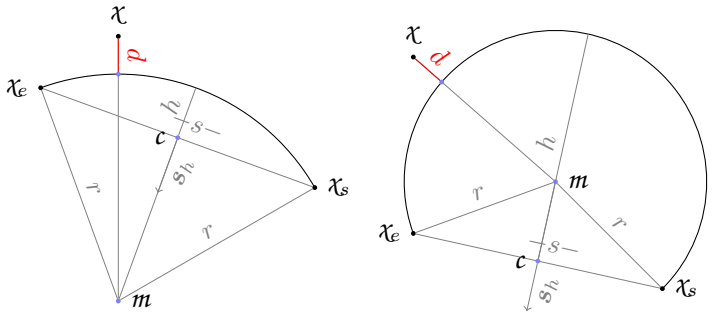


Figure 2.4: Geometric elements describing a circle segment. **Left:** special case, where the central angle is smaller than 180°, thus, $|h| < r$. **Right:** special case, where the central angle is greater than 180°, thus, $|h| > r$. The chord s is defined in the direction from χ_s to χ_e. Both cases show the circle segment right of the chord, thus, $h < 0$.

the maximum distances of any point to the arc segment is minimised. In our experiments, dealing with pixel-chains extracted from images, we got best results using the L_2-norm and hence, use this option for further evaluation.

Making the distance $d(\chi, \mathcal{S})$ explicit. To obtain an explicit equation of distance $d(\chi, \mathcal{S})$, we need to take into account two special cases of circle segments. We distinguish circle segments with central angle $< 180°$ or $\geq 180°$, cf. Figure 2.4. Depending on the arcs geometry, the distance of a point χ to $\mathcal{S}$ is given by

$$d(\chi, \mathcal{S}) = \begin{cases} |\boldsymbol{x} - \boldsymbol{m}| - r & \text{if } \{\text{sign}\,(d(\mathbf{x}, \mathbf{s})) = \text{sign}(h)\} \quad | \quad \chi \in \Delta\{\chi_s, m, \chi_e\} \\ \min(|\boldsymbol{x} - \boldsymbol{x}_s|\,, |\boldsymbol{x} - \boldsymbol{x}_e|) & \text{otherwise} \end{cases} . \tag{2.56}$$

Thereby, the centre $\boldsymbol{m}$ can then be calculated by

$$\boldsymbol{m} = \begin{cases} \boldsymbol{c} - (r - |h|)\,\text{sign}\,(h)\,\boldsymbol{s}_h & \text{if } r \geq |h| \quad \text{(cf. Figure 2.4 left)} \\ \boldsymbol{c} - (|h| - r)\,\text{sign}\,(h)\,\boldsymbol{s}_h & \text{otherwise} \quad \text{(cf. Figure 2.4 right)} \end{cases}, \tag{2.57}$$

using the chords midpoint

$$\boldsymbol{c} = \frac{\boldsymbol{x}_s + \boldsymbol{x}_e}{2}, \tag{2.58}$$

and radius

$$r = \frac{4h^2 + |\boldsymbol{x}_e - \boldsymbol{x}_s|^2}{8\,|h|} . \tag{2.59}$$

Remind our optimisation problem given in (2.55). We search for h which minimises $|\boldsymbol{d}(\mathcal{X}, \mathcal{S}(\chi_s, \chi_e, h))|_L$. Optimisation by gradient descent methods is not applicable, as the function, we wish to optimise, is non-convex and we can not give the derivative of $|\boldsymbol{d}(\mathcal{X}, \mathcal{S}(\chi_s, \chi_e, h))|_L$ with respect to h. The function we wish to minimise is one-dimensional and unconstrained. Assuming the initialisation to be accurate enough to ensure convergences to the global optimum, we choose golden section search (Kiefer, 1953) combined with a parabolic interpolation as method for optimisation (Press et al., 2007, chap.10). Golden section search determines the extremum of an one-dimensional function by sequentially dividing the search range into smaller values. More details are given in Appendix C.1. We initialise the upper and lower bound of the search space by the smallest and greatest distances $d(\chi_i, \mathcal{S})$, respectively.

In our experiments we needed on average 6 to 10 iterations for the optimisation of one circle segment.

2.1.5 Conditioning

Fitting geometric entities often lack on an unstable numeric configuration, due to the mixture of true euclidean coordinates, potentially of high range, and the homogeneous component. To avoid unstable solutions, we condition our data before estimating parameters of geometric entities. To be precise, we transform our data $\mathsf{X} = [\boldsymbol{x}_n], n = 1 \dots N$, such that all coordinates lie within $[-1, 1]$. To do so we use the transformation matrix

$$\mathsf{M} = \begin{bmatrix} 1 & 0 & -\overline{x} \\ 0 & 1 & -\overline{y} \\ 0 & 0 & 1/\lambda \end{bmatrix}, \tag{2.60}$$

using the centroid

$$\overline{\boldsymbol{x}} = \frac{1}{N} \sum_{n=1}^{N} \boldsymbol{x}_n, \tag{2.61}$$

and a scale λ which we usually derive from the biggest distance s_{max} of any point $\boldsymbol{x} \in \mathsf{X}$ to the origin, thus, $\lambda = 1/s_{max}$. Points and lines are conditioned by the simple transformations

$$\mathbf{x}' = \mathsf{M}\mathbf{x}, \tag{2.62}$$

$$\mathbf{l}' = \mathsf{M}^{-\mathsf{T}}\mathbf{l}, \tag{2.63}$$

and conics by

$$\mathbf{C}' = \mathsf{M}^{-\mathsf{T}}\mathsf{C}\mathsf{M}^{-1}. \tag{2.64}$$

The transformations (2.62) and (2.63) are linear in $\mathbf{x}$ and $\mathbf{l}$; thus, their variance propagation is trivial. The transformation for conics is bilinear. To obtain the Jacobian, we use the vec-operator, as introduced in (A.6), and rewrite (2.64)

$$\underset{9\times1}{\mathrm{vec}\mathsf{C}'} = (\underset{9\times9}{\mathsf{M}^{-\mathsf{T}} \otimes \mathsf{M}^{-\mathsf{T}}})\underset{9\times1}{\mathrm{vec}\mathsf{C}}, \tag{2.65}$$

using the identity (A.12). As C is symmetric we can use (A.13) and (A.14) to obtain

$$\underset{6\times6}{J_{c'}} = \frac{\mathrm{dvech}\mathsf{C}'}{\mathrm{dvech}\mathsf{C}} = \underset{6\times9}{\boldsymbol{S}_3}(\underset{9\times9}{\mathsf{M}^{\mathsf{T}} \otimes \mathsf{M}^{\mathsf{T}}})\underset{9\times6}{\boldsymbol{S}_3^{\mathsf{T}}}. \tag{2.66}$$

Finally, we obtain the transformed covariance of the conic by

$$\underset{6\times6}{\Sigma_{c'c'}} = \underset{6\times6}{J_{c'}}\ \underset{6\times6}{\Sigma_{cc}}\ \underset{6\times6}{J_{c'}^{\mathsf{T}}}. \tag{2.67}$$

Conditioning forth and back and the according variance propagation for points, lines, and conics is summarised in Table 2.1.

Table 2.1: Conditioning of points, lines and conicsusing the transformation M and according variance propagation.

	conditioning	backconditioning
Points	$\mathbf{x}' = \mathsf{M}\mathbf{x}$ $\underset{3\times3}{\Sigma_{\mathrm{x}'\mathrm{x}'}} = J_{\mathrm{x}'}\Sigma_{\mathrm{xx}}J_{\mathrm{x}'}^{\mathsf{T}}$ $\underset{3\times3}{J_{\mathrm{x}'}} = \mathsf{M}$	$\mathbf{x} = \mathsf{M}^{-1}\mathbf{x}'$ $\underset{3\times3}{\Sigma_{\mathrm{xx}}} = J_{\mathrm{x}}\Sigma_{\mathbf{x}i'\mathrm{x}'}J_{\mathrm{x}}^{\mathsf{T}}$ $\underset{3\times3}{J_{\mathrm{x}}} = \mathsf{M}^{-\mathsf{T}}$
Lines	$\mathbf{l}' = \mathsf{M}^{-\mathsf{T}}\mathbf{l}$ $\underset{3\times3}{\Sigma_{\mathrm{l}'\mathrm{l}'}} = J_{\mathrm{l}'}\Sigma_{\mathrm{ll}}J_{\mathrm{l}'}^{\mathsf{T}}$ $\underset{3\times3}{J_{\mathrm{l}'}} = \mathsf{M}^{-\mathsf{T}}$	$\mathbf{l} = \mathsf{M}^{\mathsf{T}}\mathbf{l}'$ $\underset{3\times3}{\Sigma_{\mathrm{ll}}} = J_{\mathrm{l}}\Sigma_{\mathrm{l}'\mathrm{l}'}J_{\mathrm{l}}^{\mathsf{T}}$ $\underset{3\times3}{J_{\mathrm{l}}} = \mathsf{M}^{\mathsf{T}}$
Conics	$\mathsf{C}' = \mathsf{M}^{-\mathsf{T}}\mathsf{C}\mathsf{M}^{-1}$ $\underset{6\times6}{\Sigma_{\mathrm{c}'\mathrm{c}'}} = J_{\mathrm{c}'}\Sigma_{\mathrm{cc}}J_{\mathrm{c}'}^{\mathsf{T}}$ $\underset{6\times6}{J_{\mathrm{c}'}} = \underset{6\times9}{\boldsymbol{S}_3}(\underset{9\times9}{\mathsf{M}^{-\mathsf{T}}\otimes\mathsf{M}^{-\mathsf{T}}})\underset{9\times6}{\boldsymbol{S}_3^{\mathsf{T}}}$	$\mathsf{C} = \mathsf{M}^{\mathsf{T}}\mathsf{C}'\mathsf{M}$ $\underset{6\times6}{\Sigma_{\mathrm{cc}}} = J_{\mathrm{c}}\Sigma_{\mathrm{c}'\mathrm{c}'}J_{\mathrm{c}}^{\mathsf{T}}$ $\underset{6\times6}{J_{\mathrm{c}}} = \underset{6\times9}{\boldsymbol{S}_3}(\underset{9\times9}{\mathsf{M}^{\mathsf{T}}\otimes\mathsf{M}^{\mathsf{T}}})\underset{9\times6}{\boldsymbol{S}_3^{\mathsf{T}}}$

2.2 Statistical Background, Terms and Notation

As already pointed out in Section 1.4.2, we will need a representation of a probability density over a space of configurations of geometric objects, which leads to statistical concepts different from those commonly used in geodetic applications. This section is desired to introduce our notation and presents the basic terms from statistics we use throughout the following sections.

2.2.1 Measure Theory

2.2.1.1 Measure

A measure μ on a set $\mathcal{S}$ is a mapping which assigns a non-negative real number to each subset of that set, which we may interpret as its size, the number of elements in the subset or its probability. A measure μ must satisfy the following conditions

- $\mu(\mathcal{S}_1 \cup \mathcal{S}_2) = \mu(\mathcal{S}_1) + \mu(\mathcal{S}_2)$, provided $\mathcal{S}_1$ and $\mathcal{S}_2$ are disjoint sets $\mathcal{S}_1 \cap \mathcal{S}_2 = \emptyset$
- $\mu(\emptyset) = 0$
- μ is continuous: Given $\mathcal{S}_n$ is a decreasing sequence of closed, bounded sets ($\mathcal{S}_n \supseteq \mathcal{S}$), with limit $\cap_n \mathcal{S}_n = \mathcal{S}$, then $\mu(\mathcal{S}_n) \to \mu(\mathcal{S})$

A probability measure additionally fulfils the requirement $\mu(\mathcal{S}) = 1$.

2.2.1.2 Measure Space

A measure space is a triple $(S, \mathcal{S}, \mu)$ where S is a set, $\mathcal{S}$ a σ-algebra of its subsets, and μ a measure. We denote $(S, \mathcal{S})$ the measurable space and μ the measure on $\mathcal{S}$. Thus, a measure space consists of a measurable space and a measure.

We call a measure space whose measure is a probability measure a probability space.

2.2.1.3 Measurable Function

We also use the term measurable function which is a mapping between measurable spaces. Suppose two measurable spaces $(S, \mathcal{S})$ and $(T, \mathcal{T})$, wherein $\mathcal{S}$ and $\mathcal{T}$ are the σ-algebras according to S and T. A function f is said to be measurable if $f^{-1}(A) \in \mathcal{S} \quad \forall A \in \mathcal{T}$. Thus, the inverse image of any subset of T is a subset of S.

2.2.2 Random Variables

We denote random variables underlined, e.g., $\underline{x}$ and the probability of the event that the random variable $\underline{x}$ takes the value x by $\mathbb{P}(\underline{x} = x)$, or short $\mathbb{P}_x(x)$ with subscript x of $\mathbb{P}$, to denote the random variable or even shorter $\mathbb{P}(x)$, if the name of the random variable is clear from context.

The result of an experiment, being the set of outcomes, does not have to be a real valued number. It could be the colour of one or more balls, picked from urns, or the image we see when tossing a coin. In our applications it will be a set of objects, i.e. rectangles within the image plane. Therefor, we need a formal way to declare random variables to describe an experiment.

A random variable $\underline{x}$ is a mapping which allows us to assign a number $\underline{x}(\omega)$ to the outcome ω of an experiment, being an element of a sample space Ω. Thus, the domain of $\underline{x}$ is the set of possible outcomes of the experiment, e.g., $\Omega = \{$'head','tail'$\}$ for a coin experiment. The range of $\underline{x}$ are the numbers assigned to each outcome, e.g., $\underline{x}$('head') $= 0$ and $\underline{x}$('tail') $= 1$.

We usually define a random variable $\underline{x}$ in terms of a probability space $(\Omega, \mathcal{A}, \mathbb{P})$ with $\mathcal{A}$, denoting a σ-algebra on Ω, defining the set of events defined for a certain experiment and a mapping $\mathbb{P} : \mathcal{A} \to [0, 1]$ which assigns a probability to each event, following Kolmogorovs axioms.

Therefrom, we may define a random variable $\underline{x}$ with range $\mathbb{R}$ more formally as a measurable function

$$x : (\Omega, \mathcal{A}) \rightarrow (\mathbb{R}, \mathcal{B}) \tag{2.68}$$

$$\omega \rightarrow x(\omega) \tag{2.69}$$

from a probability space $(\Omega, \mathcal{A}, \mathbb{P})$ to a measurable state space $(\mathbb{R}, \mathcal{B})$. We also use the following notation: If A is a subset of $\mathbb{R}$ we denote by $\{x \in A\}$ the event

$$\{x \in A\} = \{\omega \in \Omega \ : \ x(\omega) \in A\} \ . \tag{2.70}$$

The measure on $\mathcal{B}$ defined by

$$\mu(A) = \mathbb{P}\left(x^{-1}(A)\right) \tag{2.71}$$

is the distribution of $\underline{x}$.

Sometimes, we relax the dependency on ω and denote by $\mathbb{P}(\underline{x} = x)$ the probability of the event that the random variable $\underline{x}$ takes the value x. Then, we use the term probability measure synonymous to the term distribution. Thus, the distribution of a random variable $\underline{x}$ refers to the operator $\mathbb{P}(x)$ which is a probability measure. Usually the distribution is given either in terms of the distribution *function* or probability density function (probability mass function for discrete random variables) for which we usually use the definitions discussed next.

2.2.2.1 Probability Distribution Function.

We denote the (cumulative) probability distribution function (cdf) of a random variable $\underline{x}$ by

$$P_x(a) = \mathbb{P}(\underline{x} \leq a) \qquad a \in \mathbb{R} \tag{2.72}$$

whereby, the index x indicates the name of random variable $\underline{x}$. Vice versa, the probability measure for a small interval $(a, b]$ is given by

$$\mathbb{P}(a, b] = P(b) - P(a) \qquad a, b \in \mathbb{R}, a < b \tag{2.73}$$

from which we see that P must satisfy

$$\lim_{x \to -\infty} P(x) = 0 \qquad \lim_{x \to \infty} P(x) = 1 \ . \tag{2.74}$$

For a discrete-type random variable P, is piecewise constant, thus, has a finite number of jump discontinuities.

2.2.2.2 Probability density and probability mass function.

For continuous random variables, the probability density function (pdf) is a function $p : x \rightarrow \mathbb{R}^+$ satisfying

$$\begin{gathered} p(x) > 0 \quad \forall x \in \Omega \\ \int_\Omega p(x) \, \mathrm{d}x = 1 \\ \mathbb{P}(x \in \mathcal{B}) = \int_{\mathcal{B}} p(x) \, \mathrm{d}x \quad \mathcal{B} \subseteq \Omega \ . \end{gathered} \tag{2.75}$$

We say the distribution of $\underline{x}$ admits a density $p(x)$, which means that the distribution of $\underline{x}$ is absolutely continuous, but not implies that the density is continuous.[2]

For a discrete random variable, the pdf is a collection of positive discrete masses and is called probability mass function (pmf). It directly gives rise to the probability of a certain event $p(x) = \mathbb{P}(\underline{x} = x)$ and the conditions above become

$$\begin{gathered} \textstyle\sum_{x \in \Omega} \ p(x) = 1 \\ \mathbb{P}(x \in \mathcal{B}) = \textstyle\sum_{x \in \mathcal{B}} \ p(x) \quad \mathcal{B} \subseteq \Omega \ . \end{gathered} \tag{2.76}$$

The distribution of a discrete random variable cannot admit a density.

[2] A differentiable function is continuous; thus, as $\mathbb{P}$ is the derivative of p it must be continuous. The other way around must not hold.

2.2.3 The Radon-Nikodym Derivative

Given a measurable space $(\Omega, \mathcal{A})$, on which there is a measure ν which is absolutely continuous with respect to *another* measure μ on $(\Omega, \mathcal{A})$. Then, there is a measurable function $f : \Omega \to [0, \infty]$ which transforms the measure μ into the measure ν, namely

$$\nu(A) = \int_A f \, \mathrm{d}\mu \quad A \subset \Omega \, . \tag{2.77}$$

Thus, the function f is given by

$$f = \frac{\mathrm{d}\nu}{\mathrm{d}\mu} \, , \tag{2.78}$$

what is called the Radon-Nikodym derivative. If we assume the measure ν to be a probability measure, then the function f is the density admitted by ν and μ is a measure of volume for which we evaluate the density.

In our context this reads as follows. Given a distribution $\mathbb{P}(x)$ on Ω and assuming Ω to be a subset of $\mathbb{R}^d$. Further, assume $\mathbb{P}(x)$ to admit a density with respect to a measure μ. Then we write

$$\mathbb{P}(\mathrm{d}x) = p(x)\mu(\mathrm{d}x) \, . \tag{2.79}$$

Usually, we assume μ to be the Lebesgue measure, thus, the measure of volume in $\mathbb{R}^d$. We say $\mathbb{P}$ is dominated by μ. Then, the density function $p(x)$ is given by the Radon Nikodym derivative

$$p(x) = \frac{\mathbb{P}(\mathrm{d}x)}{\mu(\mathrm{d}x)} \tag{2.80}$$

of the probability measure with respect to the measure of volume.

Observe, if $\mu(\mathrm{d}x) = \mathrm{d}x$ is the unit function and using the abbreviation $\mathbb{P}(\mathrm{d}x) = \mathrm{d}P(x)$, we arrive at the classical definition of the density

$$p(x) = \frac{\mathrm{d}P(x)}{\mathrm{d}x} \, . \tag{2.81}$$

In fact (2.80) holds for any type of measure as stated in (2.78). We will use these definitions, in Section 2.5, for the construction of the density of a marked point process with respect to the probability measure of a dominating Poisson process.

2.3 Model Selection

In Section 3.3, we will deal with the task of merging neighboured segments of pixel-chains, in case they share the same geometric model. Thus, we need to decide whether two pixel-chains are best described by a joint straight line, or an ellipse segment, or by two separate line segments.

Model selection deals with the task of finding a model which best explains the observed data. Thus, the term explaining the data in our context is equivalent to describing a set of points by choosing an appropriate geometric model. The domain of models we use is {straight line, circle, ellipse}, which differ in the number of parameters. The term accuracy then, is related to the residuals caused by deviations of the points to the selected model.

For the task of model selection, we find different points of view in the literature. First, we observe a different notion of *the best* model. This can be seen by explaining the data in the most accurate way or by choosing a model which obtains the best predictive quality. The former leads to a maximum likelihood choice, which tend to choose the most complex model. At this point we refer Occam's razor, which states that the best explanation is as simple as possible, but not simpler. For the topic of model selection, this means not to use more parameters as needed to explain the data. Preferring simple models result in less calculations and less memory.

From the point of view of prediction, we could require the model nothing, but to work. Thus, model selection should be based on the quality of prediction.

At the end, we realise the trade-off between goodness-of-fit and simplicity. Both are contradictory, maximising simplicity degrades goodness-of-fit. But, on the other hand, the perfect fit requires a more complex model.

In 1978 Schwarz derived the widely used *(Schwarz's) Bayesian Information Criterion (BIC)* for model selection. Let us considering a number of N normally, i.i.d. distributed observations $\boldsymbol{l}$ with covariance Σ_{ll}, which is reasonable for our applications. We are looking for an U-dimensional parameter vector $\widehat{\boldsymbol{\theta}}$, whereby observations and parameters are related by a functional model $\boldsymbol{l} + \widehat{\boldsymbol{v}} = f\left(\widehat{\boldsymbol{\theta}}\right)$. Using the usual definition $\Omega = \widehat{\boldsymbol{v}}^{\mathsf{T}} \Sigma_{ll}^{-1} \widehat{\boldsymbol{v}}$, Schwarz derived

$$\mathrm{BIC} = \Omega + U \ln N \ , \tag{2.82}$$

as a criterion for model selection, which we minimise in order to identify the best model. It is derived from the so called model evidence, which in fact is the marginal distribution of observations. It includes a measure of fit of observations $\boldsymbol{l}$ concerning parameters $\boldsymbol{\theta}$ in terms of the likelihood function $p(\boldsymbol{l} \mid \boldsymbol{\theta})$ and some prior knowledge, of how the model should be, as penalty term. The lower the complexity of the model, given by the number of parameters U, the lower BIC. A higher number N of observations increases the relative precision of the parameters, thus, increases the reliability of the model.

There are several other methods for model selection. (Rissanen, 1978, 1983) proposed a measure for description length. The term description length refers to a measure in terms of coding length, which states how many bits we need to optimally describe the data, assuming to compactly describe them by an appropriate model. To select the best model, we aim at minimising the description length. However, it turns out that in case of normally distributed observations, it is equivalent to BIC (McGlone, 2004).

Beside BIC, one of the most used model selection methods is the one given by Akaike (1974). He proposed *An Information Criterion*[3] (AIC), which chooses the model $\boldsymbol{\theta}$ that minimises

$$\mathrm{AIC} = \Omega + 2U \ , \tag{2.83}$$

again in case of normally distributed observations.

Both, BIC and AIC, are related to coding length due to the use of information in the derivation of their criteria. Thus, when comparing two models, based on BIC or AIC, we evaluate the gain of description length.

To this end, we decided to use BIC as criterion for model selection throughout our experiments. This is due to the facts that indeed BIC is the most established criterion and we have no arguments against. Further, we actually observed no significant differences between these criteria.

2.4 Bag of Words for Image Representation

As introduced in Section 1.4.2, we will perform object classification and detection, respectively, at mid-level of our facade image interpretation system. In order to learn from data a representation that is suited for that task, we will use the so called bag of words (BoW) model for the classification of image sections. It will be used in Chapter 4 as well as in Chapter 5.

The idea of BoW is to treat image features as words and to represent an image by a collection of such words, cf. Figure 2.5. Having defined a dictionary of meaningful words, we conclude about the content of an image. According to the visualisation, the collection of meaningful words is called a bag of words. Synonymously we use the term codebook.

[3]The criterion was called *Infromation Criterion* (IC). In order to provide a list of similar proposals, Akaike (1974) added *A*, trusting *B, C* etc. would follow. And he was right.

The BoW model has its origin in information retrieval and text processing. It was first used in the work of Harris (1954), who states that, *"The degree of semantic similarity between two linguistic expressions A and B is a function of the similarity of the linguistic contexts, in which A and B can appear."* (Harris, 1954)

In other words, the meaning of a word depends on the context, in which it is used, thus, the whole sentence, or to be more precise, on the co-occurrences of certain words. This allows us to classify documents by topic, by analysing the occurrence of certain words.

In image classification or image retrieval this idea is adopted to the use of image features. Terminology used in information retrieval is easily transferred to our purposes. A document is either a text or an image. The former is a collection of terms or words, whereas the latter can be seen as a collection of image features which we interpret as words again. We use document and image synonymous. In both cases, we count the occurrences of words from a dictionary to conclude about the category of a document or an image, respectively. Thereby, the exact ordering of words within the document or the position of features within an image, respectively, is ignored, only their number is crucial.

In image processing BoW arises around the turn of the millennium in works about texture recognition (Leung and Malik, 2001; Cula and Dana, 2001; Lazebnik et al., 2003). Later it becomes popular in object categorisation and detection, respectively (Fergus et al., 2003; Csurka et al., 2004; Fei-Fei et al., 2004; Sivic et al., 2005; Sivic and Zisserman, 2006).

The principle of using BoW for image classification is shown in Figure 2.6. Initially, we will sketch this principle roughly in an abstract way, in order to give the intuition of the main idea. Later, we will go into detail of individual parts of the process.

Given a set of N training images and their according class labels, we extract image features. Due to the diversity of given images, even within classes, we get a diverse set of extracted image features. But, due to within class similarity, they are still similar features, thus, features close to each other in feature space. Clustering leads to such groups of similar images features. These clusters are supposed to represent meaningful image features which are suited to be part of a compact image description. The codebook is composed of representatives for each cluster. Finally, each image from the training set is represented by the histogram of codewords. For that, each feature in the image, which can be mapped to a codeword from the codebook, increases the according bin of the histogram. At the end, each image is compactly described by a single histogram vector $\boldsymbol{x}_n,\ n = 1 \dots N$. Its length is equal to the size of the codebook. The elements of this histogram vector are the counts of codewords within the image. Collecting all training images within matrix $\boldsymbol{X} = [\boldsymbol{x}_n]$ and their according class labels within vector $\boldsymbol{y}$ we train a classifier $\boldsymbol{y} = f(\boldsymbol{X})$.

For testing an unknown image, we extract features again and map them to the codebook. We build the histogram of words and determine the class label using the learned classifier f.

Figure 2.5: The idea behind bag of words for image interpretation: An image is represented as unordered collection of independent images features (Fei-Fei, 2005). The figure is probably the most cited image related to BoW.

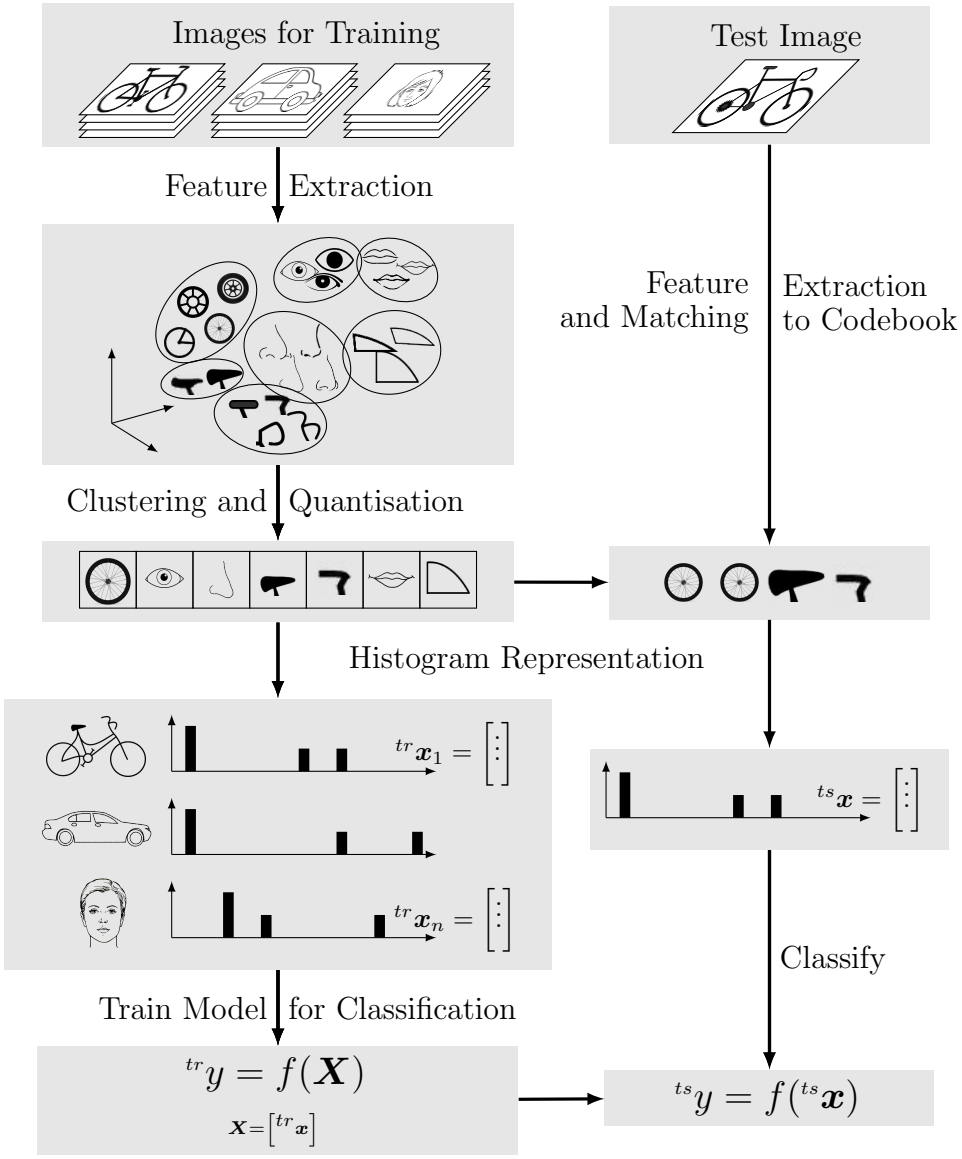


Figure 2.6: The principle of using BoW for image classification.

To get more into detail, we need to specify three parts of the method. First, we need to define appropriate image features and their description which lead to the definition of a feature space. Second, we need to choose a method to select those features which are suited to describe the images in a compact way. This leads to a definition of similarity of two features, to the choice of a clustering method, and finally, to the definition of a best representative for each cluster. Third, we need to choose a classifier. Thereby, the most influential decision in designing a BoW model is the choice of the adequate features. In the following, we will review commonly used features, and methods for codebook generation. Later, in Sections 4.1 and 4.2, we will discuss our features for the classification of facade objects. Further, this section introduces methods for normalising the histogram of words, and introduces classification methods, we will use throughout our experiments.

2.4.1 Features Used as Words

There have been used several methods for feature extraction in BoW. It may be performed by determining features on a grid or of randomly positioned patches, or by feature detection.

Fei-Fei and Perona (2005) divide the image into a regular grid and take each cell as one image patch. Csurka et al. (2004), Fei-Fei et al. (2004), Fei-Fei and Perona (2005), Sivic and Zisserman (2006) and many others use interest point detectors to collect image features, e.g., the Kadir and Brady (Kadir and Brady, 2001), Harris (Harris and Stephens, 1988), or SIFT detector (Lowe, 2004). The work of Mikolajczyk and Schmid (2005) gives an extensive review of existing feature detectors, their properties and evaluates their performance. Another idea was to randomly sample position and scale of image regions or to use image patches, given by image segmentation (Fei-Fei, 2005).

In each case, image patches serve as basic input for building the BoW model, for which we need a descriptor. One way is to described images patches by pure pixel information; thus, pixel intensities of each pixel of the patch are collected within a large vector. Possibly, the dimension of feature space is reduced by principal component analysis (PCA) (Fei-Fei et al., 2004). In case interest points where extracted, their local neighbourhood is described by an appropriate descriptor. The SIFT descriptor, developed by Lowe (2004), was shown to outperform other descriptors in the literature by Mikolajczyk and Schmid (2005). It is used in the context of BoW, e.g., in Csurka et al. (2004), Fei-Fei and Perona (2005), Nister and Stewenius (2006), Sivic and Zisserman (2006) and many others. Again, this yield one vector per image patch. In case of SIFT descriptors, they are typically of dimension 128, but not necessarily.

The main criticism on BoW is its lack of information about spatial context of selected words. Therefore, the implicit shape model (ISM) (Leibe et al., 2004), similar to BoW, uses the principle of codewords, but additionally adds information about the spatial constellation of several parts of the object model. Thereby, it builds object specific models, while in BoW a single codebook may severe as dictionary for all classes.

Inspired from Sali and Ullman (1999), who use contour fragments for object classification, Opelt and Zisserman (2006) and similarly Shotton et al. (2005) introduce this idea for object detection, and use contour fragments as codewords in an ISM (Leibe et al., 2004). Nevertheless, Ferrari et al. (2008) pick up the idea of using contour fragments as words in a BoW model. They introduce kAS-features, standing for groups of k-adjacent line segments, as basic image features which they use in a BoW model for the task of object detection. We will pick up exactly this idea when introducing our pairs of adjacent line segments in Chapter 4. We will overcome the limitation of common BoW, by incorporating a rough spatial layout by splitting the image patch into several tiles, from which the overall image descriptor is given by the concatenated histogram of words of all tiles.

2.4.2 Generating the Codebook

After having collected a large enough number of image features from training images, the goal is to generate a codebook of meaningful words out if. In most cases this is done by clustering the feature descriptors in feature space, e.g., by using k-means or mean-shift (Comaniciu and Meer, 2002). The codewords, thus, the representatives for each cluster, usually are chosen as mean value out of all cluster

members. Both cluster methods assume a distance, defined on the feature space, which is easy and fast evaluated. Usually, this is assumed either to be euclidean or a cosine distance in case of SIFT descriptors. Using k-means requires the number of clusters, which is unknown beforehand, differs between different datasets, and involves the danger of over- or under-segmentation. However, there are methods to determine the best number of clusters (Bishop, 2006, Ch. 9). Mean-shift became popular because it estimates the number of clusters from data. In return, it assumes the kernel-width as input parameter, which can be interpreted as expected distance of features to except them as similar.

For the case of more complex feature spaces which prevent the use of euclidean or cosine distances, as we will seen in Section 4.2.2, Ferrari et al. (2008) propose to use clique-partitioning for clustering.

Clique-partitioning. Assume a complete graph $\mathcal{G}(\mathcal{V}, \mathcal{E})$, with vertices $v \in \mathcal{V}$ and weighted edges $e \in \mathcal{E}$. The goal of clique-partitioning is to partition $\mathcal{G}$ into completely connected, disjoint subsets of vertices (cliques), such that the sum of remaining edge weights is maximised.

Let w_{ij} be the weight on the edge connecting vertices v_i and v_j. Further, let the unknown $e_{ij} \in \{0, 1\}$ indicating whether an edge is active or not. We formulate the clique partitioning problem as linear program

$$\begin{aligned}
\text{maximise} \quad & \sum_{1 \le i \le j \le n} w_{ij} e_{ij} \\
\text{subject to} \quad & e_{ij} + e_{jk} - e_{ik} \le 1, \qquad \forall 1 \le i < j < k \le n \\
& e_{ij} - e_{jk} + e_{ik} \le 1, \qquad \forall 1 \le i < j < k \le n \\
& -e_{ij} + e_{jk} + e_{ik} \le 1, \qquad \forall 1 \le i < j < k \le n \\
& e_{ij} \in \{0, 1\}, \qquad \forall 1 \le i < j < k \le n \,.
\end{aligned} \tag{2.84}$$

The objective function maximises the sum of intra-clique weights, while the constraints express the clique constraint, thus, ensure the full connectivity of each cluster. Equation (2.84) can be solved by linear programming, e.g., the simplex algorithm. But, as already mentioned by Ferrari et al. (2001), clique-partitioning is NP-hard with exponential complexity in the number of vertices n, in worst case. Therefore, Ferrari et al. (2001) propose a greedy procedure with polynomial complexity, $O(n^3)$ in worst case, but faster in practice.

Starting with a partitioning $\Phi = \{i\}_{1 \dots n}$ of $\mathcal{G}$, where each vertex is a singleton clique, they iteratively merge cliques which fulfil the following requirements:

1. cliques proposed for merging are their mutual best merging option
2. merging increases the total score.

To evaluate these requirements Ferrari et al. (2001) propose a cost function

$$m(c_1, c_2) = \sum_{i \in c_1, j \in c_2} w_{ij} \tag{2.85}$$

for merging cliques c_1 and c_2. The score of the best merging choice for clique c is given by function

$$b(c) = \max_{t \in \Phi} m(c, t) \tag{2.86}$$

and the name of the clique to merge with

$$d(c) = \operatorname*{argmax}_{t \in \Phi} m(c, t) \,. \tag{2.87}$$

Two cliques c_i and c_j are merged if and only if $d(c_i) = c_j$ and $d(c_j) = c_i$, which represent requirement 1., and if $b(c_i) = b(c_j) > 0$, which represents requirement 2. Merging is iteratively repeated until no two cliques fulfil the requirements anymore.

Clique-partitioning naturally deals with positive and negative weights, this leads to the definition of a best solution without knowing the number of cliques before and without introducing a stopping criterion (Ferrari et al., 2001).

Ferrari et al. (2008) use this procedure to solve the clustering. From extracted features they build a complete graph $\mathcal{G}(\mathcal{V}, \mathcal{E})$, where each vertex $v \in \mathcal{V}$ represents one feature in feature space, and edges $e \in \mathcal{E}$ are weighted by an appropriate dissimilarity measure. Thereby, the weights are defined by $w_{ij} = \tau - D(v_i, v_j)$ where D is a dissimilarity measure on features i and j and τ is a threshold which represents the excepted intra-cluster dissimilarity, similar to the mean-shift kernel width. The smaller D the similar two features. Thus, by introducing τ and taking the negative of D, similar features get heigh weights, while dissimilar features get negative weights, which again lead to the linear program (2.84).

For building the codebook, one representative per cluster is chosen: the feature with lowest distance to all other features of the current cluster, thus, the one closest to the cluster centre.

2.4.3 Normalising the BoW histogram

Using the BoW model, each image or image section is compactly represented by its histogram of codewords. To be independent on the images size, the histogram should by properly normalised. Otherwise, the number of counted words grows by the size of the image. Different normalising methods are commonly used. Most intuitively is a normalising, such that the length of the histogram vector is 1, thus, interpreting the normalized histogram as set of probabilities.

Another idea, again based on information retrieval in text processing, takes into account that certain words have little or even no discriminative power to determine the context of the document. For example, imaging a collection of documents about architecture. The word house occurs in almost all documents and does not help to conclude about the specific topic of the document. In our context, we are interested in object detection in images of facades. Here rectangular objects occur in almost all images and have little discriminative power to distinguish windows from doors or balconies.

tf-idf weighting is a weighting scheme that reduces the impact of words that occur too often to be meaningful. Given the number of occurrence n_{td} of word t within document d the according bin in the histogram, describing document d, is given by

$$h_d(t) = \frac{n_{td}}{n_d} \mathrm{lb} \frac{N}{N_t}, \tag{2.88}$$

wherein n_d is the total number of words in d, N_t is the number of documents containing at least one instances of t, and N is the total number of documents (Manning et al., 2009).

The first fraction in (2.88) is equal to the term frequency and counts the number of occurrences of word t in d, related to the total number of words within d, which we may interpret as $p(t \mid d)$. The second term in (2.88) is equal to the inverse document frequency and scales down the words with high frequency. It takes into account the number of documents containing a certain word, related to the total number of documents. We may interpret this term as $-\mathrm{lb} p(t)$, thus, the amount of information we get, when observing a word t. It is high if word t is rare, but low if t is more likely. Together with $p(t \mid d)$, we see that (2.88) turns out to be an entropy.

In our context, the tf-idf histogram bin of word t is highest when t appears many times within a small number of images; thus, it has a high discriminative power on this class of images. It is lower when t appears in many images or is rare within an image. The lowest tf-idf value is given when t occurs in almost all images.

2.4.4 Classification Methods

Having learned a codebook, we are able to represent each image or image section, respectively, in a compact way by its histogram of words which we will use as feature vector for classification. We realize object categorization by multi-class classification based on a feature vector.

Before introducing the two classification methods, we will use throughout this thesis, we shortly review commonly used methods in the context of BoW classification. Most approaches use generative classifiers that typically yield a posteriori probabilities which allow for further evaluation and processing

of classification results. On the other, hand discriminative methods have shown better discriminative power.

One of the most simplest classifier that is used in that context is the Naive Bayes classifier (Csurka et al., 2004). It relaxes the general Bayes theorem by assuming independent features. As it is derived from the general Bayes theorem, it is a generative model which yields a posteriori probabilities of assigned labels and it takes into account prior class probabilities. As stated above, the output of a posteriori probabilities is advantageous for further processing of classification results. But, as independence is a rather naive assumption, we will not take into account this model.

Another group of generative models, used in terms of BoW, are hierarchical Bayesian models (Sivic et al., 2005; Fei-Fei and Perona, 2005). They extensively exploit the power of Bayes theorem by not only modelling the full joint probability of features, but additionally introducing hyper priors, namely distribution over parameters of the prior distributions.

The probably most widespread discriminative classifier, not only, but also used in the context of BoW, is the support vector machine (SVM).

In the following, we will shortly present two discriminative classifiers, SVM and import vector machine (IVM). In Chapter 4, we will use SVM as powerful and fast classifier to evaluate the performance of proposed shapelets for the task of object categorisation. In Chapter 5, we will need the probabilistic output of a classifier, therefrom, we stick on IVM as a discriminative classifier which also yield a posteriori probabilities.

2.4.4.1 Support Vector Machines

Given features $\boldsymbol{X} = [\boldsymbol{x}_n]$ and labels $\boldsymbol{y} = [y_n] \in [-1, 1]$ $n = 1 \dots N$, a SVM, in its simplest form, solves the two class linear classification problem

$$y_n = y(\boldsymbol{x}_n) = \text{sign}\left(\boldsymbol{w}^\mathsf{T}\boldsymbol{\phi}(\boldsymbol{x}_n) + b\right) \ , \tag{2.89}$$

whereby, $\boldsymbol{w}$ and b are the parameters of a hyperplane that perfectly splits the data, assuming they are separable. The functions $\boldsymbol{\phi}$ represents a feature space transformation which allows us to separate the data even if they are not linearly separable. The parameters of the hyperplane are learned using training data. For classifying new unseen data points, we evaluate (2.89) to obtain the according label.

Converting (2.89) into another representation

$$y_n = \text{sign}\left(\sum_m y_m a_m k(\boldsymbol{x}_n, \boldsymbol{x}_m) + b\right) \ , \tag{2.90}$$

which we call the dual representation, avoids to work in the transformed feature space, represented by $\boldsymbol{\phi}$. The new parameters $\boldsymbol{a}$ are Lagrangian multipliers and result from the formulation of the constrained optimisation problem. We call

$$k(\boldsymbol{x}_n, \boldsymbol{x}_m) = \boldsymbol{\phi}(\boldsymbol{x_n})^\mathsf{T}\boldsymbol{\phi}(\boldsymbol{x_m}) \ . \tag{2.91}$$

the kernel function. It is the scalar product of transformed feature vectors and can be evaluated efficiently without having functions $\boldsymbol{\phi}$ explicitly. Furthermore, it allows us to work in high, even infinite high feature dimensions, which makes the model attractive for hardly separable data. The Gram matrix $\boldsymbol{K}$ which is the collection of kernels $k(\boldsymbol{x}_n, \boldsymbol{x}_m)$, is symmetric and positive semidefinite and expresses the similarity between every two features $\boldsymbol{x}_n$ and $\boldsymbol{x}_m$. Depending on the application, we choose an appropriate kernel function which is suited to express the similarity between features. Examples of widely used kernel functions are

- Linear kernel
$$k(\boldsymbol{x}_n, \boldsymbol{x}_m) = \boldsymbol{x}_n^\mathsf{T}\boldsymbol{x}_m \tag{2.92}$$
- Polynomial kernel
$$k(\boldsymbol{x}_n, \boldsymbol{x}_m) = \left(\gamma\ \boldsymbol{x}_n^\mathsf{T}\boldsymbol{x}_m + r\right)^M \ , \tag{2.93}$$
which contains all monomials up to order M

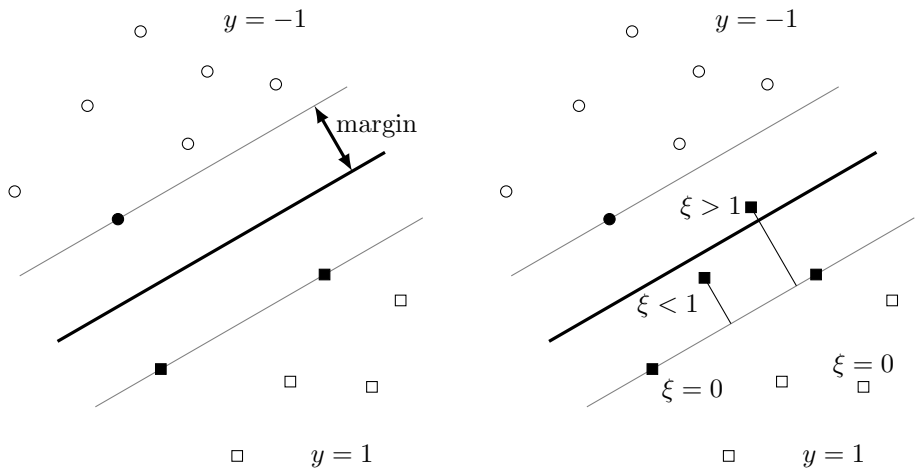


Figure 2.7: Principle of a two-class SVM. **Left:** The SVM tries to maximise the margin which is defined by the support vectors sitting on the margin. Support vectors are marked by filled circles and squares. **Right:** Slack variables. Training samples sitting on the margin or behind have $\xi_i = 0$; sitting on the wrong side of the marginthey have a value of $\xi_i > 1$, while on the correct side, but within the margin $1 \geq \xi_i \geq 0$.

- Gaussian or radial basis function (RBF) kernel

$$k(\boldsymbol{x}_n, \boldsymbol{x}_m) = \exp\left(\frac{-|\boldsymbol{x}_n - \boldsymbol{x}_m|^2}{2\sigma^2_{\text{RBF}}}\right) . \tag{2.94}$$

Depending on chosen kernel we have kernel parameters $\gamma > 0, \sigma_{\text{RBF}} > 0, r, M$.

Assuming the data to be linearly separable, possibly in their transformed feature space, there is no unique solution for a linear hyperplane that perfectly splits the data. For this, SVM introduces a margin which is defined by the smallest distance between any training sample and the decision boundary. In conclusion, the goal of the SVM is to determine the hyperplane that maximises the margin, see Figure 2.7 left.

It turns out that only a couple of Lagrangian multipliers a_n are different from zero. Thus, only their according training samples contribute to the solution. These special samples are called support vectors and it can be shown that they lie on the margin.

Different from the argumentation before, the data may not be perfectly separable. So called slack variables ξ_n allow for overlapping class distributions, thus, the training samples to deviate from the margin or even the decision boundary. To avoid over-fitting, a regularisation parameter[4] λ_{svm} controls the trade off between model complexity and accuracy. In that case, the margin still is defined by the support vectors, but they are not supposed to sit on the margin any more, see Figure 2.7 right.

All together, the standard SVM model, in its primal form, requires the solution of the optimisation problem

$$\min_{\boldsymbol{w},b,\boldsymbol{\xi}} \quad \frac{1}{2}|\boldsymbol{w}|^2 + \lambda_{svm} \sum_n \xi_n \tag{2.95}$$

$$\text{subject to} \quad y_n \left(\boldsymbol{w}^\mathsf{T} \boldsymbol{\phi}(\boldsymbol{x}_n) + b\right) \geq 1 - \xi_n \qquad \forall_n = 1 \ldots N \tag{2.96}$$

$$\xi_n \geq 0 \qquad \forall_n = 1 \ldots N \ . \tag{2.97}$$

Thus, we minimise the squared sum of weights and the sum of slack variables constrained to the correct sidedness of training samples. It is solved in its dual form, thus, by introducing a kernel function as

[4] In standard SVM literature this parameter usually is called C. But, to avoid confusion with the number of classes C, which we later use, we prefer the common regulariser name λ here.

before.

We do not go more into detail. SVM became one of the standard tools to solve regression and classification problems and is discussed in many textbooks, we refer on, e.g., Bishop (2006, Ch. 7), Vapnik (1995, Ch. 5), Schölkopf and Smola (2002). Furthermore, there exist a couple of software packages, which may use for solving our tasks, e.g., SVM^{light} by Joachims (1998) or LIBSVM by Chang and Lin (2011). Even Matlab provides SVM for classification as well as regression within the statistics toolbox.

We keep in mind that we are able to solve the classification task for complex decision boundaries by introducing an appropriate kernel function. Furthermore, we keep in mind that we need to define the regularisation parameter λ_{svm} and if needed the parameters of the kernel function, e.g., the kernel width σ_{RBF} of a RBF kernel. Either we estimate them from prior knowledge or we determine them by grid search.

The big advantages of using SVMs are, first, that the determination of the model parameters correspond to a convex optimisation problem, which guaranties a global optimum (Bishop, 2006, Section 7.1). Second, it has been shown its discriminative power in a great number of applications.

The drawbacks are two-fold. First, in its fundamental form the SVM is a two-class classifier. Thus, we have to decide how to handle multi-class problems. Typical solutions are to build one-versus-all or one-versus-one classifiers, where several two-class classifiers are combined to yield a joint decision for all classes. Second, the SVM is a discriminative classifier, thus, does not provide class conditional probabilities, which is essential when using the output of a classifier as input for further high-level processing.

To overcome the lack of class conditional probabilities, Platt (2000) propose to fit a logistic sigmoid function to the output of a trained SVM. To avoid overfitting, the data used for estimating the according parameters must be independent from those used to train the SVM itself. Thus, additional training data are needed or the amount of data available for training is reduces, respectively.

By intuition it becomes clear that the resulting probabilities may be unreliable for points at specific positions. Let us assume two classes concentrated on two dense clusters, perfectly separable. The decision boundary will be a hyperplane between them. The fitted logistic sigmoid function will tell that every sample on the correct side, but near the decision boundary have low probability to be part of the according class, while samples far away from the decision boundary will get a high probability of being part of the according class. Let us assume a sample far away from the decision boundary and far away from the cluster of training samples. Platts method will give this sample a high class conditional probability. But, looking on the situation from a generative point of view, we intuitively give this sample a low probability of being member of this class. For this, the probabilistic outcomes of SVMs tend to be too optimistic. Also Tipping (2001) showed that SVMs merely give a poor approximation of class conditional probabilities. Hence, we will use SVM as long as we do not need a probabilistic output for further processing. This is the case in the evaluation of shapelets in Section 4.6. For high-level reasoning, in Chapter 5, we need the probabilistic output of a classifier to get evidence for object classes we aim at, from bottom-up. This is realised by the IVM.

2.4.4.2 Import Vector Machine

As SVM, the import vector machine (IVM) is a discriminative classifier, therefore, usually ensures better discriminative power than generative models, but it produces classwise probabilities for test samples. It is inherently made for multi-class problems. The IVM, as proposed by Zhu and Hastie (2005), was shown to get a state-of-the-art classification performance, comparable to SVM and other commonly used methods, while having advantages given above. For that reason, we will use the IVM for object detection in Chapter 5.

The IVM is derived from the basic logistic regression model which, in case of two classes, is given

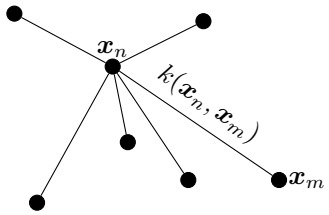


Figure 2.8: Instead of using the original features $\boldsymbol{x}$ SVM, as well as IVM, use the affinities between each feature $\boldsymbol{x}_n$ to all other features $\boldsymbol{x}_m$. These are given by the kernel function $k(\boldsymbol{x}_n, \boldsymbol{x}_m)$.

by

$$P(y = 1|\boldsymbol{x}; \boldsymbol{w}) = \frac{1}{1 + \exp(-\boldsymbol{w}^\mathsf{T}\boldsymbol{x})} \tag{2.98}$$

for feature $\boldsymbol{x}$ belonging to class 1 and parameters $\boldsymbol{w}$. The a posterior probability for $\boldsymbol{x}$ belonging to class 2 directly results from (2.98)

$$P(y = 2|\boldsymbol{x}; \boldsymbol{w}) = 1 - P(y = 1|\boldsymbol{x}; \boldsymbol{w}) \ . \tag{2.99}$$

We do not model the class conditional probabilities, but directly aim at maximising the class conditional a posteriori probability. Reversely, the decision boundary is implicitly given by the hyperplane for which $P(y = 1|\boldsymbol{x}; \boldsymbol{w}) = P(y = 2|\boldsymbol{x}; \boldsymbol{w})$.

The two-class model can be generalised to the multi-class case by

$$P(y = c|\boldsymbol{x}; \boldsymbol{w}) = \frac{\exp(\boldsymbol{w}_c^\mathsf{T}\boldsymbol{x})}{\sum_i \exp(\boldsymbol{w}_i^\mathsf{T}\boldsymbol{x})} \ . \tag{2.100}$$

Here the parameter vector $\boldsymbol{w}$ is the concatenation of parameter vectors $\boldsymbol{w}_c$ of all C classes. If the features are of dimension D, the parameter vector has $D \cdot C$ elements, which is linear in the number of classes. The goal for the learning phase is to determine the parameter vector $\boldsymbol{w}$ from training data. Thereby, we take into account that there are only $(C - 1) \cdot D$ independent parameters due the fact that all posterior probabilities sum up to 1.

We further extend the model to kernel logistic regression (KLR) to allow for non-linear models by introducing a kernel function $k(\boldsymbol{x}_n, \boldsymbol{x}_m)$

$$P(y = c|\boldsymbol{x}; \boldsymbol{w}) = \frac{\exp\left(\boldsymbol{w}_c^\mathsf{T}\boldsymbol{k}(\boldsymbol{x})\right)}{\sum_i \exp\left(\boldsymbol{w}_i^\mathsf{T}\boldsymbol{k}(\boldsymbol{x})\right)} \tag{2.101}$$

whereby, we collect all kernel functions in one vector

$$\boldsymbol{k}(\boldsymbol{x}) = [k(\boldsymbol{x}, \boldsymbol{x}_m)]_{m=1\ldots N} \ .$$

Thus, the original features are replaced by their affinities to all other features. This is visualised in Figure 2.8.

The parameters of the model are determined by minimising the regularised negative log likelihood function

$$\mathcal{Q}(\boldsymbol{w}) = -\frac{1}{N}\sum_c \sum_n t_{nc} \ln P(y_n = c|\boldsymbol{x}_n; \boldsymbol{w}) + \lambda_{\text{ivm}} \boldsymbol{w}^\mathsf{T}\boldsymbol{w} \tag{2.102}$$

whereby, t_{nc} is the c-th element of the index vector $\boldsymbol{t}_n$ which is a unit-vector with 1 at the c-th element, thus, might be interpreted as target probability for class c. To avoid overfitting, λ_{ivm} acts as regulariser over the parameters. The parameters of the model are determined iteratively using the iterated reweighted least squares optimisation, see Roscher et al. (2012).

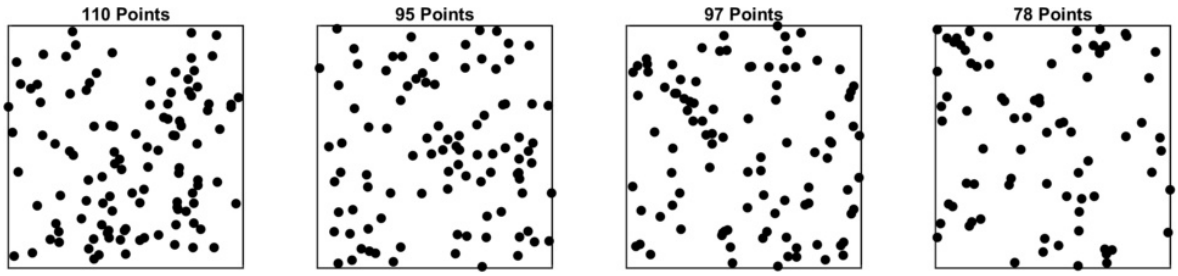


Figure 2.9: Simulation of a homogeneous spatial point process on the unit square, with $\beta = 100$.

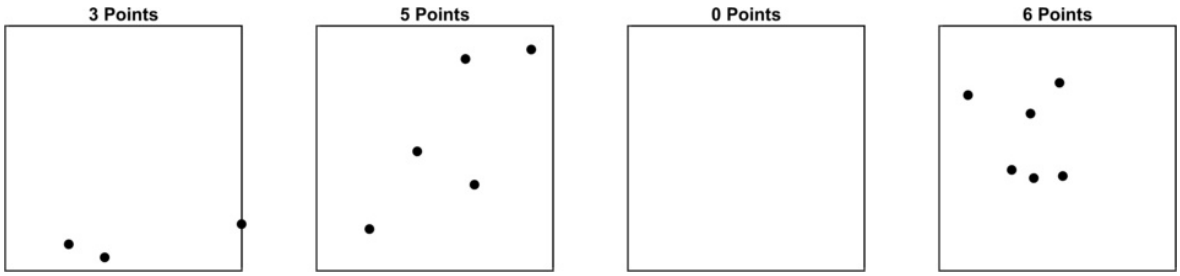


Figure 2.10: Simulation of a homogeneous spatial point process on the unit square, with $\beta = 5$.

In order to prevent the computational and memory expensive evaluation of the standard KLR model, Roscher et al. (2012), similarly to the selection of support vectors, propose a sparse version of KLR, realised by subset selection. It iteratively selects samples of the training set which most contribute to the solution, i.e. the minimising the objective function (2.102). In the sequel, the model is based on a subset of samples from the training set which are called import vectors in analogy to support vectors. The resulting method is called import vector machine.

Roscher et al. (2012) show that the IVM though conceptually being a discriminative model, inherently have a reconstructive component. As discriminative model the IVM estimates the decision boundary between target classes, but the import vectors are selected in order to minimise the cross-entropy function (2.102). As a consequence, all samples chosen as import vectors contribute to the class conditional posterior probability, but do not necessarily influence the decision boundary. Since the import vectors cover the distribution of the data samples, class conditional probabilities given by IVM are much more reliable than those of SVM and in consequence are much more suited for further usage in high-level models .

As for SVM, we keep in mind that we are able to solve the classification task for complex decision boundaries by introducing an appropriate kernel function. Furthermore, we keep in mind that we need to define the regularisation parameter λ_{ivm} and if needed the parameters of the kernel function, e.g., the kernel width σ_{RBF} of a RBF kernel, which we determine by grid search.

2.5 Marked Point Processes for Image Interpretation

In Chapter 5, we will model the geometry of image objects and their interactions to define a probability density on configurations of objects. Thereby, in our context, an object will be an axis-parallel rectangle, representing the bounding box of a target object class and parametrised by its location and some additional parameters, represented by vector $\boldsymbol{x}$. We call a set of objects $\mathcal{X} = \{\boldsymbol{x}_1 \dots \boldsymbol{x}_n\}$ a configuration. Defining a probability density $f(\underline{\mathcal{X}})$ on configurations of objects, allows us to identify the most probable configuration by maximising this density.

For that task, spatial point processes are useful as statistical model for the analysis of observed

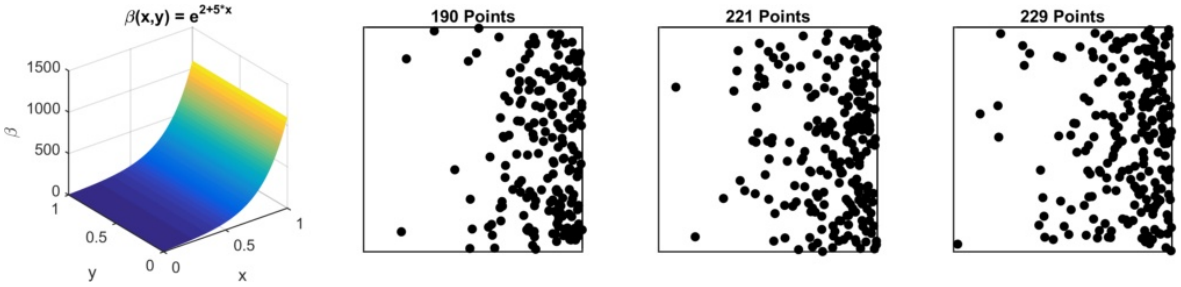


Figure 2.11: Simulation of an inhomogeneous spatial point process on the unit square, with $\beta = e^{2+5x}$.

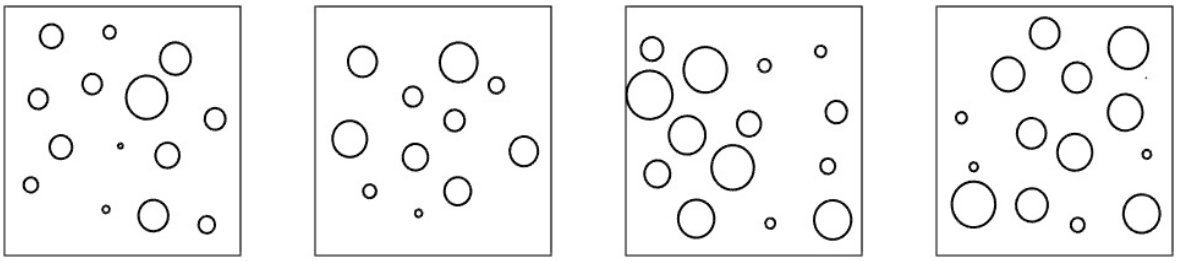


Figure 2.12: Simulation of a marked point process on the unit square. Each point is attached with the radius of a circle.

patterns of *points* represented by the *location* of objects. Further, marked point processes are of special importance in stochastic geometry for handling random sets of objects. They are a special form of spatial point process in a way that they represent configurations of objects which are points attached with some additional parameters, which are called marks. Actually, they are marked *spatial* point processes. This section gives a review of spatial point processes and especially marked point process (MPP), their basic terms and notation.

A spatial point process is a random variable $\underline{X}$, whose realisations are random configurations of *points* $\boldsymbol{x}$. One realisation is called $\mathcal{X}$. Figure 2.9 shows four samples of a spatial point process with mean number of points $\beta = 100$ within the unit square. Intuitively, these samples show points in 2D, which are uniformly spread over the unit square. The number of points for each sample is a random number, following a Poisson distribution with mean β. Figure 2.10 shows four samples of another spatial point process with mean number of points $\beta = 5$ within the unit square. We see that the empty set is a sample of such point process, too. Both Figure 2.9 and 2.10, are examples of Poisson processes, which are the most random point process which we will describe in more detail later. Further, these are examples of homogeneous point processes, where the expectation of number of points within a certain region is same everywhere, thus, β is constant. Figure 2.11 shows three samples of an inhomogeneous point process with mean $\beta = e^{2+5x}$, whose function plot is shown on the very left of Figure 2.11. We see that the density of points growth with the x-direction.

Finally, Figure 2.12 shows samples of a marked point process, where we uniformly spread points on the unit square, again, which we mark by an additional parameter representing the radius of a circle. We realize non-overlapping circles by thinning, thus, first uniformly sampling points and their radius and afterwards removing overlapping circles. We see that this is not a homogeneous point process anymore, as the density of points depends on the parameters of objects itself.

Let us assume a spatial Poisson process, as shown in Figures 2.9 and 2.10. Further, assume the distribution of points is given by the measure π. We will see that the density $f(\mathcal{X})$ of any, probably inhomogeneous, point process can be defined by its distribution $F_X(\mathrm{d}\mathcal{X}) = f(\mathcal{X})\ \pi(\mathrm{d}\mathcal{X})$ with respect to the measure π of a reference Poisson process, as those shown in Figures 2.9 and 2.10. Once, we made the formal definitions related to π, we will define $f(\mathcal{X})$ in terms of a Gibbs density ignoring π.

But, in order to optimise, thus, to find the configuration $\mathcal{X}$ which maximises $f(\mathcal{X})$, we will construct a Markov chain with $F_X(\mathrm{d}\mathcal{X})$ as its target distribution for which we need a clear understanding of π.

As the formal definitions, given in the literature, related to the applications of MPPs, are abstract and hard to follow for the non-specialist, we want to introduce in Section 2.5.1 some intuitive explanations. We give the general and more formal definitions at the end of each paragraph. To get a more detailed introduction into spatial point processes we strongly recommend Baddeley (2007).

Afterwards, Section 2.5.2 discusses the stability of a given point process, which is essential when constructing a Markov chain with F_X as its stationary distribution. Sections 2.5.3 and 2.5.4 introduce the Markov marked point process which we use to define a Gibbs energy on configurations of objects and show the relation to Markov Random Fields.

2.5.1 Background Theory on Marked Point Processes

2.5.1.1 Spatial Point Processes

Point processes are defined for arbitrary dimensions. One special form are one dimensional point processes where we usually model sequences of points during time. A one-dimensional point process has a natural ordering for which we may describe the process by arrival times or inter-arrival times. These natural ordering is absent in higher dimensions.

Throughout this thesis, we are dealing with spatial point processes, which are random configuration of points in d dimensions, usually 2D in our case.

A spatial point process is a random variable $\underline{\mathcal{X}}$ whose realisations are random configurations of points $\boldsymbol{x}$. One realisation is called $\mathcal{X}$.

As it is non trivial to describe the moments of a random configuration of points, its statistics are described in terms of region counts

$$N(\mathcal{B}) = \text{number of points falling in } \mathcal{B} \tag{2.103}$$

for a closed set $\mathcal{B} \subset \mathbb{R}^d$. Given all counting variables $N(\mathcal{B})$, for all subsets $\mathcal{B}$, we are able to reconstruct the position of all points. The positions $\boldsymbol{x}$ of points are characterised by $N(\{\boldsymbol{x}\}) > 0$, whereby, we denote by $\{\boldsymbol{x}\}$ the region containing $\boldsymbol{x}$. This leads to the definition of a point process as collection of random variables $\underline{N}(\mathcal{B})$. Given the requirements of a measure, as given in Section 2.2.1.1, we define the point process as random measure, in which the values $N(\mathcal{B})$ are non-negative integers (Baddeley, 2007).

A point process is called *locally finite* if $N(\mathcal{B}) < \infty$ with probability 1 for all $\mathcal{B} \subset \mathbb{R}^d$; thus, any bounded region contains only a finite number of points. Further, we say the process is *simple* if $N(\{\boldsymbol{x}\}) \leq 1 \quad \forall\, \boldsymbol{x} \in \mathbb{R}^d$, thus, no two points coincide.

To denote a point process, we use both notations, $\underline{\mathcal{X}}$ to denote the random configuration of points or $\underline{N}$ for the counting variable according to the same process.

Let us assume a point process in $\mathbb{R}^2$. We denote its *intensity* by β, which is the expected amount of points per unit area. Further, we denote by $\lambda_2(\mathcal{B})$ the Lebesgue measure in $\mathbb{R}^2$, thus, the area of $\mathcal{B}$. Then, the expectation of $\underline{N}(\mathcal{B})$ is given by

$$\mathbb{E}\left(\underline{N}(\mathcal{B})\right) = \beta\lambda_2(\mathcal{B}) \ . \tag{2.104}$$

2.5.1.2 Poisson Processes

The most random point process is the Poisson process, where the counting variable follows a Poisson distribution. The probability mass function of the Poisson distribution with mean μ is given by

$$\mathbb{P}\left(\underline{N} = k\right) = e^{-\mu}\frac{\mu^k}{k!} \ . \tag{2.105}$$

Therefrom, the **spatial Poisson process**, with uniform intensity $\beta > 0$, is a point process in $\mathbb{R}^2$ such that the count $\underline{N}(\mathcal{B})$ has a Poisson distribution with mean $\mu(\mathcal{B}) = \beta\lambda_2(\mathcal{B})$. For all disjoint regions $\mathcal{B}_1 \dots \mathcal{B}_m$ the number of points $N(\mathcal{B}_1) \dots N(\mathcal{B}_m)$ falling into these regions are independent.

Furthermore, given a Poisson process in $\mathcal{S} \subset \mathbb{R}^2$, containing n points, these points are conditionally independent and (spatially) uniformly distributed in $\mathcal{S}$. On the other hand, given a subset $\mathcal{B} \subset \mathcal{S}$, the number of points falling into $\mathcal{B}$ is binomial distributed

$$\mathbb{P}\left(\underline{N}(\mathcal{B}) = k \mid \underline{N}(\mathcal{S}) = n)\right) = \binom{n}{k} p^k (1-p)^{n-k} \quad \text{with} \quad p = \frac{\lambda_2(\mathcal{B})}{\lambda_2(\mathcal{S})}. \tag{2.106}$$

Given the properties above, we can simulate a Poisson process with intensity β in $\mathcal{S}$, by drawing a random number $m \sim \text{Poisson}(\beta\lambda_2(\mathcal{S}))$ and then generating m independent, uniformly distributed points on $\mathcal{S}$. To simulate the Poisson processes in Figures 2.9 and 2.10, we take $\mathcal{S} = [0,1] \times [0,1] \subset \mathbb{R}^2$, therefore, $\lambda_2(\mathcal{S}) = 1$ and $m \sim \text{Poisson}(\beta)$.

We say μ is the measure of the Poisson process on $\mathcal{S}$, such that

$$\mu(\mathcal{B}) = \mathbb{E}\left(N(\mathcal{B})\right), \quad \mathcal{B} \subset \mathcal{S} \tag{2.107}$$

called intensity measure, provided $\mu(\mathcal{B}) < \infty$ for all compact $\mathcal{B}$.

2.5.1.3 Marked Point Processes

If we attach additional parameters to each point, e.g., let each point be a circle with centre at $\boldsymbol{x}$ and additional parameter r, or a rectangle with additional parameters (w, h) for width and height, then we have a MPP. The MPP $\underline{\mathcal{Y}}$ is a random variable whose realisations are configurations of *o*bjects.

A formal definition can be found in (Baddeley, 2007): A MPP on a space $\mathcal{S}$ with marks in a space $\mathcal{M}$ is a point process $\underline{\mathcal{Y}}$ on the product space $\mathcal{S} \times \mathcal{M}$ such that $\underline{N}(\mathcal{K} \times \mathcal{M}) < \infty$ almost surely for all compact $\mathcal{K} \subset \mathcal{S}$. That is, the corresponding projected process, thus, points without marks, is locally finite.

Let $\underline{\mathcal{X}}$ be the projected point process in $\mathcal{S}$, thus, points without marks, again. Then, $\underline{\mathcal{X}}$ is a Poisson process in $\mathcal{S}$ with intensity μ, and given $\underline{\mathcal{X}}$, the marks attached to the points of $\underline{\mathcal{X}}$ are independent and identically distributed (i.i.d.) with distribution η on $\mathcal{M}$. Further, $\underline{\mathcal{Y}}$ is a Poisson process in the product space $\mathcal{S} \times \mathcal{M}$ with intensity measure $\mu \cdot \eta$.

The intensity measure of a MPP then takes the form

$$\nu(\mathcal{A} \times \mathcal{B}) = \mu(\mathcal{A})\eta(\mathcal{B}) = \beta\lambda_d(\mathcal{A})\eta(\mathcal{B}) \quad \forall\, \mathcal{A} \subset \mathcal{S}, \mathcal{B} \subset \mathcal{M}\ . \tag{2.108}$$

2.5.1.4 Distribution and Density of Finite Point Processes

Before we define the distribution of a point process, we detail the definition of the according probability spaces, which allow us again to describe the point process in terms of its counting measure instead of the whole configuration. Throughout this thesis, we are dealing with finite point processes, thus, $N(\mathcal{S}) <$ inf almost surely.

Let $(\mathcal{S}, \mathcal{B}, \lambda_d)$ be a measure space, with $\mathcal{S} \subset \mathbb{R}^d$, $\mathcal{B}$ being the σ-field on $\mathcal{S}$ and λ_d the Lebesgue measure on $\mathbb{R}^d$. Further, for a finite number n of points, let Ω_n be the set of all unordered configurations $\mathcal{X} = \{\boldsymbol{x}_1, \dots, \boldsymbol{x}_n\} \in \Omega_n$ of points $\boldsymbol{x}_i \in \mathcal{S}$. Then, we define the space of configurations by $\Omega = \cup_n \Omega_n$. Equivalently we denote by N the set of all counting measures on $\mathcal{S}$. The according σ-algebra $\mathcal{N}$ then is given by the mappings $N : \{\boldsymbol{x}_1, \dots \boldsymbol{x}_n\} \to \sum_{i=1}^{n} \mathbf{1}\{\boldsymbol{x}_i \in \mathcal{S}\}$, where $\mathbf{1}\{\cdot\}$ denotes the indicator function. Thus, N is a mapping $\Omega \to \mathsf{N}$ and formally gives the number of points falling into $B \in \mathcal{B}$. The space N together with $\mathcal{N}$ is the outcome space for the point process on $\mathcal{S}$. We also say a point process on $\mathcal{S}$ is a mapping from a probability space $(\Omega, \mathcal{A}, \mathbb{P})$ to the outcome space $(\mathsf{N}, \mathcal{N})$ (Baddeley, 2007). Its distribution is given through its counting measure N_X by the probability measure

$$P_X(A) = \mathbb{P}\left(N_X \in A\right), \qquad A \in \mathcal{N}\ . \tag{2.109}$$

For example, the distribution of the finite Poisson process $\underline{\mathcal{X}}$ on $\mathcal{S}$, with standard measure λ_d on $\mathcal{S}$, which we denote by π, is given by

$$\pi(A) = e^{-\lambda_d(S)} \sum_{n=0}^{\infty} \frac{1}{n!} \int_{\mathcal{S}} \dots \int_{\mathcal{S}} \mathbf{1}\{I_n(\boldsymbol{x}_1, \dots, \boldsymbol{x}_n) \in A\} \lambda_d(\mathrm{d}\boldsymbol{x}_1) \dots \lambda_d(\mathrm{d}\boldsymbol{x}_n) \qquad \forall A \in \mathcal{N} \tag{2.110}$$

where I_n is a mapping from the space of ordered n-tuples to the set of all counting measures with total mass n, i.e. $I_n(\boldsymbol{x}_1, \dots, \boldsymbol{x}_n) = \delta_{x_1} + \dots \delta_{x_n}$, (Baddeley, 2007). According to the Poisson distribution with intensity $\lambda_d(\mathcal{S})$, π distributes the points uniformly in $\mathcal{S}$ (Stoica et al., 2005), as expected.

Distributions of more complicated models, e.g., using another intensity, or which take into account interactions between objects, can be constructed by specifying a Radon-Nikodym derivative with respect to π. Therefor, we assume a measurable function f which satisfies

$$\int_{\mathcal{S}} f(\mathcal{X}) \pi(\mathrm{d}\mathcal{X}) = 1 \ , \tag{2.111}$$

and define the point process distribution

$$F_X(A) = \int_A f(\mathcal{X}) \pi(\mathrm{d}\mathcal{X}) \ , \tag{2.112}$$

for any event $A \in \mathcal{N}$. Due to the Radon-Nikodym theorem, the function f is the pdf of the point process with distribution F_X. We see that the distribution of the underlying Poisson process takes the role of the Lebesgue measure in $\mathbb{R}^d$. We say the distribution of the point process is dominated by the distribution of the underlying Poisson process π.

As an example we take the uniform Poisson process with intensity β, referred to the standard measure λ_d. Its pdf is

$$f_\beta(\mathcal{X}) = \beta^{N(\mathcal{X})} e^{(1-\beta)\lambda_d(\mathcal{S})} \ , \tag{2.113}$$

whereby, $e^{(1-\beta)\lambda_d(\mathcal{S})}$ is the normalising constant. It is not trivial to see that this is a valid pdf, as we need to integrate over $\mathcal{N}$. But, we verify that this is the density of the Poisson process we ask for, by putting (2.113) and (2.110) into (2.112)

$$\begin{aligned}
\int_A f_\beta(\mathcal{X}) \pi(\mathrm{d}\mathcal{X}) &= \int_A \beta^{N(\mathcal{X})} e^{(1-\beta)\lambda_d(\mathcal{S})} \pi(\mathrm{d}\mathcal{X}) \\
&= \beta^{N(\mathcal{X})} e^{(1-\beta)\lambda_d(\mathcal{S})} \cdot \\
&\quad e^{-\lambda_d(S)} \sum_{n=0}^{\infty} \frac{1}{n!} \int_{\mathcal{S}} \dots \int_{\mathcal{S}} \mathbf{1}\{I_n(\boldsymbol{x}_1, \dots, \boldsymbol{x}_n) \in A\} \lambda_d(\mathrm{d}\boldsymbol{x}_1) \dots \lambda_d(\mathrm{d}\boldsymbol{x}_n) \\
&= e^{-\beta\lambda_d(S)} \sum_{n=0}^{\infty} \frac{1}{n!} \int_{\mathcal{S}} \dots \int_{\mathcal{S}} \mathbf{1}\{I_n(\boldsymbol{x}_1, \dots, \boldsymbol{x}_n) \in A\} \beta\lambda_d(\mathrm{d}\boldsymbol{x}_1) \dots \beta\lambda_d(\mathrm{d}\boldsymbol{x}_n) \\
&= e^{-\mu(S)} \sum_{n=0}^{\infty} \frac{1}{n!} \int_{\mathcal{S}} \dots \int_{\mathcal{S}} \mathbf{1}\{I_n(\boldsymbol{x}_1, \dots, \boldsymbol{x}_n) \in A\} \mu(\mathrm{d}\boldsymbol{x}_1) \dots \mu(\mathrm{d}\boldsymbol{x}_n) \ ,
\end{aligned}$$

which is equivalent to the measure π_μ of the Poisson process with intensity $\mu = \beta\lambda_d$.

2.5.2 Stability of the Point Process

To ensure integrability of the point process density, the so called Papangelou conditional intensity must satisfy the following condition

$$\lambda(\boldsymbol{x} \mid \mathcal{X}) = \frac{f(\mathcal{X} \cup \{\boldsymbol{x}\})}{f(\mathcal{X})} \leq \Lambda \ , \qquad 0 < \Lambda < \infty, \mathcal{X} \in \Omega, \boldsymbol{x} \in \mathcal{S} \ . \tag{2.114}$$

We say the process is locally stable if it satisfies (2.114), thus, if its conditional intensity is bounded. This condition also ensures the convergence of a Markov Chain simulating the point process.

2.5.3 Markov Point Processes

Markov point processes, introduced by Baddeley and Lieshout (1993), are defined in terms of local interactions between points of the configuration.

We use the symbol $\sim$ to denote neighbouring objects. It is a symmetric and reflexive relation; thus, for

$$\boldsymbol{x}_i \sim \boldsymbol{x}_j \Leftrightarrow \boldsymbol{x}_j \sim \boldsymbol{x}_i \quad \forall \boldsymbol{x}_i, \boldsymbol{x}_j \in \mathcal{S} \tag{2.115}$$

$\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ are said to be neighbours. For example, we may define all objects within a ball of radius r to be neighboured

$$\boldsymbol{x}_i \sim \boldsymbol{x}_j \quad \text{if} \quad d(\boldsymbol{x}_i, \boldsymbol{x}_j) \leq r \tag{2.116}$$

for a distance measure d. The neighbourhood $\mathrm{Ne}(\boldsymbol{y})$ of an object $\boldsymbol{y}$ contains all its neighbours

$$\mathrm{Ne}(\boldsymbol{y}) = \{\boldsymbol{x} \in \mathcal{S} : \boldsymbol{x} \sim \boldsymbol{y}\} \ . \tag{2.117}$$

Let $\underline{\mathcal{X}}$ be a point process. $\underline{\mathcal{X}}$ is a Markov point process if its conditional intensity of a point $\boldsymbol{x}$ conditioned on a configuration $\mathcal{X}$ only depends on itself and its neighbours $\mathrm{Ne}(\boldsymbol{x})$, which can be seen as the spatial analogue to the Markov property of a time series.

Additionally, the Hammersley-Clifford theorem tells that a point process density is Markov with respect to a neighbourhood relation $\sim$ if and only if there is a measurable function ψ such that

$$f(\mathcal{X}) = \prod_{\boldsymbol{c} \subseteq \mathcal{X}} \psi(\boldsymbol{c}) \ . \tag{2.118}$$

Thereby, a configuration $\boldsymbol{c}$ is said to be a clique if all of its members are each others neighbour. Thus, the Hammersley-Clifford theorem provides a factorisation of a density of a Markov point process in terms of interactions between points.

This leads to the special type of point processes, we will use throughout this thesis. If $\underline{\mathcal{X}}$ is a finite point process and its pdf has the form

$$f(\mathcal{X}) = \exp\left(V_0 + \sum_{\boldsymbol{x} \in \mathcal{X}} V_1(\boldsymbol{x}) + \sum_{\{\boldsymbol{x} \sim \boldsymbol{y}\} \subset \mathcal{X}} V_2(\boldsymbol{x}, \boldsymbol{y}) + \ldots\right) \tag{2.119}$$

where V_k are called the potentials of order k, then $\underline{\mathcal{X}}$ is a *finite Gibbs process*.

We may interpret $-\ln f(\mathcal{X})$ as potential energy of the whole configuration $\mathcal{X}$. Usually we denote the energy by U. According to that $U_1(\boldsymbol{x}) = -V_1(\boldsymbol{x})$ might be interpreted as the energy at a single point location and $U_2(x, y) = -V_2(x, y)$ as energy between two points x and y, restricted to neighbouring points. V_0 is the normalising constant to ensure f to be a proper density.

In our application, we will use U_1 and U_2, solely. The former usually is called unary potential, unary energy or data term, while the latter is called pairwise potential, pairwise energy or configuration term. The normaliser V_0 usually is out of interest as we aim at maximising $f(\mathcal{X})$, thus, compare densities of different configurations, therefore the normaliser cancels out. We will use this representation in Chapter 5 to define the density of a MPP by modelling the interaction between neighbouring objects by pairwise energies U_2 and the consistency of objects with the image signal by unary energies U_1.

2.5.4 Relation to Markov Random Fields

A Markov random field (MRF) is a graphical model over an undirected graph $G(\mathcal{V}, \mathcal{E})$ which models the joint distribution of sites $\underline{x}$ in a configuration $\boldsymbol{x}$. We use $\mathcal{V} = \{1, \ldots, N\}$ to denote the set of nodes representing random variables $\underline{x}_n$ and $\mathcal{E} = \{\{i, j\} \mid i, j \in \mathcal{V}\}$ to denote undirected edges between nodes i and j. We use the term cliques, similar to the definition before, to denote completely connected subgraphs. The potentials are modelled by interactions between neighbouring sites within cliques and we will see that this is related to the modelling of a finite Gibbs process as seen before.

Given the Markov property and Hammersley-Cliffords theorem, the joint density over all sites of the graph factorises

$$p(\boldsymbol{x}) = p(x_1, \ldots, x_N) = \frac{1}{Z} \prod_{c \in \mathcal{C}} \psi_c(\boldsymbol{x}_c) \tag{2.120}$$

with potential functions $\psi_c(\boldsymbol{x}_c) > 0$ referring to maximal cliques $c \in \mathcal{C}$. The partition function Z sums over all states of the complete Markov field to normalise p. Equivalently, we may use the Gibbs distribution

$$p(\boldsymbol{x}) = \frac{1}{Z} \mathrm{e}^{-U(\boldsymbol{x})} \ , \tag{2.121}$$

where the energy is represented as sum over local energies, again

$$U(\boldsymbol{x}) = \sum_{c \in \mathcal{C}} U_c(\boldsymbol{x}_c) \ . \tag{2.122}$$

Please note that in the literature U_c and ψ_c are called potentials as well as V_i in (2.119). Actually, they are related by $U_c(\boldsymbol{x}_c) = -\ln \psi_c(\boldsymbol{x}_c)$. To distinguish them we call U_c local energy, and ψ_c the potential.

Please note that instead of maximising the pdf (2.120) we equivalently may minimise the energy in (2.121).

We realise that the representation in (2.119) and (2.121), (2.122) are similar. The difference is the definition of a static graph to model the configuration of sites within the MRF framework. Thus, here the potentials are defined over cliques within a static graph, whereas the Gibbs process takes into account all neighbouring objects according to the definition of neighbourhood relations, which may differ for different realisation of the process. Most importantly, the definition of the MRF graph means to define the number of objects or nodes, respectively, and the relation between objects beforehand.

Therefrom, marked point process can bee seen as extension of MRFs.

2.6 Optimisation Using Reversible Jump Markov Chain Monte Carlo Sampling and Simulated Annealing

Markov chain Monte Carlo (MCMC) methods are a large class of sampling algorithms. One of them, the Metropolis algorithm, is supposed to be one of the ten algorithms that have had the greatest influence on science and engineering in the 20th century (Beichl and Sullivan, 2000).

In this section, we will introduce MCMC for sampling from unknown and potentially complex probability density functions. We show how we use MCMC in combination with simulated annealing to reach the global optimum of such functions.

In Chapter 5, we will apply these techniques to the task of geometric image interpretation of facade images. For that we formulate an energy function which combines bottom-up evidence given by a classifier and top-down high-level knowledge about typical object configurations which we learned from training data. To find the global optimum of this function we use reversible jump Markov chain Monte Carlo (rjMCMC) coupled with simulated annealing.

Monte Carlo methods solve problems by random sampling. As an example, suppose we wish to evaluate the partition function

$$Z = \int \tilde{p}(x) \mathrm{d}x \tag{2.123}$$

with $\tilde{p}$ being an unnormalised probability density function, thus, $p(x) = \frac{1}{Z} \int \tilde{p}(x) \mathrm{d}x$. Further, assume we are not able to sample directly, but we are able to evaluate $\tilde{p}(x)$ for every x, cf. Figure 2.13. If we find a proposal density $q(x)$, from which we are able to sample easily, we can approximate Z by

$$Z = \int \tilde{p}(x) \mathrm{d}x = \int \frac{\tilde{p}(x)}{q(x)} q(x) \mathrm{d}x \approx \sum_{i=1}^{N} \frac{\tilde{p}(x_i)}{q(x_i)} / N \tag{2.124}$$

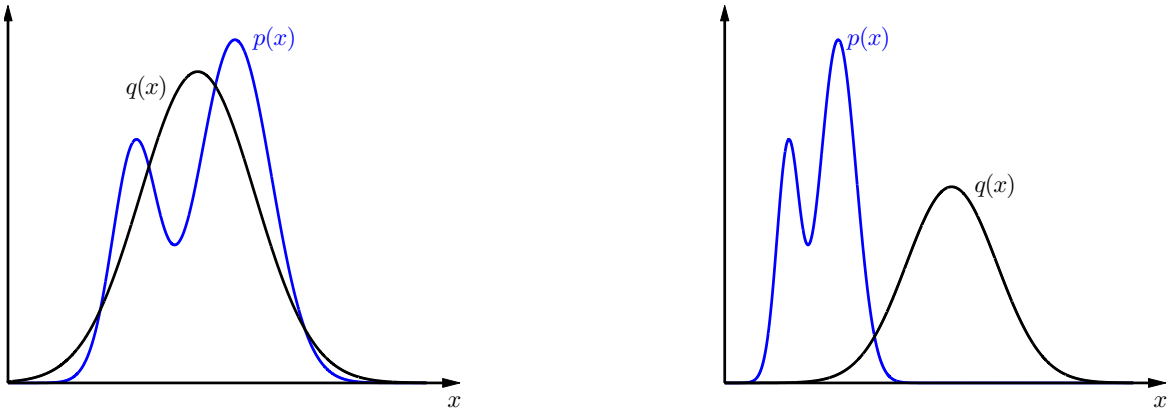


Figure 2.13: Monte Carlo sampling to evaluate the partition function. Instead of directly sampling the target density $p(x)$ we sample from a proposal density $q(x)$ to approximate the partition function Z of $p(x)$. **Left:** A proper choice of proposal $q(x)$. **Right:** In case the proposal density does not fit the target density, we need much more samples to obtain a reliable result.

assuming the number of samples N is large. This works fine, as long as the proposal fits the target density as good as possible. In the simplest case we may take a Uniform distribution in the range of $\tilde{p}(x)$. As soon as the deviation between target and proposal gets larger, we are still able to approximate Z, but the number of samples, needed to obtain a reliable result, goes to infinity.

Next, suppose we wish to sample from a density function $p(x)$. If we are able to invert the corresponding cdf $P(x)$, we may draw a sample $\underline{x}_0$ from $p(x)$ through

$$\underline{x}_0 = P^{-1}\left(\underline{u}_0\right) \ , \tag{2.125}$$

where $\underline{u}_0$ is sampled uniformly from $\mathcal{U}(0,1)$. The principle is shown in Figure 2.14.

In general, if we do not wish to sample from a standard distribution, we are not able to sample directly. But, if we are able to evaluate the target density up to the normalising constant Z as before, thus

$$p(x) = \frac{1}{Z}\tilde{p}(x) \tag{2.126}$$

and if we further find a simple density $q(x)$, from which are able to sample easily, and for which we find any k such that $k\ q(x) \geq \tilde{p}(x)\ \forall x$, we may sample from $\tilde{p}(x)$ by rejection sampling. The principle is shown in Figure 2.15. The algorithm is given in Algorithm 2.1.

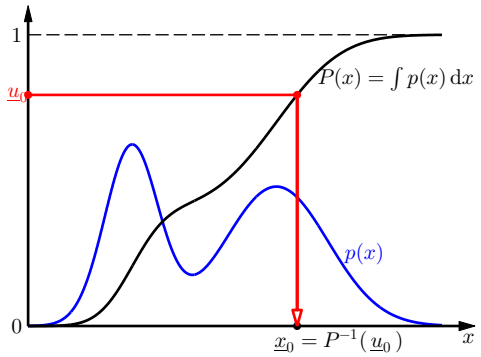


Figure 2.14: Direct sampling. If we are able to invert the distribution function $P(x)$, we can sample random numbers $\underline{x} \sim p(x)$ through uniformly sampled numbers $\underline{u} \sim \mathcal{U}(0,1)$.

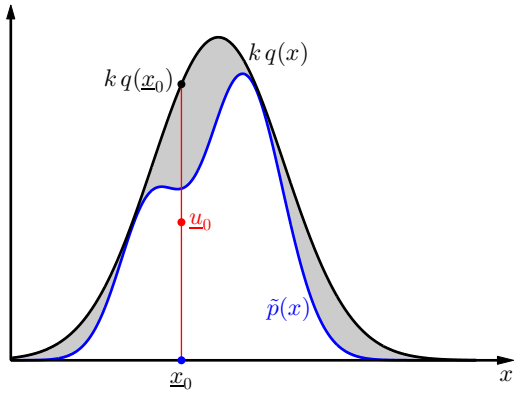


Figure 2.15: Rejection sampling. We sample $\underline{x}_0$ from a proposal density $kq(x)$. If a uniformly sampled number $\underline{u}_0 \sim \mathcal{U}(0, kq(\underline{x}_0)) \leq \tilde{p}(\underline{x}_0)$, we accept the sample. Thus, the shaded area is the rejection area.

As accepted samples are uniformly distributed under $\tilde{p}(x)$, we see that $\underline{x}_0 \sim p(x)$. The probability of accepting a sample x_0 is

$$\mathbb{P}_{\text{accept}}\,(x_0) = \frac{p(x_0)}{q(x_0)}. \tag{2.127}$$

Thus, the acceptance rate

$$\mathbb{P}_{\text{accept}}(x) = \int \frac{\tilde{p}(x)}{k\;q(x)}\;q(x)\;\mathrm{d}x = \frac{1}{k}\int \tilde{p}(x)\;\mathrm{d}x \tag{2.128}$$

depends on the shaded area of Figure 2.15 and should be as small as possible. The algorithm works if the proposal density does not fit the target density perfectly. But again, if the shaded area is too large we make a lot of rejected proposals. For this, it is not easy to find such a proposal that perfectly fits the target density.

There are several other sampling methods, e.g., adaptive rejection sampling, importance sampling or sampling-importance-resampling. For details we refer to the literature, e.g., Bishop (2006, ch. 11), Andrieu et al. (2003).

We keep in mind the idea of sampling from an unknown density by using a proposal density which is easy to sample from. However, for all of these methods, mentioned before, it is often difficult to find proper proposal densities which are easy to sample and that are good approximations of the target density.

Markov chains offer a more flexible way to sample from complex functions. Using MCMC algorithms we generate samples x exploring the state space $\mathcal{X}$ by constructing a Markov chain such that the drawn

Algorithm 2.1: Rejection sampling

Input : unnormalised target density $\tilde{p}(x)$, proposal density $kq(x) \geq \tilde{p}(x)$
Output : $\underline{x}_0 \sim p(x)$

Sample $\underline{x}_0 \sim kq(x)$
Sample $\underline{u}_0 \sim \mathcal{U}(0, kq(\underline{x}_0))$
if $\underline{u}_0 \leq \tilde{q}(\underline{u}_0)$ **then**
    accept
else
    reject
end
return

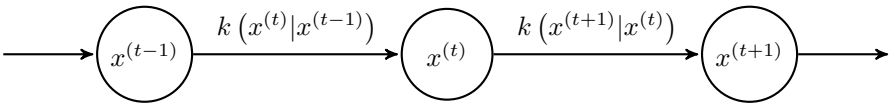


Figure 2.16: Markov Chain

samples follow the target density $p(x)$ (Andrieu et al., 2003).

2.6.1 Markov Chains

A Markov chain describes a stochastic process, cf. Fig 2.16. The state space $\mathcal{X}$ contains all possible values the random variable $\underline{x}$ may take. Whereby each state is given in terms of its probability density or probability mass function $p\left(x^{(t)}\right)$. The states of the Markov chain are governed by transition probabilities $k(\cdot|\cdot)$. For example, the transition probability of changing state $x^{(t-1)}$ to $x^{(t)}$ is given by $k(x^{(t)}|x^{(t-1)})$. Given the Markov property, the probability density of state $x^{(t)}$ only depends on the state before $x^{(t-1)}$ and the joint probability density of the chain simplifies to

$$p\left(x^{(t)}|x^{(t-1)},\dots x^{(0)}\right) = p\left(x^{(t)}|x^{(t-1)}\right) \ . \tag{2.129}$$

As an example, assume a discrete random variable $\underline{x}$ which can take three possible values. Let its state space be $\mathcal{X} = \{x_1, x_2, x_3\}$. The probability of changing from one state to another is given by distinct values shown in Figure 2.17, which we represent by transition matrix

$$K = \begin{bmatrix} 0.3 & 0.1 & 0.6 \\ 0.7 & 0.0 & 0.0 \\ 0.0 & 0.9 & 0.4 \end{bmatrix} \ . \tag{2.130}$$

For example, the probability of changing to x_3 being in x_2 is given by $K(3,2) = k(x_3 \mid x_2) = 0.9$. Further, assume an initial state $p\left(x^{(0)}\right) = \boldsymbol{p}^{(0)} = [1/3,\ 1/3,\ 1/3]^\mathsf{T}$. The transition between two states of the Markov chain is given by

$$\boldsymbol{p}^{(t)} = K\ \boldsymbol{p}^{(t-1)} \ . \tag{2.131}$$

Thus, the stochastic process evolves to $x^{(1)}$ by

$$\boldsymbol{p}^{(1)} = K\ \boldsymbol{p}^{(0)} = \begin{bmatrix} 0.33 \\ 0.23 \\ 0.43 \end{bmatrix} \ . \tag{2.132}$$

This is a simple example of a homogeneous Markov chain where K remains invariant for all t, thus, $K = K(x^{(t)}|x^{(t-1)})\ \forall_t$.

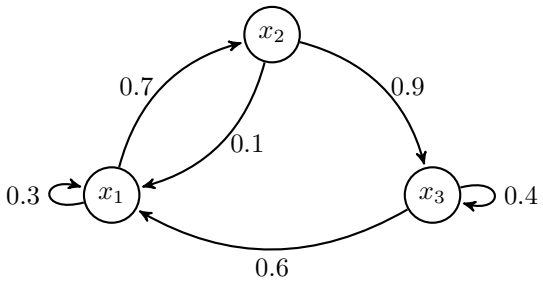


Figure 2.17: Example: A simple discrete Markov chain. The graph visualises the state space $\mathcal{X} = \{x_1, x_2, x_3\}$ of random variable $\underline{x}$ as nodes and their transition probabilities as directed edges. Its transition matrix is given in Equation (2.130).

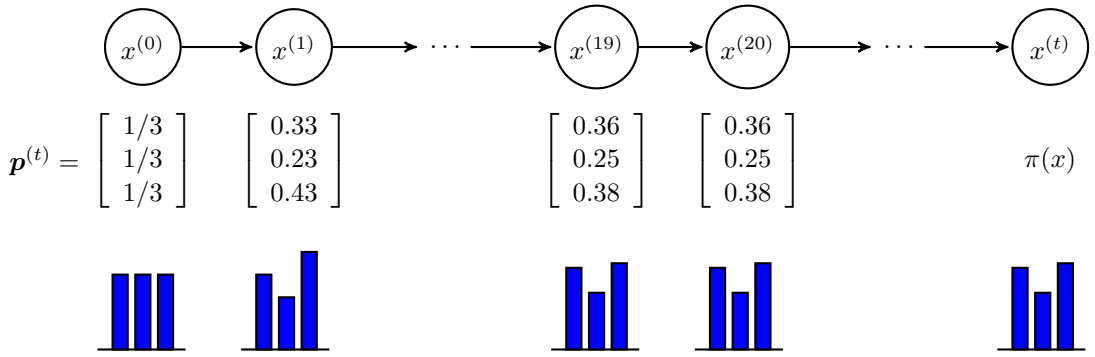


Figure 2.18: Example: Evolution of the discrete Markov chain, shown in Figure 2.17, to its stationary distribution $\pi(x)$.

After a couple of iterations the process converges to

$$K^t \, \boldsymbol{p}^{(0)} \;=\; \begin{bmatrix} 0.36 \\ 0.25 \\ 0.38 \end{bmatrix} \;=\; \pi(x), \tag{2.133}$$

independent of the initial state, cf. Figure 2.18. The stable distribution $\pi(x)$ is called stationary distribution and the goal of any MCMC simulation.

Convergence to the stationary distribution is given as long as K is a stochastic transition matrix, which has following properties

Irreducibility There must be a positive probability of visiting the whole state space, independent of the current state. In other words, the transition graph must be connected or any state of the chain has finite probability.

Aperiodicity The process is aperiodic if all its states are aperiodic. Thereby, a state is said to be aperiodic if the process may return to that state at *irregular* time. If the chain is irreducible, one aperiodic state implies all other states to be aperiodic, thus, the whole chain to be aperiodic.

Recurrence The process visits any state infinitely often almost surely.

Ergodicity The chain is aperiodic and recurrent.

The last property ensures the chain to reach the stationary distribution. We say, if the chain is stable at its stationary distribution it reaches equilibrium.

Detailed Balance To ensure to reach the desired invariant distribution $\pi(x)$, the reversibility condition

$$p\,(x)\,k(x'|x) = p\,(x')\,k(x|x') \tag{2.134}$$

must hold. Thus, changing to a new state x' given the process is at state x must be as probable as changing to state x given the process is at state x'. Detailed balance is a sufficient, but not necessary condition. If detailed balance holds, we say the Markov chain is reversible and it follows the existence of a stationary distribution.

2.6.2 Metropolis-Hastings Algorithm

In the section, we come back to sampling techniques. We introduce MCMC algorithms to generate samples x by exploring the state space $\mathcal{X}$. Therefor, we construct a Markov chain, such that at

Algorithm 2.2: Metropolis-Hastings Algorithm

1 chose initial state $x^{(0)}$;
2 **while** $t <$maxiter **do**
    // propose new sample
3 $x' \sim q(x'|x^{(t)})$;
    // acceptance probability
4 $A = A(x'|x^{(t)})$; // (2.136) or (2.137)
    // With probability A accept the move
5 $\alpha \sim \mathcal{U}(0,1)$;
6 **if** $A > \alpha$ **then**
7 $x^{(t+1)} = x'$; // accept the proposed sample
8 **else**
9 $x^{(t+1)} = x^{(t)}$; // reject
10 **end**
11 **end**

equilibrium the drawn samples follow the target density $p(x)$. We also say, we construct a chain with p as stationary or invariant density.

The most famous algorithm from that group is the Metropolis algorithm (Metropolis et al., 1953) and we will see that other algorithms, we will describe later, are adaptions of this idea.

Our goal is to sample from a density

$$p(x) = \frac{1}{Z}\tilde{p}(x) \ , \tag{2.135}$$

from which we possibly do not know the normaliser Z.

Again, we sample from a proposal density, but in contrast to rejection sampling, it depends from the current state $q = q(x)$. Assume the proposal to be symmetric $q\left(x'|x\right) = q\left(x|x'\right)$, the probability of accepting a proposed sample $x' \sim q(\cdot \mid x)$ is given by

$$A(x'|x) = \min\left(1, \frac{\tilde{p}(x')}{\tilde{p}(x)}\right) = \min\left(1, \frac{p(x')}{p(x)}\right) \ , \tag{2.136}$$

where we skip the tilde to avoid overloading the notation, keeping in mind that we still may use the unnormalised density.

In practice, we sample an auxiliary random variable $\underline{\alpha} \sim \mathcal{U}(0,1)$ and accept the proposed sample x' if $A > \alpha$ and reject, otherwise. In the latter case the chain stays simply at the current state.

Hastings (1970) relaxed the assumption of symmetric proposals and introduced sampling from arbitrary proposal densities, still dependent on the current state x. To ensure detailed balance, he introduced an additional factor into the acceptance probability, which becomes

$$A(x'|x) = \min\left(1, \frac{p(x')q(x|x')}{p(x)q(x'|x)}\right) \ . \tag{2.137}$$

In case the proposal becomes independent of state x, the second term cancels out, due to symmetry, and we obtain (2.136).

The Metropolis-Hastings algorithm is given in Algorithm 2.2. The difference between Metropolis and Hastings algorithm is the computation of the acceptance ratio. Thus, Algorithm 2.2 line 4 uses (2.136) or (2.137), depending on the kind of proposal distribution.

The transition probability for the Metropolis-Hasting algorithm is given by

$$\mathcal{Q}_{MH}(x'|x) = q(x'|x)A(x'|x) \ , \tag{2.138}$$

which we call transition kernel. We see that detailed balance holds because using (2.137)

$$\begin{aligned} p(x)q(x'|x)A(x'|x) &= \min\left(p(x)q(x'|x),\ p(x')q(x|x')\right) && (2.139)\\ &= \min\left(p(x')q(x|x'),\ p(x)q(x'|x)\right) && (2.140)\\ &= p(x')q(x|x')\ \min\left(1,\ \frac{p(x)q(x'|x)}{p(x')q(x|x')}\right) && (2.141)\\ &= p(x')q(x|x')A(x|x') && (2.142)\\ p(x)\mathcal{Q}_{MH}(x'|x) &= p(x')\mathcal{Q}_{MH}(x|x')\ , && (2.143) \end{aligned}$$

as required in (2.134). Thus, $p(x)$ indeed is the invariant density defined by the Metropolis-Hastings algorithm.

2.6.3 Mixtures of Kernels

A widely used technique within MCMC sampling is the mixture of different transition kernels. That is, if two kernels $\mathcal{Q}_1(\cdot \mid \cdot)$ and $\mathcal{Q}_2(\cdot \mid \cdot)$ have the same invariant density p, then the so called cycle hybrid kernel $\mathcal{Q}_1\mathcal{Q}_2$ and so called mixture hybrid kernel $w\mathcal{Q}_1 + (1-w)\mathcal{Q}_2$, using a weight $w \in [0,1]$, are transitions kernels with invariant density p, too (Andrieu et al., 2003). This allows us to create a mixture of kernels which explore the whole state space by using a global proposal and to explore the fine details around local maxima by using a local proposal.

In general, we express the proposition kernel as sum of several kernels, called sub-kernels, which are appropriate for the current state x, thus

$$\mathcal{Q}(\cdot \mid x) = \sum_m \mathcal{Q}_m(\cdot \mid x)\ . \tag{2.144}$$

Each sub-kernel is chosen with probability $j_m(x)$ with $\sum_m j_m(x) = 1$. Thus, we draw a new proposal by

$$x' \sim \frac{\mathcal{Q}_m(\cdot \mid x)}{j_m(x)}\ . \tag{2.145}$$

This shows up in the acceptance ratio as

$$A(x'|x) = \min\left(1, \frac{p(x')j_m(x')q_m(x|x')}{p(x)j_m(x)q_m(x'|x)}\right)\ . \tag{2.146}$$

2.6.4 Reversible Jump Markov Chain Monte Carlo Sampling

The algorithms, seen so far, are dedicated to problems where the distribution, from which we wish to sample, have densities with respect to a measure in some space of fixed dimension. Given a task where even the dimensionality of the problem is unknown, e.g., the number of components in a mixture of Gaussians distribution (Richardson and Green, 1997), the number of parts in a grammar based image interpretation task (Brenner and Ripperda, 2006; Ripperda, 2008), or the extraction of building footprints from digital surface models (Bredif et al., 2013), standard Metropolis-Hastings algorithm fails, as we can not evaluate the acceptance ratio due to different underlying measures. Grenander and Miller (1994) and Green (1995) introduced reversible jump MCMC or trans-dimensional MCMC to overcome this restriction.

Thus, our goal is to set up a Markov chain in a state space $\mathcal{S} = \cup_i \mathcal{S}_i$, which is the union of several sub state spaces $\mathcal{S}_i$ of potentially different dimensionality. Greens method allow the samples of the chain to jump between these different subspaces.

Instead of comparing densities within the acceptance ratio, we have to be more formal in the way that we compare distributions $P(\mathrm{d}x) = \mathbb{P}\left(\underline{x} \in \mathrm{d}x\right)$ under an appropriate measure of volume (Andrieu et al., 2003). The random variables and stochastic processes, we are dealing with, are continuous on $\mathbb{R}^d$ and its product spaces, respectively, and bounded. Thus, the measures of volume we use are Lebesgue and we assume the distribution to admit a density $P(\mathrm{d}x) = p(x)\mathrm{d}x$. Therefrom, the acceptance ratio

Algorithm 2.3: Metropolis-Hastings-Green Algorithm for rjMCMC

chose initial state $x^{(0)}$;
while $t < maxiter$ **do**
  with probability j_m propose a move m;
  // propose new sample
  $u \sim g_m \quad x' = \phi(x, u)$;
  // acceptance probability
  $A = A(x' \mid x) = \min\left(1, \frac{p(x')}{p(x)} \frac{j_m(x')}{j_m(x)} \frac{g'_m(u')}{g_m(u)} \left|\frac{\partial \phi^{-1}(x', u')}{\partial (x', u')}\right|\right)$;
  // with probability A accept the move
  $\alpha \sim \mathcal{U}(0, 1)$;
  if $A > \alpha$ **then**
    $x^{(t+1)} = x'$; ; // accept the proposed sample
  else
    $x^{(t+1)} = x^{(t)}$; ; // reject
  end
end

additionally takes into account the ratio of measures, the Radon Nikodym derivative, which leads to a Jacobian term due to the change of variables, i.e. the change of dimensionality.

Let us be more explicit. We need to map the samples of two different state spaces to a common dimension to allow comparison of their densities. Assume the chain is currently at subspace $\mathcal{S} \subset \mathbb{R}^d$ and we propose a jump to another subspace $\mathcal{S}' \subset \mathbb{R}^{d'}$. Green (1995) proposed to extend these, to what we call communicating spaces, to $\mathcal{T}' := \mathcal{S}' \times \mathcal{U}'$ and $\mathcal{T} := \mathcal{S} \times \mathcal{U}$, which are the product spaces of the original subspaces and some auxiliary spaces. $\mathcal{U}'$ and $\mathcal{U}$ are chosen such that $\mathcal{T}'$ and $\mathcal{T}$ are of same dimensionality. Further, we define a deterministic, differentiable and invertible function ϕ which maps $(x, u) \in \mathcal{T}$ to $(x', u') \in \mathcal{T}'$, thus

$$(x', u') = \phi(x, u) = \left(\phi^x(x, u), \phi^u(x, u)\right) \tag{2.147}$$

To ensure reversibility, the inverse function ϕ^{-1} is defined such that

$$(x, u) = \phi^{-1}\left(\phi(x, u)\right) \ . \tag{2.148}$$

Instead of sampling x' directly, as in original Metropolis-Hastings algorithm, we sample auxiliary variables u from a density g and obtain the new proposal by the mapping $(x', u') = \phi(x, u)$.

The probability of accepting the proposal x' being at state x, proposed by Green (1995), then is given by

$$A(x' \mid x) = \min\left(1, \frac{p(x')}{p(x)} \frac{j_m(x')}{j_m(x)} \frac{g'_m(u')}{g_m(u)} \left|\frac{\partial \phi^{-1}(x', u')}{\partial (x', u')}\right|\right) \ , \tag{2.149}$$

where we included a mixture of kernels, as introduced in Section 2.6.3. Thus, $j_m(x)$ is the probability of choosing the m-th kernel when being at x and $j_m(x')$ the probability of picking the inverse move. The last term is the Jacobian due to the change of variables. More details about the derivation of Green's ratio are given in Appendix C.3.

To be consistent with the literature we denote the fraction in (2.149) as Green's ratio $R(\cdot \mid \cdot)$, thus, $A(\cdot \mid \cdot) = \min(1, R(\cdot \mid \cdot))$.

2.6.5 Green's Birth and Death Sampler

In our application we wish to sample from a point process' distribution $F(\cdot)$. Given a point process $\underline{X}$ with points $\boldsymbol{x}_i$ in $\mathcal{S}$, density $f(\cdot)$ and intensity $\mu(\cdot)$ of a reference Poisson process, its distribution is given by $F(\mathrm{d}\mathcal{X}) = f(\mathcal{X})\, \pi_\mu(\mathrm{d}\mathcal{X})$, cf. Equation (2.112).

Algorithm 2.4: Basic Birth and Death sampler

Input: point process $\underline{X}$ with density $f(\mathcal{X})$,
intensity of the reference Poisson process $\mu(\cdot)$,
state of the Markov chain at time t $X^{(t)} = \mathcal{X}$,
probability of choosing birth $p_b = 0.5$, probability of choosing death $j_d = 0.5$

Output: X_{t+1}

`// with probability` p_b `propose birth`

$j \sim \mathcal{U}(0,1)$

if $j < j_b$ **then**

`// Birth: propose to add a new point` $\boldsymbol{x} \in \mathcal{S}$

$\boldsymbol{x} \sim \frac{\mu(\cdot)}{\mu(\mathcal{S})}$

$\mathcal{Y} = \mathcal{X} \cup \boldsymbol{x}$

$R(\mathcal{Y} \mid \mathcal{X}) = \dfrac{f(\mathcal{Y})}{f(\mathcal{X})} \dfrac{\mu(\mathcal{S})}{N(\mathcal{Y})}$ `// acceptance ratio birth`

else

`// Death: propose to delete an object`

if $X^{(t)} = \emptyset$ **then**

$R(\mathcal{Y} \mid \mathcal{X}) = 0$

else

choose $\boldsymbol{x}$ uniformly in $\mathcal{X}$

$\mathcal{Y} = \mathcal{X} \setminus \boldsymbol{x}$

$R(\mathcal{Y} \mid \mathcal{X}) = \dfrac{f(\mathcal{Y})}{f(\mathcal{X})} \dfrac{N(\mathcal{X})}{\mu(\mathcal{S})}$ `// acceptance ratio death`

end

end

$A(\mathcal{Y} \mid \mathcal{X}) = \min(1, R(\mathcal{Y} \mid \mathcal{X}))$; `// acceptance probability`

`// with probability A accept the move`

$\alpha \sim \mathcal{U}(0,1)$

if $A > \alpha$ **then**

$X^{(t+1)} = \mathcal{Y}$; `// accept the proposed sample`

else

$X^{(t+1)} = X^{(t)}$; `// reject`

end

Algorithm 2.5: Generic point process sampler

Input: state of the Markov chain at time t $X^{(t)} = \mathcal{X}$, probabilities j_m for choosing move type m
Output: X_{t+1}

with probability $j_m(\mathcal{X})$ choose proposition kernel $\mathcal{Q}_m$
// propose new object $\boldsymbol{x} \in \mathcal{S} \times \mathcal{M}$
$\boldsymbol{x} \sim \mathcal{Q}_m(\cdot \mid \mathcal{X})$
$\mathcal{Y} = \mathcal{X} \cup \boldsymbol{x}$
compute Green's ratio $R_m(\mathcal{X} \mid \mathcal{Y})$
$A(\mathcal{Y} \mid \mathcal{X}) = \min(1, R(\mathcal{Y} \mid \mathcal{X}))$; // acceptance probability
// with probability A accept the move
$\alpha \sim \mathcal{U}(0,1)$
if $A(\mathcal{Y} \mid \mathcal{X}) > \alpha$ **then**
    $X^{(t+1)} = \mathcal{Y}$; // accept the proposed sample
else
    $X^{(t+1)} = X^{(t)}$; // reject
end

For that task, Geyer and Møller (1994) proposed the so called Birth an Death algorithm, which turns out to be a special type of Green (1995) rjMCMC sampler. They built a Markov Chain using Algorithm 2.4. As proposition kernel they uses what is called birth and death. The former proposes to add a new point $\boldsymbol{x} \in \mathcal{S}$, the latter proposes to delete a randomly chosen point $\boldsymbol{x} \in \mathcal{X}$. We will give more details according to these moves in next section.

Provided $f(\cdot)$ fulfils the stability condition (2.114) page 75, Algorithm 2.4 was proven to build a Markov chain that is $F(\cdot)$ invariant, thus, simulates the point process $\underline{\mathcal{X}}$ (Geyer and Møller, 1994; Ortner et al., 2003). Furthermore, it was proven that it simulates a $F(\cdot)$ irreducible Markov chain that is recurrent and ergodic, which guarantees its convergence to its target distribution $F(\cdot)$.

This version of the sampler contains an interesting detail. If the proposed configuration does not fit the space of allowed configurations it allows us to stay in the current state. Thus, if a death is proposed also the configuration is empty, the process stays at the current state. Alternatively we might adapt the according kernel probabilities, e.g., change the probability of proposing a birth $j_b(\mathcal{X} = \emptyset) = 1$. But, to allow to stay in the current state is of great advantage to take into account any other exception that might occur, as we will see later.

2.6.6 A Marked Point Process Sampler

Algorithm 2.4 uses the birth and death kernel, both chosen with same probability. To improve the mixing properties of the Markov chain, Ortner et al. (2003) extended this algorithm and introduced additional proposition kernels. Beside the basic birth and death kernels, they proposed to use birth and death in a neighbourhood and so called non jumping transformations as translation, rotation or dilation of existing objects of the configuration.

In our work, we use two types of proposition kernels:

- dimensional jumping transformation: birth and death
- non jumping transformations: translation, dilation, switching

The latter randomly selects an object from the current configuration and randomly perturbs its marks. We also call these proposition kernels moves.

The generic point process sampler algorithm is given in Algorithm 2.5 which is equivalent to Hasting's and Green's Algorithm 2.3. Having M different moves we take the proposition kernel of the

Markov chain as a mixture of $m = 1 \dots M$ proposition distributions,

$$\mathcal{Q}(\cdot \mid \mathcal{X}) = \sum_{m=1}^{M} j_m(\mathcal{X}) \mathcal{Q}_m(\cdot \mid \mathcal{X}) \ , \tag{2.150}$$

each having a probability $j_m(\mathcal{X})$ to choose move type m being at $\mathcal{X}$. Depending on the chosen kernel we evaluate Green's ratio in line 4, formally given by

$$R_m \left(\mathcal{X} \mid \mathcal{Y} \right) = \frac{F \left(\mathrm{d}\mathcal{Y} \right) \ Q_m \left(\mathrm{d}\mathcal{X} \mid \mathcal{Y} \right)}{F \left(\mathrm{d}\mathcal{X} \right) \ Q_m \left(\mathrm{d}\mathcal{Y} \mid \mathcal{X} \right)} \tag{2.151}$$

To clarify the notion, we will denote the move types not by numbers, but by letters. We will use $m \in \{b, d, nj\}$, standing for birth, death, and non jumping moves, in the following.

2.6.6.1 Birth and Death

In case of birth, with probability j_b, we create a new object $\boldsymbol{x} \sim Q_b = \frac{\mu(\cdot)}{\mu(\mathcal{S})}$ and propose $\mathcal{Y} = \mathcal{X} \cup \boldsymbol{x}$. Green's ratio for birth takes the form

$$R_b \left(\mathcal{X} \mid \mathcal{Y} \right) = \frac{j_d \left(\mathcal{Y} \right)}{j_b \left(\mathcal{X} \right)} \frac{f \left(\mathcal{Y} \right)}{f \left(\mathcal{X} \right)} \frac{\mu(\mathcal{S})}{N \left(\mathcal{Y} \right)} \ . \tag{2.152}$$

In case of death, with probability j_d, we delete an object $\boldsymbol{x} \in \mathcal{X}$ and propose $\mathcal{Y} = \mathcal{X} \setminus \boldsymbol{x}$. Green's ratio for death takes the form

$$R_d \left(\mathcal{X} \mid \mathcal{Y} \right) = \frac{j_b \left(\mathcal{Y} \right)}{j_d \left(\mathcal{X} \right)} \frac{f \left(\mathcal{Y} \right)}{f \left(\mathcal{X} \right)} \frac{N \left(\mathcal{X} \right)}{\mu(\mathcal{S})} \tag{2.153}$$

2.6.6.2 Non Jumping Transformations

With this type of moves we randomly perturb the marks of an existing object. To do so we uniformly select an object $\boldsymbol{x} \in \mathcal{X}$ and throw random numbers $u \sim Z_{(\mathcal{X},u)}$ according a suitable distribution. Given a function $h : \boldsymbol{y} = h(\boldsymbol{x}, u)$ which transforms the object, we propose $\mathcal{Y} = \mathcal{X} \setminus \boldsymbol{x} \cup \boldsymbol{y}$. We will detail these moves in Section 5.3.

We construct the transformation h such that the inverse transformation $\boldsymbol{x} = h^{-1}(\boldsymbol{y}, \boldsymbol{u})$ exist and is as possible as h is, which ensures reversibility. Thus, if the translation and its inverse are equally probable and the according perturbation variables where thrown from the same distribution, Green's ratio simplifies to

$$R_{nj} \left(\mathcal{X} \mid \mathcal{Y} \right) = \frac{f \left(\mathcal{Y} \right)}{f \left(\mathcal{X} \right)} \ , \tag{2.154}$$

which was shown by Ortner et al. (2003). Details about the derivation of Green's ratios for Marked point process sampling are given in Appendix C.3.1.

2.6.7 Simulated Annealing

The algorithm introduced in preceding sections ensure sampling from a probability density $p(x)$ or $f(\boldsymbol{x})$, respectively. Having an optimisation task, we are often interested in the global maximum of $p(x)$ and not the full density, i.e.

$$\widehat{x} = \underset{x^{(t)}}{\operatorname{argmax}} \ p\left(x^{(t)}\right) \tag{2.155}$$

A naive procedure would be to draw a large number of samples from $p(x)$ and to take the value which maximises p, which is obviously inefficient, especially when $p(x)$ is complex or of high dimensionality.

The idea of simulated annealing (Kirkpatrick et al., 1983; Geman and Geman, 1984) is to narrow the target density during time and to simulate a non-homogeneous Markov chain

$$p^{(t)}(x) \propto p^{(1/T_t)}(x) \tag{2.156}$$

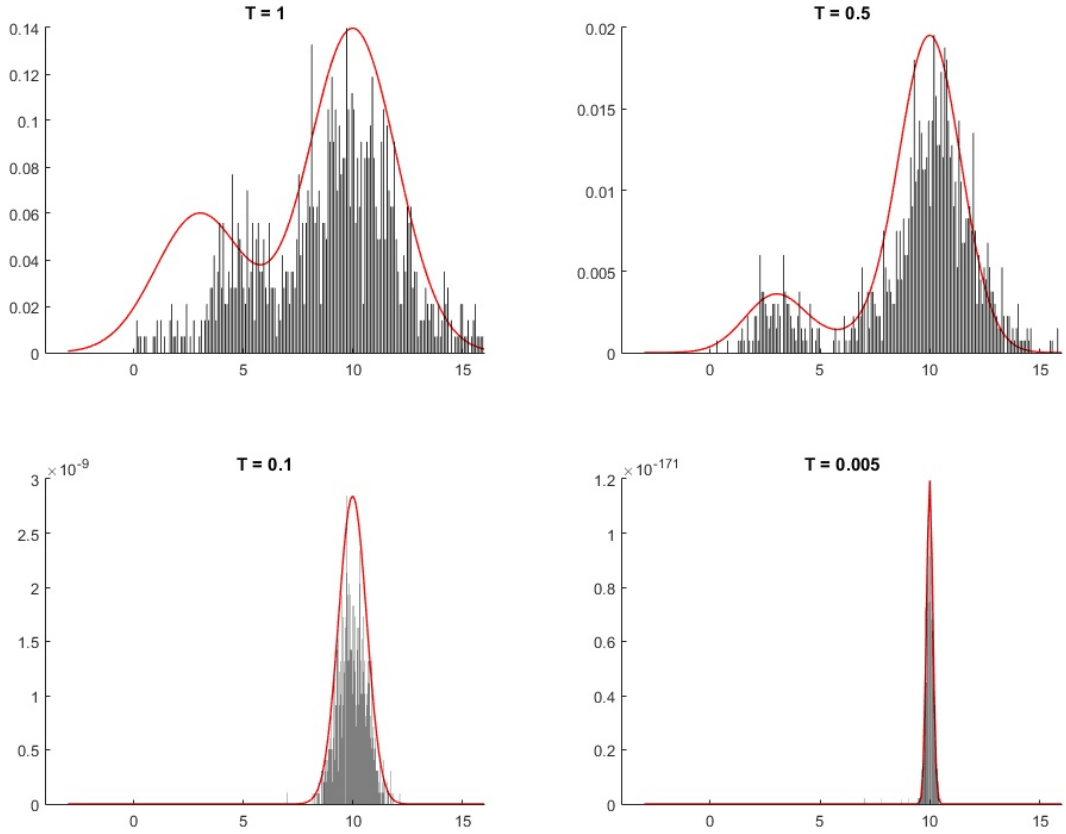


Figure 2.19: Sampling with Metropolis-Hastings and simulated annealing where we sample from $p^{1/T}$. **Top left:** samples from the original density p. **Bottom right:** samples from $p^{1/T}$ with $T = 0.005$ which are concentrated at the position where p has its maximum.

where T_t is what we call the temperature parameter and cools down the process such that

$$\lim_{t\to\infty} T_t = 0 \ . \tag{2.157}$$

We assume the chain to be aperiodic and recurrent. If the process cools down slowly, in theory logarithmic, it can be shown that $p^{\infty}(x)$ is a probability density which concentrates all of its mass at the global maximum of $p(x)$ (Andrieu et al., 2003).

The concept of simulated annealing goes back to an analogy to statistical mechanics systems. We interpret samples of a stochastic process as atoms of a physical system. There, the reduction of temperature isolates low energy states, the annealing. In other words, while the atoms are highly movable at high temperature they become less movable at low temperature, and finally, concentrate at the state of lowest energy. In our case we want to create the Markov chain, such that the samples explore the whole state space in the beginning and to concentrate around the global optimum during time, more and more. In Section 2.5.1.4 and 2.5.4, we have already seen the energy point of view in terms of Gibbs processes and Gibbs distribution. We note the equivalence of finding the global maximum of a pdf to finding the global minimum of an energy.

An one-dimensional example is given in Figure 2.19. For different temperatures we draw 1000 samples from the pdf, shown as red line in Figure 2.19 top left, using the algorithm given in Algorithm 2.6. For each plot we vary the parameter T, as given in the heading of each plot, to sample from $p^{1/T}$. In the beginning the samples explore the whole range of p, as expected. But, the lower T the more the samples concentrate at the point where the original pdf has its global optimum. Please note that the densities are not normalised, they indeed visualise $p^{1/T}$. The histograms of samples are normalised such that they fit the plot of $p^{1/T}$, for visualisation purposes.

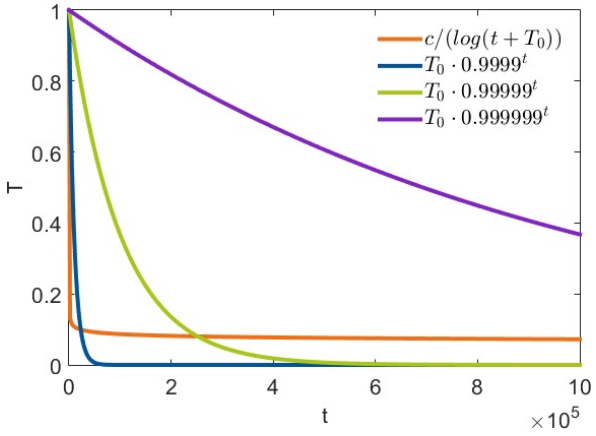


Figure 2.20: Different cooling schedules for simulated annealing. $T_0 = c = 1$

In theory convergence is ensured for cooling schedules such that

$$T(t) \geq \frac{c}{\ln(t + T_0)} \tag{2.158}$$

where t is the time index of the Markov chain, T_0 the temperature we start with and c a constant. The latter two parameters are problem dependent. In practice this cooling schedule is too slow to reach a temperature near 0 in feasible time. Therefore, different schedules where proposed which are suited to reach low temperatures in finite time, but take into account the risk of getting stuck into local optima. The most common cooling scheme, which we found in our context, is a geometric cooling

$$T(t) = T_0 \cdot \alpha^t \tag{2.159}$$

for $\alpha < 1$ e.g., $\alpha = 0.9999 \ldots 0.999999$. Figure 2.20 shows different cooling schedules by comparison. In orange we see the logarithmic scheme which guaranties convergence to the global optimum. The temperature goes down fast, but converges towards zero slow. In contrast the geometric schedules decrease in the beginning slower, while converging to zero fast, depending on α. In practice it is demanding to find proper parameters T_0 and α to reach zero temperature in finite time without getting stuck in local minima.

The algorithm for optimisation using MCMC in combination with simulated annealing is given in Algorithm 2.6. The only difference to common Metropolis-Hastings algorithm is the evaluation of the acceptance ratio at line 5, which depends on the temperature parameter. Here we wrote the most general form of the Metropolis-Hastings acceptance ratio, which we easily may adapt to mixtures of kernels and Green's ratio.

Algorithm 2.6: Optimisation with Metropolis-Hastings algorithm and simulated annealing

chose initial state $x^{(0)}$ and initial temperature T_0;
while $t <$*maxiter* **do**
    `// propose new sample`
    $x' \sim q(x'|x^{(t)})$;
    set T_t according the cooling schedule;
    `// acceptance probability`
    $A = A(x'|x) = \min\left(1, \frac{p^{1/T_t}(x')q(x|x')}{p^{1/T_t}(x)q(x'|x)}\right)$;
    `// with probability` A `accept the move`
    $\alpha \sim \mathcal{U}(0,1)$;
    if $A > \alpha$ **then**
        $x^{(t+1)} = x'$; `// accept the proposed sample`
    else
        $x^{(t+1)} = x^{(t)}$; `// reject`
    end
end

CHAPTER 3

Detecting Line, Circle and Ellipse Segments in Images

Given pixel-chains from edge detection, this chapter proposes a method to segment them into straight line, circular arc and ellipse segments. Found line segments should serve as low-level features for further image interpretation, as we will show in the subsequent chapters. To do so, we propose an adaption of Douglas-Peucker's polyline simplification algorithm using circle segments instead of straight line segments. It is robust and decreases the complexity of given polylines better than the original algorithm. We call this line simplification algorithm `circlePeucker`. In a second step, we further simplify the poly-curve by merging neighbouring segments to straight line, circular arc and ellipse segments. Merging is based on the evaluation of the minimal description length. The whole procedure is called detecting line, circle, and ellipse segments (DLCE).

3.1 Introduction of the DLCE algorithm . . . 92
3.2 Circle Peucker . . . 93
3.3 Merging Line Primitives . . . 94
3.4 Related Work . . . 96
3.5 Experiments . . . 97
3.5.1 Extracting Pixel-Chains from Images . . . 97
3.5.2 Parameter Setting . . . 98
3.5.3 Synthetic Data . . . 98
3.5.4 Evaluating the Effect of Pre-segmentation Methods . . . 100
3.5.5 Comparison to ELSD . . . 103
3.5.6 Using DLCE for Feature Extraction on Facade Images . . . 103
3.6 Summary . . . 103

3.1 Introduction of the DLCE algorithm

Polyline simplification is interesting from several points of view. First, in terms of compact description of spatial data, e.g., in the context of image description. Second, in terms of generalisation, e.g., in the context of cartography or resolution dependent visualisation of polylines.

On the other, hand finding circular and elliptical structures in images is relevant in terms of compact image description and further image interpretation. Most image interpretation systems which use bottom-up image features, thus, not pure pixel information only, are based on key point or edge detection. Directly identifying circular and elliptical structure gives rise to much more informative image features from bottom-up (Chia et al., 2012; Jurie and Schmid, 2004).

We aim at the extraction of image features from bottom-up which are more informative than using intensity values solely of single pixels or even line segments. For this, we propose a two-step polyline simplification algorithm that approximates a given set of ordered points in 2D by a sequence of straight line, circular arc, and ellipse segments. Although, the algorithm is applicable to any kind of ordered 2D points, we assume pixel chains within images, see Figure 3.1. The first intuition behind our approach is that arbitrary smooth curves can be locally characterised by an osculating circle. We will use this, in the first step of the algorithm, where we simplify the given set of points by a sequence of circle segments.

But, due to perspective distortions, in general there will be almost no circles in images. All circles in object space are projected to ellipses in image space, ellipses in object space are projected to ellipses. Only in rare cases, the images of 3D circles or 3D ellipses are mapped to hyperbola, namely in case they partially are behind the camera. The situation is different if the circles are sitting in a set of parallel planes and the viewing direction, intentionally, has been taken orthogonal to these planes, or the image has been rectified to mimic this situation. Then, almost no ellipses will occur in the images, and the proposed method can directly be transferred by replacing ellipses by circles. However, we will still observe circular arc segments in images. Every time, a projected arc segment is very short or very flat it will be locally approximate by an circular arc segment.

Therefore, eventually, the pixel-chain is represented by a sequence of straight line, circular arc and ellipse segments. This way, we are more flexible representing curved lines.

The proposed method consists of two steps, see Figure 3.1. Given the pixel-chains within the image, we first iteratively segment the region boundary into circular segments. This yields an over-segmentation due to the non existence of real circle segments. Second, we merge neighbouring segments to straight line, circular arc, and ellipse segments, based on model selection. This step optimally estimates lines, circular arcs and ellipses in a least squares sense.

One might argue, why not directly segment a pixel-chain into ellipses, but first look for circle segments, and then group them to ellipses. There are two main reasons for the two-step procedure:

1. The slope, curvature or curvature change functions of the ellipse are no simple functions, which allow to identify elliptical segments, as this is the case for straight lines (constant slope) and

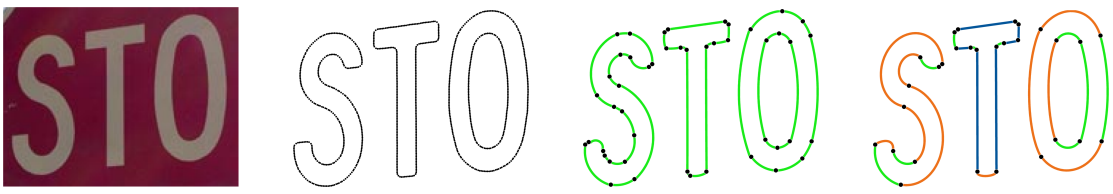


Figure 3.1: Overview of the DLCE procedure to detect segments of straight lines, circular and elliptical arcs. **From left to right:** input image, pixel-chains, segmentation into circular arcs using `circlePeucker`, final result after merging. **Colours:** Orange - ellipse segment. Green - circle segment. Blue - straight line segment

Algorithm 3.1: function `circlePeucker`. The algorithms recursively splits the chain $\mathcal{X}$ until the largest distance of a point to the corresponding arc is below a pre-set threshold t.

In: Ordered set of points $\mathcal{X} = \{\chi_1 \dots \chi_N\}$, indices i_a and i_b of start- and endpoint of the current segment, tolerance t
Out: List of nodes $\mathcal{O}$

```
// the current segment
```
$\mathcal{X}_s = \{\chi_{i_a} \dots \chi_{i_b}\}$;
if $|\mathcal{X}_s| = 2$ **then** $\mathcal{O} = \{i_a, i_b\}$,**return**;
```
// Fit an arc to the current segment
```
$\mathcal{S} =$ `fitArc`$(\mathcal{X}_s)$;
```
// Evaluate distances of all points to the arc
```
$(d_{\max}, i_s) =$ `distXS`$(\mathcal{X}_s, \mathcal{S})$;
```
// Compare greatest distance to given tolerance
```
if $d_{\max} > t$ **then**
```
    // Recursive call of PeuckerCircle algorithm
```
    $\mathcal{O}_1 =$ `circlePeucker`$(\mathcal{X}, i_a, i_s, t)$;
    $\mathcal{O}_2 =$ `circlePeucker`$(\mathcal{X}, i_s, i_b, t)$;
    $\mathcal{O} = \mathcal{O}_1 \cup \mathcal{O}_2$;
else
```
    // This is an arc
```
    $\mathcal{O} = \{i_a, i_b\}$;
end
return

circles (constant curvature),

2. there is no simple local measure telling whether a local segment belongs to an ellipse or not: Analysing the curvature, distinguishes circles and straight lines. We would need the second derivatives of the curvature to capture the properties of a local ellipse element, as an ellipse has two more degrees of freedom, than a circle. But, determining fourth derivatives is unstable.

The reminder of this chapter is as follows. Next, we present our adaption of the Douglas-Peucker Algorithm by using circles instead of lines, which we call `circlePeucker`. Section 3.3 describes the merging of neighbouring objects, whenever they are supposed to join the same geometric object. We review related approaches in Section 3.4, evaluate the proposed procedure in Section 3.5, and summarize the chapter in Section 3.6.

3.2 Segmenting Point Sequences into Circle Segments

Our concept of region boundary segmentation is based on the well known Douglas-Peucker algorithm (Douglas and Peucker, 1973). This algorithm is designed to simplify polylines. Therefor, it recursively splits the sequence of polyline edges into larger edges until the distance of an eliminated point to the corresponding edge is below a threshold t.

We adapt the original algorithm, such that it uses circle segments instead of straight lines. Given an ordered sequence of points $\mathcal{X} = \{\chi_n\}, n = 1 \dots N$, we recursively partition the sequence into segments which approximate the according points by a circular arc up to a pre-specified tolerance t. If applicable, a segment is split at that point χ_n, where the distance to the circular arc is maximum. In order to enforce continuity, we fix the start and end point of the segments and determine the best fitting arc, as shown in Section 2.1.4.2.

The algorithm for approximating a polyline by a sequence of circles, called `circlePeucker`, is given in Algorithm 3.1. It uses

1. a function `fitArc`($\mathcal{X}$) for fitting a circular arc segment $\mathcal{S}$ to a given set of points $\mathcal{X}$, constraining it to the start and end point, as described in Section 2.1.4.2, and

2. a function `distXS`($\mathcal{X}, \mathcal{S}$) for determining the index i_s and the distance $d_{\max}$ of that point $\mathcal{X}_{i_s}$, out of $\mathcal{X}$, with the largest distance to the arc segment $\mathcal{S}$, which is given in (2.56) in Section 2.1.4.2.

We call the function by $\mathcal{O}$ =`circlePeucker`($\mathcal{X}, 1, N, t$). As result we obtain a list $\mathcal{O}$ of indices which represent the end points of sought segments. This yields the required partitioning of the original point sequence.

3.3 Merging Line Primitives Based on Model Selection

Given the pre-segmentation $\mathcal{O}$ of Section 3.2, which is assumed to be over-segmented, we aim at a simplification by merging neighbouring segments which share the same model.

Deciding whether two neighbouring segments belong to the same model may be based on a statistical hypothesis test. As hypothesis tests aim at rejecting the null hypothesis, they can be used as sieve for keeping false hypothesis. Merging segments merely based on hypothesis testing, however, fails due to the risk of accepting large changes in geometry, in case the parameters of the proposed model are uncertain.

On the other hand, deciding which model fits the data best, i.e. whether a curved line is best approximated by a straight line, circle or an ellipse, is a typical model selection problem, which we introduced in Section 2.3.

Our pre-segmentation is based on circle segments, but in general there are almost no circles in natural images, as they suffer from perspective distortions. Thus, our final segmentation is meant to consist of straight line, circle and ellipse segments. Despite the argumentation before, we explicitly include circles to consider very short or flat ellipse segments.

From the pre-segmentation we only take the information which points belong to the same segment and ignore the parameters of the fitted circle segments. The final representation is achieved by fitting straight line, circle and ellipse segments through neighboured segments and single segments, respectively, using all points belonging to them. This is different from Rosin and West (1995), who only use the endpoints from the pre-segmentation.

In the following, we are interested in the optimal estimates and fit lines $\mathbf{l}$, using the approach given in Section B.1.4, circles $\boldsymbol{p}$ using the approach given in Section 2.1.4.1, and ellipses C using the approach given in Section 2.1.3.3. For each model, we estimate the weighted sum of squared residuals $\Omega = \sum_n v_n^2/\sigma^2$ as measure of precision.

Let us assume a segmentation $\mathcal{O} = \{o_m\}$ of points $\mathcal{X} = \{\mathcal{X}_m\}$ into M segments. We call the current parameter vector of the m-th segment $\boldsymbol{\theta}_m$. Thus, $\boldsymbol{\theta}_m$ acts as placeholder for $\mathbf{l}_m$, $\boldsymbol{p}_m$ or C_m and includes the number U_m of parameters needed to define the current model, which is our measure of complexity.

Initially, we select the best model for each segment by minimising its description length in terms of the (Schwarz's) Bayesian Information Criterion (BIC), cf. Section 2.3.

$$\widehat{\boldsymbol{\theta}}_m = \underset{\boldsymbol{\theta}_m}{\operatorname{argmin}} \operatorname{BIC}(\mathcal{X}_m, \boldsymbol{\theta}_m) \tag{3.1}$$

$$= \underset{\boldsymbol{\theta}_m}{\operatorname{argmin}} \Omega(\mathcal{X}_m, \boldsymbol{\theta}_m) + U_m \ln N_m \,. \tag{3.2}$$

We aim at merging neighbouring segments by evaluating the gain of description length, in terms of BIC, when fitting a new model to the joined set of points.

Assume, we already found models $\widehat{\boldsymbol{\theta}}_m$ and $\widehat{\boldsymbol{\theta}}_{m+1}$ using the points $\mathcal{X}_m$ of segment m and $\mathcal{X}_{m+1}$ of segment $m+1$, respectively. We propose the points of both segments to belong to a joined segment, thus, $\mathcal{X}_{m,m+1} = \mathcal{X}_m \cup \mathcal{X}_{m+1}$. Again, we select the best model, for this potentially merged segment, by

Algorithm 3.2: function `mergeSegments`. The algorithms merges neighbouring segments in a greedy procedure until no more merging proposals are left.

In: Initial segmentation $\mathcal{X} = \{\mathcal{X}_m\}$
Out: Final segmentation $\widetilde{\mathcal{X}}$

$\widetilde{\mathcal{X}} = \mathcal{X}$;
foreach $\mathcal{X}_m$ **do**
  `// estimate best model, cf. Equation (3.2)`
  $\widehat{\boldsymbol{\theta}}_m = \underset{\theta_m}{\operatorname{argmin}}\ \mathrm{BIC}\left(\mathcal{X}_m, \theta_m\right)$;
end
foreach $\mathcal{X}_{m,m+1} = \mathcal{X}_m \cup \mathcal{X}_{m+1}$ **do**
  `// estimate best joint model, cf. Equation (3.3)`
  $\widehat{\boldsymbol{\theta}}_{m,m+1} = \underset{\boldsymbol{\theta}_{m,m+1}}{\operatorname{argmin}}\ \mathrm{BIC}\left(\mathcal{X}_{m,m+1}, \boldsymbol{\theta}_{m,m+1}\right)$;
end
`// Start merging`
merge = true;
while *merge* **do**
  foreach $\mathcal{X}_{m,m+1} = \mathcal{X}_m \cup \mathcal{X}_{m+1}$ **do**
    `// evaluate gain of description length, cf. Equation (3.4)`
    $\Delta\mathrm{BIC}_{m,m+1} = \mathrm{BIC}\left(\mathcal{X}_{m,m+1}, \widehat{\boldsymbol{\theta}}_{m,m+1}\right) - \left(\mathrm{BIC}\left(\mathcal{X}_m, \widehat{\boldsymbol{\theta}}_m\right) + \mathrm{BIC}\left(\mathcal{X}_{m+1}, \widehat{\boldsymbol{\theta}}_{m+1}\right)\right)$;
  end
  `// get best proposal for merging`
  $\widehat{m} = \underset{m}{\operatorname{argmax}}\ \Delta\mathrm{BIC}_{m,m+1}$;
  if $\Delta BIC_{\widehat{m},\widehat{m}+1} > 0$ **then**
    `// merge`
    $\widetilde{\mathcal{X}} = \{\mathcal{X}_1, \ldots, \mathcal{X}_{\widehat{m}} \cup \mathcal{X}_{\widehat{m}+1}, \ldots \mathcal{X}_M\}$;
    $\widehat{\boldsymbol{\theta}}_{\widehat{m}} = \widehat{\boldsymbol{\theta}}_{\widehat{m},\widehat{m}+1}$;
  else
    `// nothing left to merge`
    merge = false;
  end
end

minimising the BIC

$$\widehat{\boldsymbol{\theta}}_{m,m+1} = \underset{\boldsymbol{\theta}_{m,m+1}}{\operatorname{argmin}}\ \mathrm{BIC}\left(\mathcal{X}_{m,m+1}, \boldsymbol{\theta}_{m,m+1}\right) \ . \tag{3.3}$$

The gain of description length is given by the difference of the joint description length, using the models $\widehat{\boldsymbol{\theta}}_m$ and $\widehat{\boldsymbol{\theta}}_{m+1}$ separately, and the description length when using the merged segments.

$$\begin{aligned} \Delta\mathrm{BIC}_{m,m+1} &= \mathrm{BIC}\left(\mathcal{X}_{m,m+1}, \widehat{\boldsymbol{\theta}}_{m,m+1}\right) - \left(\mathrm{BIC}\left(\mathcal{X}_m, \widehat{\boldsymbol{\theta}}_m\right) + \mathrm{BIC}\left(\mathcal{X}_{m+1}, \widehat{\boldsymbol{\theta}}_{m+1}\right)\right) \\ &= \Omega_{m,m+1} - (\Omega_m + \Omega_{m+1}) \\ &\quad + U_{m,m+1} \ln(N_m + N_{m+1}) - (U_m \ln N_m + U_{m+1} \ln N_{m+1}) \end{aligned} \tag{3.4}$$

If $\Delta\mathrm{BIC}_{m,m+1} > 0$ the description length of the merged segment is shorter than the description length of two separate segments; thus, they should be merged to reduce the overall complexity.

To evaluate the whole set of segments, we proceed in a greedy manner, cf. Algorithm 3.2. After initialising all segments by their best models, in terms of description length, we propose all neighbouring segments to be merged and select the according best model. From all neighboured pairs which have a

positive gain of description length, we select the one with the highest gain. We update the segmentation, such that one break point is removed. We iterate this process until there are no more merging proposals with positive gain of description length.

3.4 Related Work

To the best of our knowledge, there is no work about an adaption of Douglas-Peucker's algorithm to the use of circles, instead of lines, as basic elements. However, proposals exist to simplify polygons by sets of circular arcs for the efficient storage of polylines.

Günther and Wong (1990) propose, what they call Arc Tree, and which represents arbitrary curved shapes, in a hierarchical data structure, with small curved segments at the leaves of a balanced binary tree. Moore et al. (2003) propose a method for polygon simplification using circles. They aim at closed polygons given by a set of 2D points. Based on medial axis from Voronoi polygons, they propose a population of circles which they afterwards filter to get a set of circles which best approximate the given polygon. The final representation of the polygon consists of circles represented by centre and radius and tangents which link neighbouring circles. No work on using ellipses for improving storage requirements are known to us.

Finding ellipses in images has attracted many researchers. Some of them use Hough-transform methods which tend to be slow. Most techniques start from pixel-chains. Early works focussed on ellipse fitting, e.g., Pavlidis (1983); Porrill (1990), later focussed on unbiased estimates, e.g., Wu (2008); Libuda et al. (2006). We are interested in the more general problem of describing the pixel-chains by sequences of straight line, circle, and ellipse segments, a problem already addressed in Albano (1974), however, neither enforcing ellipses, nor looking for a best estimate for ellipses.

West and Rosin (1992) and Rosin and West (1995) perform a segmentation of point sequences into lines and ellipses in a multistage process. They first segment a 2D-curve into straight lines. Afterwards sequences of line segments are segmented into arcs restricted to their endpoints. We might interpret this step as merging sequences of lines to elliptical arcs. Model selection is done implicitly by evaluating a significance measure to each proposed segment, which is based on its geometry, purely. However, their criteria are non-statistical, thus, cannot easily be adapted to varying noise situations.

Ji and Haralick (1999) criticise this and propose a statistically valid criterium. Starting from Rosin's output of arc segmentation, they merge pairs of arcs belonging to the same ellipse. Moreover, they also group non-adjacent arcs and exploit the sign of the arcs for grouping. Proposals for merging are validated via hypothesis testing. They showed only few results on comparably easy images.

Nguyen and Kerautret (2011) also address the segmentation of pixel-chains into lines and ellipses. It is based on a discrete representation of tangents, circles, and an algebraic fitting through neighbouring arcs, using some key points (boundary and midpoint) solely, instead of the complete pixel-chain.

Patraucean et al. (2012) propose a parameterless line segment and elliptical arc detector, called elliptical line segment detector (ELSD). They use an ellipse fitting algorithm which uses both the algebraic distance of the conic equation and deviation from the gradient direction. Their model selection aims at avoiding false negatives, by controlling the number of false positives. Furthermore, their method adapts to noise. Their validation and model selection criteria, however, are based on fixed tolerance bands. Also, they do not enforce any continuity between neighbouring segments. Nevertheless, their algorithm yields state-of-the-art results and outperforms other existing approaches. They provide software as well as an online demo to process own images. We use this to compare our results to theirs.

An entirely different direction is the detection of circular and elliptical objects from stereo or even multi-view images. Recently, Soheilian and Brédif (2014) propose a method to detect 3D circular targets from multi-view images. In each image, they estimate full ellipses from given image points which are already supposed to belong to the ellipse. These ellipses are taken as observations both to reconstruct the 3D circle in object space and to estimate the poses of all images. The estimation is done by a Gauss-Helmert model. All observations and unknowns are adjusted by error propagation; thus, they use all available informations and yield a statistical optimal fit.

The aim of this work is different compared to ours. While they use full ellipses in images as observation to estimate circles in 3D, we aim at detecting ellipse segments, among others, within images. They assume full ellipses as observation and do not show experiments on ellipse segments. Our proposed line segmentation could replace their first step and serve as input for estimating the orientation of images using lines, circular arcs and ellipses.

3.5 Experiments

This section presents our experiments to evaluate the proposed algorithm. The goal is to show that DLCE successfully segments given pixel-chains into sequences of our line, circular, and elliptical arcs, while preserving its geometry and reduces the number of elements, compared to standard algorithms. We will show that the pre-segmentation using `circlePeucker` yields significant benefit compared to the classical Peucker algorithm and compare the whole procedure with an existing approach. Further, we give details about parameters, show the resulting segments and discuss the success of the merging step by means of some statistics. We compete with ELSD (Patraucean et al., 2012), as this is state-of-the-art and there exists code, as well as an online demo, to process own images using fixed parameters.

3.5.1 Extracting Pixel-Chains from Images

The presented algorithm is suited for every kind of point sequences. We only require the sequence of the points is given. Throughout this work, we are dealing with images, whereas, we require pixel-chains as input data. A pixel-chain is an ordered list of pixel coordinates which represent an edge or line within the image, not necessarily a closed boundary.

For the extraction of edges from images a huge amount of algorithm exist. One of the most widespread is the Canny edge detector (Canny, 1986), which is based on the spatially averaged second moment matrix. The eigenspectrum of this matrix provides information about the local image content, which is used to extract pixels which have a strong gradient in one direction.

More recently, Martin et al. (2004) presented the Berkley edge detector, where edge detection is formulated as a supervised learning problem. They combine local features based on brightness, color, and texture, associated with natural boundaries and train a classifier to identify boundaries. The training of the classifier is based on a large data set of manually labelled natural images. The output of their classifier are the posterior probabilities of each image location and orientation for being part of a boundary. The work of Martin et al. (2004) gives an excellent overview of existing edge extraction methods.

Almost all edge extraction procedures have in common the output of a labelled image, either in form of a binary image which states whether a pixel belongs to an edge or not, or in form of an intensity image, where intensities represent any kind of confidence of being boundary or not. Thus, we need a post-processing step to extract pixel-chains out of them anyway.

To generate our input data, we use the feature extraction procedure, as described in Förstner (1994) and Fuchs and Förstner (1995) called `fop`. We decided to use this approach for several reasons

- In contrast to many other procedures, it delivers region boundaries as well as thin lines.
- We get edges in the form of chains of points, immediately.
- We get them with sub-pixel coordinates.

`fop` uses the eigenspectrum of the spatially averaged second moment matrix. It includes an automatic noise estimation and an edge preserving filter as described in Förstner (2000).

There are two parameters which we adapt to our purposes. First, we have the natural scale σ_1 which is the differentiation scale and influences the noise suppression. Before differentiation, the image is smoothed with a Gaussian kernel with standard deviation σ_1 and width $4\sigma_1^2$. Second, there is the

artificial scale σ_2 which is the integration scale, therefore, defines the width of the integration, when averaging the squared gradients, and tells the maximum width of lines to be detected.

The image noise σ_n is used for non-minimum suppression, when identifying edge locations, and usually is estimated automatically. For evaluating the effect of noise on DLCE, we use synthetic images, where we predefine the noise for generating the images as well for extraction the edges. Thus, in case of synthetic data, we switch off the automatic noise estimation, to be independent of its result.

3.5.2 Parameter Setting

For using DLCE, there are only a few parameters to choose, and these have a clear meaning. Depending on the application they are intuitively adapted for the according objective.

First, we need to set the parameters for the edge extraction using `fop`. For finding fine details, we use $\sigma_1 = 0.7$ pixel for the differentiation and $\sigma_2 = 1.0$ pixel for the integration scale, throughout all experiments. In case of synthetic images we set the noise level manually. For all other images, we use the automatic noise estimation procedure implemented in `fop`.

Using the standard deviation of edge pixels σ_e, we set the tolerance t for the pre-segmentation $t = 3 \cdot \sigma_e$, for both `circlePeucker` and for Douglas-Peucker. For the merging step, thus, for fitting lines, circles, and ellipses, we assume the uncertainty of each pixel to be isotropic $\boldsymbol{\Sigma}_{ll} = \sigma_e^2 \boldsymbol{I}_2$.

From experiments we found the standard deviation of edge pixels $\sigma_e = 0.1$ pixel. To insist on pure image statistics, we should use this value. Instead, we use this value as parameter for generalisation. The higher σ_e, the more we allow the individual edge points to deviate from fitted geometric entities, the more we generalise. If not stated otherwise, we chose $\sigma_e = 0.2$ pixel for all experiments.

As our purpose, in this chapter, is the segmentation of given pixel-chains, and not the interpretation of the image, the identification of spurious scatter is out of scope. Our algorithm works stable even for small chains. Nevertheless, to simplify the visualisation, we do not show short pixel-chains, say shorter than 10 or 20 pixels, depending on the structure of the image.

For all results on DLCE tables D.1, D.2 and D.3 show the individual parameter settings.

3.5.3 Synthetic Data

First, we investigate the noise sensitivity of the procedure using synthetic images, see Figure 3.2. More results on synthetic images are shown in Appendix D.1 in Figures D.1 and D.2. We synthesise 8 bit grayvalues in the range $[0, 255]$. For the sample image used in Figure 3.2, the homogeneous regions are of intensities $40, 100, 120$ and 190 from dark ellipse to light background.

We start with synthetic generated noise free images, Figure 3.2 top row, and stepwise increase the noise level to $10, 20, 30, 40$ [gr]. To be independent of the automatic noise estimation of `fop`, we set the noise level for extracting the pixel-chains manually. Edge extraction using `fop` fails when assuming no noise; therefore, we assume a fictive image noise of $\sigma_n(0) = 3$ [gr] for the first image.

Further, when changing the noise of an image, from $\sigma_n(0)$ to $\sigma_n(k)$, we must adapt the standard deviation of edge pixels σ_e. The relation between image noise and standard deviation of edge pixels is not trivial due to the complex procedure of edge extraction. Thus, instead of variance propagation, we use the estimated variance factors, of estimated line segments, to derive this relation empirically. This way, we found a linear relation,

$$\sigma_e(\sigma_n(k)) = \sigma_n(0) + \frac{1}{100}\sigma_n(k)\,. \tag{3.5}$$

Please note that we assume the image noise to be given as 8 bit grayvalues, too. Otherwise, the factor would change. Other parameters of edge detection are set as defined in Section 3.5.2.

Figure 3.2 shows the final results of DLCE for each noise level, compared to those of ELSD. Please note that the proposed algorithm works stable up to a certain degree of noise. As long as the contrast

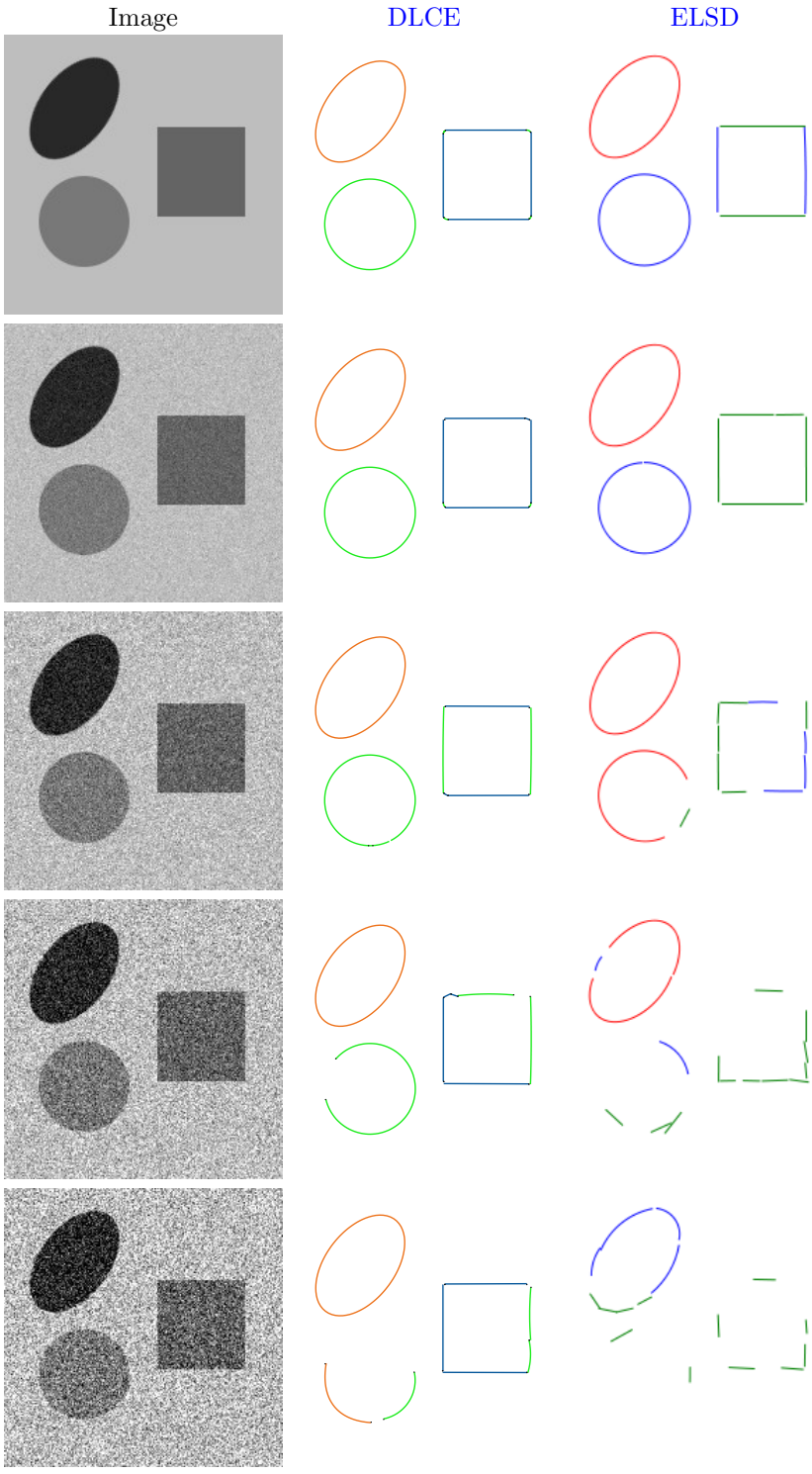


Figure 3.2: Results of DLCE on synthetic data under different noise conditions and comparison to ELSD. Intensities are $40, 100, 120$ and 190 from dark ellipse to light background. We increase the noise level **from top to bottom** $\sigma_n = 0, 10, 20, 30, 40$ grayvalues. **Colours DLCE:** Orange - ellipse segment. Green - circle segment. Blue - straight line segment. **Colours ELSD:** Red - ellipse segment. Blue - circle segment. Green - straight line segment. Parameter setting and number of segments after pre-segmentation and merging, respectively, are given in Table D.1, page 188.

is high, geometric elements are reliably and accurately detected. From visual inspection, our results outperform those of ELSD.

For images with low noise, we realise the effect of rounding the corners. Due to differentiation and integration over small image regions, we do not yield pixel-chains which perfectly restore the angles of corners. Due to the tight setting of parameters for edge extraction, we try to reduce this effect as much as possible. Other edge extraction methods, as Canny, show the same effect. As we increase the expected noise of edges, while varying the image noise, the effect vanishes for higher noise levels.

3.5.4 Evaluating the Effect of Pre-segmentation Methods

Next, we investigate the usage of our new line simplification method `circlePeucker` as pre-segmentation before the merging step.

We first perform the pre-segmentation, using both Douglas-Peucker and `circlePeucker`, and perform merging afterwards on the according results. We compare the results qualitatively by visual inspection and quantitatively by counting the number of segments after each step of the procedure. Figures 3.3 and 3.4 show the effects on synthetic and natural images, respectively. Tables D.1 and D.2 report for each individual experiment the setting of parameters and characteristic numbers, as number of line segments, and running time. We summarise the effect of using `circlePeucker` instead of Douglas-Peucker as follows:

- Depending on the objects structure, the number of segments after pre-segmentation is lower by a factor of 1.5 to 3.
- The final number of line segments, after merging, is still lower up to a factor of 1.5 to 2.
- Qualitatively, the results are more reliable and more elliptical segments are detected.
- The run time for pre-segmentation, using `circlePeucker,` is much higher compared to Douglas-Peucker, up to a factor of 35. But, due to the reduced number of segments which we need to take into account for merging, the running time for merging is significantly reduced, when pre-segmenting by `circlePeucker`. Depending on the complexity of the image, at the end, the overall running time is almost equal or slightly worse when using `circlePeucker`. This is acceptable with respect to the quality of results.

To be more precise, we show some of the advantages of pre-segmentation using `circlePeucker` by some details. For example, see Figure 3.4 c_2 and f_2, the capital O of the STOP-sign actually consist of four arcs instead of one ellipse. When `circlePeucker`, Figure 3.4 f_2, we obtain this result exactly. While using Douglas-Peucker tends to approximate arcs by lines, obviously, which we see at the slope of the 'S' in Figure 3.4 e_2, for example. The main reason for this is the identification of break points candidates when evaluating the pre-segmentation. Obviously, `circlePeucker` identifies points of changing curvature more likely than Douglas-Peucker. The same effect can be observed for the boundary lines around the worm, its eye and the O's.

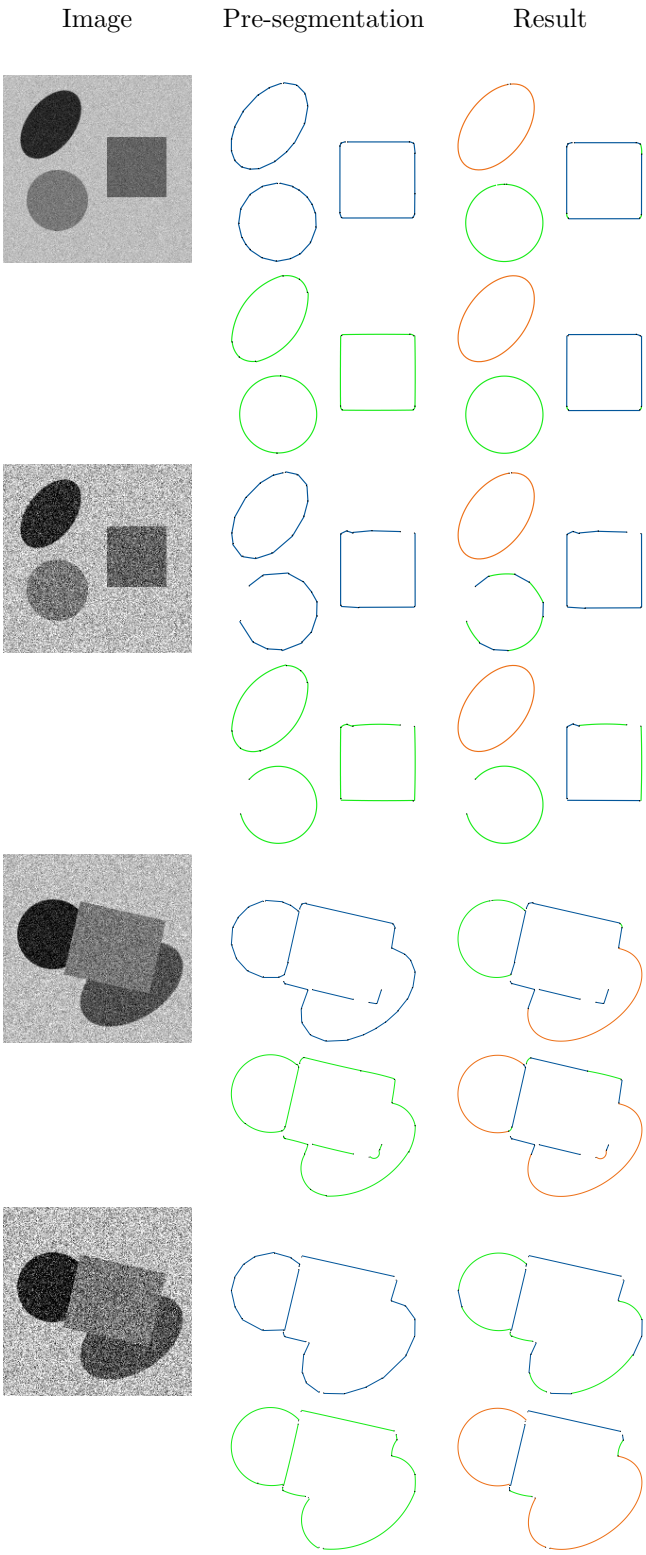


Figure 3.3: Comparison of the effect of pre-segmentation to the final result under different noise conditions. **Left:** Input image. **Middle:** result of pre-segmentation, first Dougla-Peucker, second `circlePeucker`. **Right:** Final result after merging. From top to bottom $\sigma_n = 10, 30, 20, 40$ grayvalues. Colours see Figure 3.2. Parameter setting and number of segments after pre-segmentation and merging, respectively, are given in Table D.1, page 188.

Figure 3.4: Comparison of the effect of pre-segmentation to the final result of DLCE on natural images and comparison to the results of ELSD. **a_*:** Given images as used in (Patraucean et al., 2012). **b_*:** Pre-Segmentation using Douglas-Peucker. **c_*:** Pre-Segmentation using `circlePeucker`. **e_* and f_*:** Final segmentation using b_* and c_*, respectively. **d_*** results of ELSD. Colours see Figure 3.2. We provide parameter setting and number of segments in Table D.2, page 189.

3.5.5 Comparison to ELSD

We give two more results on natural images in Figure 3.5. We compare our results to those from ELSD (Patraucean et al., 2012) in Figures 3.2, 3.4, and 3.5, and further in Appendix D.1 in Figures D.1 and D.2. Let us take the worm of the book cover, shown in Figure 3.4. ELSD resolves nearby edges, e.g., the black boundaries of the worm. The pre-processing of our method identifies these as (dark) lines which are then simplified. The slightly curved boundaries of the letters W or B are straightened by ELSD, while better resolved by our method. ELSD simplifies too much, e.g., the ellipse of O in the STOP-sign. The same can be observed in the Gothic window in Figure 3.5 left in several details.

While ELSD detects the edges independently, our method segments the edge or line pixel-chains; therefore, at sharp corners occasionally an additional short segment is preserved, e.g., the rectangles within Figure 3.2.

3.5.6 Using DLCE for Feature Extraction on Facade Images

Finally, we investigate the usage of DLCE for the extraction of image features in terms of line segments on facade images. Figure 3.6, top row, shows image patches of rectified facade images of several different window types. In the middle, we see the result of edge extraction, thus, pixel-chains which serve as input for DLCE. Finally, the bottom row shows the result of DLCE. More results on balconies and entrances are given inAppendix D.1 in Figure D.3 and D.4.

When observing the pixel-chains, we see the effect of rounding the corners, as already described in Section 3.5.3. Further, we take into account the accuracy of edges in object space. While we are able to extract edges in images up to an accuracy of 0.1 pixel, this is not related to the structure of edges in object space. Consider an edge on a facade with clinkers. The texture of these surface will result in much worse edge detection accuracy.

Thus, to avoid small segments, representing small roundings at corners, and to produce reliable and representative image features, we try to generalise as much as possible without spreading out probably important details. Therefor, we set the expected accuracy of edges $\sigma_e = 0.3$[px] and the threshold for pre-segmentation $t = 5\sigma_e$. This way, we yield promising results for further processing of these features. Boundaries of window panes are represented by long segments of straight lines, sometimes flat arc segments. Corners of window panes and scuncheons are represented as intersection of long straight line segments or arc segments in most cases.

3.6 Summary

In this chapter, we presented a line simplification approach which approximates given pixel-chains by a sequence of lines, circular, and elliptical arcs. For this, we proposed an adaption to Douglas-Peucker's algorithm for the use of circles instead of straight lines. The pre-segmentation, using `circlePeucker`, correctly identifies arc segments and especially their breakpoints. By itself, these are promising results and improve the standard algorithm in terms of reducing the number of breakpoints of a given polygon, while preserving the geometry.

Furthermore, we developed an approach for the simplification of such a segmentation by merging neighbouring segments due to their agreement to a joint geometric model in terms of description length. The merging step identifies elliptical arcs correctly and further reduces the number of segments of most given pixel-chains. Lines are identified in most cases. If not, this might be due to distortions, especially for long lines.

The approach depends on few parameters which are explainable by a priori knowledge about edge detection accuracy. Depending on the assumed edge accuracy, we showed accurate results. We showed the effects of polyline segmentation and simplification on several images with comparable good results, referring to a state-of-the-art algorithm. We proved the success of merging in terms of the reduction rate of number of segments per object.

In Chapter 4, we will use the final segmentation to derive high-level image features as input for an image interpretation system.

Figure 3.5: Results of DLCE on real images. **Left:** Gothic window. **Right:** cropped icosahedron. **From top to bottom:** given image, our final result using `circlePeucker` for pre-segmentation, result of ELSD. Colours, see Figure 3.2. We provide parameter setting and number of segments in Table D.3, page 190.

Figure 3.6: Results on facade objects, windows. **Top row:** Image patches. **Middle:** Extracted pixel-chains. **Bottom:** Results of DLCE. Colours, see Figure 3.2.

CHAPTER 4

Using Shapelets for Object Classification

In this chapter, our objective is the categorisation of the most dominant objects in facade images, like windows, entrances, and balconies. The approach, shown in this chapter, deals as intermediate level of the whole interpretation process. Given the result of line segmentation, presented in Chapter 3, we learn representative pairs of line segments, called shapelets, which we use as complex features for object classification. The proposed features are scale invariant and provide large distinctiveness for facade objects. The results will provide evidence for the high-level interpretation of whole images from top down, which we will show in Chapter 5.

4.1 Introduction of Shapelets and their Use for Object Categorisation 108
4.2 Pairs of Adjacent Line Segments 109
4.2.1 Detecting PAS 109
4.2.2 Representing PAS 111
4.2.3 A Distance of two PAS 112
4.3 Learning Shapelets 113
4.4 Using Shapelets for Classification 114
4.4.1 Soft Assignment for BoW Histograms 115
4.4.2 Normalising BoW Histograms 115
4.4.3 Including Spatial Information into BoW Histograms 115
4.5 Related Work 116
4.6 Experiments 118
4.6.1 Experimental Setup 118
4.6.2 Evaluating Shapelets 121
4.6.3 Comparison to HOG Descriptor 125
4.7 Summary 127

4.1 Introduction of Shapelets and their Use for Object Categorisation

This chapter focuses on the categorisation of objects in a facade, which is meant to serve a top-down module as a link to the image data. The scope of the chapter is to classify subsections of rectified images, based on the histogram of relevant pairs of line segments in a bag of words (BoW) model, as introduced in Section 2.4. We learn class models directly from images, and classify new instances in the presence of intra-class variations, clutter, and scale changes.

As pointed out in Section 1.4.2, learning which features best describe the target object classes is essential in order to avoid humanly modified models for classification or object detection. Therefore, we learn the parts that represent individual object classes, thereby, avoiding fixed and pre-selected features. Further, we already identified line segments as promising image features, due to their large invariance to shadows and changes in illumination and their distinctiveness for facades objects.

We propose a bottom-up approach to learn generic parts of object structure from given training image sections of these facade objects. We learn pairs of adjacent line segments (PAS) from image sections showing individual objects of the target classes. We identify which PAS are relevant to serve as words in a BoW model, thus, are suited to be part of a compact and discriminative image description and call these representatives shapelets. In order to be independent on the images scale, we choose a representation which is scale invariant. Finally, we use learned shapelets to classify new unseen image segments of learned target classes using the histogram of shapelets.

The main concepts are based on the work of Ferrari et al. (2008) about groups of adjacent contour segments for object detection. Unlike Ferrari et al., who use straight-line segments, we use line segments of more complex geometry as low-level image features.

Ferrari et al. (2008) and Wenzel and Förstner (2012) use k-aggregates of straight line segments. Thereby, a k-aggregate is a group of k-adjacent lines. Given the set of extracted line segments, the complexity of the search and the amount of possible k-adjacent segments increases exponentially. As already stated by Ferrari et al. (2008), this is infeasible for degrees greater than $k = 5$. Moreover, they proved, that pairs of adjacent line segment performs best compared to the costlier k-aggregates. This is equivalent to our observations in Wenzel and Förstner (2012). The reasons for this are intuitive. These small aggregates are simple enough to offer a moderate complexity and to guaranty repeatability. Nevertheless, they are more complex than simple gradients to capture more object information. For that reason, we search for pairs of adjacent line segments and ignore more complex aggregates.

Following the classical BoW approach, as described in Section 2.4, we start from given training image sections, together with their labels. We extract line segments using DLCE, as described in Chapter 3. From these we collect all possible PAS $\boldsymbol{t}_j \in \mathcal{T}$. Hereby, $\mathcal{T}$ is the infinite large set of all possible PAS we may ever detect. To learn which PAS are relevant for describing the target classes, we cluster PAS extracted from all training images. Identifying a representative for each cluster results in the codebook $\mathcal{S}$. Thus, we interpret identified PAS $\boldsymbol{s}_i \in \mathcal{S}$ as words of a vocabulary $\mathcal{S}$, as it is usually done in BoW approaches. In the following, PAS representing words of the vocabulary, are called shapelets. Each image section is represented by the histogram $\boldsymbol{h}(\boldsymbol{s}_i), \boldsymbol{s}_i \in \mathcal{S}$ of shapelets. Taking this as feature vector $\boldsymbol{x} = [\boldsymbol{h}(\boldsymbol{s}_i)]$, we train any classificator that is suited for our tasks.

Having learned the vocabulary $\mathcal{S}$ and the classification model, we classify a new image section by detecting shapelets of the vocabulary, taking their histogram, and estimating its most probable class using the learned classifier.

Next, we give more details about the several parts of the process. Section 4.2 introduces PAS features, their representation, and defines a distance between two PAS features. We show how to learn shapelets from PAS, collected from training images, in Section 4.3, and how to use them for object categorization in a BoW approach, in Section 4.4. We review related approaches in Section 4.5 and

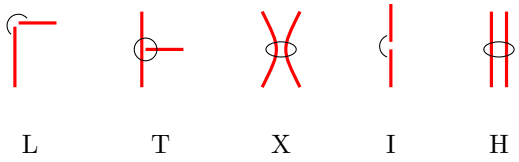


Figure 4.1: Junction types

evaluate the proposed procedure in Section 4.6. Section 4.7 summarises the chapter.

4.2 PAS - Pairs of Adjacent Line Segments

Given our line segmentation, presented in Chapter 3, we start looking for PAS. For this, we need a definition of adjacent segments and a way to identify them. Partly, this is already given by the pre-segmentation of pixel-chains. Segments neighboured within a pixel-chain are adjacent by definition. How we identify more adjacency relations, will be given in Section 4.2.1. In Section 4.2.2, we define a descriptor for PAS which will be suited to define a measure of dissimilarity in Section 4.2.3.

4.2.1 Detecting PAS

In this section, we are looking for all adjacent pairs of line segments within an image. Two line segments that meet each other build a junction which we differ depending on the way they meet. Figure 4.1 visualises junction types we distinguish. We call neighboured line segments

L-junction if they meet each other at their end-points and are not collinear,

T-junction if they meet each other at the end-point of one segment

X-junction if they are curved line segments that pass near by

I-junction if they are collinear at their intersection point. In case of curved line segments, their tangents at their intersection point are collinear.

H-junction if they are parallel straight line segments that pass near by.

In order to identify all pairs of adjacent line segments, we create a feature adjacency graph (FAG), see below, that states which segments are said to be neighboured. It is an undirected graph. Its nodes are the segments detected in the image. Two nodes, i.e. segments, are connected by an edge if they are defined to be neighboured. From the FAG, we are able to derive aggregates of adjacent line segments, up to an arbitrary degree k, by traversing all possible paths of length k. But, as already argued, we only search for pairs of adjacent line segments and ignore more complex aggregates. Segments, neighboured within a pixel-chain, are said to be adjacent, naturally. We add the according edges to the FAG immediately.

From the FAG, we are able to derive topologically neighboured line segments, but these need not to be adjacent, as their distance is arbitrary. In order to identify adjacency relations, we define the following rule to link segments which are not neighboured within the same pixel-chain:

Segments that meet at any point within a radius of ϵ pixel are said to be adjacent.

We declare all pairs of neighboured line segments, fulfilling this rule, to form a PAS feature. But, we exclude I- and H-junctions, thus, pairs of segments which meet within 180°, thus, form a line, and segments which are parallel. This allows us to

- bridge discontinuities between following pixel-chains of a joint object boundary; therefore, to encode L- junctions, but exclude I-junctions

- combine curved line segments which pass near by, X-junction, but exclude parallel straight line segments
- define T-junctions as PAS feature.

The first type is already included within the basic set of adjacent segments given by DLCE, whereas the second and third items are realised by the additional rule of adjacency.

The reasons for excluding I- and H-junctions are two-fold. First, we observe that their presence superimposes more discriminative PAS when learning the codebook. Second, their location is undefined, as they have no distinctive point of reference, due to the undefined point of intersection of the two segments.

To get more robust features, we further ignore small segments which result from pixel-chain segments smaller ϵ pixel, again. To argue on that, imagine a perpendicular corner, as in the rectangle shown in Figure 4.2. As already stated, we do not extract perfect corners due to smoothing and integrating during the edge extraction process. Therefore, the result of DLCE, depending on the size of the integration scale σ_2, yield small circular arc segments between two lines. Due to our tight setting of parameters for edge extraction, these effects are small, but stable and predictable. Independently of the resolution of an input image, and given a fixed value of σ_2, we observe these small segments of length 5 to 6 pixel. Taking the example of Figure 4.2 right, we ignore the small circular arcs and declare the straight line segments to be adjacent, as we would expect from the input image. In that case the value of 6 pixel is well-founded. For all the other cases, e.g., T-junctions, we did not find a large sensitivity to the allowed distance of adjacent segments in our experiments. Finally, we fix the allowed distance to $\epsilon = 6$ pixel, for all our experiments.

To create the initial FAG, thus to identify the adjacency relations of segments belonging to different pixel-chains, we exploit the distance transform of all detected segments. We create a binary image from segmented pixel-chains. Each pixel belonging to an extracted line segment gets a value of 1, the rest gets the value 0. We perform a distance transformation on the resulting binary image. Its complexity is linear in the number of pixels, thus, fast. The ridges of the distance transform define the points of equal distance to any adjoining segment, thus, inversely define the frontier between adjacent segments. This is equivalent to a generalised Voronoi tessellation. Thus, perceiving the names, i.e. IDs, of each segment, we infer, which segments join a common Voronoi boundary edge. If two of such segments meet within 6 pixel we add them to the FAG.

Finally, we extract all PAS from the image by collecting all pairs of adjacent segments from the FAG.

Out of the huge amount of detected PAS, we wish to select those which are relevant for further image interpretation. For this, we cluster all PAS found in a set of training images to create a set of most frequent PAS which we will use as words in a BoW approach. But, before clustering, we need a distance measure between PAS and subsequently a descriptor.

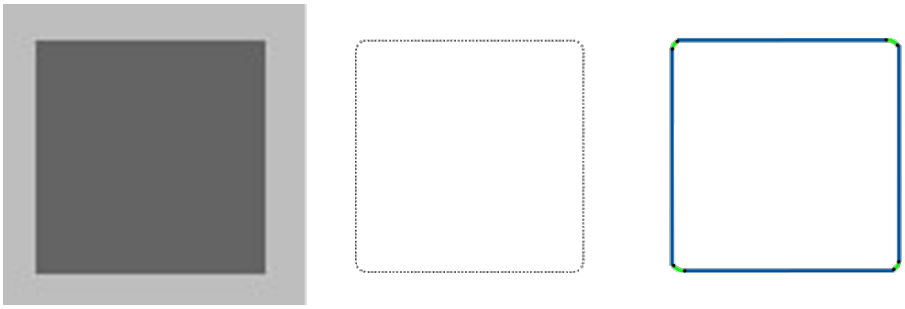


Figure 4.2: Effect of rounding corners. **Left:** input image. **Middle:** extracted pixel-chain. **Right:** result of DLCE. Please note the small circular arcs in the corners, which result from the rounding effect of the edge extraction.

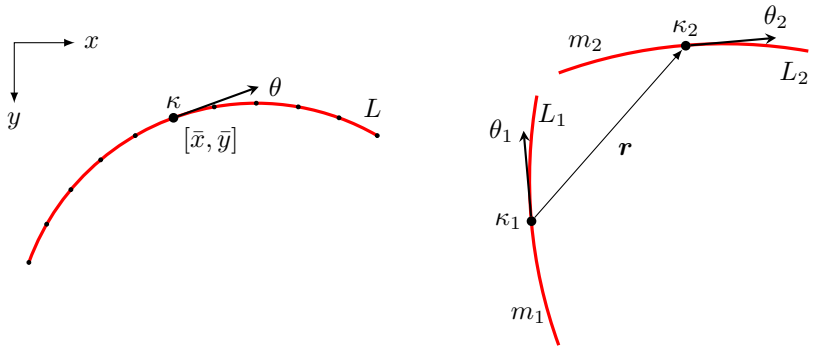


Figure 4.3: Parametrising PAS and its segments. **Left:** Each single PAS segment represents a chain of neighbouring pixels and is parametrised by its centre coordinates $[\bar{x}, \bar{y}]$, its curvature κ, length L, and orientation θ referring to the centre point. **Right:** A PAS is a group of two adjacent segments and is parametrised by their according parameters and the vector $\boldsymbol{r}$ between their centre points.

4.2.2 Representing PAS

We derive a descriptor for PAS feature according to the one given in Ferrari et al. (2008). Due to the use of curved line segments, we extended their descriptor by the curvature of line segments.

A PAS is an aggregate of two line segments, detected in the image. A segment could be a straight line, circular arc or ellipse segment. Following the approach given by Ferrari et al. (2008), we parametrise each segment by its centre $[\overline{x}, \overline{y}]$, its direction θ, its length L, and curvature κ, cf. Figure 4.3 left. The centre $[\overline{x}, \overline{y}]$ is defined as point on the segment half way along the segment. The direction θ refers to the direction of the tangent at centre $[\overline{x}, \overline{y}]$. The length L is given by the length of the path along the segment. We additionally add the curvature, due to the broader type of line segments compared to Ferrari et al. (2008). For lines the curvature is 0, for a circular arc $[x, y, r]$ it is the inverse of its radius $\kappa = 1/r$. Ellipses are more demanding as their curvature changes along the segment. In most cases we choose the maximal curvature along the segment. In case, we have a highly unsymmetrical ellipse segment, we choose the curvature at the centre. The orientation of each segment is normalised, such that $\theta \in \left[\frac{\pi}{4}, \pi + \frac{\pi}{4}\right]$. The reasons for that are two-fold. First, we avoid ambiguities due to segments which differ only by a direction of π. Second, we can not avoid discontinuities due to the circular property of θ, but we try to decrease its influence by shifting the discontinuity to $\pi/4$. When changing the orientation θ of a segment by π we change its sign of curvature, too. Following the established orientation of circle segments, c. f. Section 2.1.4.2, the curvature is positive if the arc lies on the right-hand side of the tangent. Please note, we do not discretise the geometric properties of each segment.

We derived the PAS descriptor from the parameters of its two line segments, whereby, we choose an appropriate normalization, in order to provide scale invariance. Without loss of generality, we assume two segments named 1 and 2. We choose segment 1 such that it is left of 2. If their segment centres are on top of each other, we choose 1 such that it is below 2. This way, we ensure an unique ordering of segments and therefore, unique descriptors independent of their IDs.

The PAS descriptor is given by the vector $\boldsymbol{r}$ of relative location, going from centre of segment 1 to the centre of segment 2, their tangent directions θ_1 and θ_2, their lengths L_1 and L_2, and finally, their curvature κ_1 and κ_2, as shown in Figure 4.3 right. The vectors of relative location, length, and curvature are normalised, by the length of $\boldsymbol{r}$, to provide scale invariance. Thus, a PAS descriptor is an 8-vector given by

$$\boldsymbol{t} = \left[\tilde{r}_x, \quad \tilde{r}_y, \quad \theta_1, \quad \theta_2, \quad \tilde{L}_1, \quad \tilde{L}_2, \quad \tilde{\kappa}_1, \quad \tilde{\kappa}_2\right]^{\mathsf{T}}, \tag{4.1}$$

using the normalised entries

$$\tilde{r}_{x,y} = \frac{r_{x,y}}{|\boldsymbol{r}|} \qquad \tilde{L}_{1,2} = \frac{L_{1,2}}{|\boldsymbol{r}|} \qquad \tilde{\kappa}_{1,2} = \kappa_{1,2} \cdot |\boldsymbol{r}| \,. \tag{4.2}$$

4.2.3 A Distance of two PAS

For clustering PAS features, we need a measure of dissimilarity, thus, a distance defined on two PAS-descriptors. We define the distance between two descriptors $\boldsymbol{t}^a$ and $\boldsymbol{t}^b$ as weighted sum of distances of each part of the descriptor

$$D(\boldsymbol{t}^a, \boldsymbol{t}^b) = w_r D_r + w_\theta D_\theta + w_L D_L + w_\kappa D_\kappa \qquad |\boldsymbol{w}| = 1 \tag{4.3}$$

with

$$D_r(\boldsymbol{t}^a, \boldsymbol{t}^b) = \frac{1}{\pi} \arccos\left(\tilde{\boldsymbol{r}}^a \cdot \tilde{\boldsymbol{r}}^b\right) \qquad D_r \in [0,1] \tag{4.4}$$

$$D_\theta(\boldsymbol{t}^a, \boldsymbol{t}^b) = \frac{1}{2} \sum_{i=1}^{2} \frac{1}{\pi} \arccos\left(\begin{bmatrix} \cos 2\theta_i^a \\ \sin 2\theta_i^a \end{bmatrix}^\mathsf{T} \begin{bmatrix} \cos 2\theta_i^b \\ \sin 2\theta_i^b \end{bmatrix} \right) \qquad D_\theta \in [0,1] \tag{4.5}$$

$$D_L(\boldsymbol{t}^a, \boldsymbol{t}^b) = \frac{1}{2} \sum_{i=1}^{2} \min\left(1, \left|\log_{10}\left(\frac{\tilde{L}_i^a}{\tilde{L}_i^b}\right)\right|\right) \qquad D_L \in [0,1] \tag{4.6}$$

$$D_\kappa(\boldsymbol{t}^a, \boldsymbol{t}^b) = \frac{1}{2} \sum_{i=1}^{2} \left|\tanh\left(\tilde{\kappa}_i^a - \tilde{\kappa}_i^b\right)\right| \qquad D_\kappa \in [0,1] \,. \tag{4.7}$$

The individual terms have intuitive interpretations. The distance D_r gives the angular difference of relative locations of both PAS. The larger D_r the more the PAS differ in their relative locations of involved line segments. The distance D_θ adds up the pairwise angular differences of individual line segments orientations. Thereby, we take each line segments orientation by the vector $[\cos 2\theta, \sin 2\theta]$ into account, to provide continuity. The distance D_L adds up the differences of log-length of single segments. We chose a basis of 10 for the logarithm and bound it by 1. This way, we take into account differences in length up to a factor of 10, but bound the distance of larger differences, to provide robustness. Finally, the distance D_κ measures the differences of curvature. It needs to take into account negative curvatures. Therefor, we chose tanh, which is symmetric and bounded. Please note, all individual terms are bounded such that they live in $[0,1]$. This way, we provide robustness and avoid that one term has substantial dominance over the other terms.

In order to derive the weights $\boldsymbol{w} = [w_r, w_\theta, w_L, w_\kappa]$ in (4.3) from data, we perform a grid search within a cross-validation framework. We used about 2000 manually labelled pairs of PAS, whereby, we label them as similar or not. About 1/10 of them are labelled as similar.

We vary all weights in ranges of $[1 \dots 5]$ and normalize them, such that they sum to one. For each state of grid search, we train a simple classifier on one part of the data and evaluate it on the rest of data. To be precise, we fit two Beta densities to the distances of positive and negative pairs, respectively, and estimate their intersection to determine the threshold of accepting a pair to be similar.

Figure 4.4 shows the evaluated distances for all labelled pairs of PAS using a constant value of 1/4 for all weights. We note, that distances of positive pairs are small, while distances of negative pairs are spread over the whole range of values, but by tendency larger than the positive ones, as expected. Further, we realise, that positive and negative samples are not perfectly discriminated. Fitting a broad Gaussian through the negative samples is intuitive and is confirmed empirically by the plot. Another choice would be a uniform distribution. But, actually, we expect and observe less negative samples with very small or very large values. For positive samples we expect the mode near 0 and a fast decrease for increasing distances. Both distributions are restricted to $[0,1]$, therefore, we decided to fit a Beta distribution to both distances from positive and negative pairs.

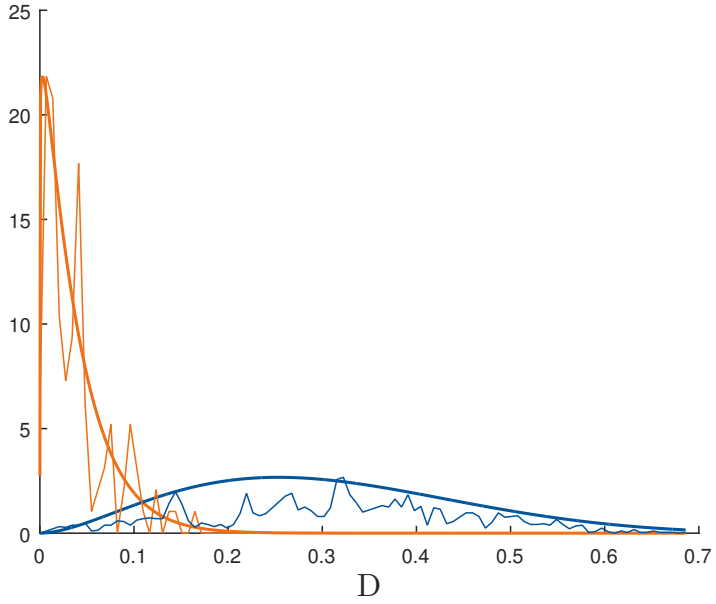


Figure 4.4: Distribution of PAS distances using weights $w_r = w_\theta = w_L = w_\kappa = 1/4$. Orange and blue lines show distances of pairs of PAS labelled as similar and not similar, respectively. Thin lines show the histograms of according distances. Bold line: Beta densities fitted through distances of similar and dissimilar samples. For better visualisation, histograms are scaled such that they fit their parametrised density approximately.

To summarise, we estimate these functions for each setting of the grid search and evaluate the quality of chosen weights by the overall accuracy of classifying the test data. To yield a reliable result, we estimate the weights within a 5-fold cross-validation.

Actually, we do not observe a large sensitivity to the choice of weights. As mean value we obtain the set of weights $w_r = w_\theta = w_L = w_\kappa = 1/4$ and a threshold for similarity of $\tau = 0.1$. Which we will use throughout all following experiments.

4.3 Learning Shapelets from PAS

We define shapelets as words of a dictionary that is suited for a compact image description. Learning the shapelets, therefore, is equivalent to learning the words of a BoW codebook. After extracting all PAS from the training images, we represent them as 8-dimensional feature vectors and identify frequently occurring shapelets by clustering.

As introduced in Section 2.4.2, clustering in BoW usually is done by k-means or mean-shift (Comaniciu and Meer, 2002). We decided to use the clique-partitioning approach proposed by Ferrari et al. (2008). As the distance of our PAS descriptor is not Euclidean, and we are not aware of an implementation of Mean-shift using an arbitrary metric, we do not use mean-shift. The design of the PAS descriptors prohibits averaging of samples as it is done in the standard mean-shift approach as well as in k-means. Therefore, we perform clique-partitioning, as described in Section 2.4.2.

From all PAS-features, extracted from the training images, we build a complete graph $\mathcal{G}(\mathcal{V}, \mathcal{E})$, where each vertex $v \in \mathcal{V}$ represents one PAS in feature space and edges $e \in \mathcal{E}$ are weighted by $w_{ij} = \tau - D(\boldsymbol{t}_i, \boldsymbol{t}_j)$. By introducing τ and taking the negative of D, similar features get heigh weights, while dissimilar features get negative weights. Using the approach of solving the CP-problem, given in Section 2.4.2, we partition the graph into completely connected disjoint subsets of vertices - cliques - such that the sum of remaining edge weights is maximised.

For building the codebook, we need to choose one representative per cluster. As already mentioned, averaging the PAS-features belonging to a cluster is inappropriate. Therefore, we choose the feature

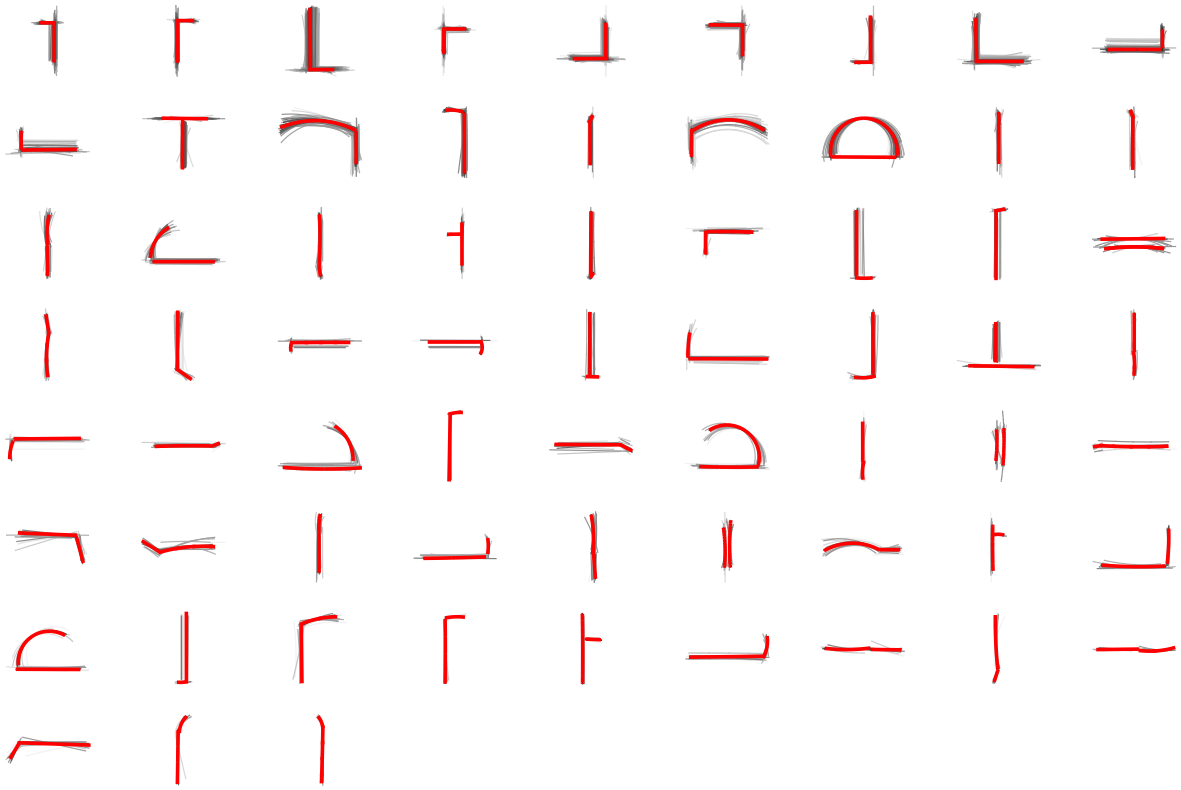


Figure 4.5: An example of a learned codebook of shapelets from a set of 196 image sections, showing arc-type windows. Light gray lines visualize PAS from the cluster. Thick red lines visualise the representative of the cluster, thus, the extracted shapelet. Shapelets are sorted by the size of the cluster, in decreasing order.

with smallest distance to all other features of the current cluster, thus, the one closest to the cluster centre.

The partitioning of the graph is complete. This means that each PAS is member of one cluster. Vice versa, each cluster contains at least one PAS. For sure, we are not interested in clusters containing less than N_t PAS. We assume them to be clutter.

Finally, the codebook contains the representatives of those clusters which contain more than N_t PAS. These members of the codebook are called shapelets. We will investigate the number of N_t in our experiments, cf. Section 4.6.

An example of a learned codebook of shapelets is shown in Figure 4.5. Here, we used a set of 198 images section, solely showing arc-type windows, cf. Section 4.6.1. We keep all clusters which contain more than $N_t = 10$ PAS. Visualised shapelets are sorted in decreasing order, such that the largest cluster appears in the beginning. As expected, shapelets with rectangular L-shape are most dominant and appear at the very beginning. But, we also observe T-junctions and shapelets which contain curved line segments which are most discriminative for arc-type windows.

4.4 Using Shapelets for Object Categorisation

Having learned the vocabulary of shapelets, we aim at learning a classification model based on shapelets as features. Therefor, each image region should be described by a single feature vector, which is the input for learning and using the classifier, respectively. In this section, we describe the design of our image descriptor based on shapelets.

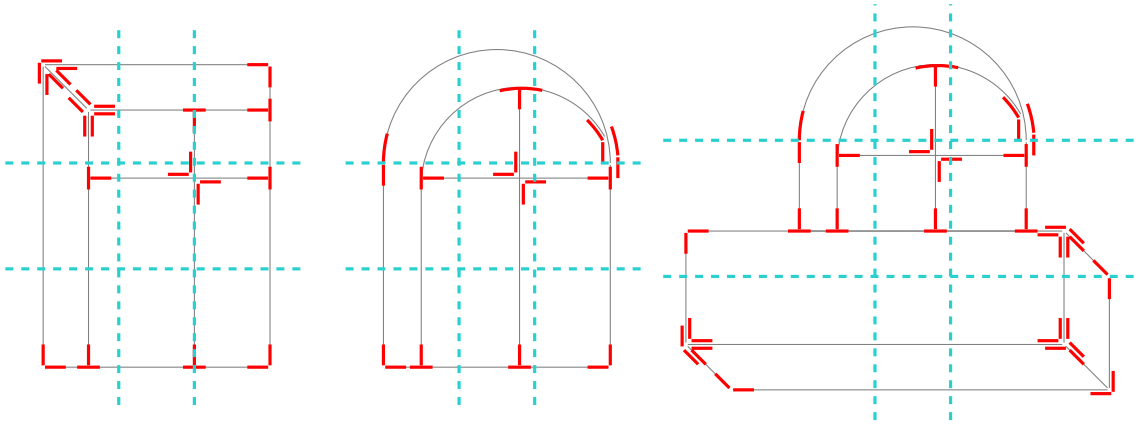


Figure 4.6: Tiling the image patches encode spatial layout: Rectangular and arc type windows are indistinguishable at their lower parts, but have their characteristic features at their top. Reversely, balconies, at their upper part, show doors or windows while having characteristic 3D structure below.

4.4.1 Soft Assignment for BoW Histograms

Having learned the codebook, we represent each image patch by its histogram of words. Thus, we count the occurrence of each codebook entry within the image patch. But, instead of assigning each found PAS to a single histogram bin, the closest one, we assign it to all types which are nearby in terms of the dissimilarity measure given in (4.3). For this we spread a total weight of 1 over all bins with $D < \tau$ proportional to the inverse of the dissimilarity. To be precise, let us take a detected PAS $\boldsymbol{t}$ with dissimilarities $\boldsymbol{D}(\boldsymbol{t}, \boldsymbol{S}) < \tau$ to I shapelets $\boldsymbol{S} = \left[\boldsymbol{s}^i\right]$, $i = 1 \dots I$. We increase the i-th bin by

$$\Delta h^i = \begin{cases} 0 & \forall_i \quad D(\boldsymbol{t}, \boldsymbol{s}^i) > \tau \\ \dfrac{1}{\sum\limits_j D(\boldsymbol{t}, \boldsymbol{s}^j)} \; \dfrac{1}{D(\boldsymbol{t}, \boldsymbol{s}^i)} & \forall_{i,j} \quad D(\boldsymbol{t}, \boldsymbol{s}^{i,j}) \leq \tau \end{cases} \tag{4.8}$$

For example, assume dissimilarities $\boldsymbol{D}(\boldsymbol{t}, \boldsymbol{S}) = [6.2 \quad 5.2 \quad 1 \quad 5.8]$ of a single PAS $\boldsymbol{t}$ to a codebook of four shapelets $\boldsymbol{s}^i$. Further, assume the threshold for similarity to be $\tau = 6$. We increase the four according bins of the histogram by $\boldsymbol{\Delta h} = [0 \quad 0.1409 \quad 0.7327 \quad 0.1263]$. The dissimilarity to the first shapelet is too large; thus, the according bin is not increased at all. The dissimilarity to the third shapelet of the codebook is the lowest; thus, it gets the highest increment, while the dissimilarities to the second and fourth shapelets are near τ. Therefore, the according bins get small increments. All weights sum up to a total amount of 1. This way, we make the representation less sensitive to the exact shape of detected PAS and allow a better generalisation of representations of object classes.

4.4.2 Normalising BoW Histograms

As argued in Section 2.4.3, the histogram should by properly normalised to be independent on the image size. Otherwise, the number of counted words grows by the size of the image. In our experiments, we will compare L2- and tf-idf weighting, as introduced in Section 2.4.3. The former means normalising such that the length of the histogram vector is 1. The latter is a weighting scheme that reduces the impact of words that occur too often to be meaningful.

4.4.3 Including Spatial Information into BoW Histograms

Instead of using a simple BoW representation which means simply to count all shapelets occurring in the image and building the according histogram, we partition each image patch into several tiles and

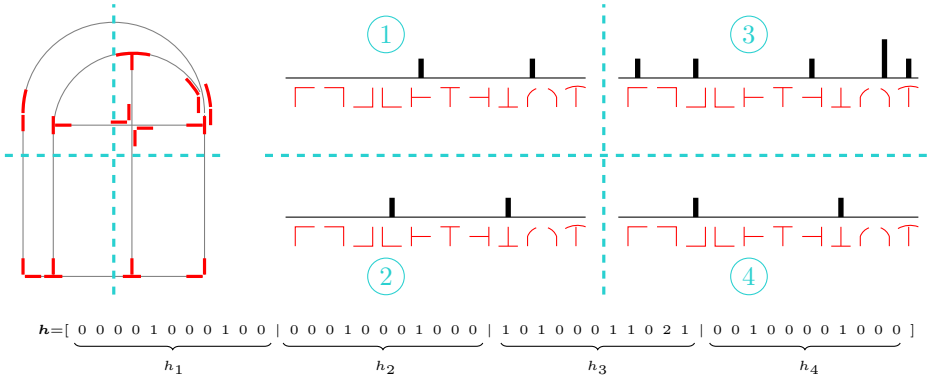


Figure 4.7: Building the BoW feature vector for using a partitioning into 2×2 tiles.

build the histogram for each of them separately and concatenate them to yield a single feature vector per image patch. This way, we incorporate spatial layout of inspected object classes without building a complex implicit shape model (ISM) (Leibe et al., 2004). This is similar to the idea used by Dalal and Triggs (2005), who introduced the widely used histogram of oriented gradients (HOG).

The idea of tiling is sketched in Figure 4.6 for a partitioning into four tiles. Ferrari et al. (2008) extensively evaluates the effect of different numbers of tiles. In their work they used groups of adjacent line segments of complexity up to $k = 4$. They found the optimal number of tiles the lower the higher the complexity of line aggregates. This is reasonable, as complex features are more discriminative concerning the shape, while simple features are less discriminative by them self; thus, the overall descriptor need to incorporate more spatial layout. For $k = 2$, thus, PAS features, they found the optimal number of tiles to be $T = 30$. However, we will investigate the effect of tiling onto the result in the experimental section.

To complete the procedure: We build the histogram of shapelets for each tile and concatenate them to a joint feature vector of length $T \times S$, whereby, T is the number of tiles and S is the number of shapelets in the codebook. Figure 4.7 visualises the architecture of a feature vector belonging to one image patch, using a partitioning of 2×2 tiles. In this simple example, the codebook consist of 10 words. We build the histogram of words for each tile separately and concatenate all four histograms to a joint feature vector $\boldsymbol{h}$ which compactly describes the according image patch and serves as input for either training a classificator or for using an already learned classificator.

In Section 4.6, we will present a framework for evaluating found shapelets. Thereby, we will evaluate different choices of parameters.

4.5 Related Work

Our approach is most related to two large groups, the first, working on object recognition and detection based on shape and contour, respectively, the second, working on learning mid-level features or even visual elements.

In order to categorise or detect objects, based on their characteristic contour, many works either use groups of edge fragments to detect learned classes of shapes, while most of them use Hough based voting techniques. They treat shapes as loose collections of 2D points (Cootes et al., 1995) or other 2D features (Leordeanu et al., 2007; Elidan et al., 2006). Belongie et al. (2002) introduce to use shape context to measure the similarity between shapes and exploit it for object recognition. For each point they exploit the spatial distribution of all other points on the shape. This allows to establish a point-to point correspondences between shapes even under non-rigid deformations.

However, to increase matching performance more informative structures than individual features are needed. For this, we found either ensembles of pairwise constraints between point features (Leordeanu

et al., 2007; Elidan et al., 2006) or a global geometric organisation of edge fragments (Shotton et al., 2005; Opelt and Zisserman, 2006; Shotton et al., 2007; Ferrari et al., 2008; Ommer and Malik, 2009; Lu et al., 2010; Srikantha and Gall, 2014), which allow to handle deformations. They belong to the principle of perceptual grouping which relies on the idea that fragments of contour, related by some perceptually salient property, are more likely to belong to the same object (Ferrari et al., 2008).

Leordeanu et al. (2007) exploit relations between all pairs of small edge segments, while Elidan et al. (2006) use pairwise spatial relations between landmark points. The former uses all line segments from a single image to initialises a model, which is then refined by iteratively removing spurious features and adding new good features identified in many other images. The latter, address the modelling of shapes of general object classes in 2D in order to outline according objects in real images. Interestingly they use drawings of the target classes as training samples which capture the fundamental contour, while avoiding problems such as clutter and shading variations. They propose a semi-parametric model which is defined in terms of landmarks, each associated with local contour information. They formulated a MRF over these landmarks, such that the outline of objects in real images can be found by standard MRF inference algorithms.

These approaches aim at describing whole contours, thereby actually focusing on the outer contour of their target objects. We learn from these approaches that contour features reveal a large distinctness to certain object classes. On the other hand, objects we focus on, reveal their distinctiveness through the appearance inside the outer contour, e.g., structure of window panes or typical structure of balconies.

In order to discover clusters of useful contour features, that are both representative and discriminative, Ferrari et al. (2006, 2008, 2010) in a series of work, introduce a family of scale invariant local shape features build from short chains of connected contour segments. As already pointed out, we adopt their concept to build our PAS features. Similar to Ferrari et al. (2006, 2008), Yu et al. (2007) aim at object detection using a shape codebook. They directly build on Ferrari et al. (2006) and use Triple-Adjacent-Segments (TAS) extracted from image edges as shape codewords and use them in a Hough-based voting scheme, again, for object detection. Also Chia et al. (2012) build on work of Ferrari et al. (2008). Similar to our work, they adapt their PAS features by the use of lines and ellipses. They describe shape tokens consisting of line–line, line–ellipse or ellipse–ellipse pairs and use them in a way similar to Ferrari et al. (2008) and our work. Different to our descriptor, they do not take curvature of line segments into account, but the spacial extend of fitted ellipses. Further, they introduce a clustering over relative positions of found shape token clusters to incorporate spatial layout.

Lu et al. (2010) aim at contour based object detection using so called part bundles. They hierarchically decomposes the given contour model of an object class into fragments and group them again into part bundles which can contain overlapping fragments. Local shape similarity, between part bundles and edge fragments, cast votes for candidate part configurations while global shape similarity, between the part configurations and the model contour, yield the optimal configuration.

From these works, we take over the idea of bundling local parts of a contour and to use these bundles as discriminative features for the description of the target objects.

In order to encode spatial layout Shotton et al. (2005) and Opelt and Zisserman (2006) learn class-specific boundary fragments including their spatial arrangements as a star configuration, which allow for object localisation using voting, similar to ISM, as introduced by Leibe et al. (2004). Shotton et al. (2007) extends this work by introducing a codebook of scale normalised contour exemplars. Further, they introduce a new multi-scale oriented chamfer distance for matching contour fragments.

These approaches require an extra clustering step over spatial locations of learned discriminative features and assume the target classes to be consistent in their spatial layout, up to a certain degree of allowed variations.

Therefore, instead of spatial voting, Dalal and Triggs (2005) and Ferrari et al. (2008) encode spatial organisation by tiling the object windows and learning which feature-tile combinations are most discriminative. We uses this principle, as it is easy to implement and fast to evaluate, and due to the large variability in aspect ratio of our target objects.

Figure 4.8: For each row some examples of given image sections for class balcony, entrance, arc-type windows, rectangular windows, and background.

4.6 Experiments

In Chapter 5, we will apply the learned codebook and classification model to the task of object detection, which turns out to be nothing more than object categorisation in its first step. In that application, we have to decide at every position of an image and for every, at least a couple of scales, about the class of the current image section. Therefore, we relax the problem for our experiments in this chapter. The goal of this section is to show that our learned shapelets and proposed feature vector architecture are suited for object categorisation, whereby the term feature vector is meant as input for the classifier. We will vary several parameters to find the best set and to investigate the sensibility against these parameters.

4.6.1 Experimental Setup

The classification task is to learn a representation of several target classes, in a way that we are able to classify new and unknown images.

We choose a dataset with $C = 5$ classes namely balconies, entrances, arc-type windows, rectangular windows, plus background samples with $400, 76, 198, 400$ and 400 samples per class, respectively, see Figure 4.8 for some examples. We consider two different classes of windows in order to prove the usefulness of learned shapelets. Later, we will regard all windows as one object class.

For each sample image patch its target class c is given. These samples are taken from annotated rectified facade images, such that each sample image contains exactly one object of its given target class. Background samples are sampled randomly from rectified facade images. Samples, taken this way, that accidentally contain too large parts of foreground objects are removed manually. Please note that they are not resized to have equal size.

Throughout our experiments, we will vary two parameters for which we want to find the best choices. First, we consider different sizes of the codebook, which we measure by the minimal number N_t of PAS belonging to a cluster, see Section 4.3. The lower N_t the higher the number of shapelets S in the codebook and the larger the BoW histograms. Second, we inspect the number of tiles T used for building the BoW histogram, see Section 4.4.3. We separate T into a number of tiles T_X and T_Y, horizontally and vertically, respectively. Thus, we vary T_X and T_Y, which give a total number of tiles of

$T = T_X \cdot T_Y$, yielding different settings for the same number of tiles T. Additionally, we exploit the effect of normalising the BoW histograms using the standard L_2-normalisation and the tf-idf weighting scheme.

We perform a five-fold cross validation. The dataset is equally split into five groups, such that different sample sizes per class are equally split, too. In each cross validation step, we choose four of the groups for learning the relevant aggregates and for training a classification model. The remaining group is used for testing, thus, detecting proposed aggregates and testing the learned classifier. Please note that we learn a single codebook for all classes. Thus, the codebook is dedicated to provide the vocabulary for all object classes.

As classifier, we decided to choose the widely used support vector machine (SVM), cf. Section 2.4.4.1. Later, in Chapter 5, we need reliable class conditional probabilities for further usage in high-level interpretation, which we might get by IVM, cf. Section 2.4.4.2. Unfortunately learning the IVM model is time consuming, especially when performing grid search to fix needed parameters. Learning the SVM model is much faster, also it includes a grid search, too. As we merely want to investigate the performance of learned codebooks under different parameter settings, and as we received comparable results, throughout our experiments using IVM and SVM, we stick on SVM here.

We use the implementation provided by Chang and Lin (2011) call LIBSVM. In its default setting it uses a RBF kernel, who's kernel width σ_{RBF}, as well as the regularisation parameter λ_{SVM}, are determined by grid search on a subset of the training data. For multi-class classification, LIBSVM provides the one-versus-one approach. Having C classes, $C(C-1)/2$ classifiers are trained which separate every pair of classes. For classifying new unseen data, they use a voting scheme where each binary classifier votes a sample for one class. At the end a sample is designated to be in that class with the maximum number of votes. In case of equal numbers of votes, they simply choose the class appearing first in the list of classes, which is nothing more than to venture a guess.

The whole experiment is set up the following way: For learning, we first collect all PAS from all images from the training set. Performing clustering, we define the vocabulary of shapelets, our codebook. Thereby, we vary the minimal number of PAS belonging to a cluster using $N_t \in [10,\ 15,\ 20,\ 30,\ 50]$. Remember, the lower N_t the larger the codebook. To investigate the incorporation of spatial layout when building the BoW histograms, we vary the number of tiles horizontally and vertically using $\{T_X, T_Y\} \in [1 \dots 7]$, resulting in a maximum number of $T_{max} = 49$ tiles. For each tile, the BoW histogram is normalised either using L_2 or tf-idf weighting.

Using all feature vectors extracted from the training images we train the SVM classifier. For testing, we extract shapelets, belonging to the learned vocabulary, and build their BoW histogram. Using the learned SVM model, we got a prediction of the target class, which we compare to the given label to build the confusion matrix.

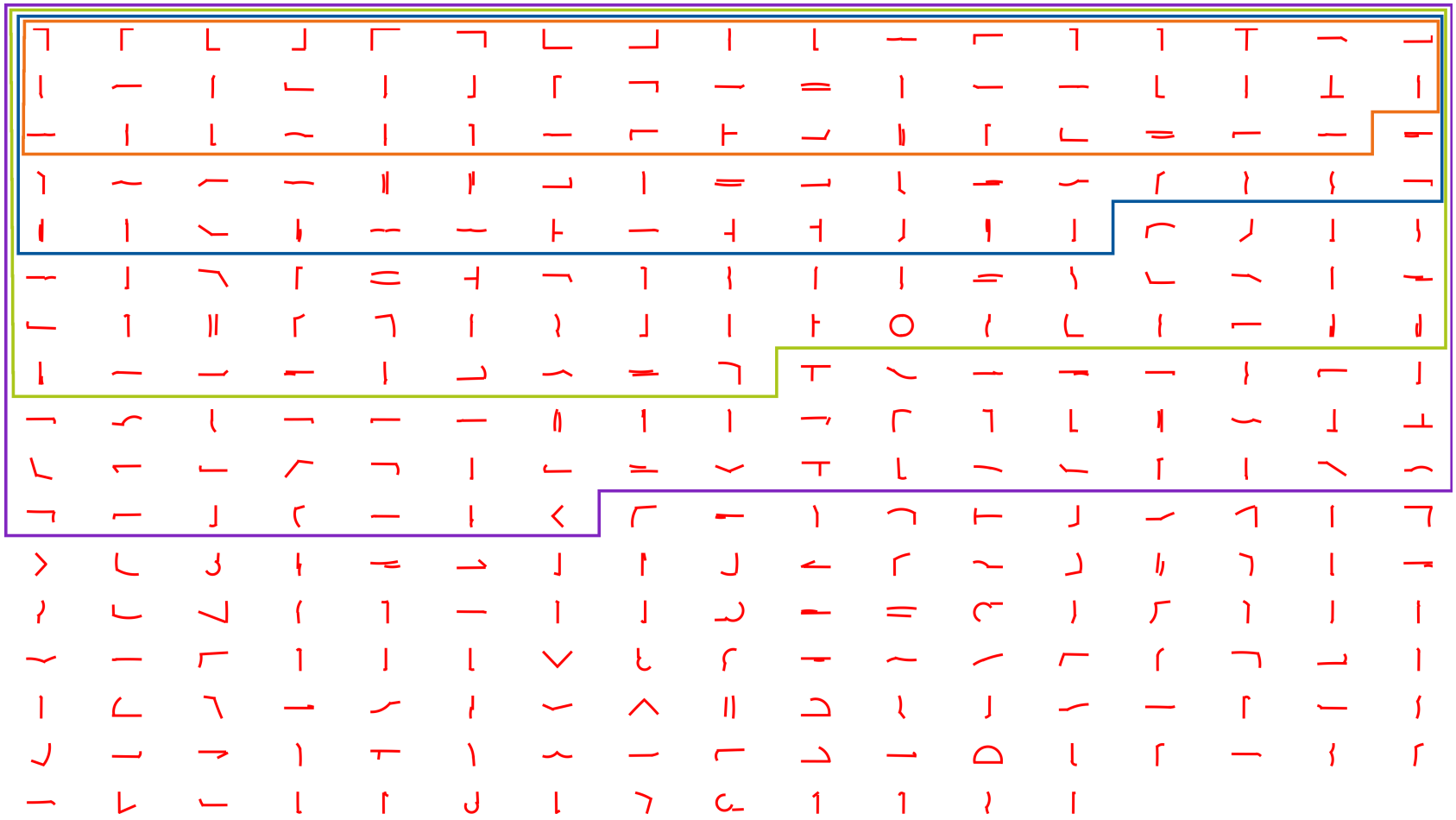


Figure 4.9: **Learned shapelets.** Shapelets learned in cross validation run 2, ordered by the number of PAS belonging to the cluster. The coloured boxes mark the codebook size for different choices of N_t. The whole set belongs to the codebook using $N_t = 10$. **Colours:** violet: $N_t = 15$, green: $N_t = 20$, blue: $N_t = 30$, orange: $N_t = 50$.

4.6.2 Evaluating Shapelets

Figure 4.9 shows the shapelets, learned during one cross validation run using a minimal number $N_t = 10$ of PAS belonging to each cluster, which yield in an average codebook size of 300 shapelets. Increasing N_t results in smaller codebooks, which we have marked by coloured boxes in Figure 4.9. The codebooks, learned in the other cross validation runs, are given in Appendix D.2 Figure D.5 to D.8, page 192.

As expected, the most dominant shapelets are those who capture the rectangular structure of facade objects. These shapelets are almost the same in all cross validation runs. Further, we realise shapelets showing angles which are characteristic for projecting rectangular 3D structure to the image plane. Due to the incorporation of curved line segments, we also find shapelets coding arc-type structure, as given in facade decorations or arc-type windows or doors, even whole or half circles.

Using these codebooks, we trained the classifier, while varying the named parameters and measure the classification performance in terms of accuracy. The overall accuracy (OA) is given as fraction of correctly classified samples, the number of true positives (TP), related to all samples N

$$\mathrm{OA} = \frac{\mathrm{TP}}{N} \,. \tag{4.9}$$

The average classwise accuracy is given as mean value of class specific accuracies a_c, Thus, as the mean of the confusion matrix diagonal.

$$AA = \frac{1}{C} \sum_c a_c \qquad a_c = \frac{TP_c}{N_c} \,, \tag{4.10}$$

whereby, N_c is the number of samples in class c, TP_c the number of correctly classified samples of class c and a_c the according accuracy of class c.

We report both overall accuracy and average classwise accuracy, respectively, due to the different number of samples per class. Therefore, the overall accuracy might be corrupted by the most dominant class and the average classwise accuracy gives a more objective view.

Figure 4.10 and 4.11 show the overall classification accuracy and classwise accuracy, respectively, depending on the codebook size and the spatial tiling for both L_2 and tf-idf weighting. We plot accuracy vs. tiling. Each line visualises the mean accuracy over the cross validation runs for one choice of N_t. The horizontal axes are sorted such that the results of the $N_t = 10$ line increase. Therefrom, we see the best choice of tiling at the very right of the plot.

First, we note that L_2 normalising significantly outperforms the tf-idf weighting scheme. This does not match our expectation, but might be related to numerical problems in Equation (2.88), page 66, due to the large number of shapelets. We will focus on L_2 normalisation in the following.

Further, we observe a significant increase of performance using the largest codebook, thus, $N_t = 10$. This becomes reasonable when looking at Figure 4.9. The smaller the codebook, the less discriminative the shapelets. The smallest codebooks, $N_t = \{50, 30\}$ contain almost entirely rectangular L-junctions, some T-junctions and only a few shapelets different from that. The most discriminative shapelets, containing arcs or other angles are part of small clusters and therefore, occur at the end of the list.

When looking at the performance depending on the chosen tiling, we find the best results for 3×3 tiles concerning overall accuracy. Concerning average classwise accuracy 3×5 seems to be the best choice.

Figure 4.12 shows the accuracy for each class separately, using the same visualisation as before. Additionally, Table 4.1 give the confusion matrices for the best choice of parameters.

We note that arc-type windows are correctly classified up to 93% accuracy. Due to their arc-type elements they are highly discriminative compared to other rectangular facade objects. Balconies are classified up to 88% true positive rate. Note that we are dealing with single images; thus, we have no 3D information, which usually guides recognition of balconies. Obviously, their images contain discriminative shapelets coding the 2D structure due to the perspective projection of 3D objects. In

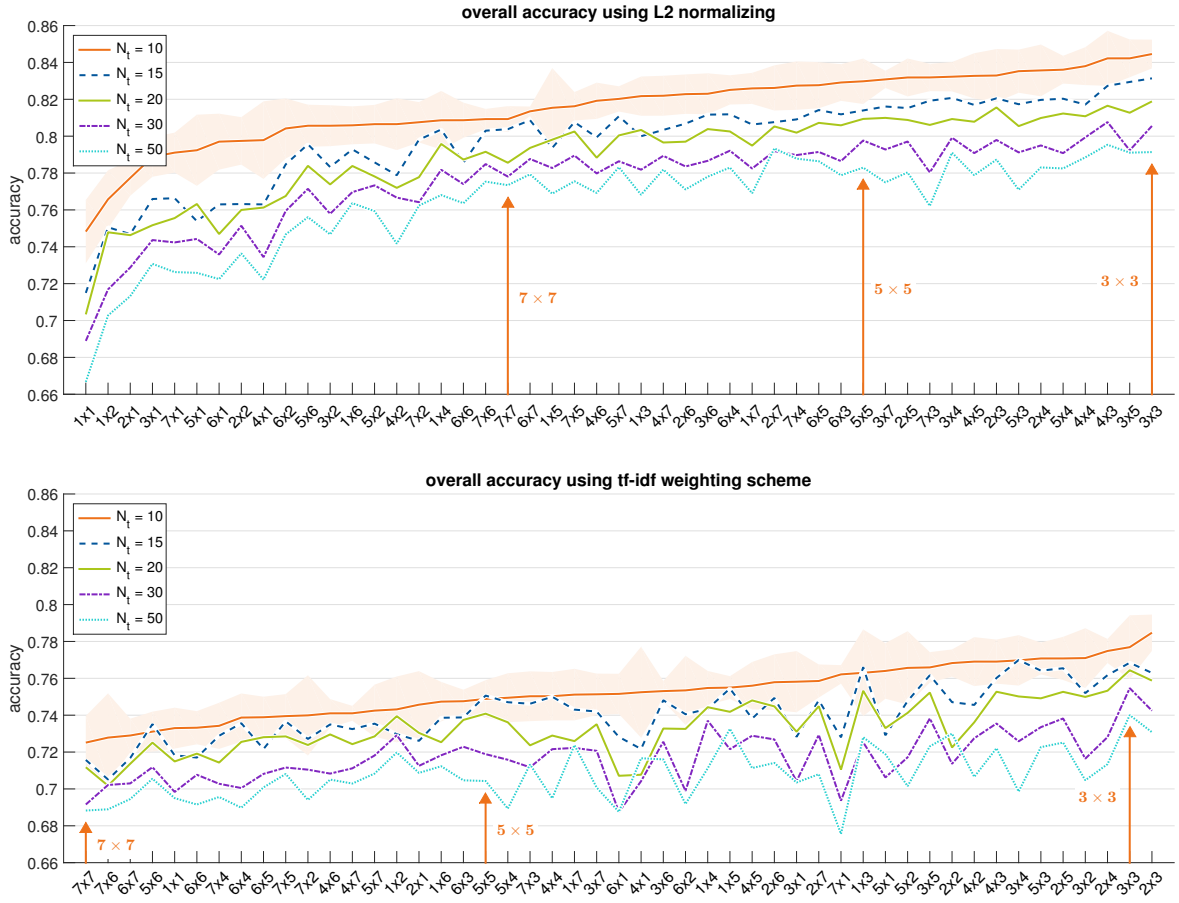


Figure 4.10: **Overall accuracy** using different tiling and different cluster size for L_2 and tf-idf weighting. The horizontal axis is sorted, such that the results for $N_t = 10$ are sorted increasingly. Please note that this sorting is different for both plots. Colours are given in the legend. To avoid overloading the figure, we give the $1\sigma-$ band only for the upper line. It looks similar for the other settings.

Table 4.1: Confusion matrix using $N_t = 10$, 3×5 tiles and L_2 weighting. Overall accuracy: 87.50%. Average class accuracy: 88.38%.

truth \ *estim*	background	balcony	entrance	win-arc	win-rect
background	**93.34**	4.07	0.23	1.84	0.52
balcony	3.37	**87.85**	6.25	0.90	1.62
entrance	0.79	9.84	**87.40**	0.79	1.18
win-arc	4.61	0.49	0.00	**92.72**	2.18
win-rect	5.62	2.21	0.11	11.34	**80.72**

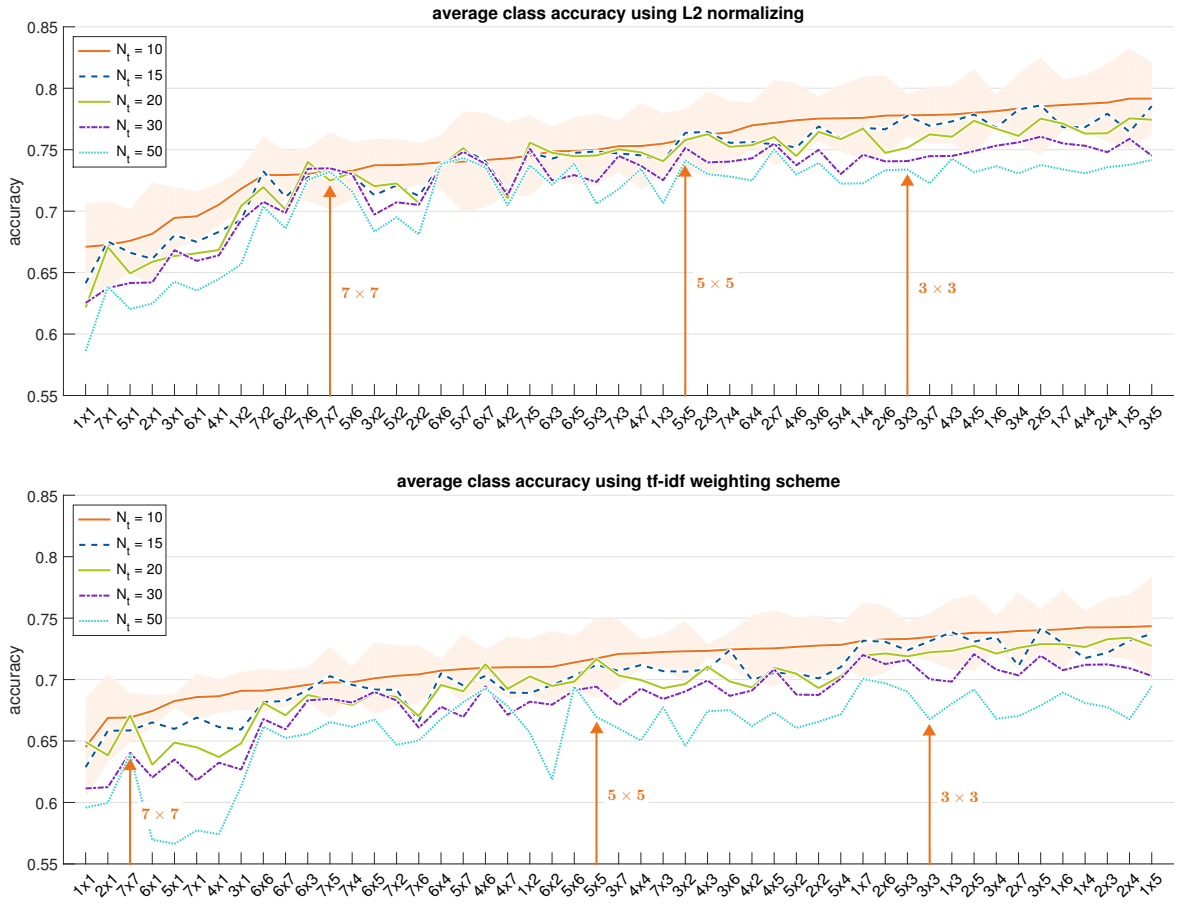


Figure 4.11: **Average class accuracy** using different tiling and different cluster size for L_2 and tf-idf weighting. Detailed description cf. Figure 4.10.

Figure 4.12: **Classwise accuracy** using L_2-weighting and different tiling and different cluster sizes. Detailed description cf. Figure 4.10.

contrast, rectangular windows are challenging and are confused with other facade elements. They are correctly classified with 83% accuracy. At most they are confused with arc-typ windows, which we may ignore when regarding all windows within one class. The confusion matrix, Table 4.1, proves that classification of entrances is possible, but still a challenging task, they are often confused with balconies. Incorporating high-level knowledge, such that entrances are most likely at the ground floor, this will be corrected.

We will use the learned classifier from this chapter in our experiments in Chapter 5. There, we will incorporate high-level knowledge about typical configurations of facade objects, which we learn from train images again.

Finally, we fix the parameters the following way

$$N_t = 10, \qquad T_X = 3, \quad T_Y = 3 \tag{4.11}$$

using the L_2 normalisation as already noted. This way, the feature space will be of dimension of approximately $300 \times 3 \times 3 = 2700$, which is large. But, the features vectors will be sparse and we will show how to handle them effectively.

4.6.3 Comparison to HOG Descriptor

We compare our shapelets to the widely used HOG descriptor of Dalal and Triggs (2005) which is the state-of-the-art in human detection and has proven competitive on other classes. Furthermore, a couple of implementations are available. Recently, it was published within the Matlab Computer Vision System Toolbox, thus, can be included into existing matlab projects the most easiest way.

We do not compare to the original groups of adjacent contour segments of Ferrari et al. (2008). Unfortunately, they provide an implementation for their successive work in Ferrari et al. (2010), but this does not include the object detection as described by Ferrari et al. (2008). In their succeeding work, they use their PAS features which are build from straight line segments, and derive object contour models for each addressed class. This includes spatial voting and an incremental refinement of contour segments belonging to the overall contour. The final model consists of a list of contour point and object detection is done by a combination of hough voting and shape matching. Thus, the work of Ferrari et al. (2010) goes beyond our task in this chapter and therefore, is not competitive to our approach.

Furthermore, in contrast to Ferrari et al. (2010), our task in this chapter is to derive a set of features which are suited for object classification within an MCMC approach. In Chapter 5, we will show how to build an integral histogram of PAS features for each image. This way, the feature descriptor for each image region, independent of size and position, can be derived in constant time, which is essential for making MCMC sampling feasible. This is impossible using the shape descriptor of Ferrari et al. (2010).

Therefore, we decided to compare to the HOG descriptor which is currently the state-of-the-art image descriptor for object categorisation.

The basic idea of HOG is that local object appearance and shape within an image can be described by the distribution of intensity gradients. Therefore, the descriptor uses the histogram of gradients to provide a descriptor for whole image patches. To incorporate spatial layout, the image is divided into small connected regions, called cells of size $a \times a$ where a typically ranges from 2 to 8 pixel or even more. For each cell a histogram of gradient directions is evaluated, for the pixels within the cell. The concatenation of these histograms then represents the descriptor. Further, Dalal and Triggs (2005) measure the intensity across several cells, called blocks, and use this to normalise all cells within the blocks. This way, the images are contrast-normalised and the descriptor is invariant to changes in illumination or shadowing.

For object detection, the HOG descriptor is used in a sliding window approach. Therefor, a classifier, typically a SVM, is trained on the HOG descriptors of a couple of training images. For detecting new instances of the target class(es), the classifier is evaluated for all possible locations of a search window into a query image. As the descriptor is not scale invariant, training images are typically resized to have equal size and the query image is evaluated over several scales.

Figure 4.13: Classification performance using the **HOG** descriptor with different tiling. **Top:** Overall accuracy as mean over 5 cross validation runs. **Bottom:** Average class accuracy as mean over 5 cross validation runs. The shaded region visualises the 1σ-band. Again, the horizontal axes are sorted, such that the results are sorted increasingly. For this reason we do not provide both lines within one plot.

For our experiments, we use the HOG implementation provided in the Matlab Computer Vision System Toolbox. In its default settings it uses a cell size of 8×8 pixels, a block size of 2×2 cells, a block overlap of half a block, which is the number of overlapping cells between adjacent blocks, and a discretisation of gradient direction into 9 bins, which are unsigned. We use this default setting except the cell size. Instead of resizing, all images used for training and testing, and using a fixed cell size, we fix the number of cells and derive the cell size for each image separately. Similar to our experiments in Section 4.6.2, we vary the number of cells to find the best setting.

We use exactly the same data for training and testing as in the experiment in Section 4.6.2. This includes the splitting of data into the cross validation subsets. We also use exactly the same classifier as in Section 4.6.2. Thus, we train a SVM using default settings, estimating needed parameters by grid search.

Figure 4.13 shows the classification accuracy using HOG descriptors with different spatial tiling. Again, we visualise mean results of overall accuracy and average classwise accuracy. It turns out that the best results are given for 6×5 cells. Table 4.2 shows the according confusion matrix. As expected, HOG performs well and stable over different choices of tiling. Indeed it outperforms our feature sets in terms of overall accuracy. But, our approach is still comparable and even better for classes balcony and entrance. This might be related to the ability to learn detailed facade structures in terms of contour

Table 4.2: Confusion matrix HOG descriptor with 6×5 tiling. Overall accuracy: 92.4%. Average class accuracy: 90.6%.

truth \ *estim*	background	balcony	entrance	win-arc	win-rect
background	**97.74**	1.44	0.25	0.19	0.38
balcony	5.39	**84.56**	9.54	0.00	0.50
entrance	3.95	13.16	**78.95**	1.32	2.63
win-arc	0.00	0.00	0.51	**98.47**	1.02
win-rect	1.65	2.59	2.12	0.82	**92.82**

fragments, which is more powerful than a collection of gradients.

It is important to note that our approach is comparable to the state-of-the-art performance of HOG. Beside that, we are able to evaluate image descriptors by using the integral histogram of PAS features, which we can not do when using HOG descriptors.

4.7 Summary

In this chapter, we proposed a new feature type which we call shapelets, and which are the representatives of most dominant pairs of adjacent line segments in images, which might be curved or not. We showed how to extract them from images and how to describe them. Further, we proposed a distance measure and showed how to cluster them to derive the according shapelet.

For classification, we use the histogram of shapelets. We incorporate a rough spatial layout by splitting the image patch into several tiles, from which the overall image descriptor is given by the concatenated histogram of shapelets of all tiles.

Using these shapelets, we provided a reasonable classification performance on a challenging dataset, including intra-class variations, clutter, and scale changes. We are slightly worse than using the HOG descriptor, but still comparable, at least for classes balcony and entrance. Furthermore, our image descriptor is suited for fast evaluation using the integral histogram which we will introduce in the next chapter. However, shown classification performance proves that the learned set of shapelets is suited to give good evidence for existence of certain facade objects from bottom-up. This will be done in the following, when including the approach into an object detection method, based on sliding window and using it as a region classifier in an MCMC framework.

CHAPTER 5

Top-Down Facade Image Interpretation by Marked Point Processes

The objective in this chapter is the interpretation of facade images in a top-down manner using a Markov marked point process. Given single rectified facade images, we aim at the accurate detection of relevant facade objects, namely windows, entrances, and balconies, using prior knowledge about their possible configurations within facade images. We represent facade objects by a simplified rectangular object model and present an energy model which evaluates the agreement of a proposed configuration with the given image and the statistics about typical configurations which we learn from training data. We show results on different datasets and provide a qualitative evaluation, which demonstrates the capability of complete and accurate detection of facade objects.

5.1 Using Bottom-Up Evidence for Top-Down Facade Interpretation 130
5.1.1 The Object Model 130
5.1.2 Generating Bottom-up Evidence 131
5.1.3 Integral Histograms 131
5.1.4 Object Detection Using a Sliding Window Approach 133
5.2 An Energy for Facade Image Interpretation 133
5.2.1 Energy Formulation 135
5.2.2 The Data Term 135
5.2.3 Prior Energy 136
5.3 Optimisation 141
5.3.1 Birth and Death. 142
5.3.2 Non Jumping Transformations. 142
5.4 Related Work. 143
5.5 Experiments 145
5.5.1 Datasets 146
5.5.2 Simulation 146
5.5.3 Evaluation 146
5.6 Summary 156

5.1 Using Bottom-Up Evidence for Top-Down Facade Interpretation

In this chapter, our objective is the interpretation of facade images from top-down. Given single rectified facade images, we aim at the accurate detection of relevant facade objects as windows, entrances, and balconies, in terms of their bounding boxes, using prior knowledge about their possible configurations within facade images. From bottom-up, we detect shapelets, as proposed in Chapter 4, and use them to classify image sections within the image. In the sequel, we use the probabilistic output of the classifier as input for further image interpretation.

Using the classifier in a naive way, yields reasonable results, but lack in accuracy and completeness. Therefore, these results will be enhanced by incorporating prior knowledge about typical configurations of facade objects. To be precise, we will evaluate spatial interactions between neighbouring objects, as alignment, size differences, or distances and use them to enforce the detection results to configurations which fit typical facades. Typical configurations might be modelled manually, but this inherently lacks in generality, needs a huge amount of rules, increasing with the number of objects classes, and needs to be well tuned to each type of facade. Therefore, we propose to learn spatial interactions for each combination of neighbouring object classes from training images. This might be restricted to a certain type of facades or not.

The task is to combine evidence from bottom-up for each single object proposal with prior knowledge from top-down. As pointed out in Section 1.4.2, we propose to use a marked point process (MPP) to deal with that task. To recap, a MPP is a random variable whose realisations are configurations of parametric objects. The number of objects is a random variable, too, and needs not to be defined beforehand in contrast to modelling as MRF. This way, we have a large degree of freedom in the definition of parametric objects, the underlying graph structure, and its dimensionality. Through incorporating spatial interactions which we learn from training images, we are able to combine the initial belief from bottom-up image categorisation and the prior knowledge, we have about typical configurations of facade objects.

The whole process is specified by three key elements:

- The object model. We model objects in images as axis-parallel rectangles, thus, points attached with marks for width and height. Additionally, each point is given a class label. This leads to the definition of the configuration space.
- The energy. We model the MPP as Gibbs process; for which we define the distribution in terms of an energy. The energy validates the quality of a configuration according to the image content and the spatial interaction of neighbouring objects in terms of the Markov property.
- An optimisation method. We aim at minimisation the energy which is complex and of varying dimensionality. Therefor, we sample the configuration space by rjMCMC coupled with simulated annealing, as introduced in Sections 2.6.4 and 2.6.7.

5.1.1 The Object Model

We model an image as continuous bounded set $\mathcal{S} = [0, I_R] \times [0, I_C] \subset \mathbb{R}^2$, using the number of rows I_R and columns I_C of the image. Objects in the image, we search for, are modelled as axis-parallel rectangles $\boldsymbol{x} = [x, y, w, h, c]$, with centre point $[x, y] \in \mathcal{S}$, attached with marks $[w, h, c] \in \mathcal{M} \subset \mathbb{R} \times \mathbb{R} \times \mathbb{N}$ for width, height, and class. These rectangles refer to the bounding boxes of our target object classes within the image. As classes we consider $c \in [1, 2, 3]$ standing for window, entrance, and balcony.

We represent the image interpretation as unordered set of of objects $\mathcal{X} = \{\boldsymbol{x}_1 \dots \boldsymbol{x}_n\}$, using $n = N(\mathcal{X})$, the number of objects in the configuration. We consider the MPP $\underline{\mathcal{X}}$, as introduced in Section 2.5.1.3, to determine the configuration that best describe the image.

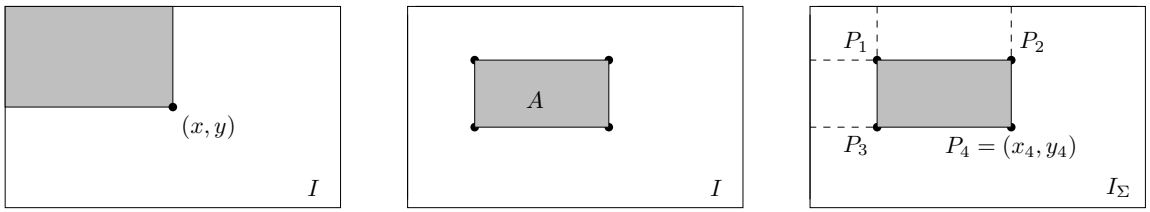


Figure 5.1: **Left:** The value of the integral image at position (x, y) contains the sum of all values in I in the rectangle spanned by $(0, 0) - (x, y)$, thus, the gray area. **Middle:** We seek for the sum of values of I in area A. **Right:** The sum of values of any area A can be derived with 3 summations of values of the integral image I_Σ.

5.1.2 Generating Bottom-up Evidence

Bottom-up, we use the output of a previously designed classifier to capture evidence about an image sections class. For each image section, given by $\boldsymbol{x}$, we extract its feature vector $\boldsymbol{h}$, as described in Section 4.4.3, and aim at an initial belief about its class.

For further interpretation from top-down, the initial belief must be appraisable in terms of its confidence. Therefor, we need reliable a posteriori probabilities. For that reason, we prefer IVM over SVM, as already argued in Section 2.4.4.2 and Section 4.6.1.

To complete the procedure: given annotated training images, we learn shapelets from training images, cf. Chapter 4. We collect the histograms of shapelets, as described in Section 4.4, from annotated image sections for each class $c = 1 \dots C$, and train an IVM for all object classes. Given a new, previously unseen image, we detect shapelets and use the classifier to produce a posteriori probabilities $P(c \mid \boldsymbol{h}(\boldsymbol{x}))$ for proposed image sections.

In our optimisation procedure, using MCMC sampling, classification of image sections is done many million times. To evaluate the classifier, we must be able to process the descriptor of shapelets for each and every position and size of image sections in the image. To allow the image processing in feasible time, we adopt the idea of integral images to the use of integral histograms.

5.1.3 Integral Histograms

In 2004 Viola and Jones introduced integral images for the task of object detection in images. The integral image is a special representation of an image which allows us a fast calculation of image features. Their idea is easily adapted to the use of integral histograms, which allow us to determine the histograms of shapelets, for an arbitrary image section, in constant time.

Integral Images. Given an image I, the integral image I_Σ contains at any position (x, y) the sum of all values of I before (x, y), which is the shaded area in Figure 5.1, left. Informally, this is given by

$$I_\Sigma(x, y) = \sum_{i \leq x} \sum_{j \leq y} I(i, j) \ . \tag{5.1}$$

Given the integral image, we may derive the sum of values A, within any axis parallel rectangular region in I, e.g., the shaded area in Figure 5.1, middle, by three additive operations

$$A = \sum_{i=x_1}^{x_2} \sum_{j=y_1}^{y_3} I(i, j) = I_\Sigma(P_4) - I_\Sigma(P_2) - I_\Sigma(P_3) + I_\Sigma(P_1) \ , \tag{5.2}$$

visualised in Figure 5.1, right. Thus, by building the integral image, which needs only one pass over the image, we may evaluate such sums in constant time for any region in the image.

This allows us to implement sliding window approaches efficient, as it speeds up the needed operations tremendously.

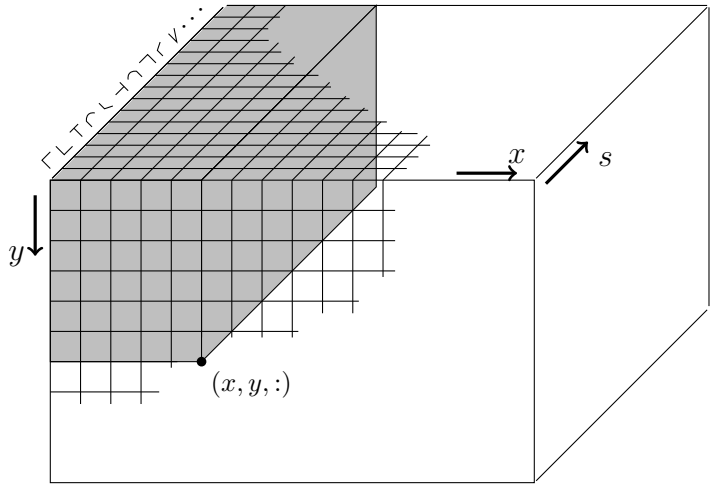


Figure 5.2: Integral histogram H_Σ.

Algorithm 5.1: Function `IntegralHistogram`, assuming a function `HistogramOfShapelets` for building the histogram of shapelets. The algorithms recursively builds the integral histogram H_Σ by one run through the image.

In: Image I, shapelets $\boldsymbol{S}$
Out: Integral histogram H_Σ

```
1  [X,Y] = sizeof(I)
2  for x=1:1:X do
3      for y=1:1:Y do
4          h = HistogramOfShapelets(x, y, S)
5          if x == 1  &  y == 1 then
6              H_Σ(x, y, :) = h
7          else if x==1 then
8              H_Σ(x, y, :) = H_Σ(x, y − 1, :) + h
9          else if y==1 then
10             H_Σ(x, y, :) = H_Σ(x − 1, y, :) + h
11         else
12             H_Σ(x, y, :) = H_Σ(x − 1, y, :) + H_Σ(x, y − 1, :) − H_Σ(x − 1, y − 1, :) + h
13         end
14     end
15 end
16 return
```

Adding one Dimension to Build an Integral Histogram. The idea of integral images is easily transferred to integral histograms. Instead of building the cumulative sum of intensity values, we sum up the occurrences of image features. For this, the integral histogram H_Σ consist of several layers, one for each feature of the codebook, shapelets $\boldsymbol{s} \in \boldsymbol{S}$ in our case, cf. Figure 5.2. Each layer s contains at any position $H_\Sigma(x, y, s)$ the number of all features of type s before (x, y).

The algorithm for building the integral histogram is given in Algorithm 5.1. At any point of the given image it adds up the histogram of shapelets $\boldsymbol{h}$ at point (x, y) to all previously seen histograms, already contained in H_Σ. Please note that we use a Matlab inspired notation, which denotes by the colon operator ':' all values of a matrix within a certain dimension. If S is the number of bins, i.e. shapelets, the vectors $\boldsymbol{h}$ are of size $S \times 1$, thus, if the image is of size $I_C \times I_R$, the integral histogram is of size $I_C \times I_R \times S$.

For querying the histogram of shapelets, within a rectangular image section A, as sketched in Figure 5.1, $3S$ additions are needed

$$\boldsymbol{h}(A) = H_\Sigma(x_4, y_4, :) - H_\Sigma(x_2, y_2, :) - H_\Sigma(x_3, y_3, :) + H_\Sigma(x_1, y_1, :) \ . \tag{5.3}$$

All operations, needed for building the integral histogram, as well as for querying the histogram values, for certain regions of the image, are additive. Thus, the integral histogram inherently works for both strict and soft assignment of matched features to their according bins.

Before going into details of our image interpretation approach, we describe the naive object detection, using a sliding window approach, and show some preliminary results which proves the generation of bottom-up evidence.

5.1.4 Object Detection Using a Sliding Window Approach

When having learned an object classifier, the most intuitive and most common way for object detection is the use of a sliding window approach. Therefor, we slide a window of size $w \times h$ over the whole image. Thereby, the size of the search window varies in both directions to allow for different scales and aspect ratios. For each position within the image, and for each size of the search window, we evaluate the classifier, to ask for the class of the according image patch. We obtain a map of scores, in term of a posteriori probabilities $P(c \mid \boldsymbol{h}(\boldsymbol{x}))$ per class, scale, and aspect ratio. We take each local maximum, fulfilling $P(c \mid \boldsymbol{h}(\boldsymbol{x})) > \frac{C-1}{C}$ as raw detection. As result, we obtain a number of mutually overlapping object detections, which we thin out in a greedy way: Recursively, we always take the object with highest a posteriori probability and delete all overlapping objects. We proceed with the next best detection until there are no more overlapping detections. Some results of sliding window object detection are given in Figure 5.3. The detection rate for windows and balconies is high, while its accuracy in size and position is inferior. We observe some false positives and missing detections especially for entrances. The latter might be detected in the first phase, but probably were eliminated by stronger window evidence. We obviously may improve these results by introducing high-level knowledge as alignment, size of neighbouring objects, or spatial distribution of certain objects, e.g., that entrances are typically located at the bottom of the image.

Therefore, we continue in Section 5.2 with the description of the MPP. We introduce the density and the according energy of the MPP. We detail the parts of the energy formulation, show how we model interactions of neighbouring objects, and how we learn typical configurations. In Section 5.3 we detail the optimisation using rjMCMC, particularly the types of proposition kernels we use to explore the configuration space. We discuss related work in Section 5.4. In Section 5.5 we present experiments and discuss the results.

5.2 An Energy for Facade Image Interpretation

We model a 2D image as configuration of objects which are points living in $\mathcal{S} \subset \mathbb{R}^2$, attached with marks living in a space $\mathcal{M}$. A random configuration $\underline{\boldsymbol{X}}$, on the product space $\mathcal{S} \times \mathcal{M}$, is a MPP, as introduced in Section 2.5.1.3. We denote the set of object configurations

$$\Omega = \cup_{n=0}^{\infty} \Omega_n \qquad \text{with} \quad \Omega_n = \{\boldsymbol{x}_1 \dots \boldsymbol{x}_n\} \subset \mathcal{S} \times \mathcal{M} \ , \tag{5.4}$$

thus, the space of all possible configurations of objects.

The most random point process is a homogeneous Poisson process. In Section 2.5.1.4 we have seen that its distribution takes the role of the Lebesgue measure in $\mathbb{R}^d$, when defining the distribution of a point process with respect to a reference Poisson point process. Thus, we define a general point process by specifying its density $f(\cdot)$ with respect to a reference measure π_μ which is the distribution

Figure 5.3: Results of object detection using the sliding window approach. We yield a reasonable detection rate, but accuracy of detected objects is low. **Colours:** blue - window, orange - entrance, green - balcony.

of the dominating Poisson process with intensity $\mu(\cdot)$, such that the distribution of the point process is given by

$$F(A) = \int_A f(\mathcal{X})\pi_\mu(\mathrm{d}\mathcal{X}) \ . \tag{5.5}$$

Moreover, we model the objects as Markov marked point process and express $f(\cdot)$ in terms of a Gibbs energy $U(\mathcal{X})$ such that

$$f(\mathcal{X}) = \frac{1}{Z} e^{-U(\mathcal{X})} \ . \tag{5.6}$$

Instead of maximising $f(\mathcal{X})$ we minimise the energy $U(\mathcal{X})$. The advantages are obvious. We get rid of the normalising constant Z. Further, the energy needs not be probabilistic. It is a matter of design to express the fitness of the configuration according the given image and expected configurations. We will introduce our energy formulation in the next section.

5.2.1 Energy Formulation

We express the point process density in its Gibbs form

$$f(\mathcal{X}) \propto \exp^{-U(\mathcal{X})} \ . \tag{5.7}$$

The energy should allow us to evaluate spatial interactions of objects and the consistency of single objects concerning the given image. We express the energy as sum of four terms

$$U(\mathcal{X}) = U_{\text{data}}(\mathcal{X}) + \lambda_1 \ U_{\text{geom}}(\mathcal{X}) + \lambda_2 \ U_{\text{conf}}(\mathcal{X}) + \lambda_3 \ U_{\text{num}}(\mathcal{X}) \ , \tag{5.8}$$

where U_{data} is the unary energy or the data term which expresses the local energy of all single objects in the current configuration $\mathcal{X}$. In our case, given the output of a classifier, it measures how well the objects fits the image content. This energy will be designed such that attractive objects contribute negative energies, whereby, we call an object attractive, if its a posteriori probability is high. We denote U_{geom} the geometric energy and U_{conf} the configuration energy which are given by prior knowledge from training data about typical geometry of objects or object pairs. The former expresses an additional unary energy in terms of the objects size and location. The latter expresses the energy of neighbouring objects given by their spatial interaction. The structure of prior energies will be designed such that configurations not fitting the learned configuration statistics get positive energies, thus, get punished. Finally, U_{num} represents an additional prior on the number of each classes objects. In contrast to the basic idea of a point process which is guided by the reference density of the Poisson process, whose intensity reflects the accepted mean number of objects, this is an alternative model for the prior. Due to the different mean numbers of each classes objects, one underlying reference measure is not suited to reflect these numbers, therefore we decide to introduce this as a more adequate prior.

Finally, the single terms are weighted relative to each other by positive constants λ_i. We detail each term in the next subsections.

5.2.2 The Data Term

As stated above, we use the output of a classifier to capture evidence from bottom-up, from which we obtain an initial belief about its class. The data term uses the output of the classifier to assign an energy to each object of a configuration. Thereby, we consider an object, $\boldsymbol{x} \in \mathcal{X}$, having class label $c \in \{1 \dots C\}$, attractive if its a posteriori probability $P(c \mid \boldsymbol{h}(\boldsymbol{x}))$ is large, preferably near 1. Vice versa, such objects should contribute a local unary energy $\Phi(\boldsymbol{x}) < 0$ to the overall energy. Thanks to the Markov property and due to the theorem of Hammersley-Clifford, the overall data energy is given by the sum of each objects local energy

$$U_{\text{data}}(\mathcal{X}) = \sum_{\boldsymbol{x}_i \in \mathcal{X}} \Phi(\boldsymbol{x}_i) \ , \tag{5.9}$$

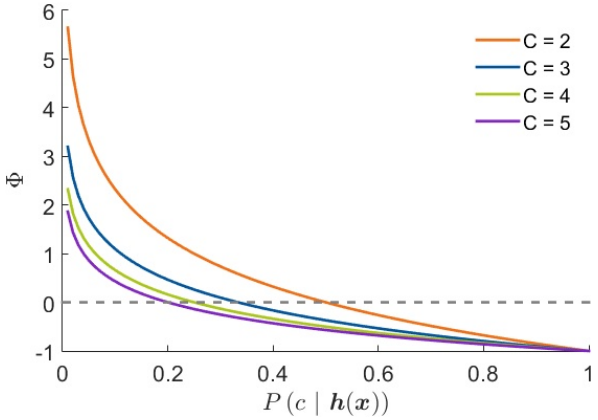


Figure 5.4: The reward function $\Phi(\boldsymbol{x})$ for different numbers of classes C, which gives the local unary energy for an object $\boldsymbol{x}$ depending on the a posteriori probability $P(c \mid \boldsymbol{h}(\boldsymbol{x}))$, as given in Equation (5.10).

whereby, we use the reward function

$$\Phi(\boldsymbol{x}) = -\log_C P\left(c \mid \boldsymbol{h}(\boldsymbol{x})\right) - 1, \quad \Phi \in [\infty, -1] \tag{5.10}$$

to express the object's local energy, which is visualised for different numbers of classes C in Figure 5.4. Please note, the number of classes C as basis of the logarithm. This way, we ensure probable samples to get a negative local energy, while objects with rather uncertain class labels or objects even not belonging to the given class, thus, having a posteriori probability below $1/C$, get positive local energy. Nevertheless, the decrease of local energy is bounded, which is important to ensure stability of the Markov chain, cf. Section 2.5.2. Obviously, the overall data energy might decrease infinitely by superimposing infinite copies of an attractive object $\boldsymbol{x}$. We avoid this by including a strong penalty term for overlapping objects in the pairwise prior energy, see below.

5.2.3 Prior Energy

The prior energy terms evaluate different aspects of the configuration, concerning the geometry of each single object or the geometric relation of neighbouring objects. In the following, we introduce our configuration model to describe these neighbourhood relations and derive the according terms of the prior energy.

5.2.3.1 The Configuration Model

We learn typical configurations of objects from training images. Thereby, we characterise a configuration by properties of single objects, as width w, height h, and location x and y, and properties of neighbouring objects as intersection, distance, alignment, and size differences.

Interacting Objects. Usually, the neighbourhood relation $\boldsymbol{x}_i \sim \boldsymbol{x}_j$ is defined in terms of a distance, such that all objects within a ball of given radius are said to be neighboured. In our application it is not possible to define such a fixed distance, it would differ from image to image and if the number of objects is low, objects even could be on opposite sides of the image. Therefore, we define the neighbourhood relation in terms of a Voronoi diagram. We reduce all rectangles to their centre points and evaluate the according adjacency graph. Objects neighboured within the adjacency graph are said to be neighbours in image space. The reason not to use the exoskeleton of the objects themselves is that at high temperature the objects might overlap, thus, get merged to one component for processing the exoskeleton.

We define different measures on interacting objects to describe the overall configuration. Not all of them are taken into account for each neighbouring object pair, e.g., we do not measure the vertical

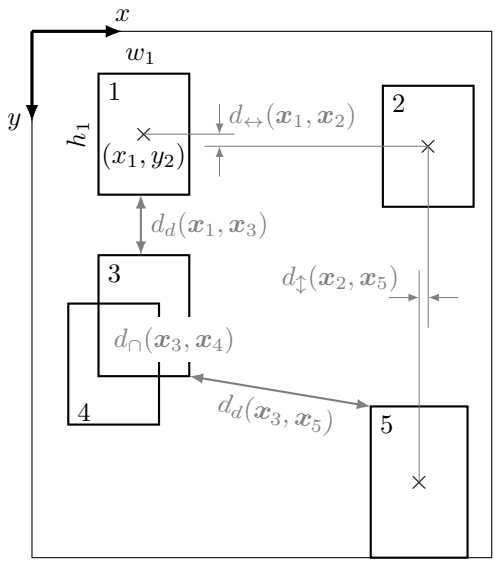


Figure 5.5: Properties of neighbouring objects to describe the configuration.

alignment for objects neighboured horizontally. In the following, we first describe the interacting measures and then detail, which measures are taken into account, under which constraints.

Interacting distances. Assuming the image scale m [px/m] to be known, we take into account the following properties of interacting objects $\boldsymbol{x}_i \sim \boldsymbol{x}_j$, cf. Figure 5.5:

- intersection area

$$d_{\cap}(\boldsymbol{x}_i, \boldsymbol{x}_j) = \frac{\text{area}\,(\boldsymbol{x}_i \cap \boldsymbol{x}_j)}{\min(w_i \cdot h_i, w_j \cdot h_j)} \qquad d_{\cap} \in [0, 1] \tag{5.11}$$

  Two objects not intersecting at all, get an intersection area of 0, while two objects superimposed get an intersection area of 1, cf. Figure 5.6

- minimal distance $d_d(\boldsymbol{x}_i, \boldsymbol{x}_j)$ which is given by the minimal distance between any two points of the objects bounding box, normalised by the image scale m.

- alignment horizontal and vertical, referred to the centre of objects

$$d_{\leftrightarrow}(\boldsymbol{x}_i, \boldsymbol{x}_j) = \frac{|y_i - y_j|}{m} \qquad d_{\updownarrow}(\boldsymbol{x}_i, \boldsymbol{x}_j) = \frac{|x_i - x_j|}{m} \tag{5.12}$$

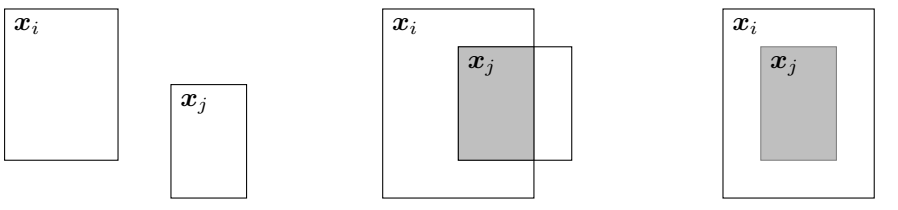


Figure 5.6: Intersection distance. **Left:** The objects do not intersect; thus, the intersection distance gets 0. **Middle:** The objects intersect partially; the intersection distance, normalised by the smaller area, gets 0.6. **Right:** Worst case, the object lay on top of each other; the intersection distance gets one.

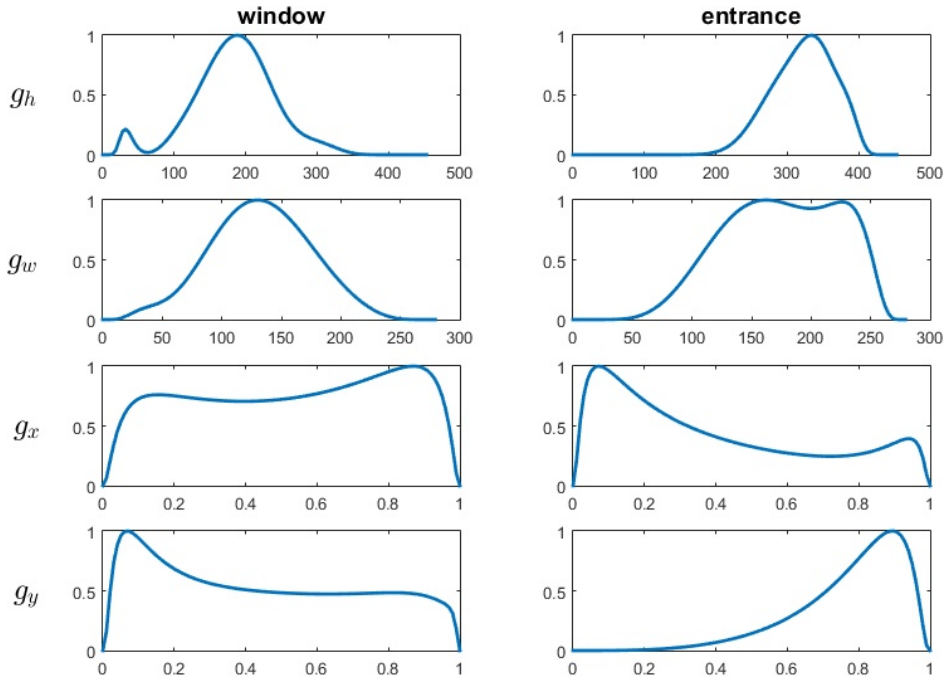


Figure 5.7: Unary statistics collected from training images from the Basel image collection, cf. Section 5.5.1. Unit of x-axis for g_h and g_w is cm. g_x and g_y are normalised to the images width, thus, they are dimensionless. The histograms are smoothed using a Gaussian kernel density estimator, with kernel widths from top to bottom: 0.3, 0.5, 1.0, 0.9. Please note the different ranges of values. The large kernel width for g_x and g_y result from the almost unary distributed data.

- size difference in height and width

$$d_{\Delta h}(\boldsymbol{x}_i, \boldsymbol{x}_j) = \frac{|h_i - h_j|}{m} \qquad d_{\Delta w}(\boldsymbol{x}_i, \boldsymbol{x}_j) = \frac{|w_i - w_j|}{m} \ . \tag{5.13}$$

In order to shorten the notation, we denote $d_k^{(i,j)} := d_k(\boldsymbol{x}_i, \boldsymbol{x}_j)$ the value of the k-th distance between two objects $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$, i.e. $k \in \{\cap, d, \leftrightarrow, \updownarrow, \Delta h, \Delta w\}$.

As already mentioned, we do not take into account all distances for all interacting objects. To denote this special behaviour, we specialise the neighbourhood relation to $\boldsymbol{x}_i \sim^{d_k} \boldsymbol{x}_j$, to declare objects $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ neighbours with respect to distance d_k. Therefor, we define the following neighbourhood relations

$$\begin{aligned} \boldsymbol{x}_i \sim^{d_\leftrightarrow} \boldsymbol{x}_j \ , \ \boldsymbol{x}_i \sim^{d_{\Delta h}} \boldsymbol{x}_j \quad \text{if} \quad d_\leftrightarrow^{(i,j)} \leq d_\updownarrow^{(i,j)} \\ \boldsymbol{x}_i \sim^{d_\updownarrow} \boldsymbol{x}_j \ , \ \boldsymbol{x}_i \sim^{d_{\Delta w}} \boldsymbol{x}_j \quad \text{if} \quad d_\leftrightarrow^{(i,j)} > d_\updownarrow^{(i,j)} \ . \end{aligned} \tag{5.14}$$

This way, we take into account the size difference in height and the horizontal misalignment, only if the objects are roughly beside each other. Vice versa, we take into account the size difference in width and the vertical misalignment, only if the objects are roughly on top of each other. Further, the neighbourhood relations $(\sim^{d_{\Delta h}}, \sim^{d_{\Delta w}})$ and $(\sim^{d_\leftrightarrow}, \sim^{d_\updownarrow})$, respectively, are pairwise exclusive each, i.e. two neighbouring object can interact only in terms of one of them.

5.2.3.2 Learning the configuration statistics

Given annotated training data, we collect all bounding boxes of our target classes, and collect for all annotated objects their location (x, y), width w, and height h, and for all interacting objects distances d_d and $d_\cap$. Further, for each possible combination of object classes, we store distances

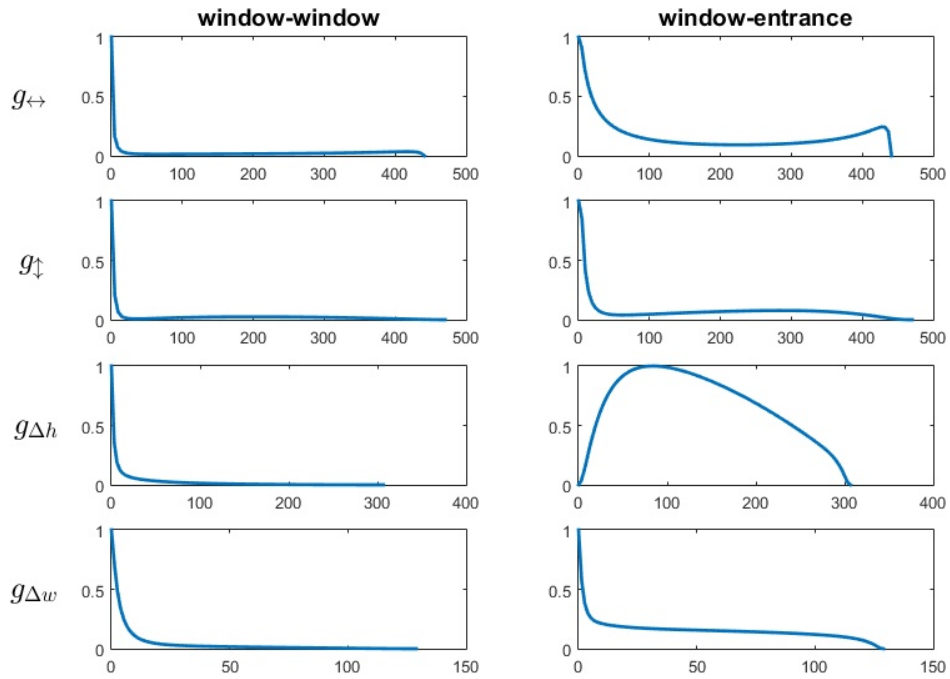


Figure 5.8: Pairwise statistics $g(d_k)$ collected from training images from the Basel image collection, cf. Section 5.5.1. Unite of x-axis is cm. Again, the histograms are smoothed using a Gaussian kernel density estimator, with kernel widths from top to bottom: 1.5, 0.9, 1.0, 1.0.

$d_{\leftrightarrow}, d_{\updownarrow}, d_{\Delta h}$, and $d_{\Delta h}$ for all pairs of objects, interacting in terms of the according neighbourhood relation and according class combination. We represent these statistics by their histogram which we denote $g_k(\cdot \mid \cdot)$, $k \in \{x,\ y,\ w,\ h,\ \leftrightarrow,\ \updownarrow, d_{\Delta h}, d_{\Delta w}\}$. We smooth the histograms using a kernel density estimator with Gaussian kernel. Its kernel width is determined automatically, under the assumption of normal distributed data, whereby, we choose for each parameter the same kernel width for all classes and pairs of classes, respectively. To enable scoring of these properties, we normalise each histogram, such that $\max(g) = 1$.

The minimal distance between neighbouring objects is an exception: We just store the minimal value $t_d\,(c_1, c_2)$, we have seen during training according to each possible combination of object class c_1 and c_2. An example is given in Figures 5.7 and 5.8. There, we collected the configuration statistics from training images taken in the city of Basel, cf. Section 5.5.1. These facades contain no balconies; thus, we merely observe windows and entrances in images. Figure 5.7 visualises the unary statistics about the objects height, width, and location. We observe that windows are almost equally distributed over the image in x- and y-direction, respectively. At the borders of the images the probability of observing a window is lower, which results from the fact that the images show exactly one facade solely, in most cases. Further, from g_y we note that the probability of observing an entrance is highest at the lower part of the image, as we expect in typical facades.

Figure 5.8 visualises the pairwise statistics collected from the images. We observe that windows are almost perfectly aligned horizontally and vertically. Further, neighbouring windows are of same size in most cases. On the other hand, entrances and windows are more likely misaligned and of different size.

5.2.3.3 Prior Energy Terms.

Having learned these statistics, we now define the prior energy. It is represented by an unary term $U_{\text{geom}}(\mathcal{X})$ and a pairwise term $U_{\text{conf}}(\mathcal{X})$, which represent the confidence of proposed samples to the learned statistics over size and location of single objects and the local arrangement of neighboured

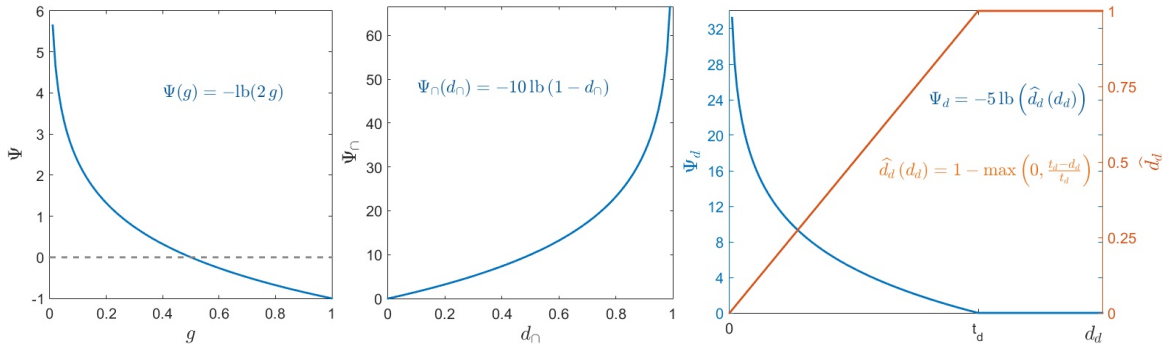


Figure 5.9: Penalty functions used in the individual terms of the prior energy. **Left:** Penalty function (5.16). Values of g near 1 are favourable and contribute a negative energy, while values of g below 0.5 get punished and contribute a positive energy. **Middle:** Penalty function (5.18) of the intersecting distance, using $a = 10$. The more two objects overlap the large $d_\cap$ the larger the increase of the energy. **Right:** Blue: Penalty function (5.19) regarding the distance of two objects with respect to the minimal distance t_d, using $b = 5$. Orange: adapted hat-function which transforms the distance d_d to a helping value $\widehat{d}_d$ which is in $[0, 1]$, such that distance above t_d contribute a constant value of 1. Vice versa the according penalty function increases the energy just for distances below t_d.

objects. The former is given by the sum of local energies of single objects

$$U_{\text{geom}}(\mathcal{X}) = \sum_i \sum_{k \in \{x,y,w,h\}} \Psi\left(g_k\left(\boldsymbol{x}_i \mid c_i\right)\right) \ , \tag{5.15}$$

using the penalty function

$$\Psi(g) = -\mathrm{lb}\left(2 \cdot g\right) \ , \quad \Psi \in [\infty, -1] \, . \tag{5.16}$$

Thus, for each object we sum up the contributions of each part of the configuration model. Using Equation (5.16), which is shown in Figure 5.9 left, objects not fitting the learned function g_k get punished and contribute a positive energy, while objects near the global maximum of g_k, which is 1, even contribute a negative energy. This way, we are able to detect objects whose data energy is low, but fit the learned configuration statistics.

We denote $\mathcal{E} = \{e_{ij} \mid x_i \sim x_j\}$ the set of all neighbouring objects. We may also say it is the set of edges e_{ij}, when representing the configuration $\mathcal{X}$ as graph. Summing over all edges, the prior energy regarding interacting object pairs is given by

$$U_{\text{conf}}(\mathcal{X}) = \sum_{(i,j) \in \mathcal{E}} \left(\Psi_\cap\left(\boldsymbol{x}_i, \boldsymbol{x}_j\right) + \Psi_d\left(\boldsymbol{x}_i, \boldsymbol{x}_j \mid c_i, c_j\right) + \frac{1}{N\left(\mathrm{Ne}(\boldsymbol{x}_i)\right)} \cdot \sum_{k \in \{\leftrightarrow, \updownarrow, \Delta w, \Delta h\}} \Psi\left(g_k\left(\boldsymbol{x}_i, \boldsymbol{x}_j \mid c_i, c_j\right)\right) \right) \tag{5.17}$$

where we use the same penalty function $\Psi(\cdot)$ as before for alignment and size differences, while intersection and distance get punished more strongly. We use a hard core penalty function for overlapping rectangles

$$\Psi_\cap\left(\boldsymbol{x}_i, \boldsymbol{x}_j\right) = -a \cdot \mathrm{lb}\left(1 - d_\cap^{(i,j)}\right) \ , \quad \Psi_\cap \in [\infty, 0] \, , \tag{5.18}$$

using a medium constant, e.g., $a = 10$. The function, which is shown in Figure 5.9 middle, prevents to increase the overall energy by superimposing infinite many objects at medium and low temperature. Nevertheless, it allows small overlaps at height temperature, which is important to explore the state space. If $d_\cap = 0$ the objects do not overlap, then there is no penalty.

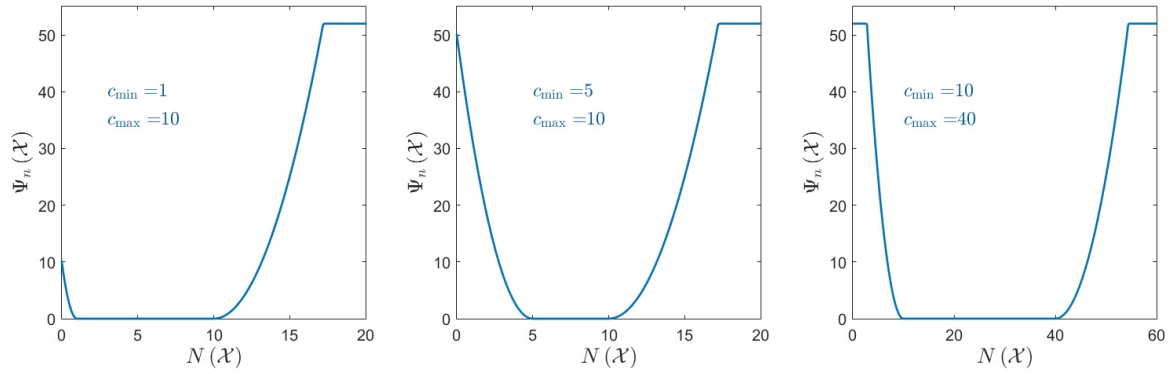


Figure 5.10: Penalty function (5.21) used in the prior U_{num} for the number of objects per class for different values of $c_{\min}$ and $c_{\max}$ and $v = 10$.

To penalise neighboured objects with distance below the class specific minimal distance $t_d(c_i, c_j)$, we use a function similar to the hat-function

$$\Psi_d\left(\boldsymbol{x}_i, \boldsymbol{x}_j \mid c_i, c_j\right) = -b \cdot \text{lb}\left(1 - \max\left(0, \frac{t_d(c_i, c_j) - d_d^{(i,j)}}{t_d(c_i, c_j)}\right)\right) \;,\; \Psi_d \in [\infty, 0]\,, \tag{5.19}$$

again using a medium constant $b = 5$. We visualise this function in Figure 5.9 right, whereby, we split it into two parts, orange showing the argument of the logarithm, blue showing the distance penalty function. Distances slightly below $t_d(c_i, c_j)$ get a small positive energy, which increases with decreasing distance. Distances above $t_d(c_i, c_j)$ do not influence the overall energy.
Again, we sum up for each object the contribution of each interacting distance. To make the configuration energy (5.17) independent on the number of interacting neighbours $\text{Ne}(\boldsymbol{x}_i)$, we normalise by their number $N\left(\text{Ne}(\boldsymbol{x}_i)\right)$.

Finally, in order to define a prior on the number of objects, we store the minimal and maximal numbers $c_{\min}$ and $c_{\max}$, respectively, of objects per class, we have seen in single images during training. Configurations fitting these numbers should not influence the overall energy, while deviations should get a punishment. Therefrom, we define the prior on the number of objects by

$$U_{\text{num}}(\mathcal{X}) = \sum_{y=1\ldots C} \Psi_n\left(\{\boldsymbol{x} \in \mathcal{X} \mid c = y\}\right) \;, \tag{5.20}$$

using

$$\Psi_n\left(\mathcal{X}\right) = \begin{cases} \min\left(-\text{lb}(\varepsilon),\ \frac{v}{c_{\min}}(N\left(\mathcal{X}\right) - c_{\min})^2\right) & \text{if} \quad N\left(\mathcal{X}\right) < c_{\min} \\ \min\left(-\text{lb}(\varepsilon),\ \frac{v}{c_{\max}}(N\left(\mathcal{X}\right) - c_{\max})^2\right) & \text{if} \quad N\left(\mathcal{X}\right) > c_{\max} \\ 0 & \text{otherwise}\;, \end{cases} \tag{5.21}$$

using a constant scaling $v = 10$ and the numerical precision $\varepsilon = 10^{-16}$. Thus, configurations fitting $c_{\min}$ and $c_{\max}$, do not influence the overall energy, while deviations get a punishment which increases with the degree of deviation, but is bounded by a fixed value of $-\text{lb}(\varepsilon)$. Three example plots for different values of $c_{\min}$ and $c_{\max}$ are shown in Figure 5.10. The prior U_{num} is of special interest for low frequent classes, such as entrances. We observed that the regular structure of window grids, in most cases, overvotes the existence of a single entrance, which we avoid by using this prior.

5.3 Optimisation by rjMCMC

Our task is to find the configuration $\mathcal{X}$ which maximises the unnormalised point process density or minimises the energy $U(\mathcal{X})$, respectively. It is a complex function with rough landscape. Even

its dimensionality is unknown due to the unknown number of objects. Therefore, we optimise with rjMCMC coupled with simulated annealing, as introduced in Section 2.6.7, to find the global optimum. Introducing the temperature parameter T, the optimiser is given by

$$\begin{aligned}\widehat{\mathcal{X}} &= \underset{\mathcal{X}}{\operatorname{argmax}}\ f(\mathcal{X})^{\frac{1}{T_t}} = \underset{\mathcal{X}}{\operatorname{argmax}}\ \exp\left(-\frac{U(\mathcal{X})}{T_t}\right) \\ &= \underset{\mathcal{X}}{\operatorname{argmin}}\ \frac{U(\mathcal{X})}{T_t}, \qquad \lim_{t\to\infty} T_t = 0\ . \end{aligned} \tag{5.22}$$

In Section 2.6.6 we introduced the sampling algorithm to optimize MPPes, which we apply using two types of moves:

1. dimensional jumping transformation: birth and death,
2. non jumping transformations: translation, dilation, switching.

The latter randomly selects an object from the current configuration and randomly perturbs its marks.

The generic point process sampler algorithm is given in Algorithm 2.5, page 86. Having M different moves, we take the proposition kernel of the Markov chain as a mixture of $m = 1 \dots M$ proposition distributions, each having a probability $j_m(\mathcal{X})$ to choose move type m being at $\mathcal{X}$. The kernel $\mathcal{Q}_m$ can be interpreted as instruction, how to throw a new sample $\boldsymbol{x}$ being at $\mathcal{X}$, using move type m which we define in more detail in the following.

We set $\mathcal{S} = \{[1, I_C] \times [1, I_R]\}$, using the image's number of rows I_R and columns I_C, and $\mathcal{M} = \{[W_{\min}, W_{\max}] \times [H_{\min}, H_{\max}] \times [1, C]\}$, using ranges $[W_{\min}, W_{\max}]$ and $[H_{\min}, H_{\max}]$ for width and height, respectively, taken from the training sample's sizes. In order to clarify the notion, we denote the move types not by numbers. We use $m \in \{\mathrm{b}, \mathrm{d}, \mathrm{nj}\}$ to denote birth, death and non jumping moves, whereby, the latter comprises different move types leading to the same Green ratio.

5.3.1 Birth and Death.

In case of birth, we create a new object $\boldsymbol{x} \in \mathcal{S} \times \mathcal{M}$ and propose $\mathcal{Y} = \mathcal{X} \cup \boldsymbol{x}$. Noting

$$\frac{f(\mathcal{Y})}{f(\mathcal{X})} = \frac{\mathrm{e}^{-U(\mathcal{Y})}}{\mathrm{e}^{-U(\mathcal{X})}} = \mathrm{e}^{U(\mathcal{X})-U(\mathcal{Y})}\ , \tag{5.23}$$

introducing the temperature parameter T_t of simulated annealing, and taking $j_b = j_d$ we obtain Green's ratio for birth and death kernels

$$R_{\mathrm{b}}(\mathcal{X}, \mathcal{Y}) = \frac{\mu(\mathcal{S})}{N(\mathcal{Y})}\ \mathrm{e}^{\left(\frac{U(\mathcal{X})-U(\mathcal{Y})}{T_t}\right)} \tag{5.24}$$

$$R_{\mathrm{d}}(\mathcal{X}, \mathcal{Y}) = \frac{N(\mathcal{X})}{\mu(\mathcal{S})}\ \mathrm{e}^{\left(\frac{U(\mathcal{X})-U(\mathcal{Y})}{T_t}\right)} \tag{5.25}$$

We do not sample $\boldsymbol{x} \sim Q_b = \frac{\mu(\mathrm{d}\boldsymbol{x})}{\mu(\mathcal{S})}$ as in common birth and death algorithm. Instead, we sample new objects data-driven; thus, by normalizing the smoothed histograms $g_k,\ k \in \{x, y, w, h\}$, such that they represent pmfs, we sample the parameters of a new objects from these pmfs. This results in a non homogeneous reference Poisson measure, but Descombes et al. (2009) show that this has no impact on the convergence to the global minimum of the energy function. There is no effect on Green's ratio nor does it effect convergence of the Markov chain, but its speed, as we avoid to propose lots of useless objects.

5.3.2 Non Jumping Transformations.

In case of non jumping moves, we randomly perturb the marks of an existing object. To do so, we uniformly select an object $\boldsymbol{x} \in \mathcal{X}$ and throw random numbers $u \sim Z_{(\mathcal{X},u)}$ according a suitable distribution. Given a function $\phi : \boldsymbol{y} = \phi(\boldsymbol{x}, u)$ which transforms the object we propose $\mathcal{Y} = (\mathcal{X} \setminus \boldsymbol{x}) \cup \boldsymbol{y}$.

Translation: manipulates the centre of an object $\boldsymbol{x}$.
We throw $\boldsymbol{u} \in \mathbb{R}^2$

$$\boldsymbol{u} = \begin{bmatrix} u_x \\ u_y \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} 0 \\ 0 \end{bmatrix}, T \begin{bmatrix} (\Delta x)^2 & 0 \\ 0 & (\Delta y)^2 \end{bmatrix} \right)$$

$$\phi : \boldsymbol{y} = \phi(\boldsymbol{x}, \boldsymbol{u}) = [x + u_x,\ y + u_y,\ w,\ h,\ c]^\mathsf{T} \ ,$$

Using control parameters Δx, Δy which we fix, such that the standard deviation of the Gaussian is $1/8$ of the image's height and width, respectively, at starting temperature T_0. The variances of the Gaussian get smaller as the temperature T decreases, thus, the proposed translations get smaller.

Dilation: manipulates width and height of an object $\boldsymbol{x}$.
We throw $\boldsymbol{u} \in \mathbb{R}^2$

$$\boldsymbol{u} = \begin{bmatrix} u_w \\ u_h \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} 0 \\ 0 \end{bmatrix}, T \begin{bmatrix} (\Delta w)^2 & 0 \\ 0 & (\Delta h)^2 \end{bmatrix} \right)$$

$$\phi : \boldsymbol{y} = \phi(\boldsymbol{x}, \boldsymbol{u}) = [x,\ y,\ w + u_w,\ h + u_h,\ c]^\mathsf{T} \ .$$

Using control parameters Δw, Δh which we fix again, such that the standard deviation of the Gaussian is $1/8$ of the image's height and width, respectively, at starting temperature T_0.

Switching: changes the label of an object $\boldsymbol{x}$.
We throw $u \in \mathbb{N},\ u \sim \mathcal{U}(\{1, \ldots, C\} \setminus c)$; thus, we throw a random number out of C classes which is not the current one c.

$$\phi : \boldsymbol{y} = \phi(\boldsymbol{x}, u) = [x,\ y,\ w,\ h,\ u]^\mathsf{T} \ .$$

In each case the inverse transformation $\boldsymbol{x} = \phi^{-1}(\boldsymbol{y}, \boldsymbol{u})$ exist and is as possible as ϕ is, which ensures reversibility. Thus, if the translation and its inverse are equally probable and the according perturbation variables where thrown from the same distribution, Green's ratio simplifies to

$$R_{\text{nj}}(\mathcal{X}, \mathcal{Y}) = \mathrm{e}^{\left(\frac{U(\mathcal{X}) - U(\mathcal{Y})}{T_t}\right)} \ , \tag{5.26}$$

which completes Algorithm 2.5, page 86.

Exceptions. Impossible proposals for moves, i.e. $\boldsymbol{y} \notin \mathcal{S} \times \mathcal{M}$, e.g., if a translation or dilation proposal moves the object out of the image, are not used. In that case we reject the according proposition, without any impact on the invariant distribution as pointed out by Ortner et al. (2003).

5.4 Related Work

In our work we are dealing with Marked Pointed process which were introduced by Baddeley and Lieshout (1993) to the field of stochastic geometry. In conjunction with reversible jump MCMC methods, introduced by Green (1995) and Geyer and Møller (1994), they become popular for several tasks of image processing and interpretation. In statistics these methods, among others, were used for Bayesian analysis of complex distributions as mixtures with an unknown number of components (Richardson and Green, 1997; Stephens, 2000). In the following, we will focus on related work from image processing.

To the best of our knowledge, around 2001 the group at INRIA around Josaine Zerubia and Xavier Descombes, started to introduce the topics of stochastic geometry and point process theory to the field of image processing in terms of structure extraction and object detection. They aim at the detection of buildings and road networks in digital aerial images (Garcin et al., 2001; Lacoste et al., 2002; Descombes et al., 2001) or rectangular road markings (Tournaire et al., 2007).

Ortner et al. (2003) continue the work on detecting parametrised objects and introduce a proposition kernel that allows to sample objects in the neighbourhood of existing objects. Ortner et al. (2007, 2008)

use this technique for the extraction of building footprints from altimetric data in dense urban areas, and the extraction of road networks and buildings from digital elevation models, which is done again by Tournaire et al. (2010) in a more efficient way to speed up the process. Lafarge et al. (2010) extend this work to 3D and proposed to reconstruct building from DSM based on a library of 3D models.

These model suffer from a lack of generality, they aim at specific applications and the complexity of interactions between the objects does not generalise to another application. Most of them rely on many tunable parameters. Therefore, Lafarge et al. (2010) propose a more generalised MPP called multi-marked point process, which can be applied to a large range of applications without changing the underlying model. Verdie and Lafarge (2013) build on former approaches, but aim at their large-scale applications by introducing an efficient parallelisation scheme.

Chai et al. (2012) propose a hybrid representation of MRFs and MPPs to represent both low-level information and high-level knowledge to provides a structure-driven approach for detecting buildings in aerial images. At high-level, they construct a MPP of rectangles to represent the buildings on ground scene. At the low-level, they use a MRF to represent the statistics of the image appearance based on the histograms of colours to represent the buildings appearance. They use rjMCMC to explore the configuration space at high-level, and Graph Cut to optimise configurations at low level. To provide the results to each others level, they propose a top-down and a bottom-up scheme to communicate the results between these levels.

In the area of structure detection Lacoste et al. (2005) and Sun et al. (2007) extract line-networks from images, the former in terms of roads or hydrographic networks from remotely sensed images, the latter in terms of vessel trees from medical images. Both propose spatial interactions between lines based on overlapping, alignment and connection considerations, whereby, they force certain types of line junctions which are more likely to appear in real networks.

Chai et al. (2013) introduce the so-called junction point processes for the extraction of line-networks. Therefor, they propose a graph-based representation of line segments with junction points marked by angles indicating the directions of the adjacent points and formulate the configuration of junction-points as a planar graph.

Another interesting application is population counting, shown by Descombes et al. (2009), where they aim at tree crown extraction and bird detection, both parametrised as disk in the plane, and thereby, count the number of detected objects. Similarly, Ge and Collins (2009) perform crowd detection, i.e. counting people, from single- and multi-view images. They represent objects as set of body shapes which they learned from training data. Utasi and Benedek (2011) similarly detect people in 3D from multi-view images, whereby, they represent objects as cylinders.

In the field of texture analysis Nguyen et al. (2010) develope a model for texture recognition. They regard a texture as the realisation of a MPP of visual keypoints and use the descriptive statistics of this MPP to form a set of texture descriptors. In other words the spatial distribution of visual keypoints discriminates the textures.

Lafarge and Gimel'farb (2008); Lafarge et al. (2010) provide a general model for extracting different types of geometric features from images, as line, rectangles and disks. They use a mixture of object interactions to reconstruct a large variety of textures. Therefor, they introduce a library of geometric models which act as primitive elements of a description of texture images. Furthermore, they propose to use the so-called jump-diffusion dynamic as alternative to the commonly used rjMCMC sampling with simulated annealing. It adds to the reversible jumps of the MCMC process a stochastic diffusion dynamic within each continuous subspace. At high temperature, the diffusion performs large random steps to avoid trapping into local optima, while at low temperature it acts as gradient descent.

Jump-diffusion is an interesting development to speed up convergence tremendously and to increase the accuracy of the final solution. But, it assumes the energy landscape to be smooth near the optimum, and further an energy for which we can evaluate the gradient efficiently. Both is not given in our application: the energy landscape is rough and the energies gradient can not be evaluated analytically. We may obtain the gradient from numerical differentiation, but this is slow and contradicts the desired speed-up. Therefrom, we do not use jump-diffusion.

We are aware of two works, dealing with MPPs in the context of facade image interpretation. In

(a) Basel old town

(b) Basel row houses

(c) City houses with balconies

Figure 5.11: Sample images from used datasets.

order to detect openings in facades, as indicator for the classification of blind facades, Burochin et al. (2014) formulate a stochastic process to sample rectangles in images polygons, formerly identified as facades, from oblique aerial images. Do to perspective distortions the resolution of rectified images is low in vertical direction, images are noise and contain shadows. Given these conditions, they define openings as dark rectangles with respect to the wall background. As the detection results are dedicated as indicator for the classification of blind facades, beside other features, they obviously do not require complete and precise detections. Their energy model consist, as usual, of a data term and a prior term, weighted by a regularisation factor. The data term depends on the gradient information on the edges of proposed rectangles, while the prior term solely penalizes overlapping rectangles and does not consider any other configuration constraints. Optimization is done by rjMCMC with simulated annealing, using birth and death, split and merge and edge translation as non-jumping kernels.

Wang et al. (2015) aim at the detection of window grids in images using a marked point process. They propose structure-driven sampling in order to yield the assumed grid structure of windows and use an energy formulation whose data term depends on a probability map given by a pixelwise classification and whose prior term consist on a repulsive term scoring interacting rectangles in terms of their horizontal and vertical distance. Their results are not convincing compared to state-of-the-art results, but to the best of our knowledge, this is the first approach dealing with marked point processes in the context of facade image interpretation. We also use the concept data-driven sampling via marked point process.

5.5 Experiments

This section evaluates the proposed MPP for facade interpretation. The goal is to show that our approach is suited for facade image interpretation, thus, for the complete and accurate detection of

facade objects. We learn the model on different datasets, showing different characteristics. To prove the learned prior model, we simulate configurations, ignoring bottom-up evidence from image data. We visualise interpretation results and provide a qualitative pixelwise evaluation.

5.5.1 Datasets

From eTrims image database (Korc and Förstner, 2009), we assembled different image collections which are characterised by similar facade structure and as similar as possible appearance of addressed facade objects. The goal is to show that we are able to learn a model from few images that we can use to evaluate new images with similar characteristics. Figure 5.11 shows samples of datasets we use for this work.

The first dataset consist of six images from Basel old town, characterised by a medium number of highly structured facade objects which are regularly arranged. There are just windows and entrances. Due to different decorations the appearance of facade objects differ between the images. Their image scale is 0.5cm/px on average. The second dataset consist of 11 images showing Basel row houses, characterised by a sparse configuration of few facade objects, which are windows and entrances. The configuration of facade objects is homogeneous within the dataset, while the appearance differ due to occlusions and decorations. Their image scale is 0.3cm/px on average. The last dataset consist of 8 images showing apartment houses, characterised by a large number of facade objects, which are windows and balconies. Although the windows are regularly arranged, which seems to be easy to evaluate, the configuration of balconies differ between the images of the collection, which results in less distinctive configuration statistics. Their image scale is 1cm/px on average.

Ground truth annotations are shown in Figures 5.13 to 5.15. Each annotated rectangle contains exactly one object. Please note that balconies are defined as the aggregate of protrusion and windows or doors, respectively, belonging to the balcony. Entrances are aggregates of doors, decoration, and stairs in front of the door.

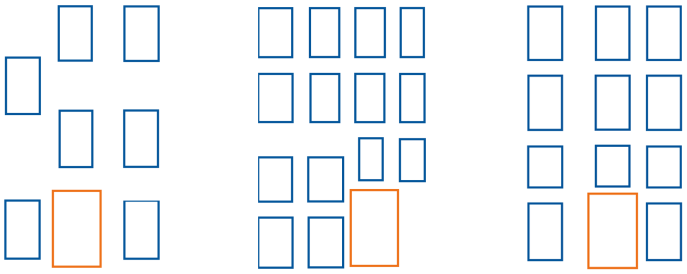
5.5.2 Simulation

Our facade model is a generative model. Therefore, we are able to visually check the quality of the model by simulating facades, based on individually learned configuration statistics, ignoring the dataterm in Equation (5.8). We fix $U_{\text{data}} = -1$ for all proposals and simulate the Markov chain, using a fast geometric temperature schedule with $T_{t+1} = T_0 \cdot \alpha^t$ with fixed parameters $T_0 = 2$ and $\alpha = 0.9999$ for all runs. Figure 5.12 shows samples, based on the configurations statistics learned from each given image collection. Actually, we simulate configurations similar to those of the given images, which reflect the characteristics of underlying training images. Samples of the balcony dataset are more variable due to the diversity of configurations of given training images. Nevertheless, homogeneity of window lattices is realised.

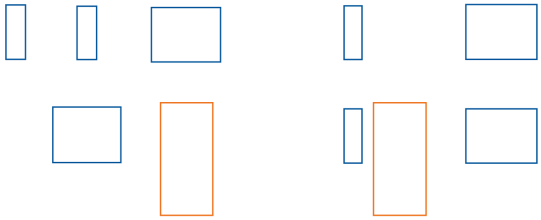
5.5.3 Evaluation

For evaluation on real images, we perform object detection in a leave-one-out cross-validation setting, i.e. we use one image of a dataset for testing and all other for learning the classifier and the configuration statistics. For all experiments we use a slow geometric temperature schedule, using $\alpha = 0.99999$ and $T_0 = 5$. The latter was determined empirically such that the average acceptance rate at beginning was around 70%. We stop the iteration when the temperature reaches $T_{\min} = 10^{-6}$. The weights λ_i, cf. Equation (5.8), were set empirically, too. The weight λ_3, for the prior on the number of objects, is set to 1 for all experiments, while the weights λ_1 and λ_2 were individually set for each experiment, thus manually chosen, but fixed for all images of one data set. The range of marks, thus, range of width and height for sampling new or manipulating existing objects, is given by the training data. The optimisation needed on average 1 to 2 million iterations.

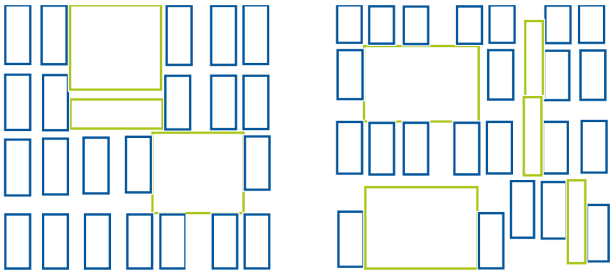
Figure 5.13 to 5.15 visualise exemplary results which we will discuss in Section 5.5.3.2.



(a) Basel



(b) Basel row houses



(c) City houses with balconies

Figure 5.12: Simulations without using bottom-up evidence by an image, just based on learned configuration statistics. **Colours:** Blue: windows. Orange: entrances. Green: balconies.

Figure 5.13: Results for Basel old town dataset, using weights $\lambda_1 = \lambda_2 = 1/7$. Colours as given in Figure 5.12.

Figure 5.14: Results for Basel row houses dataset, using weights $\lambda_1 = \lambda_2 = 1/5$.

Figure 5.15: Results for dataset city houses with balconies, using weights $\lambda_1 = \lambda_2 = 1/5$. Colours as given in Figure 5.12.

Table 5.1: Confusion matrices and confidence regions for object based evaluation. **Left column:** Confusion matrices given by the total number of correct and false detected objects per dataset. **Right column:** Confidence regions (in [%]) with lower bound, mean, and upper bound of the according entry in the confusion matrix, assuming a week Dirichlet prior.

(a) Basel old town

		prediction			confidence region		
		bg	win	entr	background	window	entrance
truth	bg	0	0	0	*22.64* < **98.06** < *100.00*	*0.00* < 0.97 < *59.50*	*0.00* < 0.97 < *59.50*
	win	2	**49**	0	*0.21* < 3.86 < *13.71*	*86.25* < **96.12** < *99.79*	*0.00* < 0.02 < *1.07*
	entr	0	0	**6**	*0.00* < 0.14 < *8.16*	*0.00* < 0.14 < *8.16*	*86.67* < **99.72** < *100.00*

(b) Basel row houses

		prediction			confidence region		
		bg	win	entr	background	window	entrance
truth	bg	0	0	0	*22.64* < **98.06** < *100.00*	*0.00* < 0.97 < *59.50*	*0.00* < 0.97 < *59.50*
	win	0	**34**	0	*0.00* < 0.03 < *1.60*	*97.33* < **99.94** < *100.00*	*0.00* < 0.03 < *1.60*
	entr	0	0	**11**	*0.00* < 0.08 < *4.71*	*0.00* < 0.08 < *4.71*	*92.21* < **99.83** < *100.00*

(c) City houses

		prediction			confidence region		
		bg	win	balc	background	window	balcony
truth	bg	0	**5**	0	*0.11* < 16.75 < *65.33*	*34.46* < **83.08** < *99.89*	*0.00* < 0.17 < *9.56*
	win	6	**146**	0	*1.03* < 3.93 < *9.04*	*90.95* < **96.07** < *98.97*	*0.00* < 0.01 < *0.36*
	balc	0	3	**40**	*0.00* < 0.02 < *1.27*	*0.81* < 6.84 < *19.85*	*80.12* < **93.14** < *99.19*

Table 5.2: Mean and standard deviation of parameter differences of detected objects with respect to ground truth, in [cm] in object space.

$\boldsymbol{x}$	(a) Basel old town	(b) Basel row houses	(c) City houses
$\widehat{\Delta x}$	5.21 ± 14.15	1.97 ± 22.36	3.33 ± 12.53
$\widehat{\Delta y}$	-11.85 ± 18.18	-3.96 ± 10.27	-6.49 ± 11.76
$\widehat{\Delta w}$	-9.37 ± 22.80	4.65 ± 24.86	-7.15 ± 19.57
$\widehat{\Delta h}$	13.35 ± 39.49	10.19 ± 23.28	13.57 ± 23.72

5.5.3.1 Qualitative evaluation

As we aim at an accurate detection of facade objects, we provide a qualitative evaluation in Table 5.1. The left column of Table 5.1 contains confusion matrices which compare predicted and true class for each object and all images of the according dataset, given as total numbers of objects. Thereby, we take into account all detected objects which overlap with ground truth objects by at least 50%. If a detected object does not overlap with any ground truth object, we match it to the ground truths background class, we may say it is a false positive. If a ground truth object does not overlap to any detected object, we have a missing detection and we virtually count it as detected background object.

Taking all correctly detected objects into account, we also provide mean differences to ground truth objects and according standard deviations, for the objects parametres, in Table 5.2.

We also would like to give confidence information according the confusion matrices. But, we have a small sample sizes and did not evaluate the experiments several time; therefore, we can not give mean values and standard deviations for these numbers. Due to the small sample size, it may happen that we observe no object at all for one class. Therefore, we can not normalize each row by the number of samples per row, to get a frequentist estimate of probabilities. However, we seek on a prediction of probabilities and confidence regions for these numbers. In the following we describe a Bayesian

estimate of probabilities and confidence regions for such confusion matrices.

A Bayesian Estimate for Confusion Matrices Probabilities. Let us take one row $\boldsymbol{y}_i = [y_{i,1} \dots y_{i,j} \dots y_{i,C}]$ of the confusion matrix, representing the numbers of ground truth objects of class $i \in \{1 \dots C\}$ predicted to belong to class $j \in \{1 \dots C\}$. To avoid overloading the notation, we leave out i in the following.

The numbers $\boldsymbol{y}$ are Multinomial distributed $\mathcal{M}(\boldsymbol{\tau})$, with parameters $\boldsymbol{\tau} = [\tau_1 \dots \tau_C]$, $\tau_j > 0$, giving the probability of each class, such that $\sum_{j=1}^{C} \tau_j = 1$. In order to get an estimate of probabilities for each entry of the confusion matrix, we set up a Bayes estimate, assuming a week prior on the parameters $\boldsymbol{\tau}$ of the Multinomial distribution. The conjugate prior of the Multinomial is the Dirichlet distribution $\mathcal{D}(\boldsymbol{\alpha})$ which is defined on the $C-1$ simplex, with parameters α_j, $j = 1 \dots C$, which we call concentration parameters and which must not sum to 1. Multiplying likelihood and conjugate prior, the posterior is again Dirichlet

$$\mathcal{D}(\widehat{\boldsymbol{\alpha}} \mid \boldsymbol{y}) = \mathcal{M}(\boldsymbol{y} \mid \boldsymbol{\tau})\mathcal{D}(\boldsymbol{\tau}) \ , \tag{5.27}$$

where the parameters are easily updated by

$$\widehat{\boldsymbol{\alpha}} = \boldsymbol{\tau} + \boldsymbol{y} \ , \tag{5.28}$$

without explicitly estimating the right hand side of (5.27). We assume the prior to have parameters

$$\tau_j = \begin{cases} 1 & \text{if } \ j = i \\ 0.01 & \text{otherwise} \ , \end{cases} \tag{5.29}$$

thus, a large prior on the number of predictions for the true class, and a small prior on the other classes. The mean of the posterior is given by

$$\mu_j = \frac{\widehat{\alpha}_j}{\sum_{c=1}^{C} \widehat{\alpha}_c} \ . \tag{5.30}$$

Estimating the variance of the Dirichlets pdf is not sufficient to provide confidence regions, due to the non-symmetry. On the other hand, estimating confidence regions for a Dirichlet distribution is not trivial. Instead, we use the fact that the the marginal distribution of the Dirichlet is the Beta distribution $\mathcal{B}(a, b)$ with parameters $a = \alpha_j$ and $b = A - \alpha_j$, with $A = \sum_c \alpha_c$. Therefrom, we provide confidence regions for each value y_j given by its marginal Beta distribution $\mathcal{B}(\alpha_j, A - \alpha_j)$, which is a one dimensional distribution for which we easily find the lower and upper bound, y_j^{lb} and y_j^{ub}, such that

$$\int_0^{y_j^{\text{lb}}} \mathcal{B}(y \mid a, b) \ \mathrm{d}y = \int_{y_j^{\text{ub}}}^{1} \mathcal{B}(y \mid a, b) \ \mathrm{d}y = \frac{\beta}{2} \ , \tag{5.31}$$

using $\beta = 1 - S$ with confidence value $S = 0.99$.

Confidence regions which we evaluate using this procedure, are shown in Table 5.1. Classes with 100% true positive detection rate, actually get a tight confidence region near 100%, e.g., bottom right entry in (a) and (b). While ground truth classes with zero predictions, e.g., top left in (a) and (b) get confidence regions which represent the chosen Dirichlet prior.

5.5.3.2 Discussion

We have seen exemplary results of our facade image interpretation procedure in Figures 5.13 to 5.15. Evaluation of the whole datasets in terms of confusion matrices and their confidence regions are given in Table 5.1. Finally, accuracy of detected objects is given in Table 5.2.

For all shown samples from dataset Basel old town and Basel row houses, the detection of facade objects is complete; thus, we do not miss any object, except few small windows, which is stressed by

the confusion matrices, Table 5.1 (a) and (b). Even in case of occlusion, the prior terms support the detection of partially hidden objects, cf. Figure 5.14, top and bottom row. We yield detection rates between 92% and 96% for windows and 100% for entrances.

The dataset city houses with balconies is more challenging, cf. Figure 5.15. The confusion matrix, Table 5.1 (c), shows more false detections and missing objects, respectively. We yield detection rates of 96% and 93% for windows and balconies, respectively. We miss some windows which, due to protrusion of balconies and perspective distortion, overlap with bounding boxes of neighbouring balconies. As our energy is designed to prevent overlapping objects, the process at equilibrium does not accept these object proposals, e.g., Figure 5.15 top row. Further, we miss some windows or do not get their exact outline. Both might be explained by the loose structure of neighbouring objects. For example, the height of windows is wrongly detected if they do not have horizontally neighboured windows, whose height supports them, cf. Figure 5.15 bottom row. The same holds for two missing windows at 3rd row of Figure 5.15. Their appearance differs from common windows of this dataset, thus, their data energy is weak, and their existence is not supported by vertically neighboured windows.

These effects show that the energy should take into account global neighbourhood relations. The pairwise energy term we use, takes into account objects which are directly adjacent. In this dataset, we may argue that the regular window grid is disturbed by balconies in between. Thus, we should model the global layout of each individual class, which is not the case in our model.

In order to make evaluation of accuracy comparable between all images and data sets, we measure differences of outline of detected objects from outline of ground truth objects with respect to the images scale, thus, in cm in object space. Mean values and standard deviations are given in Table 5.2. Averaging over all data sets, we observe a mean difference from outline of ground truth objects of about 2 cm and 11 cm in position, and 5 cm and 13 cm in size. However, standard deviations are larger than the mean values for all parameters. Therefore, we claim to perform object detection with an accuracy of 15 cm in position and 24 cm in size. Related to the ground sampling distances of about 1 pixel per cm and the usage of single view images these are reasonable results.

5.5.3.3 Failure Cases Through Varying Parameters or Local Minima

During our experiments, when varying the parameters for simulated annealing, sometimes, the process convergences to local minima. Figure 5.16 shows some of these results.

The top row demonstrates the power of the prior energy in case of sparse object distribution. In all these failure cases, the windows are almost perfectly aligned or of same height or width, respectively, but they do not meet the true windows. Figure 5.16, bottom, shows that the prior energy sometimes hallucinates an object, e.g., a window instead of wall, two windows instead of an entrance, or a balcony instead of a window. In theses cases, the data term does not hinder the result to be dominated by the learned structure of typical facades.

Figure 5.17 demonstrates the relevance of the prior U_{num}. If we set the weight $\lambda_3 = 0$, thus, ignoring U_{num}, the regular structure of window grids and their strong data term overvotes the existence of balconies or single entrances. We observed in our experiments that the data energy of minor classes, i.e. entrances and balconies, is always inferior to those of windows. Therefore, the missing prior U_{num} leads to more accepted window proposals. On the other hand, the strong structure of window grids compared to the loose structure of window - balcony grids, results in lower energies in the pairwise prior term, too.

Figure 5.16: Local minima. Through varying the parameters of simulated annealing, the optimiser sometimes converges to local minima.

Figure 5.17: Influence of the prior on the number of objects U_{num}: If we set the weight $\lambda_3 = 0$, thus, ignoring the prior on the number of objects per class, the regular structure of window grids overvotes the existence of single entrances or balconies.

5.6 Summary

In this chapter, we proposed a novel method for facade image interpretation based on a marked point process, combining bottom-up evidence, given by an object classifier, and prior knowledge, about typical configurations of facade objects, from top-down. We represented facade objects by a simplified rectangular object model and presented an energy model which evaluates the agreement of a proposed configuration with the given image and the learned statistics about typical configurations. Due to the learned prior energies, our model is almost free of tunable parameters, in contrast to other approaches, dealing with marked point processes. We showed promising results and provided a qualitative evaluation which demonstrated the capability of complete and accurate detection of facade objects.

However, we are not competitive to state-of-the-art results, e.g., Teboul et al. (2010) in terms of complexity. In contrast to them we deal with few classes per image. We are aware on the weakness of our evaluation: We proved our approach on few datasets with small sample size. Their appearance, as well as structure of facade objects, are homogeneous within each dataset. Further, the evaluation by leave-one-out-crossvalidation might by too optimistic, due to the homogeneity of the datasets. However, as soon as more training data are available and the procedures are parallelised for GPU processing, we may evaluate more data and may learn the remaining weights from data, e.g., by cross validation.

Nevertheless, compared to grammar based approaches, we are more flexible in terms of underlying structure of facade objects and even able to deal with very sparse structure. We believe that we prospectively may overcome the limitations of grammar based approaches, which have to be designed individually for each type of facade and are dedicated for large, regularly structured types of facades, such as Hausmanian, cf. Teboul et al. (2010).

CHAPTER 6

Conclusion

6.1 Summary

The goal of this theses was to develop and evaluate a scheme for interpreting facade images by combining evidence from bottom-up and prior knowledge from top-down. We aimed at the accurate detection and description of relevant facade objects from single rectified facade images.

Our approach is dedicated to building facades, characterised by architectural principles as equally sized or alignment of neighbouring facade objects, symmetry within and between facade objects, and topological rules, such as entrances are supposed to be near the ground level. Further, facade objects are supposed to be characterised by straight, sometimes curved lines, in a way that they form regular, symmetric objects. Therefore, and due to variability in appearance of facades, e.g., by lighting conditions, color or decorations, we chose line and edge features as starting point for the interpretation from bottom-up.

In a three-step procedure we extracted straight lines, circular and elliptical arc segments from images. From these we generated pairs of adjacent line segments and learned shapelets as representative pairs of adjacent line segments, in order to use them in a bag of words approach for object detection. Finally, at high-level, we proposed the image interpretation using a Markov marked point process. Therefor, we used the results of object detection and combined them with prior knowledge about typical configurations of facade objects, which we learned from training data.

For low-level image processing we presented a line simplification approach which approximates given pixel-chains by a sequence of lines, circular and elliptical arcs. This, again, was designed as a two-step procedure. For pre-segmentation we proposed an adaption to Douglas-Peucker's algorithm for the use of circles instead of straight lines. For further simplification we merged neighbouring segments due to their agreement to a joint geometric model in terms of description length.

In our experiments, we showed that the new pre-segmentation correctly identifies arc segments and especially their breakpoints. We improved the standard algorithm in terms of reducing the number of breakpoints of a given polygon, while preserving the geometry. We further showed that the merging step correctly identified elliptical arcs and further reduces the number of segments of most given pixel-chains. We tested the algorithm on several images with comparable good or even better results, referring to a state-of-the-art algorithm.

At mid-level we used the results of detecting line, circle, and ellipse segments and generated pairs

of adjacent line segments as image features. We proposed to learn which of them are most dominant for the description of facade objects and called their representatives shapelets. We showed how to extract pairs of adjacent line segments from images, how to describe them, and proposed a distance between pairs of adjacent line segments. Using the descriptor and the proposed distance, we cluster pairs of adjacent line segments to derive the shapelets.

Further, we proposed to apply a bag of words approach for the classification of facade objects, using shapelets as words of the bag of words codebook. Therefor, we proposed to use the histogram of shapelets as feature vector for classification, whereby, we incorporated a rough spatial layout by splitting the image patch into several tiles, from which the overall feature vector is given by the concatenated histogram of shapelets of all tiles.

In our experiments we examined the performance of the proposed features and classification procedure on our target object classes, thus, several types of facade objects. We reached a reasonable classification performance on a challenging dataset including intra-class variations, clutter, and scale changes. We yield an average class accuracy of 88%. Thereby, arc-type windows performed best, they were correctly classified up to 93% accuracy, which proved the utility of elliptical and circular arc segments. In contrast, rectangular windows, due to their large variability, were more challenging and were confused with other facade elements. They were correctly classified with 81% accuracy. For balconies we yielded a classification performance of up to 88% true positive rate. Which we explained by characteristic shapelets which code the projected 3D structure, mapped to the image plane and characterised by the perspective projection of 3D objects. The classification of entrances yielded an accuracy of 87%, they were mostly confused with balconies, which we predicted to avoid by incorporating prior knowledge from top-down.

Finally, combining bottom-up evidence given by the object classifier that uses learned shapelets, and prior knowledge about typical configurations of facade objects, from top-down, we proposed the facade image interpretation based on a Markov marked point process. Therefor, we used an energy model which evaluates the agreement of a proposed configuration with the given image and the learned statistics about typical configurations. We proposed to learn prior knowledge about typical configurations of facade objects from training data. For this, we collected statistics about objects geometry, i.e. position, width, and height, and about pairwise neighbourhood relations as alignment, size difference, distance, and intersection. Therefore, in contrast to existing approaches for facade image interpretation, we avoid predefining and fix sets of rules. Due to the learned prior energies, the proposed model is almost free of tunable parameters.

In our experiments we demonstrated the capability of complete and accurate detection of facade objects. We yielded object detection rates up to 100% for windows and entrances. For more difficult data sets we still yielded detection rates of 96% for window and 93% for balconies. For the estimated parameters we provided an accuracy of 15 cm in position and 25 cm in size, regarding the object space, which is reasonable related to the ground sampling distances of about 1 pixel per cm and the usage of single view images.

6.2 Discussion

Our final results showed that we are able to perform facade image interpretation providing a complete and accurate detection of target objects. We successfully proved the concept of combining bottom-up evidence and top-down prior knowledge using a marked point process for the task of image interpretation, where almost all parts of the process were learned from training data, representation as well as prior information.

However, the approach should be evaluated on more and larger datasets, showing more classes, more variable facade structure and less homogeneous appearance. We are not competitive to state-of-the-art results, e.g., Teboul et al. (2010) in terms of complexity. Nevertheless, compared to grammar based approaches, we are more flexible in terms of underlying structure of facade objects and even able to

deal with very sparse structure. We believe that we prospectively may overcome the limitations of grammar based approaches.

Our concept of learning shapelets is closely related to the concept of deep learning which is often synonymously used with the application of deep convolutional neural networks. But, this is just one very popular and powerful algorithm, yielding impressive results and decreasing the error rates on well known benchmark datasets (Russakovsky et al., 2015; Ren et al., 2015). Deep learning rather is a collective term, describing machine learning algorithms that are based on learning multiple levels of representation or abstraction (Bengio, 2009; LeCun et al., 2015). For example, this could mean to learn edges in terms of oriented contrast at the very first stage. In a next level, we could learn combinations of these edge fragments which probably form certain shapes. And so on, we could further learn even more complex features from combinations of the level before up to an arbitrary complexity. This is done automatically in what we call deep belief propagation networks. In that sense, we are related to deep learning due to learning more complex features, our shapelets, from low-level features, our line segments. But, current deep learning algorithms, such as convolutional neural networks, include feedback steps, in terms of backpropagation, where the results of the layers are backtracked to the given training data in order to learn the weights of the layers nodes. Further, they often use the concept of distributed representation, which means that similar classes share parts of the description (Bengio, 2009).

In our context, for generating bottom-up evidence, we perform the representation learning without a feed back loop or sharing parts of the description. A full deep learning approach would go one step further and in a feed back loop would simultaneously learn the low level structures.

6.3 Outlook

In our experiments we showed promising results, but we already pointed out some limitations. In this section, we will collect several suggestions or questions for future work.

Enhancement of the Energy Model. Rather minor is the learning of the remaining weights from data. Usually, such a task is solved by grid search within a cross-validation. This can be done as soon as more training data are available and the procedures are parallelised for GPU processing.

We have shown results on simple facade structures showing few object classes. Future work should focus on the enhancement of the energy model, in a way that it expresses the structure of more complex facades, including more classes. Thereby, we may include more classes in the current process easily, as long as annotated data are provided. But, we already identified the limitations of our pairwise prior energy due to missing global constraints. Thus, a question for future work is to explore, how we include global neighbourhood relations into the energy of a marked point process. Chai et al. (2012) propose to combine Markov random field and marked point process, whereby, the Markov random field is used to represent the statistics of the image appearance to provide low-level evidence. However, they show how to communicate between Markov random fields and marked point processes. So far, Markov random fields, marked point processes, and grammars are competitive models for high-level reasoning. Future work should exploit the weakness and strength of these approaches and should combine them to even more powerful methods.

Sampling of the Markov Chain. The optimization of the proposed Gibbs energy, using reversible jump Markov chain Monte Carlo is still slow, which makes the approach not applicable for large scale applications. Thus, a task that should be addressed in future work, is the speed up of optimization. Verdie and Lafarge (2013) propose sampling in parallel, where a large number of perturbations is proposed simultaneously, using a unique chain. Therefor, they exploit the Markov property by splitting the global sampling problem into spatially independent proposals in a local neighbourhood. The common Markov chain Monte Carlo sampling procedure performs all steps sequentially, which inherently prohibits parallelisation. Through parallel sampling, all calculations according to each individual

proposal can be done in parallel, which allows for parallisation as well as implementing the procedure on GPU.

Further, the data-driven sampling mechanism could be enhanced in a way that sampling should be guided by the learned structure of facade objects, too. Thus, structure-driven sampling, as proposed by Wang et al. (2015).

Another topic for future research is the sampling of several Markov chains in parallel to smooth the jointly sampled target distribution. Searching for suited strategies, for the exchange of samples between the chains, should enhance the mixing properties of the Markov chain (Mingas and Bouganis, 2012).

Deep Learning. Convergence of the Markov Chain heavily depends on the data term, thus, evidence given by the object classifier. Our classification scheme is not yet general enough, thus will not reach state-of-the-art art performance in more general schemes. Provided a huge set of annotated training data, methods from deep learning will provide more powerful and even more flexible classifiers to support the proposed energy.

Altogether, the proposed facade image interpretation system has shown promising results. The application of marked point processes to the task of facade image interpretation has shown its potential as flexible model for the modelling of configurations of facade objects.

Several aspects like adapting the classification to be more flexible to a wider range of object classes, as well as improving the energy model and its optimization procedure, due to computational time, reveal potential for future research.

APPENDIX A

Matrix Calculus

This section summarises results from Matrix calculus we use throughout this theses.

We use the following derivatives of squared forms of vectors $\boldsymbol{x}$ and $\boldsymbol{a}$ and a square matrix $\boldsymbol{A}$ with respect to $\boldsymbol{x}$.

$$\frac{\partial \boldsymbol{x}^\mathsf{T}\boldsymbol{a}}{\partial \boldsymbol{x}} = \frac{\partial \boldsymbol{a}^\mathsf{T}\boldsymbol{x}}{\partial \boldsymbol{x}} = \boldsymbol{a} \tag{A.1}$$

$$\frac{\partial \boldsymbol{x}^\mathsf{T} A \boldsymbol{x}}{\partial \boldsymbol{x}} = (A + A^\mathsf{T})\boldsymbol{x} \tag{A.2}$$

$$\frac{\partial |\boldsymbol{x}|^2}{\partial \boldsymbol{x}} = \frac{\partial \boldsymbol{x}^\mathsf{T}\boldsymbol{x}}{\partial \boldsymbol{x}} = 2\boldsymbol{x} \tag{A.3}$$

The derivative of a matrix $\boldsymbol{A} \in \mathbb{R}^I \times \mathbb{R}^J$ with respect to another matrix $\boldsymbol{B} \in \mathbb{R}^K \times \mathbb{R}^L$ is defined as the element wise derivative of each element in $\boldsymbol{A}$ with respect to each element in $\boldsymbol{B}$

$$\frac{\mathrm{d}\boldsymbol{A}}{\mathrm{d}\boldsymbol{B}} = \frac{\mathrm{d}a_{ij}}{\mathrm{d}b_{kl}}\,. \tag{A.4}$$

This can be expressed differently using the vec operator

$$\frac{\mathrm{d}\boldsymbol{A}}{\mathrm{d}\boldsymbol{B}} = \frac{\mathrm{d}\mathrm{vec}\boldsymbol{A}}{\mathrm{d}\mathrm{vec}\boldsymbol{B}}\,. \tag{A.5}$$

Whereby, the vec$(\cdot)$ operator produces a vector $\boldsymbol{a}$ from matrix $\boldsymbol{A}$ by column-wise taking its elements

$$\boldsymbol{a} = \begin{bmatrix} a_{11} \\ \vdots \\ a_{I1} \\ a_{21} \\ \vdots \\ a_{IJ} \end{bmatrix} = \mathrm{vec} \begin{bmatrix} a_{11} & \dots & a_{1J} \\ \vdots & \dots & \vdots \\ a_{I1} & \dots & a_{IJ} \end{bmatrix}\,. \tag{A.6}$$

When dealing with symmetric matrices, we use the vech$(\cdot)$ operator, which produces a vector $\boldsymbol{c}$ from a

square matrix $\boldsymbol{C} \in \mathbb{R}^N \times \mathbb{R}^N$, by column-wise taking its lower triangle elements

$$\boldsymbol{c} = \begin{bmatrix} c_{11} \\ \vdots \\ c_{N1} \\ c_{22} \\ \vdots \\ c_{N2} \\ c_{33} \\ \vdots \\ c_{NN} \end{bmatrix} = \text{vech} \begin{bmatrix} c_{11} & \dots & 0 & 0 \\ c_{21} & c_{22} & \dots & 0 \\ \vdots & & \dots & \vdots \\ c_{N1} & c_{N2} & \dots & c_{NN} \end{bmatrix} = \boldsymbol{S}_N \text{vec} \boldsymbol{C} \ , \tag{A.7}$$

where $\boldsymbol{S}_N$ is a matrix containing only 0 and one 1 per row, such that it realises (A.7), e.g.,

$$\boldsymbol{S}_2 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{A.8}$$

or

$$\boldsymbol{S}_3 = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} . \tag{A.9}$$

From Equation (A.7) we obtain for $\boldsymbol{C} \in \mathbb{R}^N \times \mathbb{R}^N$

$$\frac{\text{dvech} \boldsymbol{C}}{\text{dvec} \boldsymbol{C}} = \boldsymbol{S}_N \tag{A.10}$$

and from inversion we obtain

$$\frac{\text{dvec} \boldsymbol{C}}{\text{dvech} \boldsymbol{C}} = \boldsymbol{S}_N^\mathsf{T} . \tag{A.11}$$

Having this relation in mind and using the identity

$$\text{vec}(\boldsymbol{AXB}) = \left(\boldsymbol{B}^\mathsf{T} \otimes \boldsymbol{A}\right) \text{vec} \boldsymbol{X} \ , \tag{A.12}$$

we obtain the following useful derivatives, where again, $\boldsymbol{A} \in \mathbb{R}^I \times \mathbb{R}^J$, $\boldsymbol{B} \in \mathbb{R}^K \times \mathbb{R}^L$ and $\boldsymbol{C}, \boldsymbol{D} \in \mathbb{R}^N \times \mathbb{R}^N$

$$\frac{\text{d} \boldsymbol{AXB}}{\text{d} \boldsymbol{X}} = \frac{\text{dvec} \boldsymbol{AXB}}{\text{dvec} \boldsymbol{X}} = \frac{\text{d}(\boldsymbol{B}^\mathsf{T} \otimes \boldsymbol{A}) \text{vec} \boldsymbol{X}}{\text{dvec} \boldsymbol{X}} = \boldsymbol{B}^\mathsf{T} \otimes \boldsymbol{A} \tag{A.13}$$

$$\frac{\text{dvech} \boldsymbol{C}}{\text{dvech} \boldsymbol{D}} = \frac{\text{dvech} \boldsymbol{C}}{\text{dvec} \boldsymbol{C}} \frac{\text{dvec} \boldsymbol{C}}{\text{dvec} \boldsymbol{D}} \frac{\text{dvec} \boldsymbol{D}}{\text{dvech} \boldsymbol{D}} = \boldsymbol{S}_N \frac{\text{dvec} \boldsymbol{C}}{\text{dvec} \boldsymbol{D}} \boldsymbol{S}_N^\mathsf{T} = \boldsymbol{S}_N \frac{\text{d} \boldsymbol{C}}{\text{d} \boldsymbol{D}} \boldsymbol{S}_N^\mathsf{T} . \tag{A.14}$$

APPENDIX B

Uncertain Line Primitives

In this section, we summarise the homogeneous representations of uncertain geometric elements we need in Ch. 3. A detailed discussion of uncertain homogeneous points and lines can be found in Förstner and Wrobel (2016) and Meidow et al. (2009).

B.1 Points and Straight Lines

B.1.1 Homogeneous Representations of Uncertain Points

We represent an uncertain euclidean 2D point $\underline{\boldsymbol{x}} = [\underline{x}, \underline{y}]^\mathsf{T}$ by its mean and covariance

$$\underline{\boldsymbol{x}} \sim \mathcal{N}(\boldsymbol{\mu}_x, \Sigma_{xx}) = \mathcal{N}\left(\begin{bmatrix} \mu_x \\ \mu_y \end{bmatrix}, \begin{bmatrix} \sigma_x^2 & \sigma_{xy} \\ \sigma_{xy} & \sigma_y^2 \end{bmatrix} \right) . \tag{B.1}$$

We obtain a homogeneous representation by adding a non-stochastic homogeneous coordinate

$$\underline{\mathbf{x}} \sim \mathcal{N}(\boldsymbol{\mu}_{\mathsf{x}^\mathsf{e}}, \Sigma_{\mathsf{x}^\mathsf{e}\mathsf{x}^\mathsf{e}}) = \mathcal{N}\left(\begin{bmatrix} \mu_x \\ \mu_y \\ 1 \end{bmatrix}, \begin{bmatrix} \sigma_x^2 & \sigma_{xy} & 0 \\ \sigma_{xy} & \sigma_y^2 & 0 \\ 0 & 0 & 1 \end{bmatrix} \right) . \tag{B.2}$$

This is called the euclidean normalised point representation. When using this normalisation, we denote the first two elements the euclidean part and the third element the homogeneous part

$$\mathbf{x} = \begin{bmatrix} \boldsymbol{x}_0 \\ x_h \end{bmatrix} . \tag{B.3}$$

Sometimes we wish to normalise the coordinate vector, such that its length is 1, this is called spherical normalisation

$$\underline{\mathbf{x}}^s \sim \mathcal{N}(\boldsymbol{\mu}_{\mathsf{x}^s}, \Sigma_{\mathsf{x}^s\mathsf{x}^s}) = \mathcal{N}\left(\frac{\boldsymbol{\mu}_\mathsf{x}}{|\boldsymbol{\mu}_\mathsf{x}|}, J_s(\boldsymbol{\mu}_\mathsf{x}) \Sigma_{\mathsf{x}\mathsf{x}} J_s(\boldsymbol{\mu}_\mathsf{x})^\mathsf{T} \right) , \tag{B.4}$$

where we obtain the covariance matrix through variance propagation using the Jacobian

$$\underset{3\times 3}{J_s(\mathbf{x})} = \frac{\partial \mathbf{x}^s}{\partial \mathbf{x}} = \frac{1}{|\mathbf{x}|} \left(I_3 - \frac{\mathbf{x}\mathbf{x}^\mathsf{T}}{\mathbf{x}^\mathsf{T}\mathbf{x}} \right) . \tag{B.5}$$

The different representations of a point $\mathcal{X}$ are visualised in Figure B.1.

At the end of any estimation or reasoning procedure, we usually want to know the euclidean coordinates. For this we need the transformation back to the euclidean representation. Given an uncertain homogeneous point $\underline{\mathbf{x}}$, with mean $\boldsymbol{\mu}_{\mathsf{x}}$ and covariance $\boldsymbol{\Sigma}_{\mathsf{xx}}$, we obtain the euclidean normalised point by

$$\boldsymbol{\mu}_x = \frac{\boldsymbol{\mu}_{\mathsf{x}0}}{\mu_{\mathsf{x}\mathsf{h}}} = \begin{bmatrix} \mu_x \\ \mu_y \end{bmatrix} \tag{B.6}$$

and its covariance by

$$\underset{2\times 2}{\boldsymbol{\Sigma}_{xx}} = \underset{2\times 3}{J_{x\mathsf{x}}(\boldsymbol{\mu}_{\mathsf{x}})^{\mathsf{T}}} \underset{3\times 3}{\boldsymbol{\Sigma}_{\mathsf{xx}}} \underset{3\times 2}{J_{x\mathsf{x}}(\boldsymbol{\mu}_{\mathsf{x}})} = \begin{bmatrix} \sigma_x^2 & \sigma_{xy} \\ \sigma_{xy} & \sigma_y^2 \end{bmatrix} , \tag{B.7}$$

using the Jacobian

$$\underset{2\times 3}{J_{x\mathsf{x}}(\mathbf{x})^{\mathsf{T}}} = \frac{\partial \boldsymbol{x}}{\partial \mathbf{x}} = \frac{1}{x_h^2} \left[x_h \boldsymbol{I}_2 | - \boldsymbol{x}_0 \right] . \tag{B.8}$$

Both, $\boldsymbol{\Sigma}_{\mathsf{x}^e\mathsf{x}^e}$ and $\boldsymbol{\Sigma}_{\mathsf{x}^s\mathsf{x}^s}$, have rank two, which will cause problems as soon as we need an inverse covariance matrix. We will investigate this problem later within this section. We point out that their ellipsoids are flat.

The standard ellipsoid of $\boldsymbol{\Sigma}_{\mathsf{x}^e\mathsf{x}^e}$ lies in the euclidean plane, which is shown by its null space

$$\text{null}\left(\boldsymbol{\Sigma}_{\mathsf{x}^e\mathsf{x}^e}\right) = \boldsymbol{e}_3 = \begin{bmatrix} \mathbf{0} \\ |x_h| \end{bmatrix} . \tag{B.9}$$

The standard ellipsoid of $\boldsymbol{\Sigma}_{\mathsf{x}^s\mathsf{x}^s}$ lies in the tangent plane of the unit sphere, which is shown by its null space

$$\text{null}\left(\boldsymbol{\Sigma}_{\mathsf{x}^s\mathsf{x}^s}\right) = \mathbf{x}^s . \tag{B.10}$$

B.1.2 Homogeneous Representations of Uncertain Lines

To represent euclidean lines, we choose its Hessian parametres $\underline{\boldsymbol{l}} = \left[\underline{\phi}, \underline{d}\right]^{\mathsf{T}}$. The uncertain euclidean line is then represented by its mean and covariance

$$\underline{\boldsymbol{l}} \sim \mathcal{N}\left(\boldsymbol{\mu}_l, \boldsymbol{\Sigma}_{ll}\right) = \mathcal{N}\left(\begin{bmatrix} \mu_\phi \\ \mu_d \end{bmatrix}, \begin{bmatrix} \sigma_\phi^2 & \sigma_{\phi d} \\ \sigma_{\phi d} & \sigma_d^2 \end{bmatrix} \right) . \tag{B.11}$$

The euclidean normalised homogeneous line is given by

$$\mathbf{l}^e = \begin{bmatrix} \cos\phi \\ \sin\phi \\ -d \end{bmatrix} = \begin{bmatrix} \boldsymbol{l}_h \\ l_0 \end{bmatrix} \tag{B.12}$$

whereby, we denote the first two elements the homogeneous part and the third element the euclidean part.

The transformation $\boldsymbol{l} \to \mathbf{l}^e$ is not linear; thus, the covariance matrix can be derived approximately by variance propagation (Förstner and Wrobel, 2016). We obtain

$$\underline{\mathbf{l}}^e \sim \mathcal{N}\left(\boldsymbol{\mu}_{\mathsf{l}^e}, \boldsymbol{\Sigma}_{\mathsf{l}^e\mathsf{l}^e}\right) = \mathcal{N}\left(\begin{bmatrix} \cos\mu_\phi \\ \sin\mu_\phi \\ -\mu_d \end{bmatrix}, J_{\mathsf{l}^e}(\boldsymbol{\mu}_l) \boldsymbol{\Sigma}_{ll} J_{\mathsf{l}^e}(\boldsymbol{\mu}_l)^{\mathsf{T}} \right) \tag{B.13}$$

using the Jacobian

$$J_{\mathsf{l}^e}(\boldsymbol{l}) = \frac{\partial \mathbf{l}^e}{\partial \boldsymbol{l}} = \begin{bmatrix} -\sin\phi & 0 \\ \cos\phi & 0 \\ 0 & -1 \end{bmatrix} . \tag{B.14}$$

The spherical normalisation is given by

$$\underline{\mathbf{l}}^s \sim \mathcal{N}\left(\boldsymbol{\mu}_{\mathsf{l}^s}, \boldsymbol{\Sigma}_{\mathsf{l}^s\mathsf{l}^s}\right) = \mathcal{N}\left(\frac{\boldsymbol{\mu}_{\mathsf{l}}}{|\boldsymbol{\mu}_{\mathsf{l}}|}, J_s(\boldsymbol{\mu}_{\mathsf{x}}) \boldsymbol{\Sigma}_{\mathsf{xx}} J_s(\boldsymbol{\mu}_{\mathsf{x}})^{\mathsf{T}} \right) , \tag{B.15}$$

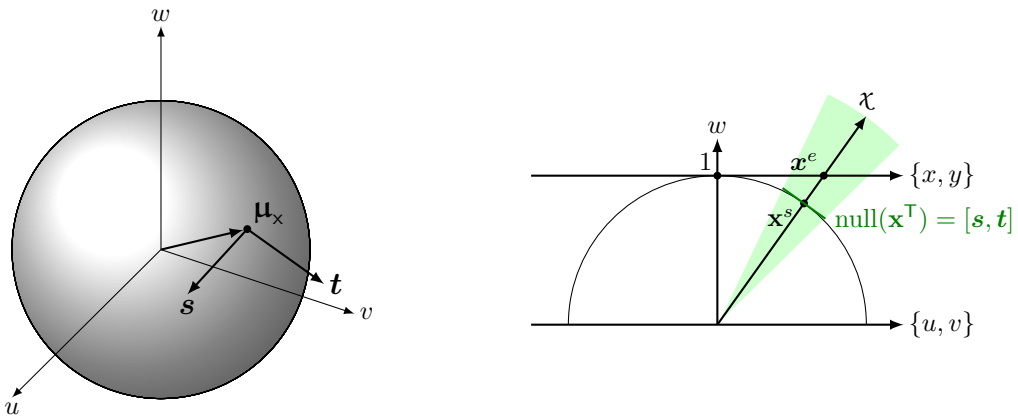


Figure B.1: Homogeneous coordinates of a point $\mathcal{X}$. **Left:** The tangent space $[\boldsymbol{s}, \boldsymbol{t}]$ at $\boldsymbol{\mu}_x$. **Right:** Cross section across the unit sphere and the representations of a point $\mathcal{X}$.

using the Jacobian given in (B.5). Again, we see that both $\Sigma_{\text{l}^\text{e}\text{l}^\text{e}}$ and $\Sigma_{\text{l}^\text{s}\text{l}^\text{s}}$ have rank two. Its nullspaces are

$$\text{null}\left(\Sigma_{\text{l}^\text{e}\text{l}^\text{e}}\right) = \begin{bmatrix} \cos\mu_\phi \\ \sin\mu_\phi \\ 0 \end{bmatrix} = \begin{bmatrix} \boldsymbol{\mu}_{l_h} \\ 0 \end{bmatrix} \tag{B.16}$$

$$\text{null}\left(\Sigma_{\text{l}^\text{s}\text{l}^\text{s}}\right) = \boldsymbol{\mu}^s \,. \tag{B.17}$$

When we are interested in the uncertain Hessian parametres, we need the transformation back to the euclidean representation. Given an uncertain homogeneous line $\underline{\mathbf{l}}$, with mean $\boldsymbol{\mu}_\text{l}$ and covariance Σ_{ll}, we obtain the euclidean parametres by

$$\boldsymbol{\mu}_l = \begin{bmatrix} \text{atan2}(b, a) \\ -\frac{c}{\sqrt{a^2+b^2}} \end{bmatrix} = \begin{bmatrix} \mu_\phi \\ \mu_d \end{bmatrix} \tag{B.18}$$

and its covariance by

$$\Sigma_{ll} = J_{ll}(\boldsymbol{\mu}_\text{l})\Sigma_{\text{ll}}J_{ll}(\boldsymbol{\mu}_\text{l})^\mathsf{T} = \begin{bmatrix} \sigma_\phi^2 & \sigma_{\phi d} \\ \sigma_{\phi d} & \sigma_d^2 \end{bmatrix} \,, \tag{B.19}$$

using the Jacobian evaluated at $\boldsymbol{\mu}_\text{l}$ of $\mathbf{l} = [a, b, c]^\mathsf{T}$

$$J_{ll}(\mathbf{l}) = \frac{\partial \boldsymbol{l}}{\partial \mathbf{l}} = \frac{1}{s^3}\begin{bmatrix} -bs & as & 0 \\ ac & bc & -s^3 \end{bmatrix} \qquad s = \sqrt{a^2+b^2}\,. \tag{B.20}$$

B.1.3 Reduced Homogeneous Coordinates

Representations of uncertain homogeneous elements, we have seen so far, suffer on a rank deficiency. This will cause problems as soon as we need to inverse the covariance matrix, e.g., when estimating a Mahalonobis distance or solving the normal equation during an adjustment procedure. To solve this, we use reduced coordinates as given in Förstner and Wrobel (2016).

The main idea is to use spherical normalised coordinates and to represent the uncertain homogeneous entity in the *tangent space* at $\boldsymbol{\mu}_{x^s}$, which is the null space of $\boldsymbol{\mu}_{x^s}^\mathsf{T}$

$$\underset{3\times 2}{[\boldsymbol{s}, \boldsymbol{t}]} = \text{null}(\boldsymbol{\mu}_{x^s}^\mathsf{T})\,, \tag{B.21}$$

which is visualised in Figure B.1. As we mentioned before, all standard ellipses, we have seen so far, are flat such that they are located in the tangent space of their homogeneous coordinates, spanned by the basis vectors $\boldsymbol{s}$ and $\boldsymbol{t}$. Any random difference from the mean is then given as a 2-dimensional random vector $\underline{\boldsymbol{x}}_r$ within the tangent space. Its mean is $\mathbf{0}$ and its covariance is denoted $\Sigma_{x_r x_r}$.

Assume, we want to make a small change $\Delta\boldsymbol{x}_r$ in the coordinates of $\underline{\mathbf{x}}^s$ or its mean value, respectively. The updated coordinates are then

$$\underline{\mathbf{x}}^s(\Delta\boldsymbol{x}_r) = \mathsf{N}\left(\boldsymbol{\mu}_{\mathsf{x}^s} + [\mathsf{s},\mathsf{t}]\Delta\mathsf{x}_\mathsf{r}\right) , \tag{B.22}$$

whereby, $\mathsf{N}(\cdot)$ is the operator for the spherical normalisation, which includes the according variance propagation.

Assume, we are given a tangent space $[\boldsymbol{s},\boldsymbol{t}]$ at $\boldsymbol{\mu}_\mathsf{x}$ and a point $\underline{\mathbf{x}}$ near $\boldsymbol{\mu}_\mathsf{x}$. Using the 3×2 Jacobian of the spherical normalised vector $\underline{\mathbf{x}}^s$, with respect to the reduced coordinates $\underline{\boldsymbol{x}}_r$,

$$\underset{3\times 2}{J_r(\mathbf{x})} = \frac{\partial \mathbf{x}^s}{\partial \boldsymbol{x}_r} = \mathrm{null}(\mathbf{x}^\mathsf{T}) , \tag{B.23}$$

we can estimate the reduced coordinates

$$\underset{2\times 1}{\boldsymbol{x}_r} \approx \underset{2\times 3}{J_r(\boldsymbol{\mu}_\mathsf{x})^\mathsf{T}}\,\underset{3\times 1}{\mathbf{x}^s} , \tag{B.24}$$

which is an approximation due to the Taylor expansion. Further, we obtain the covariance of the reduced coordinates by

$$\underset{2\times 2}{\Sigma_{x_r x_r}} = \underset{2\times 3}{J_r(\boldsymbol{\mu}_\mathsf{x})^\mathsf{T}}\,\underset{3\times 3}{\Sigma_{\mathsf{x}^s\mathsf{x}^s}}\,\underset{3\times 2}{J_r(\boldsymbol{\mu}_\mathsf{x})} . \tag{B.25}$$

Vice versa, we obtain the covariance of the spherical normalised coordinates by

$$\underset{3\times 3}{\Sigma_{\mathsf{x}^s\mathsf{x}^s}} = \underset{3\times 2}{J_r(\boldsymbol{\mu}_\mathsf{x})}\,\underset{2\times 2}{\Sigma_{x_r x_r}}\,\underset{2\times 3}{J_r(\boldsymbol{\mu}_\mathsf{x})^\mathsf{T}} . \tag{B.26}$$

For lines we have an equivalent reduced representation. A small change $\Delta\boldsymbol{l}_r$ in the coordinates of $\underline{\mathbf{l}}^s$ is given by

$$\underline{\mathbf{l}}^s(\Delta\boldsymbol{l}_r) = \mathsf{N}\left(\boldsymbol{\mu}_{\mathsf{l}^s} + J_r(\boldsymbol{\mu}_{\mathsf{l}^s})\Delta\mathsf{l}_\mathsf{r}\right) . \tag{B.27}$$

The covariance of the reduced coordinates is given by

$$\underset{2\times 2}{\Sigma_{l_r l_r}} = \underset{2\times 3}{J_r(\boldsymbol{\mu}_\mathsf{l})^\mathsf{T}}\,\underset{3\times 3}{\Sigma_{\mathsf{l}^s\mathsf{l}^s}}\,\underset{3\times 2}{J_r(\boldsymbol{\mu}_\mathsf{l})} . \tag{B.28}$$

Vice versa, we obtain the covariance of the spherical normalised coordinates by

$$\underset{3\times 3}{\Sigma_{\mathsf{l}^s\mathsf{l}^s}} = \underset{3\times 2}{J_r(\boldsymbol{\mu}_\mathsf{l})}\,\underset{2\times 2}{\Sigma_{l_r l_r}}\,\underset{2\times 3}{J_r(\boldsymbol{\mu}_\mathsf{l})^\mathsf{T}} . \tag{B.29}$$

If not stated otherwise, we will represent homogeneous points and lines by its spherical normalised coordinates and the reduced covariance; thus, a point by $\mathcal{X} : \{\mathbf{x}^s, \Sigma_{x_r x_r}\}$ and a line by $\mathcal{L} : \{\mathbf{l}^s, \Sigma_{l_r l_r}\}$ and sometimes skip the indices.

B.1.4 Statistically Best Fitting Line

Given N observed points $\mathcal{X} = \{\boldsymbol{x}_n\}, n = 1 \dots N$, we aim at the best fitting line. The simplest way is to solve the linear system $\mathcal{X}^\mathsf{T}\mathbf{l} = 0$, which leads to the algebraic best fitting line. Instead, we look for the statistically best fitting line as well as its covariance. We will need this in Section 3.3, when merging neighbouring lines based on their statistical properties. The solution can be found in (Förstner and Wrobel, 2016, Section 9.4.2).

We assume i.i.d. coordinates of each point, sharing the same isotropic covariance $\mathcal{X}_n \sim \mathcal{N}\left(\boldsymbol{x}_n, \sigma_n^2 \boldsymbol{I}_2\right)$.

In (Förstner and Wrobel, 2016, Section 9.4.2) it is shown that the statistically best fitting line passes trough the centroid of given points and that its direction is given by the principal axis of their moment matrix. Thus, we first estimate the weighted centroid of given points by

$$\boldsymbol{x}_0 = \frac{\sum_{n=1}^N w_n \boldsymbol{x}_n}{\sum_{n=1}^N w_n}, \qquad w_n = \frac{1}{\sigma_n^2} \tag{B.30}$$

and derive its direction from the moment matrix

$$M = \sum_{n=1}^{N} w_n \left(\boldsymbol{x}_n - \boldsymbol{x}_0\right) \left(\boldsymbol{x}_n - \boldsymbol{x}_0\right)^{\mathsf{T}} . \tag{B.31}$$

Eigenvalue decomposition of M yields the two principal axis $\boldsymbol{u}_1$ and $\boldsymbol{u}_2$ and their according eigenvalues λ_1 and λ_2, assuming $\lambda_1 \geq \lambda_2$. The direction of the line is given by $\boldsymbol{u}_1$, the eigenvector belonging to the larger eigenvalue λ_1. The eigenvector $\boldsymbol{u}_2$, belonging to the smaller eigenvalue λ_2, represents the normal vector of the sought line; thus, we obtain

$$\widehat{\boldsymbol{l}} = \begin{bmatrix} \boldsymbol{u}_2 \\ -\boldsymbol{u}_2 \boldsymbol{x}_0^{\mathsf{T}} \end{bmatrix} \tag{B.32}$$

as euclidean normalised line vector.

We express the theoretical uncertainty of the line in terms of the uncertainty σ_q^2 across the line and the uncertainty σ_α^2 in the direction of the line, both viewed from its centre

$$\sigma_q^2 = \frac{1}{\sum_{n=1}^{N} w_n} \qquad \sigma_\alpha^2 = \frac{1}{\lambda_1} . \tag{B.33}$$

In (Förstner and Wrobel, 2016, Section 9.4.2) it is shown that these variances are statistically independent. To derive the covariance matrix of the homogeneous line, we therefore introduce a fictive line

$$h : \{\mathbf{h}, \Sigma_{hh}\} = \left\{ \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 & 0 & 0 \\ 0 & \sigma_q^2 & 0 \\ 0 & 0 & \sigma_\alpha^2 \end{bmatrix} \right\} , \tag{B.34}$$

which represents the y-axis with an uncertainty in its position across and in its direction, independently from each other. Using the estimated parametres of line $\widehat{\mathbf{l}}$ we move line $\mathbf{h}$ on $\mathbf{l}$ by the transformation

$$\mathsf{T} = \begin{bmatrix} cos\phi & -\sin\phi & x_0 \\ \sin\phi & \cos\phi & y_0 \\ 0 & 0 & 1 \end{bmatrix}^{-\mathsf{T}} \qquad \phi = \alpha + \frac{\pi}{4} \tag{B.35}$$

and obtain its homogeneous coordinates and according covariance matrix via variance propagation.

$$\widehat{\mathbf{l}} = \mathsf{T}\mathbf{h} \tag{B.36}$$

$$\Sigma_{\widehat{l}\widehat{l}} = \mathsf{T}\Sigma_{hh}\mathsf{T}^{\mathsf{T}} . \tag{B.37}$$

Of course, (B.32) and (B.36) yield the same result. But, using (B.36), together with (B.37), we obtain the covariance information.

B.2 Representation of Ellipses

A conic is the intersection of a plane with a circular cone. This could be a hyperbola, a parabola or an ellipse with spatial case circle. In this section, we are interested in ellipses, solely. Depending on the application, we choose different representations for an ellipse which we introduce shortly in this section.

B.2.1 Homogeneous Representation of Ellipses.

We represent conics with the symmetric 3×3-matrix

$$\mathsf{C} = \left[\begin{array}{cc|c} c_{11} & c_{12} & c_{13} \\ c_{12} & c_{22} & c_{23} \\ \hline c_{13} & c_{23} & c_{33} \end{array}\right] = \begin{bmatrix} \mathsf{C}_{hh} & \boldsymbol{c}_{h0} \\ \boldsymbol{c}_{0h}^{\mathsf{T}} & c_{00} \end{bmatrix} . \tag{B.38}$$

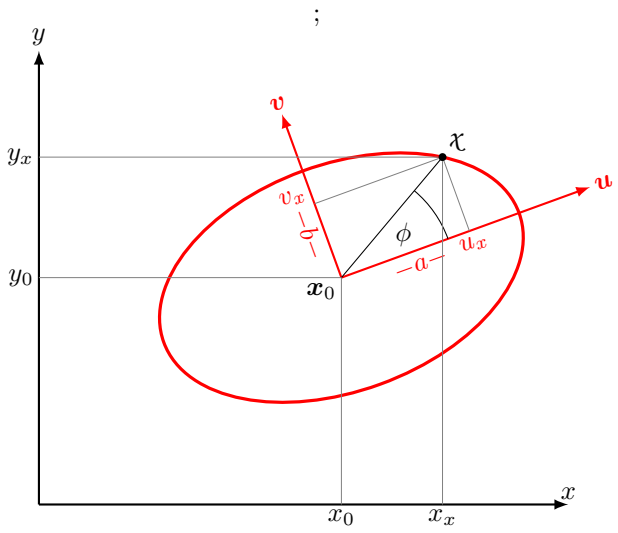


Figure B.2: The ellipse in parametric form is described by its centre $\boldsymbol{x}_0$, the length of major and minor semi-axes $a$and b and the rotation between these $(u, v)-$axes and the $(x, y)-$coordinate system.

Any point $\mathbf{x}$ on the conic fulfils

$$\mathbf{x}^{\mathsf{T}}\mathsf{C}\mathbf{x} = 0\,. \tag{B.39}$$

From (B.38) we see that C is described by 6 free parametres. Due to the homogeneous representation this results in 5 degrees of freedom. Thus, for any representation we need 5 parametres to describe the ellipse.

To ensure the conic to be an ellipse the homogeneous part of the conic must fulfil

$$|\mathsf{C}_{hh}| > 0\,. \tag{B.40}$$

Thus, when forcing the conic to be an ellipse we might constrain

$$|\mathsf{C}_{hh}| = 1\,. \tag{B.41}$$

This is a valid choice, as the conic representation is homogeneous.

B.2.2 Polynomial Representation of Ellipses.

We may rewrite (B.39) and (B.38) to obtain an implicit polynomial formulation

$$C(\boldsymbol{c}, \boldsymbol{y}) = \boldsymbol{c}^{\mathsf{T}} \cdot \boldsymbol{y} = a_{11}x^2 + 2a_{12}xy + a_{22}y^2 + 2a_{13}x + 2a_{23}y + a_{33} = 0 \tag{B.42}$$

collecting the parametres of the conic within vector $\boldsymbol{c} = [c_{11},\, c_{12},\, c_{22},\, c_{13},\, c_{23},\, c_{33}]^{\mathsf{T}}$ and the coordinates of the point within $\boldsymbol{y} = [x^2,\, 2xy,\, y^2,\, 2x,\, 2y,\, 1]^{\mathsf{T}}$, given by an euclidean point $\boldsymbol{x} = [x,\, y]^{\mathsf{T}}$. Again, we have 6 parametres, which we can scale arbitrary, from which we end up with 5 degrees of freedom.

B.2.3 Parametric Representation of Ellipses.

Sometimes, we need the geometric parametres of the ellipse explicit, e.g., for plotting or for representing an ellipse section. Figure B.2 visualises the geometric elements, we need to describe an ellipse. The point of symmetry or the centre of the ellipse, respectively, is given by

$$\boldsymbol{x}_0 = -\mathsf{C}_{hh}^{-1}\boldsymbol{c}_{h0}\,, \tag{B.43}$$

using the partition of $\mathbf{C}$ given in (B.38). We obtain the dimensions of the ellipse and the rotation between the semi-axis and the superior $(x, y)-$coordinate system by eigenvalue decomposition of the normalised homogeneous part of the conic

$$\begin{aligned} \mathsf{C}^e_{hh} &= \frac{-\mathsf{C}_{hh}}{c_{00} - \boldsymbol{c}^\mathsf{T}_{h0} C^{-1}_{hh} \boldsymbol{c}_{h0}} && \text{(B.44)} \\ &= R \Lambda R^\mathsf{T} , && \text{(B.45)} \end{aligned}$$

whereby, the eigenvectors $R = [\boldsymbol{u}, \boldsymbol{v}]$ define the local coordinate system of the ellipse. From the eigenvalues we evaluate the main radii of the ellipse by

$$a = \sqrt{\frac{1}{\lambda_2}} \qquad b = \sqrt{\frac{1}{\lambda_1}} , \tag{B.46}$$

where we assume the eigenvalues ordered, such that $\lambda_1 > \lambda_2$. Further, (B.45) leads to local (u, v) coordinates within the ellipse

$$\begin{bmatrix} u \\ v \end{bmatrix} = R^\mathsf{T} \left(\begin{bmatrix} x \\ y \end{bmatrix} - \begin{bmatrix} x_0 \\ y_0 \end{bmatrix} \right) . \tag{B.47}$$

Within the transformed coordinate systems, coordinates of points on the ellipse are given by

$$\begin{aligned} u(\theta) &= a \cos t && \text{(B.48)} \\ v(\theta) &= b \sin t \qquad t \in [0, 2\pi) , && \text{(B.49)} \end{aligned}$$

using the ellipse parametre t. Finally, the polar angle ϕ is given by

$$\phi(t) = \operatorname{atan} \left(\frac{b}{a} \tan(t) \right) . \tag{B.50}$$

Vice versa, the ellipse parametre, given a polar angle, is given by

$$t(\phi) = \operatorname{atan} \left(\frac{a}{b} \tan(\phi) \right) . \tag{B.51}$$

For plotting an ellipse segment for $\phi \in [\phi_1 \dots \phi_2]$ we obtain the coordinates in the superior $(x, y)-$system by

$$\begin{bmatrix} x(\phi) \\ y(\phi) \end{bmatrix} = \begin{bmatrix} x_0 \\ y_0 \end{bmatrix} + R \begin{bmatrix} u(t(\phi) \\ v(t(\phi)) \end{bmatrix} \tag{B.52}$$

B.3 Representation of Circles

B.3.1 Conic Representation of Circles

A circle is a special regular conic for which $\mathsf{C}_{hh} \propto I_2$.

To obtain the conic representation of an arbitrary circle, we may translate a circle with radius r at the origin

$$\mathsf{C}_0 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & -r^2 \end{bmatrix} \tag{B.53}$$

to the centre $\{x_0, y_0\}$ using the transformation

$$\mathsf{M} = \begin{bmatrix} 1 & 0 & x_0 \\ 0 & 1 & y_0 \\ 0 & 0 & 1 \end{bmatrix} . \tag{B.54}$$

The transformation yields

$$\mathsf{C} = \mathsf{M}^{-\mathsf{T}}\mathsf{C}_0\mathsf{M}^{-1} \tag{B.55}$$

$$= \begin{bmatrix} 1 & 0 & -x_0 \\ 0 & 1 & -y_0 \\ -x_0 & -y_0 & x_0^2 + y_0^2 - r^2 \end{bmatrix} . \tag{B.56}$$

Every point $\mathbf{x} = [x\, y\, 1]^\mathsf{T}$, which fulfils $\mathbf{x}^\mathsf{T}\mathsf{C}\mathbf{x} = 0$, lies on the circle.

Due to homogeneity we can scale every conic arbitrary, therefore, we can normalise a circle in conic representation, such that $\mathsf{C}_{hh} = \mathsf{I}_2$. This way, we can derive the circle parametre $\{x_0, y_0, r\}$ easily.

B.3.2 Homogeneous Representation of Circles

The conic representation, given in last section, is over-parametrised, as the matrix C has 9 elements, from which we need 3 parametres to describe the circle.

To derive a homogeneous representation for circles, we start from the implicit circle equation

$$(x - x_0)^2 + (y - y_0)^2 - r^2 = 0\,, \tag{B.57}$$

given a point $\boldsymbol{x} = [x\, y]$ sitting on the circle $\{x_0, y_0, r\}$. It is easily shown that rearranging $\mathbf{x}^\mathsf{T}\mathsf{C}\mathbf{x} = 0$, using $\mathbf{x} = [x\, y\, 1]^\mathsf{T}$ and the conic C given in (B.56), yields (B.57), too.
Rearranging (B.57) yields

$$1\left(x^2 + y^2\right) - 2x_0x - 2y_0y + x_0^2 + y_0^2 - r^2 = 0 \tag{B.58}$$

$$\begin{bmatrix} 1 \\ -2x_0 \\ -2y_0 \\ x_0^2 + y_0^2 - r^2 \end{bmatrix}^\mathsf{T} \cdot \begin{bmatrix} x^2 + y^2 \\ x \\ y \\ 1 \end{bmatrix} = 0 \tag{B.59}$$

$$\begin{bmatrix} A \\ B \\ C \\ D \end{bmatrix}^\mathsf{T} \cdot \begin{bmatrix} x^2 + y^2 \\ x \\ y \\ 1 \end{bmatrix} = 0. \tag{B.60}$$

Obviously, this is a homogeneous representation with one degree of freedom, thus, we are free to scale the parametre vector by any scalar. Equation (B.60) describes a circles only if $B^2 + C^2 - 4AD > 0$. Setting $A = 0$, Equation (B.60) allows us to represent a circle with infinite radius; thus, a line.

Al-Sharadqah and Chernov (2009) argue that (B.60) together with the constraint

$$B^2 + C^2 - 4AD = 1 \tag{B.61}$$

describe circles as well as lines, which ensures a proper least squares solution.

Given A, B, C, D we obtain centre and radius of the circle using

$$x_0 = -\frac{B}{2A} \qquad y_0 = -\frac{C}{2A} \qquad r = \sqrt{\frac{B^2 + C^2 - 4AD}{4A^2}}\,. \tag{B.62}$$

B.3.3 Direct Solution for Estimating the Homogeneous Circle Elements

In Section B.3.3 we summarise a direct unbiased solution for circle fitting as given in Al-Sharadqah and Chernov (2009). In this section, we give more details and motivate the resulting equations. Furthermore, we derive the covariance information of estimated parametres using implicit variance propagation .

Given a set of points $\boldsymbol{x}_n = [x_n\ y_n]^\mathsf{T}$, $n = 1 \ldots N$, supposed to sit on a circle, with covariance matrices $\boldsymbol{\Sigma}_{l_n l_n}$, which we assume to be $\sigma_n^2 \mathsf{I}_2$, we aim at estimating the circles parametre vector $\mathbf{p} = [A\ B\ C\ D]^\mathsf{T}$.

Using the constraints

$$g_n(\boldsymbol{x}_n, \mathbf{p}) = A(x_n^2 + y_n^2) + Bx_n + Cy_n + D = \mathbf{y}(\boldsymbol{x}_n)^\mathsf{T}\mathbf{p} \;, \tag{B.63}$$

as given in (B.60), we wish to minimise the squared sum of weighted algebraic distances

$$\Omega^2(\widehat{\mathbf{p}}) = \sum_n \frac{g_n^2}{\sigma_{g_n}^2} \,. \tag{B.64}$$

Assuming individual variances, the denominator in (B.64) is given by variance propagation

$$\sigma_{g_n}^2 = \boldsymbol{f}_n^\mathsf{T} \Sigma_{l_n l_n} \boldsymbol{f}_n \;, \tag{B.65}$$

using the Jacobian

$$\boldsymbol{f}_n^\mathsf{T} = \left. \frac{\partial g_n}{\partial \mathbf{y}_n} \frac{\partial \mathbf{y}_n}{\partial \boldsymbol{x}_n} \right|_{\boldsymbol{x}_n = \widehat{\boldsymbol{x}}_n} \tag{B.66}$$

$$= \mathbf{p}^\mathsf{T} \begin{bmatrix} 2\widehat{x}_n & 2\widehat{y}_n \\ 1 & 0 \\ 0 & 1 \\ 0 & 0 \end{bmatrix} \tag{B.67}$$

$$= \left[\, 2\widehat{x}_n A + B \mid 2\widehat{y}_n A + C \,\right] \;, \tag{B.68}$$

where we used $\mathbf{y}(\boldsymbol{x}_n) = \mathbf{y}_n$ to shorten the notation. This yields

$$\sigma_{g_n}^2 = \sigma_n^2 \left((2\widehat{x}_n A + B)^2 + (2\widehat{y}_n A + C)^2 \right) \tag{B.69}$$

$$= \sigma_n^2 4A \left(\widehat{x}_n^2 A + \widehat{x}_n B + \frac{B^2}{4A} + \widehat{y}_n^2 A + \widehat{y}_n C + \frac{C^2}{4A} \right) \tag{B.70}$$

$$= \sigma_n^2 4A \left(\underbrace{A\left(\widehat{x}_n^2 + \widehat{y}_n^2\right) + B\widehat{x}_n + C\widehat{y}_n + D}_{=0} + \frac{B^2}{4A} + \frac{C^2}{4A} - D \right) \tag{B.71}$$

$$= \sigma_n^2 \left(B^2 + C^2 - 4AD \right) \,. \tag{B.72}$$

Instead of using (B.72), Taubin (1991) suggested to replace the individual variances $\sigma_{g_n}^2$ in (B.64) by their mean variance

$$\overline{\sigma_{g_n}^2} = \frac{1}{N} \sum_N \sigma_{g_n}^2 \,. \tag{B.73}$$

Using (B.65) together with (B.67) yields

$$\overline{\sigma_{g_n}^2} = \frac{1}{N} \sum_n \sigma_n^2 \widehat{\mathbf{p}}^\mathsf{T} \begin{bmatrix} 4\widehat{x}_n^2 + 4\widehat{y}_n^2 & 2\widehat{x}_n & 2\widehat{y}_n & 0 \\ 2\widehat{x}_n & 1 & 0 & 0 \\ 2\widehat{y}_n & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \widehat{\mathbf{p}} \,. \tag{B.74}$$

Altogether the function we wish to minimise is given by

$$\Omega^2(\widehat{\mathbf{p}}) = \sum_n \frac{g_n^2(\widehat{\boldsymbol{x}}_n, \widehat{\mathbf{p}})}{\sigma_{g_n}^2} = \frac{\widehat{\mathbf{p}}^\mathsf{T} M\, \widehat{\mathbf{p}}}{\widehat{\mathbf{p}}^\mathsf{T} N\, \widehat{\mathbf{p}}} \;, \tag{B.75}$$

using

$$M = \sum_n \frac{\widehat{\mathbf{y}}_n \widehat{\mathbf{y}}_n^\mathsf{T}}{\sigma_n^2} = \begin{bmatrix} \overline{z^2} & \overline{xz} & \overline{yz} & \overline{z} \\ \overline{xz} & \overline{x^2} & \overline{xy} & \overline{x} \\ \overline{yz} & \overline{xy} & \overline{y^2} & \overline{y} \\ \overline{z} & \overline{x} & \overline{y} & 1 \end{bmatrix} \tag{B.76}$$

where we introduced the weighted mean values

$$\overline{z} = \frac{1}{N}\sum_n \frac{\widehat{x}_n^2 + \widehat{y}_n^2}{\sigma_n^2} \qquad \overline{x} = \frac{1}{N}\sum_n \frac{\widehat{x}_n}{\sigma_n^2} \qquad \overline{y} = \frac{1}{N}\sum_n \frac{\widehat{y}_n}{\sigma_n^2}\,. \tag{B.77}$$

The matrix N depends on the choice of the variances σ_n^2. When using (B.72), which was supposed by Pratt (1987), we obtain

$$N_P = \begin{bmatrix} 0 & 0 & 0 & -2 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -2 & 0 & 0 & 0 \end{bmatrix}. \tag{B.78}$$

But, when using the mean variance instead, as supposed by Taubin (1991), we obtain

$$N_T = \begin{bmatrix} 4\overline{z} & 2\overline{x} & 2\overline{y} & 0 \\ 2\overline{x} & 1 & 0 & 0 \\ 2\overline{y} & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}. \tag{B.79}$$

Furthermore, Al-Sharadqah and Chernov (2009) proposed to used

$$N = 2N_T - N_P \tag{B.80}$$

and showed that this choice leads to an unbiased solution.

Finally, the optimal parametres are given by minimising (B.75) which leads to the generalised eigenvalue problem

$$M\mathbf{p} = \lambda N\mathbf{p}\,. \tag{B.81}$$

The estimated parametres $\widehat{\mathbf{p}} = \left[\widehat{A}\ \widehat{B}\ \widehat{C}\ \widehat{D}\right]^\mathsf{T}$ are given by the eigenvector belonging to the smallest eigenvector.

The solution given in this section is based on an algebraic minimisation. We do not estimate adjusted observations nor the jacobians we usually use within the adjustment procedure. Therefore, we are not able to give the covariance of the estimated parametres directly. Instead we follow Förstner and Wrobel (2016, Section 3.6.2.5) and derive the covariance matrix of the circles parametres $[x_0, y_0, r]$ directly from observed points. To do so, we assume the observations without outlier; thus, accurate enough to assume the estimated parametres a good approximation of the true values.

First, we obtain centre $(\widehat{x}_0, \widehat{y}_0)$ and radius $\widehat{r}$ using (B.62). Given the implicit circle equations

$$g_n(\boldsymbol{x}_n, \widehat{\boldsymbol{p}}) = (x_n - \widehat{x}_0)^2 + (y_n - \widehat{y}_0)^2 - \widehat{r}^2 = 0\,, \tag{B.82}$$

using the classical circle parametres $\widehat{\boldsymbol{p}} = [\widehat{x}_0\ \widehat{y}_0\ \widehat{r}]$, we obtain the covariance of estimated parametres by implicit variance propagation, cf. (Förstner and Wrobel, 2016, Section 1.7.5)

$$\Sigma_{pp} = B^{-1}A\Sigma_{ll}A^\mathsf{T}B^{-\mathsf{T}}\,, \tag{B.83}$$

using the jacobians

$$A = \left[\boldsymbol{a}_n^\mathsf{T}\right] = \left[\left.\frac{\partial g_n(\boldsymbol{x}_n, \boldsymbol{p})}{\partial \boldsymbol{x}}\right|_{\boldsymbol{p}=\widehat{\boldsymbol{p}}}\right] \tag{B.84}$$

$$B = \left[\boldsymbol{b}_n^\mathsf{T}\right] = \left[\left.\frac{\partial g_n(\boldsymbol{x}_n, \boldsymbol{p})}{\partial \boldsymbol{p}}\right|_{\boldsymbol{p}=\widehat{\boldsymbol{p}}}\right]. \tag{B.85}$$

The first one is given by

$$A = \begin{bmatrix} 2(x_1 - \widehat{x}_0) & 2(y_1 - \widehat{y}_0) & 0 & 0 & 0 & 0 \\ \vdots & \vdots & \ddots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & 2(x_N - \widehat{x}_0) & 2(y_N - \widehat{y}_0) \end{bmatrix}, \tag{B.86}$$

from which we obtain for the inner part of (B.83)

$$\begin{aligned} A\Sigma_{ll}A^{\mathsf{T}} &= 4 \begin{bmatrix} \sigma_1^2 \left((x_1 - \widehat{x}_0)^2 + (y_1 - \widehat{y}_0)^2\right) & 0 & 0 \\ \vdots & \ddots & \vdots \\ 0 & 0 & \sigma_N^2 \left((x_N - \widehat{x}_0)^2 + (y_N - \widehat{y}_0)^2\right) \end{bmatrix} \\ &= 4\widehat{r}^2 \operatorname{diag}\left(\sigma_n^2\right) && \text{(B.87)} \\ &= 4\widehat{r}^2 W^{-1}\,. && \text{(B.88)} \end{aligned}$$

The second jacobian is given by

$$\begin{aligned} B &= \begin{bmatrix} -2(x_1 - \widehat{x}_0) & -2(y_1 - \widehat{y}_0) & -2\widehat{r} \\ \vdots & \vdots & \vdots \\ -2(x_N - \widehat{x}_0) & -2(y_N - \widehat{y}_0) & -2\widehat{r} \end{bmatrix} && \text{(B.89)} \\ &= -2\widehat{r} \begin{bmatrix} \frac{x_1 - \widehat{x}_0}{\widehat{r}} & \frac{y_1 - \widehat{y}_0}{\widehat{r}} & 1 \\ \vdots & \vdots & \vdots \\ \frac{x_N - \widehat{x}_0}{\widehat{r}} & \frac{y_N - \widehat{y}_0}{\widehat{r}} & 1 \end{bmatrix} && \text{(B.90)} \\ &= -2\widehat{r} \begin{bmatrix} c_1 & s_1 & 1 \\ \vdots & \vdots & \vdots \\ c_N & s_N & 1 \end{bmatrix} && \text{(B.91)} \\ &= -2\widehat{r}\bar{B}\,, && \text{(B.92)} \end{aligned}$$

where we used sines s_n and cosines c_n of the directions of given points to the circles center

$$s_n = \frac{x_n - \widehat{x}_0}{\widehat{r}} \qquad c_n = \frac{y_n - \widehat{y}_0}{\widehat{r}}\,. \tag{B.93}$$

Using (B.88) and (B.92), we rewrite (B.83)

$$\Sigma_{pp} = \bar{B}^{-1}W^{-1}\bar{B}^{-\mathsf{T}} = \left(\bar{B}^{\mathsf{T}}W\bar{B}\right)^{-1} = \begin{bmatrix} \overline{c^2} & \overline{cs} & \overline{c} \\ \overline{cs} & \overline{s^2} & \overline{s} \\ \overline{c} & \overline{s} & \overline{w} \end{bmatrix}^{-1} \tag{B.94}$$

with abbreviations

$$\begin{aligned} \overline{s} &= \sum_n \frac{s_n}{\sigma_n^2} & \overline{c} &= \sum_n \frac{c_n}{\sigma_n^2} & \overline{cs} &= \sum_n \frac{c_n s_n}{\sigma_n^2} \\ \overline{c^2} &= \sum_n \frac{c_n^2}{\sigma_n^2} & \overline{s^2} &= \sum_n \frac{s_n^2}{\sigma_n^2} & \overline{w} &= \sum_n \frac{1}{\sigma_n^2}\,. \end{aligned} \tag{B.95}$$

From the residuals

$$\widehat{v}_n = \widehat{r} - |\boldsymbol{x}_n - \widehat{\boldsymbol{x}}_0|\,, \tag{B.96}$$

we obtain the estimated variance factor

$$\widehat{\sigma_0}^2 = \frac{1}{N-3}\sum_n \frac{\widehat{v}_n}{\sigma_n^2}\,. \tag{B.97}$$

APPENDIX C

Optimisation

C.1 Golden section search

In Section 2.1.4.2 we used golden section search to find the minimum of an one-dimensional function. In this section, we give some details about this optimisation technique.

Golden section search was introduced in 1953 by Kiefer. A good introduction can be found in (Press et al., 2007, chap.10). This chapter includes an embedding into the whole area of optimisation techniques.

Basically the method determines the extremum of an unimodal, univariate function by sequentially dividing the search range into smaller values. We do not need the derivative, but we must be able to evaluate the function itself at every point.

Figure C.1 visualises the principle. Given a function $f(x)$, we wish to minimise, we start from a given range $[x_l, x_u]$, the lower and upper bound of the optimum $\widehat{x}$.

We introduce two interior points $x_1 = x_l + d$ and $x_2 = x_u - d$. In each iteration, one of the two interior points is selected as new bound:

- If $f(x_1) < f(x_2)$ we choose x_2 to be the new x_l and x_1 to be the new x_2.
- If $f(x_2) < f(x_1)$ we choose x_1 to be the new x_u and x_2 to be the new x_1.

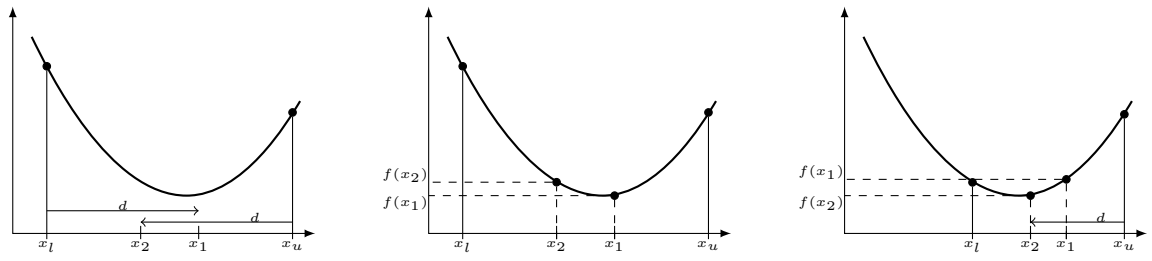


Figure C.1: Golden section search. **Left:** Given the upper and lower bound x_u und x_l, the interior points are given by $d = (\phi - 1)(x_u - x_l)$ using the golden number $\phi = 1.618$. **Middle:** Depending on the function values of interior points, the upper or lower bound are shifted to an interior point. **Right:** The process iterates until convergence.

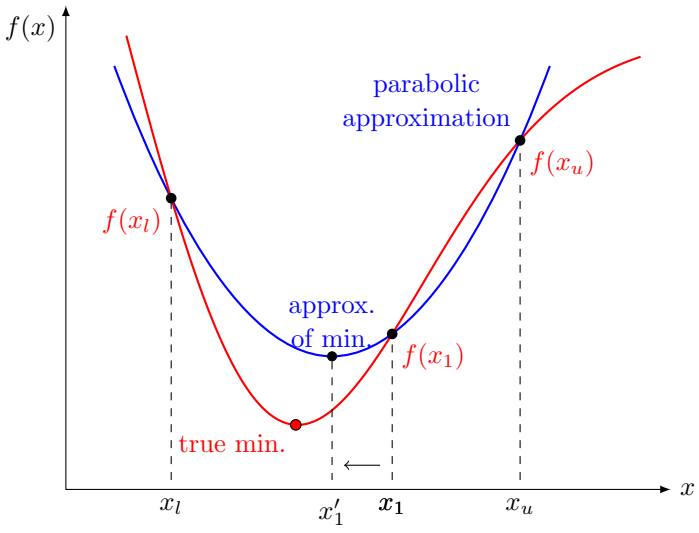


Figure C.2: Parabolic interpolation. We improve the location of x_1 by getting the minimum of the parabola through $\{x_l, x_1, x_u\}$.

The choice of d is such that the ratio of intervals

$$\frac{x_2 - x_l}{x_1 - x_2} = \frac{x_u - x_l}{x_1 - x_2} = \phi \tag{C.1}$$

holds the golden ratio

$$\phi = \frac{1 + \sqrt{5}}{2} = 1.680133. \tag{C.2}$$

Thus, d is given by

$$d = (\phi - 1) \cdot (x_u - x_l). \tag{C.3}$$

The termination condition is given in (Press et al., 2007, chap.10) with

$$|x_u - x_l| < \tau(|x_2| + |x_1|) \ , \tag{C.4}$$

where τ is the assumed tolerance of $|\widehat{x}|$. They recommend $\tau = \sqrt{\epsilon}$, where ϵ is the computational precision of $f(x)$. If (C.4) holds the optimum is given by

$$\widehat{x} = \frac{x_l + x_u}{2} \ . \tag{C.5}$$

To speed up convergence, we may include a parabolic interpolation. The principle is visualised in Figure C.2. Assume x_1 to be the lowest point we have seen so far. We can find a better approximation of the minimum of the true function by fitting a parabola to three points $\{x_l, x_1, x_u\}$ and estimating its minimum. A direct solution is given by

$$x_1' = x_1 - \frac{1}{2} \frac{(x_1 - x_l)^2 \left[f(x_1) - f(x_u)\right] - (x_1 - x_u)^2 \left[f(x_1) - f(x_l)\right]}{(x_1 - x_l) \left[f(x_1) - f(x_u)\right] - (x_1 - x_u) \left[f(x_1) - f(x_l)\right]} \tag{C.6}$$

This way, we reach an improved new interior point x_1' which we may use for the next iteration of golden section search.

Clearly, this methods fails if the points are nearly coplanar. In this case, we do not use this step.

C.2 Gauss-Helmert model

In Section 2.1.3 we estimate the parametres of an ellipse together with their covariance information, using a Gauss-Helmert model without constrains. In this section, we summarise the general solution of such an Gauss-Helmert model.

Assume N observations, collected in a vector $\boldsymbol{l}$, together with their covariance information Σ_{ll}. We look for U adjusted parametres collected in a vector $\widehat{\boldsymbol{x}}$. Furthermore, we seek for adjusted observations $\widehat{\boldsymbol{l}} = \boldsymbol{l} + \widehat{\boldsymbol{v}}$ given by their residuals $\widehat{\boldsymbol{v}}$.

Expressing the relation between observations and adjusted unknowns in G implicit constraints, the Gauss-Helmert model is given by

$$\boldsymbol{g}(\boldsymbol{l} + \widehat{\boldsymbol{v}}, \widehat{\boldsymbol{x}}) = 0 \; . \tag{C.7}$$

If the constraints are not linear, with respect to the parametres, we need to linearise using the Taylor expansion. We estimate corrections to the approximate values of the parametres in the linear substitute model, leading to updated approximate values.

The linearised Gauss-Helmert model is given by

$$\boldsymbol{g}\left(\widehat{\boldsymbol{l}^a}, \widehat{\boldsymbol{x}^a}\right) + A\widehat{\Delta x} + B^{\mathsf{T}}\widehat{\Delta l} = 0 \; , \tag{C.8}$$

using the Jacobian of $\boldsymbol{g}$ with respect to the reduced parametres and to the observations

$$\underset{N\times U}{A} = \left.\frac{\partial \boldsymbol{g}(\boldsymbol{l}, \boldsymbol{x})}{\partial \boldsymbol{x}}\right|_{\boldsymbol{l}=\widehat{\boldsymbol{l}}^a, \boldsymbol{x}=\widehat{\boldsymbol{x}}^a} \tag{C.9}$$

$$\underset{N\times G}{B^{\mathsf{T}}} = \left.\frac{\partial \boldsymbol{g}(\boldsymbol{l}, \boldsymbol{x})}{\partial \boldsymbol{l}}\right|_{\boldsymbol{l}=\widehat{\boldsymbol{l}}^a, \boldsymbol{x}=\widehat{\boldsymbol{x}}^a} \tag{C.10}$$

evaluated at an approximate set of parametres $\widehat{\boldsymbol{x}}^a$, approximate values of $\widehat{\boldsymbol{l}}^a$ and the differences between the current estimate and the approximate values

$$\widehat{\Delta x} = \widehat{\boldsymbol{x}} - \widehat{\boldsymbol{x}}^a \tag{C.11}$$

$$\widehat{\Delta l} = \widehat{\boldsymbol{l}} - \widehat{\boldsymbol{l}}^a \, . \tag{C.12}$$

Given the covariance of the observations Σ_{ll} and estimated residuals $\widehat{\boldsymbol{v}}$, we wish to minimise the weighted squared sum of residuals $\widehat{\boldsymbol{v}}^{\mathsf{T}}\Sigma_{ll}^{-1}\widehat{\boldsymbol{v}}$ subject to the constraints (C.7).

The solution can be found in McGlone (2004, Section 2.2.4.3) or Niemeier (2008, chap.5)

$$\widehat{\Delta \boldsymbol{x}} = \left(A^{\mathsf{T}}\left(B^{\mathsf{T}}\Sigma_{ll}B\right)^{-1}A\right)^{-1} A^{\mathsf{T}}\left(B^{\mathsf{T}}\Sigma_{ll}B\right)^{-1}\boldsymbol{c}_g \tag{C.13}$$

$$\widehat{\Delta \boldsymbol{l}} = \Sigma_{ll}B\left(B^{\mathsf{T}}\Sigma_{ll}B\right)^{-1}\left(\boldsymbol{c}_g - A\widehat{\Delta x}\right) \tag{C.14}$$

using the contradictions

$$\boldsymbol{c}_g = B^{\mathsf{T}}(\widehat{\boldsymbol{l}^a} - \boldsymbol{l}) - \boldsymbol{g}(\widehat{\boldsymbol{l}^a}, \widehat{\boldsymbol{x}^a}) \, . \tag{C.15}$$

We update parametres and observations

$$\widehat{\boldsymbol{x}} = \widehat{\boldsymbol{x}^a} + \widehat{\Delta x} \tag{C.16}$$

$$\widehat{\boldsymbol{l}} = \widehat{\boldsymbol{l}^a} + \widehat{\Delta l} \tag{C.17}$$

and iterate (C.9) to (C.17) until there is no change in the parametres any more.

After convergence, variance propagation yields the covariance of estimated parametres

$$\Sigma_{\widehat{x}\widehat{x}} = \left(A^{\mathsf{T}}\left(B^{\mathsf{T}}\Sigma_{ll}B\right)^{-1}A\right)^{-1} \; . \tag{C.18}$$

The variance coefficient for the observations is given by

$$\widehat{\sigma}_0^2 = \frac{\widehat{\boldsymbol{v}}^{\mathsf{T}} \boldsymbol{\Sigma}_{ll}^{-1} \widehat{\boldsymbol{v}}}{R} \tag{C.19}$$

with redundancy $R = G - U$.

C.3 Derivation of Green's Ratio

In Section 2.6.6 we introduce the algorithm for sampling unnormalised marked point process densities. The following section sketches the derivation of Green's Ratio which shows up as acceptance ratio in the marked point process sampling algorithm 2.5, page 86.

We set up a Markov chain in a state space $\mathcal{S} = \cup_i \mathcal{S}_i$, which is the union of several sub state spaces $\mathcal{S}_i$ that are potentially of different dimensionality. Greens method allow the samples of the chain to jump between these different subspaces.

Instead of comparing densities within the acceptance ratio, we have to be more formal in a way that we compare distributions $P(\mathrm{d}x) = \mathbb{P}\left(\underline{x} \in \mathrm{d}x\right)$ under an appropriate measure of volume (Andrieu et al., 2003).

First, we recap from probability theory the change of variable under different distributions. Assume a random variable $\underline{x}$ with continuous density $p_x(x)$. If we express a new random variable $\underline{y}$ as function $\underline{y} = h\left(\underline{x}\right)$, whereby h is a strictly increasing, continuously differentiable function with inverse $\underline{x} = g\left(\underline{y}\right)$. Then

$$p_y(y) = p_x(g(y)) \left| \frac{\partial g(y)}{\partial y} \right| \tag{C.20}$$

Obviously, observations in the range $[x, x + \Delta x]$ will be transformed by h into $[y, y + \Delta y]$ as long as Δx is small. Thus, $p_x(x)\Delta x \approx p_y(y)\Delta y$ and assuming both to be differentiable $p_x(x)\mathrm{d}x = p_y(y)\mathrm{d}y$. Using (C.20), we obtain

$$p_y(y)\mathrm{d}y = \left| \frac{\partial g(y)}{\partial y} \right| p_x(x)\mathrm{d}x \ . \tag{C.21}$$

The first term contains the partial derivatives of the functional with respect to involved variables, thus, the Jacobian. Please note $\mathrm{d}y$ and $\mathrm{d}x$, which realises the underlying measures of according distributions.

We construct a reversible Markov chain on a general state space $\mathcal{S}$ with invariant distribution P. The detailed balance condition, cf. Equation (2.134) page 81, reads as

$$\int_{(x,x')\in\mathcal{A},\mathcal{B}} P(\mathrm{d}x)\mathcal{Q}(\mathrm{d}x' \mid x) = \int_{(x,x')\in\mathcal{A},\mathcal{B}} P(\mathrm{d}x')\mathcal{Q}(\mathrm{d}x \mid x') \tag{C.22}$$

for all Borel sets $\mathcal{A}, \mathcal{B} \subset \mathcal{S}$. Drawing a new state x' from a proposal measure Q(dx'|x), i.e. a proposal distribution, the proposition kernel, i.e. the transition probability, cf. Equation (2.138) page 82, is given by

$$\mathcal{Q}(\mathrm{d}x'|x) = Q(\mathrm{d}x' \mid x)A(x' \mid x) \ . \tag{C.23}$$

Thus

$$\int_{(x,x')\in\mathcal{A},\mathcal{B}} P(\mathrm{d}x)Q(\mathrm{d}x' \mid x)A(x'|x) = \int_{(x,x')\in\mathcal{A},\mathcal{B}} P(\mathrm{d}x')Q(\mathrm{d}x \mid x')A(x|x') \ . \tag{C.24}$$

Green (1995) showed that $P(\mathrm{d}x)Q(\mathrm{d}x' \mid x)$ is dominated by a symmetric measure μ on $\mathcal{S} \times \mathcal{S}$. We call its Radon-Nikodym derivative with respect to μ; thus, its density, f, and rewrite the detailed balance equation

$$\int_{(x,x')\in\mathcal{A},\mathcal{B}} f(x' \mid x)A(x'|x)\mu(\mathrm{d}x, \mathrm{d}x') = \int_{(x,x')\in\mathcal{A},\mathcal{B}} f(x \mid x')A(x|x')\mu(\mathrm{d}x', \mathrm{d}x) \tag{C.25}$$

from which we see that the acceptance probability is given by

$$A(x'|x) = \min\left(1, \frac{f(x \mid x')}{f(x' \mid x)}\right) = \min\left(1, \frac{p(x')q(x \mid x')}{p(x)q(x' \mid x)}\right) \tag{C.26}$$

as expected.

In the following, we take into account the trans-dimensional case. Suppose $\mathcal{S} \subset \mathbb{R}^d$ and p to be a density with respect to a d-dimensional Lebesgue measure. We wish to draw a sample from $\mathcal{S}' \subset \mathbb{R}^{d'}$. Green (1995) proposed to overcome the dimensional jump by drawing auxiliary variables u from a joint density g, from which the proposal x' is given by the differentiable and invertible function $x' = \phi(x, u)$, which is a mapping $\phi : \mathbb{R}^d \times \mathbb{R}^r \to \mathbb{R}^{d'}$.

We rewrite the detailed balance equation

$$\int_{(x,x')\in\mathcal{A},\mathcal{B}} p(x)g(u)A(x' \mid x)\ \mathrm{d}x\ \mathrm{d}u = \int_{(x,x')\in\mathcal{A},\mathcal{B}} p(x')g'(u')A(x \mid x')\ \mathrm{d}x'\ \mathrm{d}u'\ , \tag{C.27}$$

where we assume $u' \sim g'$ and the existence of the inverse mapping

$$x = \phi^{-1}(x', u'),\ \ \phi^{-1} : \mathbb{R}^{d'} \times \mathbb{R}^{r'} \to \mathbb{R}^d\ .$$

Due to the change of variables, the product measures on both sides are related by

$$\mathrm{d}x'\ \mathrm{d}u' = \left|\frac{\partial\phi^{-1}(x', u')}{\partial(x, u)}\right| \mathrm{d}x\ \mathrm{d}u\ . \tag{C.28}$$

Again, noting the symmetry of the product measures $(\mathrm{d}x\ \mathrm{d}u)$ and $(\mathrm{d}x'\ \mathrm{d}u')$ this yields

$$p(x)g(u)A(x' \mid x) = p(x')g'(u')A(x \mid x')\left|\frac{\partial\phi^{-1}(x', u')}{\partial(x, u)}\right|\ . \tag{C.29}$$

And we see that detailed balance holds if the acceptance probability is given by

$$A(x'|x) = \min\left(1, \frac{p(x')}{p(x)}\frac{g(u')}{g(u)}\left|\frac{\partial(x', u')}{\partial(x, u)}\right|\right)\ . \tag{C.30}$$

Please note that the dimensionality of variables here remains invisible. As long as the mapping and its inverse is a diffeomorphism, dimensions of x and x' might be different. It requires that dimension-matching holds; thus, $d + r = d' + r'$. The symbol $p(x)$ remains the same independent of the current subspace.

Up to now, we omit the case of using a mixture of kernels as it is typically used to traverse the whole state space. But, this is straight forward and already given in Section 2.6.3. Here we shortly recap the result. Given a set of $m = 1 \dots M$ possible moves at a certain state x of the chain. The complete transition kernel then is given by summing over all move types

$$\mathcal{Q}(\mathcal{B} \mid x) = \sum_{m=1}^{M} \int_{x'\in\mathcal{B}} Q_m(\mathrm{d}x' \mid x)A(x' \mid x)\ . \tag{C.31}$$

Each move is chosen with probability $j_m(x)$ given the chain is at x. Accordingly, auxiliary variables u are drawn from densities g_m. Then the final expression for the acceptance probability of rjMCMC using a mixture of kernels is given by

$$A_m(x' \mid x) = \min\left(1, \frac{p(x')}{p(x)}\frac{j_m(x')}{j_m(x)}\frac{g'_m(u')}{g_m(u)}\left|\frac{\partial(x', u')}{\partial(x, u)}\right|\right) \tag{C.32}$$

as stated in (2.149).

C.3.1 Derivation of Green's Ratio for Marked Point Sampling

The generic point process sampler algorithm is given in Algorithm 2.5, which is equivalent to Hasting's and Green's Algorithm 2.3. Having M different moves, we take the proposition kernel of the Markov chain as a mixture of $m = 1 \dots M$ proposition distributions, each having a probability $j_m(\mathcal{X})$ to choose move type m being at $\mathcal{X}$. Depending on the chosen kernel we evaluate Green's ratio in Algorithm 2.5 line 4, formally given by

$$R_m\left(\mathcal{X}, \mathcal{Y}\right) = \frac{F\left(\mathrm{d}\mathcal{Y}\right)\ Q_m\left(\mathrm{d}\mathcal{X} \mid \mathcal{Y}\right)}{F\left(\mathrm{d}\mathcal{X}\right)\ Q_m\left(\mathrm{d}\mathcal{Y} \mid \mathcal{X}\right)} \tag{C.33}$$

Ortner et al. (2003) showed that $F\left(\mathrm{d}\mathcal{X}\right) Q_m\left(\mathrm{d}\mathcal{Y} \mid \mathcal{X}\right)$ is dominated by a symmetric measure $\xi(\mathrm{d}\mathcal{X}, \mathrm{d}\mathcal{Y})$. Which yield the following formulations for Green's ratio for different move types. Therefore, to clarify the notion, we will denote the move types not by numbers, but by letters. We will use $m \in \{b, d, nj\}$, standing for birth, death, and non jumping moves.

C.3.1.1 Birth and Death

In case of birth, we create a new object $\boldsymbol{x} \sim Q_b = \frac{\mu(\mathrm{d}\boldsymbol{x})}{\mu(\mathcal{S})}$ and propose $\mathcal{Y} = \mathcal{X} \cup \boldsymbol{x}$. Using the definition of the point process' distribution $F\left(\mathrm{d}\mathcal{X}\right) = f\left(\mathcal{X}\right) \pi_\mu\left(\mathrm{d}\mathcal{X}\right)$, we obtain

$$\begin{aligned} F\left(\mathrm{d}\mathcal{X}\right) Q_b\left(\mathrm{d}\mathcal{Y} \mid \mathcal{X}\right) &= f\left(\mathcal{X}\right) \pi_\mu\left(\mathrm{d}\mathcal{X}\right) j_b\left(\mathcal{X}\right) \frac{\mu(\mathrm{d}\boldsymbol{x})}{\mu(\mathcal{S})} \\ &= f\left(\mathcal{X}\right) j_b\left(\mathcal{X}\right) \frac{1}{\mu(\mathcal{S})}\ \xi(\mathrm{d}\mathcal{X}, \mathrm{d}\mathcal{Y})\ , \end{aligned} \tag{C.34}$$

using

$$\xi(\mathrm{d}\mathcal{X}, \mathrm{d}\mathcal{Y}) = \pi_\mu\left(\mathrm{d}\mathcal{X}\right) \mu(\mathrm{d}\boldsymbol{x})\ . \tag{C.35}$$

In case of death, we delete an object $\boldsymbol{x} \in \mathcal{X}$ and propose $\mathcal{Y} = \mathcal{X} \setminus \boldsymbol{x}$

$$\begin{aligned} F\left(\mathrm{d}\mathcal{X}\right) Q_d\left(\mathrm{d}\mathcal{Y} \mid \mathcal{X}\right) &= f\left(\mathcal{X}\right) \pi_\mu\left(\mathrm{d}\mathcal{X}\right) j_d\left(\mathcal{X}\right) \frac{1}{N\left(\mathcal{X}\right)} \\ &= f\left(\mathcal{X}\right) j_d\left(\mathcal{X}\right) \frac{1}{N\left(\mathcal{X}\right)}\ \xi(\mathrm{d}\mathcal{X}, \mathrm{d}\mathcal{Y})\ , \end{aligned} \tag{C.36}$$

using

$$\xi(\mathrm{d}\mathcal{X}, \mathrm{d}\mathcal{Y}) = \pi_\mu\left(\mathrm{d}\mathcal{X}\right)\ . \tag{C.37}$$

Using (C.34) and (C.36), (C.33) yields in case of birth

$$\begin{aligned} R_b\left(\mathcal{X}, \mathcal{Y}\right) &= \frac{F\left(\mathrm{d}\mathcal{Y}\right) Q_d\left(\mathrm{d}\mathcal{X} \mid \mathcal{Y}\right)}{F\left(\mathrm{d}\mathcal{X}\right) Q_b\left(\mathrm{d}\mathcal{Y} \mid \mathcal{X}\right)} \\ &= \frac{f\left(\mathcal{Y}\right) j_d\left(\mathcal{Y}\right) \mu(\mathcal{S})\ \xi\left(\mathrm{d}\mathcal{X}, \mathrm{d}\mathcal{Y}\right)}{f\left(\mathcal{X}\right) j_b\left(\mathcal{X}\right) N\left(\mathcal{Y}\right)\ \xi(\mathrm{d}\mathcal{X}, \mathrm{d}\mathcal{Y})} \\ &= \frac{j_d\left(\mathcal{Y}\right)}{j_b\left(\mathcal{X}\right)} \frac{f\left(\mathcal{Y}\right)}{f\left(\mathcal{X}\right)} \frac{\mu(\mathcal{S})}{N\left(\mathcal{Y}\right)} \end{aligned} \tag{C.38}$$

and in case of death

$$\begin{aligned} R_d\left(\mathcal{X}, \mathcal{Y}\right) &= \frac{F\left(\mathrm{d}\mathcal{X}\right) Q_b\left(\mathrm{d}\mathcal{Y} \mid \mathcal{X}\right)}{F\left(\mathrm{d}\mathcal{Y}\right) Q_d\left(\mathrm{d}\mathcal{X} \mid \mathcal{Y}\right)} \\ &= \frac{f\left(\mathcal{Y}\right) j_b\left(\mathcal{Y}\right) N\left(\mathcal{X}\right)\ \xi\left(\mathrm{d}\mathcal{X}, \mathrm{d}\mathcal{Y}\right)}{f\left(\mathcal{X}\right) j_d\left(\mathcal{X}\right) \mu(\mathcal{S})\ \xi(\mathrm{d}\mathcal{X}, \mathrm{d}\mathcal{Y})} \\ &= \frac{j_b\left(\mathcal{Y}\right)}{j_d\left(\mathcal{X}\right)} \frac{f\left(\mathcal{Y}\right)}{f\left(\mathcal{X}\right)} \frac{N\left(\mathcal{X}\right)}{\mu(\mathcal{S})}\ . \end{aligned} \tag{C.39}$$

C.3.1.2 Non Jumping Transformations

With this type of moves, we randomly perturb the marks of an existing object. To do so we uniformly select an object $\boldsymbol{x} \in \mathcal{X}$ and throw random numbers $u \sim Z_{(\mathcal{X},u)}$ according a suitable distribution. Given a function $\phi : \boldsymbol{y} = \phi(\boldsymbol{x}, u)$ which transforms the object we propose $\mathcal{Y} = \mathcal{X} \setminus \boldsymbol{x} \cup \boldsymbol{y}$. We will detail these moves in Section 5.3.

We construct the transformation ϕ such that the inverse transformation $\boldsymbol{x} = \phi^{-1}(\boldsymbol{y}, \boldsymbol{u})$ exist and is as possible as ϕ is, which ensures reversibility. Thus, if the translation and its inverse are equally probable and the according perturbation variables where thrown from the same distribution Green's ratio simplifies to

$$R_{nj}\left(\mathcal{X}, \mathcal{Y}\right) = \frac{f\left(\mathcal{Y}\right)}{f\left(\mathcal{X}\right)} \ , \tag{C.40}$$

which was shown by Ortner et al. (2003).

APPENDIX D

Further Results

D.1 More Results on DLCE

This section extends the results, shown in the experiments of our DLCE algorithm in Section 3.5. We show more results on synthetic images, Figures D.1 and D.2, and on image sections showing facade objects, Figures D.3 and D.4.

In Table D.1 to D.3 we report for all experiments the used parametres and characteristic values. We report number of line segments and running time, depending on the processing step and used pre-segmentation method.

In each table the second column gives the used pre-segmentation method, Douglas-Peucker (dp) algorithm or `circlePeucker` (cp). The third column refers to the number of evaluated pixel-chains per image. Fourth and fifth column give the number of line segments, using the according pre-segmentation method, and the final number of line segments after merging. Columns six to eight report the found numbers of each line segment type; thus, line, arc and ellipse segments. Columns nine to ten report the running times of both steps of the procedure. Remaining columns document the parametre setting for each individual experiment.

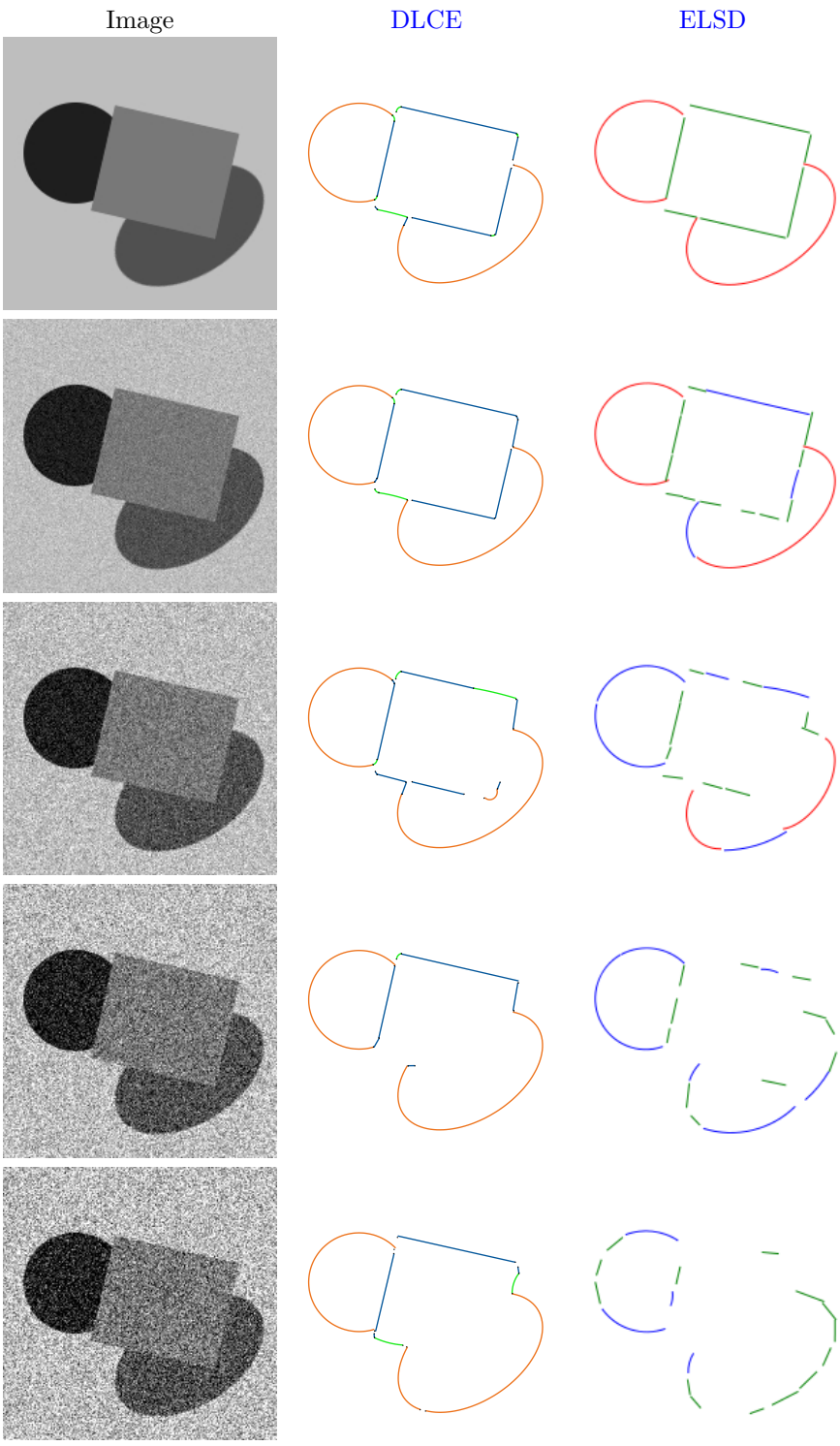


Figure D.1: Results of DLCE on synthetic data under different noise conditions for overlapping geometric objects and comparison to ELSD. Intensities are $30, 80, 120$ and 190 [gr] from dark circle to light background. We increase the noise level **from top to bottom** $\sigma_n = 0, 10, 20, 30, 40$ [gr]. **Colours DLCE:** Orange - ellipse segment. Green - circle segment. Blue - straight line segment. **Results of ELSD:** Red - ellipse segment. Blue - circle segment. Green - straight line segment.

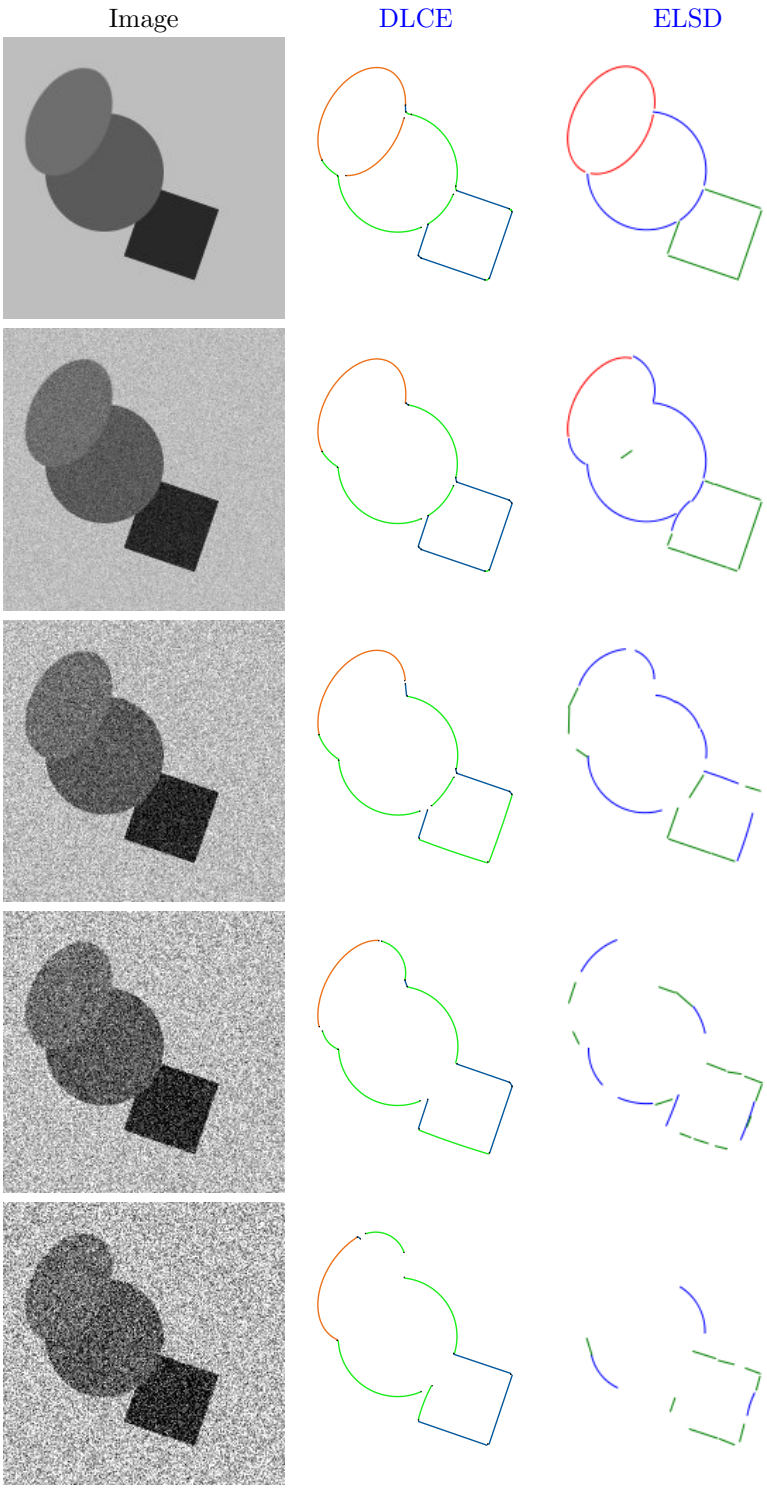


Figure D.2: Results of DLCE on synthetic data under different noise conditions for overlapping geometric objects and low contrast and comparison to ELSD. Intensities are $40, 90, 110$ and 190 [gr] from dark square to light background. We increase the noise level **from top to bottom** $\sigma_n = 0, 10, 20, 30, 40$ [gr]. Colours see Figure D.1.

Figure D.3: Balconies. Results of DLCE on facade objects. **Top row:** Image patches. **Middle:** Extracted pixel-chains. **Bottom:** Results of DLCE. Colours, see Figure D.1.

Figure D.4: Entrances. Results of DLCE on facade objects. **Top row:** Image patches. **Middle:** Extracted pixel-chains. **Bottom:** Results of DLCE. Colours, see Figure D.1.

Table D.1: Statistics of simplification. The image names refer to Figure 3.2, D.1 and D.2. Abbreviations: Methods for pre-segmentaion: cp - `circlePeucker`, dp - Douglas-Peucker. Ell - ellipse segments, Min p.l. - minimal pixel length for visualisation.

Image	Pre.Seg.	Number of						Time [sec.]		Parametre			
		Pix.Chains	Peucker	Merging	Lines	Arcs	Ell.	Peucker	Merging	σ_0[px]	t	Min p.l.[px]	σ_n[gr]
synth1	cp	3	18	10	4	5	1	0.8	1.1	0.2	3	20	3
	dp	3	63	11	6	4	1	0.0	3.5	0.2	3	20	3
synth1_s10	cp	3	15	9	5	3	1	1.0	1.1	0.3	3	20	10
	dp	3	48	12	7	4	1	0.0	2.4	0.3	3	20	10
synth1_s20	cp	3	12	7	3	3	1	0.8	0.9	0.4	3	20	20
	dp	3	40	15	10	4	1	0.0	2.0	0.4	3	20	20
synth1_s50	cp	3	13	8	4	3	1	1.0	0.9	0.5	3	20	30
	dp	3	35	18	13	4	1	0.0	1.7	0.5	3	20	30
synth1_s40	cp	4	13	8	3	3	2	1.0	0.9	0.6	3	20	30
	dp	4	26	14	11	2	1	0.0	1.5	0.6	3	20	30
synth2	cp	4	20	15	7	6	2	1.4	1.1	0.2	3	20	3
	dp	4	57	17	11	4	2	0.1	3.1	0.2	3	20	3
synth2_s10	cp	3	20	13	7	4	2	1.7	1.4	0.3	3	20	10
	dp	3	43	16	8	7	1	0.0	2.5	0.3	3	20	10
synth2_s20	cp	4	20	15	9	3	3	1.5	1.2	0.4	3	20	20
	dp	4	39	16	12	3	1	0.0	2.2	0.4	3	20	20
synth2_s30	cp	2	14	8	5	1	2	1.1	1.1	0.5	3	20	30
	dp	2	31	11	8	1	2	0.0	1.9	0.5	3	20	30
synth2_s40	cp	5	12	9	4	2	3	0.8	0.6	0.6	3	20	30
	dp	5	28	17	10	7	0	0.0	1.3	0.6	3	20	30
synth3	cp	2	21	16	6	8	2	1.2	0.8	0.2	3	20	3
	dp	2	53	15	7	6	2	0.0	2.1	0.2	3	20	3
synth3_s10	cp	1	18	14	8	5	1	1.1	0.8	0.3	3	20	10
	dp	1	37	14	8	5	1	0.0	1.7	0.3	3	20	10
synth3_s20	cp	3	13	12	5	6	1	0.9	0.5	0.4	3	20	20
	dp	3	31	21	15	6	0	0.0	1.1	0.4	3	20	20
synth3_s30	cp	3	12	11	5	5	1	0.8	0.4	0.5	3	20	30
	dp	3	24	16	10	5	1	0.0	0.8	0.5	3	20	30
synth3_s40	cp	3	10	9	4	4	1	0.7	0.4	0.6	3	20	30
	dp	3	23	19	14	5	0	0.0	0.9	0.6	3	20	30

Table D.2: Statistics of simplification. The image names refer to Figure 3.4 and 3.5. Abbreviations, see Table D.1.

Image	Pre.Seg.	Number of Pix.Chains	Peucker	Merging	Lines	Arcs	Ell.	Time [sec.] Peucker	Merging	Parametre σ_0[px]	t	Min p.l.[px]	σ_n[gr]
stop	cp	104	337	300	117	158	25	9.7	9.1	0.3	3	20	auto
	dp	94	500	412	278	131	3	0.7	13.9	0.3	3	20	auto
stop	cp	139	754	697	252	410	35	25.1	23.3	0.2	3	30	5
	dp	139	1642	1094	751	332	11	1.1	48.4	0.2	3	30	5
worm	cp	170	568	502	168	268	66	15.4	13.5	0.3	3	10	5
	dp	170	1036	726	403	316	7	0.7	23.9	0.3	3	10	5
gothic window	cp	161	445	411	146	230	35	14.6	12.9	0.3	3	20	auto
	dp	168	881	644	398	242	4	0.7	25.0	0.3	3	20	auto
icosah.	cp	492	1717	1600	620	891	89	54.8	49.6	0.3	3	30	3
	dp	492	2812	2340	1556	765	19	2.0	80.0	0.3	3	30	3
little mole	cp	51	562	479	118	312	49	29.4	20.7	0.2	5	30	auto
	dp	48	1115	784	433	345	6	0.7	34.0	0.2	5	30	auto

Table D.3: Statistics of simplification. The image names refer to Figure 3.6, D.3 and D.4. Abbreviations, see Table D.1.

Image	Pre.Seg.	Number of Pix.Chains	Peucker	Merging	Lines	Arcs	Ell.	Time [sec.] Peucker	Merging	Parametre σ_0[px]	t	Min p.l.[px]	σ_n[gr]
window 7115	cp	60	178	169	63	100	6	8.1	6.8	0.3	5	30	auto
	dp	60	286	233	148	85	0	0.2	10.3	0.3	5	30	auto
window 188	cp	66	241	236	115	113	8	9.8	8.3	0.3	5	30	auto
	dp	66	337	312	204	108	0	0.3	10.6	0.3	5	30	auto
window 15	cp	27	89	87	41	44	2	4.0	3.3	0.3	5	30	auto
	dp	27	122	115	79	36	0	0.1	3.8	0.3	5	30	auto
window 164	cp	12	47	45	23	22	0	3.1	2.1	0.3	5	30	auto
	dp	15	66	60	36	24	0	0.1	2.4	0.3	5	30	auto
window 2291	cp	10	27	25	13	10	2	1.1	0.9	0.3	5	30	auto
	dp	10	45	35	25	10	0	0.0	1.4	0.3	5	30	auto
balcony 45	cp	74	172	162	70	78	14	4.2	4.2	0.3	5	20	auto
	dp	74	258	241	162	79	0	0.2	6.0	0.3	5	20	auto
balcony 57	cp	173	462	442	206	229	7	15.4	15.1	0.3	5	30	auto
	dp	173	670	613	418	194	1	0.5	19.1	0.3	5	30	auto
balcony 60	cp	247	833	795	383	394	18	26.6	24.0	0.3	5	30	auto
	dp	247	1143	1082	764	318	0	0.9	30.1	0.3	5	30	auto
balcony 560	cp	75	297	291	133	152	6	8.2	7.8	0.3	5	30	auto
	dp	75	404	375	245	130	0	0.3	9.9	0.3	5	30	auto
balcony 691	cp	65	194	184	84	98	2	5.7	5.6	0.3	5	30	auto
	dp	65	260	249	181	68	0	0.2	6.7	0.3	5	30	auto
balcony 1026	cp	112	387	370	163	195	12	13.1	11.3	0.3	5	30	auto
	dp	112	540	498	320	178	0	0.4	14.5	0.3	5	30	auto
entrance 2	cp	80	221	216	90	122	4	7.7	7.1	0.3	5	30	auto
	dp	80	318	296	181	115	0	0.3	9.1	0.3	5	30	auto
entrance 14	cp	90	285	274	116	152	6	12.5	11.2	0.3	5	30	auto
	dp	90	403	378	250	128	0	0.3	14.2	0.3	5	30	auto
entrance 52	cp	85	278	268	103	159	6	11.6	10.3	0.3	5	30	auto
	dp	85	389	368	230	138	0	0.3	12.5	0.3	5	30	auto
entrance 99	cp	103	340	329	135	188	6	11.9	11.0	0.3	5	30	auto
	dp	103	476	431	274	154	3	0.4	14.3	0.3	5	30	auto
entrance 250	cp	137	560	529	254	264	11	16.2	14.7	0.3	5	30	auto
	dp	137	774	701	480	221	0	0.5	19.6	0.3	5	30	auto
entrance 283	cp	187	671	651	310	330	11	22.4	18.6	0.3	5	30	auto
	dp	187	959	869	566	303	0	0.7	25.6	0.3	5	30	auto

D.2 More Results on shapelets

In Section 4.6.2 we learn shapelets from PAS detected in training images. We evaluate the proposed method within a cross-validation framework and show one learned codebook, as an example. In order to complete the visualisation of results, Figures D.5 to D.8 show the codebooks, learned in the other cross-validation runs.

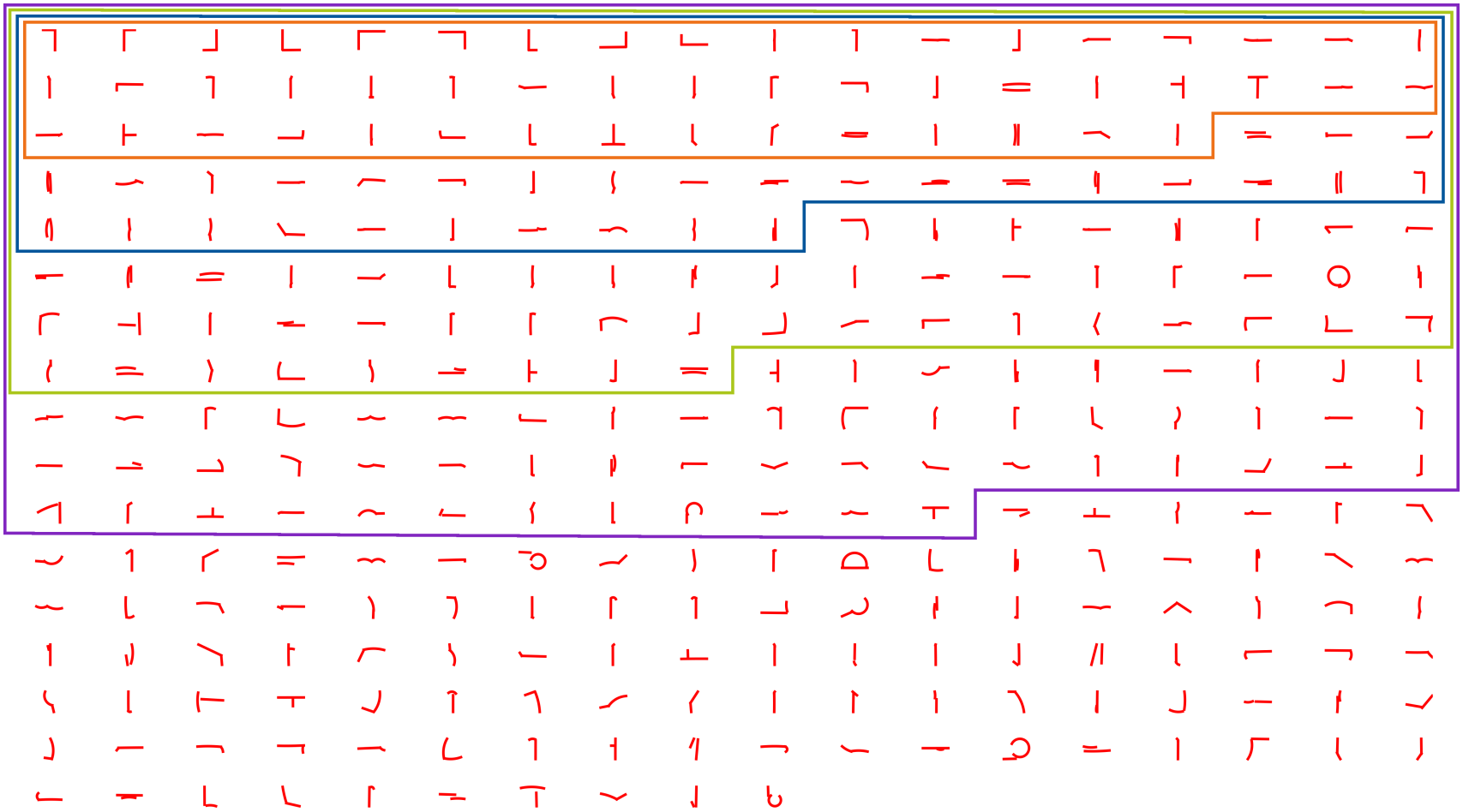


Figure D.5: **Shapelets learned in crossvalidation run 1**, ordered by the number of PAS belonging to the cluster. The coloured boxes mark the codebook size for different choices of N_t. The whole set belongs to the codebook using $N_t = 10$. Colours: violet: $N_t = 15$, green: $N_t = 20$, blue: $N_t = 30$, orange: $N_t = 50$.

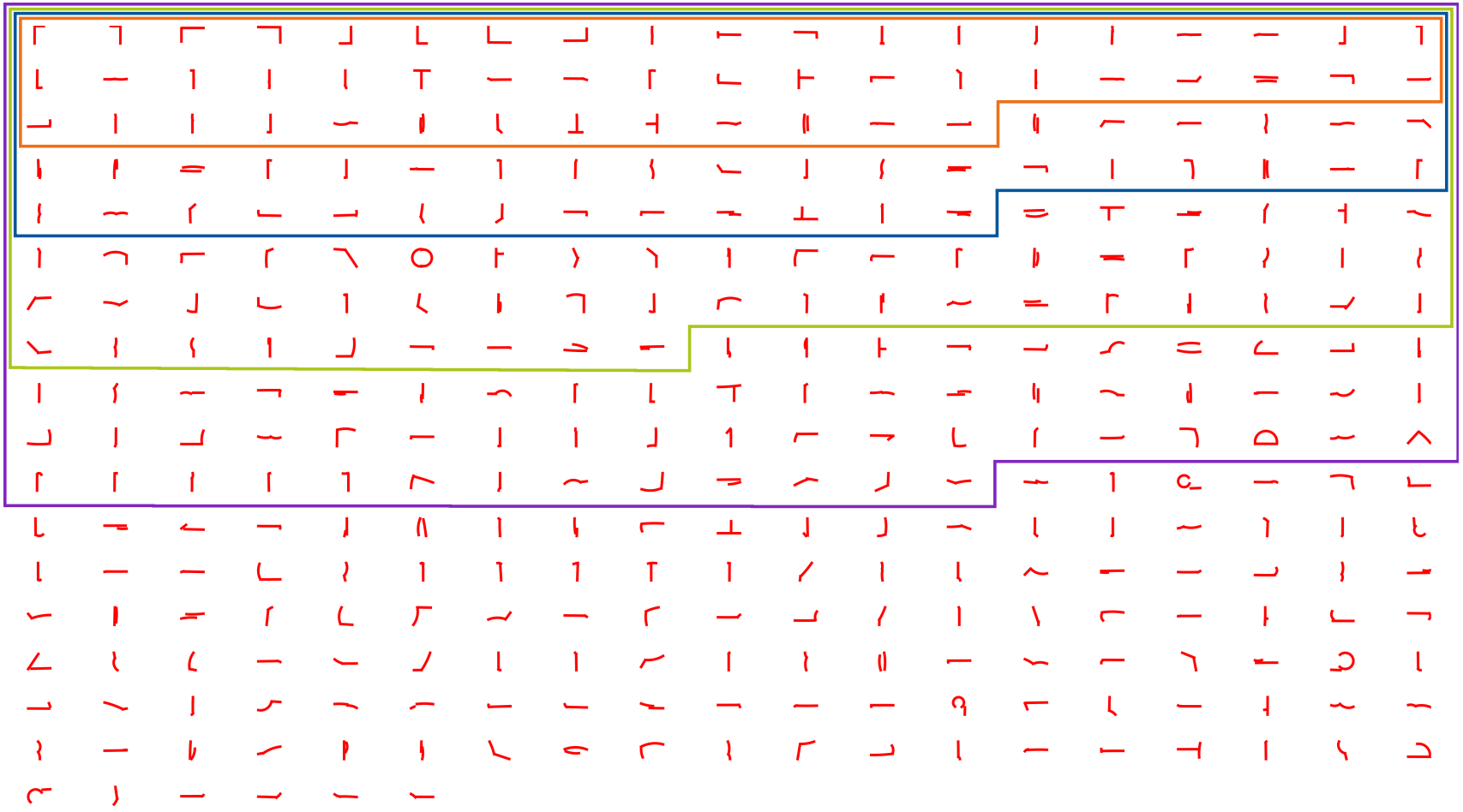


Figure D.6: **Shapelets learned in crossvalidation run 3**. For detailed description see Figure D.5.

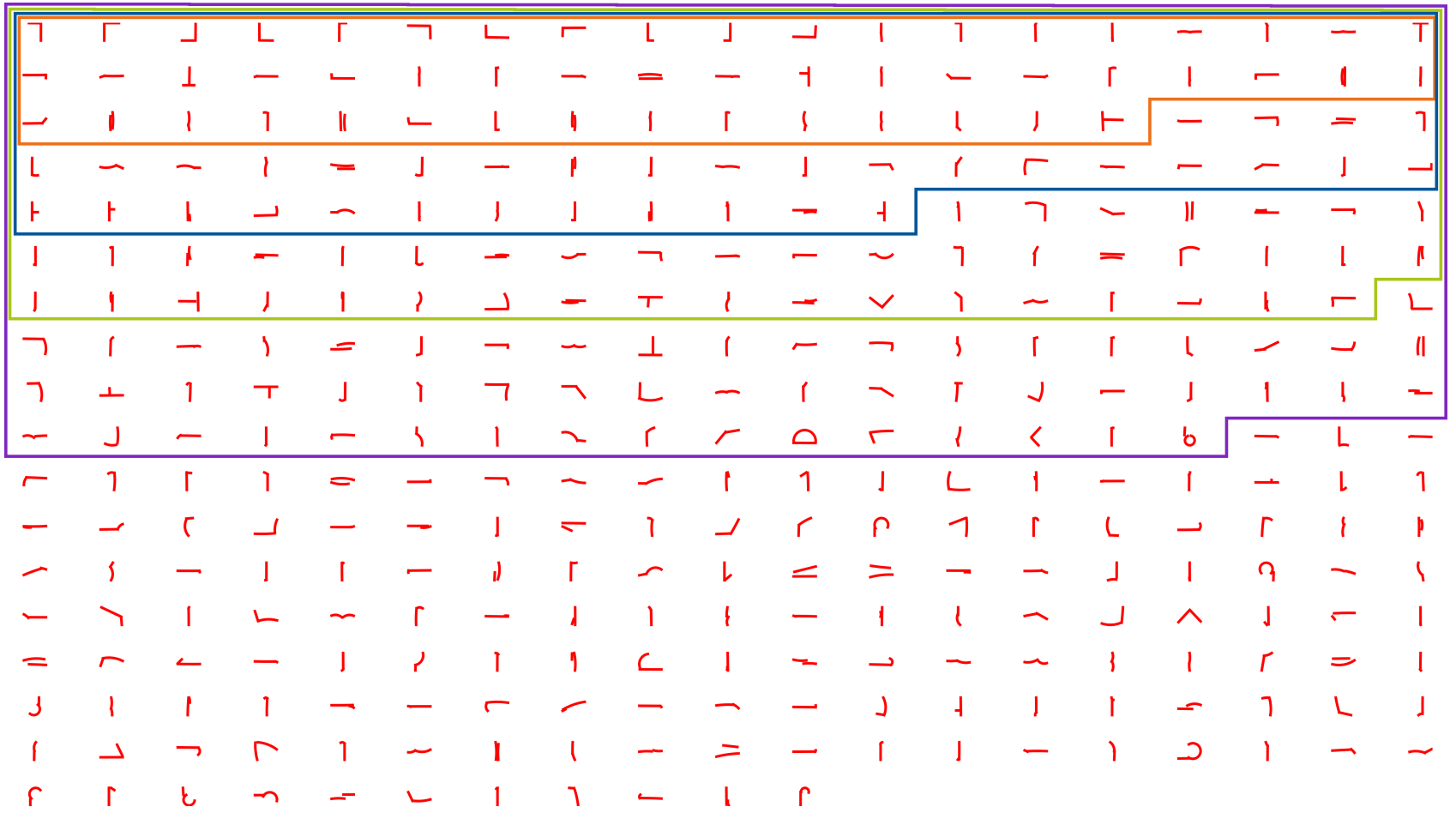


Figure D.7: **Shapelets learned in crossvalidation run 4**. For detailed description see Figure D.5.

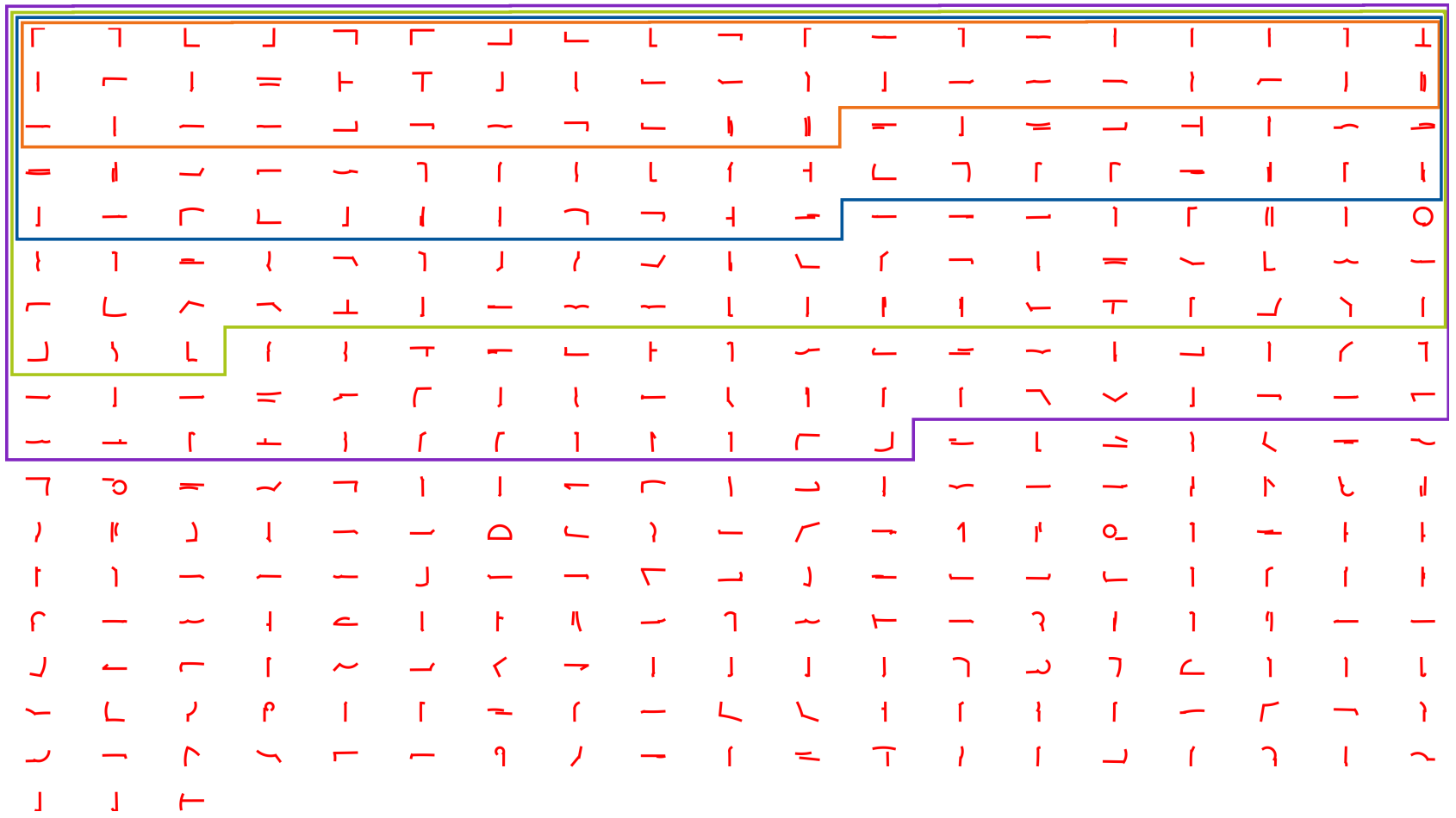


Figure D.8: **Shapelets learned in crossvalidation run 5**. For detailed description see Figure D.5.

List of Figures

1.1 3D building reconstruction, as shown in Werner and Zisserman (2002). 23
1.2 Results of window detection and reconstruction, as shown in Lee and Nevatia (2004) . 24
1.3 Results of window detection using an AdaBoost classifier, as shown in Ali et al. (2007). 24
1.4 Results of window detection, as shown in Recky and Leberl (2010) 24
1.5 Results of semi-supervised window detection, as shown in Wenzel and Förstner (2008) 24
1.6 Results of window pane detection, as shown in Čech and Šára (2008). 25
1.7 Results of blob detection, as shown in Jahangiri and Petrou (2009). 25
1.8 Results of window detection, as shown in Tyleček and Šára (2010). 26
1.9 Window detection and reconstruction, as shown in Reznik and Mayer (2008) 26
1.10 Facade image parsing, as shown in Berg et al. (2007). 27
1.11 Pixelwise labelling of facade images, as shown in Fröhlich et al. (2010). 27
1.12 Facade image segmentation on Haussmannian buildings, as shown in Teboul et al. (2010). 27
1.13 Facade image segmentation, as shown in Martinović et al. (2012). 28
1.14 Facade image segmentation and structure detection, as shown in Koziński et al. (2015). 28
1.15 Result as shown in Alegre and Dellaert (2004). 29
1.16 Result of nested repetitive structure detection, as shown in Wenzel et al. (2007). . . . 30
1.17 Result of propagating found repetitive structure for image restoration, as shown in Musialski et al. (2009). 30
1.18 Detecting and completing repetitive structure for image compression, as shown in Spinello et al. (2010) 30
1.19 Translational-symmetry-based perceptual grouping, as shown in Park et al. (2010) . . 31
1.20 Semantic 3D reconstruction, as shown in Dick et al. (2004). 32
1.21 Grammar based facade interpretation, as shown in Brenner and Ripperda (2006) and Ripperda (2008) 32
1.22 Results as shown in Müller et al. (2007). 33
1.23 System overview. 41
1.24 Organisation of the thesis 42

2.1 Ellipse segment 48
2.2 Orientation of circle segment 54
2.3 Distance measure of points χ to a given arc $\mathcal{S}$ 54
2.4 Geometric elements describing a circle segment 55
2.5 Bag of Words 62
2.6 BoW for image classification 63
2.7 SVM, margin and slack variables 68
2.8 Principle of using kernel function instead of features itself. 70
2.9 Homogeneous spatial point process 71
2.10 Homogeneous spatial point process with few points 71
2.11 Inhomogeneous spatial point process 72
2.12 Marked point process 72

2.13 Monte Carlo sampling to evaluate the partition function . . . 78
2.14 Direct sampling . . . 78
2.15 Rejection sampling . . . 79
2.16 Markov Chain . . . 80
2.17 Example: State space of Discrete Markov Chain . . . 80
2.18 Example: Evolution of a discrete Markov chain to its stationary distribution. . . . 81
2.19 Simulated annealing . . . 88
2.20 Simulated annealing cooling schedules . . . 89

3.1 Overview of the DLCE procedure . . . 92
3.2 Results of DLCE on synthetic data under different noise conditions. . . . 99
3.3 The effect of pre-segmentation on DLCE, synthetic images. . . . 101
3.4 The effect of pre-segmentation on DLCE, natural images. . . . 102
3.5 Results of DLCE on real images. . . . 104
3.6 Results on facade objects, windows. . . . 105

4.1 Shapelets, junction types . . . 109
4.2 Effect of rounding corners . . . 110
4.3 Parametrising PAS and its segments. . . . 111
4.4 Distribution of PAS distances . . . 113
4.5 An example of a learned codebook of shapelets. . . . 114
4.6 Tiling the image patches to encode spatial layout. . . . 115
4.7 BoW feature vector for 2×2 tiles . . . 116
4.8 Examples of given image sections. . . . 118
4.9 Learned shapelets. . . . 120
4.10 Overall accuracy using different tiling and different cluster size. . . . 122
4.11 Average class accuracy using different tiling and different cluster size.. . . . 123
4.12 Classwise accuracy using different tiling and different cluster size. . . . 124
4.13 Classification performance of HOG . . . 126

5.1 Integral image . . . 131
5.2 Integral histogram . . . 132
5.3 Results of sliding window detection . . . 134
5.4 Unary Energy Term . . . 136
5.5 Properties of neighbouring objects to describe the configuration. . . . 137
5.6 Intersection distance. . . . 137
5.7 Unary statistics . . . 138
5.8 Pairwise statistics . . . 139
5.9 Penalty functions . . . 140
5.10 Penalty function used in U_{num}. . . . 141
5.11 Sample images from used datasets. . . . 145
5.12 Simulations based on learned configuration statistics. . . . 147
5.13 Results for Basel old town dataset. . . . 148
5.14 Results for Basel row houses dataset. . . . 149
5.15 Results for dataset city houses with balconies. . . . 150
5.16 Local minima. . . . 154
5.17 Influence of the prior on the number of objects . . . 155

B.1 Homogeneous coordinates . . . 165
B.2 Parametric form of an ellipse . . . 168

C.1 Golden section search . . . 175
C.2 Parabolic interpolation. . . . 176

D.1 Results of DLCE on synthetic data for overlapping geometric objects. 184
D.2 Results of DLCE on synthetic data with low contrast. 185
D.3 Balconies. Results of DLCE on facade objects. 186
D.4 Entrances. Results of DLCE on facade objects. 187
D.5 Shapelets learned in crossvalidation run 1. 192
D.6 Shapelets learned in crossvalidation run 3. 193
D.7 Shapelets learned in crossvalidation run 4. 194
D.8 Shapelets learned in crossvalidation run 5. 195

List of Tables

2.1 Conditioning of points, lines and conics 57

4.1 Confusion matrix using $N_t = 10$, $\mathbf{3} \times \mathbf{5}$ tiles and L_2 weighting. 122
4.2 Confusion matrix HOG descriptor with 6×5 tiling. 127

5.1 Confusion matrices for object based evaluation. 151
5.2 Parameter differences of detected objects with respect to ground truth. 151

D.1 Statistics of simplification, synthetic images. 188
D.2 Statistics of simplification, natural images. 189
D.3 Statistics of simplification, facade objects. 190

List of Algorithms

2.1 Rejection sampling . . . 79
2.2 Metropolis-Hastings Algorithm . . . 82
2.3 Metropolis-Hastings-Green Algorithm for rjMCMC . . . 84
2.4 Basic Birth and Death sampler . . . 85
2.5 Generic point process sampler . . . 86
2.6 Optimisation with Metropolis-Hastings algorithm and simulated annealing . . . 90

3.1 Algorithm `circlePeucker` . . . 93
3.2 Algorithm `mergeSegments` . . . 95

5.1 Algorithm `IntegralHistogram` . . . 132

Bibliography

Akaike, H. (1974). A new look at the statistical model identification. *IEEE Transactions on Automatic Control 19*(6), 716 – 723. 61

Al-Sharadqah, A. and N. Chernov (2009). Error analysis for circle fitting algorithms. *Electron. J. Stat. 3*, 886–911. 52, 170, 172

Albano, A. (1974). Representation of Digitized Contours in Terms of Conic Arcs and Straight-Line Segments. *Computer Graphics and Image Processing 3*(1), 23 – 33. 96

Alegre, F. and F. Dellaert (2004). A probabilistic approach to the semantic interpretation of building facades. In *International Workshop on Vision Techniques Applied to the Rehabilitation of City Centres*, pp. 1–12. 20, 29, 35, 36, 197

Ali, H., C. Seifert, N. Jindal, L. Paletta, and G. Paar (2007). Window Detection in Facades. In *ICIAP '07*, pp. 837–842. 22, 24, 33, 34, 197

Amit, Y., D. Geman, and X. Fan (2004). A coarse-to-fine strategy for multi-class shape detection. *TPAMI 28*, 1606–1621. 43

Andrieu, C., N. de Freitas, A. Doucet, and M. I. Jordan (2003). An Introduction to MCMC for Machine Learning. *Machine Learning 50*(1-2), 5–43. 79, 80, 83, 88, 178

Baddeley, A. J. (2007). *Lecture Notes in Mathematics: Stochastic Geometry*, Volume 1892, Chapter Spatial Point Processes and their Applications, pp. 1–75. Springer Verlag , Berlin Heidelberg. 73, 74, 75

Baddeley, A. J. and M. N. M. V. Lieshout (1993). Stochastic geometry models in high-level vision. *J. of Appl. Stat. 20*(5-6), 231–256. 76, 143

Becker, S. (2009, November). Generation and application of rules for quality dependent facade reconstruction. *P&RS 64*(6), 640–653. 36

Becker, S. (2010). *Automatische Ableitung und Anwendung von Regeln für die Rekonstruktion von Fassaden aus heterogenen Sensordaten*. Ph. D. thesis, Universität Stuttgart. 36

Beichl, I. and F. Sullivan (2000). The metropolis algorithm. *Computing in Science & Engeneering 2*(1), 65–69. 77

Belongie, S., J. Malik, and J. Puzicha (2002, April). Shape matching and object recognition using shape contexts. *TPAMI 24*(4), 509–522. 116

Bengio, Y. (2009). Learning Deep Architectures for AI. *Foundations and Trends in Machine Learning 2*(1), 1–127. 159

Berg, A. C., F. Grabler, and J. Malik (2007). Parsing Images of Architectural Scenes. In *ICCV*, pp. 1–8. 23, 27, 34, 37, 197

Bishop, C. M. (2006, August). *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer. 65, 69, 79

Börcs, A. and C. Benedek (2012). A Marked Point Process Model for Vehicle Detection in Aerial Lidar Point Clouds. In *Int. Ann. Photogramm. Remote Sens. (ISPR'12)*, Volume I-3, pp. 93–98. 43

Braun, C., K. T. H., F. Lang, W. Schickler, V. Steinhage, A. Cremers, W. Förstner, and L. Plümer (1995). Models for photogrammetric building reconstruction. *Computers & Graphics 19*(1), 109–118. 22

Bredif, M., O. Tournaire, B. Vallet, and N. Champion (2013). Extracting polygonal building footprints from digital surface models: A fully-automatic global optimization framework. *P&RS 77*(0), 57 – 65. 43, 83

Brenner, C. and N. Ripperda (2006). Extraction of Facades using RJMCMC and Constraint Equations. In *PCV*, Volume XXXVI. 31, 32, 36, 38, 83, 197

Burochin, J.-P., B. Vallet, M. Bredif, C. Mallet, T. Brosset, and N. Paparoditis (2014). Detecting blind building facades from highly overlapping wide angle aerial imagery . *P&RS 96*, 193 – 209. 23, 38, 145

Canny, J. (1986, November). A Computational Approach to Edge Detection. *TPAMI 8*(6), 679–698. 97

Čech, J. and R. Šára (2008). Windowpane detection based on maximum aposteriori labeling. In *IWCIA*. 20, 23, 25, 37, 38, 197

Chai, D., W. Forstner, and F. Lafarge (2013). Recovering line-networks in images by junction-point processes. In *CVPR*. 144

Chai, D., W. Förstner, and M. Ying Yang (2012). Combine markov random fields and marked point processes to extract building from remotely sensed images. In *Int. Ann. Photogramm. Remote Sens. (ISPR'12)*, Volume I-3, pp. 365–370. 144, 159

Chang, C.-C. and C.-J. Lin (2011). LIBSVM: a library for support vector machines. *ACM Transactions on Intelligent Systems and Technology 2*(3), 27:1–27:27. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`. 69, 119

Chia, A., D. Rajan, M. Leung, and S. Rahardja (2012, November). Object recognition by discriminative combinations of line segments, ellipses and appearance features. *TPAMI 34*(9), 1758–1772. 92, 117

Cohen, A., A. Schwing, and M. Pollefeys (2014). Efficient structured parsing of facades using dynamic programming. In *CVPR*, pp. 3206–3213. 23, 37

Comaniciu, D. and P. Meer (2002, May). Mean shift: A robust approach toward feature space analysis. *TPAMI 24*(5), 603–619. 64, 113

Cootes, T. F., C. J. Taylor, D. H. Cooper, and J. Graham (1995, January). Active shape models—their training and application. *CVIU 61*(1), 38–59. 116

Csurka, G., C. Dance, L. Fan, J. Willamowski, and C. Bray (2004). Visual categorization with bags of keypoints. In *In Workshop on Statistical Learning in Computer Vision, ECCV*, pp. 1–22. 62, 64, 67

Cula, O. G. and K. J. Dana (2001). Compact representation of bidirectional texture functions. In *CVPR*. 62

Dalal, N. and B. Triggs (2005). Histograms of Oriented Gradients for Human Detection. In *CVPR*, Volume 1, pp. 886–893. 116, 117, 125

Dehbi, Y., F. Hadiji, G. Gröger, K. Kersting, and L. Plümer (2016). Statistical Relational Learning of Grammar Rules for 3D Building Reconstruction. *Transactions in GIS (accepted)*. 36, 38

Dehbi, Y. and L. Plümer (2011). Learning grammar rules of building parts from precise models and noisy observations. *P&RS 66*(2), 166 – 176. Quality, Scale and Analysis Aspects of Urban City Models. 36

Descombes, X., R. Minlos, and E. Zhizhina (2009). Object Extraction Using a Stochastic Birth-and-Death Dynamics in Continuum. *Journal of Mathematical Imaging and Vision 33*(3), 347–359. 142, 144

Descombes, X., R. Stoica, L. Garcin, and J. Zerubia (2001). A RJMCMC Algorithm for Object Processes in Image Processing. *Monte Carlo Methods and Applications 7*(1-2), 149–156. 143

Dick, A. R., P. H. S. Torr, and R. Cipolla (2004, November). Modelling and Interpretation of Architecture from Several Images. *IJCV 60*, 111–134. 31, 32, 35, 36, 197

Douglas, D. H. and T. K. Peucker (1973, December). Algorithms for the Reduction of the Number of Points Required to Represent a Digitized Line or its Caricature. Cartographica: The International Journal for Geographic Information and Geovisualization *10*(2), 112–122. 93

Elidan, G., G. Heitz, and D. Koller (2006). Learning Object Shape: From Drawings to Images. In *CVPR*, Volume 2, pp. 2064–2071. 116, 117

Fei-Fei, L. (2005). Turorial on Recognizing and Learning Object Categories, Part 1: Bag-of-words models, ICCV 2005. http://people.csail.mit.edu/torralba/shortCourseRLOC/index.html. 62, 64

Fei-Fei, L., R. Fergus, and P. Perona (2004). Learning generative visual models from few training examples: an incremental Bayesian approach tested on 101 object categories. In Workshop on Generative-Model Based Vision (CVPR). 62, 64

Fei-Fei, L. and P. Perona (2005). A Bayesian Hierarchical Model for Learning Natural Scene Categories. In *CVPR*, Volume 2, pp. 524–531. 64, 67

Fergus, R., P. Perona, and A. Zisserman (2003). Object class recognition by unsupervised scale-invariant learning. In *CVPR*, Volume 2, pp. 264–271. 43, 62

Ferrari, V., L. Fevrier, F. Jurie, and C. Schmid (2008). Groups of adjacent contour segments for object detection. *TPAMI 30*(1), 36–51. 64, 65, 108, 111, 113, 116, 117, 125

Ferrari, V., F. Jurie, and C. Schmid (2010, May). From Images to Shape Models for Object Detection. *IJCV 87*(3), 284–303. 117, 125

Ferrari, V., T. Tuytelaars, and L. J. V. Gool (2001). Real-time affine region tracking and coplanar grouping. In *CVPR*, pp. 226–233. 65

Ferrari, V., T. Tuytelaars, and L. J. V. Gool (2006). Object Detection by Contour Segment Networks. In *ECCV*, Volume 3953, pp. 14–28. 117

Fidler, S., D. Skocaj, and A. Leonardis (2006). Combining Reconstructive and Discriminative Subspace Methods for Robust Classification and Regression by Subsampling. *TPAMI 28*(3), 337–350. 43

Fitzgibbon, A. W.and Pilu, M. and R. B. Fisher (1999, May). Direct least-squares fitting of ellipses. *TPAMI 21*(5), 476–480. 48, 49

Förstner, W. (1994). A Framework for Low-Level Feature Extraction. In *ECCV*, Volume 801/1994, pp. 383–394. 97

Förstner, W. (2000). Image Preprocessing for Feature Extraction in Digital Intensity, Color and Range Images. In *Geomatic Methods for the Analysis of Data in Earth Sciences*, Volume 95/2000, pp. 165–189. Springer. 97

Förstner, W. and B. Wrobel (to appear 2016). *Photogrammetric Computer Vision - Statistics, Geometry, Orientation and Reconstruction*. Springer. 47, 52, 163, 164, 165, 166, 167, 172

Fröhlich, B., E. Rodner, and J. Denzler (2010). A Fast Approach for Pixelwise Labeling of Facade Images. In *Proc. of 20th International Conference on Pattern Recognition (ICPR'10)*, pp. 3029–3032. 23, 27, 34, 38, 197

Fuchs, C. and W. Förstner (1995). Polymorphic Grouping for Image Segmentation. In *Proc. of the 5th ICCV 1995*. 97

Gander, W., G. Golub, and R. Strebel (1994). Least-squares fitting of circles and ellipses. BIT Numerical Mathematics *34*(4), 558–578. 49

Gangaputra, S. and D. Geman (2006). A design principle for coarse-to-fine classification. In *CVPR*, Volume 2, pp. 1877–1884. 43

Garcin, L., X. Descombes, H. Le Men, and J. Zerubia (2001). Building detection by Markov object processes. In *ICIP*, Volume 2, pp. 565–568. online available as pre-version in techreport Garcin et al. (2001). 143, 208

Ge, W. and R. Collins (2009). Marked point processes for crowd counting. In *CVPR*. 144

Geman, S. and D. Geman (1984, November). Stochastic Relaxation, Gibbs Distributions, and the Bayesian Restoration of Images. *TPAMI 6*(6), 721–741. 87

Geyer, C. J. and J. Møller (1994). Simulation Procedures and Likelihood Inference for Spatial Point Processes. *Scandinavian Journal of Statistics 21*(4), pp. 359–373. 86, 143

Green, P. J. (1995). Reversible jump Markov chain Monte Carlo computation and Bayesian model determination. *Biometrika 82*(4), 711–732. 83, 84, 86, 143, 178, 179

Grenander, U. and M. I. Miller (1994). Representations of Knowledge in Complex Systems. Journal of the Royal Statistical Society. Series B (Methodological) *56*(4), pp. 549–603. 83

Gruen, A. (1997). Automation in Building Reconstruction. In *Photogrammetric Week '97*, pp. 175–186. 22

Gruen, A., O. Kuebler, and P. Agouris (Eds.) (1995). *Automatic Extraction of Man-Made Objects from Aerial Space Images*. Birkhäuser Basel. 22

Günther, O. and E. Wong (1990). The Arc Tree: An Apprcximation Scheme to Represent Arbitrary Curved Shapes. *Computer Vision, Graphics and Image Processing 51*, 313–337. 96

Harris, C. and M. Stephens (1988). A combined corner and edge detector. In *Proc. of the 4th Alvey Vision Conference*, pp. 147–151. 64

Harris, Z. (1954). Distributional structure. *Word 10*(23), 146–162. 62

Hastings, W. (1970). Monte carlo samping methods using markov chains and their applications. *Biometrika 57*(1), 97–109. 82

Jahangiri, M. and M. Petrou (2009). An attention model for extracting regions that merit identification. In *ICIP*. 23, 25, 34, 197

Jampani, V., R. Gadde, and P. Gehler (2015). Efficient facade segmentation using auto-context. In Proc. of IEEE Winter Conf. on Applications of Computer Vision (WACV), pp. 1038–1045. 34, 38

Ji, Q. and R. M. Haralick (1999). A Statistically Efficient Method for Ellipse Detection. In *ICIP*, pp. 730–734. 96

Joachims, T. (1998). *Making large-scale support vector machine learning practical*, Chapter Advances in Kernel Methods - Support Vector Learning, pp. 169–184. Cambridge, MA, USA: MIT Press. 69

Jurie, F. and C. Schmid (2004). Scale-invariant shape features for recognition of object categories. In *CVPR*, Volume 2, pp. II–90 – II–96 Vol.2. 92

Kadir, T. and M. Brady (2001, November). Saliency, scale and image description. *IJCV 45*(2), 83–105. 64

Kiefer, J. (1953). Sequential minimax search for a maximum. *Proc. Amer. Math. Soc. 4*, 502–506. 55, 175

Kirkpatrick, S., C. D. Gelatt, and M. P. Vecchi (1983). Optimization by simulated annealing. *Science 220*(4598), 671–680. 87

Korc, F. and W. Förstner (2009). etrims image database for interpreting images of man-made scenes. Technical Report TR-IGG-P-2009-01, University of Bonn, Dept. of Photogrammetry. 146

Koziński, M., R. Gadde, S. Zagoruyko, G. Obozinski, and R. Marlet (2015). A mrf shape prior for facade parsing with occlusions. In *CVPR*, pp. 2820–2828. 23, 28, 37, 38, 197

Lacoste, C., X. Descombes, and J. Zerubia (2002, July). A Comparative Study of Point Processes for Line Network Extraction in Remote Sensing. Technical Report RR-4516, INRI. 143

Lacoste, C., X. Descombes, and J. Zerubia (2005). Point processes for unsupervised line network extraction in remote sensing. *TPAMI 27*(10), 1568–1579. 144

Lafarge, F., X. Descombes, J. Zerubia, and M. Pierrot-Deseilligny (2010, Jan). Structural approach for building reconstruction from a single dsm. *TPAMI 32*(1), 135–147. 144

Lafarge, F., G. Gimel'farb, and X. Descombes (2010). Geometric feature extraction by a multimarked point process. *TPAMI 32*(9), 1597–1609. 43, 144

Lafarge, F. and G. L. Gimel'farb (2008). Texture representation by geometric objects using a jump-diffusion process. In *BMVC*. 43, 144

Lazebnik, S., C. Schmid, and J. Ponce (2003). A sparse texture representation using affine-invariant regions. In *CVPR*. 62

LeCun, Y., Y. Bengio, and G. Hinton (2015). Deep learning. *Nature 521*. 159

Lee, S. C. and R. Nevatia (2004). Extraction and integration of window in a 3d building model from ground view images. In *CVPR*, Volume II, pp. 113–120. 20, 22, 23, 24, 29, 31, 33, 38, 197

Leibe, B., A. Leonardis, and B. Schiele (2004). Combined object categorization and segmentation with an implicit shape model. In *Proceedings of the Workshop on Statistical Learning in Computer Vision*. 43, 64, 116, 117

Leordeanu, M., M. Hebert, and R. Sukthankar (2007). Beyond local appearance: Category recognition from pairwise interactions of simple features. In *CVPR*. 116, 117

Leung, T. and J. Malik (2001, June). Representing and recognizing the visual appearance of materials using three-dimensional textons. *IJCV 43*(1), 29–44. 62

Libuda, L., I. Grothues, and K.-F. Kraiss (2006). Ellipse detection in digital image data using geometric features. In *VISAPP*, pp. 175–180. 96

Lowe, D. G. (2004). Distinctive Image Features from Scale-Invariant Keypoints. *IJCV 60*(2), 91–110. 64

Lu, C., N. Adluru, H. Ling, g. Zhu, and L. Latecki (2010). Contour based object detection using part bundles. *CVIU 114*(7), 827–834. 117

Manning, C. D., P. Raghavan, and H. Schütze (2009). *An Introduction to Information Retrieval*. Cambridge University Press. 66

Martin, D. R., C. C. Fowlkes, and J. Malik (2004, May). Learning to Detect Natural Image Boundaries Using Local Brightness, Color, and Texture Cues. *TPAMI 26*(5), 530–549. 97

Martinović, A., M. Mathias, J. Weissenberg, and L. V. Gool (2012). A three-layered approach to facade parsing. In *ECCV*, Volume 7578, pp. 416–429. 23, 28, 37, 38, 197

Mayer, H. and S. Reznik (2005). Building facade interpretation from image sequences. In *In Int. Ann. Photogramm. Remote Sens. (ISPRS'05)*, Volume 36, pp. 55–60. 23, 31, 35, 38

McGlone, J. C. (2004). *Manual of Photogrammetry* (5th Edition ed.). American Society for Photogrammetry and Remote Sensing. 61, 177

Meidow, J., C. Beder, and W. Förstner (2009). Reasoning with uncertain points, straight lines, and straight line segments in 2d. *P&RS 64*(2), 125–139. 47, 163

Metropolis, N., A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller, and E. Teller (1953). Equation of state calculations by fast computing machines. *The Journal of Chemical Physics 21*(6), 1087–1092. 82

Mikolajczyk, K. and C. Schmid (2005). A Performance Evaluation of Local Descriptors. *TPAMI 27*(10), 1615–1630. 64

Mingas, G. and C.-S. Bouganis (2012). Parallel tempering mcmc acceleration using reconfigurable hardware. In *Reconfigurable Computing: Architectures, Tools and Applications*, Volume 7199 of *Lecture Notes in Computer Science*, pp. 227–238. Springer. 160

Moore, A., C. Mason, P. A. Whigham, and M. Thompson-Fawcett (2003). The use of the circle tree for the efficient storage of polygons. In *Proc. of GeoComputation*. 96

Müller, P., G. Zeng, P. Wonka, and L. Van Gool (2007). Image-based Procedural Modeling of Facades. *ACM Trans. Graph. 26*(3). 20, 31, 33, 36, 38, 197

Musialski, P., P. Wonka, D. G. Aliaga, M. Wimmer, L. van Gool, and W. Purgathofer (2012). A survey of urban reconstruction. In *EUROGRAPHICS 2012 State of the Art Reports*, pp. 1–28. 20

Musialski, P., P. Wonka, M. Recheis, S. Maierhofer, and W. Purgathofer (2009). Symmetry-based facade repair. In *Vision, Modeling, and Visualization Workshop (VMV09)*, pp. 3–10. 29, 30, 34, 35, 197

Nguyen, H.-G., R. Fablet, and J.-M. Boucher (2010). Spatial statistics of visual keypoints for texture recognition. In K. Daniilidis, P. Maragos, and N. Paragios (Eds.), *Computer Vision ECCV 2010*, Volume 6314 of *Lecture Notes in Computer Science*, pp. 764–777. Springer Berlin Heidelberg. 144

Nguyen, T. P. and B. Kerautret (2011). Ellipse detection through decomposition of circular arcs and line segments. In *Proc. of ICIAP*, pp. 554–564. 96

Niemeier, W. (2008). *Ausgleichungsrechnung* (2., überarbeitete und erweiterte Auflage ed.). De Gruyter. 177

Nister, D. and H. Stewenius (2006). Scalable recognition with a vocabulary tree. In *CVPR*, Volume 2, pp. 2161–2168. 64

Ommer, B. and J. Malik (2009). Multi-scale object detection by clustering lines. In *ICCV*, pp. 484–491. 117

Opelt, A. and A. Zisserman (2006). A boundary-fragment-model for object detection. In *ECCV*, pp. 575–588. 64, 117

Ortner, M., X. Descombes, and J. Josiane Zerubia (2008). A marked point process of rectangles and segments for automatic analysis of digital elevation models. *TPAMI 30*(1). 43, 143

Ortner, M., X. Descombes, and J. Zerubia (2003). Improved rjmcmc point process sampler for object detection by simulated annealing. Technical Report 4900, INRIA. 86, 87, 143, 180, 181

Ortner, M., X. Descombes, and J. Zerubia (2007). Building outline extraction from digital elevation models using marked point processes. *IJCV 72*(2), 107–132. 43, 143

Park, M., K. Brocklehurst, R. Collins, and Y. Liu (2010). Translation-symmetry-based perceptual grouping with applications to urban scenes. In *ACCV*, Volume 3, pp. 1631–1645. 29, 31, 34, 35, 197

Patraucean, V., P. Gurdjos, and R. G. von Gioi (2012). A parameterless line segment and elliptical arc detector with enhanced ellipse fitting. In *ECCV*, Volume 7573, pp. 572–585. 96, 97, 102, 103

Pavlidis, T. (1983, January). Curve fitting with conic splines. *ACM Trans. Graph. 2*(1), 1–31. 96

Platt, J. C. (2000). Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. *Advances in Large Margin Classifiers*, 61–74. 69

Porrill, J. (1990). Fitting ellipses and predicting confidence envelopes using a bias corrected kalman filter. *Image and Vision Computing 8*(1), 37 – 41. 96

Pratt, V. (1987). Direct least-squares fitting of algebraic surfaces. In *Proc. of the 14th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '87, pp. 145–152. 172

Press, W. H., S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery (2007). *Numerical Recipes 3rd Edition: The Art of Scientific Computing* (3 ed.). New York, NY, USA: Cambridge University Press. 55, 175, 176

Recky, M. and F. Leberl (2010). Windows detection using k-means in cie-lab color space. In *ICPR*, Volume 0, pp. 356–359. 23, 24, 33, 38, 197

Ren, S., K. He, R. B. Girshick, and J. Sun (2015). Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. In *NIPS*, Volume abs/1506.01497. 159

Reznik, S. and H. Mayer (2008). Implicit shape models, self-diagnosis, and model selection for 3d facade interpretation. *PFG 3*, 187–196. 23, 26, 31, 35, 197

Richardson, S. and P. J. Green (1997). On bayesian analysis of mixtures with an unknown number of components. *Journal of the Royal Statistical Society 59*, 731–792. 83, 143

Ripperda, N. (2008). Determination of facade attributes for facade reconstruction. In *In Int. Ann. Photogramm. Remote Sens. (ISPRS'08)*. 20, 31, 32, 36, 38, 83, 197

Rissanen, J. (1978). Modeling by shortest data description. *Automatica 14*(5), 465 – 471. 61

Rissanen, J. (1983). A universal prior for integers and estimation by minimum description length. *Annals of Statistics 11*(2), 416–431. 61

Roscher, R., W. Förstner, and B. Waske (2012). I^2VM: Incremental Import Vector Machines. *Image and Vision Computing 30*, 263–278. 70, 71

Rosin, P. L. (1993). A note on the least square fitting of ellipses. *Pattern Recognition Letters 14*, 799–808. 49

Rosin, P. L. and G. A. W. West (1995, December). Nonparametric Segmentation of Curves into Various Representations. *TPAMI 17*(12), 1140–1153. 94, 96

Russakovsky, O., J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei (2015). ImageNet Large Scale Visual Recognition Challenge. *IJCV 115*(3), 211–252. 159

Sali, E. and S. Ullman (1999). Combining class-specific fragments for object classification. In *BMVC*. 64

Schaffalitzky, F. and A. Zisserman (2000, June). Planar grouping for automatic detection of vanishing lines and points. *Image and Vision Computing 18*(9), 647–658. 33, 40

Schmittwilken, J. and L. Plümer (2010). Model-based reconstruction and classification of facade parts in 3d point clouds. In *PCV*. 31

Schölkopf, B. and A. J. Smola (2002). *Learning with kernels : support vector machines, regularization, optimization, and beyond*. Adaptive computation and machine learning. MIT Press. 69

Schwarz, G. (1978). Estimating the dimension of a model. *The Annals of Statistics 6*(2), 461–464. 61

Shotton, J., A. Blake, and R. Cipolla (2005). Contour-based learning for object detection. In *ICCV*, pp. I: 503–510. 64, 117

Shotton, J., A. Blake, and R. Cipolla (2007, July). Multi-scale categorical object recognition using contour fragments. *TPAMI 30*(7), 1270–1281. 117

Sivic, J., B. C. Russell, A. A. Efros, A. Zisserman, and W. T. Freeman (2005). Discovering object categories in image collections. In *ICCV*. 62, 67

Sivic, J. and A. Zisserman (2006). Video Google: Efficient visual search of videos. In J. Ponce, M. Hebert, C. Schmid, and A. Zisserman (Eds.), *Toward Category-Level Object Recognition*, Volume 4170 of *LNCS*, pp. 127–144. Springer. 62, 64

Soheilian, B. and M. Brédif (2014). Multi-view 3d circular target reconstruction with uncertainty analysis. *ISPRS Ann. Photogramm. Remote Sens. Spatial Inf. Sci II-3*, 143–148. 96

Spinello, L., R. Triebel, D. Vasquez, K. O. Arras, and R. Siegwart (2010). Exploiting repetitive object patterns for model compression and completion. In *ECCV*, Volume 6315, pp. 296–309. 29, 30, 34, 197

Srikantha, A. and J. Gall (2014). Hough-based object detection with grouped features. In *ICIP*, pp. 1653–1657. 117

Stephens, M. (2000, Feb). Bayesian analysis of mixture models with an unknown number of components – an alternative to reversible jump methods. *The Annals of Statistics 28*(1), 40–74. 143

Stoica, R., P. Gregori, and J. Mateu (2005). Simulated annealing and object point processes: Tools for analysis of spatial patterns. *Stochastic Processes and their Applications 115*(11), 1860 – 1882. 75

Sun, K., N. Sang, and T. Zhang (2007). Marked point process for vascular tree extraction on angiogram. In *Proc. of Int. Conf. on Energy Minimization Methods in Computer Vision and Pattern Recognition (EMMCVPR)*, pp. 467–478. 144

Taubin, G. (1991, November). Estimation of planar curves, surfaces, and nonplanar space curves defined by implicit equations with applications to edge and range image segmentation. *TPAMI 13*(11), 1115–1138. 171, 172

Teboul, O., I. Kokkinos, L. Simon, P. Koutsourakis, and N. Paragios (2011). Shape grammar parsing via reinforcement learning. In *CVPR*. 38

Teboul, O., I. Kokkinos, L. Simon, P. Koutsourakis, and N. Paragios (2013). Parsing facades with shape grammars and reinforcement learning. *TPAMI 35*(7), 1744–1756. 36, 38

Teboul, O., L. Simon, P. Koutsourakis, and N. Paragios (2010). Segmentation of building facades using procedural shape priors. In *CVPR*, pp. 3105–3112. 23, 27, 34, 36, 38, 156, 158, 197

Teeravech, K., M. Nagai, K. Honda, and M. Dailey (2014). Discovering repetitive patterns in facade images using a ransac-style algorithm. *P&RS 92*(0), 38 – 53. 29, 33, 34

Tipping, M. E. (2001, Jun). Sparse bayesian learning and the relevance vector machine. *JMLR 1*, 211–244. 69

Tournaire, O., M. Bredif, D. Boldo, and M. Durupt (2010). An efficient stochastic approach for building footprint extraction from digital elevation models. *P&RS 65*, 317–327. 43, 144

Tournaire, O., N. Paparoditis, and F. Lafarge (2007). Rectangular road marking detection with marked point processes. In *PIA*, Volume 36, pp. 149–154. 43, 143

Tyleček, R. and R. Šára (2010). A weak structure model for regular pattern recognition applied to facade images. In *ACCV*, Volume 1, pp. 445–458. 23, 26, 35, 197

Utasi, Á. and C. Benedek (2011). A 3-D marked point process model for multi-view people detection. In *CVPR*, pp. 3385–3392. 144

Van Gool, L., G. Zeng, F. Van den Borre, and P. Müller (2007). Towards mass-produced building models. In *PIA*, Number 36(3/W49A), pp. 209–220. 31, 36

Vapnik, V. N. (1995). *The Nature of Statistical Learning Theory*. New York, NY, USA: Springer-Verlag New York, Inc. 69

Verdie, Y. and F. Lafarge (2013, August). Detecting parametric objects in large scenes by Monte Carlo sampling. *IJCV*. 43, 144, 159

Viola, P. and M. J. Jones (2004, May). Robust real-time face detection. *IJCV 57*(2), 137–154. 131

Wang, J., T. Fang, Q. Su, S. Zhu, J. Liu, S. Cai, C. Tai, and L. Quan (2015). Structure-driven facade parsing with irregular patterns. In *ACPR*. 38, 145, 160

Weidner, U. and W. Förstner (1995). Towards automatic building extraction from high-resolution digital elevation models. *P&RS 50*(4), 38–49. 22

Wendel, A., M. Donoser, and H. Bischof (2010). Unsupervised Facade Segmentation Using Repetitive Patterns. In *DAGM*, Volume 6376, pp. 51–60. 29, 34

Wenzel, S., M. Drauschke, and W. Förstner (2007). Detection and Description of Repeated Structures in Rectified Facade Images. *PFG 7*, 481–490. 21, 29, 30, 34, 197

Wenzel, S., M. Drauschke, and W. Förstner (2008, September). Detection of repeated structures in facade images. *Pattern Recognition and Image Analysis 18*(3), 406–411. 22, 23

Wenzel, S. and W. Förstner (2008). Semi-supervised incremental learning of hierarchical appearance models. In *Int. Ann. Photogramm. Remote Sens. (ISPR'08)*, pp. 399–404 Part B3b–2. 22, 24, 29, 34, 35, 197

Wenzel, S. and W. Förstner (2012). Learning a Compositional Representation for Facade Objects. In *Int. Ann. Photogramm. Remote Sens. (ISPR'12)*, Volume I-3, pp. 197–202. 21, 108

Wenzel, S. and W. Förstner (2013). Finding Poly-Curves of Straight Line and Ellipse Segments in Images. *PFG 4*, 297–308. 21

Wenzel, S. and W. Förstner (2016). Facade Interpretation Using a Marked Point Process. In *Int. Ann. Photogramm. Remote Sens. (ISPRS'16) (accepted)*. 21

Wenzel, S. and L. Hotz (2010). The Role of Sequences for Incremental Learning. In *Proc. of the Int. Conf. on Agents and Artificial Intelligence (ICAART 2010)*, Volume 1, pp. 434–439. 22

Werner, T. and A. Zisserman (2002). New techniques for automated architecture reconstruction from photographs. In *ECCV*, Volume 2, pp. 541–555. 22, 23, 31, 35, 197

West, G. A. W. and P. L. Rosin (1992). Multi-stage Combined Ellipse and Line Detection. In *BMVC*, pp. 197–206. 96

Wonka, P., M. Wimmer, F. Sillion, and W. Ribarsky (2003, July). Instant architecture. *ACM Trans. Graph. 22*(3), 669–677. Proceedings ACM SIGGRAPH 2003. 35

Wu, J. (2008). Robust Real-Time Ellipse Detection by Direct Least-Square-Fitting. In *Proc. of Int. Conf. on Computer Science and Software Engineering*, Volume 1, pp. 923–927. 96

Yang, M. Y. and W. Förstner (2011). A hierarchical conditional random field model for labeling and classifying images of man-made scenes. In *IEEE/ISPRS Workshop on Computer Vision for Remote Sensing of the Environment (ICCV)*. 37, 38

Yu, X., L. Yi, C. Fermuller, and D. Doermann (2007). Object detection using a shape codebook. In *BMVC*, pp. 1–10. 117

Zhao, P., T. Fang, J. Xiao, H. Zhang, Q. Zhao, and L. Quan (2010). Rectilinear parsing of architecture in urban environment. In *CVPR*, pp. 342–349. 31, 34

Zhu, J. and T. Hastie (2005). Kernel Logistic Regression and the Import Vector Machine. *J. Comput. Graph. Stat. 14*(1), 185–205. 69