CHRISTIAN-ALBRECHTS-UNIVERSITY KIEL

DISSERTATION

Automatic Multi-Scale and Multi-Object Pedestrian and Car Detection in Digital Images Based on the Discriminative Generalized Hough Transform and Deep Convolutional Neural Networks

---

Author:
Eric GABRIEL

Supervisors:
Prof. Dr. Carsten MEYER
Prof. Dr.-Ing. Reinhard KOCH
Prof. Dr. Hauke SCHRAMM

A thesis submitted in fulfillment of the requirements
for the degree of Doctor of Engineering (Dr.-Ing.)

in the

Research Group Pattern Recognition and Machine Learning
at Kiel University of Applied Sciences, and the
Multimedia Information Processing Group
at the Technical Faculty of the Christian-Albrechts-University Kiel (CAU)

May, 2019

1. Gutachter: Prof. Dr. rer. nat. Carsten Meyer
Institut für Informatik
Christian-Albrechts-Universität zu Kiel

2. Gutachter: Prof. Dr.-Ing. Reinhard Koch
Institut für Informatik
Christian-Albrechts-Universität zu Kiel

Datum der mündlichen Prüfung: 12. März 2019

"Wanderer, your footsteps are the road, and nothing more;
wanderer, there is no road, the road is made by walking.
By walking one makes the road, and upon glancing behind
one sees the path that never will be trod again.
Wanderer, there is no road—Only wakes upon the sea."

Antonio Machado, Campos de Castilla

Christian-Albrechts-University Kiel

Abstract

Technical Faculty of the Christian-Albrechts-University Kiel (CAU)

Doctor of Engineering (Dr.-Ing.)

Automatic Multi-Scale and Multi-Object Pedestrian and Car Detection in Digital Images Based on the Discriminative Generalized Hough Transform and Deep Convolutional Neural Networks

by Eric GABRIEL

Many approaches have been suggested for automatic pedestrian and car detection to cope with the large variability regarding object size, occlusion, background variability, aspect and so forth. Current state-of-the-art deep learning-based frameworks rely either on a proposal generation mechanism (e.g., "Faster R-CNN") or on the inspection of image quadrants / octants (e.g., "YOLO" or "SSD"), which are then further processed with deep convolutional neural networks (CNN).

In this thesis, the Discriminative Generalized Hough Transform (DGHT), which operates on edge images, is analyzed for the application to automatic multi-scale and multi-object pedestrian and car detection in 2D digital images. The analysis motivates to use the DGHT as an efficient proposal generation mechanism, followed by a proposal (bounding box) refinement and proposal acceptance or rejection based on a deep CNN. The impact of the different components of the resulting DGHT object detection pipeline as well as the amount of DGHT training data on the detection performance are analyzed in detail. Due to the low false negative rate and the low number of candidates of the DGHT as well as the high classification accuracy of the CNN, competitive performance to the state-of-the-art in pedestrian and car detection is obtained on the IAIR database with much less generated proposals than other proposal-generating algorithms, being outperformed only by YOLOv2 fine-tuned to IAIR cars. By evaluations on further databases (without retraining or adaptation) the generalization capability of the DGHT object detection pipeline is shown.

Christian-Albrechts-University Kiel

Abstract

Technical Faculty of the Christian-Albrechts-University Kiel (CAU)

Doctor of Engineering (Dr.-Ing.)

Automatic Multi-Scale and Multi-Object Pedestrian and Car Detection in Digital Images Based on the Discriminative Generalized Hough Transform and Deep Convolutional Neural Networks

by Eric GABRIEL

Die automatische Detektion von Personen und Fahrzeugen ist seit jeher eine wichtige und sicherheitsrelevante Aufgabe im Bereich der Bildverarbeitung. In den letzten Jahrzehnten sind hierzu diverse Ansätze vorgeschlagen worden, die sich mit der Handhabung der Objektvariabilität hinsichtlich der Größe und der Perspektive sowie im Allgemeinen mit der Hintergrundvariabilät sowie der Verdeckung von Objekten beschäftigen. Dennoch gilt das Problem als immer noch nicht gelöst. Der aktuelle Stand der Technik setzt sich hauptsächlich aus ein- und zweistufigen "Deep Learning"-Ansätzen zusammen: Erstere (z.B. "YOLO" und "SSD") arbeiten direkt auf dem normierten und anschließend in ein Raster eingeteilten Eingangsbild und analysieren die resultierenden Zellen mithilfe von Convolutional Neural Networks (CNN; faltendes neuronales Netz). Letztere (z.B. "Faster R-CNN") basieren auf einer initialen Generierung von Kandidatenfenstern auf dem Eingangsbild, die in einem zweiten Schritt ebenfalls durch ein CNN nachverarbeitet, d.h., in ihrer Größe angepasst und klassifiziert, werden.

In der vorliegenden Arbeit wird die Diskriminative Generalisierte Hough-Transformation (DGHT), ein allgemeiner, formmodellbasierter Ansatz zur Objektdetektion, der auf Kantenbildern arbeitet, hinsichtlich ihrer Eignung zur automatischen Personen- und Fahrzeugerkennung in digitalen 2D-Bilddaten analysiert. Hieraus geht ein zweistufiger Ansatz hervor, in dem die DGHT als effizienter Ansatz zur Generierung von Kandidatenfenstern eingesetzt wird, welche dann mittels CNN-Regression in ihrer Größe und Position angepaßt und über eine CNN-Klassifikation zurückgewiesen oder akzeptiert werden. Alle beteiligten Komponenten sowie die Menge der für die DGHT verwendeten Trainingsdaten werden hinsichtlich ihres Einflusses auf die Detektionsgenauigkeit untersucht. Die allgemein sehr niedrige Falsch-Negativ-Rate und die geringe Anzahl der DGHT-Kandidaten sowie die hohe Genauigkeit der CNNs bzgl. der Regression und Klassifikation führen zu einer Detektionsrate auf der IAIR-Datenbank für Personen- und Fahrzeugerkennung, die viele Vergleichsverfahren übertrifft und mit dem aktuellen Stand der Technik vergleichbar ist. Durch Übertragungen der Detektionspipeline auf weitere Datenbanken zur Personen- und Fahrzeugerkennung konnte – ohne jegliche Anpassungen – die Fähigkeit des Ansatzes zur Generalisierung gezeigt werden.

Acknowledgements

First of all, I am very grateful to the Ministry of Education, Science and Cultural Affairs of Schleswig–Holstein, Germany, for funding the research that has been conducted in my Ph.D. studies for the most part.

Moreover, I would like to express my sincere gratitude to my advisors Prof. Dr. Carsten Meyer and Prof. Dr. Hauke Schramm for the continuous support during my Ph.D. studies. Their motivation, patience, experience and guidance, as well as the countless insipring discussions, helped me to cope with all the difficulties throughout the course of my Ph.D. studies.

Besides my advisors, I would particularly like to thank Prof. Dr.-Ing. Reinhard Koch for the possibility to conduct the Ph.D. studies documented in this thesis in a collaboration between the Technical Faculty of the Christian-Albrechts-University (CAU) and Kiel University of Applied Sciences. Also, his encouragement and the opportunity to present and discuss my studies in his Multimedia Information Processing (MIP) group have been very insightful for my research.

I would also like to acknowledge my colleagues, Gordon Böer, Georg Ferdinand Hahmann, and Oliver Mader, for the innumerable fruitful discussions over cups of coffee, for sharing many experiences, and for their moral support.

Finally, I want to thank my parents, my sister Eileen, my wife Maren with my daughter Greta, and also my friends, for supporting me through the highs and lows within the last five years. Without you, I would not have been able to find the motivation and the fortitude to complete this dissertation.

Contents

List of Figures

List of Tables

List of Abbreviations

ABO	**A**verage **B**est **O**verlap
AP	**A**verage **P**recision
AUC	**A**rea **U**nder **C**urve
BRIEF	**B**inary **R**obust **I**ndependent **E**lementary **F**eatures
CE	**C**ross **E**ntropy
CNN	**C**onvolutional **N**eural **N**etwork
DET	**D**etection **E**rror **T**rade-off
DGHT	**D**iscriminative **G**eneralized **H**ough **T**ransform
DPM	**D**iscriminative **M**odel **C**ombination
DPM	**D**erformable **P**art **M**odels
DoG	**D**ifference **o**f **G**aussian
EER	**E**qual **E**rror **R**ate
FAST	**F**eatures from **A**ccelerated **S**egment **T**est
FL	**F**ocal **L**oss
FLOP	**F**loating **P**oint **OP**eration
FN	**F**alse **N**egative
FP	**F**alse **P**ositive
FPDW	**F**astest **P**edestrian **D**etector in the **W**est
FPN	**F**eature **P**yramid **N**etwork
FPPI	**F**alse **P**ositives **P**er **I**mage
GHT	**G**eneralized **H**ough **T**ransform
GSE	**G**eneral **S**tructured **E**dges
HOG	**H**istograms of **O**riented **G**radients
HT	**H**ough **T**ransform
ICF	**I**ntegral **C**hannel **F**eatures
IoU	**I**ntersection **o**ver **U**nion
ISM	**I**mplicit **S**hape **M**odels
LoG	**L**aplacian **o**f **G**aussian
mAP	**m**ean **A**verage **P**recision
MLP	**M**ulti **L**ayer **P**erceptron
NMS	**N**on-**M**aximum **S**uppression
PCA	**P**rinciple **C**omponent **A**nalysis
ReLU	**Re**ctified **L**inear **U**nit
ROI	**R**egion **O**f **I**nterest
RP	**R**egion **P**roposals
RPN	**R**egion **P**roposal **N**etwork
SCM	**S**hape **C**onsistency **M**easure
SED	**S**tructured **E**dge **D**etection
SGD	**S**tochastic **G**radient **D**escent
SIFT	**S**cale-**I**nvariant **F**eature **T**ransform
SSD	**S**um of **S**quared **D**ifferences or **S**ingle **S**hot Multibox **D**etector

SSE	Class-**s**pecific **S**tructured **E**dges
SURF	**S**peeded-**U**p **R**obust **F**eatures
SVM	**S**upport **V**ector **M**achine
TN	**T**rue **N**egative
TP	**T**rue **P**ositive
VGG	**V**isual **G**eometry **G**roup (University of Oxford)
VJ	**V**iola-**J**ones Detector
YOLO	**Y**ou **O**nly **L**ook **O**nce

List of Symbols

$\mathcal{M}$	(D)GHT shape model	
$\mathbf{m}_j$	model point j	
$\mathbf{x}_j$	local coordinates of the model point j	
γ_j	direction of the model point j	
$\mathbf{I}$	Input image	$\Omega \rightarrow \mathbb{R}$
$\mathbf{I}_E$	Edge image j	
$\mathbf{e}$	coordinates of an edge pixel	
$\gamma(\mathbf{e})$	direction of the edge pixel $\mathbf{e}$	
$\mathbf{H}^{\mathcal{M}}(\mathbf{c}, \mathbf{I}_E)$	Hough space consisting of Hough cells $\mathbf{c}$ created from input image $\mathbf{I}_E$ using the shape model $\mathcal{M}$	
$\mathbf{c_i}$	Hough cell i	
ρ	Quantization factor	
σ_r	Class "regular shape" in the SCM	
σ_i	Class "irregular shape" in the SCM	
$\hat{\mathbf{c}}$	Highest-scoring localization hypothesis / object candidate	
P	Predicted detection, i.e., a bounding box	
D	Set of predicted bounding box transformations	
$\hat{P}$	Refined prediction	
G	Ground truth bounding box	
$\mathcal{GT}$	Set of ground truth annotations	
$\mathcal{T}$	Set of test images	
C	(Ordered) list of candidates	
θ	Rejection threshold	

To Greta

Chapter 1

Introduction

1.1 Motivation

Pedestrian and car detection gained a lot of attention in the past decades and yet remain important and challenging tasks in computer vision [140, 29, 12, 102, 138, 99, 100, 90, 134, 130, 31] and, in particular, in safety-relevant automotive applications. Several evaluations of the state-of-the-art [29, 12, 138] have shown the great progress that has been made, however, the task is still far from being solved in terms of reliable and accurate detections that are computed in real-time.

Traditionally, feature- or model-based techniques have been employed. The first widely used real-time detection framework using Haar-like features was proposed by Viola and Jones [123] and also adapted to pedestrian detection as in [124]. Inter alia, the subsequent major breakthroughs in pedestrian detection were the invention of histograms of oriented gradients (HOG) as input features to a support vector machine (SVM) as proposed by Dalal and Triggs [23] as well as deformable part models (DPM) of Felzenszwalb et al. [37]. One of the best-performing, recent rigid object detectors, called `Roerei`, was proposed by Benenson et al. [11]. There also exist Random Forest-based approaches such as Marin et al. [91].

Since the success of deep convolutional neural networks (CNN) in image classification tasks [71, 117, 136], such networks have also been applied to object detection in general [86, 99, 102] and pedestrian (e.g., Angelova et al. [6] or Costea et al. [21]) and car detection in particular (e.g., Ren et al. [101]). First attempts involved a proposal generation mechanism [49, 102], where in the first step regions of interest have been extracted. Apart from the standard "sliding window" paradigm (e.g., used in [123]) that typically requires the classification of $10^4 - 10^5$ windows for accurate single-scale object detection, several algorithms exist to generate so-called region proposals [65]. They can be divided into grouping and window scoring approaches. One of the most prominent grouping approaches is SelectiveSearch proposed by Uijlings et al. [122] that is based on grouping hierarchical low-level segmentations. Objectness as proposed by Alexe et al. [4, 3] or EdgeBoxes [142] are examples for window scoring approaches that compute an output score for each sampled window how likely this window contains an object. These approaches are usually class-agnostic. Recent approaches aim at integrating the proposal generation step into the deep network [102] enabling end-to-end training. Lenc and Vedaldi [77] even show that a separate proposal generation step is not necessary and a plain CNN can accomplish the complete object detection task. However, it is known that CNNs usually need significant amounts of training data, which have to cover the expected object variability. Thus, one can argue that proposal generation may still be a useful component either to

handle a larger range of object variability or to increase efficiency or both. Besides, (groups of) small objects still remain a problem for CNNs [77, 99], which might be overcome with accurate region proposals.

To summarize, the main challenges of the state-of-the-art in object detection still involve the accurate detection of small instances, i.e., an object height smaller than 80px [77, 99, 139]. Another problem are region proposals that contain more than one object [65, 99]. Moreover, as the output of neural networks is of non-deterministic nature, the results could either intentionally be frauded [120] or show strange effects [103], e.g., depending on the object's position, a certain constellation etc., when relying on neural networks alone.

Thus in this thesis, an alternative object detection approach is developed. It is based on the Discriminative Generalized Hough Transform (DGHT), a general object detection algorithm that uses a voting approach and a shape model to detect objects of arbitrary shape in edge images. The DGHT framework is extended to handle multiple objects (of a given category) at multiple scales, even in the presence of large background variability. However, to arrive at an accurate object detection performance, the DGHT proposals are accepted or rejected by a subsequent deep convolutional neural network (CNN).

Overall, it is demonstrated that the advantage of the DGHT-based proposal generation is the relatively low number of misses (false negatives) at a moderate number of generated candidates (a few tens to hundreds per 2D image) along with the efficiency of a Hough-based approach.

On the IAIR database, which exhibits many sources of variability (background, object size, pose, aspect, etc.), competitive performance to state-of-the-art approaches as of February 2018 is achieved on the tasks of pedestrian and car detection, being outperformed only by YOLOv2 fine-tuned to IAIR Cars. Additional evaluations on further databases – performed without tuning any component to the new database – demonstrate the generalization capability of the system.

1.2 DGHT Object Localization Framework

The Discriminative Generalized Hough Transform (DGHT) is a Hough-based object localization approach that operates on edge images[1]. It uses a voting approach and a shape model describing the target object to transform an input edge image into a parameter space, the so-called Hough space. The dimensions of the resulting Hough space reflect the transformation parameters that are applied to the shape model. In order to reduce the computational complexity of the approach, these transformation parameters are often restricted to translations only. This leads to a quantized 2D Hough space where the number of votes ("counts") in the individual cells reflect the degree of matching between the respectively transformed shape model and the input edge image. In single-object localization, the DGHT outputs the cell accumulating the largest number of votes ("peak" in Hough space) as the most probable object location.

[1] In this thesis, only edge images are used, but generally, several types of input features are possible.

It has been shown that the DGHT can be successfully applied in tasks with limited variability, for instance, in the medical domain for the localization[2] of epiphyses [55], subsequent bone age assessment [54] or the localization of knee joints [109]. Moreover, also constrained real-world tasks with limited variability could successfully be solved, e.g., state-of-the-art iris localization at the time of publication [56].

Anticipatively, initial analyses revealed that the baseline system in its current form is not suitable for automatic object detection in real-world scenarios, i.e., rather unconstrained tasks where there is less knowledge about the object(s) in the testing images such as general pedestrian detection.

First, when transferring the application of the DGHT from the medical domain, where most images[3] exhibit a dark background not containing any confusable structures, to real-world images, it was noticeable that many peaks in the Hough space occur at confusable structures in the background.

Second, in tasks like pedestrian detection, there is no such information about the objects of interest as uniform or known scales nor the number of occurrences as in medical images. The objects might exhibit large size variability that is only constrained by the image dimensions. Besides, the number of occurrences is generally unconstrained.

Furthermore, the current system by design only outputs the single best-scoring localization hypothesis (peak in Hough space), even if the image contains multiple or no objects at all.

To summarize, this thesis concentrates on the following major problems of the DGHT framework as proposed by [107, 56, 55] (see Chapter 4):

- Background variability
- Object size variability
- Detection of no or multiple instances

These problems will in detail be addressed in Chapter 5.

1.3 Contributions

The main contribution of this thesis is the development and thorough evaluation of an automatic object detection framework based on the DGHT and deep convolutional neural networks (CNN) for pedestrian and car detection in real-world scenarios. In particular, the DGHT is used as a proposal generator for the deep CNN, and can handle background and object size variability as well as multiple objects, and the deep CNN refines and potentially rejects the DGHT proposals and finally outputs the detected object bounding boxes. Please see also Sect. 1.4 for a list of publications of the author that are related to aspects of the research.

In more detail, the contributions can be summarized as follows:

- Analysis of the baseline DGHT performance in a real-world pedestrian and car detection task with limited object size variability

[2] For an exact disambiguation, e.g., of the terms "localization" and "detection", the reader is referred to Sect. 1.6

[3] Mostly, hand or leg radiographs, i.e., X-ray scans, were used.

- Evaluation of different edge detection algorithms and parameterizations on the DGHT object detection performance
- Extension of the DGHT object detection framework towards objects with large object size variability via the traditional image scaling approach and a new, efficient model scaling approach on a pedestrian and car detection task
- Extension of the DGHT object detection framework towards detection of multiple objects via generating and assessing multiple candidates extracted from the Hough space on a pedestrian and car detection task[4]
- Development and evaluation of a new rejection scheme of DGHT object proposals based on a deep convolutional neural network (CNN), apart from a rejection based on the Shape Consistency Measure (SCM)
- Development and evaluation of a bounding box regression step for refining the DGHT proposals prior to the potential rejection by the CNN, further improving performance on both pedestrian and car detection
- Detailed analysis of the impact on detection performance of the different components in the current detection pipeline, and of the influence of the amount of (DGHT) training data (analyzed for pedestrians)
- Comparison of the DGHT-based region proposals to a fixed region proposal scheme similar to Lenc and Vedaldi [77]
- Detailed comparison to the state-of-the-art on pedestrian and car detection databases (see Chapter 3) including detection error trade-off (DET) curves and a detailed error analysis of the remaining detection errors, revealing suggestions for further improvements
- Analysis of the parallelization potential of the DGHT regarding runtime and memory consumption

1.4 Publications

As introduced in Sect. 1.3, several aspects of the research described in this thesis have already been published. The following list provides an overview on the publications and their contents:

- **A Shape Consistency Measure for Improving the Generalized Hough Transform** [56] by *Ferdinand Hahmann, Gordon Böer, Eric Gabriel, Carsten Meyer* and *Hauke Schramm*. This conference paper was presented at the *International Conference on Computer Vision Theory and Applications (VISAPP)* in 2015 and included an approach to assess the model point voting pattern in Hough space ("Shape Consistency Measure", SCM, see Sect. 4.3) in order to improve iris localization based on the DGHT.

[4] For the localization of multiple vertebrae in a medical context, Ruppertshofen et al. already reported the following post-processing strategy using *a priori* knowledge [107, 108]: The first vertebra is given by the highest-scoring peak in the Hough space. Afterwards, further peaks are identified by using a tubular region of interest that covers the spine and a minimum distance of the known vertebra size. In this thesis, a general multi-scale candidate generation approach for real-world scenarios is employed and evaluated.

- **Classification of Voting Patterns to Improve the Generalized Hough Transform for Epiphyses Localization** [55] by *Ferdinand Hahmann, Gordon Böer, Eric Gabriel, Thomas M. Deserno, Carsten Meyer* and *Hauke Schramm*. This conference paper was presented at the *SPIE Conference on Medical Imaging* in 2016 and included an evaluation of the SCM (see Sect. 4.3) for improved epiphyses localization in hand radiographs.
- **Structured Edge Detection for Improved Object Localization Using the Discriminative Generalized Hough Transform** [41] by *Eric Gabriel, Ferdinand Hahmann, Gordon Böer, Hauke Schramm,* and *Carsten Meyer*. This conference paper was presented at the *International Conference on Computer Vision Theory and Applications (VISAPP)* in 2016 and included an analysis of the impact of reducing background variability by applying different edge detection algorithms to the DGHT framework; the corresponding algorithms, experiments and findings are reported in Sect. 5.1.
- **Analysis of the Discriminative Generalized Hough Transform for Pedestrian Detection** [43] by *Eric Gabriel, Hauke Schramm,* and *Carsten Meyer*. This conference paper was presented at the *International Conference on Image Analysis and Processing (ICIAP)* in 2017 and included an evaluation of handling object size variability and detecting multiple objects using the DGHT framework for pedestrian detection; the corresponding system extensions, experiments and findings are reported in Sects. 5.2 and 5.3.
- **The Discriminative Generalized Hough Transform as a Proposal Generator for a Deep Network in Automatic Pedestrian Localization** [44] by *Eric Gabriel, Hauke Schramm,* and *Carsten Meyer*. This conference paper was presented at the *International Conference on Computer Vision Theory and Applications (VISAPP)* in 2018 and included a new model scaling scheme for handling size variability and detailed component analysis of the DGHT object detection pipeline; the corresponding concept is explained in Sect. 5.2.2 and the experiments in Sect. 5.2.3).
- **Analysis of the Discriminative Generalized Hough Transform as a Proposal Generator for a Deep Network in Automatic Pedestrian and Car Detection** [42] by *Eric Gabriel, Michael Schleiss, Hauke Schramm,* and *Carsten Meyer*. This publication was published in the *SPIE Journal of Electronic Imaging* in 2018 and contained a thorough analysis of the DGHT object detection pipeline with added bounding box regression, the impact of different components and the amount of training data. This journal publication merely summarizes the main algorithmic contributions, experiments and findings of the dissertation (reported in Chapters 4 and 5). Further, it includes a high-level comparison with YOLOv2 (reported in Sect. 6.2), a comparison of different CNNs for proposal rejection (reported in Sect. 6.3) and a computational and runtime analysis (reported in Sect. 6.4).

1.5 Structure

The remainder of this thesis is organized as follows:

Chapter 2 describes the foundations as well as the state-of-the-art of feature extraction, traditional (feature-based) and deep learning object detection approaches. Each

topic is discussed in general as well as focusing on pedestrian or car detection. Chapter 3 lists and describes all publicly available data sets that are used for training and evaluation in this thesis. Also, example images of the respective databases are presented.

Chapter 4 describes the baseline system available at the beginning of the thesis. This system has already been successfully applied to tasks with limited object variability before this thesis and is extended to the automatic detection of pedestrians and cars in real-world images in the present work. The necessary concepts and algorithmic extensions to solve these tasks are outlined in Chapter 5 along with the corresponding experimental setups and parameters.

The results of all experiments are presented and thoroughly analyzed in Sects. 5.1.4 and 5.6. A detailed discussion is conducted in Chapter 6 summarizing strengths and limitations of the proposed approach. Finally, Chapter 7 concludes the main findings of this thesis along with suggestions for further improvements and future work.

On a final note, the Appendix A provides information on often used basic concepts, namely integral images and convolutional neural networks (CNN).

1.6 Term Definitions

As the following terms are often interchanged, pooled or differently used in the context of image processing and object detection in particular, this section provides exact definitions of how the terms will be used in this thesis:

FIGURE 1.1: Illustration of the terms of object detection tasks

Classification Assignment of a class label and/or class probability to each input image.

Localization Assigning the most probable object location (reference point, e.g., the geometrical center) to an input image with a single object only.

Detection Assigning bounding boxes and class labels to an input image containing multiple objects and instances.

Segmentation Providing a class or a class and instance-based labeling on pixel level for an input image.

Fig. 1.1 shows an illustration of the different terms.

Chapter 2

Related Work

Foundations of Object Detection in Computer Vision

2.1 Feature Extraction

Generally, the first step for detecting objects in 2D RGB images is the extraction of features that represent structures and/or contents present in the image. The main goal of this preprocessing step is to get a lower dimensional representation, i.e., dimensionality reduction, of local or global image content, which is then used for object detection.

Characteristics of a feature and its quality are [59]:

- Separation capability, i.e., similar structures/objects produce a similar feature vector while different structures/objects do not
- Repeatability
- Computation time
- Robustness to scale/rotation/shifting/noise
- Compact representation of the feature vector, i.e., less memory consumption, no dependencies or redundancies

This section provides an overview of those features that are commonly used for object detection in 2D RGB images and focuses on those that are used in the scope of this thesis as well as comparison approaches.

2.1.1 Intensity Features

Intensity features can directly be extracted from the channels of the input image, i.e., these features are very easy to compute, if the desired channels are present. Often used channels or color spaces for the extraction of intensity features are:

- Gray values
- RGB (Red, Green, Blue) color model (see Fig. 2.1)
- HSV (Hue, Saturation, Value) color space
- Luv (Luminance and two chromaticity coordinates) color space (see Fig. 2.2)

FIGURE 2.1: RGB color model: Input image, red channel, green channel, blue channel (from left to right)

FIGURE 2.2: Luv color space: Input image, luminance channel, u channel, v channel (from left to right)

For instance, an advantage of the Luv color space is that the luminance can be decoupled from the chromaticity, i.e., the "color", which accounts for lighting invariance, and that color distances can be easily computed using the Euclidean distance as color differences are modeled equidistantly.

The disadvantages of intensity features are that (a) these are not invariant to transformations and (b), for instance, not robust to illumination changes. One could use histograms of intensities in given regions of interest (ROI) to overcome these limitations. Moreover, this type of feature is very dense, i.e., it contains lots of uncompressed information, and, thus, has to be used very carefully in order to maintain a compact feature representation and processing speed.

2.1.2 Edge Features

To achieve a more compact representation of a gray scale or color image, edge features can be used to represent interesting structures or parts of the image. In general, edge pixels are characterized by exhibiting discontinuities in image brightness.

Edge detection has been a very traditional task in image processing since more than 30 years. In the following subsections, only those edge detection approaches are described which are used in the context of this thesis, i.e., Canny edge detection (Sect. 2.1.2.1) and Structured Edge Detection (SED, Sect. 2.1.2.2). There exist many other approaches that are outlined and summarized, for instance, in related studies or surveys, e.g., [141, 89].

2.1.2.1 Canny Edge Detection

A well-known, general and robust approach for edge detection in digital images was introduced by Canny [18].

First, the input image $\mathbf{I} : \Omega \rightarrow \mathbb{R}$ is smoothed using a Gaussian filter. Subsequently, the values of the first derivatives in horizontal and vertical direction are obtained by applying the Sobel operator to the smoothed input image. Using these values, the

FIGURE 2.3: (Left) Input image. (Right) Canny edge image.

gradient magnitude and the edge direction can be calculated. The resulting edges are thinned using non-maximum suppression (NMS). Subsequently, the remaining edge pixels are classified using a high and a low threshold in the so-called hysteresis. Edges above the high threshold are kept, edges below the low threshold are discarded. Edges between the low and the high threshold are only kept if there is an edge pixel within the respective 8-connected neighborhood. This leads to a binary edge image $\mathbf{I}_E : \Omega \rightarrow \{0, 1\}$; for an example edge image see Fig. 2.3.

2.1.2.2 Structured Edge Detection (SED)

A more sophisticated, yet still real-time edge detection framework incorporating learning and the use of information of the objects of interest has been proposed by Dollár and Zitnick [30]. Therefore – and in contrast to Canny edge detection –, this approach requires a training procedure using an annotated training corpus.

Here, a Random Forest [15] maps patches of the input image $\mathbf{I}$ to output edge image patches using pixel-lookups and pairwise-difference features of 13 (3 color, 2 magnitude and 8 orientation) channels. While testing, densely sampled, overlapping image patches are fed into the trained detector. The edge patch outputs which refer to the same pixel are locally averaged. The resulting intensity value (which lies in the interval $[0, 1]$) can be seen as a confidence measure for the current pixel belonging to an edge. Subsequently, a NMS can be applied in order to sharpen the edges and reduce diffusion. For further details see Dollár and Zitnick [30]. An example of an edge image $\mathbf{I}_E : \Omega \rightarrow [0, 1]$ is shown in Fig. 2.4. As opposed to Fig. 2.3, only the edges belonging to the object of interest – here, pedestrians – as well as other strong vertical edges are remaining.

2.1.3 Haar-like Features

Haar-like features are based on the idea of Haar wavelets [53], i.e., a simple wavelet consisting of a combination of rectangular functions, and were developed by Viola and Jones [123] for object detection (see Sect. 2.2.2.1). Usually, Haar-like features are rectangular features that describe intensity differences of inner rectangular regions. Depending on the amount of inner rectangular regions, there exist 2-, 3- and 4-rectangle features (see Fig. 2.5). An example of their appearance is shown in Fig. 2.6. The concept of a Haar-like feature is to define expected directions of intensity

FIGURE 2.4: (Left) Input image. (Right) Structured edge image trained on pedestrian contours.

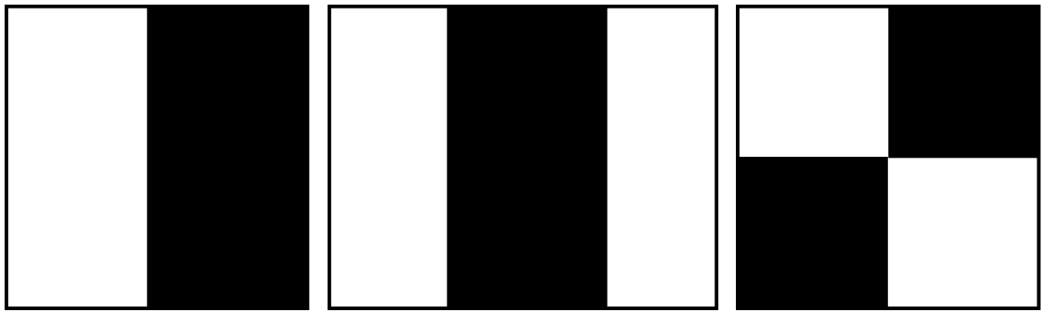


FIGURE 2.5: From left to right: Illustration of 2-, 3- and 4-rectangle Haar-like features

differences between the inner regions. To compute those features, the intensities in each inner region are summed up, and afterwards the difference of the sums of two adjacent regions is computed. The major advantage of these features is the small amount of time needed for computation. To achieve this, Viola and Jones propose to use integral images, i.e., summed-area tables as in [79] (see also Appendix A.1 for a detailed description). These allow for fast lookups of summed intensities for rectangular regions in constant time regardless of the feature size.

2.1.4 Histograms of Oriented Gradients (HOG)

Dalal and Triggs proposed to use feature descriptors that are computed on a dense grid of rectangular regions [23]. For each of these cells, a local histogram of gradient orientations is computed using a proper orientation binning, e.g., eight gradient directions. Afterwards, the histograms of a block, i.e., a region of 2×4 cells for instance, are contrast-normalized to account for illumination invariance. In order to achieve good representations and a good performance, the authors suggest to use rather fine-scale gradients and a fine orientation binning, a relatively coarse spatial binning as well as a high-quality local contrast normalization for overlapping descriptor blocks. The HOG representation is also related to the histogram concepts of the Scale-Invariant Feature Transform (SIFT, see Sect. 2.1.5.2).

FIGURE 2.6: (Top row) Top-2 Haar-like features for face detection selected by AdaBoost. (Bottom row) Typical training image with overlaid features. From [123]

Fig. 2.7 shows an example of the visualization of HOG feature descriptors. HOG feature descriptors were used in combination with a Support Vector Machine (SVM) for object detection in general and pedestrian detection in particular (see Sect. 2.2.2.2).

2.1.5 Interest Point Detection

An interest point, which is often used as an input feature in computer vision, can be described by the following characteristics [112, 85]:

- Clear (mathematical) definition
- Unique location in image space
- Significant local image structure – for instance, intensity contrasts or texture – surrounding the location of the interest point
- Repeatability, i.e., the interest point description is invariant to local or global changes such as illumination or noise

First, this subsection describes variants of corner detection algorithms. Corner detection can be seen as a subtask of interest point detection, which has been performed for many decades. Afterwards, different feature descriptors are presented, which are generally applied in traditional object detection.

2.1.5.1 Corner Detection

Corner detection can be seen as a subtask of interest point detection. Here, a corner is characterized by two intersecting edges or, more formally, a point for which two significant and different edge directions exist in the local neighborhood. Among many

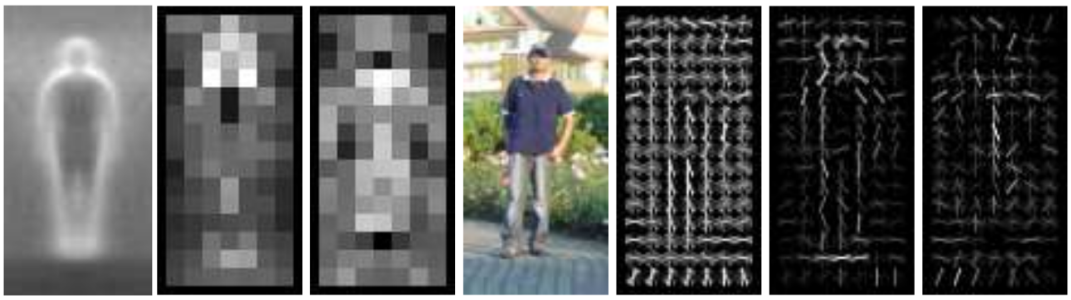


FIGURE 2.7: Illustration of HOG feature descriptors and SVM weights used for classification. (a) Average gradient image of all training images, where the pedestrian bounding boxes have been aligned, i.e., translated and scaled. (b) Positive SVM weights. (c) Negative SVM weights. (d) A sample test image. (e) HOG descriptor. (f) HOG descriptor weighted with positive SVM weights. (g) HOG descriptor weighted with negative SVM weights. From [23]

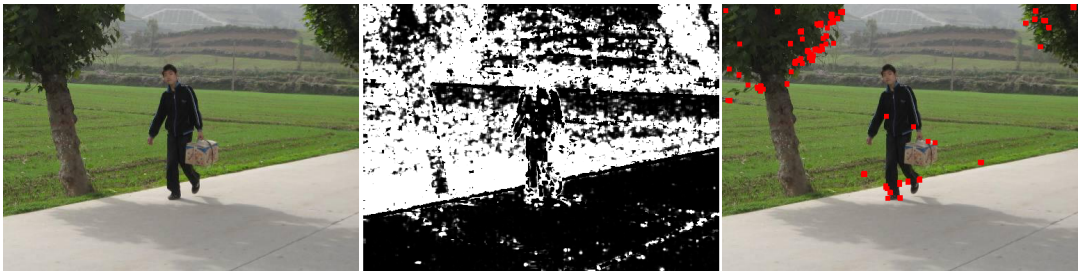


FIGURE 2.8: (Left) Input image. (Middle) Response of the function M (please see text). (Right) Resulting Harris corners after NMS.

other existing corner detection approaches, two popular techniques are described in this subsection, namely the Harris corner detector (Sect. 2.1.5.1.1) and Features from Accelerated Segment Test (FAST; Sect. 2.1.5.1.2).

2.1.5.1.1 Harris Corner Detector

Harris and Stephens [58] proposed to detect corners (and also edges) based on the so-called local auto-correlation. The weighted sum of squared differences (SSD)[1], denoted as S, between an image patch and a shifted version of it (referring to the concepts of auto-correlation) is used to detect corners and edges. Using a Taylor expansion, S can be approximated (written in matrix form):

$$S(x,y) \approx \begin{pmatrix} x & y \end{pmatrix} A \begin{pmatrix} x \\ y \end{pmatrix} \tag{2.1}$$

with A being a 2×2 structure tensor summarizing the predominant gradient directions in the specified neighborhood.

In this context, a corner is characterized by a large variation of S. This could be evaluated using an eigenvalue decomposition of A. If in 2D case, both eigenvalues

[1] If a Gaussian filter is used, the concept is very similar to Difference of Gaussians (DoG) as described in Sect. 2.1.5.2

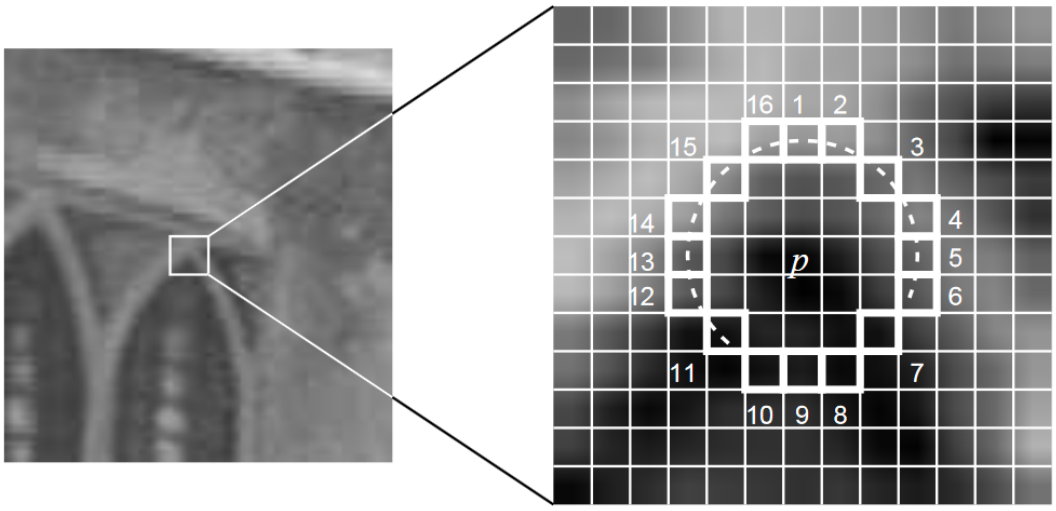


FIGURE 2.9: Illustration of the FAST corner detection approach. The set of 12 adjacent pixels brighter than the intensity of p plus a threshold t is indicated by the dashed line. From [105]

are approximately 0, the current pixel is no interest point. If both eigenvalues have large positive values, the current pixel belongs to a corner, if otherwise only one of the eigenvalues has a large positive value, the current pixel belongs to an edge.

To reduce computation time, the authors prove that it is sufficient to use a function M based only on the determinant and the trace of A instead of computing the complete eigenvalue decomposition.

As a visualization example, please see Fig. 2.8.

2.1.5.1.2 Features from Accelerated Segment Test (FAST)

In contrast to the Harris corner detector, Rosten and Drummond [105] proposed an approach that focuses on computational efficiency. Here, a circle of 16 pixels with the image pixel in consideration as the origin is used to classify if the current pixel is a corner. These pixels are clock-wisely indexed.

Based on these indices, a set of $N < 16$ adjacent pixels has to have a higher intensity than the intensity of the current pixel plus a threshold t or a lower intensity than the current pixel minus t in order to be classified as a corner. Please see Fig. 2.9 for an example illustration of this corner check.

Note that the approach directly uses intensities instead of having to compute image derivatives. Thus, it is still used for real-time video processing as well as the tracking and mapping of objects.

As an extension, a so-called high-speed test can be executed to faster exclude non-corner points. For $N \geq 4$, this is done by checking the intensities of the indices 1, 5, 9 and 13 (see Fig. 2.9) in order to rapidly assess whether a set of N adjacent pixels satisfying the intensity condition is generally possible .

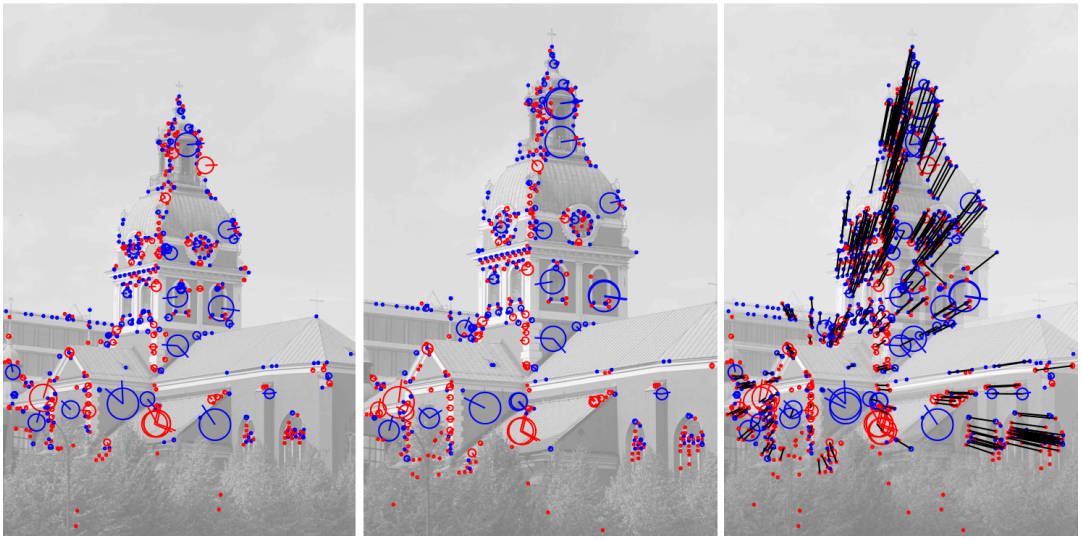


FIGURE 2.10: (Left and middle) Detected SIFT key-points with scale (radius of the circle) and orientation (arrow from center point) on two images of the same scene. Red circles indicate bright and blue circles represent dark feature descriptors. (Right) Key-point matches of both images are indicated by the black lines. From [84]

2.1.5.2 Scale-Invariant Feature Transform (SIFT)

These feature descriptors proposed by Lowe [87] were considered one of the major breakthroughs in the development of feature descriptors used for RGB images and feature matching. The key concepts are to use a pyramidal approach as well as assigning orientations to the detected key-points in order to achieve invariance to the scale and the rotation, respectively.

The scale filtering is done using Difference of Gaussians (DoG), which are approximations of the Laplacian of Gaussian (LoG) filter [85], in order to reduce computation time. A DoG is basically the difference when applying a Gaussian blurring twice to an image each with a different σ value[2]. To find corners of arbitrary size, a set of DoGs with different σ values, a so-called octave, is computed for a fixed input image dimension. In order to further increase scale invariance, DoG octaves are computed for an image pyramid, i.e., for different input image dimensions. Maxima can be extracted by a search over scale and space. This means that not only the neighbors of the pixel in consideration in the current scale but also the neighbors in adjacent scales are considered to find extrema. The locations are further refined using the Taylor expansion of the scale-space function.

As the second attribute to each key-point descriptor, the orientation of the maximum needs to be computed. In order to do so, the principal orientation is extracted using a voting approach on gradient directions considering a neighborhood around the current key-point. This principal orientation is used for normalizing a histogram of gradient directions that has been set up with respect to the respective neighborhood. The resulting orientation is then computed using the highest peak of the histogram and any other peak larger than 80% of the highest peak.

[2] σ denotes the standard deviation

To summarize, SIFT feature descriptors are very robust and accurate for feature matching. An example is shown in Fig. 2.10. The major drawback is their computational heaviness.

Exemplary for several extensions, Ke and Sukthankar [69] extended the standard SIFT feature descriptor computation by using the Principal Component Analysis (PCA) being more distinctive, more compact, more robust to image deformations and, thus, leading to increased accuracy and faster matching.

2.1.5.3 Speeded Up Robust Features (SURF)

Speeded Up Robust Features (SURF) are a patented robust feature detector and descriptor [9, 40]. In a sense, they can be seen as an extension of the SIFT feature descriptor (see Sect. 2.1.5.2) as they use the same steps (interest point detection, description using the local neighborhood and matching of the descriptors) but each with different approaches.

Instead of DoGs, the authors propose to perform an image convolution with square filters on the integral image (similar to Haar-like features (Sect. 2.1.3), see also Appendix A.1 for a detailed description) to reduce computation time. Simple box filters are used, which approximate Gaussian second order derivatives, for computing the Hessian matrix. The filter response is then given by the determinant of the approximated Hessian matrix:

$$\det(\mathcal{H}_{approx}) = D_{xx}D_{yy} - (0.9D_{xy})^2 \tag{2.2}$$

with D_{xx}, D_{yy} and D_{xy} being the approximations of the Gaussian derivatives.

Instead of detecting features over an image pyramid, here, different filter kernel sizes (starting with the lowest scale at 9×9 corresponding to $\sigma = 1.2$) are used to detect features over scales.

For the final interest point localization over all scales, a $3 \times 3 \times 3$ non-maximum suppression (NMS) on the filter responses is applied. Because the difference in scale between the layers in every octave is quite large, the maxima of the determinants need to be appropriately interpolated in image and scale space using the method of Brown et al. [16].

To compute the orientation for the resulting key-points, Haar-wavelet responses (see Sect. 2.1.3) of the horizontal and vertical direction are used taking into account a certain[3] circular neighborhood. Finally, the estimation of the principle orientation is done by taking the sum of a window around each key-point, again, using the wavelet responses in an integral image concept (see Fig. 2.11 for an illustration of SURF interest points, their scale and orientation, Haar wavelets and detection windows).

Feature matching is performed using the Euclidean distance of two feature descriptors. Additionally, only those feature descriptors are allowed to be matched if they exhibit the same sign of the Laplacian, i.e., the trace of $\mathcal{H}_{approx}$ (see Eq. 2.2). This sign is able to distinguish between bright blobs on dark backgrounds and dark blobs on

[3]The radius depends on the scale in which the corresponding key-point was detected

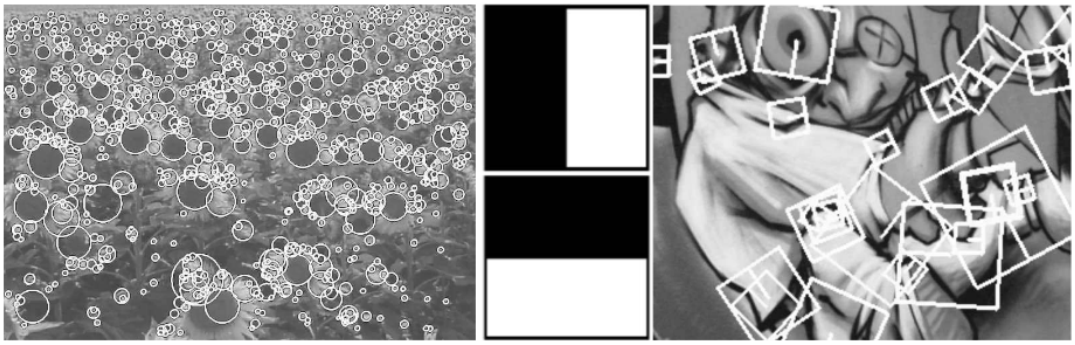


FIGURE 2.11: Left: Detected SURF interest points along with their scales (radius of the circles) for a sunflower field. Middle: Haar wavelet (see Sect. 2.1.3) examples used for SURF. Right: Detail of a graffiti scene showing descriptor windows and their orientation at different scales. From [9]

bright backgrounds. Moreover, this information was already computed while detecting key-points.

The authors state, that results using SURF features are comparable to those using SIFT features while being three times faster.

2.1.5.4 Binary Robust Independent Elementary Features (BRIEF)

Calonder et al. [17] propose to simply use binary strings as feature descriptors for interest points. The binary descriptor only relies on a number n of simple intensity difference tests to describe an image patch. As a prerequisite, the input image needs to be smoothed using a Gaussian filter. Afterwards a set of n comparisons between two points p_1 and p_2 is executed. In general, if the intensity of p_1 is higher, a 1 is appended to the binary string, otherwise a 0. The pattern of pairs (p_1, p_2) can either be fixed, randomly chosen or learned from data. Fig. 2.12 show five example patterns.

As for binary descriptors, the common distance measure called Hamming distance, i.e., the sum of the bitwise XOR operation between two descriptors, can be efficiently computed. The distance is then used for feature matching across frames.

Depending on the length of the binary string, BRIEF features have been shown to perform similarly to SURF feature descriptors (Sect. 2.1.5.3) at much higher speed on data sets where the variability is restricted to viewpoint changes, compression artifacts, illumination changes and blur. This approach is also designed to be directly executed on the input image. As a major drawback, it is not invariant to orientation and scale resulting in a lower performance on data sets that inhibit large variability regarding object rotation.

2.1.5.5 Oriented FAST and Rotated BRIEF (ORB)

Oriented FAST and Rotated BRIEF (ORB) was proposed by Rublee et al. [106] as a faster and more efficient (and also not licensed or patented) alternative to SIFT and SURF. As can be seen from the name, the approach is a combination of FAST (see

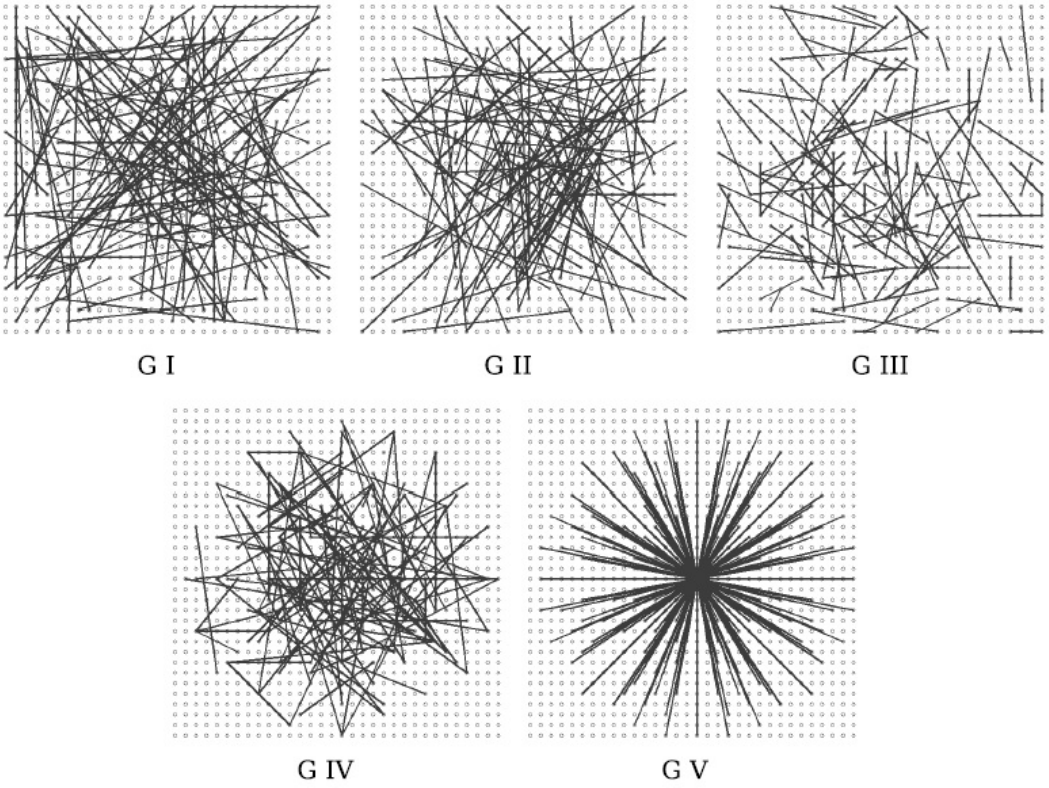


FIGURE 2.12: Sample spatial arrangements of binary tests for constructing BRIEF feature descriptors. G I - IV were randomly generated. From [17]

Sect. 2.1.5.1.2) corner or interest point detector and BRIEF feature descriptors (Sect. 2.1.5.4) while overcoming the limitations of both approaches.

Therefore, the authors add a fast and accurate orientation component to FAST leading to oFAST. First, FAST key-points are computed over an image pyramid to account for multi-scale features. As there is no intrinsic "cornerness" response when computing FAST features, the authors calculate the corner measure of the Harris corner detector [58] (see Sect. 2.1.5.1.1) for each key-point. Then, the top N key-points according to the corner measure are returned over all scales.

To add an orientation to these key-points, the intensity-weighted centroid C of a circular region with radius r[4] and the current key-point O as the origin is computed using the moments of this region as in [104]. The resulting orientation ϑ is then given by the vector $\vec{OC}$.

Additionally, BRIEF features are efficiently computed considering the orientation. The first approach, called steered BRIEF, uses the computed key-point orientation ϑ, discretized to bins of $2\pi/30 = 12$ degrees, to apply a precomputed BRIEF pattern for the specific orientation bin. On the downside, this leads to a loss of variance. To overcome this, the authors use a greedy search for a set of uncorrelated binary tests with high variance and means near 0.5 (as for original BRIEF features) out of all possible binary tests in a training set. This means that the approach, called rBRIEF, learns good binary features from training data.

[4] r is set to the patch size also used for FAST key-point extraction

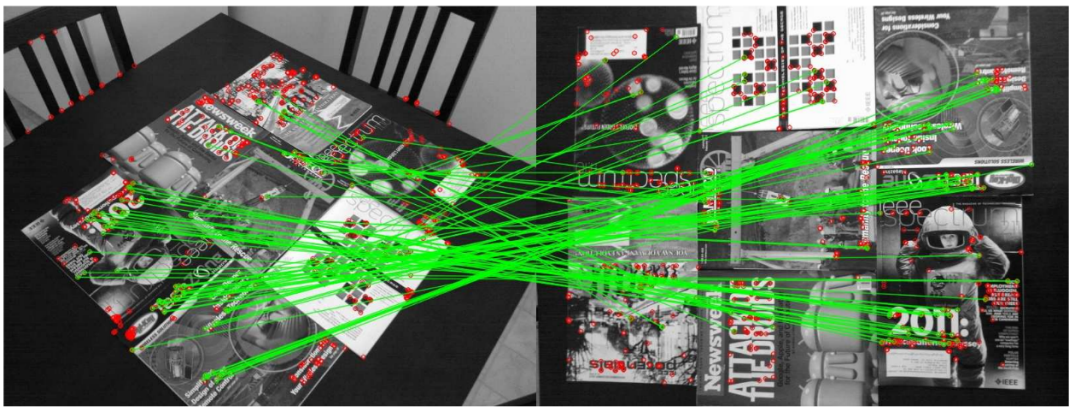


FIGURE 2.13: Illustration of detected ORB features in two frames with a large viewpoint change and their matching. From [106]

ORB now consists of rBRIEF feature descriptors computed on oFAST key-points. For matching of N ORB features over frames, a so-called multi-probe Local Sensitive Hashing (LSH) [46, 88] is used as a nearest neighbor search using hash tables and buckets.

ORB has been shown to be one and two orders of magnitude faster than the SURF (Sect. 2.1.5.3) and SIFT (Sect. 2.1.5.2) feature descriptors, respectively, while achieving similar performance. As an example, the computed ORB features and their matching over two frames are shown in Fig. 2.13.

The authors state that scale invariance has not been adequately addressed, however, a pyramid scheme is used for globally handling scales.

2.2 Traditional Object Detectors

After having revisited the most commonly used techniques to extract features, this section provides an overview of traditional approaches to detect objects in digital 2D RGB images using (subsets of) these features.

As object detection is currently mostly exploited by algorithms using Deep Learning (see Sect. 2.3), the wording "traditional" summarizes all other approaches being proposed until the first major breakthroughs of Deep Learning [71, 136, 117, 49, 114] or not making use of Deep Learning techniques as the main component.

2.2.1 Hough-based Approaches

2.2.1.1 Hough Transform

The Hough Transform (HT) was patented by and is named after Paul VC Hough [66] and was extended by Duda and Hart [32]. It is a global and robust approach to detect parameterizable objects such as lines or circles in binary input images. As binary input images, often Canny edge images (see Sect. 2.1.2.1) are used.

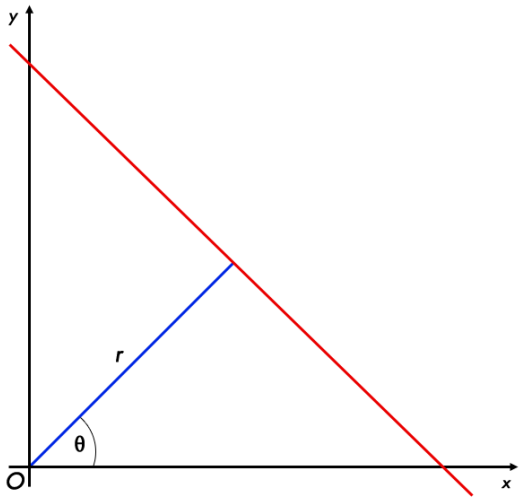


FIGURE 2.14: Illustration of a line (red) in Hesse normal form with r being the distance from the origin (O, blue) and θ being the angle between the x-axis and normal to the line.

As a prerequisite, the so-called Hough space is set up, i.e., a parameter space with as many dimensions as are needed to describe the object to search for, and which is then used as an accumulator.

In the line example, the Hough space has two dimensions, namely r and θ, corresponding to the Hesse normal form of a line:

$$r = x \cos\theta + y \sin\theta \tag{2.3}$$

with r being the distance from the origin and θ being the angle between the x-axis and the line (see Fig. 2.14). As proposed by [32], the Hesse normal form is used because vertical lines in the Cartesian representation ($y = mx + b$) would have an undefined slope m.

The accumulator is then filled by a voting approach. Each point in the binary input image has a set of all straight lines which go through this point. Thus, each point "votes" independently for this set that can be represented by a sinusoid curve in the parameter space, since all lines passing through the point share the same value of r and have different values of θ. Voting is performed by increasing the number of votes of the respective cells of the sinusoid curve in the Hough space. If, for instance, four points lie on a straight line, there will be a peak in the Hough space where the corresponding four sinusoid curves intersect, i.e., at the parameters r and θ of this line.

The final line detection (where the line is unlimited) is then performed by identifying peaks in the resulting Hough space. Please see Fig. 2.15 for an illustration of line detection by applying the HT to a simple binary input image.

By choosing the quantization of the Hough space, one can trade-off between accuracy and efficiency. Limitations of this approach are that (a) the resulting lines are unlimited and might need to be cut, i.e., matched to the image's content, and (b) that

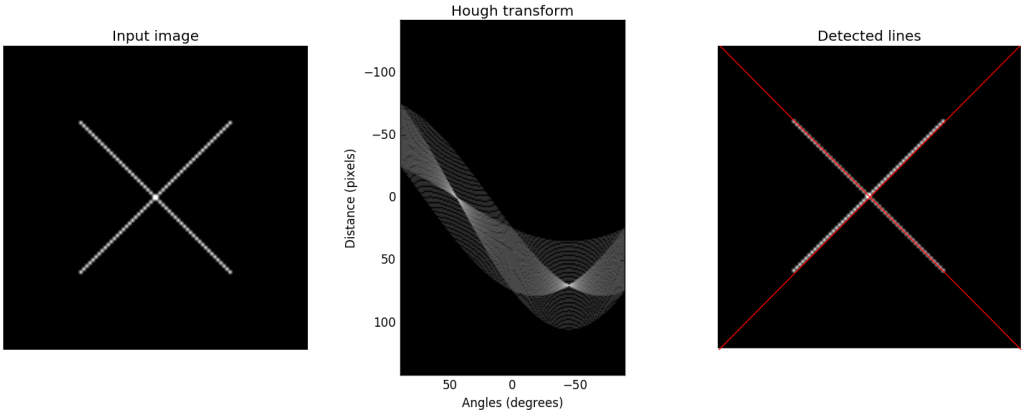


FIGURE 2.15: Illustration of applying the Hough transform for line detection. (Left) Input Canny edge image. (Middle) Hough space (accumulator) with two peaks. (Right) Input edge image with overlaid detected lines.

the complexity of the approach increases exponentially with the number of parameters. Therefore, it is mainly only applied to line and circle detection.

2.2.1.2 Generalized Hough Transform

The Generalized Hough Transform (GHT) [8] is well-known as a general model-based approach for object localization.

Using this technique, non-parametric objects of arbitrary shape can be detected. The dimensions of the resulting Hough space reflect the transformation parameters that are applied to the shape model, e.g., translation in $x-$ and $y-$direction, rotation or scaling. In order to reduce the computational complexity of the approach, these transformation parameters are often restricted to translations only. This leads to a quantized 2D Hough space with respect to a quantization factor ρ, in which the accumulated votes ("counts") in an individual cell reflect the degree of matching between the respectively transformed shape model and the input edge image.

As introduced, the GHT is based on a shape model

$$\mathcal{M} = \{\mathbf{m}_j | j = 1, \ldots, M\} \tag{2.4}$$

consisting of M model points $\mathbf{m}_j$. Each $\mathbf{m}_j$ is represented by its coordinates $\mathbf{x}_j$ in a local coordinate system with respect to some chosen reference point (e.g., the object's center of gravity), and its direction φ_j:

$$\mathcal{M} = \{(\mathbf{x}_j, \varphi_j) | j = 1, \ldots, M\} \subset \mathbb{R}^2 \times [0, 2\pi[. \tag{2.5}$$

In most cases, the direction φ_j of a model point $\mathbf{m}_j$ is defined as the expected gradient γ of the object (edge) at location $\mathbf{x}_j$ in the local coordinate system.

Using the shape model $\mathcal{M}$, the GHT transforms an edge image $\mathbf{I}_E$ (for instance, a Canny edge image as in Sect. 2.1.2.1) into a Hough space $\mathbf{H}^{\mathcal{M}}$ by a voting procedure. Specifically, for each edge point $\mathbf{e} \in \mathbf{I}_E$ the edge gradient direction $\gamma(\mathbf{e})$ is computed

TABLE 2.1: Sample of an R-table for a shape model $\mathcal{M}$ in the Generalized Hough Transform (GHT). For each bin i, the column R_{ϕ_i} contains the local coordinates $\mathbf{x}_j$ of all model points $\mathbf{m}_j$ the expected direction ϕ_i of which falls into this bin i.

Bin number i	$\gamma(\mathbf{e}) \in$	R_{ϕ_i}
1	$[0, \Delta\phi[$	$\mathbf{x}_1, \mathbf{x}_4 \dots \mathbf{x}_n$
2	$[\Delta\phi, 2\Delta\phi[$	$\mathbf{x}_2, \mathbf{x}_6 \dots \mathbf{x}_m$
3	$[2\Delta\phi, 3\Delta\phi[$	$\mathbf{x}_3, \mathbf{x}_5 \dots \mathbf{x}_o$
$\dots$	$\dots$	$\dots$

in the original image. If the edge gradient $\gamma(\mathbf{e})$ matches the expected gradient φ_j of any model point j (up to some tolerance $\Delta\phi$), a vote with value $\mathbf{I}_E(\mathbf{e})$[5] is generated in a Hough cell $\mathbf{c} \in \Omega$ matching the position $\mathbf{c} = \mathbf{e} - \mathbf{x}_j$ of the object's reference point in the global coordinate system (up to a quantization factor ρ). As in the standard Hough Transform, all allowed combinations of edge and model points can cast votes independently.

Thus, the contribution of model point j to the Hough space $\mathbf{H}^{\mathcal{M}}$ can be written as:

$$f_j(\mathbf{c}, \mathbf{I}_E) = \sum_{(\mathbf{e},\gamma(\mathbf{e}))\in\mathbf{I}_E} \begin{cases} \mathbf{I}_E(\mathbf{e}), & \text{if } \mathbf{c} = \lfloor(\mathbf{e} - \mathbf{x}_j)/\rho\rfloor \\ & \text{and } |\gamma(\mathbf{e}) - \varphi_j| < \Delta\phi \\ 0, & \text{otherwise.} \end{cases} \tag{2.6}$$

The Hough space is then generated by summing up the contributions of all model points:

$$\mathbf{H}^{\mathcal{M}}(\mathbf{c}, \mathbf{I}_E) = \sum_{j\in\mathcal{M}} f_j(\mathbf{c}, \mathbf{I}_E) \tag{2.7}$$

In order to reduce computational cost, the shape model $\mathcal{M}$ is initially transformed into a template table, the so-called R-table. Here, the local coordinates $\mathbf{x}_j$ of each model point $\mathbf{m}_j$ with respect to the reference point are organized in dependence of their gradient direction φ_j, which is divided into bins (intervals) of length $\Delta\phi$ (for an example see Tab. 2.1). Using this R-table, the votes that need to be casted by a single edge pixel $\mathbf{e}$ with direction $\gamma(\mathbf{e}) \in$ bin i can directly be looked up and are based on all model points $\mathbf{x}_j$ which are associated to that bin i.

Fig. 2.16 shows an example of detecting a circle using the GHT approach. Please note that there is also a (slightly lower) peak at the confusable object on the right. In general, an advantage of the GHT is its robustness to partial occlusion and noise.

2.2.1.3 Implicit Shape Models (ISM)

An approach of Leibe et al. uses a class-specific codebook of interest point descriptors for detecting objects in an image [75]. This codebook of local appearance is learned from training data. Each entry of this codebook gets assigned the set of observed offsets to the annotated object centroids in the training data. The authors propose to use patches of 25×25 pixels.

[5] For binary (Canny) edge images, this value is simply equal to 1. Otherwise, this value corresponds, e.g., to an edge confidence $\in [0, 1]$ as in Structured Edge Detection (see Sect. 5.1.1).

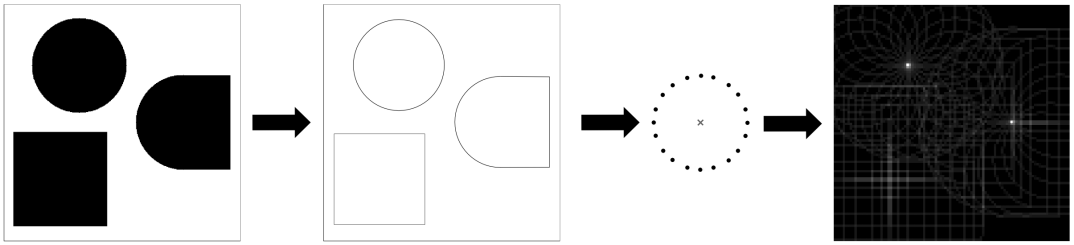


FIGURE 2.16: Generalized Hough Transform (GHT): Input image, binary edge image $\mathbf{I}_E$, shape model $\mathcal{M}$ representing a circle, resulting Hough space $\mathbf{H}^{\mathcal{M}}$ with two peaks (from left to right). Based on [57]

FIGURE 2.17: Car detection using an Implicit Shape Model (ISM). (Left) Input image with detected Harris corners. (Middle) Matches of descriptors with the car-specific codebook. (Right) Sample votes of the two tire patches identifying the object centroid. From [75]

While testing, interest point descriptors are computed as the initial step. Here, Harris corners (see Sect. 2.1.5.1.1) are used, but other interest point detectors could be applied as well (for an overview see Sect. 2.1.5, e.g., SIFT and ORB feature descriptors in Sects. 2.1.5.2 and 2.1.5.5, respectively). The computed descriptors are then matched against the codebook. For each valid match, probabilistic votes are cast that correspond to the set of offsets observed in the training data and stored in an Hough-like accumulator.

For the final detections, peaks in the Hough space are identified. Fig. 2.17 shows an example of detecting a car using an Implicit Shape Model (ISM).

To sum up, ISM show good generalization capabilities when using large codebooks. The construction of these, however, can be seen as a large-scale clustering problem. Moreover, the computational cost is linearly correlated with the number of codebook entries.

2.2.1.4 Hough Forests

Similar to Implicit Shape Models (ISM, see Sect. 2.2.1.3), Gall and Lempitsky proposed to use the Generalized Hough Transform (GHT, see Sect. 2.2.1.2) for class-specific object detection [45]. Here, it is used inside a class-specific Random forest [15] that directly maps the appearance of an image patch to the Hough votes it might cast. This replaced the so far often used generative codebooks by a Random forest.

Each leaf in each Hough tree decides if the current patch belongs to the background or to a part of the object class of interest. If so, it casts a probabilistic vote[6] about the object's centroid w.r.t. the center of the current patch.

[6]Using the probabilistic evidence of the respective patch appearance about the existence of the object at different locations in the image as observed from training data.

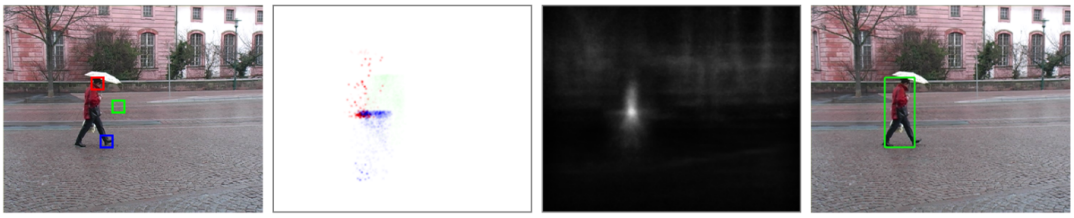


FIGURE 2.18: Example of applying a Hough forest to pedestrian detection. From left to right: Input image with three sample patches; Color-coded votes for these patches; Hough space resulting from the votes of all patches; Final detection. From [45]

The Random forest is trained with supervision using background patches and patches of the object of interest. For training and testing, the patch size is always set to 16×16. Each patch gets assigned a corresponding class label as well as an offset vector from the patch center to the object centroid. For background patches, this offset vector is undefined. In each tree, non-leaf nodes learn binary tests on the patch appearance. A binary test simply compares the difference of a specific channel at two positions against a threshold. The channels could be raw intensities (see Sect. 2.1.1), filter responses or similar.

Each tree in the Hough forest is constructed following the standard ideas of Breiman [15], i.e., here, it learns binary tests, such that the class label uncertainty, as well as the offset uncertainty in the generated leaf nodes, is minimized.

To detect objects, a test image is fed into the trained Hough forest. In each tree, patches are extracted and the binary tests are applied. Each patch ends up in one leaf and then casts a probabilistic vote. This vote is computed based on the Gaussian Parzen-window estimate of the set of offset vectors stored in this leaf as well as the corresponding proportion of object patches in the training data. These votes of all patches are accumulated in a Hough space, which is finally averaged over all trees in the Hough forest. The final detections are returned after having identified the peaks in the resulting Hough space. The class as well as the height and width of the bounding box is fixed for each Hough forest. To achieve scale-invariance, the authors propose to use an image scaling approach.

Fig. 2.18 shows an example of applying a Hough forest to pedestrian detection. Note that the green background patch casts very few votes in the tree-specific Hough space that is shown in the second image. The third image shows the resulting Hough space averaged over all trees in the Hough forest leading to the final output presented in the fourth image.

2.2.2 Rigid Detectors

2.2.2.1 Viola-Jones (VJ)

A classic work of Viola and Jones proposed to use a boosted cascade of classifiers that use Haar-like features (see Sect. 2.1.3) for rapid object detection (primarily used for near-real-time face detection) [123].

In detail, strong classifiers are trained using a modified version of the AdaBoost algorithm [39]. A strong classifier is represented by a linear combination of weak

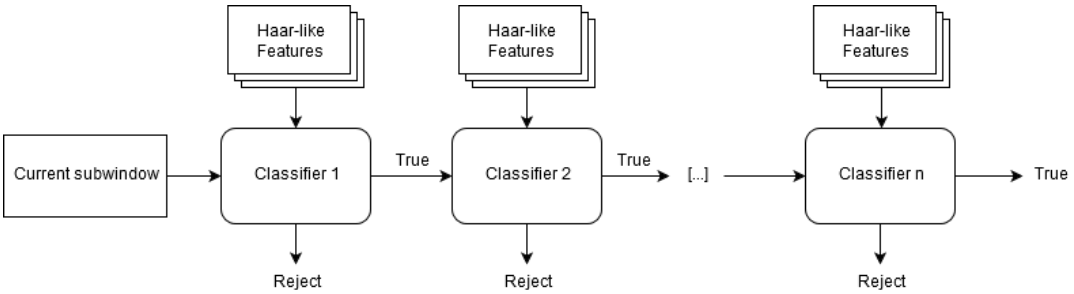


FIGURE 2.19: Cascade of classifiers in the Viola-Jones detector

learners. These weak learners are iteratively generated in the boosting procedure and depend only on a single Haar-like feature that is selected by AdaBoost out of all possible Haar-like features for the current window, thus, being most discriminative.

These strong classifiers are set up in a cascade with increasing complexity as outlined in Fig. 2.19. The reason behind this is to rapidly exclude windows not containing the object of interest. The first, rather simple classifier of the cascade is able to completely reject the current sub-window. If it is not rejected, it is passed to the next more complex classifier and so on. Only, if a sub window passes all classifiers, a detection is output.

Fig. 2.6 shows the top-2 features selected by AdaBoost for face detection overlaid on a sample training image.

In order to rapidly compute the response of the Haar-like features, the concepts of an integral image are used. These allow for retrieving the sum of intensities inside of a rectangular region in constant time as described in more detail in A.1.

Object size variability is addressed by applying a defined set of scaling factors to the Haar-like features used in the weak classifiers at each sub-window.

This contribution is seen as one of the first breakthroughs in CPU-based real-time object detection achieving good detection rates in constrained scenarios.

Still, for many databases (e.g., the Caltech Pedestrian Benchmark [29, 1]), this approach is often used as an absolute standard baseline for comparison with simple and rigid object detectors.

2.2.2.2 HOG + SVM

HOG + SVM is a quite simple object detector proposed by Dalal and Triggs introducing and using HOG features (see. Sect. 2.1.4) and a linear support vector machine (SVM) for classification [23]. Initially, it was intended to be used for pedestrian detection.

In more detail, the approach uses a simple linear SVM to classify a detection window that consists of a dense and overlapping uniform grid of HOG feature descriptors after some pre-processing steps as outlined in Fig. 2.20 and described in Sect. 2.1.4. The detection windows are generated in an overlapping sliding-window fashion using multiple detection window scales to handle object size variability. The final detections are retrieved by a non-maximum suppression (NMS) applied to the output pyramid.

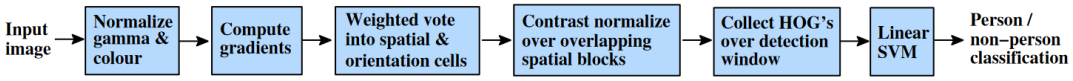


FIGURE 2.20: Pipeline of the HOG + SVM detector including image pre-processing for contrast enhancement (gamma[7]and color normalization) and the extraction of HOG features. From [23]

Regarding the INRIA Person database (see Sect. 3.2.2), the linear SVM is initially trained using the HOG feature descriptors of 1239 positive training samples (+ flipped versions) and 12180 randomly sampled windows from the 1218 negative training images. Afterwards, hard negative samples are identified using the trained SVM. In a second iteration, the training set is augmented with the hard negative samples and the final linear SVM is trained.

At publication time, this approach achieved very good detection rates and could reduce the false positive (FP) rates of the best VJ-based detector [93] by one order of magnitude.

As for the standard Viola-Jones detector (see Sect. 2.2.2.1), this approach is also often used as an absolute standard baseline for comparison with simple and rigid object detectors.

2.2.2.3 Cluster Boosted Tree

In order to handle large intra-class variability as caused by illumination, viewpoint or pose, Wu and Nevatia proposed to use Cluster Boosted Trees (CBT) for binary object/non-object classification [131]. Instead of manually identifying and labeling intra-class sub-categories for a specified object class, CBT performs an unsupervised clustering of the sample space based on discriminative image features resulting in a tree classifier with a model branch per identified sub-category.

In the iterative training procedure, the classifier first consists of only one sub-category that contains all training samples. In each boosting iteration, a weak classifier is selected from the weak hypothesis space for each branch and appended to it. If the discriminative accuracy of the selected weak classifier is too low, the current branch and thus the training samples will be split using an unsupervised clustering based on the weak classifier's feature responses. After each tree update, the corresponding parental classification functions are retrained using the new sub-category split. The growth of the tree is limited by an accuracy threshold.

To detect objects in an image, the class-specifically trained CBT classifier is applied to densely sampled detection windows w.r.t. location and scale extracted from each test image.

The authors reported state-of-the-art results at publication time (2007), e.g., 97.5% and 93.5% recall at equal error rate (EER) on the single- and multi-scale UIUC Image Database for Car Detection, respectively.

[7] $\mathbf{I}_{in} = \mathbf{I}_{out}^{\gamma}$

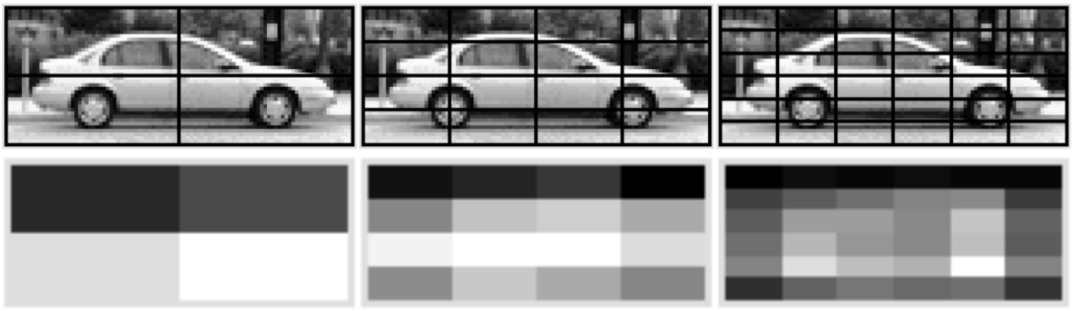


FIGURE 2.21: Spatial pyramid SVM weights used as classifier in the Efficient Subwindow Search (ESS) for car detection. The top row shows the grids of the pyramid levels 2, 4 and 6. The bottom row shows the corresponding learned SVM weight energies for each cell. From [73]

2.2.2.4 Efficient Subwindow Search

Efficient Subwindow Search (ESS) was proposed by Lampert et al. [73] for object detection in contrast to the standard sliding window approach. Here, a branch-and-bound instead of an exhaustive sliding window search is performed that also allows for the (single!) globally optimal localization. The branch-and-bound search is done by splitting the complete parameter space, i.e., all possible sub-windows, hierarchically into disjoint subsets defined by minimum/maximum tuples of the four rectangle coordinates.

In each iteration, the current subset in consideration is split into halves along its largest coordinate interval. Then, for each of the created subsets, the upper bound of the classifier score is evaluated and the search is continued with the most promising one. The procedure is repeated until a single rectangle with the highest score, i.e., the globally optimal solution, is left. Hence, object size variability is implicitly handled.

As an addition, it is possible to incorporate geometric priors to further reduce the number of necessary sub-windows. If multiple objects need to be detected, the search has to be repeatedly applied after having removed the best-scoring region after each iteration.

The overall speedup by reducing the number of sub-windows that need to be evaluated enabled the use of more sophisticated classifiers leading to higher detection performance. For car detection on the UIUC Image Database for Car Detection – Multiscale, the authors applied a so-called hierarchical spatial pyramid that computes bag-of-visual-words histograms of SURF features (see Sect. 2.1.5.3) at 10 pyramid levels, i.e., different grid sizes. The resulting histograms are then classified by a linear SVM to obtain a classification score (see Fig. 2.21). Using the best three car sub-windows per image (obtained by three ESS applications in a row) and an appropriate confidence threshold, the authors reported 98.6% recall at equal error rate (EER) on the UIUC Image Database for Car Detection – Multiscale test set.

2.2.2.5 Fast Car Detection Using Image Strip Features

Image strip features, i.e., lines and arcs with either edge- or ridge-like strip patterns, were proposed by Zheng and Liang in particular for efficient car detection [140].

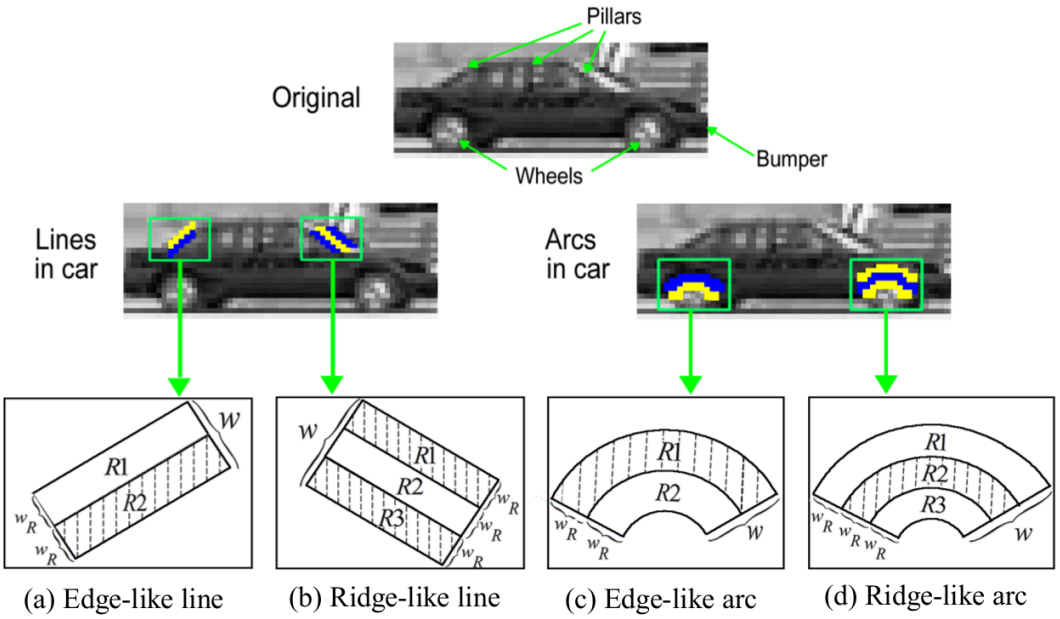


FIGURE 2.22: Different types of image strip features as derived from structural characteristics of cars. From [140]

Please see Fig. 2.22 for an illustration of the different feature types. The feature response describes the contrast of the underlying region by using absolute differences of the mean intensities of the involved regions (R_i with $i \in 2, 3$). Because only intensities are needed to compute the response, the authors use an integral image approach (see Sect. A.1) to reduce memory and computation costs of the approach. To compute non-horizontal or -vertical strip features, smaller rectangles oriented along the centered edgelet are used to approximate the mean intensities of every single strip.

Training on the UIUC Image Database for Car Detection [2] is performed using a so-called complexity-aware RealBoost[8] [110] that accounts for a balance of discriminative capability (by using more complex features) and computational complexity. The 550 positive training samples are resized to 64×32px, i.e., the single-scale detection window size, and also horizontally flipped to double the amount of training data. As negative samples, $10\,000$ images without cars are collected from the Internet.

In order to detect objects in the test set, densely sampled 64×32px detection windows are generated by an exhaustive sliding window strategy (stride of 1, and optionally an image scaling factor of 0.9 per level). Each window is then classified by the boosted classifier. The final detections are obtained by using the standard mean shift algorithm [19] on the positive classifier responses. The authors reported competitive performance to the state-of-the-art on the UIUC Image Database for Car Detection.

2.2.2.6 Integral Channel Features (ICF)

Based on the ideas of Dollár et al. [28], the term Integral Channel Features (ICF) denotes a family of detectors with a similar strategy:

[8] A variant of AdaBoost that uses a real-valued confidence instead of a binary thresholding method.

In a first step, a set of pixel-wise features, namely gradient and color channels, are computed. Using rectangular regions as in Sects. 2.1.4, 2.2.2.1 and 2.2.2.2, feature vectors are built, which are then classified using a linear combination of boosted weak classifiers.

2.2.2.6.1 `ChnFtrs`

`ChnFtrs` [28] applies a `VJ`-like approach (see Sect. 2.2.2.1) to oriented gradients instead of directly using image intensity channels, which results in a significantly improved performance. In detail, gradient histograms (six channels), gradient magnitude (one channel) and the Luv color space (three channels) are used as features, while the feature pool consists of $30\,000$ rectangles randomly distributed over the image. A linear combination of $2\,000$ decision trees (with depth 2 and using AdaBoost [39]) is used for classification.

A three-stage training of the strong classifier is performed using a first stage with $5\,000$ random negative samples and two stages with bootstrapping for mining $5\,000$ additional hard negatives for each stage.

The classifier is applied in a sliding window fashion with a step size of 4px and densely sampled detection window scales to account for handling object size variability. A shrinking factor of 2 or 4 could be used for speeding up without significantly losing detection accuracy. A non-maximum suppression-like (NMS) approach is used to obtain the final detections.

2.2.2.6.2 Fastest Pedestrian Detector in the West (`FPDW`)

Dollár et al. further modified their `ChnFtrs` detector being 1 - 2 orders of magnitude faster than the original approach leading to the so-called Fastest Pedestrian Detector in the West (`FPDW`) [27].

This is done by rescaling the input image only N/K instead of N times. Features are only computed on the rescaled images. The major speed-up is related to their finding that the feature response of the remaining $N - N/K$ scales can be rapidly approximated. The approximation not only works for gradient magnitude and intensities but also for HOG features in adjacent scales up to half an octave in each direction. Until then, typically $8 - 16$ scales per octave were used in image scaling approaches. Within each octave, a small classifier pyramid is used for multi-scale detection. Please see Fig. 2.23 for an illustration of this approach.

The rest of the detection pipeline remains the same as for the `ChnFtrs` detector. Dollár et al. showed that the processing time could be reduced by up to two orders of magnitude with only a minor loss of detection accuracy.

2.2.2.6.3 VeryFast

The `VeryFast` detector of Benenson et al. [10] learns models based on `ChnFtrs` at multiple canonical scales instead of resizing the input image. Therefore, this approach is in accordance with one of the core ideas of the Viola-Jones (`VJ`) detector (see Sect. 2.2.2.1), i.e., to scale the features and not the images.

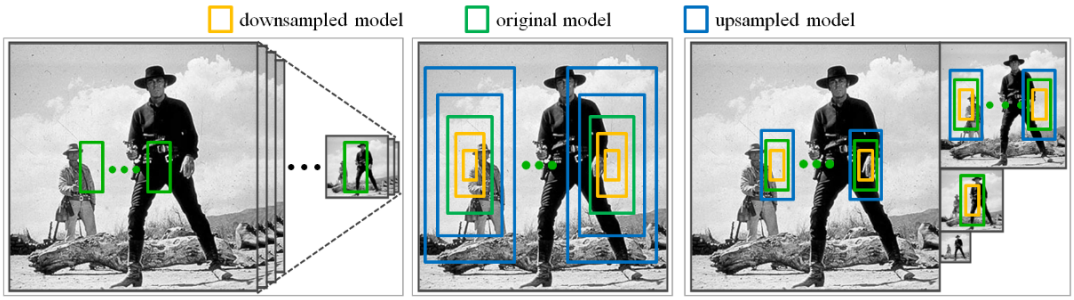


FIGURE 2.23: Hybrid scaling approach of FPDW using feature approximation of nearby scales (c) in contrast to standard image scaling (a) and feature scaling (b) approaches. From [27]

Basically, the insights of the FPDW approach (see Sect. 2.2.2.6.2) are reverted. Instead of approximating the feature response of nearby scales, they adjust the trained decision stumps such that they produce a classification output as if the feature response had been computed at another scale. Remember that a decision stump consists of a channel index, e.g., a specific color, gradient magnitude or gradient direction channel, a receptive field or region of interest (ROI) as well as a threshold τ. To scale such a decision stump using a relative scaling factor s, the region of interest and the threshold need to be updated by simply scaling the receptive field and by computing $\tau' = \tau \cdot r(s)$ with a channel-specific scaling function $r(s)$ (see [10]), respectively.

At test time, the integral channel features are computed on the input image. Using the described approximation, decision stumps for all necessary scales can be generated from the few trained decision stumps. Finally, the complete classifier responses can be computed.

With having adapted the FPDW code for use with GPU and incorporating the idea of soft-cascades [137], the approach achieves 50 FPS on the INRIA Person database (see Sect. 3.2.2).

By additionally exploiting the geometry of stereo images by means of ground plane estimation and computing stixels for each column of the input image, the authors achieve 135 FPS. For more details, please refer to [10].

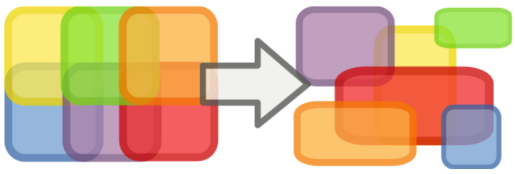
2.2.2.7 Roerei

The Roerei[9] detector was introduced by Benenson et al. [11] while revisiting core assumptions of HOG + SVM (see Sect. 2.2.2.2). At that time, most state-of-the-art detectors were built on the idea of component models with each component consisting of deformable parts (see Sect. 2.2.3.1). However, the parts itself are so-called "weak" classifiers often made of simple HOG + SVM detectors.

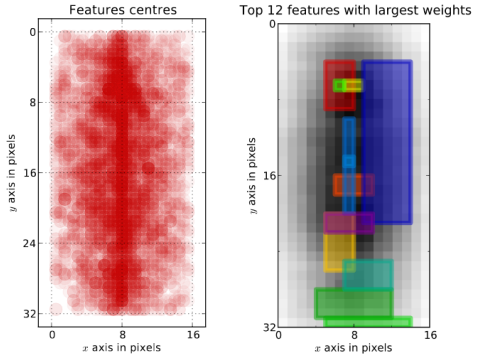
The idea was to build a rigid classifier with improved assumptions and design choices in contrast to the original HOG + SVM detector:

- Irregular cell pattern for computing histograms learned during training (see. Fig. 2.24)
- Feature pre-processing (global normalization)

[9] Dutch for scrambled eggs



(A) Instead of a regular pattern as in HOG + SVM (see Sect. 2.2.2.2), the authors propose to learn the cell pattern as well.



(B) Example of a trained model for pedestrian detection. (Left) Centers of the individual features. (Right) Top-12 features and their corresponding cells.

FIGURE 2.24: Illustration of design aspects of the `Roerei` detector. From [11]

- HOG + LUV feature channels
- Non-linear classifiers (decision trees over HOG features)
- Multi-scale models
- Learning algorithm (boosting instead of linear SVM)

Fig. 2.24 shows an illustration of the irregular cell pattern as well as an example of a trained model.

With these thoroughly evaluated design choices, the `Roerei` detector was able to outperform all other approaches on INRIA Person [23] (see Sect. 3.2.2) available at publication time (2013) [11].

2.2.2.8 Spatial Pooling

Paisitkriangkrai et al. state that a careful combination of discriminative features is still crucial in order to achieve top performance on tasks like pedestrian detection. To this end, the authors proposed to use spatially pooled low-level features (from now on this approach is referred to as Spatial Pooling) [98]. As the final feature, a 9×9 covariance matrix of nine low-level features, namely the pixel location, first and second order derivatives along both axes, gradient magnitude and two differently computed edge orientations, is computed. Since a covariance matrix is symmetric, only the upper triangular part (45 values) is stored. To increase processing speed, the covariance features are precomputed following the ideas of integral images (see Sect. A.1) applied to the covariance descriptor as in [121] and, for each patch, stored in a vector for easier processing. In order to improve robustness and spatial invariance of the covariance descriptor, spatial pooling (in particular, the max pooling operation) is used. Here, the covariance descriptors of all pixels belonging to the current patch are summarized into a single output covariance descriptor, called *sp-Cov*.

For pedestrian detection, multi-scale sp-Cov patches (8×8, 16×16 and 32×32) are extracted with stride 1, a pooling region of 4×4 and a pooling stride of 4 in each 64×128 detection window. Additionally, Luv color features are extracted. The final

detector that uses these features consists of $2\,048$ weak classifiers, here, level-3 decision trees, used in combination with AdaBoost [39] as a strong classifier. For the first iteration of training the strong classifier, features extracted from the positive training samples as well as randomly sampled negative background patches are used. Afterwards, three stages of hard negative mining (also referred to as bootstrapping) are performed to increase performance. As a new state-of-the-art at publication time, the authors reported a miss rate of 0.04 at 1 FPPI on the INRIA Person test set [23] (see Sect. 3.2.2).

2.2.3 Part-Based Approaches

2.2.3.1 Deformable Part Models (DPM)

Different from rigid object detectors as described in Sect. 2.2.2, Felzenszwalb et al. proposed the first really successful application of so-called Deformable Part Models (DPM), i.e., models that consist of deformable parts following the pictorial structure ideas of Fischler and Elschlager [38], which are discriminatively learned and finally applied at multiple scales [37].

Such a DPM consists of a "root" model which basically is a rigid HOG + SVM detector (see Sect. 2.2.2.2) applied at a coarse resolution. Furthermore, it consists of star-structured parts similar to the parts of the "spring" model of Fischler and Elschlager [38]. These parts, again, are HOG + SVM detectors but applied at a finer resolution. The "springs" are modeled by a spatial model of the parts' locations introducing deformation costs for individual offsets. The final score of a model at a specific location is given by the score of the "root" model plus the scores of the individual parts minus the respective deformation cost for each part. The score is defined as the dot product between a set of weights (either of the "root" or a part model) learned by a latent SVM (LSVM) and the underlying HOG features. An illustration of the complete detection process at one scale can be found in Fig. 2.25.

The "root" model is trained on the bounding box annotations as described in Sect. 2.2.2.2, also initializing the individual parts. A latent SVM is a generalization of a standard linear SVM that incorporates latent values (e.g., specific placements of the individual parts). Thus, a binary classification problem can be iteratively solved using a score function $f_\beta(x)$ for each example x:

$$f_\beta(x) = \max_z \beta \cdot \Phi(x, z) \tag{2.8}$$

where β is a vector of model parameters and z are the latent values, i.e., a specific model configuration, if z is specified as the best scoring configuration for the current (positive) sample x. For optimization, a coordinate descent algorithm [129] is used.

As a pre-processing step while testing, a HOG feature pyramid of the input image is computed at multiple resolutions. In order to handle size variability, an image scaling approach is used. Then, the trained DPM is applied in a sliding-window-fashion computing scores as described above. Fig. 2.26 shows a sample pedestrian DPM and the resulting detection on the original image.

Until the breakthroughs of object detectors based on Deep Learning (see Sect. 2.3), deformable part models were one of the most accurate object detectors. They also allow for easily interpretable results.

2.2.3.2 Part Alphabet and Pose Dictionary

Yao et al. proposed to use a discriminatively learned part alphabet and pose dictionary for pedestrian detection by transferring ideas from text recognition [135]. Both pedestrians and text can be composed of a (visual) alphabet as structured objects including significant variability.

The visual alphabet is automatically learned from training data by using a discriminative clustering algorithm of Singh et al. [118] and consists of 3×3 HOG (see Sect. 2.1.4) representations of body parts, e.g., head, shoulders, arms, and an offset to the object center. Human poses are then represented by sequences of parts as words are represented by sequences of letters. A dictionary of valid poses is constructed from training data with all variability modes regarding pose, occlusion and viewpoint contained in the training set. The pose as a 1D sequence can now be compared to valid poses in the pose dictionary by a rather simple string matching approach as suggested by Navarro [95].

In a first step, the detection pipeline generates hypotheses by applying the individual part detectors to densely sampled multi-scale patches of the input image and performing a Hough voting for object centers using the respective offsets. Afterwards, these hypotheses are verified by building the corresponding 1D part sequence and computing the average edit distance [78] originally used for the dissimilarity of two strings between the detected pose and T closest entries in the pose dictionary as obtained by dictionary search [72]. To compensate for using only local parts, each hypothesis is also verified by a root filter, i.e., using the output probability of a Random Forest that has been trained on HOG descriptors of positive training hypotheses. Please see Fig. 2.27 for an illustration of the two-step approach.

On the TUD Pedestrians ([5], see Sect. 3.2.3) test set, the authors reported state-of-the-art results at the time of publication (2014). On the INRIA Person data set ([23], see Sect. 3.2.2) the approach ranked between VeryFast/FPDW and Integral Channel Features (ICF).

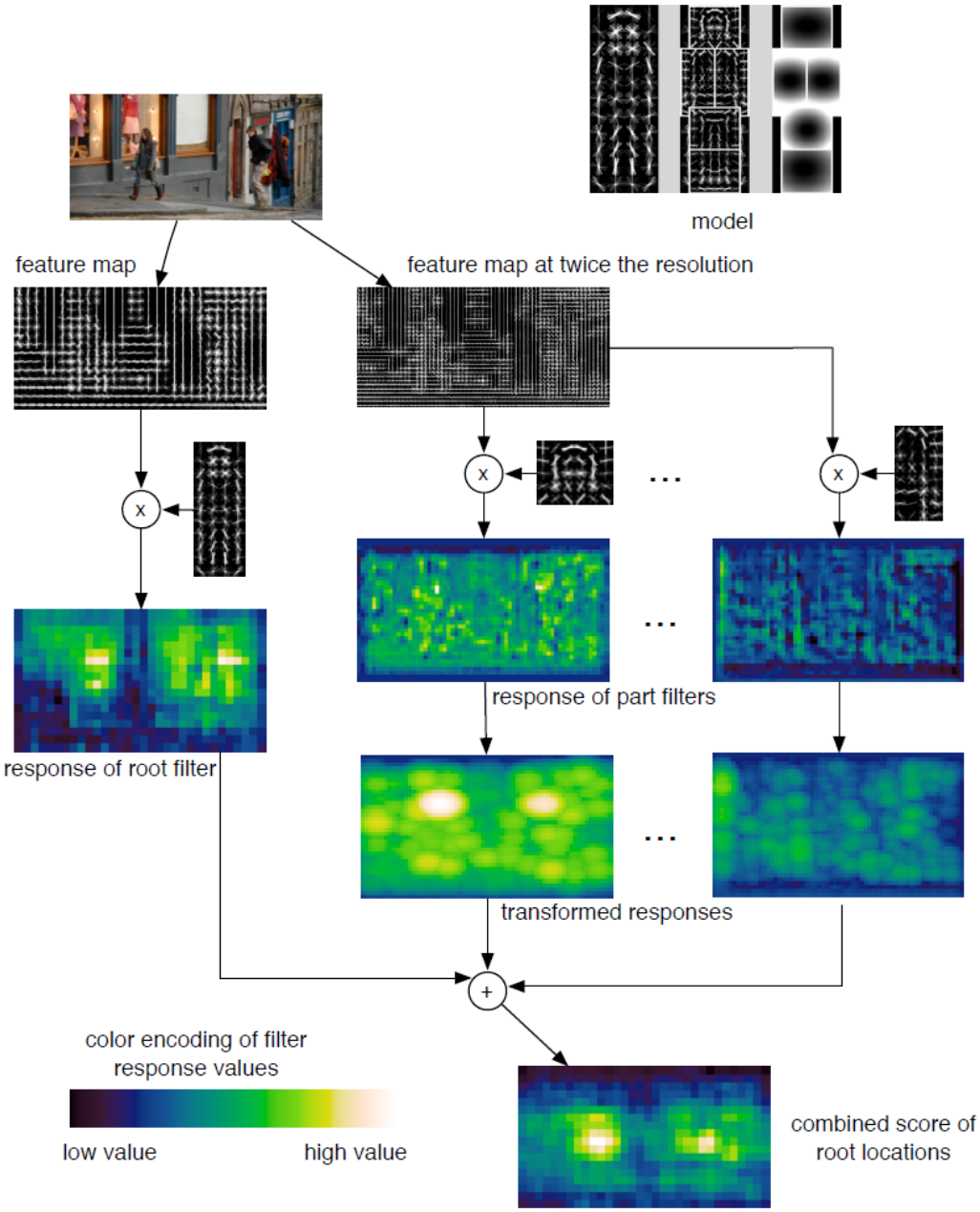


FIGURE 2.25: Example detection with a pedestrian deformable part model (DPM) for a single scale. From [36]

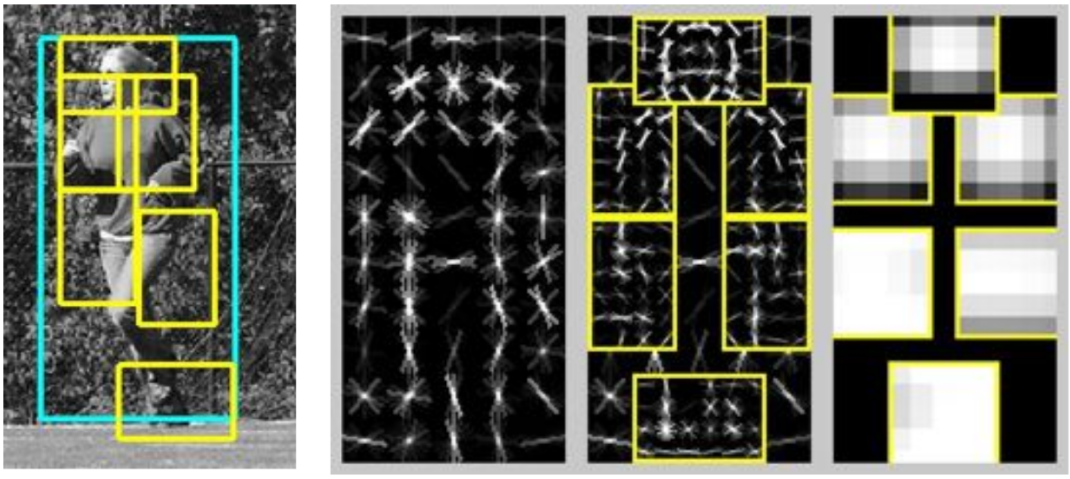


FIGURE 2.26: Example detection with a pedestrian deformable part model (DPM). From left to right: Input image with detected pedestrian; Rigid and coarse "root" HOG template; Higher resolution part templates; Spatial model of part locations. From [37]

2.3 Deep Learning Based Object Detectors

Recently, a lot of general object detection approaches have evolved that make use of the further increasing large-scale computation capabilities of graphics processing units (GPU).

Initially, convolutional neural networks (CNN, see Sect. A.2 for a brief explanation of the basic concepts) were developed and already applied in the late 1980s and 1990s for, e.g., digit recognition on CPUs. In 1989, LeCun et al. applied backpropagation to learn the CNN weights, i.e., the convolutional filter kernels, directly from images leading to a fully automatic training procedure in contrast to the often used handcrafted features [74]. Here, the most important step was to replace networks consisting only of fully connected layers with a network that mainly uses convolutional and pooling layers for image recognition.

The three basic concepts of CNNs drastically reduced the number of parameters that have to be learned: Local receptive fields, parameter sharing and pooling layers[10] that simplify or condense the output information of the previous layer (see Sect. A.2).

However, support vector machines were widely used at that time and after that. Only when Krizhevsky et al. [71] showed significantly improved state-of-the-art results on the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) [24], CNNs gained enormous popularity and are heavily used and exploited until now. This major breakthrough was achieved by increasing the number of convolutional and pooling layers in combination with using much more training data, which required the use of parallel GPU computation capabilities.

This section follows the popular classification of CNN object detection approaches into one- and two-stage detectors.

[10] Max pooling layers were first mentioned by Weng et al. [126, 127] in 1992. Cireşan et al. adapted this operation to be used in CNNs on GPUs [20].

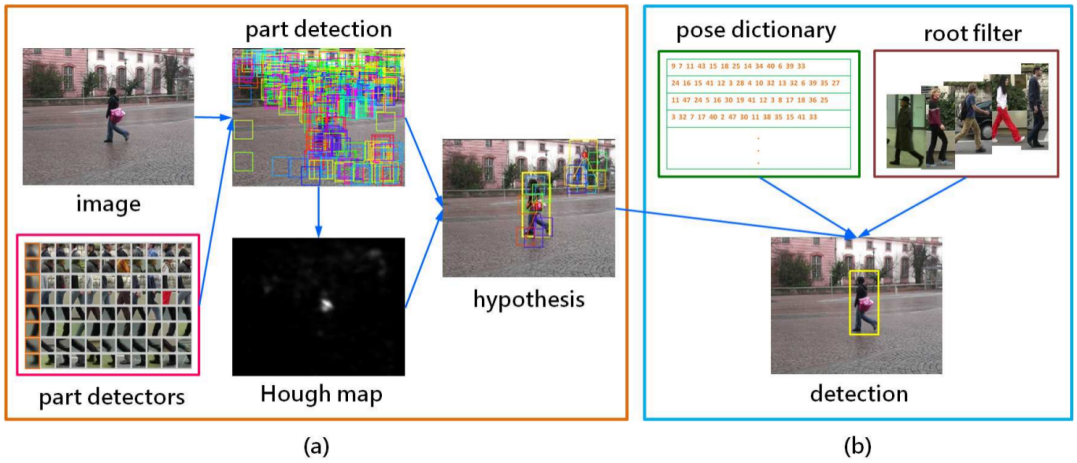


FIGURE 2.27: Illustration of pedestrian detection using a learned part alphabet and pose dictionary. (a) Hypothesis generation using part detectors and Hough voting. (b) Hypothesis verification using a pose dictionary containing valid poses and root filters learned from training data. From [135]

2.3.1 Two-Stage Detectors

Two-stage detectors are proposal-driven approaches, i.e., as an initial step, object proposals (candidate locations) are generated based on the input image. Subsequently, these object proposals are then classified whether they belong to a class of interest or to the background. Sect. 2.3.1.1 describes an early, biologically inspired approach to car detection, where a rather shallow classification network is applied to candidate windows generated in a sliding window manner.

As an overview regarding general networks for candidate rejection, i.e., those needed for the second stage, Sect. 2.3.1.2 describes convolutional neural networks that are used for classification, i.e., the CNN output is a class label only without any localization or bounding box prediction (see Sect. 1.6). These networks need to be used in combination with a proposal generation stage, e.g., a sliding window approach or SelectiveSearch [122].

Typically, two-stage approaches achieve state-of-the-art results on nearly all object detection data sets, while one-stage approaches (see Sect. 2.3.2) achieve slightly lower accuracies but at much higher frame rates.

2.3.1.1 Biologically-Inspired Approach: Sparse Localized Features

Mutch and Lowe proposed a biologically inspired model for (multi-)class object detection [94]. In particular, the authors extended the visual cortex-based model of Serre et al. [115] with biologically inspired properties such as feature sparsity to increase generalization capability by discarding features with low weights, and lateral inhibition, i.e., preserving locally dominant responses while eliminating weaker ones.

The proposed model consists of an initial input image layer and four processing layers (see Fig. 2.28). The input image layer includes an image pyramid with 10 scales (image scaling). The processing layers consist of alternating simple convolutional

and complex pooling layers. In the first convolutional layer (S1), Gabor filters with four orientations are applied at each location and scale. The following complex layer performs a $10 \times 10 \times 2$ max pooling with a location stride of 5 and a scale stride of 1 in order to achieve subsampling. The intermediate feature layer (S2) performs template matching via convolutions between patches of the C1 layer at every position and scale and d[11] prototype patches of different sizes learned from training data. The final global invariance layer (C2) performs pooling such that d outputs are generated, in particular, the highest response for each of the d prototype patches over all positions and scales. Finally, a linear SVM performs the classification of the input image using the C2 layer output.

Concerning a detection task on the UIUC Image Database for Car Detection test sets, the authors use a dense sliding window approach with a stride of 5 to generate input image patches and classify each patch whether it contains a car. Regarding the multiscale test set, also different sizes of input image patches are used in order to handle object size variability. At publication time (2006), the authors reported state-of-the-art results on a classification task on the Caltech-101 data set [35] and on a detection task on the UIUC Image Database for Car Detection [2], respectively.

2.3.1.2 Classification Networks

This section describes convolutional neural networks (CNN) that generate a classification output with respect to an input image patch. Thus, this type of network can be used for candidate or proposal rejection, if they are applied in combination with a proposal generation stage, e.g., a sliding window approach or SelectiveSearch [122].

2.3.1.2.1 VGG16

VGG16[12] is a popular convolutional neural network (CNN) used for image classification and was proposed by Simonyan and Zisserman [117]. The main idea is to use a deeper architecture (consisting of 16 weight layers) than typical CNNs at this time but with only small convolutional filters (3×3).

The standard input image size is $224 \times 224 \times 3$. This input data is passed sequentially to five blocks each consisting of two or three 3×3 convolutional layers followed by a 2×2 max pooling layer. This is followed by a stack of three fully connected layers, where the first two each have $4\,096$ channels. The last fully connected layer consists of $1\,000$ channels, one for each class in the ImageNet database [24]. The final classification output is then given by a softmax layer. Please see Fig. 2.29 for a visualization of the VGG16 architecture.

Training is performed using $224 \times 224 \times 3$ input image crops and corresponding class labels. To train the network, first, a subset of 11 convolutional layers (1, 3, 5, 6, 8, 9, 11, 12) and the three fully connected layers is trained on the ImageNet training set with random initialization. Then, the first four convolutional and the three fully connected layers of the final network are initialized with the already trained weights, all remaining layers are randomly initialized and the training is repeated. When training the whole network, the multinomial logistic regression objective, i.e.,

[11] The authors used $d = 4\,096$.

[12] It is named after the **V**isual **G**eometry **G**roup of the University of Oxford.

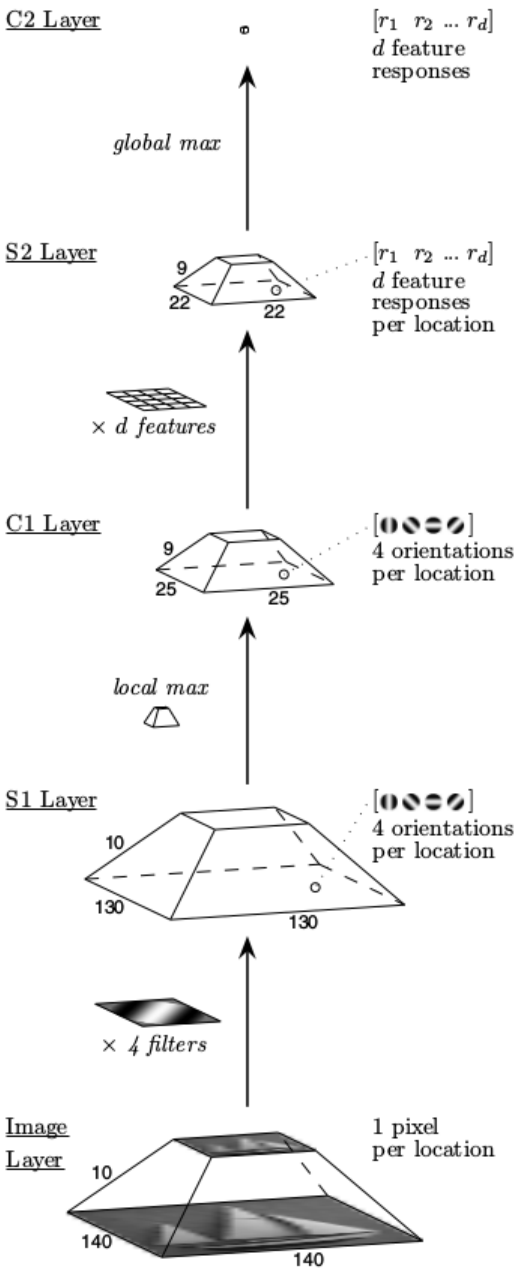


FIGURE 2.28: Illustration of the sparse localized features model architecture. An input image layer (image pyramid with 10 scales) is processed by alternating simple ("S") convolutional and complex ("C") pooling layers. The final feature responses are then classified by a linear SVM. From [94]

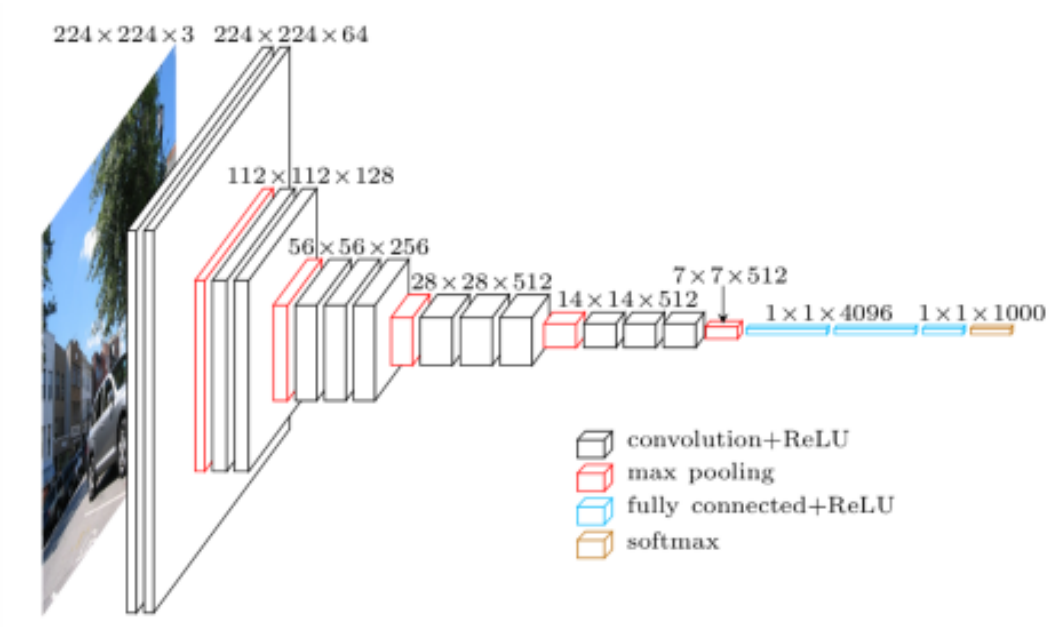


FIGURE 2.29: Illustration of the VGG16 architecture. From [14]

a generalization of the logistic regression to multi-class problems, is optimized using a standard mini-batch stochastic gradient descent (SGD) with backpropagation [117].

As state-of-the-art results on the ImageNet classification challenge at publication time (2015), the authors reported 76.3% top-1 validation accuracy and 93.2% top-5[13] validation and test accuracy.

2.3.1.2.2 ResNet

Although CNNs with increasing depth (e.g., VGG16, see Sect. 2.3.1.2.1) often achieved state-of-the-art results, it generally is not sufficient to arbitrarily increase the depth in order to achieve better performance. One main reason that prevented convergence of even deeper CNNs, the so-called "vanishing gradient" problem, i.e., near-zero or zero gradients caused by an increasing number of subsequent multiplications, could be tackled by using normalized initialization [51, 63] and intermediate normalization layers as in [68]. When being able to converge, deeper CNNs often faced a problem of performance degradation that counter-intuitively is not due to overfitting, see for instance [61].

To overcome these problems, He et al. proposed ResNet [62]. Instead of implicitly learning the underlying non-linear mapping $\mathcal{H}(x)$ of a stack of layers, the central idea is to decompose the desired underlying mapping function $\mathcal{H}(x)$, which is expected to be close to the identity[14], into the identity function x and a residual mapping $\mathcal{F}(x) : \mathcal{H}(x) = x + F(x)$. The identity function can be realized by a shortcut connection, i.e., a connection that skips one or more layers. Here, a shortcut connection simply performs identity mapping by element-wise addition; hence no extra parameters are introduced. The shortcut connection together with the standard

[13] I.e., the correct class must appear within the five highest-scoring predictions.
[14] This is justified, e.g., when additional layers are added to the network.

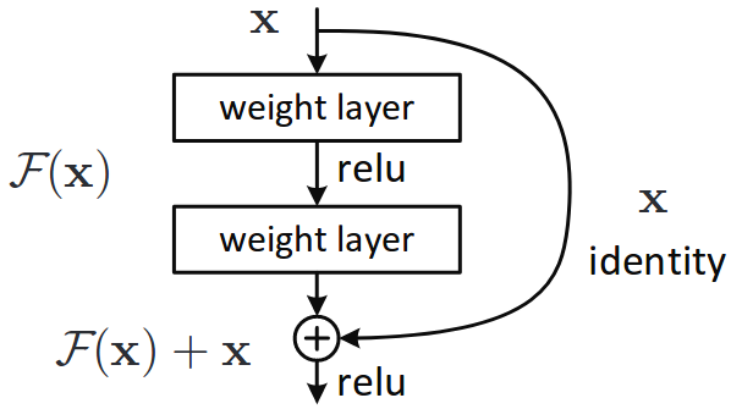


FIGURE 2.30: A building block used for residual learning in ResNet. From [62]

weight layers constitute a so-called building block (see Fig. 2.30). The advantage is that the standard weight layers now only have to learn the residual function $F(x)$, expected to be close to 0, which can be better learned than the original mapping function $H(x)$.

Using these building blocks, the authors enabled building significantly deeper nets that do not suffer from performance degradation. Instead, they reported state-of-the-art results on the ImageNet classification challenge achieving 94.75%, 95.40% and 95.51% top-5 validation and test accuracy for 50, 101, and 152 layer networks. As for VGG16, a single fully connected (FC) layer with $1,000$ neurons and a softmax layer yield the final classification result. Due to the much larger depth, however, the previous two FC layers with each $4\,096$ channels could be omitted. This design choice also leads to a much lower complexity: Even the largest ResNet-152 only has 11.3 billion FLOPs[15] as compared to 15.3 billion FLOPs of the standard VGG16 model.

2.3.1.2.3 MobileNets

Howard et al. proposed MobileNets [67], i.e., light-weight deep convolutional neural networks to be used for embedded or mobile computer vision tasks. Two hyper-parameters allow for a task- and constraint-specific trade-off between computational complexity and accuracy.

The efficiency of their approach is realized by depth-wise separable convolutions. These can be seen as a factorization of a standard convolution into a depth-wise and a subsequent 1×1 (point-wise) convolution that drastically reduces the computational complexity. The depth-wise convolution only applies a single filter to each input channel. As the input channels are therefore not combined in this operation, the point-wise convolution is needed to form a linear combination of the filter responses. This building block is followed by the standard batch normalization [68] and ReLU (Rectified Linear Unit) to achieve non-linearity. Intermediate down-sampling is realized by the convolutional stride.

[15] Floating point operations

The final network architecture consists of an initial standard convolutional layer, 13 depth-wise separable convolutional layers, a global average pooling layer [81], the final fully connected layer with $1\,000$ neurons, and a softmax to create the final classification output.

To further reduce complexity, e.g., when there are hardware restrictions, a width multiplier α is introduced. This parameter is used to uniformly downscale the network at each layer, for instance, with α set to $0.75, 0.5$ or 0.25 with a parameter reduction of approximately α^2. Note that this only creates a downscaled network structure that needs to be trained from scratch.

Additionally, a resolution multiplier ρ can be used to reduce the computational complexity by implicitly setting the input image size (a size of 224×224 corresponds to $\rho = 1$, and, e.g., a size of 128×128 corresponds to $\rho = 0.5714$). The parameter reduction is also approximately ρ^2.

The full yet extremely efficient MobileNet, i.e., $\alpha = 1$ and $\rho = 1$, achieves 70.6% accuracy on ImageNet (4.2 million parameters) in comparison to 71.5% accuracy of the standard VGG16 (138 million parameters).

2.3.1.3 R-CNN and Variants

The first most prominent two-stage detector, namely R-CNN, was introduced by Girshick et al. [49]. The full name *Regions with CNN features* already explains the two steps of this approach:

First, general, i.e., not class-specific, region proposals are generated on the input image. Here, the Selective Search algorithm of Uijlings et al. [122] is used generating approximately $2\,000$ region proposals. In general, their approach does not rely on a specific proposal generation mechanism.

In a second step, a CNN[16] is used to extract a fixed-length[17] feature vector from a mean-subtracted $227 \times 227 \times 3$ input patch. To match these required input dimensions, a simple warping regardless of the size or aspect ratio is used including a context of 16 pixels in each direction around the bounding box or proposal in the warped result.

For obtaining the final detections, the feature vector is passed to a set of class-specific SVMs for classification. Additionally, the same feature vector is used for bounding box regression in order to improve inaccurate bounding box proposals that might have been generated by Selective Search. As a last post-processing step, a greedy non-maximum suppression (NMS) is applied for each class independently. An illustration of this pipeline is shown in Fig. 2.31.

The authors stated a relative 30% higher mean average precision (mAP[18]) at the PASCAL Visual Object Classes (VOC) challenge as compared to the state-of-the-art at publication time (2014). On the downside, their approach needs approximately more than 40 seconds per image on a GPU to evaluate.

[16] The architecture is the same as proposed by Krizhevsky et al. [71] consisting of five convolutional and two fully-connected layers.

[17] The authors use a $4\,096$-dimensional feature vector

[18] Mean of the average precision (AP), i.e., an approximation of the area under curve (AUC) of the precision-recall-curve, over all classes.

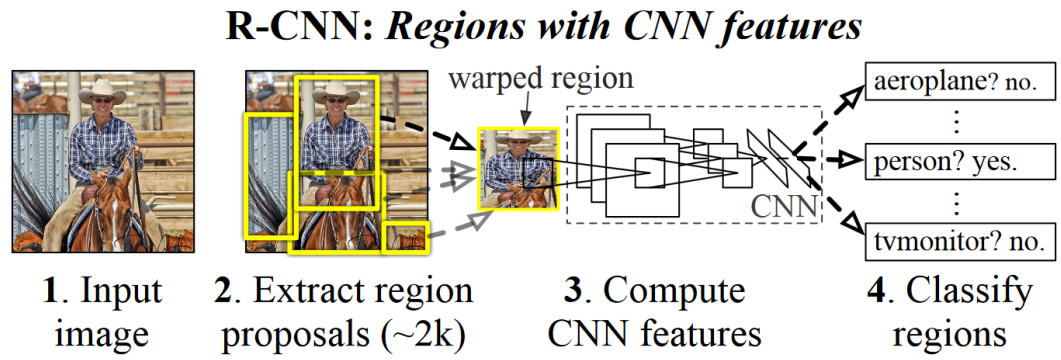


FIGURE 2.31: Illustration of the R-CNN object detection system. From [49]

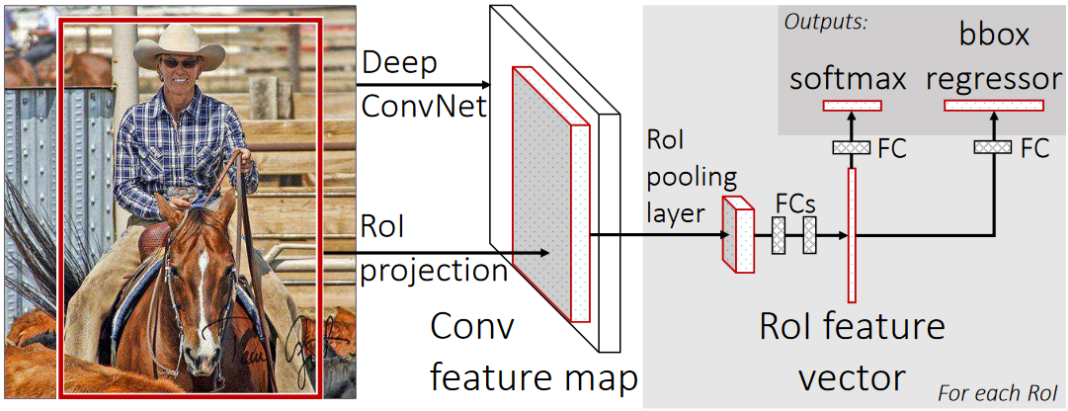


FIGURE 2.32: Illustration of the Fast R-CNN object detection system. From [48]

The training paradigm of Transfer Learning used in this approach still holds true for many recent approaches. First, they perform a supervised[19] so-called pre-training on the large-scale ImageNet [24] database. Here, only image-level annotations are used. Afterwards, the pre-trained network is adapted to the domain and task at hand in the so-called fine-tuning step. For this publication, they wanted to solve a detection task on warped PASCAL VOC proposals.

2.3.1.3.1 Extensions and Variants

Fast R-CNN is a follow-up work by Girshick [48] tackling the main drawbacks of R-CNN:

- **Multi-stage training:** Needs fine-tuning of CNN, SVM and bounding box regressor
- **Expensive training:** By means of training time and memory/disk usage
- **Runtime:** The approach needs more than 40 seconds per image to be evaluated on a GPU

[19] In general, unsupervised pre-training is suggested. Here, the authors used the architecture of Krizhevsky et al. [71] that demonstrated top performance on ImageNet [24] using only supervised learning.

To do so, Girshick proposed to feed the complete input image to a CNN generating a convolutional feature map instead of passing the generated proposals again and again to the complete CNN. Using this convolutional feature map, regions of interest (ROIs), i.e., the proposals generated by Selective Search, can directly be extracted and then are warped by a ROI pooling layer. Here, any proposal (ROI of size $h \times w$) is transformed into a small feature map of fixed dimensions $H \times W$, e.g., 7×7, by using max pooling on sub-windows that are approximately sized according to $h/H \times w/W$.

Afterwards, two fully-connected layers extract the ROI feature vector that is used to simultaneously compute a class score as well as perform bounding box regression, i.e., a so-called multi-task network. The pipeline of Fast R-CNN [48] is illustrated in Fig. 2.32.

With these improvements, Girshick transformed the multi-stage training into a single-stage training. The complete network can be trained end-to-end using a multi-task loss. This also removed the necessity to store the feature vectors on disk as for R-CNN. A forward pass can now be performed in 320ms instead of 47 seconds. This also leads to a reduced training time by almost one order of magnitude. However, Selective Search still adds another two seconds to the overall processing time per image. To sum up, an absolute 4% gain in mAP on PASCAL VOC 2012 was achieved.

Faster R-CNN [102], a further follow-up work of Ren et al., wanted to eliminate the remaining bottleneck, namely Selective Search, by speeding up the region proposal (RP) step . Instead of running Selective Search separately, the authors propose to use the already computed convolutional feature map in an intermediate mini network, the so-called Region Proposal Network (RPN), that only adds approximately 10ms to the complete processing time. The RPN generates region proposals which are then processed in the same manner as in Fast R-CNN (see the illustration in Fig. 2.33).

The RPN basically uses a sliding-window technique on the convolutional feature map. For each location, an "objectness" score , i.e., object/non-object probabilities as well as four values (2D center position, aspect ratio, scale) for bounding box regression are computed for each of the k anchor boxes, i.e., a proposal bounding box consisting of a scale and an aspect ratio (see Fig. 2.34). These k[20] boxes are hand-picked, but could also be learned from training data. Thus, the complete network including all components is learned in an end-to-end fashion.

All in all, Faster R-CNN achieved state-of-the-art detection accuracy inter alia on the PASCAL VOC challenge while running at 5 frames per second (FPS) using at least the top-300 region proposals generated by the RPN.

There exist more variants such as Mask R-CNN by He et al. that extends Faster R-CNN by an additional branch for predicting the object mask at pixel-level [60]. Lin et al. proposed Feature Pyramid Networks (FPN) [82] that build feature pyramids inside CNNs[21] to more explicitly address multi-scale detection problems.

As a near-hybrid or 1.5-stage approach, Lenc and Vedaldi proposed R-CNN minus R [77]. Here, the authors stated that for many detection tasks[22] it might be sufficient

[20] Typically, $k = 9$ anchor boxes are used by the authors (three scales and three aspect ratios).
[21] Here, the authors used the standard Faster R-CNN architecture.
[22] For instance, the authors exclude detection tasks on images containing many small objects [77].

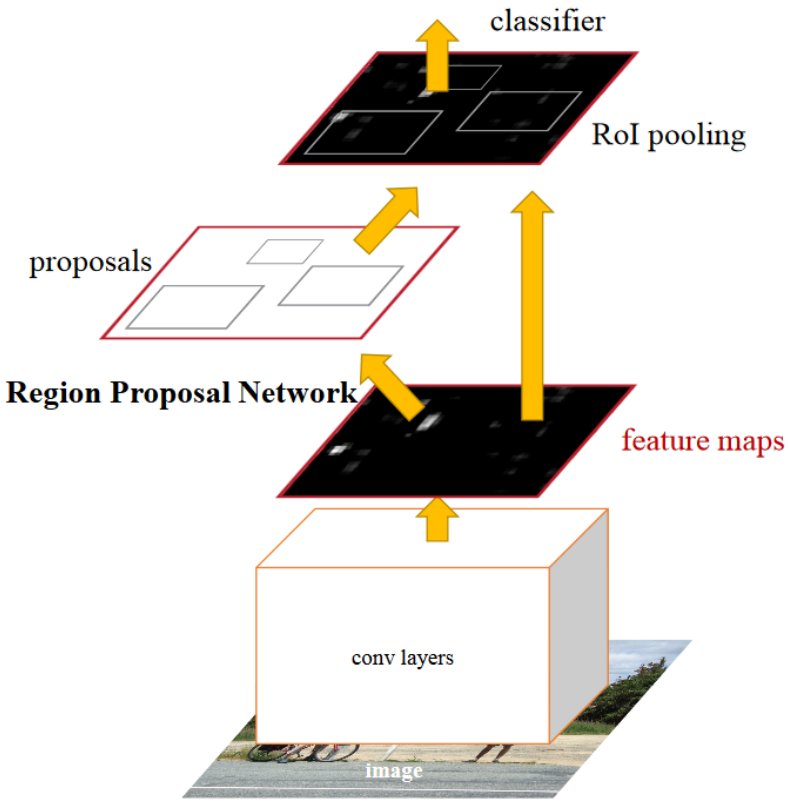


FIGURE 2.33: Intermediate Region Proposal Network (RPN) in Faster R-CNN. From [102]

to use a rather coarse fixed region proposal (RP) scheme[23] instead of generating region proposals. In particular, this approach relies on an appropriate bounding box regression step because of the fixed class-agnostic RPs. Their interpretation is that the CNNs already incorporate geometric information of the objects in the intermediate layers. In their experiments, the authors used 3 000 fixed RPs obtained from the statistics of the ground truth bounding box annotations in the training set. Using fixed RPs, they were able to show similar results as the simple comparison network with explicit RP generation [64] on the PASCAL VOC 2007 challenge.

2.3.2 One-Stage Detectors

One-stage or single-shot detectors are convolutional neural networks that are directly applied to the whole input image, generating output bounding boxes and multi-class scores per bounding box at the same time. This drastically improves detection speed enabling real-time multi-class object detection, however, mostly at the cost of (slightly) lower detection accuracy.

2.3.2.1 You Only Look Once (YOLO)

Redmon et al. proposed the first successful one-stage detector called You Only Look Once (YOLO) [99]. On an abstract level, they suggested seeing object detection as a

[23] I.e., a fixed scheme to systematically cover the complete image such that no region proposals have to be computed (see Sect. 5.5.2.1).

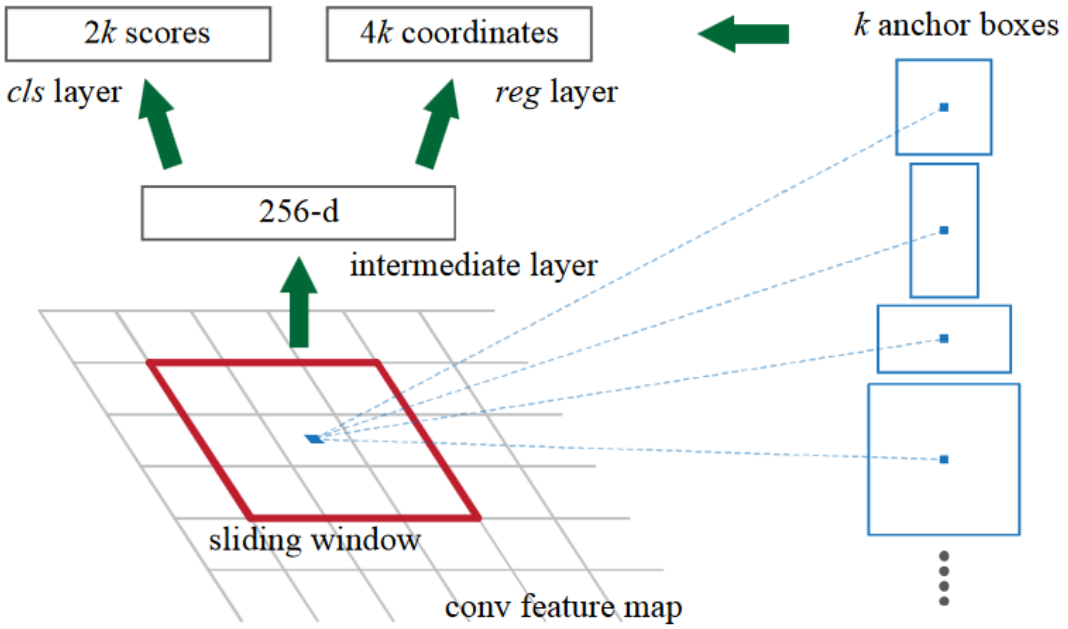


FIGURE 2.34: Architecture of the Region Proposal Network (RPN) in Faster R-CNN containing a classification layer (`cls`) that outputs object/non-object probabilities for each of the k anchor boxes and a regression layer (`reg`) that outputs the 2D center position as well as the aspect ratio and the scale for each anchor. From [102]

regression problem on both spatially separated bounding boxes and corresponding class probabilities. This way, the network can be trained end-to-end directly on the detection performance.

In practice, they rescale the input image to a fixed (quadratic) size and divide it into an uniform grid of $S \times S$ cells. Each cell then predicts B bounding boxes each of these with four relative coordinates and a general confidence along with a set C of class probabilities, i.e., $B * 5 + C$ output values per cell. In their experiments, they set $S = 7$, $B = 2$ and $C = 20$ leading to a $7 \times 7 \times 30$ output tensor.

From this output tensor, the final detections are decoded as follows: First, a global threshold on the general confidence is applied in combination with a non-maximum suppression (NMS) such that at most one detection per grid is generated. For each remaining and, thus, final detection, the class is determined by the maximum probability in the set C and the bounding box is converted from relative into absolute coordinates simply by using the size of the original input image (see Fig. 2.35).

The convolutional layers in the YOLO architecture are initially pre-trained at half the final resolution on the ImageNet database [24]. For training the detection layers, the pre-trained layers are upscaled and then four convolutional layers and two fully-connected layers are added initialized with random weights. The whole network is then optimized for the sum of squared errors slightly modified to account for several imbalances, e.g., regarding the bounding box size and the number of "empty" grid cells vs. those that contain an object.

Using this architecture, the authors reported the first state-of-the-art real-time detector on the PASCAL VOC challenge running at 45 frames per second (FPS) while in

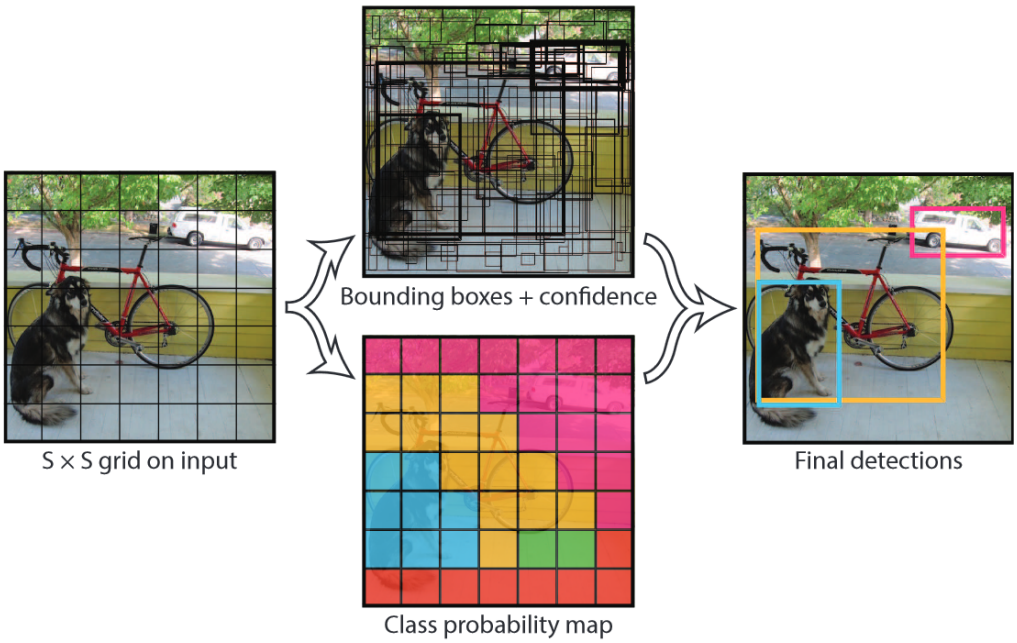


FIGURE 2.35: The one-stage YOLO object detection approach with illustrations of the grid concept as well as the predicted bounding boxes and scores per grid cell. From [99]

comparison achieving approximately 10% less mean average precision (mAP) than Faster R-CNN [102] that runs at 7 FPS.

In a follow-up work, Redmon and Farhadi suggested further improvements leading to YOLOv2 [100]. The authors revealed that YOLO suffers from a lower recall as well as more localization errors as compared to two-stage detectors, namely Faster R-CNN (see Sect. 2.3.1.3). To overcome this, the following improvements are proposed:

- Using batch normalization
- Pre-training the network on the full resolution
- Using anchor boxes as in Faster R-CNN – instead of directly predicting locations – leading to a higher recall
- Generating anchor boxes from training data by clustering using an intersection over union (IoU) criterion
- Using a 13×13 grid for finer-grained features
- Adding a pass-through layer that adds features from an earlier 26×26 layer, i.e., two feature maps are used for detection
- Training at multiple scales, i.e., after every 10 batches a random input size of the set $\{320, 352, \ldots, 608\}$ (multiples of 32) is chosen, the network is scaled accordingly and training is continued in order to capture different levels of detail
- Using Darknet-19 which only requires 5.58 instead 8.52 billion floating point operations (FLOPs) as in YOLO (Darknet-16)

With these improvements, the YOLOv2 approach is still (early 2018) state-of-the-art on the PASCAL VOC 2007 challenge achieving 78.6 mAP at 40 FPS as compared to SSD500 (see Sect. 2.3.2.2) with 76.8 mAP at 19 FPS on a NVIDIA Titan X.

2.3.2.2 Single-Shot MultiBox Detector (SSD)

Liu et al. proposed another popular one-stage detector called Single-Shot MultiBox Detector (SSD) [86]. The authors adopted the grid-based concept of YOLO ([99, 100], see Sect. 2.3.2.1) and the feature map as well as the anchor boxes from Faster R-CNN ([102], see Sect. 2.3.1.3) and combined them into a fast and accurate one-stage detector.

VGG16 [117] (see Sect. 2.3.1.2.1), a high-quality network for image classification, is used as the base network. The output layer of the standard VGG16 is removed and the fully-connected layers are converted into convolutional layers. In addition, further[24] convolutional feature layers of decreasing size are added which allow for detections at different resolutions. Each feature map has a fixed grid size corresponding to the size of the feature map. For each cell, a set of k (set to six) "default" bounding boxes is associated with respect to the corresponding feature map size following the ideas of anchor boxes of the region proposal network in Faster R-CNN (see Fig. 2.36 for an illustration).

For each of these anchor boxes at each grid cell in each feature map, c class scores and 4 offset values are predicted. As for most approaches, a final non-maximum suppression step in conjunction with a confidence threshold is applied to obtain the final detections.

Training can be executed in an end-to-end fashion using slight modifications of the strategy to match default boxes to ground truth annotations as well as the training objective of MultiBox [33]. The general aim is to optimize only those predictions that best match the ground truth annotations by considering both the assignment of prediction to ground truth as well as the corresponding confidence. For the best matches, the locations of the predictions are optimized such that the matching is improved and the confidence is maximized. The confidences of all remaining predictions are minimized. The SSD training also accounts for choosing the k anchor boxes as well as for hard negative mining, where only those negative examples with the highest confidence loss are chosen to achieve a ratio of 3:1 (negative:positive).

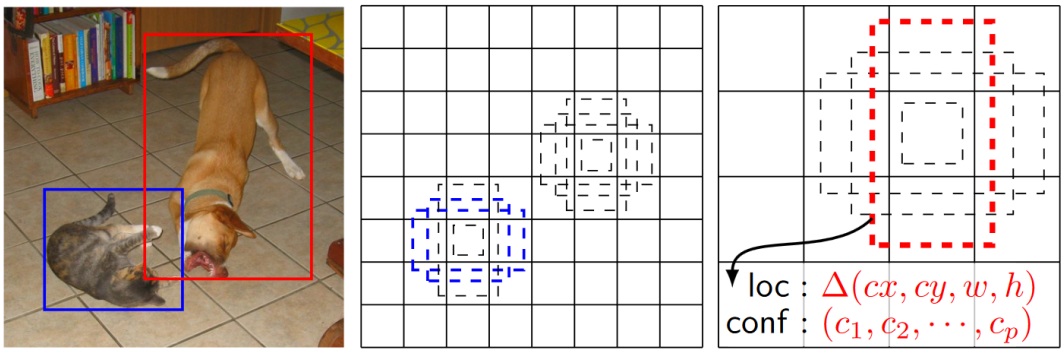
At publication time (2016), SSD with an input size of 512×512 was able to outperform YOLOv1 by more than 10% mAP on PASCAL VOC 2007 while running nearly at the same speed (19 vs. 21 FPS).

2.3.2.3 RetinaNet

Lin et al. addressed the question whether it is possible to combine the simplicity and speed of one-stage detectors such as YOLO ([99, 100], see Sect. 2.3.2.1) and SSD ([86], see Sect. 2.3.2.2), while achieving the high detection accuracy of two-stage approaches as Faster R-CNN ([102], see Sect. 2.3.1.3) or FPN [82].

As the main reason, the authors discovered the extraordinary class imbalance of fore- and background examples, when training one-stage detectors. To overcome

[24] The authors used four additional convolutional feature layers.



(a) Image with GT boxes (b) 8×8 feature map (c) 4×4 feature map

FIGURE 2.36: Illustration of differently sized feature maps and the corresponding anchor boxes in the Single-Shot MultiBox Detector (SSD). From [86]

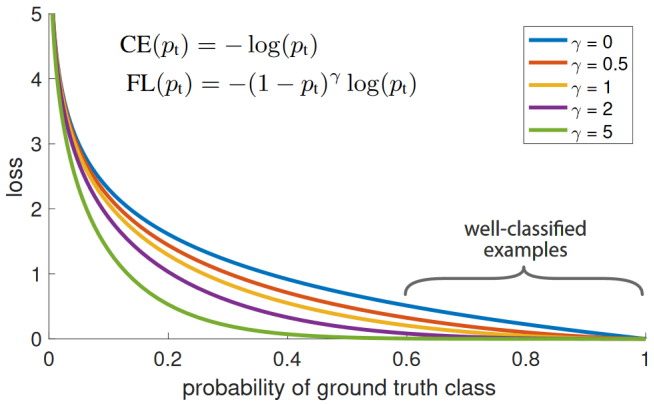


FIGURE 2.37: Comparison of cross-entropy loss (CE) and Focal Loss (FL) with different scaling factors γ. From [83]

this, they propose to use a novel loss type, the so-called Focal Loss, and proved its benefit by suggesting a one-stage detector called RetinaNet [83].

Two-stage detectors often solve the problem of class imbalance during training implicitly. One the one hand, most easy negative samples are already filtered out by the proposal generation stage. On the other hand, often either a fixed foreground-background ratio or hard negative mining is used when training the classification stage.

On the contrary, the training procedure of one-stage detectors produces very dense samples per training image, of which the majority is negative leading to an extreme class imbalance. Instead of applying standard techniques such as hard negative mining or bootstrapping, Lin et al. introduced the Focal Loss (FL) function:

$$FL(p_t) = -\alpha_t(1-p_t)^{\gamma}\log(p_t) \tag{2.9}$$

The Focal Loss is based on a balanced cross-entropy loss, i.e., using class priors α_t for weighting, but is dynamically scaled based on the confidence in the correct class. Basically, the scaling factor γ manages to down-weight the contribution of easily classified samples leading to an intrinsic focus on hard examples (see Fig. 2.37).

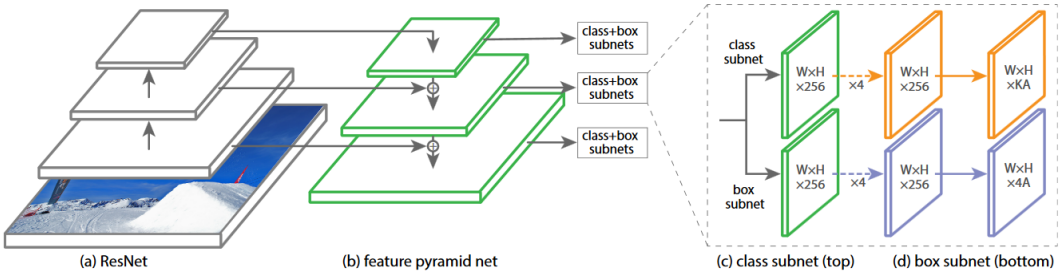


FIGURE 2.38: Architecture of RetinaNet. From [83]

For showing the effectiveness of the Focal Loss function, the authors propose RetinaNet, a one-stage detector, that consists of a backbone network responsible for computing the convolutional feature maps at different resolutions on the whole input image and two simple subnetworks that are used for classification and bounding box regression, respectively. The architecture is outlined in Fig. 2.38. As the backbone, they use an FPN [82] using the ResNet101 architecture [62]. Afterwards, A anchor boxes as in a RPN [102] are used that are passed at the same time to a classification and a bounding box regression subnetwork, respectively. Each of the subnetworks consist only of four convolutional layers with C 3×3 filters, where C is the number of channels of the input feature layer, and an output convolutional layer with KA filters[25] and sigmoid activation. Those subnet outputs that exceed the detection threshold of 0.05 (at most the top-scoring $1\,000$ proposals) are merged. A subsequent NMS is applied to form the final detections.

Using different numbers of layers and input image resolutions, the authors reported state-of-the-art results as compared to the best-performing one-stage (32.5% mAP at 73ms (RetinaNet-50-500) vs. 28.0% mAP at 61ms (SSD [86], see Sect. 2.3.2.2)) and two-stage detectors (37.8% mAP at 198ms (RetinaNet-101-800) vs. 36.2% mAP at 172ms (Feature Pyramid Networks (FPN) with Faster R-CNN architecture [82] as introduced in Sect. 2.3.1.3.1)) achieving a similar speed, respectively.

[25]The number of anchor boxes A is set to 9, the number of channels C is 256 and K corresponds to the number of classes.

Chapter 3

Databases

3.1 Requirements

To evaluate the object detection accuracy of the DGHT object detection approach, the following requirements are specified:

First, it is decided to use independent 2D images instead of video sequences since no tracking algorithms are applied in the DGHT object detection system. Thus, a direct comparison of the resulting approach to systems using tracking algorithms is not possible. Alternatively, if the video sequence would be treated as independent frames, the evaluation results would be largely influenced by the duration for which the same object appears in the video sequence, and thus would be biased and of limited value.

Second, this thesis mainly focuses on handling object size and background variability as well as multiple objects. In order to specifically evaluate the object detection performance as well as strengths and limitations of the DGHT object detection approach under these conditions, scenarios exhibiting large variability regarding unusual body poses or truncations shall be excluded.

Since the IAIR [133] corpus (Sects. 3.2.1 and 3.3.1) ideally fulfills the stated requirements, the main experiments are conducted on this data set (see Sect. 5.6). Additional evaluations are performed on other (smaller) corpora described in Sects. 3.2.2 and 3.2.3 as well as 3.3.2 and 3.3.3 for pedestrians and cars, respectively.

3.2 Pedestrian Detection

3.2.1 IAIR Pedestrian

FIGURE 3.1: Example images of the IAIR Pedestrian data set.

As explained above, most experiments are performed on the IAIR [133] database (see Fig. 3.1), because it contains a reasonable amount of independent 2D images and additionally offers difficulty labels (e.g., occlusion, low contrast) for each annotation. This additional information is also used for the detailed error analysis. As suggested

by Wu et al. [133], training is performed on a random 50%-split of the available pedestrian images, i.e. in total 1 046 images containing 2 341 pedestrians with an object height range from 45 to 383px (mean height: 160px). The remaining 1 046 images (2 367 pedestrians with a similar object height range and mean height) are used for evaluation. Training and test corpus each contain all types of difficulties present in the IAIR corpus.

3.2.1.1 IAIR Pedestrian Subset

In first experiments, a subset of the IAIR Pedestrian corpus was used where the object height range was restricted. Here, the data set was filtered for images containing simple pedestrians, i.e., no difficulty label is assigned to these instances, and then the mean height was computed (150px). In order to include size variability to a moderate extent, a pedestrian height range of approximately 25% of the mean height was chosen. This leads to a pedestrian height range of 130 to 170px. All images containing pedestrians of a size within this range are kept and images containing only smaller ones are discarded. Images with larger pedestrians were downscaled by a random factor such that the scaled pedestrian height falls into the specified range. Following this procedure, 457 images were obtained of which the first 300 were used as the training set and the remaining 157 images were used for evaluation. In this test set, only pedestrians within the size range remain annotated.

3.2.2 INRIA Person

FIGURE 3.2: Example images of the INRIA Person data set.

The resulting approach is also evaluated on the well-known INRIA Person [23] database (see Fig. 3.2). The test set contains 288 images which contain 561 annotated persons with a height range from 100 - 788px (mean height: 299px).

3.2.3 TUD Pedestrians

FIGURE 3.3: Example images of the TUD Pedestrians database.

Moreover, the framework is applied to the TUD Pedestrians [5] data set (see Fig. 3.3). The test set consists of 250 images containing 311 annotated pedestrians with a height range from 71 to 366px (mean height: 213px).

3.2.4 Penn-Fudan Database for Pedestrian Detection and Segmentation

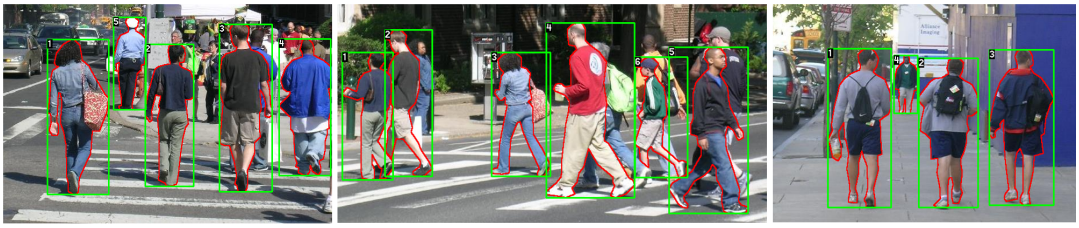


FIGURE 3.4: Example images of the Penn-Fudan data set and the corresponding annotations.

To train the Structured Edge Detector (see Sect. 2.1.2.2), the Penn-Fudan [125] data set is used, because in addition to bounding boxes labeled masks are provided as annotations, which can be used in order to extract outlines (see Fig. 3.4). The data set consists of 170 images containing 315 annotated upright pedestrians with a height range from 180 to 390px (mean height: 264px).

3.3 Car Detection

3.3.1 IAIR Car

FIGURE 3.5: Example images of the IAIR Car data set.

Also for car detection, most experiments are performed on the IAIR [133] database for the same reasons (see Fig. 3.5). As suggested by Wu et al. [133], training is performed on a random 50%-split of the available car images, i.e. in total 852 images containing 1 640 cars with an object height range from 16 to 363px (mean height: 88px), and object width range from 30 to 511px (mean width: 146px) and therefore an aspect range of 0.5 to 3.8 (mean aspect: 1.68; additional variability type). The remaining 852 images (1 652 cars with a similar object height, width and aspect range and mean values) are used for evaluation. Training and test corpus each contain all types of difficulties present in the IAIR corpus.

3.3.2 UIUC Image Database for Car Detection – Single-scale

The approach is also evaluated on the UIUC Image Database for Car Detection [2]. The training set consists of 550 positive (centered cars with an object size of approximately 85×30px) and 500 negative (random background structures, other objects etc.) image patches with a patch size of 100×40px (for training images see Fig. 3.6).

The single scale test set consists of 170 images which contain 200 annotated cars with an object height of roughly 30px and an object width of roughly 80px (see Fig.

FIGURE 3.6: Example training images of the UIUC Image Database for Car Detection. Top row: Positive samples; Bottom row: Negative samples.

FIGURE 3.7: Example test images of the UIUC Image Database for Car Detection.

3.7). This test corpus only contains side views of cars, therefore it rarely has aspect variability.

3.3.3 UIUC Image Database for Car Detection – Multi-scale

The multi scale test set of the UIUC Image Database for Car Detection [2], which is more difficult than the single scale test set, consists of 108 images which contain 139 annotated cars with an object height range from 24 to 66px (mean height: 45px) and an object width range from 74 to 198px (mean width: 135px). This test corpus only contains side views of cars, therefore it rarely has aspect variability.

3.3.4 Caltech-101: *car_side*

FIGURE 3.8: Example images of the category *car_side* of the Caltech-101 database. The fourth image illustrates the available bounding box and contour annotation for each ground truth instance.

Similar to the pedestrian databases, the *car_side* category of the Caltech-101 [35] data set (see Fig. 3.8) is used in order to train the Structured Edge Detector (see Sect. 2.1.2.2). The data set consists of 123 images each containing one car in the image center with an object size of 180×60px.

Chapter 4

Baseline System

This chapter describes the baseline system that was available and has already been used for automatic object localization in digital images: It has been successfully applied in the context of medical image processing, e.g., for the localization of epiphyses [55], subsequent bone age assessment [54] or the localization of knee joints [109]. Moreover, it also performed well on an iris localization task [56]. The object localization[1] system mainly consists of the Discriminative Generalized Hough Transform (DGHT, see Sect. 4.2) that uses Canny edge images (see Sects. 2.1.2.1 and 4.1) as input features and, optionally, a post-processing step called Shape Consistency Measure (SCM, see Sect. 4.3). The individual components are described in more detail in the following sections.

4.1 Canny Edge Detection

As already introduced, Canny edge images as described in Sect. 2.1.2.1 are used as standard input features to the Discriminative Generalized Hough Transform (DGHT). Precisely, for the baseline system the gray scale input image $\mathbf{I} : \Omega \rightarrow \mathbb{R}$ is smoothed using a 5×5 Gaussian filter with $\sigma = 1$. The low and the high thresholds are not completely fixed but set database-specifically.

Using these parameters, the procedure as described in Sect. 2.1.2.1 leads to a binary edge image $\mathbf{I}_E : \Omega \rightarrow \{0, 1\}$; for an example edge image see Fig. 2.3 or 4.6.

4.2 Discriminative Generalized Hough Transform

Based on the Generalized Hough Transform (GHT) as described in Sect. 2.2.1.2, the Discriminative Generalized Hough Transform (DGHT) [107] extends the GHT (Eq. 2.7) by assigning individual model point weights $\lambda_j \in \mathbb{R}$:

$$\mathbf{H}^{\mathcal{M}}(\mathbf{c}, \mathbf{I}_E) = \sum_{j \in \mathcal{M}} \lambda_j f_j(\mathbf{c}, \mathbf{I}_E) \tag{4.1}$$

with

[1] For the exact terminology of localization and detection as used in this work, the reader is referred to Sect. 1.6

$$f_j(\mathbf{c}, \mathbf{I}_E) = \sum_{(\mathbf{e},\gamma(\mathbf{e}))\in\mathbf{I}_E} \begin{cases} \mathbf{I}_E(\mathbf{e}), & \text{if } \mathbf{c} = \lfloor(\mathbf{e} - \mathbf{x}_j)/\rho\rfloor \\ & \text{and } |\gamma(\mathbf{e}) - \varphi_j| < \Delta\phi \\ 0, & \text{otherwise} \end{cases} \tag{4.2}$$

as in the standard GHT (see Sect. 2.2.1.2).

For ensuring good localization quality, the model should yield a large number of votes at true object locations and only a small number of votes at locations of confusable objects (see Fig. 4.1; please compare in particular the shape model $\mathcal{M}$ as well as the resulting Hough space to Sect. 2.2.1.2 and Fig. 2.16). The DGHT achieves this by an iterative, discriminative training procedure illustrated in Fig. 4.2.

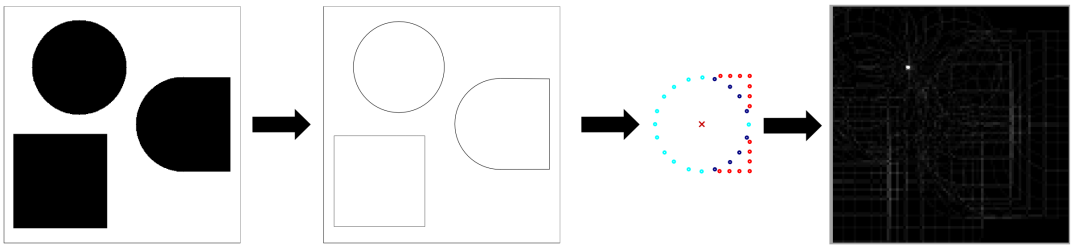


FIGURE 4.1: Discriminative Generalized Hough Transform (DGHT): Input image, binary edge image $\mathbf{I}_E$, DGHT shape model $\mathcal{M}$ with color-coded model point weights (blue: $+1$, cyan: $+0.5$, red: -1), resulting Hough space $\mathbf{H}^{\mathcal{M}}$ with one peak (from left to right). Based on [57]

To start with, an initial, so-called *a-priori* point model is created (1) by superimposing N^2 annotated (Canny) edge images at the reference point and keeping only those model points where an edge pixel was present at least k^3 times. Naturally, all *a-priori* model points are initialized with $\lambda = 1$. In each training iteration, the current model is applied to the set of training images that were not used for generating the *a-priori* model (2). Using these results, the individual model point weights λ_j are optimized by a Minimum Classification Error (MCE) approach (3), the so-called Discriminative Model Combination (DMC) [13][4]. To reduce model size, all model points with a low (absolute) weight are eliminated and, optionally, the M points exhibiting the highest absolute weights are selected (4). The optimized model is applied again to the current set of training images (5). To begin with the next iteration, the model is extended by target structures from training images which still have a high localization error (6).

This procedure is repeated until all training images are used or have a low localization error. Thus, the training process allows to automatically generate the shape model set $\mathcal{M}$. Further details on this technique can be found in Ruppertshofen [107].

Using the trained shape model $\mathcal{M}$, the edge image $\mathbf{I}_E$ of a test input image can then be transformed into a Hough space $\mathbf{H}^{\mathcal{M}}$ using Eq. 4.1. If the DGHT is the last processing step in the localization pipeline, the localization result is given by the highest-scoring hypothesis in $\mathbf{H}^{\mathcal{M}}$:

[2] N is set to 10.

[3] k is set to 3.

[4] The DMC was initially applied in the domain of speech recognition and then has been extended to work with the DGHT by Ruppertshofen [107], e.g., as an alternative to boosting.

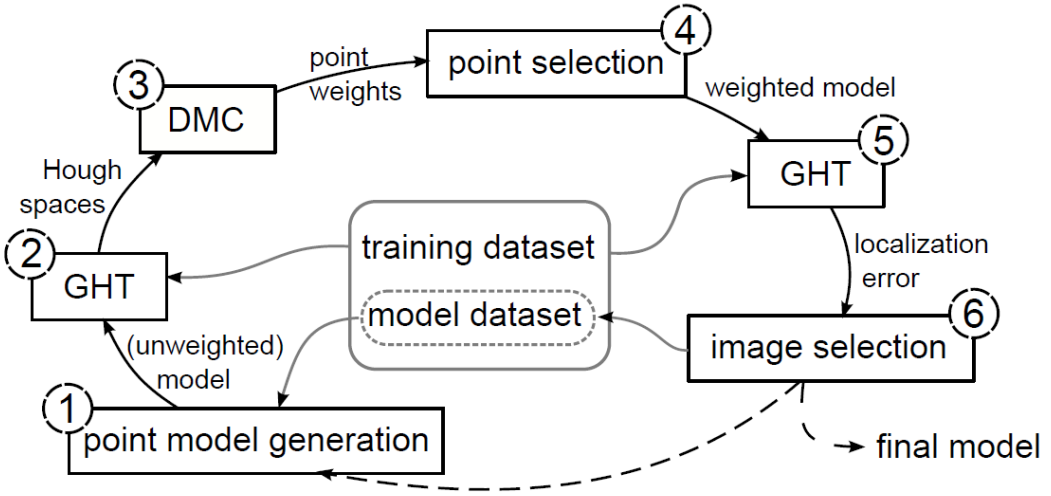


FIGURE 4.2: Training procedure of the Discriminative Generalized Hough Transform (DGHT). As step 5 already contains discriminatively trained model point weights, this can be seen as an application of the DGHT. From [107]

$$\hat{\mathbf{c}} = \arg\max_{\mathbf{c}\in\Omega} \mathbf{H}^{\mathcal{M}}(\mathbf{c}, \mathbf{I}_E) \tag{4.3}$$

4.3 Shape Consistency Measure (SCM)

Using the iterative training procedure described in Sect. 4.2, a DGHT model $\mathcal{M}$ may cover medium object variability (e.g., different object sizes or aspect ratios in general, different postures of pedestrians, side and frontal views of cars, etc.) by containing model points that represent the most important modes of variation observed in the training data. Due to the independent voting procedure (see Eq. 4.1), a Hough cell might get a large number of votes from different variability modes, for instance, from model points both of frontal and side views of a car, although these variability modes are mutually exclusive, which in consequence may lead to a mislocalization.

To this end, Hahmann et al.[56] suggested to analyze the model point voting pattern for a particular Hough cell $\hat{\mathbf{c}}_i$. More specifically, a Random Forest [15] is applied to classify the model point voting pattern into a class "regular shape" σ_r (representing, e.g., a frontal or a side view of a person) and a class "irregular shape" σ_i (e.g., random patterns). Please see Fig. 4.3 for an illustration of model point patterns.

To train the Random Forest Classifier, the final DGHT shape model $\mathcal{M}$ is applied to each training image, generating a Hough space $\mathbf{H}^{\mathcal{M}}(\mathbf{c}, \mathbf{I}_E)$ for each training image (where the dependence of the Hough space on the training image is dropped for notational simplicity). Then, examples for the SCM training are extracted from the individual Hough spaces as follows: All Hough cells with localization error $< \varepsilon_1$ are considered as positive examples, labeled σ_r; cells with an error $> \varepsilon_2$ are considered as negative examples, labeled σ_i. To avoid class imbalance, only the feature vectors of the top-K[5] hypotheses per class and training image are used for SCM

[5] K is set to 50.

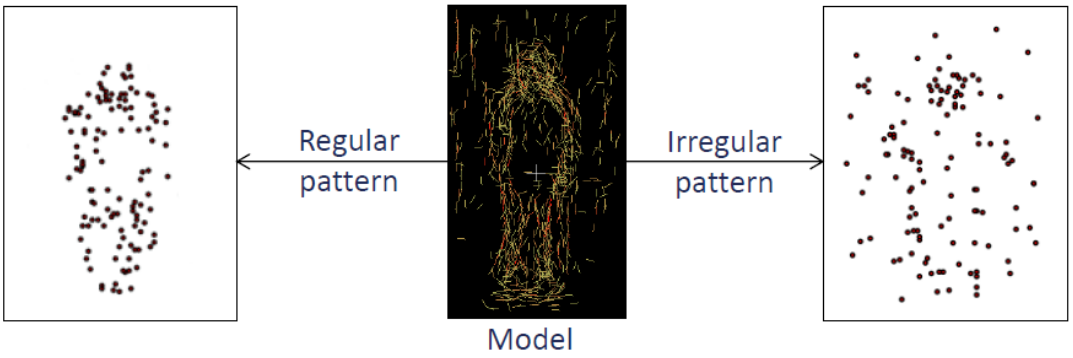


FIGURE 4.3: DGHT pedestrian model and two possible contributing subsets of model points that could represent the classes regular and irregular shape, respectively.

training. The feature vector $R(\mathbf{c}_i, \mathbf{I}_E)$ for each SCM training sample (corresponding to a Hough cell $\mathbf{c}_i$) should reflect which of the M model points contributed to the votes in cell $\mathbf{c}_i$. Thus, a M-dimensional feature vector

$$R(\mathbf{c}_i, \mathbf{I}_E) = \{r_1(\mathbf{c}_i, \mathbf{I}_E), r_2(\mathbf{c}_i, \mathbf{I}_E) \ \ldots, r_M(\mathbf{c}_i, \mathbf{I}_E)\} \tag{4.4}$$

is defined, the component r_j of which would – in its simplest form – be 0 if model point j did not vote for cell $\mathbf{c}_i$ and 1 if model point j voted for cell $\mathbf{c}_i$. Due to small localization errors and since the set of model points voting for a specific Hough cell is rather sparse, however, this requirement is too strict. Instead, each r_j should reflect whether model point j has voted "near" cell $\mathbf{c}_i$ (instead of exactly for $\mathbf{c}_i$) [56], i.e., inside of a $(2\vartheta + 1) \times (2\vartheta + 1)$ neighborhood around $\mathbf{c}_i$. Furthermore, instead of binary components $r_j \in \{0, 1\}$ the value r_j should reflect the 2D Chebyshev distance $d(\mathbf{c}_i, \mathbf{c}_k)$ between the location $\mathbf{c}_k$ of its vote and the considered Hough cell $\mathbf{c}_i$ with their respective coordinates $((\mathbf{c}_i)_x, (\mathbf{c}_i)_y)$ and $((\mathbf{c}_k)_x, (\mathbf{c}_k)_y)$

$$d(\mathbf{c}_i, \mathbf{c}_k) = \max(|(\mathbf{c}_k)_x - (\mathbf{c}_i)_x|, |(\mathbf{c}_k)_y - (\mathbf{c}_i)_y|). \tag{4.5}$$

Since too large distances are not meaningful, this distance is thresholded at ϑ[6]. Therefore in [56], the component $r_j(\mathbf{c}_i, \mathbf{I}_E)$ of the feature vector is defined as

$$r_j(\mathbf{c}_i, \mathbf{I}_E) = \min_{\mathbf{c}_k} \begin{cases} d(\mathbf{c}_i, \mathbf{c}_k), & \text{if } f_j(\mathbf{c}_k, \mathbf{I}_E) > 0 \\ & \text{and } d(\mathbf{c}_i, \mathbf{c}_k) < \vartheta \\ \vartheta, & \text{otherwise,} \end{cases} \tag{4.6}$$

where $f_j(\mathbf{c}_i, \mathbf{I}_E)$ are the accumulated votes from model point j as described in Eq. 2.6. Please see Fig. 4.4 for an illustration of different neighborhood sizes.

For a test edge image $\mathbf{I}_E$, a DGHT model $\mathcal{M}$ is applied to generate a Hough space $\mathbf{H}^{\mathcal{M}}$. For each local maximum $\hat{\mathbf{c}}_i$ in $\mathbf{H}^{\mathcal{M}}$, the Random Forest Classifier is used to calculate the probability $p_{\hat{\mathbf{c}}_i}(\sigma_r)$ that the set of model points voting for $\hat{\mathbf{c}}_i$ has a regular shape. The obtained probability is used as an additional weighting factor for

[6] ϑ is set to 5, if not otherwise specified.

the Hough space votes, i.e., $\mathbf{S}^{\mathcal{M}}(\hat{\mathbf{c}}_i, \mathbf{I}_E) = \mathbf{H}^{\mathcal{M}}(\hat{\mathbf{c}}_i, \mathbf{I}_E) \cdot p_{\hat{\mathbf{c}}_i}(\sigma_r)$. As in [56], the final localization result is now given by the highest-scoring hypothesis in $\mathbf{S}^{\mathcal{M}}$:

$$\hat{\mathbf{c}} = \arg\max_{\mathbf{c} \in \Omega} \mathbf{S}^{\mathcal{M}}(\mathbf{c}, \mathbf{I}_E) \tag{4.7}$$

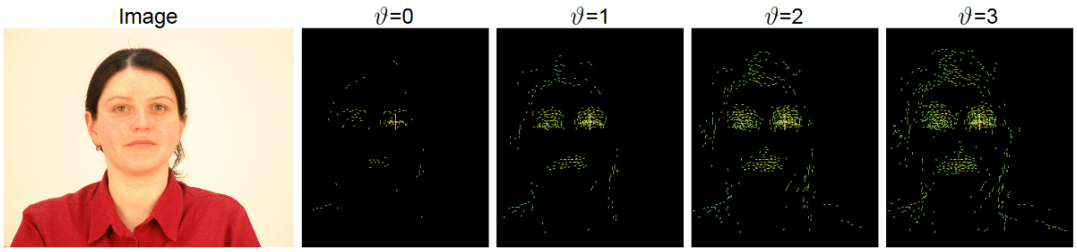


FIGURE 4.4: Contributing model points for different $(2\vartheta+1)\times(2\vartheta+1)$ neighborhood sizes as used in the feature vector construction in the SCM. From [56]

4.4 Detection Pipeline

Using the components described in the previous sections, an automatic object localization pipeline can be set up as outlined in Fig. 4.5. When using only the DGHT, the localization result is given by Eq. 4.3, i.e., the maximum of the Hough space $\mathbf{H}^{\mathcal{M}}$. If the SCM is applied as well, the localization result is given by Eq. 4.7. To obtain the final detection, a bounding box using the mean size of the objects in the training images is placed around the localization result with respect to the reference point. Fig. 4.6 illustrates the individual steps of object detection using the DGHT + SCM.

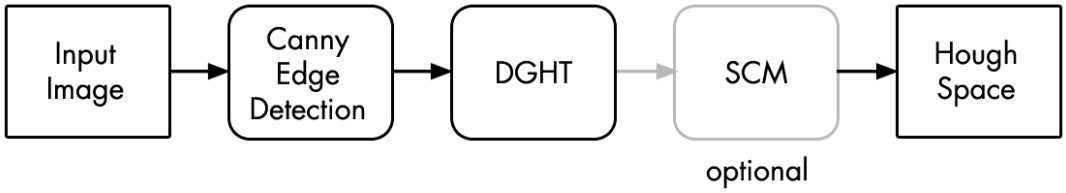


FIGURE 4.5: Components of the initial baseline localization pipeline

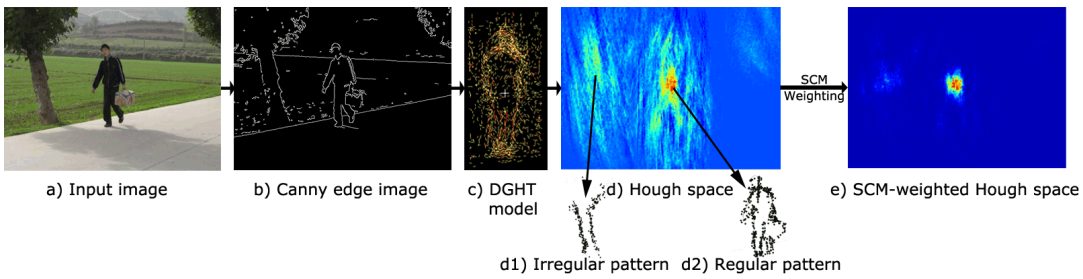


FIGURE 4.6: Example of the initial baseline detection pipeline. From [43]

4.5 Baseline Performance

4.5.1 Experiments

To assess the object detection performance of the baseline system and, in particular, to identify general and approach-specific problems or limitations as outlined in Sect. 1.2, the DGHT + SCM as described in this chapter has been applied to a simple pedestrian and car detection task on the IAIR Pedestrian Subset (see Sect. 3.2.1.1) and the UIUC Single-scale Image Database for Car Detection (see Sect. 3.3.2), respectively.

The annotated training images of the training sets of the respective pedestrian or car database are used to train two class-specific DGHT shape models using Canny edge detection and the iterative training procedure as described in Sect. 4.2. The quantization parameter ρ is set to 2 in x- and y-direction and $\Delta\phi$ to $\pi/8$ (16 bins, see [107]).

Subsequently, training samples in order to train the Shape Consistency Measure (SCM, see Sect. 4.3) need to be created. This is achieved by applying the respective pedestrian or car DGHT model to each training image of the corresponding training set. Afterwards, localization hypotheses below ε_1 set to 5 and 10 Hough cells (with quantization factor $\rho = 2$, i.e., 10 and 20px) for UIUC and IAIR, respectively, are extracted as samples for regular shapes (class σ_r). Localization hypotheses above ε_2 set to 15 and 25 Hough cells for UIUC and IAIR, respectively, are used as samples for irregular shapes (class σ_i, see Section 4.3) with ϑ set to 5.

When having applied either the DGHT model or both the DGHT model and the SCM to the respective test image set, the highest-scoring localization hypothesis $\hat{c}$ according to Eq. 4.3 or 4.7, respectively, is computed for each test image $\mathbf{I}_E$.

In addition to the standard localization result, a detection P, i.e., a bounding box of the respective mean object size represented by the DGHT shape model $\mathcal{M}$ centered around $\hat{c}$ and then transferred to image space, is generated leading to an object detection system.

4.5.2 Evaluation Metrics

When evaluating the highest-scoring hypothesis $\hat{\mathbf{c}}$, the **localization error** between the respective coordinates of $\hat{\mathbf{c}}$ transferred to image space (by multiplying each component with the used quantization factor ρ) and the ground truth annotation is a simple and easily interpretable performance metric. If there is more than one ground truth annotation present for the current test image, this metric is computed with respect to the closest ground truth object since the baseline system can only localize a single object instance per image.

In order to evaluate the correctness of a detection, i.e., a predicted bounding box P as described in Sect. 4.5.1, with respect to a ground truth bounding box G out of the set of ground truth annotations $\mathcal{GT}$, the very common **intersection over union (IoU)** measure[7] is used:

[7] The intersection over union (IoU) measure is also known as the Jaccard index or Jaccard similarity coefficient.

$$\text{IoU}(P, G) = \frac{P \cap G}{P \cup G} \tag{4.8}$$

A detection is correct, if the IoU exceeds 0.5 [34, 29] (see Fig. 4.7 for example IoU scores). The task-specific single-object accuracy (short "**accuracy**"), which is used to evaluate the different input edge features to the DGHT on the complete test sets, is defined as the number of test images which are correct ($\max_{\forall G \in \mathcal{GT}} \text{IoU}(P, G) > 0.5$) divided by the total number of test images.

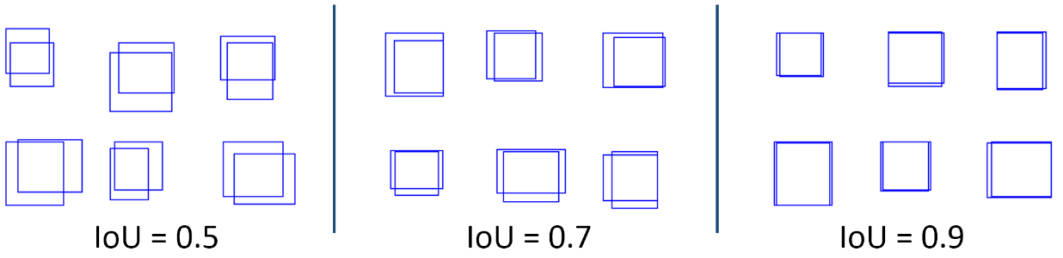


FIGURE 4.7: Example IoU scores. From [142]

For measuring the single-object detection quality, a slight single-object modification of the **Average Best Overlap (ABO)** score from Uijlings et al. [122] is used:

$$ABO_{single} = \frac{1}{|\mathcal{T}|} \sum_{\mathbf{I} \in \mathcal{T}} \max_{G_\mathbf{I} \in \mathcal{GT}_\mathbf{I}} \text{IoU}(P_\mathbf{I}, G_\mathbf{I}) \tag{4.9}$$

Here, the maximum IoU between the single-object detection $P_\mathbf{I}$ per image $\mathbf{I}$ and the corresponding set of ground truth annotations $\mathcal{GT}_\mathbf{I}$ is determined and averaged over all test images $\mathcal{T}$.

When the DGHT object detection framework is able to handle object size and aspect variability as well as the detection of multiple instances, multi-object evaluation metrics will be needed for a detailed analysis:

As suggested for single frame[8] evaluation of all corresponding detections in *Pedestrian Detection: Evaluation of the State of the Art* by Dollár et al. [29], **Detection Error Trade-off (DET) curves** are computed plotting the **miss rate** ($1-$ recall[9]) against the number of false positives per image (FPPI) on a log scale by modifying the rejection threshold θ (see Sect. 5.3.2).

Usually, the miss rates at 1 FPPI are used for comparison of the approaches. When using the final DGHT object detection pipeline, the miss rates are compared at 0.5 FPPI as this is – for some configurations of the pipeline – the highest FPPI rate achieved by the DGHT object detection pipeline (all other false positive candidates are rejected by the deep CNN classifier).

Miss rates on INRIA Person are shown at 1 FPPI. For the TUD Pedestrians and the UIUC Multi-scale database, the **recall at equal error rate (EER)** is used as the evaluation metric, as other groups have frequently used this measure when evaluating on these data sets.

[8] Here, i.e., a single test image with no temporal relation to other test images.
[9] Recall denotes the true positive rate, i.e., $\frac{TP}{TP+FN}$.

Additionally, for multi-scale and multi-object detections, the **Average Best Overlap (ABO)** score from Uijlings et al. [122] is used for measuring the candidate quality:

$$ABO = \frac{1}{|\mathcal{GT}|} \sum_{G \in \mathcal{GT}} \max_{\forall P} \text{IoU}(P, G) \tag{4.10}$$

Here, the maximum IoU between each ground truth bounding box annotation $G \in \mathcal{GT}$ and each prediction P generated from the candidate list C of the respective image is computed and averaged over all ground truth annotations in all test images.

4.5.3 Results

Tab. 4.1 and 4.2 show the results for the DGHT and DGHT+SCM baseline system applied to the test sets of the IAIR Pedestrian Subset (see Sect. 3.2.1.1) and the UIUC Single-scale Image Database for Car Detection (see Sect. 3.3.2), respectively, using the experimental setup as described in Sect. 4.5.1 and the single object metrics as reported in Sect. 4.5.2.

TABLE 4.1: Initial detection results of the DGHT applied to the test set of the IAIR Pedestrian Subset

Feature	Model	Mean Local. Error [px]	Accuracy [%]	ABO_{single} [%]
Canny	DGHT	33.56	79.62	63.40
	DGHT+SCM	20.07	85.35	64.19

TABLE 4.2: Initial detection results of the DGHT applied to the test set of the UIUC Image Database for Car Detection – Single-scale

Feature	Model	Mean Local. Error [px]	Accuracy [%]	ABO_{single} [%]
Canny	DGHT	9.19	87.65	70.87
	DGHT+SCM	4.46	96.47	77.69

4.5.4 Analysis

For both pedestrian and car detection, a significant improvement with respect to the single-object accuracy as well as the mean localization error is reported when using the SCM. This confirms that analyzing the model point voting pattern can support downweighting Hough vote counts that have been accumulated by a set of model points that does not indicate a regular pattern as learned from training data.

A visualization of the impact of weighting each cell in a Hough space with the respective SCM probability for class σ_r is presented in Fig. 4.8 and 4.9 for pedestrian and car detection, respectively. Many Hough scores that originate from irregular model point voting patterns could successfully be reduced, potentially improving the detection performance (see each first example for pedestrian and car detection). In addition, the remaining peaks are often more focused and, thus, possibly leading to more precise detections as can be seen from the reduced mean localization error and improved ABO_{single} score.

Still, a lot of peaks remain at confusable structures either in the background or at other object classes (see each second example for pedestrian and car detection in

Fig 4.8 and 4.9, respectively). The single-object detection accuracy is significantly improved when using the SCM but is not sufficient to completely solve both constrained single-scale detection tasks. On the evaluated real-world detection tasks, the SCM by itself is not entirely able to both suppress high Hough scores from irregular voting patterns, i.e., handling large background variability, and also to allow for a limited object variability contained in the DGHT shape model.

To further improve the DGHT detection performance and in order to perform automatic multi-scale and multi-object detection, Chapter 5 describes conceptual extensions to the baseline system.

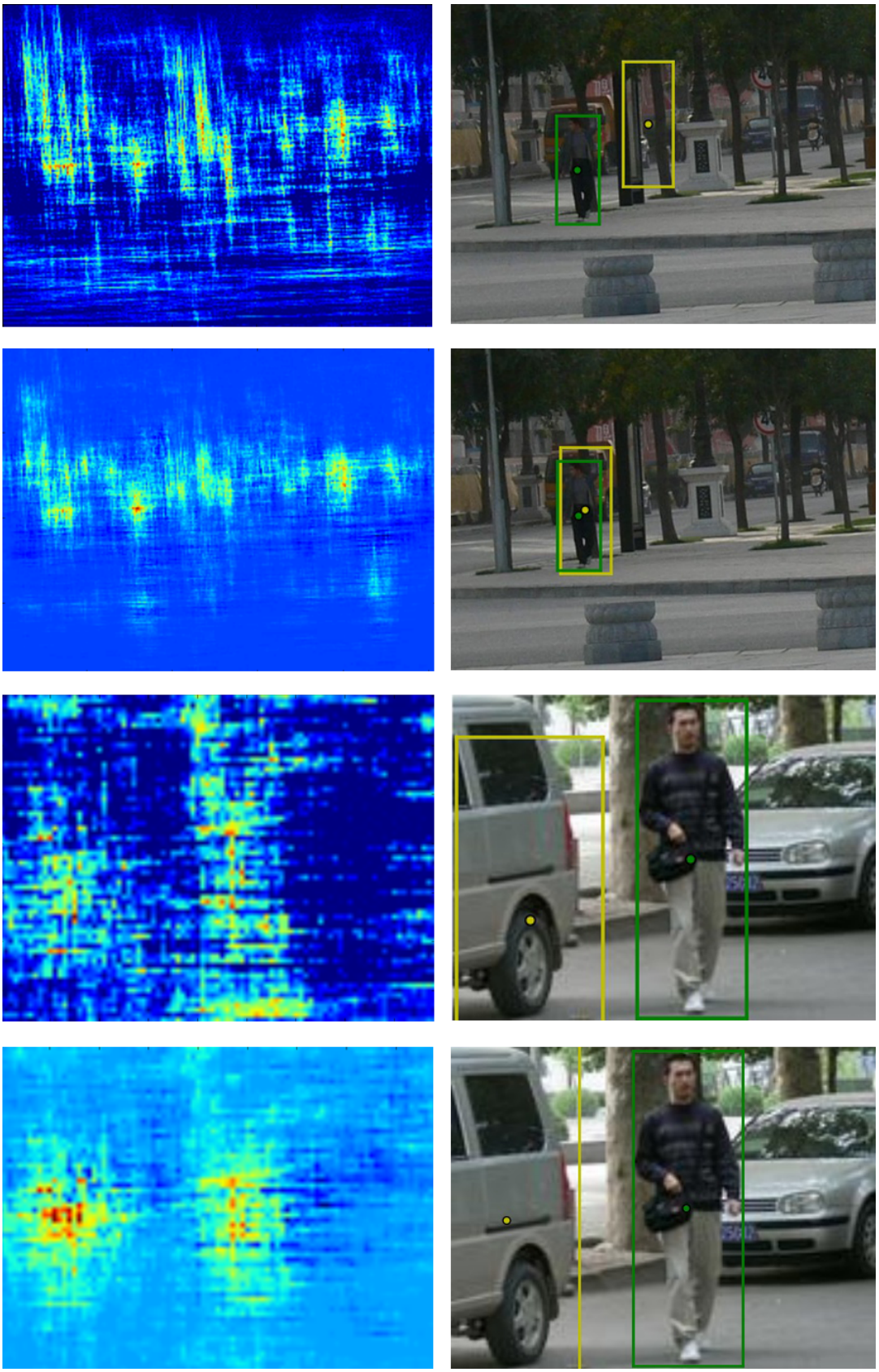


FIGURE 4.8: Example Hough spaces and corresponding detection results on the IAIR Pedestrian Subset. Odd rows: DGHT only, even rows: DGHT+SCM. (Left) Hough space (Right) Detection result; yellow: prediction, green: ground truth annotation. (Best viewed in color.)

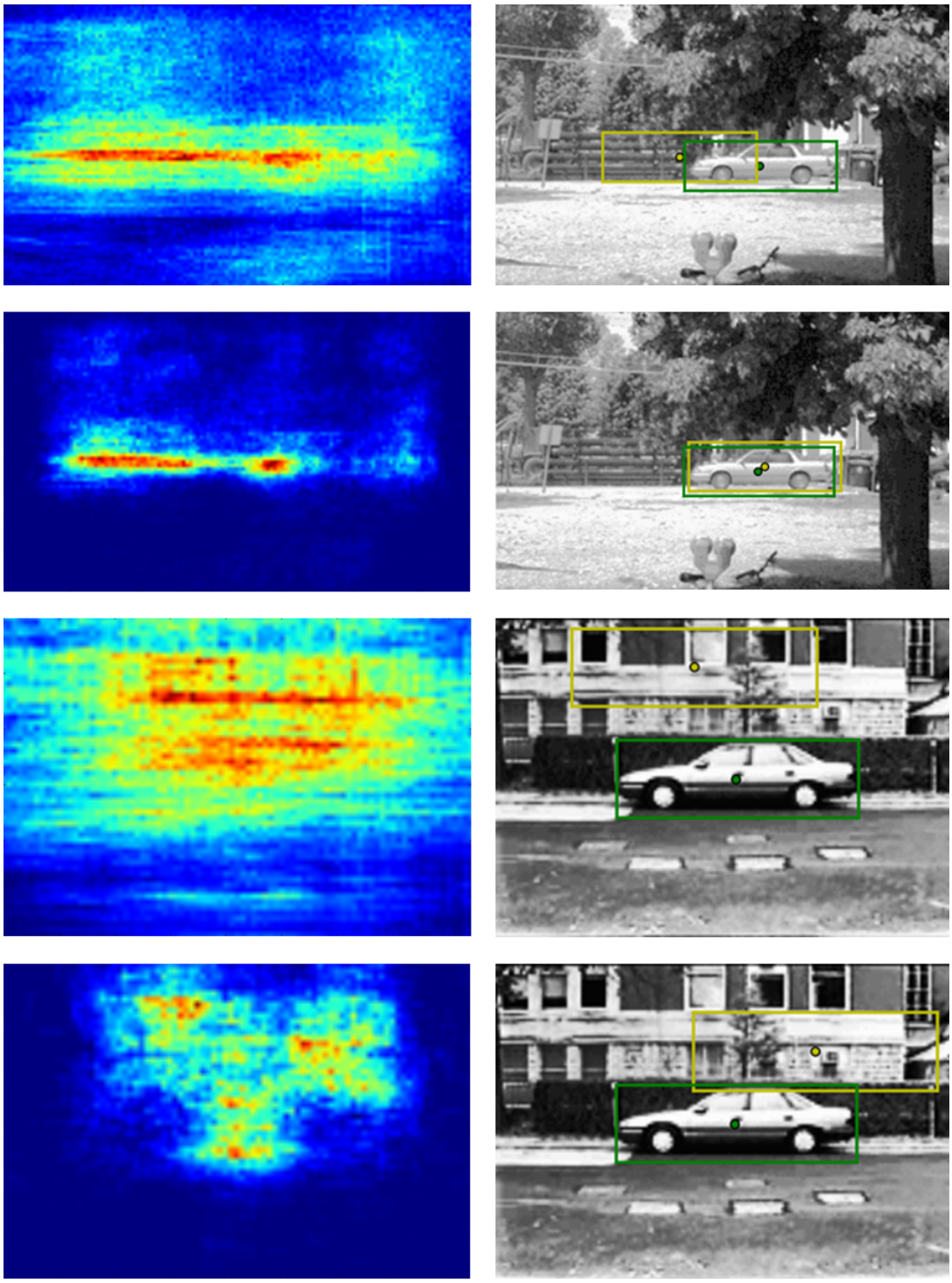


FIGURE 4.9: Example Hough spaces and corresponding detection results on the UIUC Image Database for Car Detection – Single-scale. Odd rows: DGHT only, even rows: DGHT+SCM. (Left) Hough space (Right) Detection result; yellow: prediction, green: ground truth annotation. (Best viewed in color.)

Chapter 5

System Extensions – Experiments and Results

This chapter describes the conceptual extensions that have been made to the baseline system (see Chapter 4) in order to overcome the problems which were faced when applying the DGHT to real-world object detection tasks as introduced in Sect. 1.2 and experimentally analyzed in Sect. 4.5. Additionally, this chapter contains descriptions of the experiments corresponding to each extension as well as the results of the extensive evaluations that were conducted to assess the gain in detection performance as measured by the evaluation metrics described in Sect. 4.5.2.

5.1 Handling Background Variability

Parts of this section have initially been published in [41] (see also Section 1.4). As already introduced in Sect. 1.2 and analyzed in Sect. 4.5.4, it was noticed that when applying the baseline system to real-world images, e.g., to pedestrian and car detection, many localization hypotheses occur at confusable structures in the background. In the previously investigated medical applications, most scans typically exhibit a dark background rarely containing confusable structures.

Thus, the intention of this extension is to reduce background variability by applying structured edge detection (Sect. 2.1.2.2) instead of Canny edge detection in order to improve the detection performance of the DGHT. Since the individual pixels in structured edge images reflect confidences ($\mathbf{I}_E : \Omega \rightarrow [0,1]$), this leads to non-binary contributions $f_j(\mathbf{c}, \mathbf{I}_E) \in [0,1]$ (see Eq. 4.2) in contrast to Canny edge images, where $f_j(\mathbf{c}, \mathbf{I}_E) \in \{0,1\}$. These non-binary contributions are then weighted by the model point weights λ_j as in Eq. 4.1. To assess whether this reduction is beneficial for the detection performance, the detection results when using structured edge images (see Sect. 2.1.2.2 and 5.1.1, respectively) as input features for the DGHT are compared to the detection results when using standard Canny edge images.

5.1.1 Structured Edge Detection

As described in Sect. 2.1.2.2, the approach of Dollár and Zitnick [30] can be used to transform an input image $\mathbf{I}$ to an output structured edge image $\mathbf{I}_E : \Omega \rightarrow [0,1]$. The algorithm of [30] involves supervised training, i.e., a training corpus with annotated target edges has to be provided. The annotated edges should represent all objects which should be separated from the background. In this thesis, a pedestrian edge

detector is trained by annotating the contours of all pedestrians in an image, and a separate car edge detector by annotating the contours of all cars. To this end, dedicated corpora containing pedestrian and car annotations are used, as explained in Sect. 5.1.3. Alternatively, a class-agnostic edge detector can be trained when using general edges, e.g., from segmentation data sets. For an example see Fig. 5.1 as in illustration of a predecessor of the current structured edge detector, the so-called Boosted Edge Learning (BEL) [25].

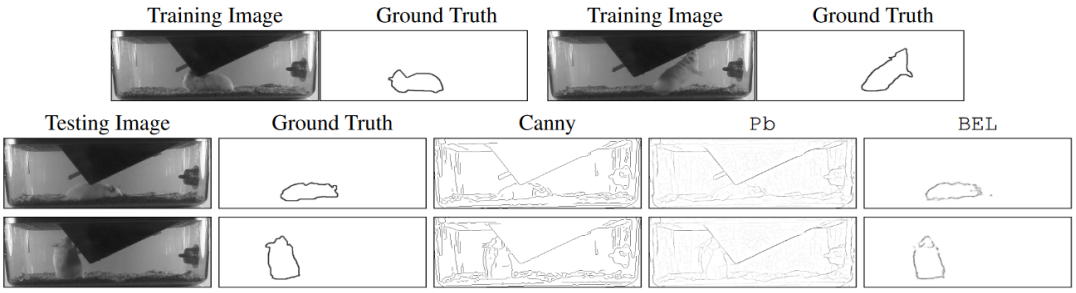


FIGURE 5.1: Boosted edge learning (BEL) results using class-specific annotations as compared to Canny (Sect. 2.1.2.1) and Pb [92] edge detection (which are class-agnostic). The top row shows two of fourteen training images and the respective ground truth annotations. The remaining two rows show the results of the different edge detectors on two of seven testing images as well as the ground truth. From [25]

5.1.2 Detection Pipeline

Fig. 5.2 shows the changes that were made to the DGHT detection pipeline. Specifically, Canny and structured edge images are compared as input for the DGHT and their effect on the detection performance is analyzed as described in Sect. 5.1.3.

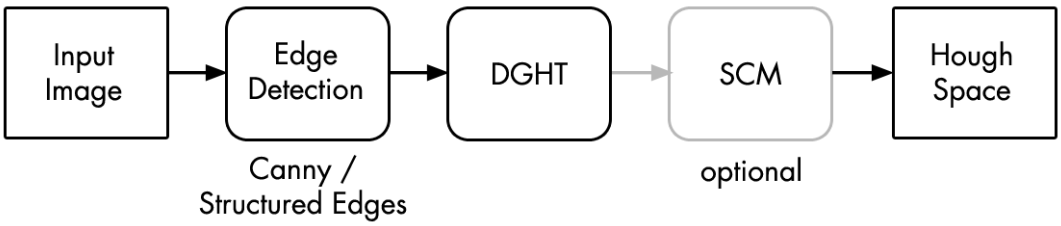


FIGURE 5.2: Components of the detection pipeline for handling background variability

5.1.3 Experiments

In order to test the influence of structured edge detection on reducing background variability on a pedestrian and a car detection task, the following experiments are conducted on the IAIR Pedestrian Subset (see Sect. 3.2.1.1) and on the UIUC Single Scale Database for Image Detection (see Sect. 3.3.2), respectively. Since object detection using the DGHT is currently restricted to single-scale, the experiments are conducted on the respective single-scale data (sub-)sets.

As the standard edge detector, the Canny Edge Detector is used as described in Sect. 2.1.2.1. Here, the low and high thresholds are specifically optimized for each detection task. This is done by qualitatively assessing the Canny edge images of the respective training images on a sample basis, searching for a trade-off between keeping essential edges of the target object and not having too much background clutter. For the UIUC Image Database for Car Detection (single-scale, see Sect. 3.3.2), a high threshold percentage of 0.9 and a low threshold percentage of 0.6 is determined. For the pedestrian detection task on the IAIR Pedestrian Subset (see Sect. 3.2.1.1), a high threshold percentage of 0.8 and a low threshold percentage of 0.5 is used.

As a second edge detector, the Structured Edge Detector is applied to the input images. For computing the structured edge images, publicly available code[1] that has been provided by the authors is used. As explained in Sects. 2.1.2.2 and 5.1.1, the Structured Edge Detector can be trained with domain-specific, annotated edge images. In this case, the *car_side* category of Caltech-101 database [35] and the Penn-Fudan data set [125] are used for cars and pedestrians, respectively[2]. In this manner, the class-specific Structured Edge Detector models are trained to highlight edges belonging to the respective object category and suppress background edges or those of non-target objects. The domain-specific Structured Edge Detector is in the following referred to as SSE.

For comparison, a general purpose Structured Edge Detector is applied that was provided by Dollár and Zitnick. This edge detector is trained on the general BSDS500 segmentation data set [7], which provides contour/boundary information, and is referred to as GSE.

The experiments for the different edge images on both data sets are in detail conducted as follows:

First, the different feature images (Canny, GSE and SSE) are generated for the training sets of both data sets, which serve as input images for the DGHT training. Then, a DGHT shape model $\mathcal{M}$ is trained for each feature type and data set, respectively, using the iterative training process according to Section 4.2. The quantization parameter ρ is set to 2 in x- and y-direction and $\Delta\phi$ to 16 for all experiments. Afterwards, the Shape Consistency Measure (SCM) is trained by applying the resulting DGHT model to each training image and extracting localization hypotheses below ${\varepsilon_1}^3$ as samples for σ_r and above ${\varepsilon_2}^4$ as samples for σ_i (see Section 4.3) with ϑ set to 5. After these training steps, either the DGHT model or both the DGHT model and the SCM are applied to the respective test image set, where the same edge detection algorithm as in training is applied to each test image. Then, for each test image $\mathbf{I}_E$ the best localization hypothesis $\hat{c}$ according to Eq. 4.3 or 4.7 is computed for DGHT and DGHT + SCM, respectively. To obtain the final detection P, a bounding box of the respective mean object size represented by the DGHT shape model $\mathcal{M}$ is centered around $\hat{c}$.

[1] `http://research.microsoft.com/en-us/downloads/389109f6-b4e8-404c-84bf%2D239f7cbf4e3d`, accessed: 2018-11-11

[2] These databases, and not the training sets of the UIUC and IAIR databases, are used for the training of the specific structured edge detectors, because the former already provide a ground truth contour annotation.

[3] 5 and 10 Hough cells (with quantization factor $\rho = 2$, i.e., 10 and 20px) for UIUC and IAIR, respectively.

[4] 15 and 25 Hough cells for UIUC and IAIR, respectively.

5.1.4 Results

This section presents the results of the experiments described in Sect. 5.1.3. In particular, Canny and structured edge images are compared as input features to the DGHT, and their impact on the single-object detection accuracy is quantitatively evaluated.

5.1.4.1 Pedestrian Detection

Table 5.1 shows the single-object detection results on the test set of the IAIR Pedestrian Subset database. One can see from the accuracy and the ABO_{single} score (see Eq. 4.9 and Fig. 4.7 for the metric and exemplary IoU scores, respectively) that using either general or class-specific structured edges improves the single-object detection performance by reducing the background variability and thus avoiding potential high-scoring peaks at these structures. Without the SCM, using GSE (Tab. 5.1, line 3) leads to a slightly better accuracy as well as ABO_{single} score as compared to using SSE as input features (Tab. 5.1, line 5). The reason for this is that in 12 of the 26 error cases, pedestrians outside the annotated size range ($130 - 170$px, see Sect. 3.2.1.1) are detected when using SSE. This leads to an IoU of 0% in each case. In contrast, when using GSE features, the majority of the false negatives is due to localization errors ($0 <$ IoU $< 50\%$) leading to a higher ABO_{single} score.

As an illustration, two example detections for each edge image input are presented in Fig. 5.3. Besides, Fig. 5.4 shows an example for detecting a pedestrian outside the annotated size range using SSE features without the SCM.

TABLE 5.1: Detection results on the IAIR Pedestrian Subset test set using different input features for the DGHT. Based on [41]

Feature	Model	Mean Local. Error [px]	Accuracy [%]	ABO_{single} [%]
Canny	DGHT	33.56	79.62	63.40
	DGHT+SCM	20.07	85.35	64.19
GSE[5]	DGHT	31.26	86.62	67.69
	DGHT+SCM	12.56	91.08	69.65
SSE[6]	DGHT	25.26	83.44	65.23
	DGHT+SCM	15.71	92.99	69.27

5.1.4.2 Car Detection

Table 5.2 shows the single-object detection results on the test set of the UIUC Single-scale database. The already high accuracy when using Canny edges as input features is perfect, i.e., 100% single-object detection accuracy, when using class-specific structured edge images as input features while increasing the ABO_{single} by more than 5%. Similar to the pedestrian detection task, two example detections for each edge image input are presented in Fig. 5.5.

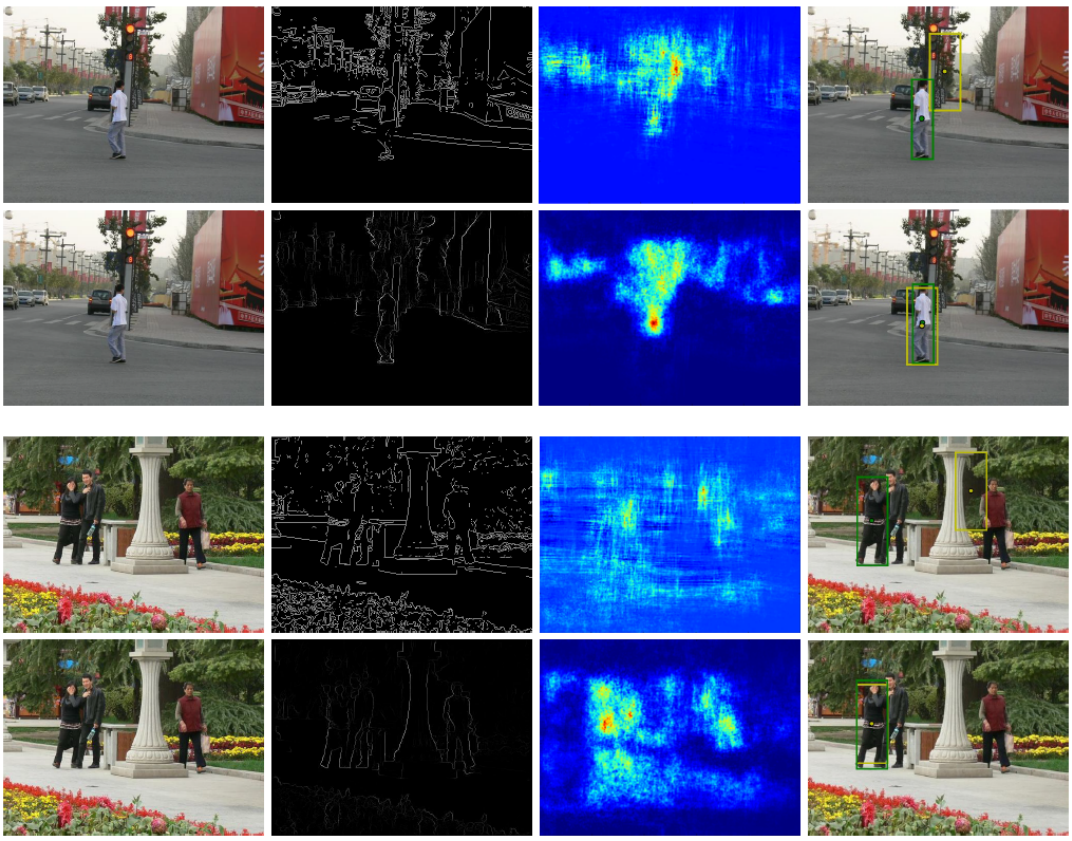


FIGURE 5.3: Example edge images and corresponding detection results on the IAIR Pedestrian Subset. Odd rows: Canny edge detection, even rows: Class-specific structured edge detection (SSE). (First column) Input image (Second column) Edge image (Third column) Resulting Hough space (Fourth column) Detection result; yellow: prediction, green: ground truth annotation. (Best viewed in color.) From [41]

5.1.4.3 Statistical Evaluation

To assess the statistical significance of the accuracy, a binomial distribution is assumed for the single-object detection accuracy per image (correctly detected versus not correctly detected, corresponding to IoU > 0.5 versus IoU $<= 0.5$, respectively), and the 95%-Clopper-Pearson confidence interval for the accuracy is computed (see Tab. 5.3). On both tasks, the experimentally achieved accuracy using SSE edges is beyond the confidence interval for the accuracy when using Canny edges as input features. This demonstrates a significant improvement in terms of an increase of the accuracy by using structured edges as compared to Canny edges. On the pedestrian detection task, this also holds when using GSE edges. To assess the statistical significance of the continuous overlap parameter IoU when using class-specific structured edges as input features, the non-parametric Wilcoxon signed-rank test [128] is used, since the parameter IoU is not normally distributed (p-values of all experiments in the Shapiro-Wilk tests [116] are $< 10e-3$). Comparing the independent groups Canny and SSE for both data sets in the one-sided Wilcoxon signed-rank test, p-values of $1.172e^{-6}$ and $1.656e^{-7}$ are obtained for the IAIR Pedestrian Subset and

[6] **G**eneral **S**tructured **E**dges
[6] Class-**s**pecific **S**tructured **E**dges

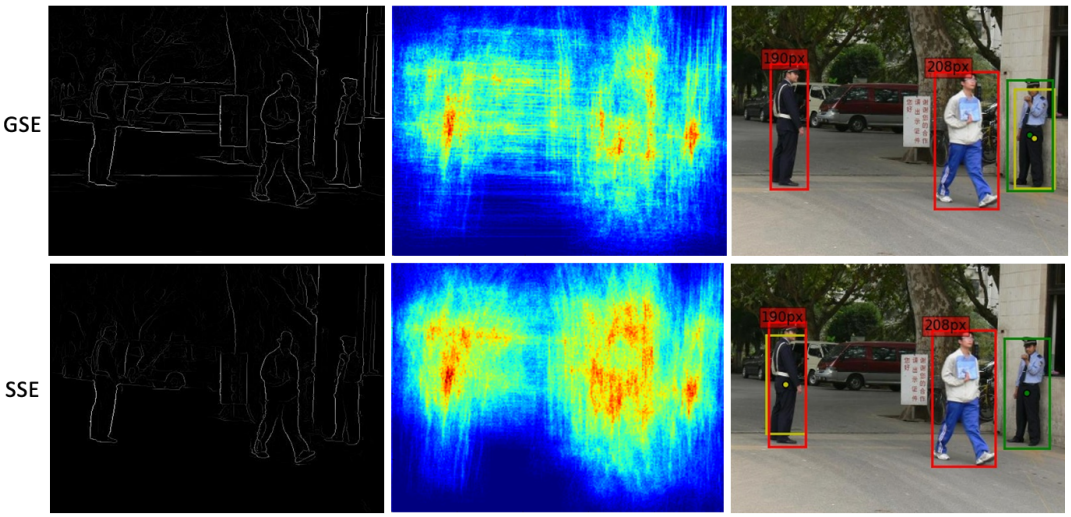


FIGURE 5.4: Detected pedestrian outside the annotated size range using SSE features on the IAIR Pedestrian Subset. Top row: general structured edges (GSE), bottom row: class-specific structured edges (SSE).
(First column) Edge image (Second column) Resulting Hough space (Third column) Detection result; yellow: prediction, green: ground truth annotation. (Best viewed in color.). From [41]

TABLE 5.2: Detection results on the UIUC Single-scale test set using different input features for the DGHT. Based on [41]

Feature	Model	Mean Local. Error [px]	Accuracy [%]	ABO_{single} [%]
Canny	DGHT	9.19	87.65	70.87
	DGHT+SCM	4.46	96.47	77.69
GSE	DGHT	7.26	92.35	75.15
	DGHT+SCM	4.39	97.65	79.72
SSE	DGHT	3.86	98.24	81.46
	DGHT+SCM	3.13	100.00	82.83

UIUC Single-scale, respectively. Thus, the ABO_{single} for the SSE edge detection tests is larger at the 95% confidence level than the ABO_{single} for the Canny Edge Detection tests. Additionally, the resulting localization errors for Canny and SSE edge features are statistically evaluated in the same way as described above, i.e., using a one-sided Wilcoxon signed-rank test. Here, p-values of $2.2e^{-16}$ and $9.113e^{-15}$ are obtained for the IAIR Pedestrian Subset and UIUC Single-scale, respectively. Therefore, the mean localization error for the SSE edge detection tests is lower at the 95% confidence level than the mean localization error for the Canny Edge Detection tests.

5.1.4.4 Conclusion

It has been shown that the single-object detection performance obtained by the voting-based DGHT approach in real-world tasks with variable background and clutter can be significantly improved by a sophisticated edge detection algorithm, namely the Structured Edge Detector. This applies to general structured edge features without additional training effort as well as class-specific structured edge detectors in particular. More precisely, absolute improvements in detection accuracy of 7.64%

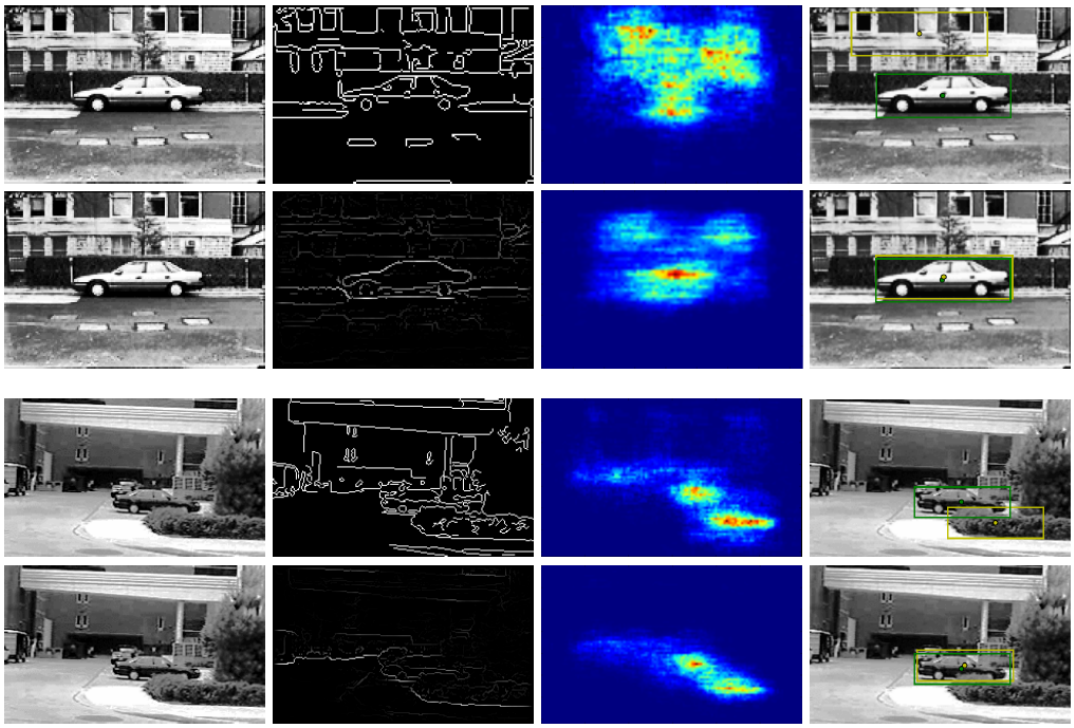


FIGURE 5.5: Example edge images and corresponding detection results on the UIUC Single-scale test set. Odd rows: Canny edge detection, even rows: Class-specific structured edge detection (SSE). (First column) Input image (Second column) Edge image (Third column) Resulting Hough space (Fourth column) Detection result; yellow: prediction, green: ground truth annotation. (Best viewed in color.) From [41]

and 3.53% have been reported on a pedestrian and car localization task, respectively. Intermediately, it is concluded that the DGHT framework can be successfully used for object detection also in real-world images with more variable and more confusable background.

This improvement removed a lot of votes coming from background structures leading to more defined peaks and less noise in the resulting Hough spaces. Thus, it also enabled the evaluation of identifying and assessing multiple peaks in the resulting Hough space for real-world scenarios. This will be used both across multiple scales of the input image or the shape model to handle object size variability as described in Sect. 5.2 and within each scale to detect multiple instances (see Sect. 5.3).

5.2 Handling Object Size Variability

The content of this section has initially been published in [44]. Up to this point, the used (single-scale) data sets and therefore also the DGHT shape models contained contained only a limited variability (up to 10%) with respect to the object size, i.e., for the test sets it is known at which size the object(s) can occur.

However, in general object detection tasks such as pedestrian detection, there is generally no prior information on the expected scale of the objects of interest, in contrast,

TABLE 5.3: Clopper-Pearson intervals for the single-object detection accuracy achieved using different input features for the DGHT. From [41]

Data set	Feature	Clopper-Pearson interval
IAIR Pedestrian Subset	Canny	[78.83%, 90.48%]
	SSE	[87.81%, 96.45%]
UIUC Single-scale	Canny	[92.48%, 98.69%]
	SSE	[97.85%, 100.00%]

e.g., to medical images. In particular, the objects might exhibit very large size variability.

To cope with this large variability regarding the size of objects to be detected in new test images, two alternative approaches are considered, which are described in the following subsections:

5.2.1 Image Scaling

This is a very traditional approach that has been used in several object detection frameworks [27, 37, 26]. Here, an image pyramid is defined by a fixed set of scaling factors (usually $8 - 16$ scales per octave[7]) to cover the expected object size range. Hence, the edge images have to be recomputed for each scaling factor as applied in Gabriel et al. [43]. Afterwards, the DGHT shape model $\mathcal{M}$ is (independently) applied to each scaled edge image, thus, generating a set of Hough spaces (one for each scaling factor, see Fig. 5.6 (a)).

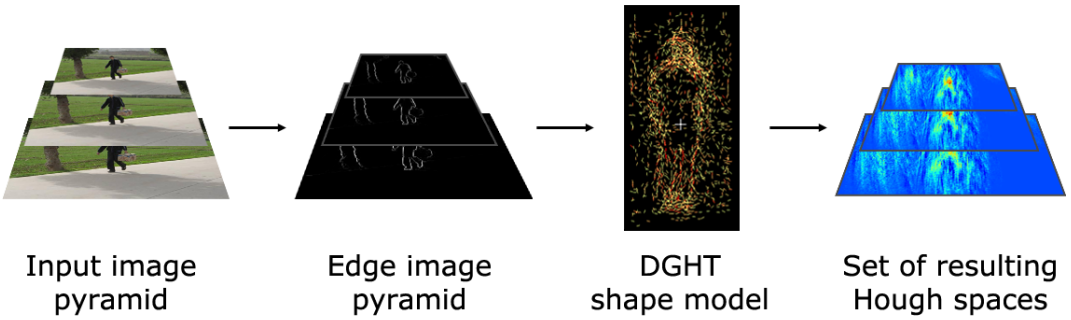
5.2.2 Model Scaling

In contrast, an alternative approach is suggested to handle the variability of object sizes by adopting the template pyramid as in Dollár et al. [27] and Ohn-Bar and Trivedi [97] to the DGHT object detection framework as presented by Gabriel in [44]. Here, the central idea is to handle object variability by applying a set of transformations, which cover the expected object variability, to the DGHT shape model $\mathcal{M}$. Since a large variability regarding object sizes should be handled, a set of simple scaling operations is applied to the set of model points $\mathbf{m}_j \in \mathcal{M}$, in particular, to the (local) coordinates $\mathbf{x}_j$. In general, other operations such as rotations are also possible, but not considered in this work. Here, the experiments are restricted to object size variability, represented by scaling transformations.

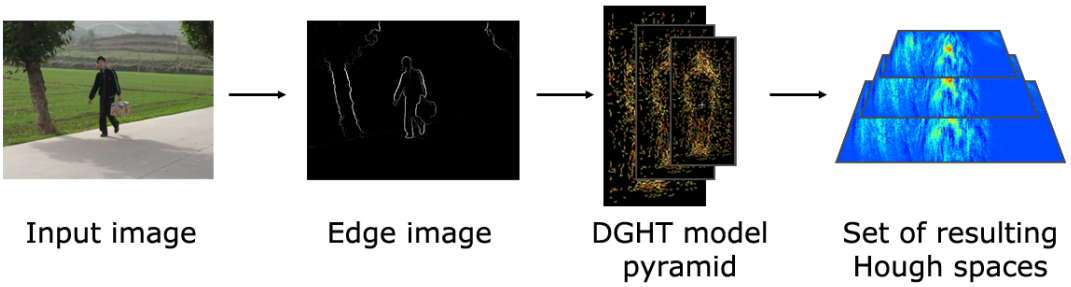
For simplicity, the same scaling factor for both the x- and the y-axis is used. In this model scaling scheme, the edge image has to be computed only once. Afterwards, the model pyramid is in parallel applied to the single input edge image, again generating a set of Hough spaces (one for each model scale, see Fig. 5.6 (b)).

An implicit limitation of this approach is that at very small or large scales model points could intersect or get too coarse, respectively, which might lead to mislocalizations.

[7] In an octave (depending on the direction), the image size is halved/doubled for each image dimension, i.e., the area of a 2D image is one fourth of or four times the initial image area, respectively.



(A) Image scaling



(B) Model scaling

FIGURE 5.6: Visualization of the image and model approach for handling size variability.

5.2.3 Detection Pipeline

The extended detection pipeline, which is able to handle object size variability, is shown with its variants in Fig. 5.7. Either Canny or structured edge images are used as input features for the DGHT. In case of image scaling, edge images are computed for an input image pyramid using fixed scaling factors. Subsequently, the DGHT shape model $\mathcal{M}$ is applied to the set of edge images, each time generating a corresponding Hough space. In case of model scaling, only a single edge image is used as in the baseline system (see Chapter 4). Afterwards, a set of DGHT shape models[8] is applied to the edge image, again, each time generating a corresponding Hough space.

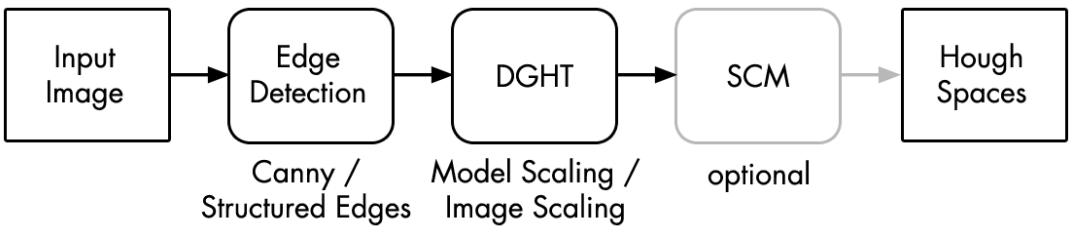


FIGURE 5.7: Components of the detection pipeline for handling object size variability

[8]Regarding object size variability, the simplest approach is to use scaled versions of a single model, trained on training data with limited variability. If lots of training data for different scales is available, it might be beneficial to train size-specific DGHT shape models by dividing the training data into different groups with similar object size. In this work, scaled versions of a single model are used.

Currently, the best localization hypothesis $\hat{c}$ according to Eq. 4.7 or Eq. 4.3 if using the SCM or not, respectively, is computed per Hough space, such that one detection would be generated per scale. This is not realistic, since the target object may be present at most at adjacent scales, but not across all scales. Thus, potential detections need to be further processed and possibly rejected. Since a similar approach is needed for the detection of multiple instances of an object, the corresponding algorithms and evaluations are presented in Sects. 5.3 and 5.5, respectively.

5.3 Detecting Multiple Instances

As introduced in Sects. 1.2 and 5.2.3 and in contrast to landmark localization in medical images, the number of target objects contained in a test image in a general computer vision task like pedestrian and car detection is unknown, which is a problem for the baseline system (see Chapter 4).

Currently, the best localization hypothesis $\hat{c}$ and the corresponding detection P is returned as the result for each Hough space. When applying multi-scale techniques as in Sect. 5.2, the approach would thus generate one result per scale. As explained above, this is not realistic, since a target object may be present at most at adjacent scales, but not across all scales. Furthermore, if more than a single instance of an object is present in an image, more hypotheses need to be generated. Moreover, if no target object is present at all, there would be too many results.

To solve these problems, an approach based on identifying local maxima in the Hough space(s) and subsequent processing is suggested: Similar to two-stage detectors (see Sect. 2.3.1), first a list of candidates or object proposals is generated for the (set of) Hough space(s). As this might lead to a lot of false positives (FP), a suitable rejection mechanism has to be incorporated. Here, simple comparisons of the peak height, in particular across multiple scales, are not sufficient, as the characteristics of the peak heights vary for each scale. This results from either different feature extraction characteristics with respect to varying input image sizes or, e.g., from intersecting model points and, therefore, multiple casted votes from a single edge pixel (for image and model scaling, respectively). This section describes the approach on how to handle the proposal generation as well as applicable rejection mechanisms.

5.3.1 Candidate Proposals

Candidate proposals are generated as follows: In each resulting Hough space, the object proposals are obtained by identifying the local maxima as follows:

For a test edge image $\mathbf{I}_E$, a DGHT shape model $\mathcal{M}$ is applied to generate a set of (scale-specific) Hough spaces $\mathbf{H}_s$ (see Fig. 5.6). In each resulting Hough space $\mathbf{H}_s$, local maxima $C_s = \{\hat{\mathbf{c}}_i\}$ are identified after applying a non-maximum suppression (NMS, for the exact parameterization see Sect. 5.5.1). A list $C_s = (\hat{\mathbf{c}}_1, \ldots, \hat{\mathbf{c}}_n)$ of most probable object positions $\hat{\mathbf{c}}_i$, ordered according to the respective Hough values, is derived for each scale s.

When additionally applying the SCM, the Random Forest Classifier is used to calculate the probability $p_{\hat{\mathbf{c}}_i}(\sigma_r)$ for each local maximum $\hat{\mathbf{c}}_i$ in $\mathbf{H_s}$ that the set of model

points voting for $\hat{\mathbf{c}}_i$ has a regular shape. The obtained probability is used as an additional weighting factor for the Hough space votes, i.e., $\mathbf{S_s}(\hat{\mathbf{c}}_i, \mathbf{I}_E) = \mathbf{H_s}(\hat{\mathbf{c}}_i, \mathbf{I}_E) \cdot p_{\hat{\mathbf{c}}_i}(\sigma_r)$. The local maxima in $\mathbf{H}_s$ are now sorted according to decreasing $\mathbf{S}_s(\hat{\mathbf{c}}_i, \mathbf{I}_E)$ to provide an ordered list $C_s = (\hat{\mathbf{c}}_1, \ldots, \hat{\mathbf{c}}_n)$ of most probable object positions $\hat{\mathbf{c}}_i$ for each scale s.

Finally, C denotes an ordered list compiled from all scale-specific results C_s each with the respective (scale-specific) bounding box coordinates. It therefore simply represents the combined list of all candidate proposals, which are then further processed in the object detection pipeline.

5.3.2 Candidate Rejection

As most likely not all candidates in C are correct detections, a proper rejection mechanism needs to be applied to the proposals in order to avoid a large number of false positives (FP). In this section, two alternative approaches are considered, namely a purely Hough-based rejection using the SCM and a rejection mechanism based on convolutional neural networks (CNN) for each candidate operating in image space.

5.3.2.1 SCM Rejection

The most evident approach for rejecting candidates is to use the respective, already computed probability $p_{\hat{\mathbf{c}}_i}(\sigma_r) \rightarrow [0, 1]$ for each candidate and to apply a rejection threshold θ. This means that in the initial, purely Hough-based detection pipeline, any candidate $\hat{\mathbf{c}}_i \in C$ is rejected if $p_{\hat{\mathbf{c}}_i}(\sigma_r) < \theta$, where θ is an appropriate rejection threshold, e.g., optimized on training or validation data.

5.3.2.2 CNN Rejection

As an alternative approach to the SCM rejection mechanism – motivated by its suboptimal performance and the corresponding error analysis (see Sect. 5.6.1.1) –, deep CNNs are used to accept or reject each proposal $\hat{\mathbf{c}}_i$ out of the list C generated by the DGHT or, alternatively, by the DGHT + SCM. Specifically, each candidate position $\hat{\mathbf{c}}_i \in C$ is transferred from Hough space to image space or to scaled image space when using image scaling (see Sect. 5.2.1). Then, a bounding box corresponding to the (scale-specific) mean object size is centered around that position. Note that in case of model scaling, the mean object size (and thus the bounding box) is scaled in the same way as the model. The image patch P corresponding to this bounding box is rescaled to a fixed input size given by the architecture of the used CNN. The patch pixel intensities of all three color channels are normalized to $[0, 1]$, and then used as input for the deep CNN.

For each candidate $\hat{\mathbf{c}}_i \in C$, the respective bounding box input is then classified by the CNN as either being accepted, i.e., containing the object of interest, or rejected, i.e., not containing the object of interest, using the CNN output $p_{\hat{\mathbf{c}}_i}(\text{object})$ and a rejection threshold θ.

In order to train the class-specific deep CNN that will be used for candidate rejection, a two-class classification problem is assumed. As positive samples, the annotated ground truth objects in the training set of the respective class are used. To select

negative samples, an implicit hard negative mining is performed using the class-specific SCM (see Sect. 4.3). The training samples for the SCM class σ_i (irregular shapes) correspond to hypotheses that inhibit a high Hough score but would be counted as false positives. Using this set of hypotheses, a certain number of negative samples can be extracted to achieve class balance (usually, a relation of $1:1$ to $1:3$ is used). Note that in general this rejection step would also be able to assign multiple class probabilities to each candidate such that only a single network pass for multiple class-specific DGHT shape models would be needed.

5.3.3 Detection Pipeline

The resulting detection pipeline that includes proposal generation and subsequent rejection in order to detect multiple instances of objects across scales is presented in Fig. 5.8.

The process described in Sects. 5.3.2.1 and 5.3.2.2, respectively, is applied to the compiled list C of all location + bounding box candidates resulting from all Hough spaces of either the image or model pyramid. In order to avoid double detections, the candidates which remain in each Hough space after candidate rejection are greedily grouped over the complete image or model pyramid, i.e., over all image or model scales, as follows:

Initially, when having applied image scaling, all bounding box predictions are transferred from scaled image space to input image space. Similarly, when having applied model scaling, scaled bounding boxes are located in input image space. In both cases, a group G_1 is initialized with the prediction with the largest SCM or CNN prediction score $p_{\hat{\mathbf{c}}_i}(\sigma_r)$ or $p_{\hat{\mathbf{c}}_i}(\text{object})$, respectively. For any additional prediction P from the candidate list C (processed in descending order of scores) the maximal IoU between P and any member of a group G_i is calculated; if the maximal IoU exceeds a fixed IoU threshold of 0.3, P is added to G_i; otherwise P initializes a new group G_j.

A NMS using $p_{\hat{\mathbf{c}}_i}(\sigma_r)$ or $p_{\hat{\mathbf{c}}_i}(\text{object})$ for SCM and CNN rejection, respectively, is then applied to each group. The detection results are therefore given by those candidates with $\max p_{\hat{\mathbf{c}}_i}(\sigma_r)$ or $\max p_{\hat{\mathbf{c}}_i}(\text{object})$ for each group, i.e., the bounding box P corresponding to either the maximal SCM or CNN classification score ($\max p_{\hat{\mathbf{c}}_i}(\sigma_r)$ or $\max p_{\hat{\mathbf{c}}_i}(\text{object})$) is identified and used as final output. Please see Fig. 5.9 for an illustration of the rejection, grouping and NMS steps.

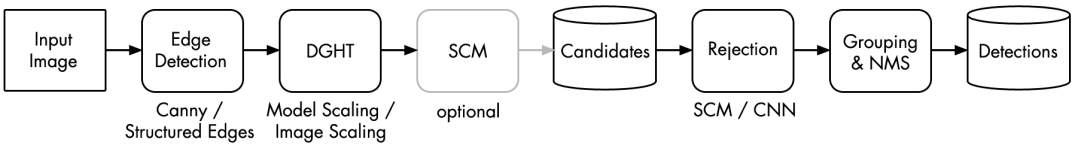


FIGURE 5.8: Components of the general detection pipeline for the detection of multiple instances.

Regarding the detection of multiple instances, a comparison of the two rejection mechanisms and a detailed error analysis is reported in Sect. 5.6. This also contains an evaluation of the impact on the detection performance of the individual components of the detection pipeline, of the length of the list of candidates C, and of the amount of DGHT training data. In detail, the complete experimental setup and all

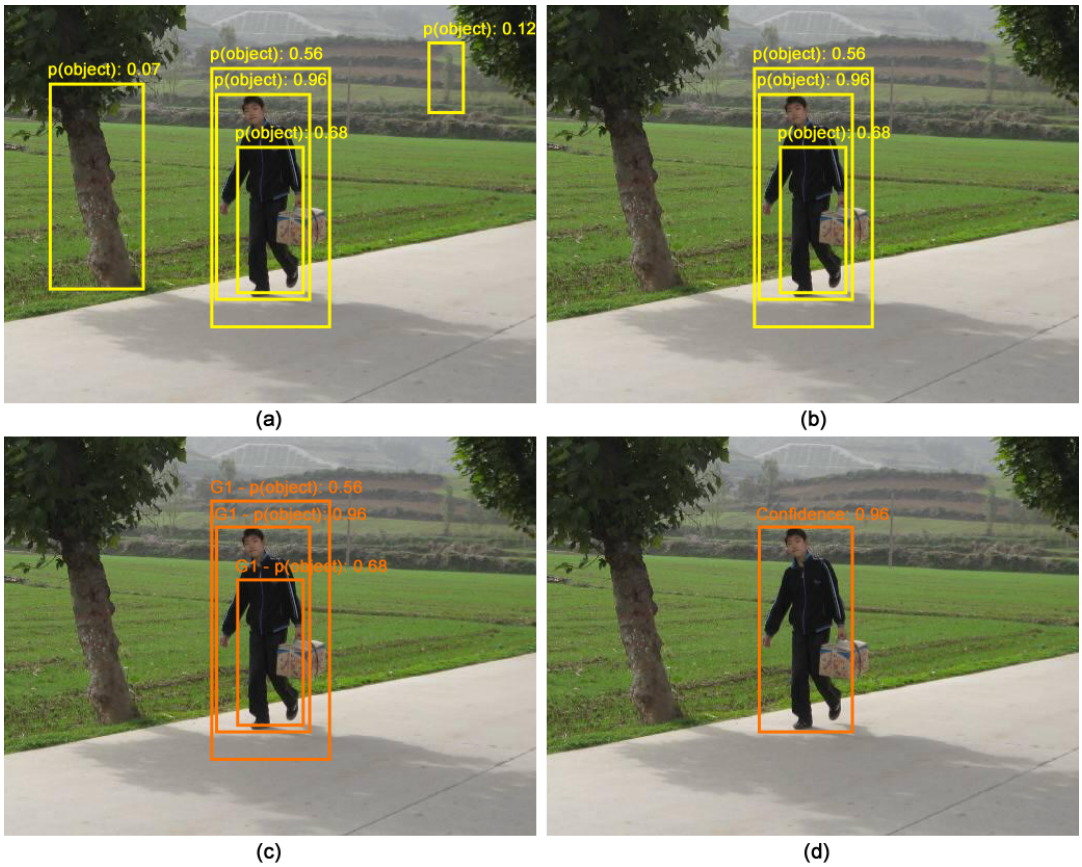


FIGURE 5.9: Illustration of the rejection, grouping and non-maximum suppression (NMS) steps. (a) Bounding box predictions and their corresponding object scores (b) Candidate rejection with threshold $\theta = 0.5$ (c) Grouping (d) Final detection result after NMS.

individual experiments are described in Sect. 5.5 after having introduced the final system extension that accounts for candidate refinement (see Sect. 5.4).

5.4 Candidate Refinement

The proposals generated by the DGHT consist of the most probable object positions as determined by the DGHT (alternatively by the DGHT + SCM) plus a mean bounding box (scaled when using model scaling) around that position (in the scaled image when using image scaling). Since the bounding box represents only a mean object size, the generated proposals are not always well aligned with the shape of an individual object. In some cases, a proposal might, e.g., contain only the torso of a person, without the extremities. Therefore, an additional extension to the object detection pipeline, namely bounding box regression, is introduced when applying CNN rejection[9].

[9]Bounding box regression would of course also help when applying SCM rejection. Here, a separate regression mechanism operating in Hough or in image space (or both) would need to be trained. Due to the much lower detection performance when using SCM rejection, this has not been evaluated.

5.4.1 Bounding Box Regression

Because of the inaccuracies resulting from a rather coarse input image pyramid or using mean bounding boxes, respectively, a CNN bounding box regression is suggested to refine the coordinates of any bounding box proposed by the DGHT (alternatively DGHT + SCM). Specifically, for each image patch P given to the CNN as input (using the same setup as in Sect. 5.3.2.2), the refinement CNN predicts a set

$$D = \{d_x(P), d_y(P), d_w(P), d_h(P)\} \tag{5.1}$$

of bounding box transformations, where $d_x(P)$ and $d_y(P)$ specify scale-invariant translations of the center point of P and $d_w(P)$ and $d_h(P)$ specify log-space translations of the bounding box's width and height, respectively, using standard linear regression as described by Girshick et al. [49] based on the bounding box regression ideas of Felzenszwalb et al. [36].

Using the predicted set D, the candidate P is transformed using

$$\begin{aligned} \hat{P}_x &= P_w d_x(P) + P_x \\ \hat{P}_y &= P_h d_y(P) + P_y \\ \hat{P}_w &= P_w \exp(d_w(P)) \\ \hat{P}_h &= P_w \exp(d_h(P)) \end{aligned} \tag{5.2}$$

to obtain a refined candidate $\hat{P}$.

When using this component, two CNNs are applied sequentially, one for bounding box regression, one for proposal rejection, explained in Sect. 5.5.1. The advantage of this sequential architecture is that already refined image patches, which are better centered and sized around the object candidate, are used as an input to the classification CNN. While it is worthwhile to perform bounding box regression and classification in parallel (sharing convolutional features and thus reducing runtime) using a combined end-to-end classification/regression training procedure, the effect of potential mismatches of the generated bounding boxes might have to be compensated for, e.g., by using larger image patches.

In order to train the class-specific regression CNNs, the ground truth bounding boxes of the respective training set are used to create variations of these annotations with respect to position and scale. The resulting pairs of variation and corresponding ground truth box (G^*, G) form the training set for the regression CNN, which is used to train the following regression targets t_* based on a standard L2 loss function, i.e., least mean squares regression:

$$\begin{aligned} t_x &= (G_x - G^*_x)/G^*_w \\ t_y &= (G_y - G^*_y)/G^*_h \\ t_w &= \log(G_w/G^*_w) \\ t_h &= \log(G_h/G^*_h) \end{aligned} \tag{5.3}$$

5.5 Experiments

This section describes the extensive experiments and the corresponding setup in order to evaluate the DGHT object detection framework with respect to handling object size variability and detecting multiple instances using the different components and configurations as described in the previous sections.

5.5.1 Experimental Setup and System Parameters

As introduced in Sect. 1.4, the experimental setup and system parameters have already been described in [42]. Regarding the proposal rejection mechanism, two specific configurations of the general DGHT object detection framework presented in Fig. 5.8 are considered in more detail: (a) An initial, purely Hough-based detection pipeline (Fig. 5.10) used for a detailed error analysis, which motivates to use the DGHT as proposal generator. This leads to (b) the final detection pipeline (Fig. 5.11), where an additional CNN-based bounding box regression and candidate rejection step have been integrated, as described in Sects. 5.4 and 5.3.2.2, respectively. In this section, the individual components and configurations of the DGHT object detection pipeline including the parameters used in the experiments are summarized in detail.

Edge detection: To all training and test images, either the Canny (see Sect.2.1.2.1) or the Structured Edge Detector (see Sect. 2.1.2.2) are consistently applied, depending on the chosen configuration of the pipeline[10]. The Structured Edge Detector is trained specifically on the Penn-Fudan database [125] and on the *car_side* category of the Caltech-101 database [35] for pedestrians and cars, respectively. Compared to Canny edge detection, the Structured Edge Detector suppresses most of the background edges and thus significantly reduces background variability [41] (see, e.g., Figs. 5.3 and 5.5).

DGHT model and SCM training: A class-specific DGHT model is trained for pedestrians and cars, respectively, which is used for both image and model scaling and which comprises a limited amount of size variability[11]. Specifically, a size range of the mean object height $\pm 10\%$ is allowed. All training images with objects not in this size range are scaled to an object size selected randomly from the allowed range (uniform distribution), separately for each object in an image. To train the DGHT shape models (see Sect. 4.2), only those training images are used that contain "simple" objects (IAIR difficulty type "S", $1\,406$ pedestrians / 775 images and 915 cars / 567 images, respectively). With these trained DGHT models, the respective SCMs are additionally trained on the full IAIR training sets for pedestrians and cars, respectively, comprising all difficulty types and all objects scaled to the allowed range as described above (see also Sect. 4.3).

Regarding the pedestrian model, the mean object height $\pm 10\%$ corresponds to $144 - 176$px. For class σ_r (regular shapes) in SCM training, only those hypotheses are used where the model covers the object almost completely, i.e., an intersection over union (IoU) of $> 70\%$. This approximately corresponds to a center point localization error

[10] The purely Hough-based detection pipeline of Fig. 5.10 is only used with class-specific structured edge detection (SSE) input features for performance reasons as described in [41].

[11] Except from the object size constraint, the training data contains other modes of variation (e.g., frontal/side views).

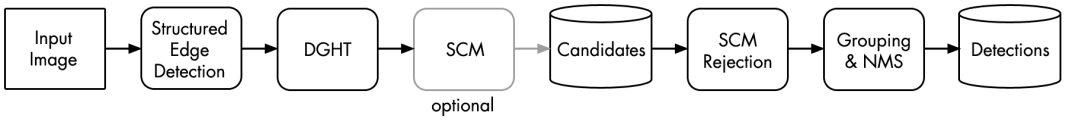


FIGURE 5.10: Components of the initial, purely Hough-based detection pipeline. From [42]

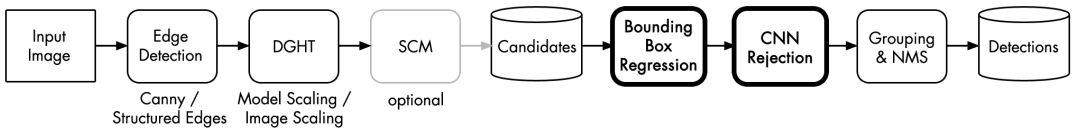


FIGURE 5.11: Components of the final detection pipeline. Based on [42]

of $< 20\%$ of the smaller model dimension, which for pedestrians is the model width. For class σ_i (irregular shapes), those hypotheses are used, where the model almost has no overlap with the object of interest, i.e., an IoU of $< 20\%$. This corresponds to a center point localization error of $> 20\%$ of the larger model dimension, here, the model height. Therefore, for pedestrian detection, ε_1 is set to 10px and ε_2 to 30px. Please see Fig. 5.12 (A) for an illustration of the localization error thresholds.

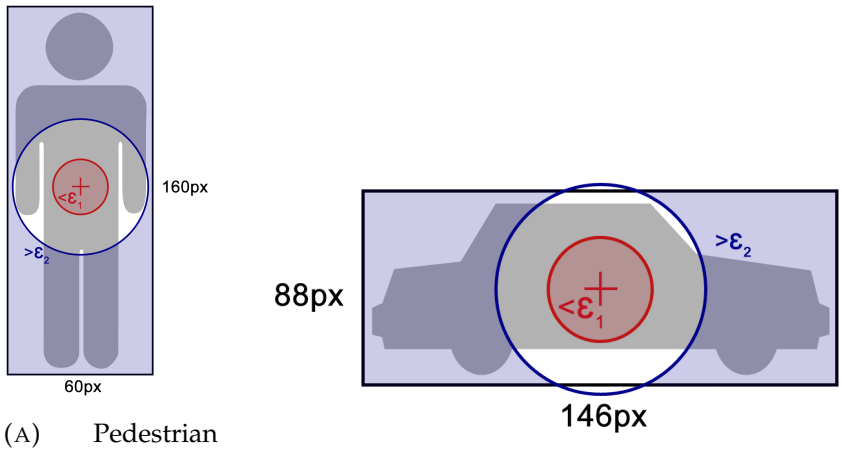
For training the DGHT car model, the mean object height $\pm 10\%$ corresponds to $79 - 97$px. All available aspects, i.e. all views, are trained into one single model. Using the same rule as for the pedestrian SCM training, ε_1 for class σ_r is set to 15px and ε_2 for class σ_i to 30px (for cars, the larger model dimension is the model width, see Fig. 5.12 (B)).

Handling of object size variability: To handle the large range of object sizes contained in the test data[12], either model or image scaling is applied as described in Sect. 5.2. First, a set of object sizes expected in the test data is defined. Here, this size range is simply calculated from the training data, but could be manually set instead. Thus, an expected minimal and a maximal size of a test object is specified.

For image scaling, the minimal and maximal observed object size in relation to the mean object size used to train the DGHT shape model define a (rounded) minimal and a maximal scaling factor. Then, a set of intermediate scaling factors is determined heuristically such that every expected size of a test object can be scaled to a value within the size range (mean object height $\pm 10\%$) that has been used to train the DGHT model. For the pedestrian and car detection tasks, this leads to the set of scaling factors presented in Tab. 5.4 (lines "Image Scaling"). Each test image is scaled with all scaling factors for the respective category (note that instead of a separate set of scaling factors for each object category, a single set of scaling factors could be easily defined instead). The Canny or structured edge image is computed separately for each scaled input image. The trained DGHT model is then applied independently to each scaled (edge) image, leading to a set of Hough spaces (one for each scaling factor, see Sect. 5.2.1).

When using model scaling, the scaling factors are inversely correlated to the image scaling factors, i.e., large objects can be detected with a large model scaling factor,

[12] If the large range of object sizes would be trained into a single DGHT model, preliminary experiments revealed that the detection performance would decrease significantly. Moreover, the aim is to also handle object sizes in the test data which have not been observed in the training data.



(A) Pedestrian Model

(B) Car Model

FIGURE 5.12: Illustration of ε_1 and ε_2 used for the selection of training samples when training the SCM (see Sect. 4.3). Localization errors smaller than ε_1 (within the red circle, i.e., $< 20\%$ of the smaller model dimension) are used as examples for regular shapes, whereas localization errors larger than ε_2 (outside the blue circle, i.e., $> 20\%$ of the larger model dimension) are used as negative examples.

but with a small image scaling factor. Thus, the ranges of the model and image scaling factors differ. Unlike in the initially performed image scaling, the intermediate scaling factors can be exactly computed using the mean model height $\pm 10\%$ such that adjacent, non-overlapping intervals are formed. For the pedestrian and car detection tasks, this leads to the scaling factors summarized in Tab. 5.4 (lines "Model Scaling"). Note that again a single set of scaling factors could have been defined instead of a separate set for each object category. All scaled models are independently applied to the edge image of each input image, thus again generating a set of Hough spaces (one for each model scaling factor, see Sect. 5.2.2). Note that in contrast to image scaling, the edge image has to be computed only once for each test image.

Traditionally, $8 - 16$ scales were used per octave [27]. For the DGHT object detection pipeline, it is sufficient to use roughly ten scales to cover three or more octaves.

TABLE 5.4: Scaling factors used in the experiments to detect test objects with a large object size variability, for pedestrian and car detection and for image and model scaling (see text). From [42]

Class	Approach	Scaling Factors
Pedestrian	Image Scaling	$0.5, 0.625, 0.75, 1.0, 1.5, 2.0, 2.25, 2.5, 2.75, 3.0$
Pedestrian	Model Scaling	$0.37, 0.45, 0.55, 0.67, 0.82, 1.0, 1.22, 1.49, 1.82, 2.22$
Car	Image Scaling	$0.25, 0.31, 0.37, 0.5, 0.75, 1.0, 1.5, 2.0, 2.25, 2.5$
Car	Model Scaling	$0.37, 0.5, 0.62, 0.75, 1.0, 1.5, 2.0, 2.5, 3.0, 3.5$

Proposal generation: As described above, for each test image a set of Hough spaces is generated (one for each scaling factor, either using the image scaling or the model scaling approach). In each individual Hough space, local maxima $C_s = \{\hat{\mathbf{c}}_i\}$ are identified using a NMS with a minimum distance of $1/3$ of the model width (for both object classes). These local maxima can be ordered according to their Hough scores $\mathbf{H}_s(\hat{\mathbf{c}}, \mathbf{I}_E)$, leading to an ordered list $C_s = (\hat{\mathbf{c}}_1, \ldots, \hat{\mathbf{c}}_n)$ of object proposals as described at the end of Sect. 4.2.

Optionally, the SCM can be used for an additional weighting of the Hough scores (see Sect. 4.3). In this case, the ordered list is generated according to the SCM-weighted Hough scores $\mathbf{S}_s(\hat{\mathbf{c}}, \mathbf{I}_E)$, again leading to an ordered list $C_s = (\hat{\mathbf{c}}_1, \ldots, \hat{\mathbf{c}}_n)$ of object proposals.

The analyses presented in Sects. 5.6.1.4 and 5.6.2.2 reveal that those candidates $\hat{\mathbf{c}}_i$ with

$$\mathbf{S}_s(\hat{\mathbf{c}}_j, \mathbf{I}_E) < 0.2 \cdot \max_{\hat{c}_i} \mathbf{S}_s(\hat{\mathbf{c}}_i, \mathbf{I}_E) \tag{5.4}$$

can be discarded, when having applied the SCM, or

$$\mathbf{H}_s(\hat{\mathbf{c}}_j, \mathbf{I}_E) < 0.2 \cdot \max_{\hat{c}_i} \mathbf{H}_s(\hat{\mathbf{c}}_i, \mathbf{I}_E) \tag{5.5}$$

otherwise, in order to reduce the amount of candidates. Finally, all scale-specific results C_s are combined to form the final ordered list C.

Bounding box regression: Any candidate position $\hat{\mathbf{c}}_i$ from each Hough space is independently transferred to image space (scaled in case of image scaling) yielding a location and a bounding box, which corresponds to the mean model size (scaled in case of model scaling) centered around the respective position in image space. A deep CNN is used to predict a refined proposal $\hat{\mathbf{c}}_i^*$. For these experiments, the standard Keras VGG16 model (see Sect. 2.3.1.2.1) is used, which is initialized on ImageNet. Keeping only the convolutional layers, a fully connected layer with 4 output units for regression is added. This CNN is then fine-tuned on the IAIR Pedestrian or Car training corpus, respectively. The training set (see Sect. 5.4.1) consists of pairs (G^*, G) where G is the ground truth bounding box of the annotated pedestrian or car and G^* a variation of G where width, height and center point coordinates are randomly modified. For the center point translation, a uniform distribution is used for each direction, whose boundaries are given by the respective image dimension. For width and height variation, a uniform distribution of the respective model scales as described in Tab. 5.4 is used. Initially, 50 variations per ground truth bounding box G are generated. If a variation G^* leads to an IoU with G of less than 0.5 in the case of pedestrians or 0.1 for cars, the variation is discarded. For fine-tuning, the Adam optimizer [70] is chosen with a mean squared error loss, a learning rate η of 0.001, which is reduced on plateaus, an input dimension of $(64 \times 64 \times 3)$, a batch size of 128 and 50 epochs.

CNN candidate rejection: Another deep CNN is then used to reject a (refined) candidate if $p(\text{object}) < \theta$ (see Sect. 5.3.2.2). Again, the convolutional layers of the standard Keras VGG16 model (see Sect. 2.3.1.2.1) are used, which are initialized on ImageNet. In order to significantly reduce the number of network parameters (and the number of computations), the fully connected layers of the standard VGG16 model are replaced by a global average pooling layer [81], followed by a fully connected layer with two nodes and a softmax. With this modification, the number of parameters of the VGG16 classification network can be reduced by almost 90%, from $138\,357\,544$ to $14\,715\,201$, nearly without performance loss on the pedestrian proposal rejection task as evaluated in [111]. Similar to the regression network, all CNN layers are fine-tuned on the IAIR Pedestrian or Car training corpus, respectively, using the annotated pedestrian or car bounding boxes scaled to $(64 \times 64 \times 3)$ as positive samples. As negative samples, the same candidates as for class σ_i in the

SCM training are used, i.e., high scoring peaks with a minimum error of 15 Hough cells for both pedestrians and cars. This step already performs hard negative mining, thus, the easy negative samples are filtered out accounting for a controllable class balance in CNN training (in these experiments a ratio of $1:3$ for positive and negative samples, respectively, is used). The training set is augmented through affine transformations and color jittering as described by Krizhevsky et al. [71]. Otherwise, the same training hyperparameters as for the regression network are used.

Combining scales and post-processing: Subsequent to the rejection step, the remaining candidate bounding boxes are greedily grouped (if the IoU is larger than 30%, see Sect. 5.3.3 and Fig. 5.9 for the exact grouping procedure) over the complete image or model pyramid, i.e., over all scales (see Tab. 5.4), and finally a second NMS is applied to each group using $p_{\hat{\mathbf{c}}_i}(\sigma_r)$ or $p(\text{object})$ as criterion for SCM and CNN rejection, respectively, in order to avoid double detections.

Analysis: As suggested by Zhang et al. [138], a detailed error analysis is conducted for the initial, purely Hough-based detection pipeline including oracle experiments: (a) localization oracle (all false positives (FP) that overlap with the ground truth, i.e., IoU > 0, are ignored) and (b) background vs. foreground oracle (all FP that do not overlap with the ground truth, i.e., IoU = 0, are ignored). In addition, another oracle experiment (c) is conducted: perfect rejection oracle (DGHT as a proposal generator). For each ground truth annotation, the rejection oracle picks the candidate with largest IoU out of the list C generated by the DGHT (or DGHT + SCM) and rejects all other candidates. Thus, the minimal miss rate for the DGHT as proposal generator can be quantified, assuming a perfect proposal rejection mechanism.

Additionally, for this final detection pipeline, the false negatives (FN) of the reported results are quantitatively analyzed whether they are misclassified (i.e., wrongly rejected by the CNN) or not included in the DGHT candidates. For all FP and FN of the reported results, possible reasons for these errors are qualitatively assessed.

5.5.2 Comparison to State-of-the-Art Approaches

The DGHT object detection framework is compared against several state-of-the-art algorithms. On the IAIR Pedestrian and Car test corpora, the latest DPM release (DPMv5) [47] trained on PASCAL, Faster R-CNN [102] and the pre-trained YOLOv2 [100] full model (both pre-trained on ImageNet and fine-tuned on PASCAL) are chosen as comparison approaches, as the latter is as of February 2018 the best performing algorithm on PASCAL VOC while running at a much higher frame rate than SSD [86, 100]. Thus, a comparison against the arguably best current one-stage (YOLOv2) and the most popular two-stage (Faster R-CNN) object detection approaches [83] is performed. Additionally, for both investigated object classes, the respective pre-trained YOLOv2 full model is fine-tuned on the IAIR Pedestrian and Car training set, respectively. The details of these state-of-the-art approaches can be found in Chapter 2 or in the respective references. For the INRIA Person database, the respective benchmark results from the Caltech Pedestrian Benchmark [1] of February 2018 are used. For TUD Pedestrian and UIUC Multi-scale, the benchmarks collected by Yao et al. [135] and Zheng and Liang [140] are used, respectively.

5.5.2.1 Fixed Region Proposal (RP) scheme

Furthermore, the detection performance in terms of the minimal miss rate for the proposals generated by the DGHT or the DGHT + SCM followed by a bounding box regression step is also compared against a fixed region proposal (RP) scheme similar to the one described by Lenc and Vedaldi [77] followed by the same bounding box regression step; for the latter system, the number of generated proposals is varied. The basic approach is to systematically cover the whole image similar to a sliding window technique, but generally with a low number of windows because the subsequent regression might compensate for inaccurate proposals.

If only very few fixed candidates (< 50) should be generated, large windows need to be used to ensure that all potential objects in the image are covered by at least one window. Mostly the regression CNN has problems, however, if the deviation between the proposal size and the true object size is too large or if several objects are contained in one window. Therefore, windows of smaller scales are added. At most, ten scales are used, whose range correspond to the minimum and maximum object size of the training data set. Regarding the aspect ratio, the mean ratio of the training data set is used. For each scale, the stride is defined as half of the width and height for the horizontal and vertical direction, respectively. Naturally, the smaller the window size, the more windows are needed to cover the whole image.

In order to ensure a fair comparison, these fixed region proposals are post-processed in the same way as the proposals generated by the DGHT or, alternatively, the DGHT + SCM.

5.6 Results of Handling Size Variability and Detecting Multiple Instances

This section presents the results of the main experimental evaluation that has been described in Sect. 5.5. Here, the approaches for handling large size variability as well as detecting multiple instances are quantitatively evaluated. Also, oracle experiments (see Sect. 5.5.1 "Analysis"), evaluations for different pipeline configurations, the impact of the amount of DGHT training data and the length of the list C of proposals are reported. On the top level, this section is divided into pedestrian and car detection.

5.6.1 Pedestrian Detection

As a first step in the evaluation of the DGHT for pedestrian detection, the initial, Hough-based detection pipeline is applied to the IAIR Pedestrian database. In addition to a quantitative evaluation, also a qualitative error analysis in accordance with Zhang et al. [138] as well as oracle experiments are conducted. Afterwards, these results are compared against the CNN-based proposal refinement and rejection followed by the impact assessment of the individual components. Finally, the DGHT object detection framework for pedestrian detection is applied to two other pedestrian data sets without retraining in order to validate the generalization capabilities of the approach.

5.6.1.1 Analysis of the Initial, Hough-based Detection Pipeline

Initially, the pedestrian detection performance of the purely Hough-based DGHT + SCM pipeline (Fig. 5.10; using image scaling and structured edges; without proposal refinement and CNN rejection) is analyzed, where candidates are rejected by the SCM if $p_{\hat{\mathbf{c}}_i}(\sigma_r) < \theta$ and θ is chosen such that the FPPI rate over the complete test set is equal to 1. This leads to the results shown in Tab. 5.5 in comparison to other approaches on the IAIR Pedestrian test corpus as described in Sect. 5.5.2.

TABLE 5.5: Comparison of the initial, purely Hough-based detection pipeline without CNN ("DGHT + SCM") to other approaches – including oracle experiments (see Sect. 5.5.1 "Analysis") – on the IAIR Pedestrian test corpus. The different difficulty categories of the IAIR corpus are denoted as:
S: Simple, D1: Occlusion, D2: Low Contrast, D3: Infrequent Shape
From [42]

Approach	Miss Rate at 1 FPPI				
	S	D1	D2	D3	All
DGHT + SCM	0.22	0.40	0.66	0.32	0.34
Two-Layer HOG [133]	0.25	0.47	0.44	0.50	0.35
DPM [133]	0.29	0.37	0.45	0.36	0.34
DPMv5 [47]	0.16	0.32	0.40	0.40	0.25
Faster R-CNN [102]	0.07	0.23	0.47	0.07	0.18
YOLOv2 Pre-trained [100]	0.12	0.25	0.28	0.20	0.19
YOLOv2 Fine-tuned [100]	0.05	0.12	0.11	0.10	0.09
DGHT Localization Oracle (a) at 1 FPPI	0.20	0.35	0.49	0.30	0.31
DGHT BG vs. FG Oracle (b) at 0.3 FPPI	0.09	0.24	0.24	0.23	0.16
DGHT Perfect Rejection Oracle (c) at 0 FPPI	0.01	0.01	0.20	0.00	0.04

Perfect Rejection Oracle (c)	Minimal Miss Rate	ABO [%]
DGHT + SCM	0.04	78.2

As can be seen, the overall pedestrian detection performance of the DGHT is comparable to the Two-Layer HOG and the published original DPM result of Wu et al.[133], but worse than the current DPMv5, Faster R-CNN as well as the pre-trained and fine-tuned YOLOv2. As expected, low contrast (difficulty "D2") is the hardest difficulty category for the DGHT. Concerning the corresponding 317 instances, a miss rate of 0.66 at 1 FPPI is achieved. Possible reasons are that due to the low contrast (1) insufficient edge representations or no edges at all could be generated and (2) that the altered model point voting pattern is classified as an irregular shape by the SCM.

The presented results motivate a detailed error analysis in accordance with Zhang et al. [138]. For comparison, the DPMv5 is used since it is also a model-based approach that operates on feature images. Also, oracle experiments are performed as described in Sect. 5.5.1 "Analysis" and in the following paragraphs:

Fig. 5.13 and 5.14 show the manual, qualitative error analysis of all false positives (FP) and false negatives (FN) for DGHT+SCM and DPMv5, respectively.

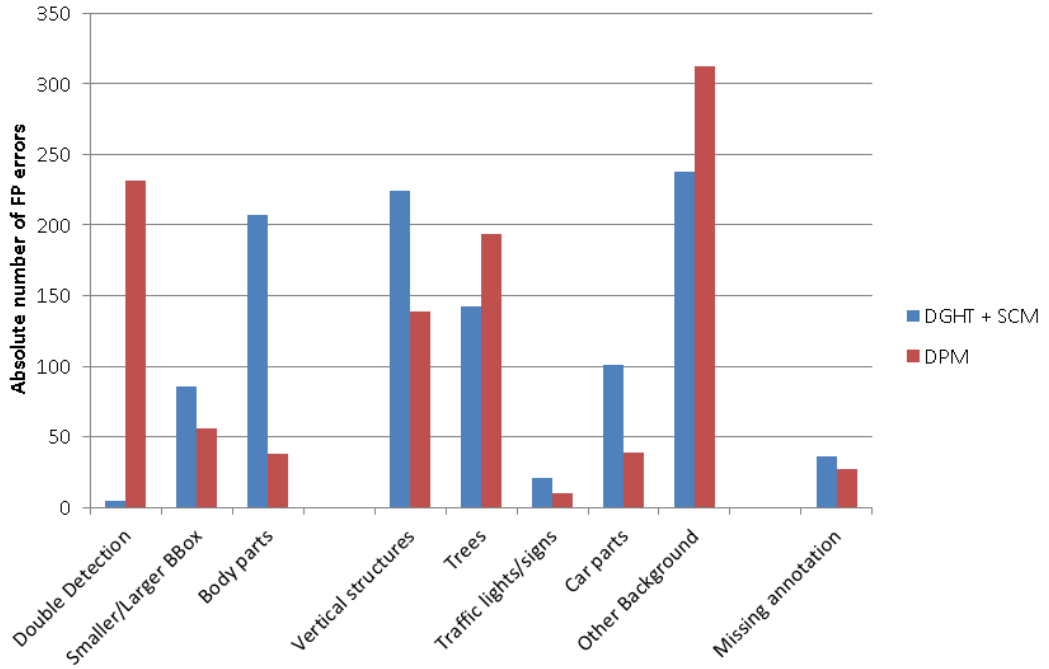


FIGURE 5.13: Analysis of all false positive (FP) errors of the DGHT+SCM in comparison to the DPMv5.

28% (DGHT) / 31% (DPM) of all false positives (FP) occur due to localization errors (first three pairs of bars in Fig. 5.13), mostly because of body part (DGHT) and double detections (DPM). For both approaches, the vast majority of FP are background detections (69% (DGHT) / 66% (DPM), bar pairs 4 to 8 in Fig. 5.13) often due to confusing vertical structures or trees. The DPM often detects pedestrian groups as one detection. The DPM has more FN because of small scales or occlusion. Low contrast or missing edges are a problem for both approaches but more pronounced for the DGHT. Small scales and occlusion are better handled by the DGHT pipeline. However, the DGHT FN are often only slightly below the rejection threshold θ (see "low score" in Fig. 5.14; with a lower θ, however, too many FP would have been generated), indicating that the SCM as a rejection mechanism could work more properly.

The results of the oracles (a), (b) and (c) (see Sect. 5.5.1 "Analysis") are shown in Tab. 5.5. Since the localization oracle only reduces the miss rate at 1 FPPI by 0.03, it shows that the DGHT detections are usually quite accurate (except for a few outliers). On the contrary, there is still much room for improvement regarding background vs. foreground errors. If the SCM would be able to reject all FP in the background (prediction with an IoU of 0%), the miss rate drops from 0.34 (DGHT + SCM result at 1 FPPI) to 0.16 at only 0.3 FPPI. This again indicates that the DGHT candidates are very accurate, but the rejection using only the model point voting pattern on structured edge images is not sufficient to properly overcome the well-known problems of small-scale detections or those of confusable background structures.

In case of perfect rejection of DGHT proposals, a miss rate of only 0.04 with an ABO score of 78.2% would be obtained (this is referred to as the perfect rejection oracle). This clearly indicates that the main drawbacks of the DGHT are (a) FP in the background and (b) non-optimal selection of candidates based on $\mathbf{S}(\hat{\mathbf{c}}_\mathbf{i}, \mathbf{I}_E)$. The low miss rate of the perfect rejection oracle motivates to use the DGHT as a proposal generator but to improve the proposal rejection. To this end, the DGHT object detection

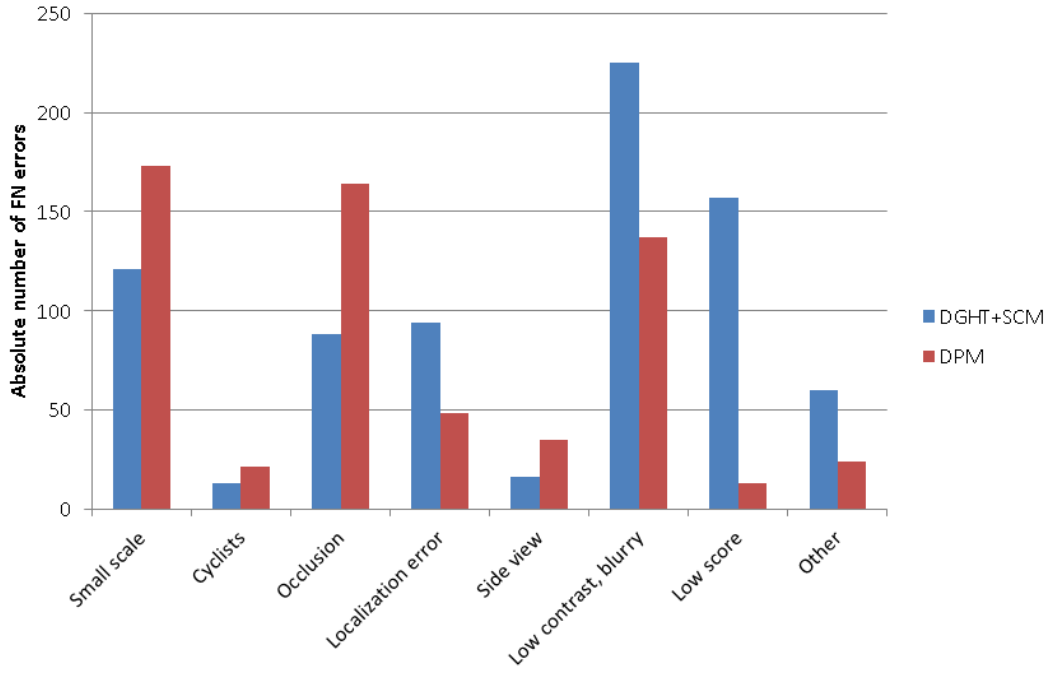


FIGURE 5.14: Analysis of all false negative (FN) errors of the DGHT+SCM in comparison to the DPMv5.

pipeline is extended by applying an additional proposal rejection step based on a deep convolutional neural network (CNN), as outlined in Sects. 5.3.2.2 and 5.5.1. Furthermore, the DGHT proposals are optionally refined by an additional bounding box regression step.

5.6.1.2 Final Detection Pipeline: Detection Results

Pedestrian detection results for the final detection pipeline – i.e., including an additional CNN proposal rejection step on top of the SCM rejection ("DGHT + SCM + VGG16") and an additional bounding box regression step ("DGHT + SCM + Regression + VGG16"), see Sect. 5.5.1 – are shown in Tab. 5.6; example detections are shown in Fig. 5.15. Full detection error trade-off (DET) curves are presented in Fig. 5.16. The final detection pipeline that includes bounding box regression ("DGHT + SCM + Regression + VGG16") achieves similar results as the recent YOLOv2 approach (for FPPI rates > 0.25), which has also been fine-tuned on the IAIR Pedestrian training corpus, and outperforms all other investigated approaches. Note, however, that the other approaches have not used IAIR training data. Without bounding box regression ("DGHT + SCM + VGG16"), comparable results to YOLOv2 Pre-trained and Faster R-CNN are achieved, which in contrast make use of a bounding box regression component. In contrast to using the SCM for proposal rejection, the CNN proposal rejection drastically improves the detection results. As for the SCM results, low contrast (D2) still is one of the hardest difficulty categories, however, the miss rate could be drastically reduced from 0.74 to 0.23 and is now on par with occlusion (D1) for the full detection pipeline. When additionally using bounding box regression ("DGHT + SCM + Regression + VGG16"), another significant detection performance improvement can be reported. The complete DGHT detection pipeline

TABLE 5.6: Comparison of different configurations of the detection pipeline to state-of-the-art algorithms on the IAIR Pedestrian test corpus; Evaluation at 0.5 FPPI since this is commonly the highest FPPI rate for the DGHT-based approaches. Note that the different approaches are trained on different training data, as indicated in the second column. The different difficulty categories of the IAIR corpus are denoted as:
S: Simple, D1: Occlusion, D2: Low Contrast, D3: Infrequent Shape
From [42]

Approach	Training Data	Miss Rate at 0.5 FPPI				
		S	D1	D2	D3	All
DGHT+SCM	IAIR	0.32	0.51	0.74	0.50	0.44
DGHT+SCM+VGG16	IAIR / ImageNet	0.09	0.30	0.40	0.20	0.19
DGHT+SCM+ Regression+VGG16	IAIR / ImageNet	0.03	0.23	0.23	0.10	0.11
DPMv5 [47]	PASCAL	0.18	0.37	0.47	0.45	0.29
Faster R-CNN [102]	ImageNet + PASCAL	0.08	0.26	0.53	0.08	0.20
YOLOv2 Pre-trained [100]	ImageNet + PASCAL	0.13	0.28	0.31	0.23	0.21
YOLOv2 Fine-tuned [100]	Im.Net+PASC.+IAIR	0.06	0.15	0.13	0.13	0.10

Perfect Rejection Oracle (c)	Minimal Miss Rate	ABO [%]
DGHT + SCM + Regression + VGG16	0.03	85.32

achieves the best overall performance in the difficulty category "S" out of all investigated approaches.

Regarding the DET curves (Fig. 5.16), the miss rates when not using bounding box regression ("DGHT + SCM + VGG16") are similar to those of Faster R-CNN. When additionally using bounding box regression ("DGHT + SCM + Regression + VGG16"), the miss rates are similar to "DGHT + SCM + VGG16" for FPPI rates < 0.15 and similar to the fine-tuned YOLOv2 for FPPI rates > 0.25.

5.6.1.3 Final Detection Pipeline: Component Analysis

In this section, the influence of the different components of the detection pipeline is analyzed (see Fig. 5.11). All configurations involve the DGHT for proposal generation, bounding box regression and proposal rejection by a deep CNN ("DGHT + Regression + VGG16"). The other components are varied: Either Canny edge detection ("Canny", Sect. 4.1) or class-specific structured edge detection ("SSE", Sect. 5.1.1) and either image or model scaling to handle object size variability (Sect. 5.2) are used. As another option, either the SCM is included prior to the CNN (Sect. 4.3) or removed, i.e., not used at all. In the latter configuration (without SCM), the candidate list is generated only based on the Hough scores $\mathbf{H}(\hat{\mathbf{c}}, \mathbf{I}_E)$, i.e., without SCM weights. The detection results (miss rate at 0.5 FPPI) for all different configurations are shown in Tab. 5.7. Similar results are achieved for all configurations of the DGHT + Regression + VGG16 pipeline, comparable to YOLOv2 Fine-tuned, with image scaling, SSE and including the SCM being the (minimally) best configuration. Similar results are achieved, because the bounding box regression compensates for slightly different configuration-specific detection proposals. In consequence, this enables to application-specificly choose a trade-off between maximum accuracy, runtime and

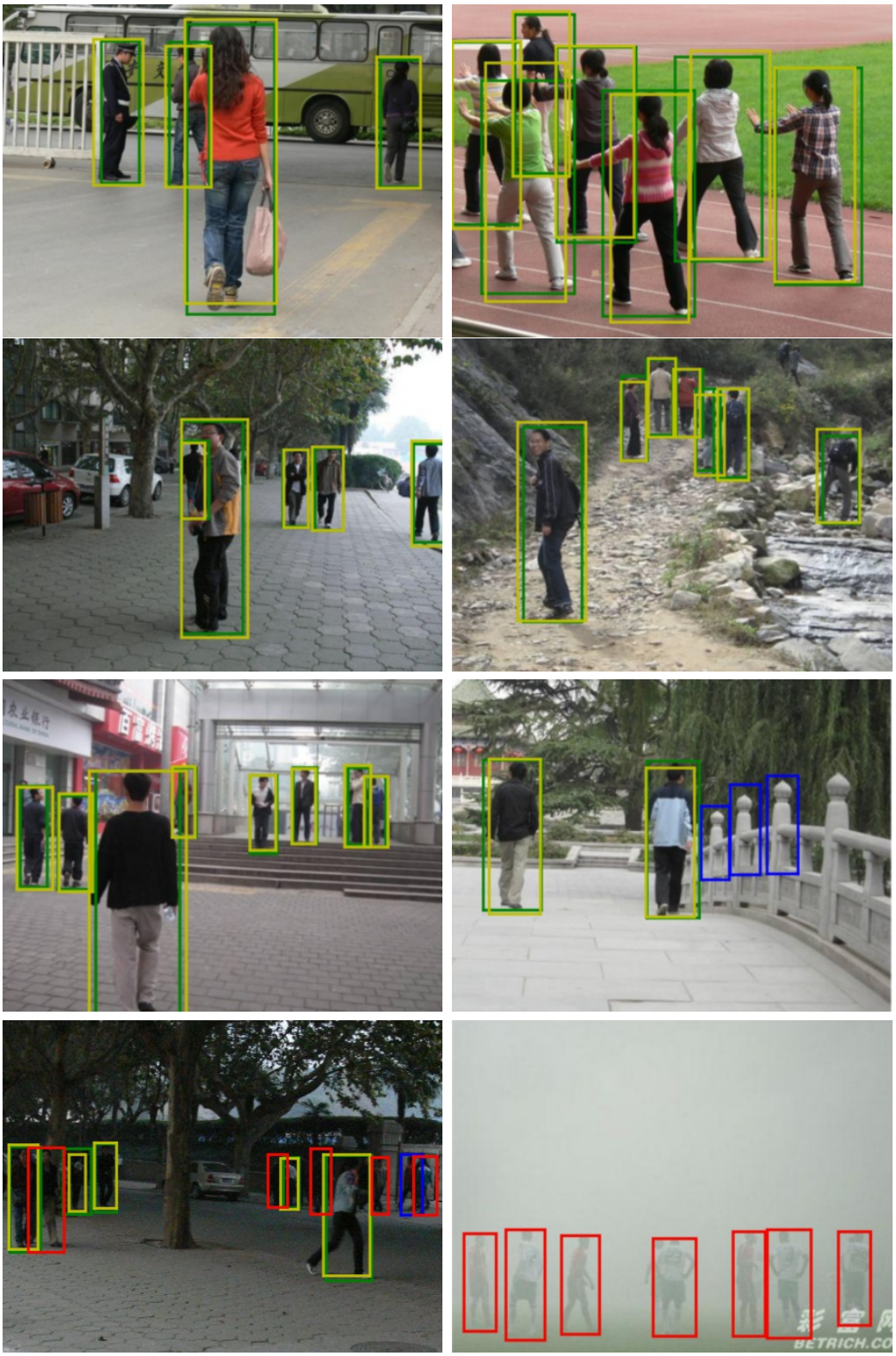


FIGURE 5.15: Example DGHT+SCM+Regression+VGG16 detections on IAIR Pedestrian.
(Green) ground truth (yellow) correct detection (blue) FP (red) FN; best viewed in color. From [42]

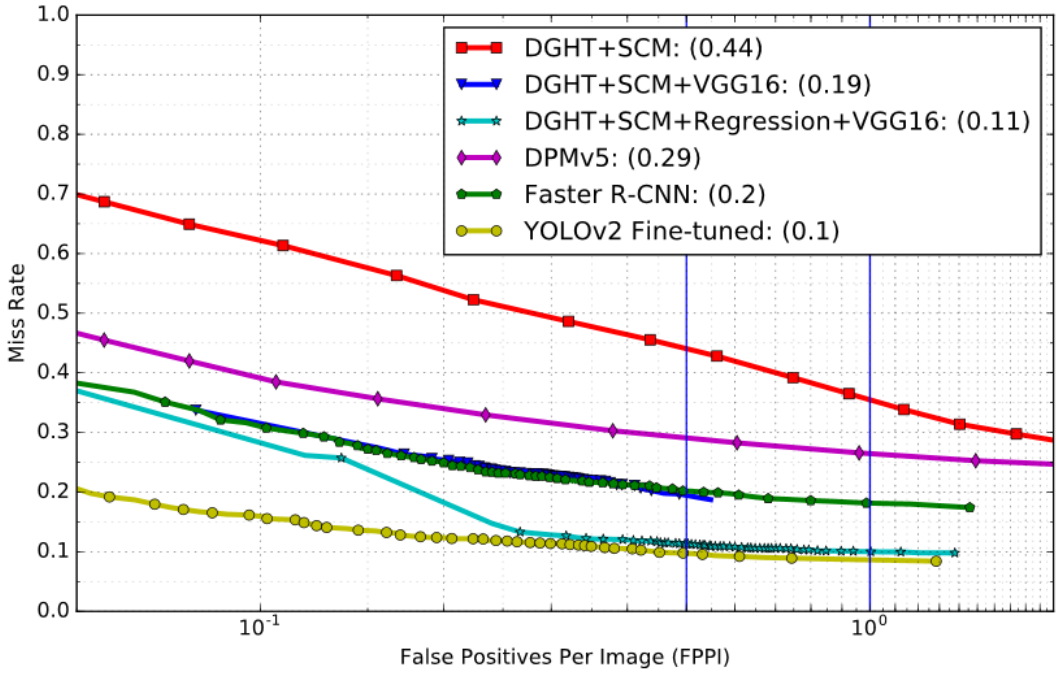


FIGURE 5.16: Comparison of detection results (DET curves) on the IAIR Pedestrian test corpus. From [42]

training effort. For instance, using Canny edge detection and no SCM would reduce the training effort, whereas using the SCM reduces the number of candidates (see Sect. 5.6.1.4 for the impact of the SCM and Sect. 6.4 for a runtime analysis). Canny edge detection can help to detect low contrast instances (difficulty "D2"). The detection of simple pedestrians (difficulty "S", $1\,416$ / $2\,367$ pedestrians) is of higher accuracy than YOLOv2 Fine-tuned and compensates for the higher miss rates when detecting occluded (difficulty "D1", 378 / $2\,367$ pedestrians) and low contrast pedestrians (difficulty "D2", 317 / $2\,367$ pedestrians).

5.6.1.4 CNN Rejection: Error Analysis

Table 5.8 shows the results after repeating the oracle experiments with the final detection pipeline. Again, these results confirm that the DGHT proposals are accurate because there is no gain in performance when ignoring those FP with $0 <$ IoU < 0.5 (localization oracle). Problems arise due to misclassifications between fore- and background. Only a slightly better miss rate (0.10 instead of 0.11) but at a much lower FPPI rate (0.1 instead of 0.5 FPPI) would have been achieved, if there were no FP errors at confusable background structures (BG vs. FG oracle).

Assuming a perfect rejection oracle – which, for each ground truth annotation, selects the candidate with the largest IoU out of the list C generated by either the DGHT, the DGHT+SCM or the DGHT+SCM+Regression and rejects all other candidates – the minimal miss rates in case of perfect proposal rejection is also quantified (see last line in Tab. 5.8 for the configuration "image scaling, SSE and SCM"). It clearly shows that low contrast (difficulty "D2") is the main error source. The reason for this is that the DGHT uses edge images to generate proposals.

TABLE 5.7: Performance comparison of different configurations of the DGHT+Regression+VGG16 pipeline on the IAIR Pedestrian test corpus: Canny or structured edge detection (SSE), model (MS) or image scaling (IS), with/without SCM. For ease of comparison, the results for DPMv5, Faster R-CNN and YOLOv2 detections are again included from Tab. 5.6. Note, however, that the training data used for the other algorithms differ as indicated in Tab. 5.6.
S: Simple, D1: Occlusion, D2: Low Contrast, D3: Infrequent Shape
From [42]

Approach	Setup	Miss Rate at 0.5 FPPI				
		S	D1	D2	D3	All
DGHT+Regression+VGG16	MS, Canny	0.05	0.24	0.18	0.11	0.12
DGHT+Regression+VGG16	MS, Canny, SCM	0.05	0.24	0.18	0.11	0.12
DGHT+Regression+VGG16	MS, SSE	0.04	0.23	0.21	0.12	0.12
DGHT+Regression+VGG16	MS, SSE, SCM	0.06	0.22	0.22	0.13	0.12
DGHT+Regression+VGG16	IS, Canny	0.04	0.23	0.21	0.18	0.12
DGHT+Regression+VGG16	IS, Canny, SCM	0.04	0.22	0.21	0.18	0.12
DGHT+Regression+VGG16	IS, SSE	0.03	0.22	0.25	0.10	0.12
DGHT+Regression+VGG16	IS, SSE, SCM	0.03	0.23	0.23	0.10	0.11
DPMv5 [47]	Training	0.18	0.37	0.47	0.45	0.29
Faster R-CNN [102]	data as	0.08	0.26	0.53	0.08	0.20
YOLOv2 Pre-trained [100]	indicated	0.13	0.28	0.31	0.23	0.21
YOLOv2 Fine-tuned [100]	in Tab. 5.6	0.06	0.15	0.13	0.13	0.10

TABLE 5.8: DGHT+SCM+Regression+VGG16 oracle results on the IAIR Pedestrian test corpus. Setup: image scaling, SSE and SCM.
From [42]

Experiment	S	D1	D2	D3	All
DGHT + SCM + Regression + VGG16 at 0.5 FPPI	0.03	0.23	0.23	0.10	0.11
DGHT Localization Oracle (a) at 0.5 FPPI	0.03	0.22	0.23	0.10	0.11
DGHT BG vs. FG Oracle (b) at 0.1 FPPI	0.03	0.21	0.21	0.09	0.10
DGHT Perfect Rejection Oracle (c) at 0 FPPI	0.01	0.01	0.17	0.00	0.03

In Fig. 5.17, the minimal miss rate is analyzed as a function of the number $|C|$ of proposals per image with and without the SCM and with or without additional bounding box regression. With the SCM, the number of proposals per image (controlled by the threshold θ) can be significantly smaller than without the SCM at no performance loss, since the SCM effectively removes many wrong proposals from the list. Compared to a fixed region proposal (RP) scheme (see Sect. 5.5.2.1) with the same additional bounding box regression as in the DGHT object detection pipeline, the DGHT + SCM pipeline achieves much smaller miss rates for a given number of candidates (in the regime with at most 600 candidates per image). Curves for the actual miss rate (including the same CNN-based proposal rejection step as well as grouping and post-processing for the final detection pipeline and for the fixed RP scheme) are presented in Fig. 5.18. Using only the top-scoring 100 proposals, the DGHT object detection pipeline achieves the same miss rate at 0.5 FPPI as the best miss rate achieved by the fixed RP scheme (0.12 when using 2 973 fixed region proposals).

Still, there is quite a large gap between the minimal miss rate of 0.03 (85.32% ABO) and the detection result of 0.11 at 0.5 FPPI. 35% of all FP are body parts, so the CNN still has problems with partly detected pedestrians, most of the time within

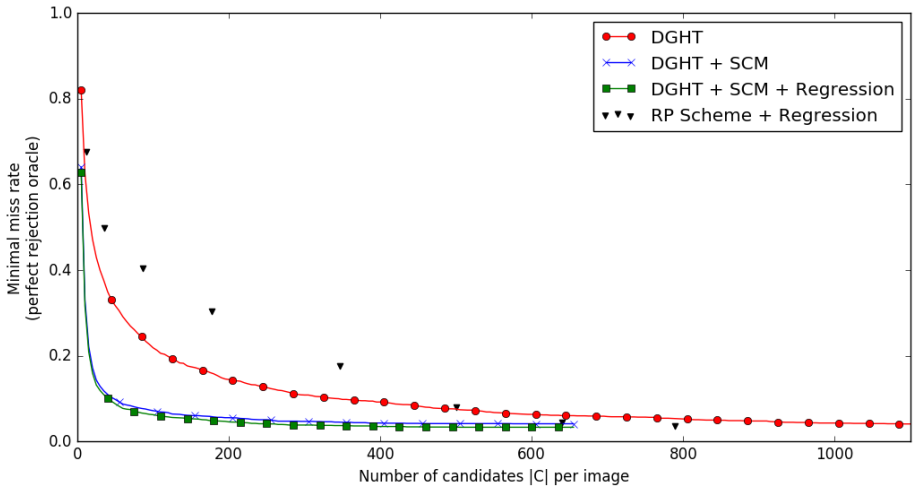


FIGURE 5.17: Minimal miss rate on the IAIR Pedestrian test corpus based on an ordered list C of proposals provided by the DGHT / DGHT+SCM / DGHT+SCM+Regression, as function of the length $|C|$ of the list (setup: image scaling, SSE). From [42]

correctly detected complete pedestrians. Another 40% are false detections in the background, mostly at confusable, e.g., vertical, structures. The remaining 25% of the FP show pedestrians, which are not annotated, i.e., rather small pedestrians in large distances, cyclists and heavily occluded or sitting persons. Fig. 5.19 presents examples for FP detections on the IAIR Pedestrian test set.

The question remains, if the missed pedestrians are falsely rejected by the CNN classifier or not included in the DGHT proposals: As expected from the perfect rejection oracle, 173 of the 249 FN at 0.5 FPPI (69.5%) are included in the DGHT candidates (with an ABO of 78.3%), but rejected by the CNN. Main reasons for these false rejections are low contrast (37%), occlusion (16%) and the small size (14%).

The remaining 76 FN (which are not detected by the DGHT) have an average size of 27×75px, i.e., roughly half of the mean pedestrian model size. In addition to the small size, those missed pedestrians are of very low contrast, either in dark lighting conditions or foggy backgrounds (see Fig. 5.20).

5.6.1.5 CNN Rejection: DGHT Training Data

An advantage of the DGHT is the relatively low amount of training material needed (as compared to deep networks). To demonstrate this, the amount of training data is reduced by randomly selecting 25% and 50% from the IAIR Pedestrian training corpus. This restricted set is used to train the DGHT model and the SCM (both CNNs are fine-tuned on the whole IAIR training corpus). The detection results on the IAIR Pedestrian test corpus (using image scaling, SSE and including the SCM) are shown in Tab. 5.9. These results show that the overall detection performance of the final pipeline is not very much affected by the amount of training data used for the DGHT shape model training. Instead, the minimal miss rate drops when using less training data indicating that a more general pedestrian shape model has been learned.

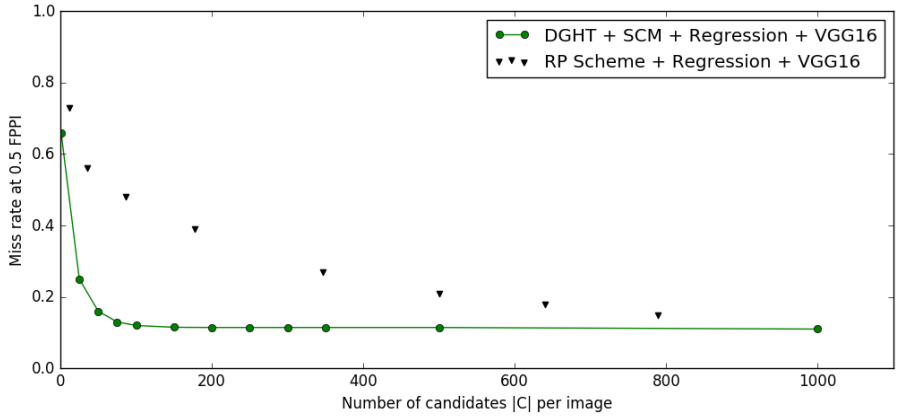


FIGURE 5.18: Actual miss rates at 0.5 FPPI on the IAIR Pedestrian test corpus based on an ordered list C of proposals provided by the DGHT+SCM+Regression+VGG16, as function of the length $|C|$ of the list (setup: image scaling, SSE)

TABLE 5.9: Detection results using fractions of the IAIR Pedestrian training corpus for "DGHT+SCM+Regression+VGG16". Setup: image scaling, SSE and SCM. From [42]

Training Data	Miss Rate at 0.5 FPPI	Minimal Miss Rate
100%	0.11	0.03
50%	0.10	0.01
25%	0.11	0.01

5.6.1.6 Evaluation on Further Pedestrian Databases

In order to address the detection performance of the DGHT object detection pipeline on further pedestrian test corpora (different data collections than used in training), detection results are reported on the TUD Pedestrian and INRIA Person data sets in Tab. 5.10 and 5.11, respectively. Specifically, for comparison purposes, two different configurations of the detection pipeline are used, both including CNN proposal rejection (but no bounding box regression):

Setup 1**:** Image scaling, structured edges, with SCM

Setup 2**:** Model scaling, structured edges, no SCM

The goal of these experiments is to show that the DGHT detection approach, in particular, the proposal generation, also performs well on other databases, without retraining any component of the pipeline. To this end, the performance of YOLOv2 or Faster R-CNN has not additionally been evaluated, instead, the result of other state-of-the-art approaches reported for those databases in the literature have been used for comparison.

While comparing the DGHT results to the reported state-of-the-art, please note the difference in training data: Whereas the other algorithms at least partly use domain-specific training data (TUD or INRIA, respectively), the DGHT object detection pipeline does not. These experiments show that the DGHT pedestrian detection pipeline – trained on the IAIR corpus – achieves competitive or acceptable detection results also on the TUD Pedestrians and INRIA Person corpora, respectively. In particular,

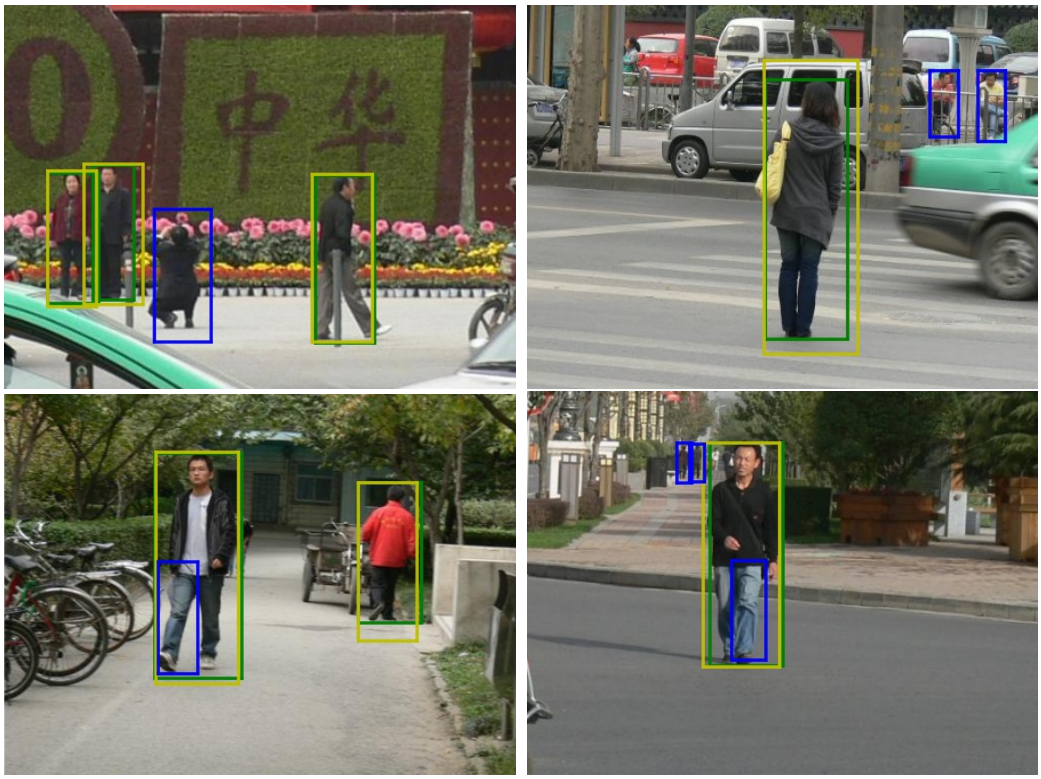


FIGURE 5.19: Example DGHT+SCM+Regression+VGG16 false positive detections on IAIR Pedestrian.
(Green) ground truth (yellow) correct detection (blue) FP (red) FN; best viewed in color

TABLE 5.10: Detection performance (Recall at equal error rate (EER)) and perfect rejection oracle on TUD Pedestrians. DGHT: DGHT + VGG16 rejection (but no bounding box regression). Setup 1: image scaling, SSE, SCM; Setup 2: model scaling, SSE, no SCM. Note that no component of the DGHT detection pipeline has been retrained on TUD Pedestrians. From [44, 42]

Approach	Training Data	Recall at EER
DGHT (Setup 1)	IAIR / ImageNet	0.88
DGHT (Setup 2)	IAIR / ImageNet	0.85
PartISM [5]	TUD + INRIA	0.84
Hough Forests [45]	TUD + INRIA	0.87
Part Alph. and Pose Dict. [135]	TUD + INRIA	0.92
Perfect Rejection Oracle (c)	Minimal Miss Rate	ABO [%] (proposals)
DGHT (Setup 1)	0.01	75.8 (55)

minimal miss rates of 0.01 (75.8% ABO) at 55 candidates per image and 0.01 (76.8% ABO) at 102 candidates per image are reported on TUD Pedestrians and INRIA Person, respectively. This shows the good generalization capability of the DGHT pedestrian shape model, because almost all ground truth pedestrians could be detected with very few candidates (≤ 102) if a perfect rejection mechanism is assumed.

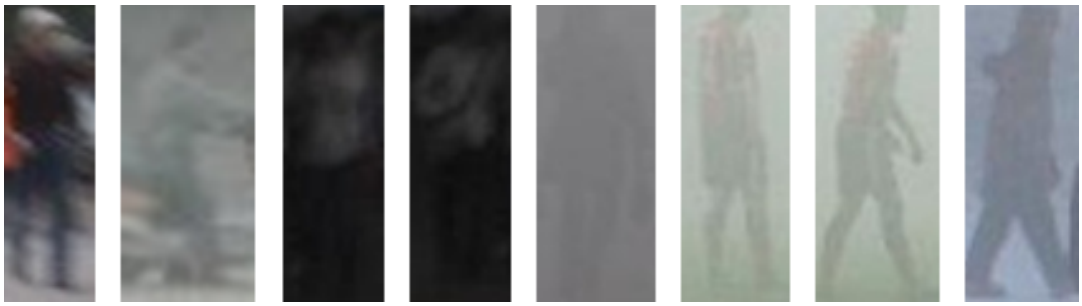


FIGURE 5.20: Examples for false negatives not detected by the DGHT on IAIR Pedestrian test corpus. From [42]

TABLE 5.11: Detection performance (miss rate at 1 FPPI) and perfect rejection oracle on INRIA Person. DGHT: DGHT+VGG16 rejection (but no bounding box regression). Setup 1: image scaling, SSE, SCM; Setup 2: model scaling, SSE, no SCM. Note that no component of the detection pipeline has been retrained on INRIA Person. From [44, 42]

Approach	Training Data	Miss Rate
DGHT (Setup 1)	IAIR / ImageNet	0.14
DGHT (Setup 2)	IAIR / ImageNet	0.12
`ChnFtrs` [28]	INRIA	0.14
Part Alph. and Pose Dict. [135]	INRIA	0.12
`FPDW` [27]	INRIA	0.09
`VeryFast` [10]	INRIA	0.07
Spatial Pooling [98]	INRIA + Caltech	0.04
Perfect Rejection Oracle (c)	Minimal Miss Rate	ABO [%] (proposals)
DGHT (Setup 2)	0.01	76.8 (102)

5.6.2 Car Detection

As a second object class, the DGHT object detection pipeline is also evaluated on a car detection task. Mainly, the experiments are performed on the IAIR Car data set, which additionally includes a large aspect variability. Due to the significantly better performance of the CNN rejection mechanism and because all aspects are trained into a single DGHT shape model, the configuration using CNN refinement and rejection is directly evaluated followed by an error analysis. Similar to the pedestrian detection task, the resulting detection pipeline is applied to another database for car detection without retraining of any component.

5.6.2.1 CNN Rejection: Detection Results

Table 5.12 shows the detections results (miss rate at 0.5 FPPI) on the IAIR Car test corpus for the final detection pipeline using model scaling, structured edges and the SCM in comparison to the same state-of-the-art algorithms as in the pedestrian detection task; some example detections are shown in Fig. 5.21. The corresponding DET curves are shown in Fig. 5.22. The results for all difficulty levels outperform the previously published results of 0.24 and 0.36 for the DPM and the Two-layer HOG, respectively, from Wu et al. [133]. The best-performing algorithm on this corpus is

TABLE 5.12: Comparison of different configurations of the DGHT car detection pipeline to state-of-the-art algorithms on the IAIR Car test corpus; DGHT: DGHT + SCM + Regression + VGG16; Evaluation at 0.5 FPPI since this is commonly the highest FPPI rate for the DGHT. For comparison, also the previously published results at 1 FPPI of Wu et al. [133] are shown. Note that the different approaches are trained on different training data, as indicated in the second column. The different difficulty categories of the IAIR corpus are denoted as: S: Simple, D1: Occlusion, D2: Low Contrast, D3: Infrequent Shape From [42]

Approach	Training Data	Miss Rate at 0.5 FPPI				
		S	D1	D2	D3	All
DGHT (Image Scaling)	IAIR / ImageNet	0.05	0.27	0.22	0.09	0.11
DGHT (Model Scaling)	IAIR / ImageNet	0.04	0.21	0.14	0.14	0.10
DPMv5 [47]	PASCAL	0.12	0.44	0.39	0.10	0.22
Faster R-CNN [102]	Im.Net + PASCAL	0.03	0.34	0.51	0.08	0.16
YOLOv2 Pre-trained [100]	Im.Net + PASCAL	0.05	0.36	0.29	0.03	0.14
YOLOv2 Fine-tuned [100]	Im.Net+PASC.+IAIR	0.02	0.14	0.12	0.03	0.05

Approach	Training Data	Miss Rate at 1 FPPI				
		S	D1	D2	D3	All
Two-Layer HOG [133]	IAIR	0.23	0.59	0.46	0.36	0.36
DPM [133]	IAIR	0.19	0.31	0.43	0.15	0.24

Perfect Rejection Oracle (c)	Minimal Miss Rate	ABO [%]
DGHT + SCM + Regression + VGG16	0.02	88.04

TABLE 5.13: DGHT+SCM+Regression+VGG16 oracle results on the IAIR Car test corpus. Setup: image scaling, SSE and SCM. From [42]

Experiment	S	D1	D2	D3	All
DGHT + SCM + Regression + VGG16 at 0.5 FPPI	0.04	0.21	0.14	0.14	0.10
DGHT Localization Oracle (a) at 0.5 FPPI	0.04	0.21	0.14	0.14	0.10
DGHT BG vs. FG Oracle (b) at 0.2 FPPI	0.04	0.22	0.16	0.14	0.10
DGHT Perfect Rejection Oracle (c) at 0 FPPI	0.01	0.05	0.11	0.00	0.02

YOLOv2 fine-tuned on the IAIR car training corpus, while the DGHT object detection pipeline outperforms the YOLOv2 version pre-trained on ImageNet + PASCAL as well as the DPMv5 and Faster R-CNN for FPPI rates > 0.3.

When using model scaling, lower miss rates at 0.5 FPPI than for image scaling were achieved for the difficulty categories "D1" (occlusion, 206 of 1 346 cars) and "D2" (low contrast, 57 of 1 346 cars) in contrast to the use of image scaling.

For FPPI rates > 0.3, the DGHT object detection pipeline achieves better results than Faster R-CNN and YOLOv2 pre-trained being only outperformed by YOLOv2 fine-tuned.

FIGURE 5.21: Example DGHT+SCM+Regression+VGG16 detections on IAIR Car. Please note that some false positives are vans or jeeps that would have been correctly detected (see example 5 and 6 in the third row).
(Green) ground truth (yellow) correct detection (blue) FP (red) FN; best viewed in color. From [42]

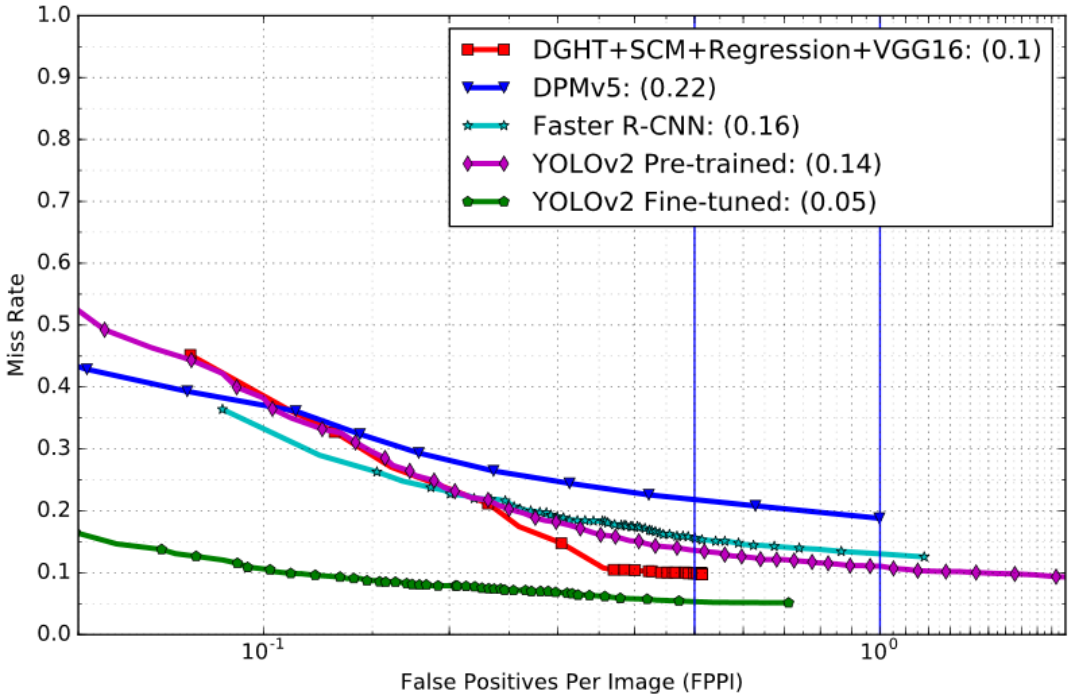


FIGURE 5.22: Comparison of detection results (DET curves) on the IAIR Car test corpus. From [42]

5.6.2.2 CNN Rejection: Error Analysis

As for pedestrian detection, the oracle experiments as well as the error cases of the final car detection pipeline are analyzed.

The results of the oracles (a), (b) and (c) (see Sect. 5.5.1 "Analysis") for the car detection task are presented in Tab. 5.13. As in the pedestrian detection task, the localization oracle shows that the DGHT proposals are accurate because there is no performance gain when ignoring those FP with $0 < \text{IoU} < 0.5$. Again, the discrimination between fore- and background is more important. If the FP with an IoU = 0 would be discarded in the BG vs. FG oracle, a miss rate of 0.10 but at a lower FPPI rate (0.2 instead of 0.5 FPPI) could be achieved.

Assuming a perfect rejection oracle – which selects for each ground truth annotation the candidate with the largest IoU out of the list C generated by the DGHT + SCM + Regression and rejects all other candidates – also the minimal miss rate is quantified in case of perfect proposal rejection (see last line in Tab. 5.13 for the configuration "image scaling, SSE and SCM"). In Fig. 5.23, the minimal miss rate is analyzed as a function of the number $|C|$ of proposals per image with the SCM and with additional bounding box regression.. Compared to a fixed region proposal (RP) scheme (see Sect. 5.5.2.1) with the same additional bounding box regression as in our pipeline, much smaller miss rates are achieved for a given number of candidates (in the regime with at most $1\,000$ candidates per image). Curves for the actual miss rate (including the same CNN-based proposal rejection step as well as grouping and post-processing for the final detection pipeline and the fixed RP scheme) are presented in Fig. 5.24. Using only the top-scoring 200 proposals, the DGHT object detection pipeline achieves the same miss rate at 0.5 FPPI as the best miss rate achieved by the fixed RP scheme (0.12 when using $3\,164$ fixed region proposals).

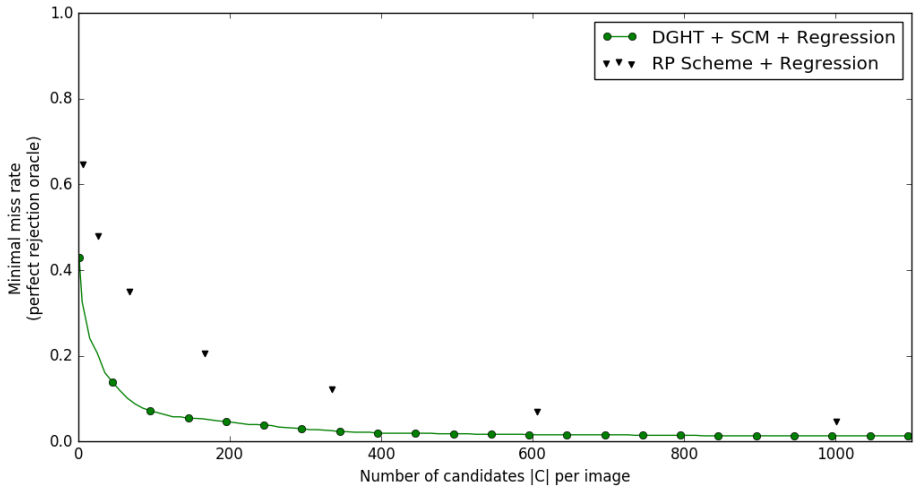


FIGURE 5.23: Minimal miss rate on the IAIR Car test corpus based on an ordered list C of proposals provided by the DGHT+SCM+Regression, as function of the length $|C|$ of the list (setup: model scaling, SSE). From [42]

There is also quite a large gap between the minimal miss rate of 0.02 (88.04% ABO) and the achieved detection result of 0.10 at 0.5 FPPI. Main reasons for these false rejections are: 30% of the FP are detections of car parts, similar to the pedestrian error analysis. 16% show cars, which are not annotated. This might be because of small sizes due to large distances to the camera and therefore blurry or occluded representations. Another 19% are vans or trucks, which are detected by the DGHT pipeline, but not annotated because they do not belong to the class "car". 10% are group detections and the remaining 25% are random background detections. Fig. 5.25 shows examples for false positive detections using the DGHT object detection pipeline.

As for pedestrian detection, most of the FN are included in the DGHT candidates, but were rejected by the CNN: 100 of the 133 FN at 0.5 FPPI (75.2%, with an ABO of 75.9%). Main reasons for these false rejections are: small size (27%), occlusion (25%) and blurry representations or low contrast (23%).

The remaining 33 FN (which are not detected by the DGHT) have an average size of 49×36px, which is roughly a third of the mean DGHT car model size. In addition to the small sizes, these cars are mostly occluded and of low contrast or blurry (see Fig. 5.26).

5.6.2.3 Evaluation on Further Car Database

To address the detection performance of the DGHT car detection pipeline on another car test corpus (different data collection than used in training), detection results are reported on the UIUC Multi-scale car database in Tab. 5.14. Specifically, for comparison purposes, two different configurations of our detection pipeline are used, both including CNN proposal rejection (but no bounding box regression):

Setup 1**:** Image scaling, structured edges, with SCM

Setup 2**:** Model scaling, structured edges, no SCM

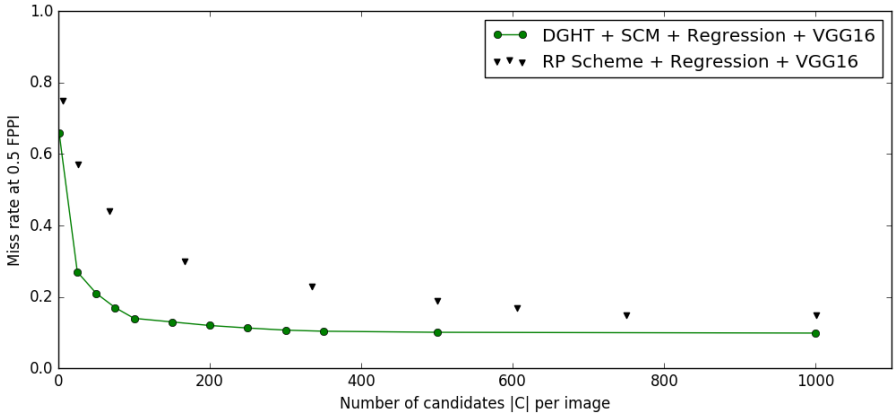


FIGURE 5.24: Actual miss rates at 0.5 FPPI on the IAIR Car test corpus based on an ordered list C of proposals provided by the DGHT+SCM+Regression+VGG16, as function of the length $|C|$ of the list (setup: model scaling, SSE)

Similar to Sect. 5.6.1.6, the DGHT results are compared to other state-of-the-art results reported in the literature (and not to YOLOv2 and Faster R-CNN).

While comparing the DGHT results to the reported state-of-the-art, please note the difference in training data: Whereas the other algorithms at least partly use domain-specific training data (UIUC training data), the DGHT does not. These experiments show that the DGHT car detection pipeline – trained on the IAIR Car training corpus – achieves state-of-the-art performance on this data set using model scaling with only 50 candidates per image. The missing two cars are included in the DGHT candidates but were rejected by the CNN, which has not been retrained on this database. A minimal miss rate of 0.00 (76.3% ABO) at 16 candidates per image (setup 1) and 0.00 (77.6% ABO) at 58 candidates per image (setup 2) are obtained. As for the pedestrian model, this shows the good generalization capability of the DGHT car shape model. All cars could be detected with on average only 16 (image scaling) or 58 (model scaling) proposals if a perfect rejection mechanism is assumed.

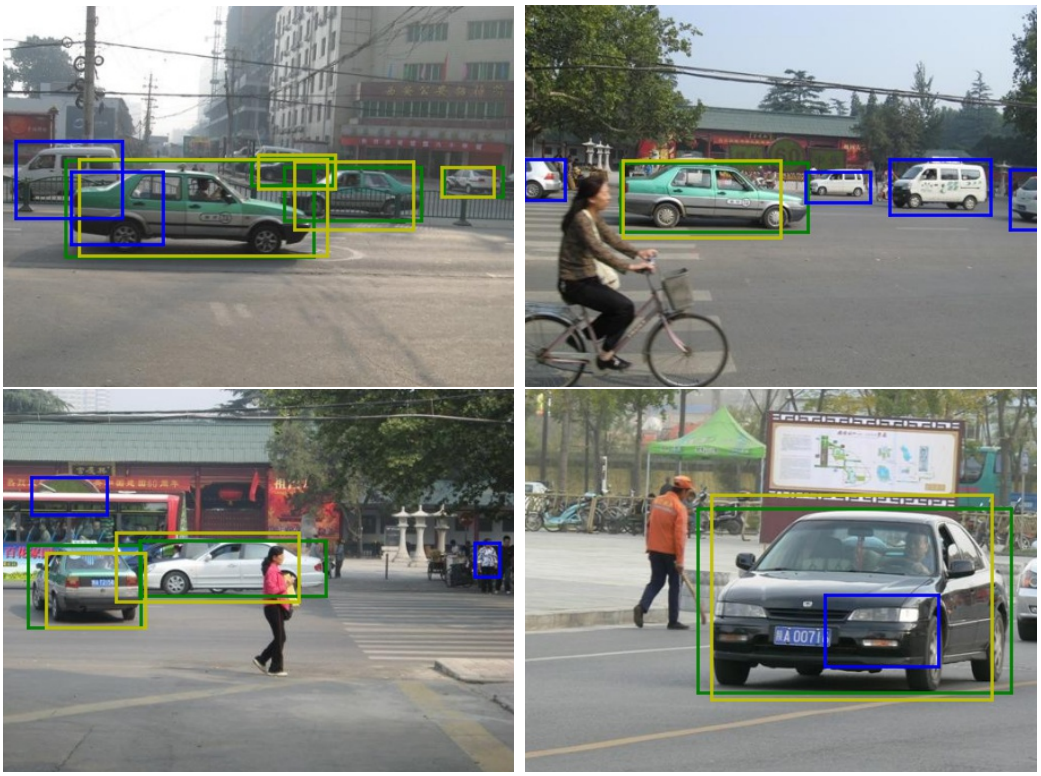


FIGURE 5.25: Example DGHT+SCM+Regression+VGG16 false positive detections on IAIR Car.
(Green) ground truth (yellow) correct detection (blue) FP (red) FN; best viewed in color

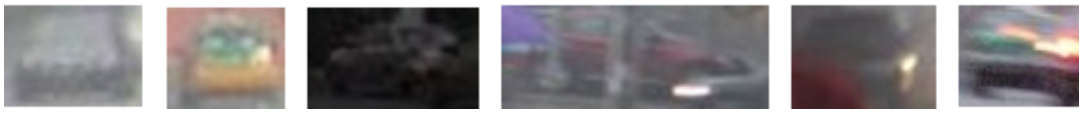


FIGURE 5.26: Examples for false negatives not detected by the DGHT on IAIR Car test corpus. From [42]

TABLE 5.14: Detection performance (Recall at equal error rate (EER)) on the UIUC Multi-scale car database. DGHT: DGHT + VGG16 rejection (but no bounding box regression). Setup 1: image scaling, SSE, SCM; Setup 2: model scaling, SSE, no SCM. Note that no component of the DGHT detection pipeline has been retrained on the UIUC car database. From [42]

Approach	Training Data	Recall at EER
DGHT (Setup 1)	IAIR	97.1%
DGHT (Setup 2)	IAIR	98.6%
Sparse Localized Features [94]	UIUC	90.6%
Cluster Boosted Tree [132]	UIUC	93.5%
ISM [76]	UIUC	95.0%
Image Strip Features [140]	UIUC	96.0%
Hough Forests [45]	UIUC	98.6%
Efficient Subwindow Search [73]	UIUC	98.6%
Perfect Rejection Oracle (c)	Minimal Miss Rate	ABO [%] (proposals)
DGHT (Setup 1)	0.00	76.3 (16)
DGHT (Setup 2)	0.00	77.6 (58)

Chapter 6

Discussion

6.1 Summary of Results, Strengths and Limitations

On the IAIR database, the final DGHT detection pipeline with bounding box regression ("DGHT + SCM + Regression + VGG16") achieves competitive results for pedestrian and car detection, being outperformed only by the YOLOv2 approach fine-tuned on the IAIR Car database, while outperforming or performing similarly to all other investigated approaches.

Without retraining any component, competitive results on TUD Pedestrians for the pedestrian detection task and on the UIUC Multi-scale test set for the car detection task could be shown. The performance on INRIA Person is on par with `ChnFtrs` ([28], Sect. 2.2.2.6.1) and Part Alphabet and Pose Dictionary ([135], see Sect. 2.2.3.2). On all three databases, minimal miss rates of $0.00 - 0.01$ with ABO scores $> 75\%$ were achieved, showing the excellent generalization capability of the DGHT shape models that are used for proposal generation.

The bounding box regression component further improves the ABO scores of the candidates generated by either the DGHT or the DGHT + SCM and generally enables the DGHT object detection pipeline to additionally handle large aspect variability (car front views versus side views) using a single DGHT shape model. The ABO scores of $> 75\%$ (see Fig. 4.7 for a visualization of exemplary IoU scores) for all databases show the good quality of the proposals generated by the DGHT.

For all configurations of the pipeline components, similar overall detection results were achieved (as evaluated on the pedestrian detection task). In particular, the following findings regarding the contribution of the individual components were obtained for pedestrian detection: The model scaling approach is more efficient, as it reduces the voting time of the DGHT CPU implementation by 40% (see Sect. 6.4 for a detailed runtime analysis). In contrast to the image scaling approach, also only a single edge image has to be computed for each test image. The SCM can help in reducing the number of candidates without affecting the overall detection performance. The class-specific Structured Edge Detector is able to already suppress unwanted edges prior to the proposal generation, thus potentially leading to higher detection performance, whereas in case of CNN-based proposal rejection the Canny edge detection is faster, has slightly better representations of low contrast instances and does not need to be trained.

In general, the DGHT is suitable for proposal generation due to the very low false negative rates as indicated by the perfect rejection oracle experiments and the comparably small number of candidates compared to Selective Search [122] (2 000 - 10 000

candidates) or the region proposals of Faster R-CNN [102] (300+ candidates). The generalization capability, in particular concerning the DGHT shape models, has been shown – in addition to the test results obtained in the same domain, namely the IAIR corpus – on two further pedestrian and one further car database.

Regarding pedestrian detection, it has been shown that using less training data for generating the DGHT shape model does not negatively affect the overall detection performance (evaluated up to a training corpus size of 262 images containing 352 pedestrian instances, i.e., 25% of the original training corpus size).

Some problems for the DGHT proposal generation still remain for both pedestrian and car detection and provide room for further improvements: Due to the edge images, the DGHT intrinsically misses those objects which are of low contrast, since they do not generate well-pronounced edges or no edges at all. The CNN sometimes detects object parts (torsos, legs or heads for pedestrians and, e.g., fenders for cars, respectively) as complete detections. Additionally, confusable structures in the background or heavily occluded objects can be misclassified.

In scenarios with extremely large size variability, the model scaling approach used as an efficient alternative to handle object size variability could be stretched to its limits. In case of very small scales, many model points would intersect after having scaled the mean size DGHT shape model, which may lead to an unwanted voting behaviour (multiple model points could cast a vote from a single edge pixel). The opposite applies to very large scales, where the model points are very coarse such that these large gaps might have a negative influence on the voting behaviour. To overcome this, a hybrid approach of image and model scaling as suggested by Dollár et al. [27] could be adapted to the DGHT object detection framework.

6.2 High-Level Comparison with YOLOv2

The following strengths and weaknesses of the DGHT object detection approach as compared to YOLOv2 are reported as initially published in [42]: Since the amount of proposals is restricted by the used (rather coarse) grid structure in YOLOv2, this approach can face difficulties in detecting (groups of) small objects as it detects at most one bounding box center per grid cell [99]. Moreover, because of the fixed input image size and the relatively coarse features used for bounding box prediction [99], YOLOv2 might have problems in detecting rather small objects in large input images, for instance in high-resolution aerial imagery. As advantages, YOLOv2 offers a complete end-to-end training procedure and a very fast processing time per image.

The two-stage DGHT object detection approach can overcome the mentioned weaknesses of YOLOv2 using an efficient proposal generation step regardless of the size of the input image (see. Sect. 6.4 for runtime, complexity and memory consumption). Further advantages of the DGHT approach are the relatively low amount of training material needed to train the DGHT and the SCM as well as to fine-tune the CNNs, and the relatively low amount of flexibly generated proposals, i.e., without a pre-specified grid or number of candidates. All in all, the DGHT approach seems advantageous in particular, if only few training data is available or if large images frequently contain only a few objects or groups of objects.

TABLE 6.1: Evaluation of different pre-trained CNN architectures for proposal rejection in the DGHT detection pipeline ("DGHT+SCM+Regression") on the IAIR Pedestrian test corpus. All components preceding proposal rejection (including bounding box regression, which is in all cases performed with the modified VGG16 network, see Sect. 5.5.1) are identical for the three investigated proposal rejection CNNs. Furthermore, all investigated proposal rejection CNNs are fine-tuned on the same data. For comparison, the result for the YOLOv2 (fine-tuned) detection approach is repeated from Tab. 5.6. Setup: image scaling, SSE, SCM. From [42]

Approach	Training Data	Miss Rate at 0.5 FPPI
DGHT+VGG16 (modified)	IAIR / ImageNet	0.11
DGHT+ResNet50 [62]	IAIR / ImageNet	0.10
DGHT+MobileNet [67]	IAIR / ImageNet	0.14
YOLOv2 Fine-tuned [100]	ImageNet+PASCAL+IAIR	0.10

A disadvantage is that the DGHT detection pipeline is still sequential, in particular lacking an end-to-end training. However, it might be possible to realize the DGHT voting and the SCM by a CNN (see the idea of Girshick et al. to realize the DPM as a CNN [50]), such that the complete detection pipeline might be realized on a CNN basis, including an end-to-end-training. This, however, remains to be investigated in future work.

6.3 Comparison of CNNs for Proposal Rejection

This section was initially published in [42]. Table 6.1 shows the detection results on the IAIR Pedestrian test corpus using three different classification CNNs for proposal rejection, i.e., the modified VGG16 (see Sect. 5.5.1), ResNet50 [62] and MobileNet [67]. All networks have input dimensions of $64 \times 64 \times 3$ and were fine-tuned on the IAIR Pedestrian training set using respective standard Keras network models pre-trained on ImageNet in all cases. The best performance was achieved when using ResNet50 for proposal rejection, while MobileNet is the fastest approach with the lowest memory consumption (see Sect. 6.4 for the exact comparison of runtime and memory consumption). VGG16 represents a good trade-off being nearly as accurate as ResNet50 while achieving a runtime close to MobileNet when replacing the fully connected layers of the standard VGG16 by a global average pooling layer and a fully connected layer as described in Sect. 5.5.1.

In conclusion, there is still potential to optimize the network architecture used for proposal rejection with regard to performance, runtime or memory demand and potential trade-offs. Similar results can be expected for the bounding box regression network, but are beyond the scope of this work.

6.4 Computational and Runtime Analysis

This section was initially published in [42]. Table 6.2 shows the results of a runtime and memory demand analysis for the main components of the DGHT detection pipeline. The used system specifications are as follows:

TABLE 6.2: Runtime and memory demand of the main components of the DGHT detection pipeline on IAIR Pedestrian. "Modified VGG16" refers to replacing the three fully connected layers of the standard VGG16 model by a single fully connected layer (plus a global average pooling layer in case of proposal rejection). From [42]

Processing Step / Approach	GPU?	Runtime [ms]	Memory [MB]
Edge Detection			
Canny Edge Detection	✓	0.77 / image	235
Structured Edge Detection	-	72.08 / image	630
Voting			
(D)GHT	✓	1.75 / scale	230
SCM (optional)	-	480 / scale	200
Regression			
VGG16 (modified)	✓	0.36 / patch	2 347 [batch size: 32]
Proposal Rejection			
VGG16 (modified)	✓	0.34 / patch	2 347 [batch size: 32]
ResNet50[62]	✓	1.93 / patch	8 353 [batch size: 32]
MobileNet[67]	✓	0.21 / patch	986 [batch size: 32]

GPU: NVIDIA Titan X (Pascal), 12GB RAM

CPU: Intel Xeon E5 − 1607 3GHz, 16GB RAM

Runtime and memory demand are reported as averages over all IAIR Pedestrian images, including training and test set, for both feature extraction steps and the (D)GHT voting, and over the IAIR Pedestrian test set for all other components, respectively.

Due to the independent voting of the model points in the DGHT shape model, the DGHT has a high potential for parallelization. Using the parallelized GHT implementation of openCV[1], mean voting times for one model scale on the IAIR Pedestrian edge images of 1.75ms were achieved on a NVIDIA TITAN X (Pascal). Note that the model scales could as well be computed in parallel. As one can see in Eq. 2.6, the worst-case number of voting operations per image or model scale is defined by the number of model points multiplied by the number of edge pixels. In practice, this number is drastically reduced by only allowing votes from combinations, where the model and edge point have similar gradient directions (see Sect. 2.2.1.2).

The optional SCM is currently not available as a GPU implementation and therefore comparably slow. However, as it is a standard Random Forest, it can be easily ported to a GPU as shown by Grahn et al. [52].

Per-patch runtimes are reported for three well-known proposal rejection networks, which were used for the performance comparison in Tab. 6.1. Note that the modified VGG16 architecture – using only a single fully connected layer following the convolutional layers of the standard VGG16 model (plus a global average pooling layer in case of proposal rejection) – has already quite a small average runtime for proposal rejection, due to the large reduction of the number of network parameters.

The total runtime of the DGHT approach highly depends on the number of generated proposals and, of course, on the input image size. According to the analysis presented in Tab. 6.2, on the IAIR Pedestrian data with an image resolution of

[1] https://opencv.org

512×384px, using model scaling with ten scales and Canny edge detection, it should be possible to generate 350 proposals – which are predicted on average on the IAIR Pedestrian data – in less than 15ms per image. These proposals could be evaluated in 245ms using the non-optimal sequential bounding box regression and proposal rejection. Thus, a frame rate of $3 - 4$ frames per second could be achieved on the IAIR Pedestrian corpus. The frame rate could be further increased by a "parallel", i.e., multi-task, architecture for CNN bounding box regression and classification. First investigations show that this would nearly halve the time needed for bounding box regression and classification, i.e., nearly doubling the frame rate (potentially at the expense of a slight performance degradation). Further speed-ups may be achieved by additional CNN optimizations (compare Tab. 6.2), and by realizing the DGHT (and potentially the SCM) by a CNN.

Chapter 7

Conclusions

In a first step of this thesis, the performance and the issues of the DGHT that has successfully been applied in localization tasks with limited variability as, e.g., in the medical image processing context, were analyzed in real-world object detection tasks such as pedestrian and car detection.

It has been shown that analyzing the model point voting pattern using the SCM could improve the single-object detection performance of the DGHT also on real-world tasks. Still, many peaks at confusable background structures show up in the Hough space. This indicates that the SCM by itself is not entirely able to both suppress high Hough scores from irregular voting patterns, i.e., handling large background variability. In addition, the SCM can only handle limited object variability contained in the DGHT shape model. Hence, the assessment of multiple peaks is still hampered.

To this end, approaches to overcome the identified problems have been suggested and implemented as system extensions to the resulting DGHT object detection system. In particular, structured edge detection is suggested to handle the large background variability present in real-world tasks such as pedestrian and car detection as opposed to medical images. Moreover, size variability could be handled either by standard image or efficient DGHT model scaling each with comparably few scales. The detection of multiple instances is done by multiple peak assessment using a subsequent SCM- or CNN-based proposal rejection. An additional bounding box regression component was introduced to further improve the precision of the DGHT proposals and thus the detection accuracy. This component also enabled handling large aspect variability for car detection with a single DGHT shape model containing all views of the training set. In several experiments, the impact of the different components on the detection performance has been thoroughly evaluated leading to a two-stage object detection approach with the DGHT as a Hough-based proposal generation mechanism and a CNN-based candidate refinement and rejection as in the R-CNN "family" [49, 48, 102].

In the final detection system, the DGHT as a proposal generator – in combination with CNN-based bounding box regression and proposal rejection – has been applied to a pedestrian and a car detection task, respectively. Competitive performance to state-of-the-art approaches has been obtained on the IAIR Pedestrian and Car database, with the DGHT being outperformed only by YOLOv2 fine-tuned on the IAIR Car database, while itself outperforming or performing similarly to the other investigated algorithms. Additionally, it was demonstrated that the DGHT framework (trained on IAIR) generalizes well to other data sets used for pedestrian and

car detection. Per image, the DGHT generates between 20 and 350 proposals, depending on the database, the image size and its contents, which is much less than current proposal generation approaches. Furthermore, using only 25% of the pedestrian training images (corresponding to 352 pedestrians) for DGHT and SCM training does not negatively affect the overall detection performance; instead, the shape model seems to be even more generic. The main advantages of the DGHT proposal generation are (a) the relatively low amount of training images needed for training of DGHT and SCM (on the order of 100 or less per variability class), (b) the low minimal miss rates, (c) the relatively low amount of proposals generated per image and (d) the low amount of resources needed at test time.

All in all, the DGHT object detection approach seems advantageous in particular, if only few training data is available or if large images frequently contain only few objects. Moreover, the final pipeline can be used, if the amount of proposals is restricted, e.g., due to complex post-processing, or if there are other qualitative requirements regarding the proposals, e.g., a small upper limit for the number of proposals.

7.1 Future Work

In future work, the very low minimal miss rates motivate to further improve the CNN-based proposal rejection, e.g., by using faster and more accurate classification CNNs as rejection mechanisms. Due to the modular structure of the framework, the CNN rejection component could simply be interchanged by a more powerful network with the input being the only interface that would need to be adapted.

Moreover, it is worthwhile to investigate "parallel" architectures for CNN bounding box regression and classification. These so-called multi-task network architectures allow for combined classification/regression training jointly optimizing both tasks in order to further reduce runtime. This may involve developing strategies to compensate for the potentially mismatched bounding boxes used in proposal classification since the bounding box regression step would then be performed in parallel to the proposal rejection.

In addition, an interesting future line of research is to formulate both the DGHT and the SCM by a (layer of a) CNN, thus enabling to realize the DGHT detection pipeline completely within a CNN framework, including an end-to-end training. Girshick et al., for instance, have presented ideas on how to realize a deformable part model (DPM) as a convolutional layer in a CNN [50].

Further potential aspects could be derived from the R-CNN $\rightarrow$ Fast R-CNN $\rightarrow$ Faster R-CNN improvement chain such as the convolutional feature map, that needs to be computed only once and saves runtime when refining or evaluating candidates.

Appendix A

Basic Concepts

A.1 Integral Images

Initially, summed-area tables were proposed in 1984 by Crow [22] for texture mapping in computer graphics. Viola and Jones successfully applied the ideas of summed-area tables to image processing and invented the term "integral image" [123] based on the work of Lewis [79].

The main advantage of this concept is that the sum of any rectangular region in the input image can be extracted from the integral image in constant time. In particular, if many look-ups are required, this approach can be used for an efficient response computation with respect to certain sub-windows. The integral image at position (x, y) is defined by:

$$I(x, y) = i(x, y) + I(x, y - 1) + I(x - 1, y) - I(x - 1, y - 1) \tag{A.1}$$

where $i(x, y)$ corresponds to the pixel value of the input image i at coordinates (x, y). Thus, the construction of the integral image can be performed in a single pass leading to a complexity of $\mathcal{O}(n)$, i.e., linear complexity. An example input image and the corresponding integral image are presented in Fig. A.1.

Instead of only summing up the intensity values of the input image, also extensions to continuous domains or multi-dimensional input images can be considered. Moreover, e.g., the variance or standard deviation, the covariance or any other desired information could be realized using multiple integral images, e.g., $I(x, y)$ and $I^2(x, y)$.

To extract the sum of intensities of a certain region, here, a rectangle that is spanned by $A = (x_1, y_1)$, $B = (x_2, y_1)$, $C = (x_1, y_2)$, and $D = (x_2, y_2)$, the following look-up is used:

$$\sum_{x_1 < x \leq x_2; y_1 < y \leq y_2} i(x, y) = I(D) + I(A) - I(B) - I(C) \tag{A.2}$$

For the example in Fig. A.2 this would lead to:

$$\sum_{x_1 < x \leq x_2; y_1 < y \leq y_2} i(x, y) = 230 + 53 - 134 - 79 = 70 \tag{A.3}$$

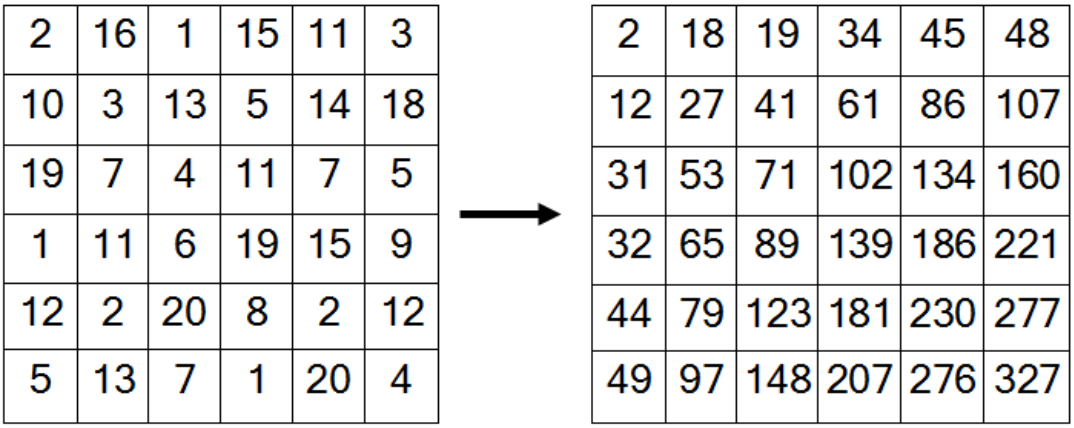


FIGURE A.1: Construction of an integral image. (left) input image (right) integral image

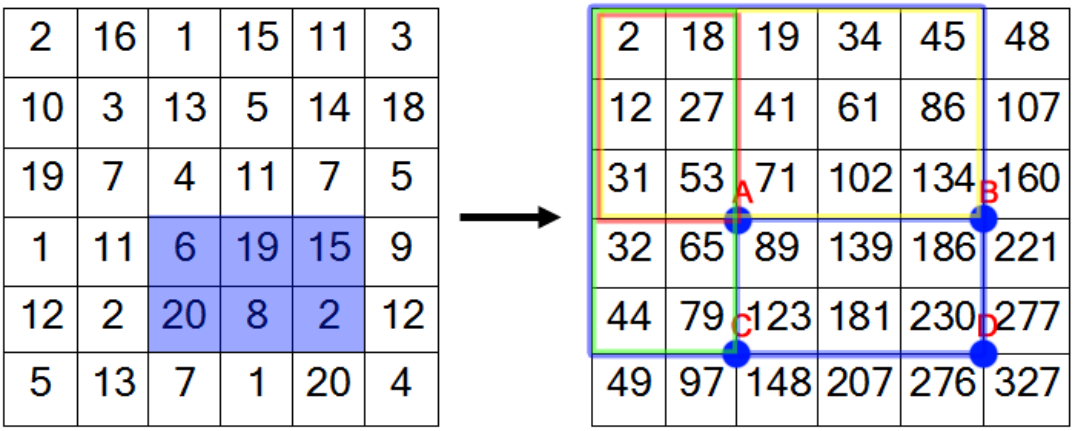


FIGURE A.2: Computation of an area in the integral image. (left) input image (right) integral image. The points A (red rectangle), B (yellow rectangle), C (green rectangle), and D (blue rectangle) spanning the respective rectangles are placed in the lower right corner of the corresponding pixel.

Due to the efficiency of the approach, in particular, if many look-ups are necessary, it is a widely-used concept introduced by Viola and Jones for the fast computation of Haar-like features (see Sect. 2.1.3) for their boosted cascade detector (see Sect. 2.2.2.1).

Apart from that, other feature descriptors such as Speeded Up Robust Features (SURF [9], see Sect. 2.1.5.3), the Image-Strip Features of Zheng and Liang [140] (see Sect. 2.2.2.5) as well as the covariance descriptors used in the "Spatial Pooling" approach of Paisitkriangkrai et al. [98] (see Sect. 2.2.2.8) made use of this concept.

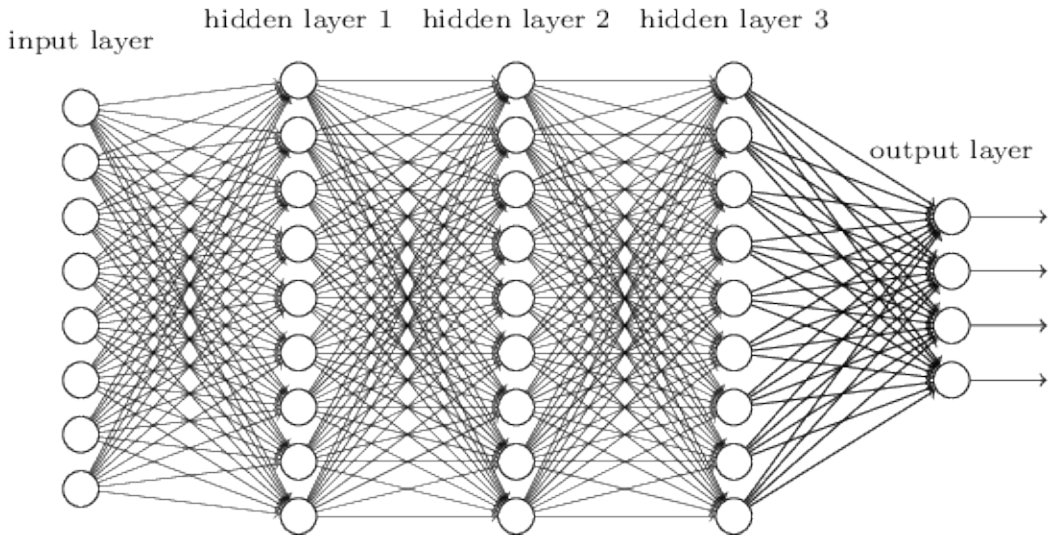


FIGURE A.3: Basic illustration of a standard neural network (multi-layer perceptron) with three hidden layers. From [96]

A.2 Convolutional Neural Networks (CNN)

A.2.1 Introduction and Motivation

For images as input, convolutional neural networks (CNN) provide an efficient way of processing this input data with much fewer parameters than a regular feedforward neural network, e.g., a multi-layer perceptron (MLP).

Given a rather small gray-scale input image of size 128×128, a MLP would already need $16\,384$ weights – one for each connection – to only connect the input layer to a single hidden neuron in the first hidden layer. Moreover, the spatial structure of the input image is lost, since the input data is concatenated into a vector (see Fig. A.3). The number of parameters would increase even further with each additional layer and the overall amount would either require tons of training data to converge, lead to overfitting or both.

In contrast, CNNs keep the spatial structure of the input data and only use a certain number of small filters per convolutional layer (see Sect. A.2.2.1) to be applied to the respective input data leading to a three-dimensionally organized volume of neurons. The width and the height of this volume correspond to the filter size, which is usually much smaller than the input size (usually 3×3 and 5×5, or 1×1 for dimensionality reduction). The depth is simply given by the number of filters, typically multiples of 32, e.g., $\{32, 64, 128, 256, \ldots\}$.

The different layer types and their concepts to support efficient processing of the input data are described in detail in the following Sect. A.2.2.

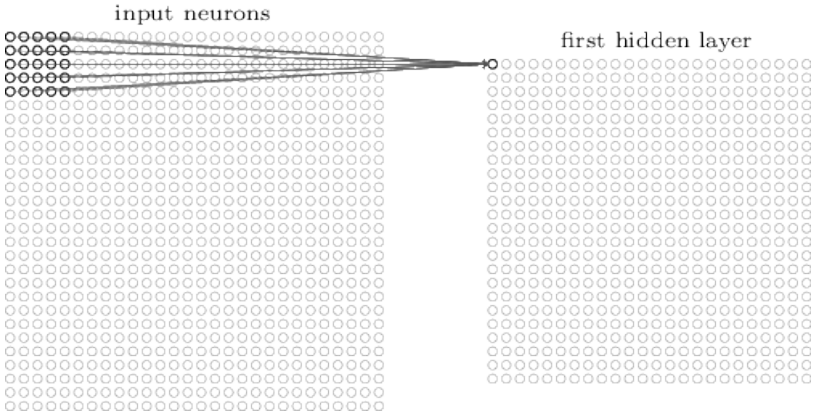


FIGURE A.4: Illustration of the local receptive field for the first output neuron in a convolutional layer (using a filter size of 5×5 and without zero padding). From [96]

A.2.2 Layer Types

A.2.2.1 Convolutional Layers

Instead of connecting each input neuron to each output neuron as in fully connected layers (see Sect. A.2.2.3), each output neuron is only connected to a small region of the input data, the so-called *local receptive field*, corresponding to the used filter size in the current layer (e.g., 3×3). As for usual neural networks, a single weight is learned for each of the nine connections. In order to maintain the spatial structure, this receptive field is slid over the input image using a specified stride, for instance, a step size of 1 in each dimension (see Fig. A.4). If the output dimensions should correspond to the input dimensions, a (zero) padding can be used.

When sliding the receptive field over the input data, the same nine weights are used to compute the response of the respective output neuron by merely computing the dot product between the pixel values of the input data which are within the receptive field (using all input channels) with the corresponding filter entries (again using all channels). Strictly speaking, this refers to a cross-correlation of the input data with the filter, which corresponds to a convolution of the input data with a flipped filter (if the filter is symmetric). The output of the convolution (or cross-correlation) operation for a given filter, i.e., the response of the input to that filter, is called a "feature map". The concept of using the same weights at each position is called *parameter sharing* and means that a certain feature is searched for at each location. Sharing these filters is a main concept to drastically reduce the number of parameters of the network. This allows for detecting several features in the input data per layer and still using much fewer parameters.

The responses of each local receptive field of each filter are passed through an activation function to compute the activation in each output neuron. This also accounts for introducing non-linearities. In the beginning, sigmoid activation functions have been used, for instance, $\tanh(x)$. Currently, rectified linear units (ReLU) are used due to their efficiency. A ReLU simply computes the activation of an input x by calculating $\max(0, x)$.

To summarize, a 2D activation map is computed per filter and then stacked over all filters to create the output volume. Remember from Sect. A.2.1, using a MLP, an

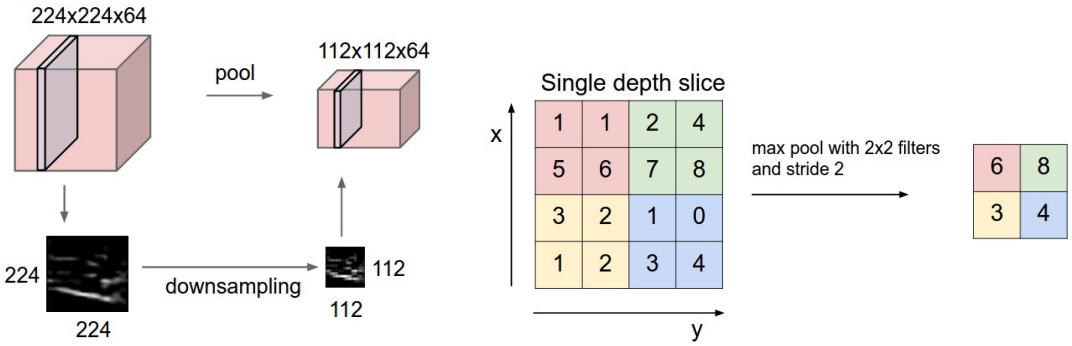


FIGURE A.5: Illustration of the max pooling layer. From [80]

image of size 128×128 would need $16\,384$ weights for each hidden neuron. Even when detecting 128 different features in a convolutional layer using 3×3 filters, only $9 \times 128 = 1\,152$ weights are needed regardless of the input image size.

A.2.2.2 Pooling Layers

Pooling layers are periodically used to simplify or condense the activation information as generated by the previous convolutional layer(s). This leads to a reduction of the number of parameters needed in subsequent layers as well as to avoid overfitting and increases the receptive field size as seen by subsequent neurons.

The approach is rather simple: A spatial area, usually 2×2 is taken into account per location. If the pooling should not be overlapping, the stride is set accordingly to 2. For each location, only the maximum value inside of the pooling area is output in case of max pooling (see Fig. A.5). Alternatively, average or L2-norm pooling could be used, but max pooling can be computed efficiently and often achieves good results. The sample spatial extent would lead to an output volume of half the size of each input dimension, while the input depth remains since the pooling operation is applied to each feature map independently.

A.2.2.3 Fully Connected Layers

As in standard neural networks, in fully connected (FC) layers every output neuron is connected to every neuron of the preceding layer. Thus, these layers are used at the end of CNNs to create or pre-process the final output. In classification CNNs, the number of output neurons in the last FC layer usually corresponds to the number of classes. There may be also preceding FC layers with more neurons. Often, the activations of the last FC layer are post-processed by a softmax function to generate the final classification probabilities.

A.2.3 Typical Architecture

Typically, CNN architectures for classification tasks consist of an input image layer and several stacks of one to three convolutional layers with ReLU activation that are followed by a pooling layer. The level of abstraction gets higher with smaller spatial extent of the output volume. If the desired degree is reached, one or more

fully connected layers are used to generate the class-specific activations. A softmax function could then be used to create the final class probabilities.

Instead of the usual sequential stacking of layers, recent approaches such as ResNet [62] or Inception [119] introduce other concepts, for instance, shortcut connections, inception modules or global average pooling [81, 119] that further reduce the number of parameters.

A.2.4 Further Reading

For further information on convolutional neural networks, concepts, training procedures and so on, the reader is referred to [96] or the lecture of the University of Stanford [80]. In addition, Schmidhuber provided an exhaustive overview of concepts and extensions until 2015 in [113].

Bibliography

[1] URL: http://www.vision.caltech.edu/Image_Datasets/\-CaltechPedestrians/ (Last retrieved on Dec. 10, 2018).

[2] Agarwal, Shivani and Awan, Aatif and Roth, Dan. "Learning to Detect Objects in Images via a Sparse, Part-based Representation". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)* 26.11 (2004), pp. 1475–1490.

[3] Alexe, Bogdan and Deselaers, Thomas and Ferrari, Vittorio. "Measuring the Objectness of Image Windows". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)* 34.11 (2012), pp. 2189–2202.

[4] Alexe, Bogdan and Deselaers, Thomas and Ferrari, Vittorio. "What is an Object?" In: *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE. 2010, pp. 73–80.

[5] Andriluka, Mykhaylo and Roth, Stefan and Schiele, Bernt. "People-Tracking-by-Detection and People-Detection-by-Tracking". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE. 2008, pp. 1–8.

[6] Angelova, Anelia and Krizhevsky, Alex and Vanhoucke, Vincent and Ogale, Abhijit S and Ferguson, Dave. "Real-Time Pedestrian Detection with Deep Network Cascades". In: *Proceedings of the British Machine Vision Conference (BMVC)*. Vol. 2. 2015, p. 4.

[7] Arbelaez, Pablo and Maire, Michael and Fowlkes, Charless and Malik, Jitendra. "Contour Detection and Hierarchical Image Segmentation". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)* 33.5 (2011), pp. 898–916.

[8] Ballard, Dana H. "Generalizing the Hough Transform to Detect Arbitrary Shapes". In: *Pattern Recognition* 13.2 (1981), pp. 111–122.

[9] Bay, Herbert and Tuytelaars, Tinne and Van Gool, Luc. "SURF: Speeded Up Robust Features". In: *European Conference on Computer Vision (ECCV)*. Springer. 2006, pp. 404–417.

[10] Benenson, Rodrigo and Mathias, Markus and Timofte, Radu and Van Gool, Luc. "Pedestrian Detection at 100 Frames Per Second". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE. 2012, pp. 2903–2910.

[11] Benenson, Rodrigo and Mathias, Markus and Tuytelaars, Tinne and Van Gool, Luc. "Seeking the Strongest Rigid Detector". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2013, pp. 3666–3673.

[12] Benenson, Rodrigo and Omran, Mohamed and Hosang, Jan and Schiele, Bernt. "Ten Years of Pedestrian Detection, What Have We Learned?" In: *Proceedings of the European Conference on Computer Vision (ECCV)*. Springer. 2014, pp. 613–627.

[13] Beyerlein, Peter. "Discriminative Model Combination". In: *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*. Vol. 1. IEEE. 1998, pp. 481–484.

[14] Blier, Leonard. "A Brief Report of the Heuritech Deep Learning Meetup 5". In: *blog.heuritech.com* (2016). URL: https://blog.heuritech.com/2016/02/29/a-brief-report-of-the-heuritech-deep-learning-meetup-5/ (Last retrieved on Dec. 10, 2018).

[15] Breiman, Leo. "Random Forests". In: *Machine Learning* 45.1 (2001), pp. 5–32.

[16] Brown, Matthew and Lowe, David G. "Invariant Features From Interest Point Groups". In: *Proceedings of the British Machine Vision Conference (BMVC)*. Vol. 4. 2002.

[17] Calonder, Michael and Lepetit, Vincent and Strecha, Christoph and Fua, Pascal. "BRIEF: Binary Robust Independent Elementary Features". In: *Proceedings of the European Conference on Computer Vision (ECCV)*. Springer. 2010, pp. 778–792.

[18] Canny, John. "A Computational Approach to Edge Detection". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)* 6 (1986), pp. 679–698.

[19] Cheng, Yizong. "Mean Shift, Mode Seeking, and Clustering". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)* 17.8 (1995), pp. 790–799.

[20] Cireşan, Dan and Meier, Ueli and Schmidhuber, Jürgen. "Multi-column Deep Neural Networks for Image Classification". In: *arXiv preprint arXiv:1202.2745* (2012).

[21] Costea, Daniel Arthur and Nedevschi, Sergiu. "Semantic Channels for Fast Pedestrian Detection". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016, pp. 2360–2368.

[22] Crow, Franklin C. "Summed-Area Tables for Texture Mapping". In: *ACM SIGGRAPH Computer Graphics*. Vol. 18. 3. ACM. 1984, pp. 207–212.

[23] Dalal, Navneet and Triggs, Bill. "Histograms of Oriented Gradients for Human Detection". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. Vol. 1. IEEE. 2005, pp. 886–893.

[24] Deng, Jia and Dong, Wei and Socher, Richard and Li, Li-Jia and Li, Kai and Fei-Fei, Li. "ImageNet: A Large-Scale Hierarchical Image Database". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE. 2009, pp. 248–255.

[25] Piotr Dollar, Zhuowen Tu, and Serge Belongie. "Supervised Learning of Edges and Object Boundaries". In:

[26] Dollár, Piotr and Appel, Ron and Belongie, Serge and Perona, Pietro. "Fast Feature Pyramids for Object Detection". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)* 36.8 (2014), pp. 1532–1545.

[27] Dollár, Piotr and Belongie, Serge J and Perona, Pietro. "The Fastest Pedestrian Detector in the West". In: *Proceedings of the British Machine Vision Conference (BMVC)*. Vol. 2. 3. 2010, p. 7.

[28] Dollár, Piotr and Tu, Zhuowen and Perona, Pietro and Belongie, Serge. "Integral Channel Features". In: *Proceedings of the British Machine Vision Conference (BMVC)*. BMVC Press, 2009.

[29] Dollár, Piotr and Wojek, Christian and Schiele, Bernt and Perona, Pietro. "Pedestrian Detection: An Evaluation of the State of the Art". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)* 34.4 (2012), pp. 743–761.

[30] Dollár, Piotr and Zitnick, C Lawrence. "Fast Edge Detection Using Structured Forests". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)* 37.8 (2015), pp. 1558–1570.

[31] Du, Xinxin and Ang Jr, Marcelo H and Karaman, Sertac and Rus, Daniela. "A General Pipeline for 3D Detection of Vehicles". In: *arXiv preprint arXiv:-1803.00387* (2018).

[32] Duda, Richard O and Hart, Peter E. "Use of the Hough Transformation to Detect Lines and Curves in Pictures". In: *Communications of the ACM* 15.1 (1972), pp. 11–15.

[33] Erhan, Dumitru and Szegedy, Christian and Toshev, Alexander and Anguelov, Dragomir. "Scalable Object Detection Using Deep Neural Networks". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2014, pp. 2147–2154.

[34] Everingham, Mark and Van Gool, Luc and Williams, Christopher KI and Winn, John and Zisserman, Andrew. "The PASCAL Visual Object Classes (VOC) Challenge". In: *International Journal of Computer Vision (IJCV)* 88.2 (2010), pp. 303–338.

[35] Fei-Fei, Li and Fergus, Rob and Perona, Pietro. "One-shot Learning of Object Categories". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)* 28.4 (2006), pp. 594–611.

[36] Felzenszwalb, P. F. and Girshick, R. B. and McAllester, D. and Ramanan, D. "Object Detection with Discriminatively Trained Part Based Models". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)* 32.9 (2010), pp. 1627–1645.

[37] Felzenszwalb, Pedro and McAllester, David and Ramanan, Deva. "A Discriminatively Trained, Multiscale, Deformable Part Model". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition(CVPR)*. IEEE. 2008, pp. 1–8.

[38] Fischler, Martin A and Elschlager, Robert A. "The Representation and Matching of Pictorial Structures". In: *IEEE Transactions on Computers* 100.1 (1973), pp. 67–92.

[39] Freund, Yoav and Schapire, Robert E. "A Decision-Theoretic Generalization of Online Learning and an Application to Boosting". In: *Journal of Computer and System Sciences* 55.1 (1997), pp. 119–139.

[40] Funayama, Ryuji and Yanagihara, Hiromichi and Van Gool, Luc and Tuytelaars, Tinne and Bay, Herbert. *Robust Interest Point Detector and Descriptor*. US Patent 8,165,401. 2012.

[41] Gabriel, Eric and Hahmann, Ferdinand and Böer, Gordon and Schramm, Hauke and Meyer, Carsten. "Structured Edge Detection for Improved Object Localization Using the Discriminative Generalized Hough Transform". In: *Proceedings of the 11th Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications - Volume 4: VISAPP*. 2016, pp. 393–402. DOI: https://doi.org/10.5220/0005722803930402.

[42] Gabriel, Eric and Schleiss, Michael and Schramm, Hauke and Meyer, Carsten. "Analysis of the Discriminative Generalized Hough Transform as a Proposal Generator for a Deep Network in Automatic Pedestrian and Car Detection". In: *Journal of Electronic Imaging* 27.5 (2018). 051228. DOI: https://doi.org/10.1117/1.JEI.27.5.051228.

[43] Gabriel, Eric and Schramm, Hauke and Meyer, Carsten. "Analysis of the Discriminative Generalized Hough Transform for Pedestrian Detection". In: *Battiato S., Gallo G., Schettini R., Stanco F. (eds) Image Analysis and Processing - ICIAP 2017. ICIAP 2017. Lecture Notes in Computer Science, vol 10485*. Springer, Cham. 2017, pp. 104–115. DOI: https://doi.org/10.1007/978-3-319-68548-9_10.

[44] Gabriel, Eric and Schramm, Hauke and Meyer, Carsten. "The Discriminative Generalized Hough Transform as a Proposal Generator for a Deep Network in Automatic Pedestrian Localization". In: *Proceedings of the 13th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications - Volume 5: VISAPP*. 2018. DOI: https://doi.org/10.5220/0006542401690176.

[45] Gall, Juergen and Lempitsky, Victor. "Class-specific Hough Forests for Object Detection". In: *Decision Forests for Computer Vision and Medical Image Analysis*. Springer, 2013, pp. 143–157.

[46] Gionis, Aristides and Indyk, Piotr and Motwani, Rajeev and others. "Similarity Search in High Dimensions Via Hashing". In: *Proceedings of the International Conference on Very Large Data Bases (VLDB)*. Vol. 99. 6. 1999, pp. 518–529.

[47] Girshick, R. B. and Felzenszwalb, P. F. and McAllester, D. *Discriminatively Trained Deformable Part Models, Release 5*. URL: http://people.cs.\-uchicago.edu/~rbg/latent-release5/ (Last retrieved on Dec. 10, 2018).

[48] Girshick, Ross. "Fast R-CNN". In: *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*. 2015, pp. 1440–1448.

[49] Girshick, Ross and Donahue, Jeff and Darrell, Trevor and Malik, Jitendra. "Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2014, pp. 580–587.

[50] Girshick, Ross and Iandola, Forrest and Darrell, Trevor and Malik, Jitendra. "Deformable Part Models are Convolutional Neural Networks". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2015, pp. 437–446.

[51] Glorot, Xavier and Bengio, Yoshua. "Understanding the Difficulty of Training Deep Feedforward Neural Networks". In: *Proceedings of the International Conference on Artificial Intelligence and Statistics (AISTATS)*. 2010, pp. 249–256.

[52] Grahn, Håkan and Lavesson, Niklas and Lapajne, Mikael Hellborg and Slat, Daniel. "CudaRF: a CUDA-based Implementation of Random Forests". In: *Proceedings of the 9th IEEE/ACS International Conference on Computer Systems and Applications (AICCSA)*. IEEE. 2011, pp. 95–101.

[53] Haar, Alfred. "Zur Theorie der Orthogonalen Funktionensysteme". In: *Mathematische Annalen* 69.3 (1910), pp. 331–371.

[54] Hahmann, Ferdinand and Berger, Inga and Ruppertshofen, Heike and Deserno, Thomas and Schramm, Hauke. "Bone Age Assessment Using the Classifying Generalized Hough Transform". In: *German Conference on Pattern Recognition*. Springer. 2013, pp. 313–322.

[55] Hahmann, Ferdinand and Böer, Gordon and Gabriel, Eric and Deserno, Thomas M and Meyer, Carsten and Schramm, Hauke. "Classification of Voting Patterns to Improve the Generalized Hough Transform for Epiphyses Localization". In: *Medical Imaging 2016: Computer-Aided Diagnosis*. Vol. 9785. International Society for Optics and Photonics. 2016.

[56] Hahmann, Ferdinand and Böer, Gordon and Gabriel, Eric and Meyer, Carsten and Schramm, Hauke. "A Shape Consistency Measure for Improving the Generalized Hough Transform". In: *Proceedings of the International Conference on Computer Vision Theory and Applications (VISAPP)*. 2015.

[57] Hahmann, Ferdinand and Ruppertshofen, Heike and Böer, Gordon and Stannarius, Ralf and Schramm, Hauke. *Eye Localization Using the Discriminative*

Generalized Hough Transform. Conference paper presentation at the DAGM-OAGM, Graz, Austria. 2012.

[58] Harris, Chris and Stephens, Mike. "A Combined Corner and Edge Detector". In: *Alvey Vision Conference*. Vol. 15. 50. Citeseer. 1988, pp. 10–5244.

[59] Hassaballah, M and Abdelmgeid, Aly Amin and Alshazly, Hammam A. "Image Features Detection, Description and Matching". In: *Image Feature Detectors and Descriptors*. Springer, 2016, pp. 11–45.

[60] He, Kaiming and Gkioxari, Georgia and Dollár, Piotr and Girshick, Ross. "Mask R-CNN". In: *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*. IEEE. 2017, pp. 2980–2988.

[61] He, Kaiming and Sun, Jian. "Convolutional Neural Networks at Constrained Time Cost". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE. 2015, pp. 5353–5360.

[62] He, Kaiming and Zhang, Xiangyu and Ren, Shaoqing and Sun, Jian. "Deep Residual Learning for Image Recognition". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2016, pp. 770–778.

[63] He, Kaiming and Zhang, Xiangyu and Ren, Shaoqing and Sun, Jian. "Delving Deep Into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification". In: *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*. 2015, pp. 1026–1034.

[64] He, Kaiming and Zhang, Xiangyu and Ren, Shaoqing and Sun, Jian. "Spatial Pyramid Pooling in Deep Convolutional Networks for Visual Recognition". In: *Proceedings of the European Conference on Computer Vision (ECCV)*. Springer. 2014, pp. 346–361.

[65] Hosang, Jan and Benenson, Rodrigo and Dollár, Piotr and Schiele, Bernt. "What Makes for Effective Detection Proposals?" In: *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)* 38.4 (2016), pp. 814–830.

[66] Hough, Paul VC. *Method and Means for Recognizing Complex Patterns*. US Patent 3,069,654. 1962.

[67] Howard, Andrew G and Zhu, Menglong and Chen, Bo and Kalenichenko, Dmitry and Wang, Weijun and Weyand, Tobias and Andreetto, Marco and Adam, Hartwig. "MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications". In: *arXiv preprint arXiv:1704.04861* (2017).

[68] Ioffe, Sergey and Szegedy, Christian. "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift". In: *arXiv preprint arXiv:1502.03167* (2015).

[69] Ke, Yan and Sukthankar, Rahul. "PCA-SIFT: A More Distinctive Representation for Local Image Descriptors". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. Vol. 2. IEEE. 2004.

[70] Kingma, Diederik and Ba, Jimmy. "Adam: A Method for Stochastic Optimization". In: *arXiv preprint arXiv:1412.6980* (2014).

[71] Krizhevsky, Alex and Sutskever, Ilya and Hinton, Geoffrey E. "ImageNet Classification with Deep Convolutional Neural Networks". In: *Advances in Neural Information Processing Systems (NIPS)*. 2012, pp. 1097–1105.

[72] Kukich, Karen. "Techniques for Automatically Correcting Words in Text". In: *ACM Computing Surveys (CSUR)* 24.4 (1992), pp. 377–439.

[73] Lampert, Christoph H and Blaschko, Matthew B and Hofmann, Thomas. "Beyond Sliding Windows: Object Localization by Efficient Subwindow Search". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE. 2008, pp. 1–8.

[74] LeCun, Yann and Boser, Bernhard and Denker, John S and Henderson, Donnie and Howard, Richard E and Hubbard, Wayne and Jackel, Lawrence D. "Backpropagation Applied to Handwritten Zip Code Recognition". In: *Neural Computation* 1.4 (1989), pp. 541–551.

[75] Leibe, Bastian and Leonardis, Ales and Schiele, Bernt. "Combined Object Categorization and Segmentation With an Implicit Shape Model". In: *Workshop on Statistical Learning in Computer Vision at ECCV*. Vol. 2. 5. 2004, p. 7.

[76] Leibe, Bastian and Leonardis, Aleš and Schiele, Bernt. "Robust Object Detection with Interleaved Categorization and Segmentation". In: *International Journal of Computer Vision (IJCV)* 77.1-3 (2008), pp. 259–289.

[77] Lenc, Karel and Vedaldi, Andrea. "R-CNN minus R". In: *Proceedings of the British Machine Vision Conference (BMVC)*. BMVC Press, 2015.

[78] Levenshtein, Vladimir I. "Binary Codes Capable of Correcting Deletions, Insertions, and Reversals". In: *Soviet Physics Doklady*. Vol. 10. 8. 1966, pp. 707–710.

[79] Lewis, John P. "Fast Template Matching". In: *Vision Interface*. Vol. 95. 120123. 1995, pp. 15–19.

[80] Li, Fei-Fei and Johnson, Justin and Yeung, Serena and Karpathy, Andrej. "Convolutional Neural Networks for Visual Recognition". In: (2018). URL: `http://cs231n.github.io/` (Last retrieved on Dec. 10, 2018).

[81] Lin, Min and Chen, Qiang and Yan, Shuicheng. "Network in Network". In: *arXiv preprint arXiv:1312.4400* (2013).

[82] Lin, Tsung-Yi and Dollár, Piotr and Girshick, Ross B and He, Kaiming and Hariharan, Bharath and Belongie, Serge J. "Feature Pyramid Networks for Object Detection". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. Vol. 1. 2. 2017, p. 4.

[83] Lin, Tsung-Yi and Goyal, Priya and Girshick, Ross and He, Kaiming and Dollár, Piotr. "Focal Loss for Dense Object Detection". In: *arXiv preprint arXiv:1708.02002* (2017).

[84] Lindeberg, T. "Scale Invariant Feature Transform". In: *Scholarpedia* 7.5 (2012). Revision #153939, p. 10491. DOI: `10.4249/scholarpedia.10491`.

[85] Lindeberg, Tony. "Scale Selection Properties of Generalized Scale-Space Interest Point Detectors". In: *Journal of Mathematical Imaging and Vision* 46.2 (2013), pp. 177–210.

[86] Liu, Wei and Anguelov, Dragomir and Erhan, Dumitru and Szegedy, Christian and Reed, Scott and Fu, Cheng-Yang and Berg, Alexander C. "SSD: Single Shot Multibox Detector". In: *Proceedings of the European Conference on Computer Vision (ECCV)*. Springer. 2016, pp. 21–37.

[87] Lowe, David G. "Object Recognition From Local Scale-Invariant Features". In: *Proceedings of the IEEE Conference on Computer Vision (ICCV)*. Vol. 2. IEEE. 1999, pp. 1150–1157.

[88] Lv, Qin and Josephson, William and Wang, Zhe and Charikar, Moses and Li, Kai. "Multi-probe LSH: Efficient Indexing for High-dimensional Similarity Search". In: *Proceedings of the International Conference on Very Large Data Bases (VLDB)*. VLDB Endowment. 2007, pp. 950–961.

[89] Maini, Raman and Aggarwal, Himanshu. "Study and Comparison of Various Image Edge Detection Techniques". In: *International Journal of Image Processing (IJIP)* 3.1 (2009), pp. 1–11.

[90] Mao, Jiayuan and Xiao, Tete and Jiang, Yuning and Cao, Zhimin. "What Can Help Pedestrian Detection?" In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE. 2017.

[91] Marin, Javier and Vázquez, David and López, Antonio M and Amores, Jaume and Leibe, Bastian. "Random Forests of Local Experts for Pedestrian Detection". In: *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*. 2013, pp. 2592–2599.

[92] Martin, David R and Fowlkes, Charless C and Malik, Jitendra. "Learning to Detect Natural Image Boundaries Using Local Brightness, Color, and Texture Cues". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)* 26.5 (2004), pp. 530–549.

[93] Mohan, Anuj and Papageorgiou, Constantine and Poggio, Tomaso. "Example-based Object Detection in Images by Components". In: *IEEE Transactions on Pattern Analysis & Machine Intelligence (PAMI)* 4 (2001), pp. 349–361.

[94] Mutch, Jim and Lowe, David G. "Multiclass Object Recognition with Sparse, Localized Features". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. Vol. 1. IEEE. 2006, pp. 11–18.

[95] Navarro, Gonzalo. "A Guided Tour to Approximate String Matching". In: *ACM Computing Surveys (CSUR)* 33.1 (2001), pp. 31–88.

[96] Nielsen, Michael A. *Neural Networks and Deep Learning*. 2015. URL: `http://neuralnetworksanddeeplearning.com/` (Last retrieved on Dec. 10, 2018).

[97] Ohn-Bar, Eshed and Trivedi, Mohan M. "Detection and Localization with Multi-scale Models". In: *Proceedings of the 23rd International Conference on Pattern Recognition (ICPR)*. IEEE. 2016, pp. 1382–1387.

[98] Paisitkriangkrai, Sakrapee and Shen, Chunhua and van den Hengel, Anton. "Strengthening the Effectiveness of Pedestrian Detection with Spatially Pooled Features". In: *Proceedings of the European Conference on Computer Vision (ECCV)*. Springer. 2014, pp. 546–561.

[99] Redmon, Joseph and Divvala, Santosh and Girshick, Ross and Farhadi, Ali. "You Only Look Once: Unified, Real-time Object Detection". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016, pp. 779–788.

[100] Redmon, Joseph and Farhadi, Ali. "YOLO9000: Better, Faster, Stronger". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017.

[101] Ren, Jimmy and Chen, Xiaohao and Liu, Jianbo and Sun, Wenxiu and Pang, Jiahao and Yan, Qiong and Tai, Yu-Wing and Xu, Li. "Accurate Single Stage Detector Using Recurrent Rolling Convolution". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017.

[102] Ren, Shaoqing and He, Kaiming and Girshick, Ross and Sun, Jian. "Faster R-CNN: Towards Real-time Object Detection With Region Proposal Networks". In: *Advances in Neural Information Processing Systems (NIPS)*. 2015, pp. 91–99.

[103] Rosenfeld, Amir and Zemel, Richard and Tsotsos, John K. "The Elephant in the Room". In: *arXiv preprint arXiv:1808.03305* (2018).

[104] Rosin, Paul L. "Measuring Corner Properties". In: *Computer Vision and Image Understanding (CVIU)* 73.2 (1999), pp. 291–307.

[105] Rosten, Edward and Drummond, Tom. "Machine Learning for High-Speed Corner Detection". In: *European Conference on Computer Vision (ECCV)*. Springer. 2006, pp. 430–443.

[106] Rublee, Ethan and Rabaud, Vincent and Konolige, Kurt and Bradski, Gary. "ORB: An Efficient Alternative to SIFT or SURF". In: *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*. IEEE. 2011, pp. 2564–2571.

[107] Ruppertshofen, Heike. *Automatic Modeling of Anatomical Variability for Object Localization in Medical Images*. BoD–Books on Demand, 2013.

[108] Ruppertshofen, Heike and Lorenz, Cristian and Rose, Georg and Schramm, Hauke. "Discriminative Generalized Hough Transform for Object Localization in Medical Images". In: *International Journal of Computer Assisted Radiology and Surgery* 8.4 (2013), pp. 593–606.

[109] Ruppertshofen, Heike and Lorenz, Cristian and Schmidt, Sarah and Beyerlein, Peter and Salah, Zein and Rose, Georg and Schramm, Hauke. "Discriminative Generalized Hough Transform for Localization of Joints in the Lower Extremities". In: *Computer Science – Research and Development* 26.1-2 (2011), pp. 97–105.

[110] Schapire, Robert E and Singer, Yoram. "Improved Boosting Algorithms Using Confidence-Rated Predictions". In: *Machine Learning* 37.3 (1999), pp. 297–336.

[111] Schleiss, Michael. "Investigations on Feature and Classifier Performance for Improved Pedestrian Localization". MA thesis. Kiel University of Applied Sciences, July 2017.

[112] Schmid, Cordelia and Mohr, Roger and Bauckhage, Christian. "Evaluation of Interest Point Detectors". In: *International Journal of Computer Vision (IJCV)* 37.2 (2000), pp. 151–172.

[113] Schmidhuber, Jürgen. "Deep Learning in Neural Networks: An Overview". In: *Neural Networks* 61 (2015), pp. 85–117.

[114] Sermanet, Pierre and Eigen, David and Zhang, Xiang and Mathieu, Michaël and Fergus, Rob and LeCun, Yann. "OverFeat: Integrated Recognition, Localization and Detection using Convolutional Networks". In: *arXiv preprint arXiv:1312.6229* (2013).

[115] Serre, Thomas and Wolf, Lior and Poggio, Tomaso. "Object Recognition with Features Inspired by Visual Cortex". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE. 2005.

[116] Shapiro, Samuel Sanford and Wilk, Martin B. "An Analysis of Variance Test for Normality (Complete Samples)". In: *Biometrika* 52.3/4 (1965), pp. 591–611.

[117] Simonyan, Karen and Zisserman, Andrew. "Very Deep Convolutional Networks for Large-scale Image Recognition". In: *Proceedings of the International Conference on Learning Representations (ICLR)*. 2015.

[118] Singh, Saurabh and Gupta, Abhinav and Efros, Alexei A. "Unsupervised Discovery of Mid-level Discriminative Patches". In: *Proceedings of the Europoean Conference on Computer Vision (ECCV)*. Springer, 2012, pp. 73–86.

[119] Szegedy, Christian and Ioffe, Sergey and Vanhoucke, Vincent and Alemi, Alexander A. "Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning". In: *AAAI*. Vol. 4. 2017, p. 12.

[120] Szegedy, Christian and Zaremba, Wojciech and Sutskever, Ilya and Bruna, Joan and Erhan, Dumitru and Goodfellow, Ian and Fergus, Rob. "Intriguing Properties of Neural Networks". In: *arXiv preprint arXiv:1312.6199* (2013).

[121] Tuzel, Oncel and Porikli, Fatih and Meer, Peter. "Region Covariance: A Fast Descriptor for Detection and Classification". In: *Proceedings of the European Conference on Computer Vision (ECCV)*. Springer. 2006, pp. 589–600.

[122] Uijlings, Jasper RR and Van De Sande, Koen EA and Gevers, Theo and Smeulders, Arnold WM. "Selective Search for Object Recognition". In: *International Journal of Computer Vision (IJCV)* 104.2 (2013), pp. 154–171.

[123] Viola, Paul and Jones, Michael. "Rapid Object Detection Using a Boosted Cascade of Simple Features". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. Vol. 1. IEEE. 2001, pp. I–I.

[124] Viola, Paul and Jones, Michael J and Snow, Daniel. "Detecting Pedestrians Using Patterns of Motion and Appearance". In: *International Journal of Computer Vision (IJCV)* 63.2 (2005), pp. 153–161.

[125] Wang, Liming and Shi, Jianbo and Song, Gang and Shen, I-Fan. "Object Detection Combining Recognition and Segmentation". In: *Proceedings of the Asian Conference on Computer Vision (ACCV)*. Springer. 2007, pp. 189–199.

[126] Weng, John Juyang and Ahuja, Narendra and Huang, Thomas S. "Cresceptron: A Self-Organizing Neural Network Which Grows Adaptively". In: *Proceedings of the International Joint Conference on Neural Networks (IJCNN)*. Vol. 1. IEEE. 1992, pp. 576–581.

[127] Weng, John Juyang and Ahuja, Narendra and Huang, Thomas S. "Learning Recognition and Segmentation Using the Cresceptron". In: *International Journal of Computer Vision (IJCV)* 25.2 (1997), pp. 109–143.

[128] Wilcoxon, Frank. "Individual Comparisons by Ranking Methods". In: *Biometrics Bulletin* 1.6 (1945), pp. 80–83.

[129] Wright, Stephen J. "Coordinate Descent Algorithms". In: *Mathematical Programming* 151.1 (2015), pp. 3–34.

[130] Wu, Bichen and Iandola, Forrest N and Jin, Peter H and Keutzer, Kurt. "SqueezeDet: Unified, Small, Low Power Fully Convolutional Neural Networks for Real-Time Object Detection for Autonomous Driving". In: *Workshops of the IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017, pp. 446–454.

[131] Wu, Bo and Nevatia, Ram. "Cluster Boosted Tree Classifier for Multi-view, Multi-pose Object Detection". In: *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*. IEEE. 2007, pp. 1–8.

[132] Wu, Bo and Nevatia, Ram. "Cluster Boosted Tree Classifier for Multi-view, Multi-pose Object Detection". In: *Proceedings of the IEEE 11th International Conference on Computer Vision (ICCV)*. IEEE. 2007, pp. 1–8.

[133] Wu, Yang and Liu, Yuanliu and Yuan, Zejian and Zheng, Nanning. "IAIR-CarPed: A Psychophysically Annotated Dataset With Fine-grained and Layered Semantic Labels for Object Recognition". In: *Pattern Recognition Letters* 33.2 (2012), pp. 218–226.

[134] Xu, Dan and Ouyang, Wanli and Ricci, Elisa and Wang, Xiaogang and Sebe, Nicu. "Learning Cross-modal Deep Representations for Robust Pedestrian Detection". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017.

[135] Yao, Cong and Bai, Xiang and Liu, Wenyu and Latecki, Longin Jan. "Human Detection Using Learned Part Alphabet and Pose Dictionary". In: *Proceedings of the European Conference on Computer Vision (ECCV)*. Springer. 2014, pp. 251–266.

[136] Zeiler, Matthew D and Fergus, Rob. "Visualizing and Understanding Convolutional Networks". In: *Proceedings of the European Conference on Computer Vision (ECCV)*. Springer. 2014, pp. 818–833.

[137] Zhang, Cha and Viola, Paul A. "Multiple-instance Pruning for Learning Efficient Cascade Detectors". In: *Advances in Neural Information Processing Systems (NIPS)*. 2008, pp. 1681–1688.

[138] Zhang, Shanshan and Benenson, Rodrigo and Omran, Mohamed and Hosang, Jan and Schiele, Bernt. "How Far Are We From Solving Pedestrian Detection?" In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016, pp. 1259–1267.

[139] Zhang, Xiaowei and Cheng, Li and Li, Bo and Hu, Hai-Miao. "Too Far to See? Not Really! – Pedestrian Detection With Scale-Aware Localization Policy". In: *IEEE Transactions on Image Processing* 27.8 (2018), pp. 3703–3715.

[140] Zheng, Wei and Liang, Luhong. "Fast Car Detection Using Image Strip Features". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE. 2009, pp. 2703–2710.

[141] Ziou, Djemel and Tabbone, Salvatore and others. "Edge Detection Techniques – An Overview". In: *Pattern Recognition and Image Analysis* 8 (1998), pp. 537–559.

[142] Zitnick, C Lawrence and Dollár, Piotr. "Edge Boxes: Locating Object Proposals from Edges". In: *Proceedings of the European Conference on Computer Vision (ECCV)*. Springer. 2014, pp. 391–405.