Graz University of Technology

Institute for Computer Graphics and Vision

Dissertation

Object detection based on local evidence

Paul Wohlhart

Graz, Austria, January 2014

Thesis supervisors

Prof. Dr. Horst Bischof

Prof. Dr. Jürgen Gall

Having all the answers just means you've been asking boring questions.

---

the freedom of uncertainty.

e horne and j comeau, asofterworld.com

Abstract

For robust detection of objects in images the main challenges are handling various kinds of variations of the objects' appearance and dealing with partial occlusions. Implicit Shape Models (ISMs) and, in particular, Hough Forests provide an efficient way of describing the visual appearance of objects of specific classes from training images and a methodology to detect them in new images. The bottom-up nature of this specific part-based approach makes handling of transformations and occlusions easier than with holistic object detection approaches that describe the whole image patch that contains the object in one rigid layout.

This thesis investigates the ISM object detection paradigm, the specifics of Hough Forests and how to improve particular aspects of the method. The approach is first contrasted with other object detection paradigms to identify its relation and individual strengths and weaknesses. In a series of experiments and theoretical considerations the influence of the most important parameters and their interdependence are evaluated and discussed. A novel method to calculate the information gain of a test splitting the data in the nodes of Hough Forests is presented, which improves the quality of the regression with more efficient trees.

ISMs are very flexible in detecting objects that are formed out of constellations of sub-parts that have not been observed during training. This flexibility is, however, also identified as a potential weakness. One of the contributions of this work is to reintroduce a holistic view on the object into the Hough Forest framework, in order to reduce false positives from invalid object part configurations.

Additionally, a closer investigation of the voting process of the generalized Hough transform, used in the inference process at test time, reveals that many of the votes are

systematically conflicting with each other, which is usually simply ignored by marginalizing other them. This leads to additional evidence for objects aside from their true locations. Thus, the last part of this thesis introduces an algorithm to resolve these conflicting votes and thus greatly improves the detection of objects occluding each other or being occluded by others.

Finally, the benefit of each of the proposed improvements is shown in a series of experiments on real world data.

Kurzfassung

Die größten Herausforderungen für die robuste Detektion von Objekten in Bildern sind die Berücksichtigung verschiedenster Variationen des Erscheinungsbildes der Objekte, sowie die Behandlung von partiellen Verdeckungen der Objekte. Implicit Shape Models (ISMs) und im speziellen Hough Forests bieten effiziente Methoden zur Beschreibung der visuellen Erscheinung von Objekten spezifischer Klassen und zum Finden dieser Objekte in neuen Bildern. Die Art und Weise dieser Methoden, Objekte in Teile zu zerlegen und von lokaler Information im Bild auf mögliche Detektionen zu schließen macht die Behandlung von Transformationen und eventuellen Verdeckungen der Objekte einfacher, als dies bei holistischen Methoden der Fall ist, die das ganze Objekt in einem rigiden Layout beschreiben.

Die vorliegende Arbeit untersucht das ISM Objektdetektions-Paradigma, die Besonderheiten von Hough Forests und wie bestimmte Aspekte dieser Methoden verbessert werden können. Zunächst wird die Herangehensweise anderen aus der Literatur gegenübergestellt um Stärken und mögliche Schwächen zu identifizieren. In einer Reihe von Experimenten und theoretischen Überlegungen werden der Einfluss der wichtigsten Parameter der Methode und ihre Abhängigkeit von einander aufgezeigt, evaluiert und diskutiert. Im Zuge dessen wird eine neue Methode vorgestellt die Qualität eines Split-Tests in einem Knoten eines Hough Forests zu optimieren, die zu effizienteren Entscheidungsbäumen und einer besseren Performance des gesamten Detektors führt.

ISMs sind sehr flexibel und erlauben Objekte zu detektieren die in dieser Form, in dieser Konstellation von Einzelteilen, nicht in einem der Trainingsbilder abgebildet waren. Diese Flexibilität kann jedoch auch ein Nachteil sein, wenn zufällige Zusammenstellungen von einzelnen Teilen im Hintergrund zu falschen Detektionen führen. Einer

der Beiträge dieser Arbeit ist demnach, in Hough Forests teils wieder eine holistische Sichtweise einzuführen um solche Fehldetektionen zu reduzieren.

Zusätzlich zeigt eine genauere Analyse des Prozesses der "generalized Hough transform", der zur Testzeit zum Finden von Objekten eingesetzt wird, dass viele der von den Einzelteilen abgegeben "votes" für Objekte einander systematisch ausschließen, was herkömmlicherweise einfach ignoriert wird, indem trotzdem alle berücksichtigt werden. Dies führt zu zusätzlichen Hinweisen auch mögliche Objekte, abseits von den wirklich im Bild befindlichen Objekten. Daher wird im letzten Teil dieser Arbeit ein Algorithmus vorgestellt, mit dem die Konflikte aufgelöst werden können, was vor allem die Detektion von teilweise verdeckten und einander verdeckenden Objekten verbessert.

Die Vorteile der vorgeschlagenen Verbesserungen und Methoden werden in einer Reihe von Experimenten auf "real-world" Daten gezeigt.

Statutory Declaration

I declare that I have authored this thesis independently, that I have not used other than the declared sources / resources, and that I have explicitly marked all material which has been quoted either literally or by content from the used sources.

Place	Date	Signature

Eidesstattliche Erklärung

Ich erkläre an Eides statt, dass ich die vorliegende Arbeit selbstständig verfasst, andere als die angegebenen Quellen/Hilfsmittel nicht benutzt, und die den benutzten Quellen wörtlich und inhaltlich entnommene Stellen als solche kenntlich gemacht habe.

Ort	Datum	Unterschrift

Acknowledgments

Without the contribution of many people around me most of this work would not have been possible, or at least a lot tougher and certainly less fun. The support I received ranges from scientific input and technical help to moral backup, a great deal of humor, motivation to focus as well as necessary and welcome distraction.

My gratitude goes to Prof. Horst Bischof. First, for sparking my interest in computer vision with his interesting lectures, then, for giving me the opportunity to pursue my research with much liberty by letting me join his group at the Institute for Computer Graphics and Vision and supervising this thesis. I thank Prof. Jürgen Gall for inspiring much of my work with the ideas on object detection he presented throughout the last years and for his work as my second supervisor.

Great thanks goes to all the members of the Learning Recognition and Surveillance group, of which I had the pleasure of being a member. Among them Peter M. Roth, *the boss*, with his seemingly endless energy when it comes to taking care of the students under his custody, discussing problems, forging texts like a Japanese sword, proofreading until perfection. Special thanks go to Martin Köstinger and Samuel Schulter, my office- and project-mates and companions through all the years. I greatly enjoyed our interesting discussions of new ideas and great deal of jokes, keeping the spirits high. I hope we can continue to work and party together. I thank Michael Donoser for collaborating with me on several ideas and publications and for organizing our reading group, a constant supply of interesting ideas, resulting from vivid discussions about a wide range of topics in computer vision and beyond, improving our interdisciplinary thinking. Generally, I have to thank everybody at the ICG for chats and discussions, but especially for creating the spirit that makes working there really enjoyable. Thanks to

the coffee group for providing me with my daily dose of finest espresso. Also, thanks to our local Mensa for supplying my daily lunch and the whole team of Mensa-men, bravely accepting their fate and eating whatever was served, without batting an eyelid. While the food was mostly interesting, the conversation was always delicious.

Finally, I thank my dearest, who have accompanied me all along the way up to this point. I especially want to thank my girlfriend Iva for being at my side at all time, supporting me all the more when progress is slow and batteries are running low before deadlines or during the writing process. Thank you for the love we are sharing. I thank my friends, my whole family, my Mum and Dad and my brother for their interest in what I am doing and their unconditional support and love.

This thesis was created at the Institute for Computer Graphics and Vision at Graz University of Technology between 2009 and 2013. In that period this work was supported by the Austrian Science Foundation (FWF) project Advanced Learning for Tracking and Detection in Medical Workflow Analysis (I535-N23), by the Austrian Research Promotion Agency (FFG) projects Multimedia Documentation Lab (818800) and Human Factors Technologies and Services (2371236).

Contents

1 Introduction

The goal of computer vision is to make machines understand images. In a world where much of the information is conveyed visually vision is one of the most essential sensory modalities, and from an engineering point of view, crucial for building systems that can interact naturally with humans and solve complex tasks. One of the core tasks in this endeavour is the detection of objects in images.

Object detection has a wide range of applications. For instance, in robotics, an autonomous system needs to locate objects to solve specific tasks as well as identify its surroundings. In human-computer interaction, visual input allows the system to locate human users and understand their actions. In multi-media content management systems, images can be indexed and retrieved automatically by their content. In general, identifying and locating objects in images is a key ingredient to a deeper understanding of the contents of an image.

Class specific Object Detection

For a finer specification, the task of *class specific object detection* can be distinguished from image classification and detection of specific objects.

Given an input image, in image classification the task is to tell which kind of object(s) it displays. In early work, this usually meant that the image only contained one object which was shown prominently in the image, filling it almost completely, without any other objects, although, possibly featuring some random background. Thus, the input was defined to be an image that can unambiguously be given a single object class label. Today, it is also extended to images containing more than one object and the task is then to list all labels of objects appearing in the image.

In object detection, on the other hand, the tasks not only to tell which object is present, but also to tell where it is, *i.e.*, to localize it. Depending on the application, input images may contain a variety of objects on arbitrary backgrounds. The output can usually be either a bounding box rectangle around the correctly identified object(s), or in more detailed systems, a full, per-pixel segmentation mask covering and outlining the object. Also, the image might not even contain any objects of interest, in which case the output should be empty.

In *class specific* object detection, the task in narrowed down from identifying all objects in the image to identifying and localizing objects of one particular object class. This requires a definition of an object class or category. Generally, the object category is defined by a semantic concept such as "face", "car" or "person". The granularity and scope of the target category is mostly defined by the task for which the object detections are needed, specifying what is of interest to the system and subsequent modules, building on top of its output. Thus, target object categories can be defined on a quite broad level, such as an automatic driver-assistance system in a car, that needs to identify all "vehicles" in its direct environment. Such broad definitions often include a wide range of objects that are difficult to describe jointly, because of the large variance of the appearance. A category like "vehicle" is said to have large *intra-class variance*. Thus, in practical system they are often broken down into narrower and more homogeneous sub-categories (such as "car", "bus", "motorcycle", "bike", etc. for the broad term "vehicle") and treated independently, with separate detectors for each sub-category. Since the viewing angle from which the object is captured is another source of large variations in appearance, sub-categorization can also address this aspect, such as in "frontal face" and "profile face", defining sub-classes of the "face" class.

On the other side of the spectrum, the differentiation is against finding a specific object instance, such as the "Where's Waldo" task in the famous children's book. This task also has lots of applications, such as in autonomous robotics where, for instance, a robot has to identify specific landmarks in order to localize and orient itself in its environment or operate in a cooperative setting, asked to hand the human co-worker a specific item. The essential difference is that the target object in question itself remains the same. The detector still has to be invariant to changes in the appearance of the object, stemming from different imaging modalities, such as viewing angle, lighting, occlusion and deformation, but not to changes in the texture and general appearance. In contrast, different objects of a particular class might come in different colors, with different textures, different shapes, some may have a particular visually detectable feature, others might not. An

object class detector, as discussed in this thesis, thus, has to deal with a certain amount of intra-class variance, causing individual instances to look quite different.

Goals and Motivation

The challenge to be addressed in this thesis is, thus, to robustly detect objects of a particular class in images under all the modes of variation described above. Additionally, we are interested in robust detection of objects that are partially occluded by other objects or overlapping each other.

As described in Chapter 2, a wealth of object detection systems was developed over the past decades and presented in the literature. They all present different approaches to handle the multi-modality of the input data that comes with the variations in appearance of a single object or between objects of the same class. Among them, the line of work following the Implicit Shape Model (ISM) [93] presents a way to robustly model objects in a bottom-up fashion, starting from local evidence and thus allowing for easier handling of partial occlusions. Hough Forests [66] are a variant of ISMs that allows for more efficient recognition of local parts and, thus denser evaluation, resulting in more robust estimates.

However, while the bottom-up nature of Hough Forests and the capability of the classifier to handle multi-modal input at the local level implicitly allows for capturing all different kinds of variations, it makes it harder to enforce consistency of the whole detection. Also, completely independent treatment of the local parts prohibits to learn the importance of each part for the overall detection. The first goal of this thesis is, thus, to introduce modifications to the system such as to arrive at valid constellations and suppress false detections in background clutter.

Additionally, during the detection process Hough Forests generate information for each pixel about whether there is an object at this particular location and where the center of this object is. This information is currently used by summing over all evidence, ignoring the fact that it might be mutually inconsistent. The second major goal is, thus, to leverage this information to explicitly reason about with parts of the image belong to which object instance and detect occlusions.

Outline

The thesis is structured as follows. Chapter 2 gives an overview over object detection methods. It identifies key concepts of object detection and compares several systems from the literature. Aside from giving an overview over the related work, the goal is to clarify the relation and (dis)similarities of ISMs and Hough Forests to other methods in the field, in order to identify individual strengths and weaknesses of the approach.

Chapter 3 gives a detailed introduction to the Hough Forest framework, along with a set of extensions from the literature. Additionally, a novel method to optimize the split functions in the decision trees of the Hough Forests is presented that better handles the multi-modality of the regression targets and, thus, generates more efficient trees. When training Hough Forests one is, thus, confronted with several options to choose from and parameters to set for the training procedure. Therefore, the end of this chapter is a thorough evaluation of the impact of choosing specific sub-modules and parameter settings. Finally, the performance of the best performing setup is compared to the state-of-the-art on two challenging benchmarks.

The problem of enforcing consistency in the constellations of detected objects is addressed in Chapter 4. By recording the configuration of local features that were fused into an object detection, a description of the detection hypothesis is generated that can be used to discriminate right from wrong constellations. Additionally, by looking at the whole object instead of only individual local parts, the significance of the estimates from each local element can be determined. Using this information by weighting their contribution to the overall detection leads to improved accuracy.

Chapter 5 addresses the problem of finding consistent object detections, especially in situations where objects are occluded and multiple objects are overlapping each other. A random field formulation is presented to jointly solve the problems of finding high scoring detection hypotheses and assigning local features to them. Aside from improving detection of occluded objects, it presents a principled way to perform non-maxima suppression for ISM methods.

Finally, Chapter 6 concludes the thesis by discussing the achieved results and possible future work.

2 Object Detection Paradigms

This chapter discusses related work in object detection. Since it is a very central task in many computer vision applications, a wealth of methods and systems has been proposed over the last decades.

Object detection systems consist of several components that make up the whole object detection pipeline, from acquiring input data, preprocessing it, extracting features to inferring potential object locations, resolving conflicting object detection hypothesis and delivering the output. Additionally, an object detection framework must provide tools to prepare and preprocess training data for a specific object class and algorithms to adjust (learn) the object model's parameters in order to fit the model to the target class. Comparing object detection methods to each other is often difficult, because in many cases all individual components are different from one system to the other, such that the influence of each individual choice is not easy to evaluate. Thus, this chapter identifies the core components that every object detection system consists of and the respective choices that were made for the individual systems, as presented in the literature. Given the vast amount of literature in this field, the aim here is to give an overview over the dominating approaches and explain the main concepts of how object categories are modeled, how parameters for those models are learned and how images are processed in order detect objects in new input images.

Each of the following sections describes one of the major building blocks or aspects of an object detection system and gives examples from the literature.

2.1 Input Representation

Images are presented to the system in form of grey value or color images and/or other modalities such as depth maps. The values in those raw input pixel maps are often noisy and change drastically with changing input capturing conditions such as illumination, resulting in different levels of brightness and contrast, as well as shadows. Thus, in most object detection systems, the first step is to transform this raw input via low level operations into a representation that is more robust to varying imaging modalities. For these initial computational steps the multitude of existing object detection approaches draws from the huge spectrum of low level image transformations developed for all kinds of computer vision tasks. The basic steps are defining where in the image information should be extracted and encoding it into a representation that allows for robust and invariant identification. The following Sections discuss the options using examples from the literature.

2.1.1 Regions

The first choice to make is where to extract information in the image. The simplest solution is to compute information from and for each single pixel of the input image and, thus, generate a dense description of the image. For reasons of efficiency, however, many methods first try to identify regions or locations within the image that may contain interesting information and exclude the others from further processing.

A generic approach to identify interesting areas in images and differentiate them from random background, is *visual saliency detection* (*e.g.*, [25, 80]). The goal is to assign each pixel in the image a value, specifying how likely it is that the immediate neighborhood contains content relevant to the detection of objects of interest or generally information revealing the predominant nature and content of the image. From these saliency maps, covering the whole image, interesting locations or regions can be extracted by looking at maxima of the likelihood, thresholding it or delineating bounding box containing mostly salient areas. Aside from judging the performance of systems building on top of the deduced input regions, salience maps can also be evaluated by comparing them to recordings of eye tracking equipment, showing which parts of the image humans put their focus of attention on.

A slightly different approach, that was developed only recently, is to directly and efficiently judge if a certain region might contain any kind of object, not on the pixel level, but in form of candidates of region bounding boxes. The computed measure for

the regions is called the *objectness* [3, 4, 29, 121]. Similar approaches in this direction include category-independent region proposals [48], constrained parametric min-cuts [28] and selective search [151].

However, the predominant approach to reduce the input data and focus on the essential parts of the image are *interest point detectors*, aiming to identify specific salient or interesting points in scale space. The main quality measure for an interest point detector is *repeatability*, *i.e.*, if the same point on an object or in a scene will be re-detected in a differently captured image (different lighting, viewing angle, ...). In most methods the detection procedure is based on the statistics of gradient information in a small neighborhood around a point in scale space, facilitating detection of edges or corners in the image.

One of the first and also widely used interest point detectors is the Harris operator [68], further developed into the "Good features to track", by Shi and Tomasi [137]. The main idea is that corners (points with strong gradients in two directions) are easier to distinguish from their immediate neighborhood and therefore easier to re-identify in another image than homogeneous regions or edges and ridges, that look the same under translation along the direction of the edge.

The Harris operator detects points invariant to rotation, but not invariant to scale. The Laplacian of Gaussian (LoG) blob detector and its faster approximation Differences of Gaussians (DoG), used as interest point detectors in [99], additionally estimate the scale of the interest point. The current state-of-the-art in interest point detecting with a special focus on speed includes FAST [127, 128] and SURF [12]. FAST uses decision trees comparing the brightness of the center pixel with its surrounding in each node to quantify the saliency of each location. SURF approximates gradient computation by Haar-like features, efficiently computed on integral images. Comprehensive and thorough surveys were presented in [108, 150].

Super pixels [1, 136, 154] aggregate neighboring pixels into small and usually compact and homogeneous groups (same color, texture, ...). The result is a partitioning of the image into a smaller number of elements, thus reducing combinatorial complexity and making further analysis easier. Even larger image regions (sometimes also called super pixels) are provided by unsupervised image segmentation methods, such as [9, 57].

The dual problem of finding contiguous homogeneous image regions is *contour detection*, where the outlines of the regions are sought (*e.g.*, [9, 98]). From those boundaries the regions can be derived. Additionally, boundaries or also just boundary fragments can also serve as basic elements on which to build the detection inference pro-

cess [14, 47, 61, 62, 63, 87, 101, 114, 122, 125, 138]. The main benefit of using smaller boundary fragments instead of full regions is robustness to partial occlusions.

While generic unsupervised image segmentation methods do not explicitly address repeatability of the segmented regions, Maximally Stable Extremal Regions (MSER) [104] attempt to identify regions that are homogeneously bright or dark with strong contrast to its surrounding and therefore likely to be re-detectable in other images.

In recent works, however, a general trend towards dense representations is noticeable (*e.g.*, [13, 17, 39, 44, 90]), especially when detection quality is the primary concern and traded-off against speed. One reason is that the interest point localization is often not reliable enough. Additionally, extracting information only from visually salient locations might not capture all the discriminative information for a given object category. Also, the question if determining interest points really results in a speedup depends on how expensive it is to calculate them and which type of computations follow. For instance, the very fast pedestrian detectors in [44] and [13] calculate dense descriptions, albeit on quite low resolution versions of the image.

2.1.2 Robust and Invariant Description

Whether the first step of the processing was interest point detection, segmentation or every pixel in the image is considered, the task now is to describe the local neighborhood of each given point in scale space in a robust and compact way. Given its description the system must be able to efficiently and reliably be re-detected the point in another image of the same object instance or, more general, of another object of the same class.

To effectively reduce some of the noise on the raw input, a first and often important step in the data preparation of many object detection methods is a slight Gaussian smoothing of the input, with a small kernel width (*e.g.*, [41, 45, 46]). Another simple and fast way to extract information more robustly to noise is not to consider individual single pixels, but to sum values over larger areas, which for reasons of efficiency are usually rectangles. To reduce the computational effort of calculating a large number of those sums, as for instance in the computation of Haar-like feature responses (*e.g.*, [12, 155, 156]), the first step is to compute an *integral image*, making the computation of the sum over the values in a rectangular area an $O(1)$ operation.

In addition to histogram equalization and global or local contrast normalization (*e.g.*, [13]), gradient computation is the dominating method to gain insensitivity to varying illumination. The idea to take magnitude and orientation of gradients is mostly motivated by early work on analysis of the visual cortex of animals in the seminal work of Hubel

and Wiesel [74]. Mostly, raw gradient computation is realized by Sobel filters, or approximated by Haar-like filters (as, *e.g.*, in [12]). Another, biologically inspired variant are Gabor filter banks (*e.g.*, [109, 134]), effectively computing gradient magnitudes in different scales and different orientations. Other features attempting to robustly capture distributions of local variations of brightness include Local Binary Patterns (LBP) [112, 119] and the Census Transform [160].

In order to gain robustness against slight translations of the image content, information is usually aggregated (or *pooled*) from groups of neighboring pixels into larger units. For instance, super-pixels or larger segments can be used as areas over which descriptions are aggregated. Pooling is also a central part of the computation of the Histogram of Oriented Gradients (HOG) descriptor [41] and SIFT descriptors [99]. Gradient magnitudes are first binned into histograms by their orientation, then those per-pixel histograms are aggregated (summed) over small (*e.g.*, 8×8) rectangular areas, termed cells, to achieve invariance to small translations. Additional local contrast normalization is achieved by normalizing the histograms of each cell by a factor calculated over groups of neighboring cells, termed blocks.

Additionally, the state-of-the-art in keypoint description includes methods such as RootSIFT [7], SURF [12], BRIEF [26], BRISK [96], FERN [95] and BinBoost [147], each with different trade-offs between matching performance, computation speed and descriptor compactness. Additionally, an overview and comparison over combinations of interest point detectors and descriptors (although not including the more recent ones above), especially with a focus on object recognition as the target application, was presented by Zhang *et al.* [161].

A generic approach to compute dense representations are Integral Channel Features, presented in [46] and used in [13, 44, 45]. Also Hough Forests [66] use a similar representation. A set of different, individually calculated per-pixel representations are assembled into a stack of channels, each the size of the input image. Representations include channels of different color spaces (Lab, HSV, ...), first and second derivatives in x and y direction, individual bins of histograms (*e.g.*, of oriented gradients in a small neighborhood). Again, over each channel an integral image can be calculated in order to facilitate fast computation of sums over rectangular areas. This concept is easy to extend to more layers, using any kind of low level operation that generates a feature map containing information that might be relevant for the task. In order to reduce amount of data the representation can be compressed, for instance by principal component analysis (PCA) over the feature channels. In [56] the HOG-like feature channels are compressed

by observing from PCA that the channels resulting from different normalizations of the histograms are strongly correlated and, making use of this fact, designing an easy to compute representation with reduced redundancy.

Dispite their differences for all these input processing methods the process is bottom-up / feed forward, generic, and mostly designed by hand (although in some cases parametrized by unsupervised learning methods, relying on general statistics of natural images). The goal and hope is that the representation will be suitable for subsequent steps to extract the necessary information for recognition and detection in a robust and easy way. However, the methods described above do not perform a direct optimization of this first level representation for the targeted task. That is, if you do not consider the researcher-in-the-loop trying different representations and parametrizations and analyzing the impact on the test results, as an optimization procedure. For instance, which feature channels in a detector like [45] are really important to increase performance on a specific task and object category is mainly subject to educated guessing, or trial and error.

2.1.3 Learning the Encoding

A recent development is to try to learn descriptors in a supervised way from input data, directly targeted for a specific application. A range of approaches has been proposed (*e.g.*, [30, 67, 71, 105, 118, 143]) that build on top of hand designed descriptors and then learn a metric that better captures similarities as defined by the target application. Other methods, such as [24, 135, 146, 147], try to directly optimize the layout of the descriptor. Additionally, learning invariant descriptions is one of the main goals of applying (deep) neural networks to computer vision (*e.g.*, [88, 133, 158]).

Keypoint based methods often use a vector quantization step to break down the complexity of the matching of descriptors. A common approach in this direction is to use dictionaries of visual words [141]. The codebook entries are defined as the centers of a k-means clustering of the descriptors extracted from training data. New data is encoded by finding the closest codebook entry. Usually L_2 distance is used, but distance metrics can also be learned discriminatively and specifically for the task, by metric learning approaches such as [83], or discriminative embeddings [24, 71]. In [110] a hierarchically organized codebook was introduced, termed *vocabulary tree*, for faster matching. Fisher Vectors [40] store the derivative of the model with respect to the model parameters at the encoded data. The VLAD descriptor [8, 77] can be considered as a simplification of the Fisher kernel [117], allowing for very compact and powerful representations for

image retrieval. The approximation of the derivative of the model, is defined as the difference vector between the code book entry and the encoded vector.

In a series of object detection approaches a set of linear filters has to be convolved with the input, increasing run-time linearly with the number of filters. To reduce the complexity the linear filters can be approximated by finding a common bases for them. For instance, for the Deformable Part Model [56] this was addressed in [120]. The response of each individual filter is approximated by finding a common basis of steerable filters that well describe the range of filters. Then only this reduced set of basis filters has to be run on the new input and to receive approximate response coefficients for each part. A similar approach was taken in Sparselet Models [142], where the learning of the filter basis is based on sparse coding.

Sparse Coding is also a widely and successfully used technique to learn codebooks for interest point based models [73, 78, 100, 159]. Recently it was discovered in [32] that the training of a good sparse basis is less important than the sparse encoding scheme.

2.2 Object Models

Generally, the object model is a description of the object class, its appearance and shape, in mathematical and/or algorithmic terms. Applying the model to an input gives a score for the input to contain an object of the target class. Depending on the model this output score can be a probability, or more generally a scalar, where high values indicate higher likelihood for an object of the target category. The following sections discuss aspects and examples of object models.

2.2.1 Holistic vs. Part-Based

A major discriminating feature, by which object detection methods can be categorized, is whether the object is modeled as a whole or if there exists a notion of parts into which the object can be subdivided.

2.2.1.1 Rigid / Holistic

Object detection systems with holistic models impose a rigid structure on the object. Images of instances of the object class are warped into a common coordinate system. The extent of the object is defined by a (usually rectangular) reference frame (*e.g.*, a bounding box). Within this reference frame characteristic elements or features of the

object are expected to be at defined locations. Tests against the values of the input representation expected at those defined locations make up the object model.

At test time the reference frame is defined over the new image. Depending on the representation of the input, features are extracted at those defined locations and tested against those values stored in the model to generate a detection score for the object, specifying the likelihood of the image to contain an instance of the target class at the location of the reference frame.

Effectively, in this scheme the input patch, or its representation respectively, is seen as one feature vector that is classified. The classification score expresses if there is an instance of the target object class in the image patch. Any kind of classification procedure from the machine learning literature can be used, as explained in more detail in Section 2.3.

Pooling One way to overcome the restriction that elements have to be at exact locations within the reference frame is to aggregate features over larger areas in the image. This is also referred to as *feature pooling*. As mentioned in Section 2.1, in some systems, pooling is already performed in the low level calculation of the feature representation, such as in the computation of Histograms of Oriented Gradients, where the gradient histogram are aggregated within predefined cells.

In methods employing techniques such as Spatial Pyramid Matching [90], Pyramidal Histogram of Oriented Gradients (PHOG) or Pyramidal Histogram of Words (PHOW) [17], another pooling step is performed on the object level. In this case, a pyramid of pooling areas is constructed (again with a rigid layout), starting with the whole bounding box as one root area and then recursively subdividing it by splitting in half along x and y until a maximum pyramid level (usually 3 or 4) is reached. The final representation of the image is then the concatenation of the representations pooled over each individual area.

2.2.1.2 Part-Based

In contrast to holistic models that look at the whole object at once, part-based models see the object as a collection of sub-parts. The main idea is that those sub-parts may have a more uniform appearance and thus be easier to model. Parts are detected individually and then bound together via a geometrical model. In this way variances and multi-modality of the appearance of the object, stemming from deformations of the object, causing the different parts to appear at different locations relative to each other, are

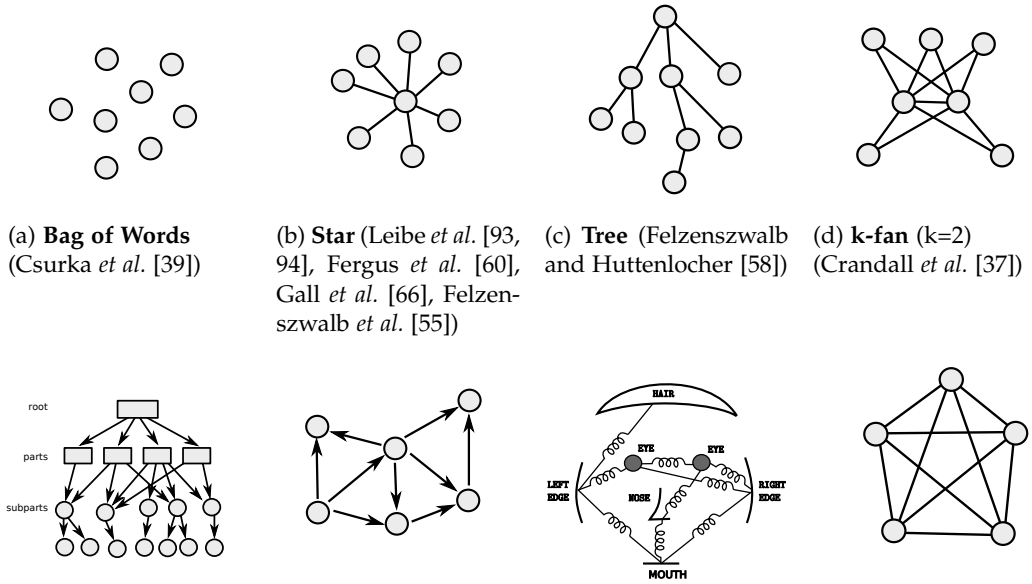
easier to handle. There are a couple of reasons why the positions of parts relative to each other might be different from image to image. One reason is a change in the camera pose and thus viewing angle from which the object was captured. Another reason might be the flexibility of an object instance itself, such as when the parts of the object are physically connected by flexible joints. A third reason is intra-class variance, *i.e.*, different instances of objects of the same class have their parts in different locations, such as, for instance, differently shaped cars.

Early work in part-based object detection mainly focused on identifying semantically meaningful parts the object consists of, such as the torso, head, arms, legs of a human body or the wheels, front, back, lights, doors or windshield of a car. Manually defined and semantically meaningful models have the clear advantage that the resulting detection and fit of the model allows for further high level, semantic inference based on the location and constellation of the parts.

However, apart from being difficult to define and laborious to identify and manually annotate, those semantically meaningful parts have the disadvantage that detecting them individually can be more challenging than detecting the whole object or combinations of some of the parts in specific constellations. Depending on the pose, parts may be occluding each other, or be barely visible due to foreshortening (such as arms pointing at the camera).

Thus, recently, many methods focus on either purely unsupervised or weakly supervised discovery of parts that are not necessarily semantically meaningful, but discriminative and easy to detect in an image. For instance, Deformable Part Models (DPM) [55] use an iterative process to find a fixed number of parts, by initializing part locations and sizes evenly distributed over the object, training detectors for each part and then refining the locations of the sub-parts in each image. The output is an appearance model for each part and a position of each part in each image.

A different concept of parts, called Poselets [20], was introduced in the context of person detection, segmentation and pose estimation. A poselet is defined as a specific local configuration of a set of parts, corresponding to a specific pose of a human body. Training images are annotated in detail with body joint locations. Candidates for poselets are created by placing reference frames at random locations on the body and finding clusters of similar joint locations and poses in the neighborhood of the reference center. For each of these candidate clusters a patch around the local configuration of joints is cropped from all corresponding person images and used to learn a detector (based on HOG and SVM). The set of candidates is pruned to a smaller list of poselets that can be



(a) **Bag of Words** (Csurka *et al.* [39])

(b) **Star** (Leibe *et al.* [93, 94], Fergus *et al.* [60], Gall *et al.* [66], Felzenszwalb *et al.* [55])

(c) **Tree** (Felzenszwalb and Huttenlocher [58])

(d) **k-fan** (k=2) (Crandall *et al.* [37])

(e) **Hierarchy** (Bouchard and Triggs [18])

(f) **Sparse flexible** (k=2) (Carneiro and Lowe [27])

(g) **Pictorial Structures** (Fischler and Elschlager [65])

(h) **Constellation** (Fergus *et al.* [59], Fei-Fei *et al.* [53], Cootes *et al.* [35])

Figure 2.1: Examples of geometric layouts of part-based models from the literature (Figure adapted from Carneiro and Lowe [27]).

detected reliably with the learned detectors. The main idea is that local subsets of joints in specific configurations, stemming from a limited range of similar poses are simpler to detect than individual joints in arbitrary poses. Also, if such local configurations can be detected, they are strong local pose estimators, that can be fused into a reliable estimate of the overall pose. Although the manual labeling effort is quite substantial, the process can be seen to be weakly supervised, since only the, for a human annotator, relatively easy task of labeling body joints is done manually. The difficult task of identifying discriminative sets of image patches with similar local joint configurations is performed automatically.

Geometric Layout One important property, by which part-based object models can be categorized, is the geometric layout of the parts. The model defines which parts are connected to each other and the type of connection specifies how the parts can move relative to each other.

Figure 2.1 shows samples of connectivity patterns of geometric layouts from the object detection literature, ordered by increasing complexity. Bag-of-Words models (Fig. 2.1 (a)) are a special case (listed here only for completeness), as they impose no

geometric arrangements of the parts at all. All parts that are found within a bounding box are assembled into a single description, no matter where they are. For some object classes this method has proven to be surprisingly effective and is therefor still widely used in object classification and detection. The simplicity of the model also allows for very efficient inference algorithms [89].

The simplest connectivity model, enforcing a geometric layout of the parts, is a *star model* (Fig. 2.1 (b)). Each part is connected to a central node, while being independent of the other parts, keeping the complexity low and allowing for fast inference in the detection processtirolese. *Tree models* (Fig. 2.1 (c)) define a hierarchy of parts, where each part has only one parent node and, thus, no connections are allowed between the different branches. The tree structure still allows for efficient inference in $O(N^2)$. Trees also have the advantage of being a quite natural representation for several somewhat flexible object classes, such as for instance the human body.

On the other side of the spectrum, in *constellation models* (Fig. 2.1 (h)), every part is connected to every other part (fully connected graph). This gives the densest description, allowing for the most detailed interactions. The position of each part depends on every other part. This means, however, that the complexity of learning and inference is exponential in the number of parts.

In between the two extremes, several intermediate setups have been presented. In a *k-fan model* (Fig. 2.1 (d)), a central group of k fully connected parts is surrounded by parts that have connections to the central group, but not among each other. *Hierarchical models* (Fig. 2.1 (e)) define layers of parts and sub-parts as a directed acyclic graph. In the *sparse flexible model* (Fig. 2.1 (e)) the graph is defined in a dynamical way, where each part's location depends on that of its k nearest neighbors, allowing for flexible and deformable objects and a complexity that can be adjusted by setting k. One of the first part based models was presented by Fischler and Elschlager [65] (Fig. 2.1 (g)). The model termed *pictorial structures* connects parts by spring-like connections. Not every part is connected to all the others, as in the constellation model, however the loops appearing in the graph create a similarly high complexity. The concept was thus reduced to tree structures for fast inference in [58].

After defining the connectivity in terms of a graph structure, the connections themselves have to be modeled. Usually, parts have an optimal relative configuration discribed by translation and rotation, or more general an affine transformation. Around this optimal relative position, deviations are allowed with a certain penalty for unlikely deformations. Often, connections are modelled each as a single Gaussian distribution,

effectively defining a zero-configuration or optimal relative positioning, given by the mean, and a penalty for deformations along orthonormal directions given by the covariance matrix [55]. Usually the variation is simply modeled over translation in x and y direction, sometimes even with a diagonal covariance matrix, such that inference over both directions can be decoupled during inference. This representation makes inference relatively easy. However, by defining it over relative angles and distances also prismatic and revolute joints can be modeled [57]. Also mixtures of Gaussians have been used [66, 93, 94], in combination with efficient mode finding methods for inference.

2.2.1.3 Implicit Shape Models

A special sub-category of part based models are Implicit Shape Models (ISMs) [66, 91, 93, 114]. Contrary to the part-based approaches above, ISMs define the object in a bottom-up fashion. Models with explicitly defined parts, as discussed above, assume a relatively small number of parts that may change in appearance and relative position to each other, but nevertheless appear on every object instance, unless it is occluded and marked as such. In ISMs the model is again a collection of parts. However, the main difference is that ISMs consist of a potentially huge collection of parts, of which only a subset appears on each individual image. The parts are characteristic features of the target object class extracted from individual training images. While not every part is found on every image, a valid detection is ideally densely covered with correctly identified individual parts. Figure 2.2 illustrates the processes of training and testing with an ISM.

The geometric model underlying an ISM is a star-model. For each characteristic feature extracted from the training image the relative offset to a common reference frame of all training images (usually the center of the bounding box) is recorded. For each part, small deviations from these original positions are tolerated, resulting in a mixture of Gaussians model for the location of each part.

The parts are treated independently, and there are no interactions between them encoded in the geometric model, with every part only connected to the reference center. However, the dense coverage of the object results in overlaps of the parts, which reintroduces interactions. Thus, constellations of parts cannot be completely arbitrary, because neighboring parts share a large portion of the input image area from which they were inferred.

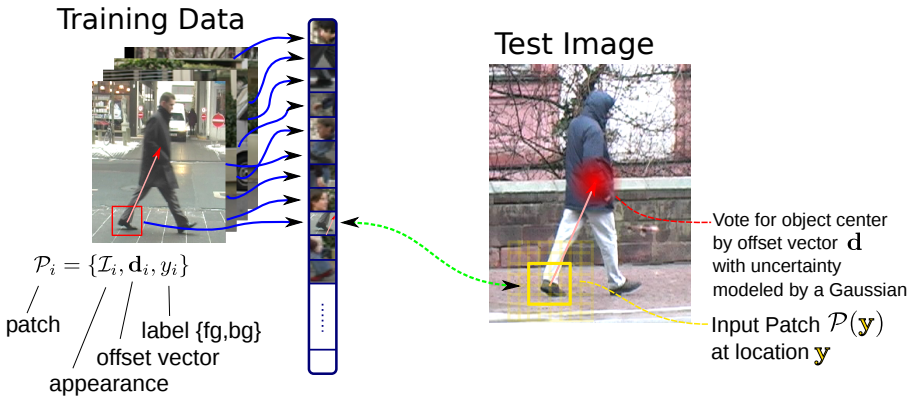


Figure 2.2: Implicit Shape Model workflow. Left: During training patches are extracted from the training images and stored along with their offset from the object center. Additionally patches are extracted from negative images. Right: During testing every patch (like exemplary patch P(y)) extracted from the new input image are matched to similar patches from the database (usually via some kind of dictionary). The offset vectors of the similar patches are used to vote for the object center, weighted by the ratio of similar positive and negative patches.

2.2.2 Components / Aspects

One of the major challenges to meet when designing an object model is the potential high intra-class variance. One way to deal with it are classifiers that are inherently able to deal with multi-modal inputs (discussed below, in Section 2.3). Another possibility is to define the model to be a combination of several sub-models, sometimes called *aspects* or *components*. Essentially, the idea is to partition the data and create a separate model for each subset. Each of the sub-models can be focused on a range of specific cases and, thus, does not have to deal with all kinds of variations appearing in the whole dataset. The training samples assigned to each component can be seen as a sub-class, usually chosen to be more homogeneous in appearance.

In the Deformable Part Model [55], where the term *component* was introduced, a fixed set of components is initialized, each specializing on a group of object instances with similar aspect ratio. Thus, for instance, there are separate models for side views and front or back views of cars. During the learning of the full model, the assignment of individual training samples to a component is updated by alternating training the models for each component and evaluating the intermediate component models to identify the one that fits best for each example.

Modeling all samples with a single component would be difficult, because the input representation (HOG-based) implies a fixed aspect ratio of the bounding box. Resizing

all training samples to this aspect ratio would result in significant distortion, resizing while keeping the original aspect ratio would result in highly multi-modal data and positive bounding boxes capturing a lot of background, for the thinner objects. It would, thus, also require a classifier that deals better with multi-modal input than the linear one, which used for efficient inference.

In the Latent Hough Transform [123] the assignment of training samples to components is again treated as a latent variable in an optimization procedure. The contribution of each training sample to the score of a detection is linear and can be computed from the back-projection [124] of the detection. An Interacting Simulated Annealing (ISA) procedure is used to update an assignment matrix, defining the mapping of samples to components. The formulation was also relaxed to soft assignments, where each sample can be associated with more than one component, which has been shown to increase the overall detector performance.

Similarly, also the separate models for frontal and profile face detectors shipped with the OpenCV implementation of the Viola&Jones object detector [155] can be seen as different components of a joint multi-view face detector model. The concept is taken to an extreme by the Ensemble of Exemplar SVMs [103], where a separate model is learned for each individual training sample.

If a detector consists of multiple components, in the end the question is how to fuse their output. Usually, each component is treated separately and the outputs are fused by taking the most confident component for each location. Another possibility is to have another classifier to decide which component should be applied to a specific input, as proposed, for instance, in [42, 72].

2.3 Learning

The goal of the learning procedure is to create a classifier that, applied to a specific input representation, produces a score, indicating whether it contains an object of the target class or not. In case of a holistic model, the input is a single, rigid descriptor vector and learning the classifier from positive and negative samples (in discriminative settings) is a standard machine learning task. In case of part-based based models, a classifier is learned for each part, either separately or in a joint model, where identifying individual parts and discriminating them against the background can be seen as a multi-class classification problem. Additionally, also the parameters of the geometric model need to be estimated from the training data.

Generally, for learning the parameters of the model from training data, many algorithms are available from the machine learning community. However, the combination of input representation and model often suggests a certain learning algorithm and thus, the learning algorithm is tightly integrated into the whole object detection framework.

Classifiers

Classification methods can be subdivided into generative and discriminative approaches. In generative models, only the distribution of positive data is modeled. A new test sample is then compared to the model and the classification is based on how similar it is to the positive training data. In discriminative settings, the distribution of positive data for one class is contrasted with that of the negative data or of other classes, respectively. The decision functions are then constructed such as to optimally discriminate between them. Discriminative modeling allows for identifying relevant (combinations of) features of the data and usually leads to better results. However, it depends strongly on proper acquisition and selection of the training data, because lack of adequate negative data may distort the decision boundaries inappropriately (which will also be demonstrated in the experiments in Section 3.6.1.1).

The simplest classification model is a linear classifier, such as the weight vector discriminatively learned in a Support Vector Machine [129, 152] and extensions such as the structural SVM [16, 79, 148]. The score is simply the dot product of the classification vector and the vector formed by the input representation. Such a model is, *e.g.*, used in the seminal work by Dalal and Triggs [41], proposing HOG as the underlying representation. These simple models, however, due to their linear nature have problems capturing multi-model input descriptors. Thus, as mention above in Section 2.2.2, the model has to be split into components or aspects, in order to deal with intra-class variance of the appearance of the target objects.

Another possibility are (mixtures of) Gaussian models over the positive training data. These models store a set of vectors that form a basis of a subspace of the input space, containing most of the variance appearing across the training data. The input representation is projected into the space by multiplying it with each of the basis vectors and the vector of resulting coefficients gives information about how well the new input is represented in the space of the training samples. Moving from generative to discriminative classification, these coefficients can be compared with another (set of) Gaussian(s) defined over the negative training data (background class). These kinds of models have mainly been used in recognition tasks, such as face recognition (“Eigenfaces” [82, 149]),

but also for detection, such as in the identification of individual parts in the original Pictorial Structures framework [65].

In the seminal work of Viola and Jones [155, 156] a cascade of boosted decision trees are trained using the AdaBoost algorithm. Shallow decision trees are used as weak learners, evaluating the response of a Haar-like feature in each node and thereby trying to discriminate positive from negative samples. The output of each tree is a binary decision. Ensembles of such trees are trained iteratively, one tree after the other. After adding a tree to the ensemble, the training samples are reweighted by the error of the current ensemble, such that the next tree focuses on those examples that have not been classified correctly so far. The output of the ensemble is the sum over the outputs of all trees, weighted by their individual error rates. In order to speed up the classification process, trees are grouped into cascade stages. Each stage is trained to achieve a certain minimal recall and precision. During evaluation, as soon as one cascade stage classifies the new test sample as negative, the computation can be stopped. Extensions include *soft-cascades* [19], where the evaluation can be stopped after each individual tree, and even faster *crosstalk cascades* [44], that avoid evaluation of the detector in non-promising regions of the image at all.

Generally, boosting algorithms with feature selection [145] at their core, are especially suitable in situations, where the potential input space is huge. This is the case in the example above, since it operates on the set of all possible Haar-like features that can be extracted from a given training image patch.

Random Forests have been used in holistic settings [131], as well as for parts in ISM-like models, such as Hough Forest [66]. The essential advantages of Random Forests in these setting are the implicit multi-class capabilities, and the ability to deal with multi-modal inputs. In comparison with the codebooks of visual words learned for the original ISMs, the tree structure of Hough Forests allows for fast matching and the leaves to which new inputs are associated are directly optimized for the subsequent regression problem of the object center.

Convolutional neural networks (*e.g.*, [88]) are a special case, because the input representation and encoding, as well as the final classifier are learned jointly.

Geometry

For part-based methods, if not defined by hand, the training procedure also needs to estimate the parameters of the geometrical model. If the parts are manually defined and annotated, relative distances between connected parts and variations thereof can be

recorded from the training data. Depending on the type of connection, the individual samples are stored or the distribution of relative part locations is approximated with a model (*e.g.*, as stated above, a single Gaussian).

In the case of ISMs the model captures the relative location of the object center with respect to each part extracted from the training images. Small deviations from these original positions of the parts are allowed, resulting in a mixture of Gaussians model for the location of each part, or the location of the object center as seen from the part, vice versa. This distribution can simply be stored in form of a list of all relative offsets, as is the case in Hough Forests, which store all offset vectors of all training patches arriving at a particular leaf node of each tree. In PRISM [91] the distribution is stored more compactly as an approximation by a mixture of Gaussians with a reduced number of components.

In the DPM, in each iteration of the process described above, estimates for the latent locations of the automatically found sub-parts are updated. Given those newly fixed locations a Gaussian model is learned for the location of each part, relative to its parent node. Additionally, in the Pictorial Structure framework [58, 162] not only the connection parameters, but also the tree structure itself is learned from training data. Relative (in theses cases not latent) part locations are analyzed and a tree is constructed, using the Chow-Liu algorithm [31], connecting those parts with the most stable relative position (least variance).

2.4 Inference

At test time an inference process is executed, applying the model to the input image and returning object detections as the output. How inference is performed obviously strongly depends on the model. However, there are some processes common to most methods that are discussed in following.

2.4.1 Sliding Window

With holistic object models, in order to localize an object in a sub-region of an input image, depicting more than just one central object that fills out the whole image, the image is processed in a *sliding window* manner: Rectangular subareas (windows) are extracted at every potential location and scale by transforming the reference frame to the current sub-window to test. Then, the resulting crop again contains the entire object with characteristic features at predefined locations. In this scheme, the detection is

effectively reduced to a classification problem of the sub-windows. Thus, any image categorization method, applied on sub-windows, can also be used to localize objects.

2.4.2 Fusing Parts

In part-based methods, individual parts are either extracted from interest points, or the image is processed in a sliding window manner, as described above. The results in both cases are sparse or dense maps of candidate locations with scores for the individual parts. The next step is to fuse the individual part detections into geometrically consistent object detections, according to the model.

For the Deformable Part Model [55] this process is realized efficiently, building on the generalized distance transform [54]. The model consists of a detector for a root part, capturing the whole object roughly, and more detailed sub-parts. Each of the parts uses HOG as representation and a linear classifier, learned with an SVM, to produce a matching score. The deformation cost is modeled as a Gaussian with diagonal covariance matrix, and is thus separable in x and y. The main observation is that the optimal placement of the root node at a specific location depends on its own matching score at this point and the placement of its child nodes, their matching quality at each location and the cost of placing the child node at those locations relative to the root. The inference process starts by computing score maps for each part in a sliding window manner. The maps of the sub-parts are processed by a generalized distance transform, using the deformation cost parameters of each sub-part. This aggregates the cost of placing sub-parts relative to the root node in a common reference frame. The transformed maps are summed up and the cost map of the root node is added. From the resulting score map, maxima can be picked and sub-part placements inferred at no additional cost. The same procedure is applicable for deeper tree structured models, by beginning at the leaf nodes and iteratively working up the hierarchy, up to the root node, as shown in [57].

In ISM-like models a voting process is used to infer object detections from local parts. Local features are extracted from the image (either from interest points or densely). For each local feature the best matching part stored in the model is identified (a visual words from the codebook in ISM, leaf nodes of the decision trees in Hough Forests). Each identified part casts votes according to its recorded spatial distribution model. The score for an object at a specific position and scale is calculated by the sum of the votes for this location. In Hough Forests, the summing is executed by accumulating the votes in a Hough space (Generalized Hough Transform), with one bin allocated per object detection hypothesis, followed by a Gaussian smoothing to account for small

translations of each part. In the original ISM, due to the reduction on interest points, there are fewer votes and thus the modes of the summed posterior distribution are extracted by a mean-shift procedure. The whole process is explained in more detail in Section 3.1.4.

2.4.3 Non-Maxima Suppression

A last and important, although sometime overlooked, step in the object detection processing pipeline is non-maxima suppression to create the final output in form of object detections. Sliding window based methods, as well as voting based methods generally do not directly deliver the final set of distinct detections. Rather, maps with scores for each object detection hypothesis are generated. Finding detections that the system can deliver as output then amounts to identifying the most confident hypotheses, while pruning away the others. A first step is thus to set a threshold on the detection score, that can be estimated on a validation set.

Usually, due to the desired invariance of the detector to small translations and variations in scale, around one highly scoring hypothesis there is a range of similar hypotheses that score similarly high, not being caused by a real separate object but by the same, already detected instances. In order to report correct objects only once, those neighboring hypothesis must be discarded. This is usually referred to as *non-maxima suppression*.

Methods range from calculating bounding boxes for each hypothesis and discarding those with significant overlap to a higher scoring one, to mode seeking with mean-shift or searching maxima in the Hough space and erasing neighboring evidence, for voting based methods. More details on non-maxima suppression methods are given in Section 3.5, and a new variant specialized on the recovery of close-by and overlapping object instances is presented in Chapter 5.

2.5 ISMs vs. Holistic Models

The top-down, sliding window inference with holistic models and the bottom-up, vote fusion of ISMs and Hough Forests seem to define very different approaches to object detection. However, Lehmann *et al.* [91] point out that for linear additive classifiers they are analogous. Whether the local point casts a vote for an object center and those votes are accumulated at specific locations, or a reference frame is set to that location and local features around it are counted, summing up the score, is merely a different way

of looking at it. The former represents a feature-centric view, the latter is object-centric. Thus, the question arises what differentiates the ISM from linear, holistic models. The answer to this is in the modeling and learning procedure.

The feature centric and voting based view suggested a different kind of modeling of the object's shape and appearance. The recording of offsets between feature and object center, and modeling of the distribution without discretization is quite different from the schemes used for standard holistic models. The representation of the content of a sliding window – termed *object footprint* and chosen in [91] to show the analogy to voting based methods – is an intermediate representation in the invariant space, coupling feature and object centric views. It is not a simple feature vector, as in classical linear holistic models, such as the BOW or HOG, that can be classified by computing the dot product with a weight vector, learned by an SVM. Instead, it is a continuous $4D$ function over x and y, scale and visual word index. Each occurrence of a particular codebook entry is recorded as a Dirac function at their relative position in the reference frame. The model W, that is multiplied with this representation to compute the detection score is, thus, also a continuous $4D$ function.

Thus, it is true that, by inverting the directions of the offsets (feature $\rightarrow$ center, center $\rightarrow$ feature), one representation can be converted into the other and the models and the formation of the score are analytically equivalent (see also Equations (4.1)–(4.5) in Section 4.2.1 for the computation of the score of a Hough Forest in the object centric view). However, for an object centric approach, this type of model is not very practical. This is also one of the reasons Lehmann *et al.* in the end stick to the feature centric view for inference.

Another difference concerns the learning. In both the original ISM and Hough Forests, the learning focuses on individual parts. The weights for each voting element and thus the contribution to the overall score are determined for each visual word on its own. Thus, the object is not considered as a whole, as in the holistic models, where weights for all features at all positions in the image and over all training images are optimized jointly.

On the one hand, this is an advantage, as it enables the detection of objects featuring combinations of parts that have not been observed in training. For instance, a lower body from one person can be combined with the upper body of another to fit the combination in a new image. Similarly, the training data does not need to contain all combinations of front leg and hind leg configurations of an animal. At the same time this can be a drawback, since it also allows classifying various illegitimate constellations as positive

object detections, such as persons with more than two arms in different poses. This makes the method susceptible to strong responses on arbitrary collections of patches in the background that look like object parts. One work addressing this problem is the Latent Hough Transform [123], described in Section 2.2.2. It splits the detector into components and restricts those voting elements to vote only for a common object that together form a consistent object.

Another aspect that gets lost when not looking at the whole object, but parts individually is the relative importance of each voting element for the overall detection. This particular issue was approached in the Max-Margin Hough Transform [102].

In Chapter 4 both of those problems will be addressed jointly, by deriving a descriptor of a detection hypothesis that records the contributions of each voting element and judging the validity of the detection by looking at this overall picture.

2.6 Conclusion

This chapter summarized and discussed the main approaches in object detection. As individual object detection systems are defined by specific processing pipelines, it is difficult to compare different approaches. Thus, we first identified the common building blocks and, then, discussed different options by means of examples from the literature.

From a general point of view, it can observed that pre-processing steps, such as the computation of a robust input representation, as well as post-processing steps, such as non-maxima suppression, are usually exchangeable between systems. In contrast, the object model and the learning framework used to adapt the model's parameters to a specific target class are strongly interlinked. The choice of the classifier itself is mainly governed by the specifics and requirements of the learning task. These include the dimensionality of the features, the amount of available training data, the distribution of the data and test time requirements.

The major challenges for object detection systems targeted in this thesis are (a) the robust handling of all modes of variation of single objects and between objects of a common class as well as (b) robust handling of occlusions, in particular, separate detection of mutually overlapping object instances.

One way to handle the variations is an input encoding that is insensitive to insignificant variations, *e.g.*, stemming from noise or minimal transformations. Larger variations (*e.g.*, moving parts of flexible objects) need to be addressed explicitly in the object model, because transforming all possible different appearances of the objects into one invari-

ant (or insensitive) representation in the first step usually erases too much information needed to discriminate the objects from the background. Some of the multi-modality of the input can be handled in the model. Components (or aspects) are used to effectively partition the input into more homogeneous subgroups. Spatial transformations of flexible objects can be handled by subdividing the object into parts and connecting them with a geometrical model. Finally, depending on the choice of the learning procedure, it can be left to the classifier to deal with all (or the remaining) variations.

Part-based methods explicitly capture the variations caused by deformations of the object. Additionally, ISMs allow for combining parts from different training samples to form new configurations and thus define a detector that can implicitly handle highly multi-modal input data. As discussed in Section 2.5, an analogy can be drawn between voting based methods and sliding window processing with holistic models. This, however only works on the level of score formation after the individual scores for local elements was computed. The type of classifiers learned on the local features for ISMs, Hough Forests and related methods leads to an object model that is very different from classical holistic object models. Furthermore, Hough Forests make the matching of local features more efficient, thus, allowing for a denser evaluation of the input images. Additionally, the learning procedure solves the classification and regression task jointly, thus, arriving at a better optimized "dictionary" of local parts.

The rest of this thesis will, thus, be focused on part-based approaches and ISMs and Hough Forests in particular. In fact, due to the bottom-up nature of the process and the rather small used local elements the latter ones are particularly suited for robust detection and occlusion reasoning. Thus, in the following Hough Forests will be presented, analyzed, improved and solutions to particular problems with invalid object configurations and occlusions will be given.

3 Algorithmic Framework

In this chapter the basic object detection framework used throughout this thesis is introduced. First, the related work of Hough Forests is presented in detail. Following this introduction, several extensions and modifications are presented that improve training and testing and thus the overall performance. This results in a range of different options for different parts of the algorithm, each coming with a set of hyper-parameters to tune. The last part of this chapter is, therefore, an extensive evaluation of the influence of the choices of options, hyper-parameters and interdependences between them.

3.1 An Introduction to Hough Forests

The basic framework for our method is a Hough Forest [66]. Hough Forests are based on the Implicit Shape Model [93] and thus model an object as a collection of a large number of local features (image patches and derived descriptions) $\mathcal{P}_i = \{\mathcal{I}_i, \mathbf{d}_i, y_i\}$. Here $\mathcal{I}_i$ is the appearance description of the patch (several feature channels calculated from the raw input pixel patch). The label y_i of the patch specifies whether a patch was extracted from a positive or negative training image. Each of the positive local patches additionally stores an offset vector $\mathbf{d}_i$ pointing from the location of the patch to the center of the object.

In the detection phase patches are extracted from the test image and compared to the patches stored in the database. The ratio of positive and negative samples in the set of most similar patches gives an estimate of how likely the location in the test images from where the patch was extracted is located on an object of the target class. Those patches which were identified as being on the object are then used to get an estimate for the center of the object. For each of them the offset vectors $\mathbf{d}_i$ of the most similar patches

from the database are taken and weighted votes are cast relative to the location of the patch. The aggregation of these votes leads to areas of high probability for an object of the given class and thus object detection hypotheses. Thresholding the probabilities and suppressing multiple detections of the same object by local non-maxima suppression gives the final object detections.

Comparing each patch from the test image to every single patch from the database of training images in order to get the set of similar ones would be inefficient, if not infeasible. Therefore, in the ISM the patches of the database are clustered into a codebook. Test patches only have to be compared to the set of codebook entries (visual words) to retrieve a set of similar patches. In contrast to this flat, generative codebook, Hough Forests build on Random Forests as the structure used to identify appropriate similar patches from the database in a discriminative way. It can be seen as an ensemble of hierarchical codebook structures, facilitating faster lookup and soft-assignment.

Random forests, introduced by Breiman [23] and inspired by ideas of Ho [69, 70] and Amit and Geman [5], are ensembles of decision trees $\mathcal{F} = \{\mathcal{T}_t\}_{t=1}^{T}$, where $\mathcal{T}_t$ is a randomized decision tree, and T is the number of trees in the ensemble. Each tree is composed of nodes. Each node is either the parent node of exactly two child nodes or a terminal leaf node. Each non-leaf node has a simple test with a binary decision output associated to it. The outcome of the test on a data sample defines if it is passed on to the left or right child node. To classify a sample the process is started at the root node of each tree. The test associated with the root node is evaluated and depending on the result the sample is passed to the appropriate child node. This process is iterated until the sample arrives at a leaf node. A leaf node contains information that was aggregated over the samples that reached this node during training. For the simplest case of binary classification, this could be the percentage of samples of each class. The decision of the overall forest is generated by aggregating the output of all individual trees. Different methods for the aggregation have been proposed, also depending on the type of information stored in the leafs. For instance, in classification the output can be the mean of the probabilities for each class, or individual trees can each make a hard decision and the class is taken that most trees agree on.

3.1.1 Feature Representation

A central part of each object detection method is the representation of the raw input image data that is then fed to the algorithm for learning and testing, as discussed in Section 2.1.

In the case of Hough Forests, as presented in [66], similar to the ChnFtrs [46], each patch is represented by a series of channels calculated from the raw RGB input data. L,a,b-color space channels, first and second derivatives of the L channel in x and y. Additionally there are *HOG-like* channels. Here, histograms of oriented gradients are not computed in a rigid structure of blocks and cells, as in the original HOG [41], but for each pixel in its local neighborhood. In practice, the gradients in a 5×5 neighborhood are binned into a histogram of 9 orientations with linear interpolation. The resulting 16 channels are each processed with a local 5×5 min- and max-filter, leading to a 32 channel representation in total.

Usually the representation is not calculated per patch but for the whole image, from which then the patches are extracted. If only very few patches are extracted from an image (*e.g.*, some random patches from a very large negative image) it is more efficient to really crop out the patches and free the memory for the rest of the image. If, however, the patches are sampled densely (such as in the detection phase where a patch is extracted around each pixel), its more efficient to store the representation for the whole image and to represent a patch only by its location on the image.

Several other feature channels have been proposed, such as different color spaces, other kinds of gradient histograms [46, 56] and a range of normalization schemes for each channel, such as in [13, 46, 56]. However, all results presented in this thesis are based on the feature channels of the original Hough Forests.

3.1.2 Split Tests

As mentioned above, each inner (non-leaf) node of each tree in the forest performs a binary test to decide whether to pass the sample on to the left or right child node.

The simplest test is to choose a single feature out of the feature vector and compare it against a threshold. This is commonly used in standard machine learning tasks. The advantage is that this kind of test is very generic and makes no assumptions about the relations between features.

In the case of image patches single feature tests are not very expressive and robust. For example, they are not invariant to brightness changes. Thus, in the original Hough Forests pixel-pair tests are used. Each test selects two locations in the patch and compares the difference between the feature values at those locations to a threshold.

A generalization of pixel-pair tests, that inspect two features, are oblique split test [23, 106], which calculate the dot product between the full feature vector and a split test vector of the same dimensionality, effectively defining arbitrarily oriented splitting hy-

perplanes in the original feature space. Candidates are usually randomly sampled, or optimized split directions [106].

Recently, Schulter *et al.* [132] proposed Ordinal Random Forests. There, the test is defined by a random set of pixels in the patch. The outcome of the test depends on which of the pixels has the highest feature value.

Another alternative, that was also recently employed in the context of Hough Forests are Haar-like features, as used in [115, 116, 156], which calculate the difference of sums over adjacent rectangular areas. This summing over larger areas makes the feature response more robust against noise and small translations. The sums can be computed efficiently using integral images [156]. A generalization of Haar-like features are sums over (sets of) arbitrary rectangular areas. Such tests have been used in boosted decision trees for pedestrian detection [44, 45, 46].

In this thesis pixel-pair tests are used throughout all experiments. As a small difference to the original formulation also tests at the same pixel location but on different channels are allowed. This especially makes sense for the HOG-like channels, where differences of strength of edges in different directions can be checked.

3.1.3 Training

The goal of Hough Forest training is to create a classifier that reliably estimates whether an image patch was extracted from an object of the target category or from the background. At the same time, for foreground patches also their location on the object should be estimated (or vice versa, where the center of the object is located relatively to the patch). Thus, when building a Hough Forest, for each node a split test is sought that either optimizes the purity of the class distribution (foreground/background) or the offsets, or a combination of both, in the resulting left and right child nodes. The training procedure consists of generating for each node a random set of split tests — in our case, the positions of the two pixels for the pixel-pair test (x,y and channel) and a list of candidate thresholds applied on the result of computing the difference. The resulting subsets for the child nodes are then evaluated with the selected optimization criterion and the test is chosen that gives the best information gain. This procedure is recursively applied to each resulting new child node until a stopping criterion is met.

In the original Hough Forests, for each node one of the criteria is randomly chosen, whereas in [113] the two criteria are combined. Either way, due to the optimization of both criteria, the leaf nodes store relevant statistics for both, classification into foreground and background and regression of the object's center.

Accuracy of the classification

To optimize the accuracy of the classification, the information gain

$$\Delta H = H(P_n) - \frac{|P_l|}{|P_n|} H(P_l) - \frac{|P_r|}{|P_n|} H(P_r) \tag{3.1}$$

is used to find the most promising split for the local set of patches P_n of the current node n. P_l and P_r denote the sets of patches which will be sent to the left and the right child node by the currently evaluated split test, respectively (*i.e.*, $P_n = P_l \cup P_r$). $H(\cdot)$ denotes the entropy of a set $-\sum_{k=1}^{K} p_k \cdot \log(p_k)$, where p_k is the probability of the current node to belong to class k, estimated from the ratio of positive and negative samples. The first term in (3.1) $H(P_n)$ measures the entropy in the parent node. Since the set of patches in the parent node is fixed, this entropy is constant during the optimization procedure testing different split tests that only results in different partitions of the samples into P_l and P_r. It can thus be dropped in the optimization.

Accuracy of the regression

To optimize the offset impurity and thus improve the performance of the regression, only the positive local patches are considered and splits are searched that minimize the variance of the offset vectors in the child nodes:

$$\min \sum_{\mathbf{d}_i \in P_l} ||\mathbf{d}_i - \bar{\mathbf{d}}^l||^2 + \sum_{\mathbf{d}_i \in P_r} ||\mathbf{d}_i - \bar{\mathbf{d}}^r||^2 \ , \tag{3.2}$$

where $||\cdot||$ is the Euclidean norm and $\bar{\mathbf{d}}^l$ and $\bar{\mathbf{d}}^r$ are the means of all offset vectors $\mathbf{d}_i$ falling into the left and right child nodes

$$\bar{\mathbf{d}}^l = \frac{1}{|P_l|} \sum_{\mathbf{d}_i \in P_l} \mathbf{d}_i \ , \ \bar{\mathbf{d}}^r = \frac{1}{|P_r|} \sum_{\mathbf{d}_i \in P_r} \mathbf{d}_i \ . \tag{3.3}$$

This node split evaluation criterion will further on be denoted as *reduction-in-variance*.

Another, more general and information theoretic measure to asses the improvement in quality of the regression, which was also used in some recent publications (*e.g.*, [34, 38, 111]), is the *differential entropy* [36] of a set S:

$$H(S) = -\frac{1}{|S|} \sum_{x \in S} \int_y p(y|x) \log p(y|x) \mathrm{d}y \ . \tag{3.4}$$

For the task at hand, the conditional probability $p(y|x)$ is modeled as a multivariate Gaussian $\mathcal{N}(\mu(S), C(S))$ where $\mu(S)$ is the mean over the outputs for the data in the node and $C(S)$ is its covariance matrix. With this definition (3.4) can be calculated as

$$H(S) = \frac{d}{2}(1 - \log(2\pi)) + \frac{1}{2}\log(|C(S)|) \ , \tag{3.5}$$

where d is the dimensionality of the output (*i.e.*, 2 for our purposes). The first part is constant and the optimization only depends on the determinant of the covariance matrix of S. Defining the quality of the regression as the entropy of an associated distribution provides a more general, information theoretic formulation. It can be integrated into the same formula as for the classification (*i.e.*, plugging (3.5) into (3.1)), thus, defining a common framework for classification and regression. Putting all together gives the total formula for the information gain of a split for the regression task as:

$$\begin{aligned} \Delta H = &-\frac{d}{2}(1 - \log(2\pi)) + \log(|C(P_n)|) - \\ &- \sum_{i \in l,r} \frac{|P_i|}{|P_n|} \left(\frac{d}{2}(1 - \log(2\pi)) + \log(|C(P_i)|) \right) \ . \end{aligned} \tag{3.6}$$

Skipping the constant parts during optimization leads to

$$\Delta H \propto -|P_l| \log(|C(P_l)|) - |P_r| \log(|C(P_r)|) \ . \tag{3.7}$$

This criterion will be denoted as *Gauss entropy*. The main difference between *reduction-in-variance* and *Gauss entropy* in the context of regressing a $2D$ offset vector is that in the former the variance is estimated over the $1D$ distances to the mean, thus, defining an isotropic Gaussian over the $2D$ vectors. In contrast, in *Gauss entropy* the multivariate Gaussian is defined by a full covariance matrix estimated from the data in each child node. This enables the optimization procedure to find splits that do not necessarily reduce the overall variance of the distances but nevertheless reduce spread along one direction. The effects of choosing one or the other, as well as a new variant to measure the quality of the offset vector regression are demonstrated in experiments in Section 3.3.

Complexity and implementation details

The calculation of (3.2) for the *reduction-in-variance*, or (3.7) in the case of the differential entropy criterion has to be done for each split test in order to determine the one with the best information gain or lowest variance. Let s be the number of split tests to be

evaluated, t the number of thresholds to be tested on each individual split test and b the number samples in the current node. A straight forward implementation first evaluates each split test on the data ($\mathcal{O}(s\,n)$). Then it applies each threshold ($\mathcal{O}(s\,t\,n)$), thereby determining the subsets of the data for left and right child node. Finally, it evaluates (3.2) or (3.7) for each subset, which are linear in the number of samples in the subsets ($\mathcal{O}(n)$ for each subset). Thus the overall complexity is $\mathcal{O}(s\,t\,n)$.

The evaluation of each test on each sample has to be performed in any case. However, with a few observations an algorithm can achieve a runtime of $\mathcal{O}(s\,(n\log n + t))$. If this results in a speedup depends on whether $t > \frac{n}{n-1}\log n$. Even for a quite low value of $t = 10$ this is true also for large databases in the large majority of nodes deeper down the tree. In practice, in the experiments a significant speedup could be observed.

First, by sorting the results of the split tests (in $\mathcal{O}(n\log n)$) the left and right subsets for all thresholds can be determined in one pass over the samples. Second, all values needed to compute the information gain or the variance can be calculated incrementally, allowing for online computation within the single pass over the samples.

At the beginning all samples are in the right subset, the left one is empty. In order of increasing split test response value, samples are one by one moved from the right set to the left set. For each sample the information gain or offset vector variance is updated incrementally, as described below. The split test response value is compared with the current threshold. When arriving at a sample with a response value above the threshold, the subsets for the threshold and all derived information are ready to be stored or compared to the ones of the currently best split test, and we can proceed to the next threshold.

In the following, only the calculation for the left subset will be described, the computations for the right subset are equivalent (or inverse). The left side of (3.2) has the form

$$\sigma_l = \sum_{i=1}^{n} ||\mathbf{d}_i - \bar{\mathbf{d}}_n^l||^2 \;\;,\;\; \bar{\mathbf{d}}_n^l = \frac{1}{n}\sum_{i=1}^{n}\mathbf{d}_i \;\;, \tag{3.8}$$

where n is the number of samples in the left set. This can also be written as

$$\sigma_n^l = \sum_{i=1}^{n}\mathbf{d}_i'\mathbf{d}_i - 2\bar{\mathbf{d}}_n^{l\,\prime}\mathbf{d}_i + \bar{\mathbf{d}}_n^{l\,\prime}\bar{\mathbf{d}}_n^l = \underbrace{\sum_{i=1}^{n}\mathbf{d}_i'\mathbf{d}_i}_{s_n^2} - 2\bar{\mathbf{d}}_n^{l\,\prime}\underbrace{\sum_{i=1}^{n}\mathbf{d}_i}_{\mathbf{s}_n} + \underbrace{\sum_{i=1}^{n}\bar{\mathbf{d}}_n^{l\,\prime}\bar{\mathbf{d}}_n^l}_{n\bar{\mathbf{d}}_n^{l\,\prime}\bar{\mathbf{d}}_n^l} \;\;. \tag{3.9}$$

Along with σ_n^l the current values s_n^2 and $\mathbf{s}_n$ are stored. When sample $\mathbf{d}_{n+1}$ is moved from the right to the left subset the mean vector is calculated incrementally as

$$\bar{\mathbf{d}}_{n+1}^l = \frac{n\,\bar{\mathbf{d}}_n^l + \mathbf{d}_{n+1}}{n+1} \quad . \tag{3.10}$$

Similarly, the other terms in (3.9) can be calculated incrementally as

$$s_{n+1}^2 = s_n^2 + \mathbf{d}'_{n+1}\mathbf{d}_{n+1} \quad , \tag{3.11}$$

$$\mathbf{s}_{n+1} = \mathbf{s}_n + \mathbf{d}_{n+1} \quad . \tag{3.12}$$

Thus, σ_{n+1}^l can be calculated from s_n^2, $\mathbf{s}_n$, $\bar{\mathbf{d}}_n^l$ and $\mathbf{d}_{n+1}$ in $\mathcal{O}(1)$.

The same holds for the differential entropy (3.7). There, the term that changes when adding a sample to the left subset is its covariance matrix. It can be calculated as

$$C(P_n^l) = \sum_{i=1}^{n} \left(\mathbf{d}_i - \bar{\mathbf{d}}_n^l\right)\left(\mathbf{d}_i - \bar{\mathbf{d}}_n^l\right)' \tag{3.13}$$

$$= \sum_{i=1}^{n} \mathbf{d}_i\mathbf{d}_i' - \sum_{i=1}^{n} \mathbf{d}_i\bar{\mathbf{d}}_n^{l\prime} - \sum_{i=1}^{n} \bar{\mathbf{d}}_n^l\mathbf{d}_i' + \sum_{i=1}^{n} \bar{\mathbf{d}}_n^l\bar{\mathbf{d}}_n^{l\prime} \quad . \tag{3.14}$$

Again, all terms can be updated incrementally to compute $C(P_{n+1}^l)$ in $\mathcal{O}(1)$.

On a side note, when all patches in the set are almost (or completely) colinear, the determinant of the covariance matrix will get to zero. Thus, no matter how the set is split, the total information gain will always be zero. This is clearly not intended, since the regression could still be improved along the one direction in which the samples are still potentially widely distributed. A simple remedy to this is to clip the eigenvalues of the covariance matrix to a minimum value (which can also be done very efficiently for small covariance matrices).

Updating the entropy over a histogram, as used in the information gain (3.1), can also be done incrementally. The entropy of a histogram over samples in set S is defined as

$$H(S) = -\sum_{k=1}^{K} p_k \log p_k = -\sum_{k=1}^{K} \frac{h_k}{n} \log \frac{h_k}{n} = -\frac{1}{n}\left(\sum_{k=1}^{K} h_k \log h_k - h_k \log n\right) = \tag{3.15}$$

$$= -\frac{1}{n}\left(\sum_{k=1}^{K} h_k \log h_k - \log n \sum_{k=1}^{K} h_k\right) = \log n - \frac{1}{n}\sum_{k=1}^{K} h_k \log h_k \quad , \tag{3.16}$$

where K is the number of bins in the histogram, h_k is the number of samples in bin k of the histogram and $n = \sum_{k=1}^{K} h_k = |S|$ is the total number of samples in set S. When testing a sorted set of thresholds for a split function, the histograms in the left and right child nodes created by the next threshold are straightforward to update incrementally. Samples that now fall below the threshold are removed from the according bin of the right histogram and added to the left one. Thus Equation (3.16) can be updated incrementally in $\mathcal{O}(1)$ by subtracting the term with the old value of h_k from the sum, updating h_k and n and adding the new terms to the sum.

Stopping Criteria

The process of building a tree by splitting intermediate nodes is iterated (either depth first or breath first) until a stopping criterion is met and a leaf node is created. The most obvious case is if only a single data sample is left. Another criterion might be a minimum number of samples, or in the case of weighted samples, a minimum amount of total weight. This criterion is often applied to reduce the risk of overfitting and thus reduce the generalization error. This can also be achieved by using more (uncorrelated) trees in the ensemble which, however, leads to a linear increase in test time.

The decision not to split a node further can also be based on the observation that the best split does achieve an information gain above a given threshold. Usually, however, in standard Random Forests, as presented by Breiman, the trees are grown to full depth and are not pruned, unless the node is already completely pure (only samples of one class left) and there is no information gain at all. In the case of Hough Forests, an intermediate node that only contains positive samples can still be split based on the regression criterion in order to improve the precision of the estimation for the object center.

Another criterion that is mainly motivated by runtime considerations is a predefined maximum depth of the trees. This ensures an upper bound on the number of split tests that have to be performed on every test sample.

3.1.4 Testing

Running a Hough Forest object detector on a test image consists of two phases: A forest evaluation and voting phase operating on a patch around each pixel of the image and a subsequent non-maxima suppression phase.

Forest Evaluation and Voting

During testing, from a test image $\mathcal{I}$, small local patches $\mathcal{P}(\mathbf{y})$ with appearance $\mathcal{I}(\mathbf{y})$ are densely extracted at all locations (pixels) $\mathbf{y}$ and mapped onto the codebook entries, *i.e.*, the leaf nodes, by evaluating the respective split tests of all trees of the forest on $\mathcal{I}(\mathbf{y})$. Each leaf L of tree $\mathcal{T}_t$ stores the set of offset vectors D_L of all the training patches that fell into that node. In the following, $D_t(\mathbf{y})$ denotes the set of offset vectors stored in the leaf node of tree $\mathcal{T}_t$ that the test patch $\mathcal{P}(\mathbf{y})$ reaches, and $C_t(\mathbf{y})$ is the ratio of positive to negative training samples that reached this leaf node during training (*i.e.*, the probability of being foreground for the test patch at $\mathbf{y}$, according to tree $\mathcal{T}_t$).

Following the generalized Hough transformation [10] procedure, each local patch in the test image casts votes for object centers. This means, for each offset vector $\mathbf{d} \in D_t(\mathbf{y})$ a weight w is added to location $\mathbf{y} + \mathbf{d}$ in the Hough space. The weight w that each vote casts is defined as the probability of the patch to be foreground $C_t(\mathbf{y})$ distributed over all offset vectors in the leaf node, *i.e.*, $w = \frac{C_t(\mathbf{y})}{|D_t(\mathbf{y})|}$. This definition is somewhat adhoc. It follows the intuition that the contribution of a set of votes in a leaf node to a detection should be high, only if the foreground probability is high and the voting vectors are tightly clustered.

All votes from all trees are accumulated in the Hough space. Detecting objects of different scale is handled by scaling the input image and running the detector trained for a fixed scale for each input scale independently. The result is then a pyramid of Hough spaces with accumulated votes. Figure 3.1 shows a visualization of a Hough space pyramid color coded and superimposed and over the original image, for an image of the test set of the *TUD-pedestrian* dataset.

Non-maxima suppression

Local maxima in the Hough space indicate prospective object centers. Due to small deformations of the object and noise the votes usually do not perfectly agree on one single location, but are scattered around it. This also means that several local maxima in a small neighborhood usually stem from the same object. In order to prevent the system to report several detections for the same object instance, such smaller local maxima around a dominating have to be suppressed. If several scales are considered, this neighborhood also extends to neighboring scales. Several approaches to perform the non-maxima suppression are described in more detail in Section 3.5 and a novel, more principled approach is presented in Chapter 5.

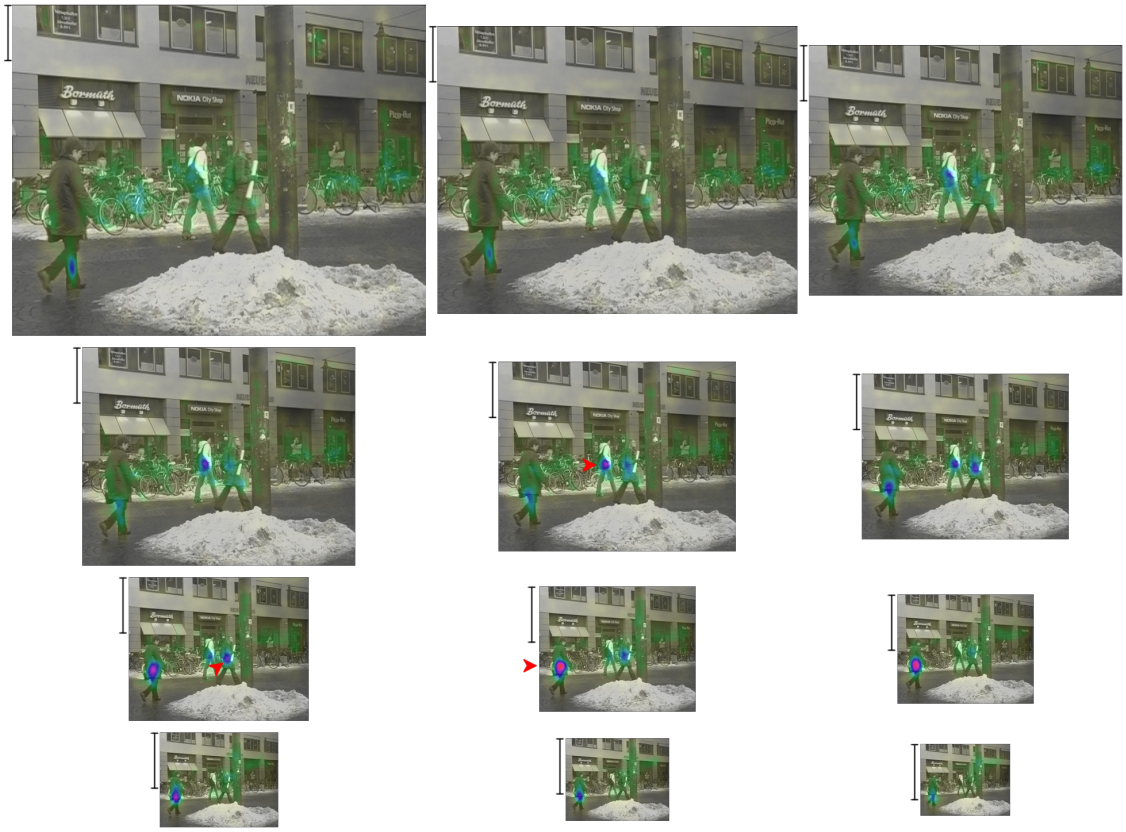


Figure 3.1: Hough space pyramid for a test image of the TUD pedestrian dataset. Transparent to green areas are low values, blue to purple are areas with high scores. Pedestrians of different sizes are detected in different scales. The height of the detector applied is indicated on the left of each image. The detector is trained to be invariant to small scale variations, in order to allow for the detecting objects in scales that are in between two of the discrete scales that are tested. Thus, neighboring scales of the true scale also show evidence for an object. Local maxima for each person in scale space are indicated by red arrows.

After the non-maximum suppression, the remaining local maxima indicate true object centers. The output of the detector is then a list of bounding boxes centered on each local maximum; width and height are derived from the respective scale where the maximum was found. Since, for reasons of computation time the resolution of the scale space is typically not fine enough to capture all sizes of occurring object instances exactly, it is important to interpolate between scales, in order to get more accurate bounding boxes. The score associated with each detection is the height of the local maximum in the Hough space. Figure 3.2 shows the bounding boxes of detections derived from the Hough spaces in Figure 3.1 after non-maxima suppression.

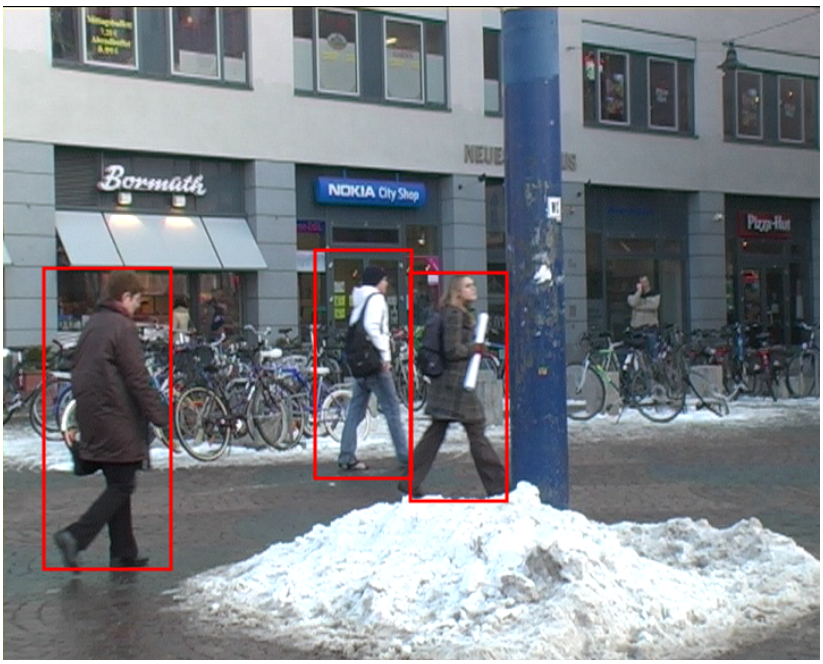


Figure 3.2: Detection bounding boxes recovered from the Hough space pyramid in Figure 3.1 after thresholding the detection score and non-maxima suppression. Notice also, how the non-maxima suppression eliminates the quite strong false evidence on the leg of the person to the left, visible in the first scale in Figure 3.1.

3.2 Leaf Node Post-Processing

Depending on the size of the training database and the maximal depth of the trees the leaf nodes might contain a large number of positive samples. All the offset vectors associated with those samples have to be used in the voting step. This leads to an increased test time. Thus, it is desirable to summarize the offset vectors and to create more compact prediction models. A simple possibility, that was also explored in [140], is to perform a mean shift [33] mode detection step and only keep a fixed number of dominating modes.

The mean shift mode extraction step reduces the true distribution of offset vectors to a small set of modes. Since the number of votes aggregated in each mode may vary considerably, it is important to associate a weight with each extracted mode, reflecting the density of the original distribution at that point. Different schemes have been proposed to determine this weight. Here, a formulation is explored that estimates a Gaussian over the set of samples that converged to each mode.

The weight for a mode is, thus, defined as

$$w_i = \frac{p_{\text{fg}}}{n\sqrt{2\pi}} \sum_{j=1}^{n} \exp\left(-\frac{\left|\left|\mathbf{x}_j - \overline{\mathbf{x}}\right|\right|}{\sigma^2}\right) \quad , \tag{3.17}$$

where p_{fg} is the probability foreground in this leaf node, n is the number of samples that converged to the mode during the mean-shift procedure, $\overline{\mathbf{x}}$ is the mean over those samples and σ^2 is the variance. Since sample mean and variance are calculated over the samples Equation (3.17) is in fact proportional to $\frac{n}{\sigma^2}$.

Experiments showing the effect of meanshifting on detection time and performance are presented in Section 3.6.1.7.

3.3 Offset-Vector Histogram Entropy for Regression Nodes

As stated in Section 3.1.3, the training of Hough Forests randomly chooses between optimizing the classification of the patches into foreground and background and optimizing the uncertainty of the prediction of the object center (regression) when choosing a split function for an inner node of the decision trees. Recalling Equation (3.2), in the standard Hough Forest, as presented in [66], the regression evaluation criterion is the variance of the offset vectors in the current node, *i.e.*, the sum of squared distances from the mean. This formulation forces the offset vectors that end up in a leaf node to be tightly clustered around a central point. The *Gauss entropy* criterium still favors unimodal, but in this case unisotropic distributions of the offset vectors.

However, both approaches do not handle multi-modalities in the distribution of offset vectors well, as illustrated in Figure 3.3, for an exemplary set of patches on a car. The node to split contains patches of the same sub-parts of the car, namely the wheels. Those patches look exactly the same, except for noise. A test that perfectly splits the set of patches in the current node into sets that cover the same part of a wheel, *e.g.*, the top parts and the left parts, does hardly decrease the variance because they are still located on very different regions of the object (in the front and in the back). Thus, such a split will not be chosen under the *reduction-in-variance* criterion. There might be a slight improvement in the *differential entropy*, depending on the alignment of the two sub-clusters, but a much larger gain can still be achieved by splitting into a set patches on the left and one on the right.

Additionally, in the first few levels of the tree it is virtually impossible to significantly divide the set of all patches based on their appearance into compact clusters, unless all

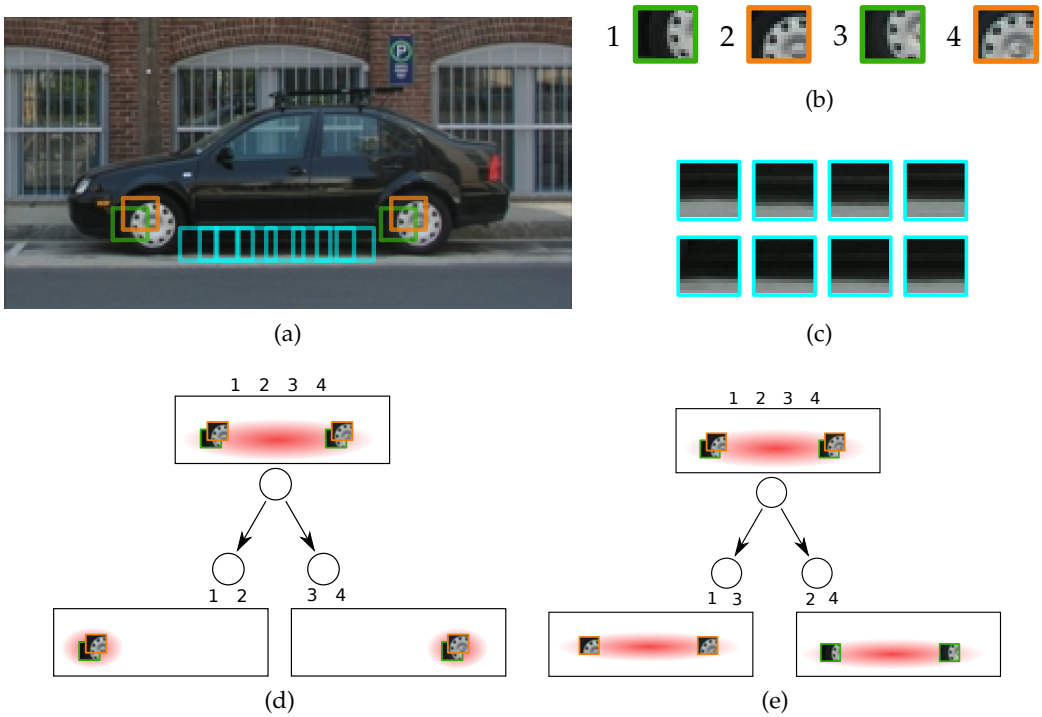


Figure 3.3: (a) Sample patches on a training sample for a car detector (taken from the *ETHZ-cars* database [92]). An intermediate node of a decision tree might contain similar looking patches like the ones in (b) and (c). Since the patches in (b) come from sub-parts of the object that are effectively the same (a wheel), discriminating patch 1 from 3, or 2 from 4 can only be done by overfitting to noise. Also, this kind of discrimination is not desirable; if the system is looking at a patch like 1 or 3 in a test image, it should indicate that the center of a car might be to the upper left or the upper right of that patch, not decide for one direction. However, the difference between patches 1 and 2 is significant, since they can be judged from their appearance to come from slightly different positions on the object. Being able to tell the difference between 1 and 2 (or 3 and 4, respectively) effectively increases the precision of the voting. Thus, the training procedure should favor appearance tests that split the set of patches into subsets $\{1,3\}$ and $\{2,4\}$, as shown in (e). However, splitting in to $\{1,2\}$ and $\{3,4\}$, as shown in (d), results in a much bigger gain, if computed with the *reduction-in-variance* or *Gauss entropy* criterion, as indicated by the much smaller red areas, showing the covariance of the distribution of the patches. Additionally, in (c) the set of patches is spread out quite considerably. Taking all the offset vectors of this set to vote for the center of the object does result in a quite accurate estimate in the y direction, but with a very big variance in x. However, as in (b), it is not meaningful to try to reduce the uncertainty of the voting in the x direction, since again differences in the appearances are only due to noise and not discriminative features of the object. Here, the evaluation should favor splits that separate this set of patches from others from above and below.

patches on one side of the object look the same and completely different from patches on the other side.

Thus, the trees cannot start by clustering together different parts of shapes and only later focus on specifying the exact locations, if they are scattered around multiple locations in the object. To address these problems we introduce a new evaluation criterion for the regression nodes. We divide the space into a grid of cells, forming a two-dimensional histogram. Offset vectors are assigned to the four closest bin centers by linear interpolation. A split dividing the set of patches into two partitions for the left and right child nodes thus results in two histograms of the respective distributions of the offset vectors. The quality of the split is then measured by difference of the entropy of the histogram over all samples in the parent node and the entropy of the histograms in the left and right child nodes, weighted by the size of the respective set:

$$\Delta H = H(I_l \cup I_r) - \frac{|I_r|}{|I_l| + |I_r|} H(I_l) - \frac{|I_r|}{|I_l| + |I_r|} H(I_r) \ , \tag{3.18}$$

where the entropy H(I) is defined as

$$H(I) = \sum_{x,y} p_I(x,y) \log p_I(x,y) \ . \tag{3.19}$$

Here $p_I(x,y)$ is the percentage of samples of set I assigned to the bin at (x,y):

$$p_I(x,y) = \frac{h_I(x,y)}{\sum_{x,y} h_I(x,y)} \ . \tag{3.20}$$

This new criterion will be denoted as offset vector *histogram entropy*.

This strategy effectively turns the regression problem into a classification problem where patches are soft-assigned to the four nearest histogram bins, which can then be seen as a set of distinctive foreground classes. It is somewhat similar in spirit to [98], where also the outputs, in this case local edge structures, are clustered into different classes, which are then used in a standard multi-class learning setup.

Note that the use of a histogram in the evaluation procedure and the discretization of the target values does not influence the collection of information in the leaf nodes and the subsequent voting procedure. The offset vectors do not necessarily have to be quantized into this set of distinctive classes for these subsequent steps. The histogram is only used for the evaluation of split functions.

Also note that using a histogram is possible in this case, because of the known and

limited extent and the low dimensionality of the offset vectors. For a task like pose estimation of a rigid object, where 3 position and 3 rotation parameters have to be regressed, this would maybe not be practicable.

One parameter to choose is the size of the histogram bins. It mainly depends on the size of the objects to detect (training scale), and the precision of the voting (*i.e.*, the width of the Gauss kernel used for smoothing the raw Hough maps). For instance, in the pedestrian detection experiments, where the object size is roughly 40×100, the cell size is set to 8×8 resulting in a 5×12 grid of cells. These parameters were not evaluated extensively, but a set of preliminary experiments showed that it is not very sensitive and the above settings give good results.

Just as the *reduction-in-variance* and the *Gauss entropy* criterion in Section 3.1.3, also the histograms can be calculated incrementally while sweeping over the thresholds. First, all samples are in the right histogram. In one pass over the samples all samples with a response value lower than the current threshold are removed from the right and inserted into the left histogram. The entropy of the two histograms can also be updated incrementally, without passing over the whole histogram, just as in Equation (3.16). Thus, the dimensionality of the histogram does not influence the speed of the calculation. Again, whether this results in a speedup depends on the relation of number of thresholds tested vs. the number of samples in the node.

3.4 Dataset Subsampling per Node

The split tests and the criteria to evaluate them have to be evaluated on all samples that end up in an intermediate node during traing. Especially in the root node, which contains the whole dataset, and also in the first levels below, this is quite expensive for large datasets.

An alternative to testing all samples in the node has been presented by Schulter *et al.* [130]. Their online formulation of Random Forests builds the trees incrementally, by feeding the samples as they arrive and splitting nodes as soon as enough data is available for them. Thus, to evaluate split tests and to create the tree structure, only a small set of samples is considered per node. Interestingly, simulating this behaviour in the offline case, by only considering a small, fixed size subset of the data available for a node, even resulted in increased performance of the final forest in several cases. The method was also adopted successfully by Cootes *et al.* [34] in the context of facial landmark localization.

Obviously, this method provides a dramatic speedup for the training of the split tests, especially in the first few levels of the trees. But there is another interesting aspect to this approach. It can be considered as a variant of bagging, but not per whole tree but on the node level. The decision of the root node of each tree is based on a very different subset of the data. This decreases the correlation of the trees, just as intended in bagging. In contrast to bagging, however, in deeper levels of the trees, the fixed size subset encompasses more and more of the available samples and thus gradually the whole dataset is considered. All trees see all the data, but in different sub-partitions created by the higher levels. This might explain the good performance of this approach.

In the experiments in this work, generally 50 samples per class are used. This means in every node a subset of 50 positive and 50 negative samples is drawn from the available samples in the node to evaluate the split test candidates. In case the training set is unbalanced (*e.g.*, twice as many negative than positive patches) the samples are weighted to account for this fact. The subsampling per node then takes as many samples from each class as necessary to reach a minimum total weight, such as to produce a stratified subsample.

3.5 Non-Maxima Suppression

A crucial element of many object detection systems is how the final output, usually in form of detection bounding box rectangles, is calculated. The core of most systems, and also the central part of innovations, is the method that creates and scores individual detection hypothesis. However, most of the time, these results need some kind of post-processing to get to the final results. Often the central part of the algorithm delivers a series of overlapping detections that, in fact, capture the same object instance. These have to be processed in a final step consisting of some kind of non-maxima suppression. In the end, in many application scenarios, each object instance should only be reported once and multiple secondary detections will count as false positives and lower the overall performance scores.

In many publications, this step is considered to be trivial and either completely left out in the description or just mentioned in one or two lines. However, as everyone working in object detection will have experienced, this final post-processing step has a strong influence on the overall performance of the system.

Additionally, the question of how to produce the final output is also interlinked with the way the system is evaluated. The most widely used metric comes from the

evaluation protocol of the Pascal VOC Challenge [49], one of the most common object detection benchmarks. It considers a detection to be a true positive if the overlap (intersection over union) of the detection rectangle with the ground truth annotation of a real object instance is higher than 50%. Yet, out of multiple detections of the same object only the most confident one is considered true positive; the others are counted as false positives. However, as mentioned above, the required output from an object detection system depends on the task. For instance, if the task is to detect pedestrians in images from a camera mounted on a car, such that the car can warn the driver or even brake automatically, the most important criterion is to not have any false negatives. False positives in a scene without any person are similarly problematic, because they render the system useless and can also lead to dangerous situations with a car hitting the brakes hard for no apparent reason. On the other hand, additional false positives very close to a true positive can easily be ignored in the evaluation since the car must warn or stop anyway. Equally, detecting people behind other people might not be relevant for this task. Failure to detect those additional true positives should thus not be punished in the evaluation. Thus, in this case, during non-maxima suppression all evidence for persons further away (higher up vertically) at the same angle (x-direction) bould be excluded from the search.

For rigid detectors that return a single confidence score for each location in scale space the usual form non-maxima suppression is based on the bounding boxes of detection candidates (*e.g.*, [56], [41]). First, all detection candidates are sorted by their score, not considering those with a score below a minimal threshold. Then, the currently highest scoring bounding box is chosen and added to the list of detections outputs. All remaining candidates are checked for the overlap with the selected detection. If the overlap is above a threshold these bounding boxes are deleted from the list of candidates. This procedure is repeated until no candidates are left.

In the implementation of the Viola and Jones detector [155] in the OpenCV library [22] a slightly different method is used. There the classifier does not deliver a confidence score but only a binary decision. Detection output rectangles that are classified as positive but heavily overlap each other are consequently grouped together and a mean rectangle is calculated as overall output. It is important to notice that at least in this implementation a big gain in performance can be achieved by only accepting outputs that result from a grouping of at least 3 detections. A single location classified as positive without any neighboring detections will not be considered for the output. This method builds on the assumption that the detector was trained to be slightly translation

invariant and thus fire multiple times around a true positive. It effectively eliminates spurious false positives.

Since Hough Forests [66] make use of the Generalized Hough Transform to gather the evidence for object instances, it is straightforward to do the non-maxima suppression in Hough space. After the global maximum is found and the according detection is reported all evidence for neighboring hypotheses is erased from the Hough space, before the next maximum is sought.

However, there are still several choices to make. One is the size and form of the neighborhood to erase. In many applications of the Hough transform usually a small rectangular area (or hypercube, in the multidimensional case) around the last chosen maximum is erased, such as to avoid picking an extremely similar hypothesis at the next step. Another alternative is to adjust the range in order to exactly erase all those hypotheses that create detections which should be rejected due to significant overlap with the central detection. For example, deleting all hypotheses that correspond to detection rectangles with a overlap of more than 50%, as defined in the PASCAL challenge, would result in a diamond shaped suppression region in the Hough space (if only one scale is considered). In the implementation of [66] the authors chose to erase all hypotheses that have its center within the bounding box of an already selected detection.

Another question is how to deal with multiple scales. Typically each scale is processed individually, by resizing the input image and voting into a Hough space of the same size as the input. The result is a scale space pyramid of Hough spaces. Evidence for an object will typically also appear in neighboring scales, since the detection procedure has to be robust to slight scale changes, in order to deal with objects that do not appear in exactly one of the discrete scales that are tested. In keeping with the first alternative above, one can choose to erase evidence only from the directly neighboring scales in the pyramid. For that purpose, the center of the detection and the range of the neighborhood must be adjusted to the corresponding scale.

However, in [66] the implementation is different. One scale is selected as reference scale, and the Hough spaces of all others are resized to this common resolution. The resulting structure is a stack of equally sized Hough spaces. This makes the interpolation in scale easier, since hypotheses of different size are at the same x and y coordinate in each scale. However, the reference frame is chosen as the smallest resolution and the Hough spaces from higher resolution input scales get down-sampled. Sharp peaks of correct detections get smoothed out or almost erased by the down-sampling to a much lower resolution. Thus, this method is only suitable if a very narrow range of scales

is tested. In a generic setting, where objects can appear in a wide range of scales, the non-maxima suppression has to be performed, as described above, in the full pyramid of scales.

Experiments with the two variants described above are presented in Section 3.6.1.9. More sophisticated methods for resolving evidence for mutually exclusive detection hypotheses will be presented in Section 5.

3.6 Evaluation

The list of alternatives for several aspects of Hough Forest, given in this chapter, leaves us with choices to make and parameters to set for the training. Thus, this section presents a thorough evaluation of a range of different setups.

A central observation throughout all the experiments with several variants of Hough Forests and its extensions was that the overall performance strongly depends on the parametrization. The setting of parameters is not very sensitive, in the sense that the range of acceptable values is mostly not extremely narrow. However, central parameters and their reasonable value range heavily depend on each other. Switching between different extensions and options of, *e.g.*, the criteria used to evaluate the quality of a split test for a node, or the way information in the leaf nodes is post-processed, only makes sense and improves the final detection results if also other parameters are adjusted accordingly.

Additionally, in our experiments, the performance of some setups has shown quite large variance over different runs (although, typically, good parametrizations also lead to quite stable results). Nevertheless, many published results for systems based on Random Forests (or Boosting, or any other method that includes some kind of random process) report only mean performance over a set of runs without stating the range of typical outcomes (*e.g.*, by reporting the standard deviation), or even just the results of a single run. Such a limited display of results should raise skepticism.

The interdependence of parameters combined with the effort it takes to thoroughly evaluate a single fixed setup makes it really hard to jointly optimize over all parameters. Fortunately, usually there are some restrictions on reasonable parameter ranges. A good starting point for the optimization are the run-time requirements, raising questions such as: How many trees and up to which depth can be evaluated during run-time (in terms of total time per frame and/or energy efficiency)? Is training time an important issue? Is the runtime- and/or training-platform capable of parallelization? Starting from those

parameters fixed by requirements, a coordinate ascent approach can be used to optimize the others. This, however, means that any kind of conclusion about the superiority of one method over the other can only be valid for the specific setups for which they have been tested. Any kind of generalization beyond that should at least be taken with a grain of salt.

In the following, the most important parameters, their influence and their interplay are presented. Experimental evaluations follow below.

Maximal Tree Depth It is common practice to set a maximal depth a tree can grow to. Nodes at this maximum level are converted to leaf nodes, although they might contain many samples with distributions that do not allow for a clear decision for a class or a value to regress. The main reason to set a maximal depth is to bound test time, since at each node in a long path down a tree a test has to be performed. If, on the other hand, the information in the leaf nodes is the whole list of samples, not summarized in any way, like in the original Hough Forest formulation [66], and all samples have to be considered at run-time, training deeper trees might even speed up the evaluation (as shown in the experiments in Section 3.6.1.3), since less samples end up in each leaf node. Whether limiting the trees in this way influences the performance depends mainly on how many samples are left in the nodes at the last level and how meaningful the statistics over those samples is. This, in turn, depends on the distribution of the data (multi-modality of the distributions of each class), the size of the database, the type of split functions and the method to evaluate split functions, influencing how balanced the trees are.

Number of Trees Generally, the more trees are used the better the classifier. Due to the uncorrelatedness of the trees, the ensemble is not susceptible to overfitting [23]. Thus, in standard machine learning tasks, where the evaluation is done on each sample, usually the performance rises up to numbers of more than 100 or even 200 trees in the ensemble.

However, in Hough Forests we observe that a large number of trees does not increase the performance significantly. In fact, beyond 10 to 15 trees the performance is more or less saturated.

In a Hough Forest-like detector the importance of a correct classification for each tested input patch is not that high. The estimation of the probability for an object in the Hough space is not only influenced by the averaging over the trees but also by the high number of patches that are sampled from a test image and contribute

to a maximum. Additionally, the Gaussian smoothing that provides invariance to small translations of the individual patches adds to the robustness. This might be the reason why the performance gets saturated with a much lower number of trees in the object detection task with Hough Forests than on machine learning tasks with Random Forests.

As a practical consideration, more trees result in longer training and testing time. On the other hand they can be trained and evaluated in parallel.

Minimum Number of Samples in Leaf Node One way to reduce overfitting of individual trees and to gather more reliable statistics in each leaf node is to set a minimal number of samples in each leaf node. If during the training of the trees the number of samples in an intermediate node is below this number, or no split can be found that creates two child nodes that each fulfill the requirement, the splitting is stopped and the node is converted to a leaf node. If the samples in the training set are weighted (*e.g.*, to balance the dataset) this criterion can also be expressed as a minimum total weight of all samples in the node.

However, the importance of this criterion depends on other factors. One of those is the size of the database used to train each tree, in combination with the maximal tree depth. For instance, a tree trained with 2 million samples to a depth of 16 (resulting in a maximal number of $2^{16} = 65536$ leaf nodes) will never encounter a node with really few samples, unless, of course, the tree is very unbalanced. This, in turn, depends strongly on the type and number of split test functions that are evaluated to find a good split, as well as the metric to evaluate them.

Additionally, as mentioned above, another way to counterbalance inaccurate predictions resulting from poor statistics in one leaf node is to increase the number of trees, thereby reducing the importance of the setting of minimum number samples per node. In fact, in machine learning tasks, where the run-time is not that critical and large numbers of trees can be used, the trees are often grown to full depth, without restriction of tree depth and until there is only one sample left [23].

Type of Regression Split Node Tests The choice of a criterion for the evaluation of a split test to assess the gain in performance of the regression results in quite different partitions of the data in each node and, thus, distribution throughout the tree. If the trees are grown deep enough, the importance of this criterion is again reduced, because the clusters will be quite compact with any of the criteria. How-

ever, as will be shown below, this choice has a quite profound impact on the tree depth that is required to achieve top performance.

Type of Leaf Node Post Processing The offset vectors in the leaf nodes can be either stored as plain lists of all samples arriving in the leaf node or post-processed as described in Section 3.2, by describing the distribution with a parametric model or clustering with mean-shift or other types methods. The choice which variant to take mainly depends on the data that typically reaches a leaf node.

If, for instance, the trees are grown to full depth, with only one sample remaining per leaf node, there is simply no information left to aggregate. Also, the choice of the type of split node evaluation function influences whether the distributions of samples in a leaf node tend to be multi-modal or clustered more tightly around a single mode. Of course, this also depends on the amount and nature of the data.

Database Size Typically, in most machine learning tasks the intuition, as well as the experience, is "the more, the better". Generally, this is also true for the task at hand. However, as mentioned in the discussion above, other factors depend on the amount of data. Keeping the setup the same and just increasing the number of training samples, thus, does not necessarily increase the performance.

The evaluations in [140] show drastic improvements for taking large amounts of training data. In their setting collecting more data (virtually unlimited amounts) is easy, because it is created artificially by rendering parametrically adjustable models of human bodies in a huge range of possible poses. In typical learning tasks the data is given and it is hard to collect more. In the case of object detection with Hough Forests on the datasets used in this work, the number of training images is limited. One way to increase the database size are artifical transformations of the input images, *e.g.*, to account for variations in scale. Furthermore, patches are sampled from the training images and the amount of patches can be increased (until all images are densely sampled). However, sampling densely and from images that are transformations of each other will likely result in redundancy. Thus, it is expected that the performance is saturated long before all possible patches are included in the dataset.

3.6.1 Experiments

This section presents experiments to provide empirical data and substantiate the theoretical analysis above. Each subsection is focused on one of the parameters in order to identify its influence in relation to others. In particular, all setups show variations of the discussed parameters while keeping all others at a default value. If not stated otherwise, these are: Database size = 10 patches per image (*i.e.*, 32000 positive samples for the TUD pedestrian detection setup), twice as many random negative patches, number of trees = 15, maximum tree depth = 50, *histogram entropy* as split node evaluation criterion, minimum number of samples per leaf node = 20, meanshift mode seeking in leaf nodes is turned off. Non-maxima suppression over multiple scales is performed in a Hough space cube, as described in Section 3.5. Generally, evaluations in this chapter judge the resulting detectors based on Precision/Recall Curves (PRCs). These curves are generated by varying the threshold on the detection score, resulting in a specific number of objects that can be recalled and an associated ratio of true and false detections in the reported output. Results of multiple runs of a fixed setup are averaged by combining all lists of detection outputs sorted by their confidence and generating a joint PRC. When many such curves have to compared they are summarized by the area under the curve (AUC). Where suitable, individual PRCs are shown.

3.6.1.1 Training Data

Throughout this thesis a number of databases will be used to demonstrate the effectiveness of different object detection algorithms and the effects of variations of parameters and choices of methods. The three main application areas will be car detection, pedestrian detection and multiview face detection.

Car Detection The *ETHZ-cars* [92] dataset contains images capturing cars under 7 different viewpoints. For each viewpoint, 60 images are randomly selected for training and the remaining images are equally split into validation and test images. The training set thus consists of 420 positive images, the validation set contains 428 and the test set 429 images. Additionally, also the horizontally flipped version of each image is considered. Figure 3.4 (a) shows some examples.

Preliminary evaluations where also performed on UIUC cars [2]. However, this can only serve as a validation set or sanity check, since the benchmark is already completely saturated by the baselines.

(a) ETHZ-cars.

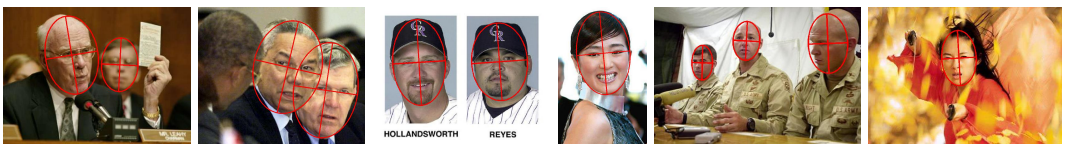
(b) TUD Pedestrian, train + masks.

(c) TUD Pedestrian, test.

(d) TUD crossing.

(e) TUD campus.

(f) AFLW + annotations.



(g) FDDB + annotations.

Figure 3.4: Sample images of the benchmark databases.

Pedestrian Detection The *TUD-pedestrian* [6] dataset consists of 400 training images and 250 test images containing 311 pedestrians in typical street side scenes. Samples are shown in Figures 3.4 (b)-(e). The test images have a resolution of 720×576 and feature mostly only one to three pedestrians on a range of different and sometimes challenging backgrounds. Targets appear in various scales, mostly relatively small compared to the image size, but not smaller than a vertical resolution of 70 pixels, with the median and also mean height around 215 pixels. The related test image sequences *TUD-crossing* and *TUD-campus* consist of 201 and 71 images, respectively. Both also show pedestrians in side views, however, in contrast to the *pedestrian* test set, those two sets consist of frames taken from continuous video sequences, showing pedestrians walking in front of (more or less) static backgrounds. On *TUD-crossing* the pedestrians appear in a very small range of scales, whereas on *TUD-campus* the variation in scale is much bigger. *TUD-crossing* features the most crowded of the three scenes, with many of the persons walking close-by, in front of one another and thus occluding each other. Thus, the three test sequences show quite different characteristics and pose very different challenges for an object detector, as will become evident in the experiments.

The training set comes with accurate segmentation masks for each person. The test sequences are annotated with bounding boxes. The original annotation of the *TUD-crossing* sequence only includes fully visible persons and, thus, ignores many persons that are partly occluded. This is why Riemenschneider *et al.* [126] presented a new annotation with detailed and accurate segmentation masks for every visible person, from which accurate bounding boxes can be derived for conventional PASCAL-overlap evaluation.

Another pedestrian detection database that is used is this thesis is the PETS 2009 Benchmark Data [64]. It includes several datasets of which View001 of sequence S1.L1 again features pedestrians mostly in side views, which makes it suitable for evaluation of methods trained on the TUD pedestrian training set. The sequence shows several groups of people walking closely and thus heavily occluding each other. In total the ground truth annotation contains 4348 persons.

Multiview Face Detection The Face Detection Data Set and Benchmark (FDDB) [75] is currently one of the most extensive benchmark for real-world, multi-view face detection. It contains annotations for a subset of images of the Faces in the Wild dataset [15]. In total, there are 5171 faces in 2845 images. The annotation is in form of an elliptical region capturing the frontal part of the head, defining extent and in-plane rotation. This

form of annotation is more natural for faces since the front of a human head can be roughly described as an ellipsoid, that always results in an ellipse when projected into an image, no matter the pose. In contrast, with classical bounding box annotations it is tough to define a common annotation scheme that fits frontal, as well as profile views of faces. The elliptical annotation makes evaluation of detectors that return bounding box rectangles harder, since no rectangle can perfectly fit to the ellipse. However, the dataset comes with a set of tools that takes care of the evaluation, given detection outputs in form of either ellipses of rectangles and thus fixes the evaluation protocol. Examples of test images with annotation are shown in Figure 3.4 (g).

The AFLW Database [84] is a large scale collection of 25,993 faces in 21,997 images in real-world situations. It comes with detailed annotation of 21 facial landmarks. From those landmark positions an ellipse (as defined in FDDB) and bounding box rectangles, as well as the head pose (roll, pitch, yaw) can be derived, by fitting a generic $3D$ face model. In the experiments below it will serve as the training database. Examples of images with annotations are shown in Figure 3.4 (f).

Data Preparation When preparing a set of images for training a detector several points have to be considered. Generally, to achieve invariance to scale and in-plane rotation, the detector can deal with this variations by making the input representation invariant. However, making the representation invariant results in a loss of discriminative information. Thus, usually the detector is trained on a fixed rotation and scale and detection of rotated and scaled objects is achieved by rotating and scaling the input and applying the detector. Since the set of tested rotations and scales is discrete, the detector still has to be invariant to variations within the range of the discretization.

Instead of making the representation invariant, the training algorithm of the detector can be confronted with examples of all variations that it should be able to handle. In the experiments with the *TUD-pedestrian* dataset, for instance, the training set is augmented with 3 randomly rescaled versions of each image, in the range of $[0.9, 1.1]$. Additionally, to make the detector symmetric, usually horizontally flipped versions of the input images are added to the dataset (notice that in the *TUD-pedestrian* training set all persons are facing left). In total, the *TUD-pedestrian* training set is thus increased from 400 to 3200 images.

In the experiments that use AFLW as training data, where the in-plane rotation of the faces is known from the approximate pose estimation, all faces can be oriented upright, again, adding a small amount of jitter in the rotation angle to increase robustness.

If the detector is trained on a fixed scale, this scale has to be chosen as well. Generally, the resolution has to be high enough to capture discriminative details. However, with large resolutions also the dimensionality of the description can get very high dimensional, causing the classifier to overfit on random details and noise. Thus, in many settings very small resolutions have proven to result in the best overall performance. Face detectors are often trained on patches as small as 36×36 [85], or even 24×24 [22]. The latest state-of-the-art in pedestrian detections uses a resolution of 64×128 [13, 46, 157]. However, in [13, 46] the representation derived from this resolution is again downsampled by a factor of 4 (*i.e.*, to 16×32) before training and inference.

In Hough Forests, also the connection between image resolution and patch size must be considered. The relation should be chosen such that the patches are small enough for efficient classification, but large enough to capture relevant and discriminative structures. In pratice, in this thesis, the patch size is always kept at 16×16. The pedestrian training images are scaled to a height of 100 pixels. Face images from AFLW are resized such that the distance between the eyes is 30 pixels, and cropped such that the center between the eyes is horizontally centered in the training image patch and vertically at one third from the top. The overall size of the cropped patch is 120×120.

3.6.1.2 Database Size and Bootstrapping

The first experiment shows the influence of the size of the database used for training. As mentioned above, although the number of training images is typically given by the limited size of the collected database, in the Hough Forest setting the size of training database can be easily varied by setting the number of patches extracted from each training image.

Figure 3.5 shows the performance of detectors trained on datasets for which 2, 5 or 10 patches were extracted from each positive training image. The solid curves show runs where for each positive patch first one patch was extract from random positions in the background and then in a bootstrapping run, again the same amount of hard negative samples was collected, such that the final ratio of negative to positive samples is $2 : 1$. The final detectors where trained from scratch on this augmented dataset. The dashed lines show runs where for each positive patch two negatives were randomly extracted from negative training images right from the start and no bootstrapping was performed.

The plots show that taking more data only slightly increases the overall performance. Also, when taking more data, trees have to be trained deeper (additional experiments in Section 3.6.1.3) to differentiate well, especially if the data in the leaf nodes

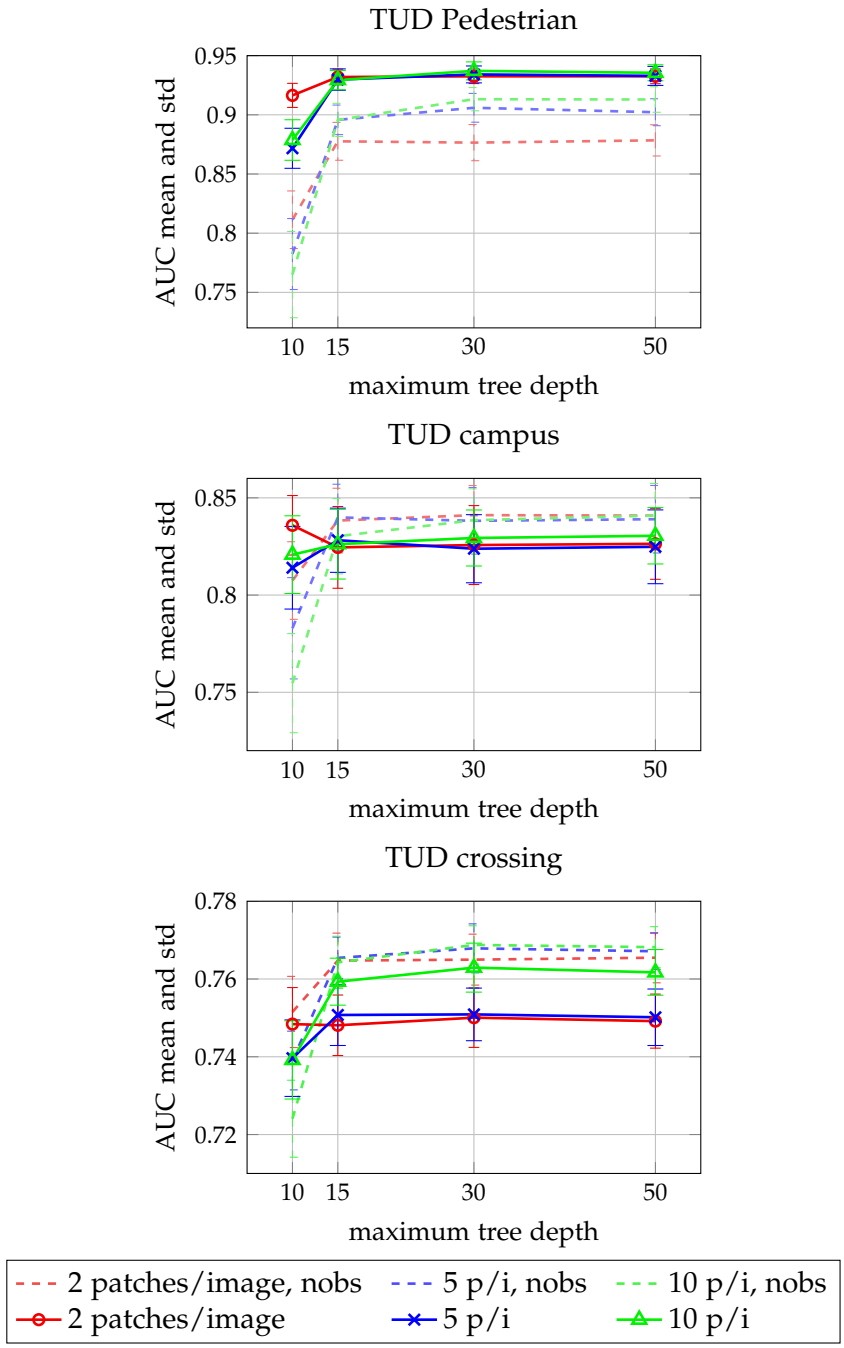


Figure 3.5: Influence of the number of patches extracted from the training images of on the detector's performance on the three test sets. Taking only 2 patches per image already shows good performance that can be increased only slightly by cropping out more patches. Bootstrapping has a big positive influence on *TUD-pedestrian*. On the other two sequences, performance is reduced a little, although without bootstrapping (dotted lines) the variance is slightly higher.

is post-processed to reduce the number of voting elements (additional experiments in Section 3.6.1.7). It has to be mentioned that, although taking only 2 patches from an image seems to be very little, the dataset was created by adding 3 randomly scaled versions of each image, as described above. Thus, in fact 8 patches are taken from each image. This is still a quite low number, which demonstrates the power of ISMs to synthesize new detection from a relatively low number of input patches.

Bootstrapping has a huge positive impact on the performance on *TUD-pedestrian*, with an increase of over 5% for the smallest dataset and still more than 2% for the best performing setup. Also in tests on FDDB an improvement of over 10% precision at 200 false positives was observed. However, on *TUD-crossing* and *TUD-campus*, where the backgrounds are not that diverse and relatively easy, bootstrapping even has a slightly negative impact, clearly showing the diversity of the test sequences.

3.6.1.3 Tree Depth

Figure 3.6 shows the effect of varying the maximal depth of the trees of an object detector trained on the TUD pedestrian dataset on the three related test sequences. All other parameters are kept fixed to values that in total generate the best known results. In this setup, the results consistently improve with increasing tree depth, until around 30. Beyond that the improvements are not statistically significant.

Since the maximal tree depth directly influences the number of samples and, thus, the statistics in the leaf nodes, it also has essential influence on the behaviour of many other parameters. Thus, most of the evaluations below also include variations over this parameter. Overall, however, a maximal depth of 50 performs best across a wide range of other parameter settings.

As stated above, most of the evaluations in this chapter judge the resulting detectors based on the area under the curve (AUC) of its precision-recall curve (PRC). While being an established, widely adopted and useful measure, it summarizes and thus hides the actual shape of the PRC. This is necessary for large experiments, since simply showing all PRCs of several runs of several setups would just result in bloated and confusing charts. However, when going back to the individual PRCs, there is a trend visible, showing that forests with deeper trees, despite of having a bigger AUC score and holding good precision up to higher levels of recall, tend to produce more high scoring false positives. This is shown in Figure 3.7, for a standard setup. Figure 3.7 (a) shows mean PRCs for varying maximal tree depth. The curves for depths 50 and 100 clearly show spikes right at the beginning in the upper left, caused by high scoring false positives.

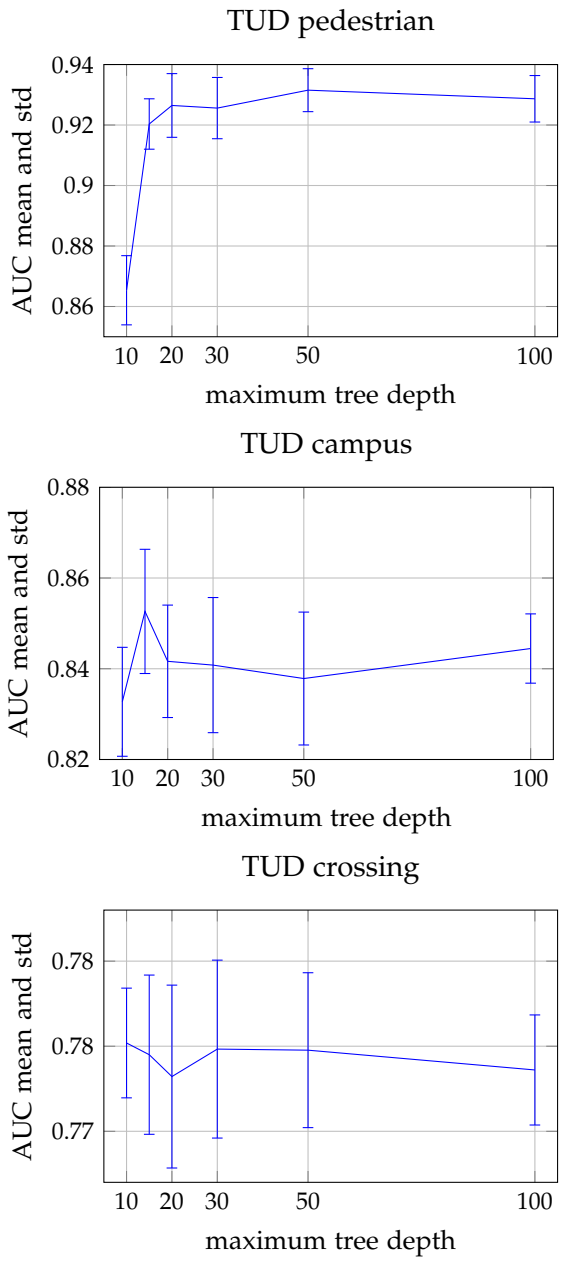
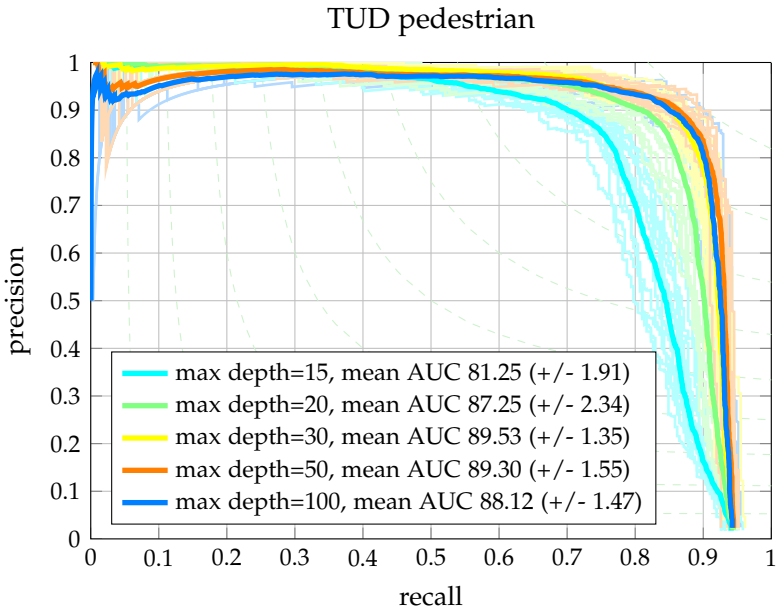
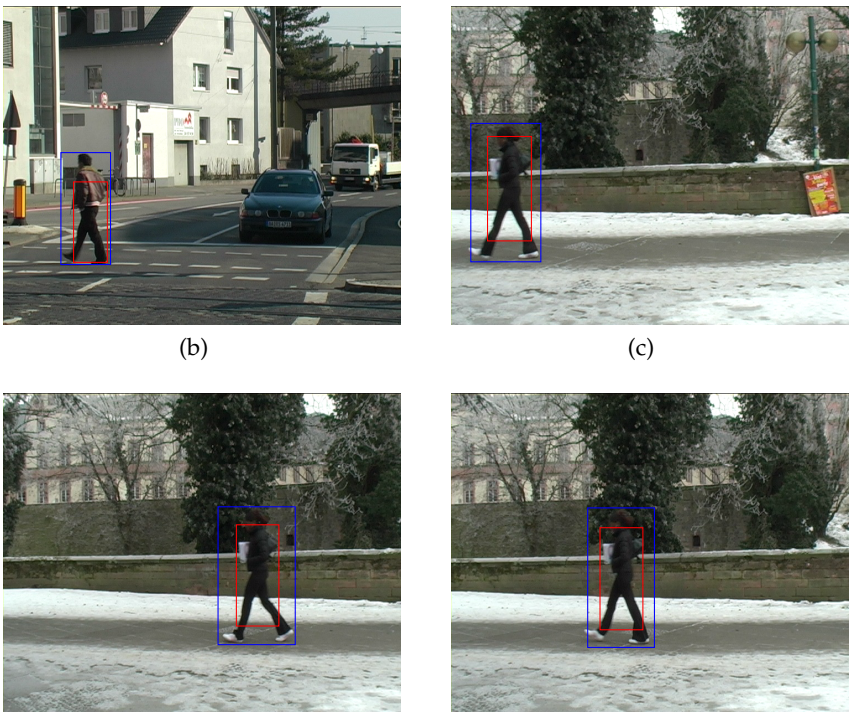


Figure 3.6: **Tree Depth**. Accuracy (Area Under the Curve) of a pedestrian detector on the three test sets of the TUD pedestrian database. The x-axis indicates the *maximum depth* at which splitting the nodes of the trees is stopped and leaf nodes are created. Down to a maximum depth of about 30 the accuracy increases significantly, at least for the pedestrian dataset. On the campus and crossing sequences, the background is static and much less cluttered. Beyond a maximum depth of 30 the results are saturated and more or less stable (within the region of confidence of the estimate). Mean and variance are calculated over 15 runs for each setup.



(a) ROC curves



(b) (c) (d) (e)

Figure 3.7: PRC curves of Hough forest detectors trained to different maximal depth levels. Deep trees result in detectors with high AUC, but tend to produce few very high scoring false positives. Sample images of such false positives in (b)-(e) show that many of them are on a pedestrian, but on a slightly too small scale.

Figures 3.7 (b)-(e) show examples of such false positives, to which the detector assigns a very high confidence. All of these samples show detections that actually are on a person, but in a slightly too small scale, such that the PASCAL overlap with the ground truth is below 50%. The detection in 3.7 (b) is even the strongest of all responses of the respective detector on the whole database. The reason for this behaviour is not entirely clear. Training deeper trees results in less votes per leaf node (without leaf node post-processing by meanshift) and thus sparser votemaps. This, however, does not completely explain this bias towards smaller scales.

3.6.1.4 Number of Trees

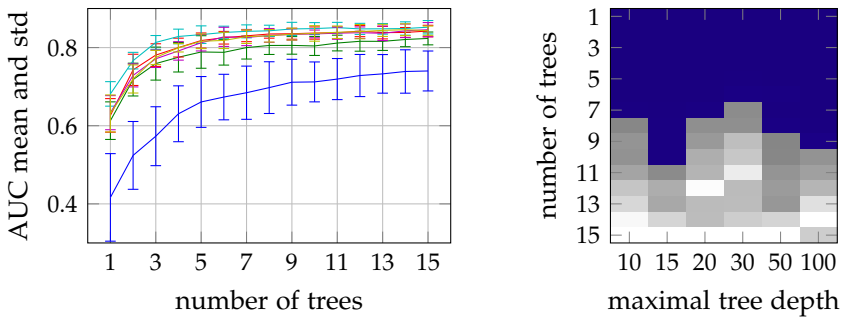
As stated above, in the beginning of Section 3.6, the number of trees in the ensemble required for a good performance of a Hough Forest is quite low, compared to the number of trees usually employed in Random Forest classifiers for standard machine learning tasks. This is demonstrated in Figure 3.8, which reports the performance of a person detector on the three test sequences of the TUD pedestrian, campus and crossing database. It shows that there is a big improvement from taking only one tree to about 10, but from there on the performance is more or less saturated. In fact, beyond 9 or 10 trees the improvement is not statistically significant.

These results are also in accordance with the observations of other researchers. Fanelli *et al.* [50, 51] use 15 trees in their experiments on head detection and pose estimation, and only 7 trees in [52], working on consumer depth camera data, a setting that allows for real-time performance of their framework. Shotton *et al.* [140] only use 3 trees although improvements are visible for up to 6 trees, which is the maximum that was tested. Again, run-time requirements are the main driving force to take such a low number of trees and the evaluations show that taking more trees does not increase the performance enough to justify the higher computational effort.

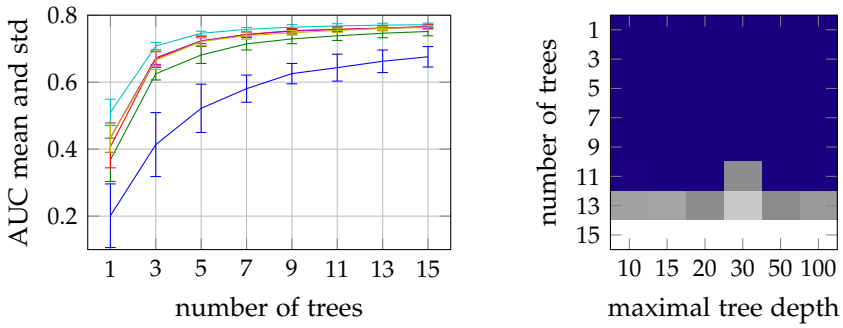
In all further experiments, as in the original Hough Forests [66], 15 trees are used, since it gives the best results and evaluation speed is not the main target here.

3.6.1.5 Minimal Number of Samples per Node

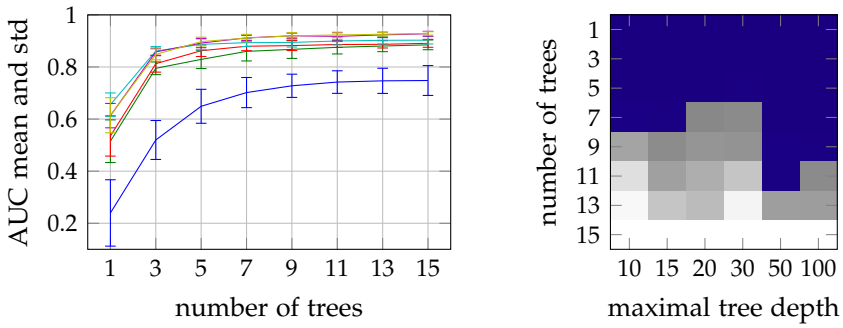
Figure 3.9 shows the effect of setting a lower limit on the number of samples in a node. Generally, it improves the performance of the otherwise best performing setups with *histogram entropy* as node split test evaluation criterion. On the TUD pedestrian and TUD crossing datasets this improvement is also statistically significant. On TUD campus, the



(a) TUD campus



(b) TUD crossing



(c) TUD pedestrian

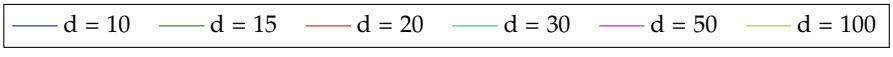
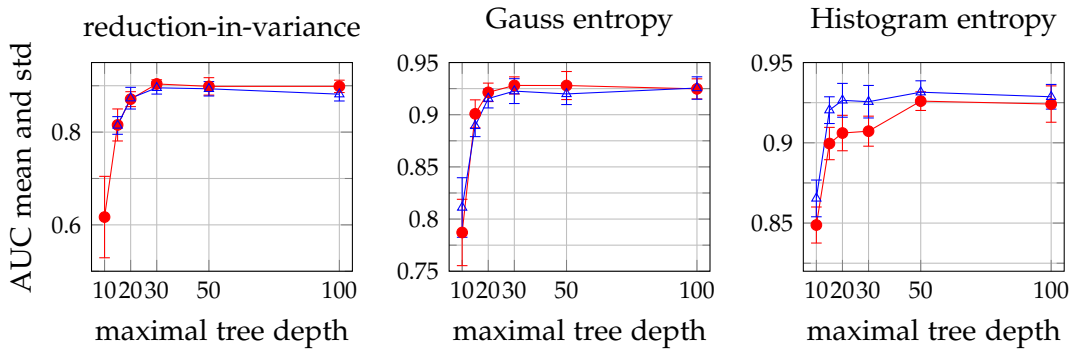
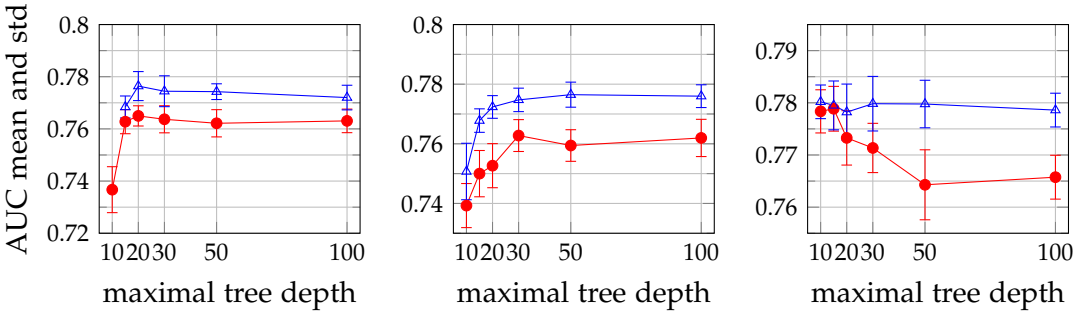


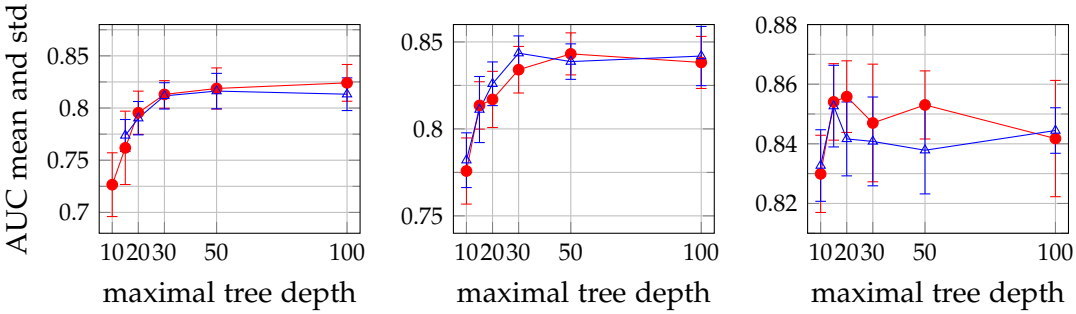
Figure 3.8: **Number of trees:** Left: AUC depending on the number and the maximal depth of the trees. Right: Statistical significance of the results. Each column represents the same maximal depth for different numbers of trees. The brightness indicates the confidence. On blue colored fields the performance of this setup is statistically significant ($P < 5\%$) worse than the best setup. Generally, the performance constantly improves with the number of trees (no overfitting). However, above 9 trees (13 for campus) the improvement is not statistically significant anymore. This trend is independent of the maximal tree depth and is consistent over the three test datasets.



(a) TUD pedestrian.



(b) TUD crossing.



(c) TUD campus.

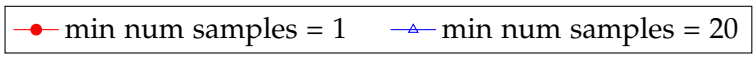


Figure 3.9: Setting a **minimal number of samples per node** generally improves the accuracy of the overall detector. The improvement is significant on the TUD pedestrian and crossing datasets with *histogram entropy* as split test evaluation criterion, even for the best performing setting with a maximal tree depth of 50 ($P = 2.65\%$). On the TUD campus dataset, however, curiously the performance gets slightly reduced.

performance is slightly reduced, however the drop is not statistically significant and might be an artifact of other elements of the detection pipeline on this dataset.

3.6.1.6 Regression Node Evaluation Criteria

The following experiments show the effect of the choice of one of the split test evaluation criteria used to improve the quality of the regression in each node. Figure 3.10 shows mean and standard deviation over 15 runs of the AUC of detectors trained with the standard setup. The three curves show the behavior of the three different variants over different levels of maximal tree depths. As a first result, the *reduction-in-variance* criterion always performs worse than the other two. The *Gauss entropy* criterion reaches the performance of the *histogram entropy*, but only for deeper trees. The *histogram entropy* obviously manages to find better splits earlier on, in higher levels of the trees.

Besides evaluating the performance on a test set, another useful tool to analyze the trees is to look at the distribution of samples over leaf nodes in the various depth levels. This information is visualized in Figure 3.11. With the *histogram entropy* criterion much more nodes reach a stopping criterion in higher levels of the tree. In fact, with the standard setup (database of 32k positive and 64k negative patches and a minimal number of 20 samples in a node to continue splitting), the trees never grow deeper than to level 39, whereas with the other two criteria, at a maximal tree depth of 50 there are still a few nodes that contain over 200 samples. The amount of such nodes and their size can be seen in Figure 3.12, showing a histogram of the mean number of nodes in the trees at a given depth and containing a certain number of samples. For instance, the entry marked with a star in the right plot of Figure 3.12 (a) denotes that there are on average 15.12 nodes in the final tree level that contain between 21 and 47 samples. Besides delivering worse estimates, these large leaf nodes also increase the runtime of the voting process, if the information in the leaf nodes is not post-processed, as shown in Section 3.6.1.7.

3.6.1.7 Mean Shift Post-Processing of Leaf Nodes

The effect of performing meanshift over the samples in each leaf of the trees and only keeping up to 3 modes of the resulting clusters is demonstrated in Figure 3.13. It shows that a quite significant speedup can be achieved for small trees with nodes containing many offset vectors that need to be considered during voting, yet accompanied by a drop in performance. However, for trees with a maximal depth of 20 to 30, which show

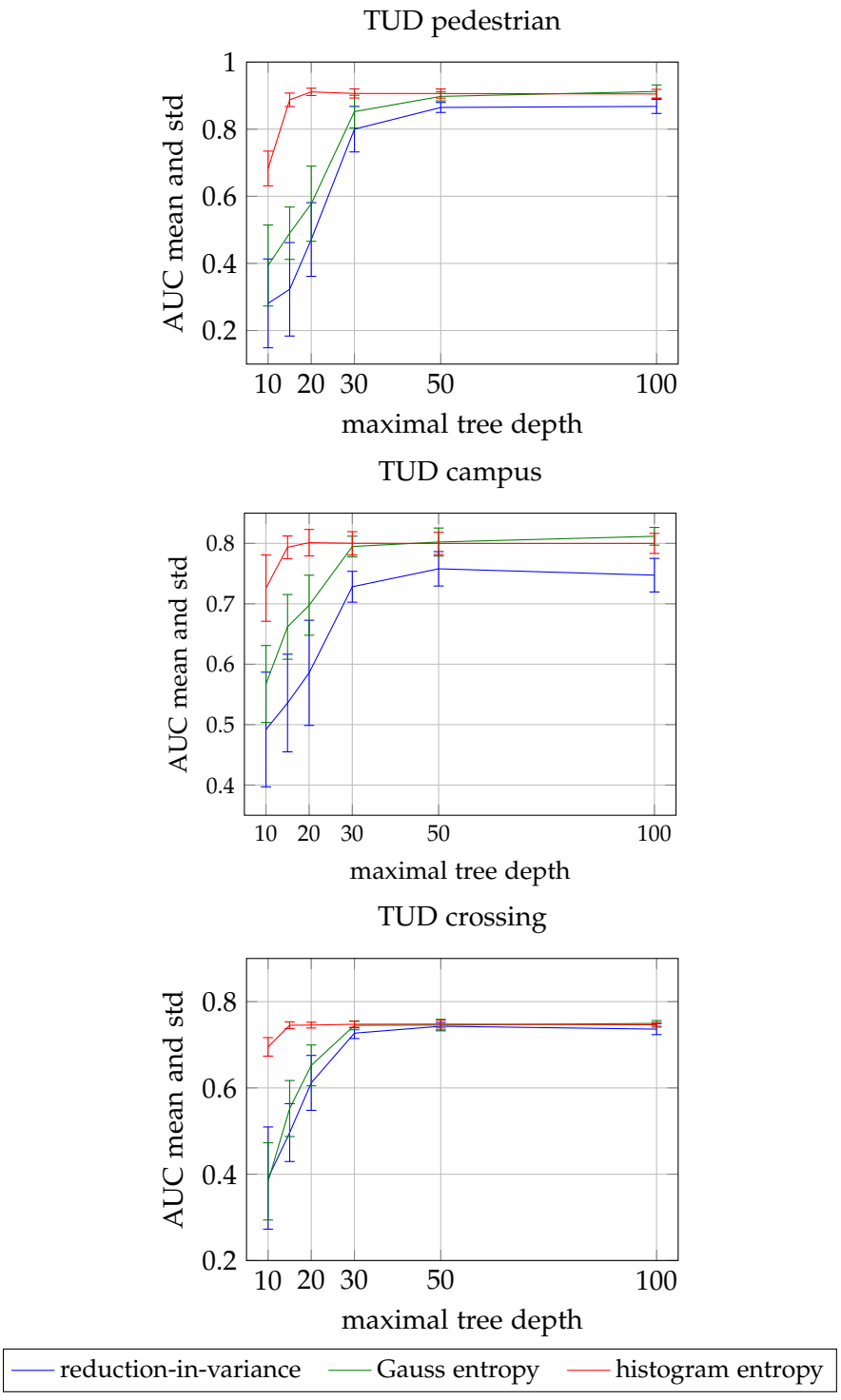


Figure 3.10: Comparison of split test evaluation criteria. *Reduction-in-variance* always performs worse than the other two. Forests trained with the *Gauss entropy* criterion reach the performance of those trained with *histogram entropy*, but need deeper trees to get to this level.

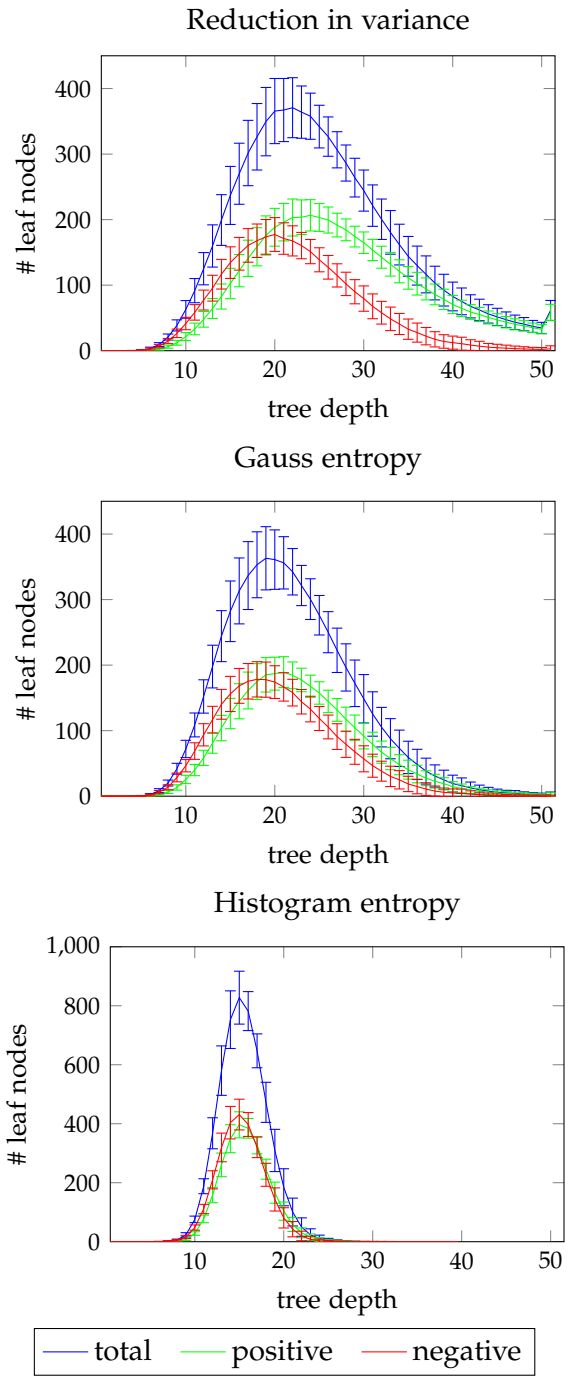
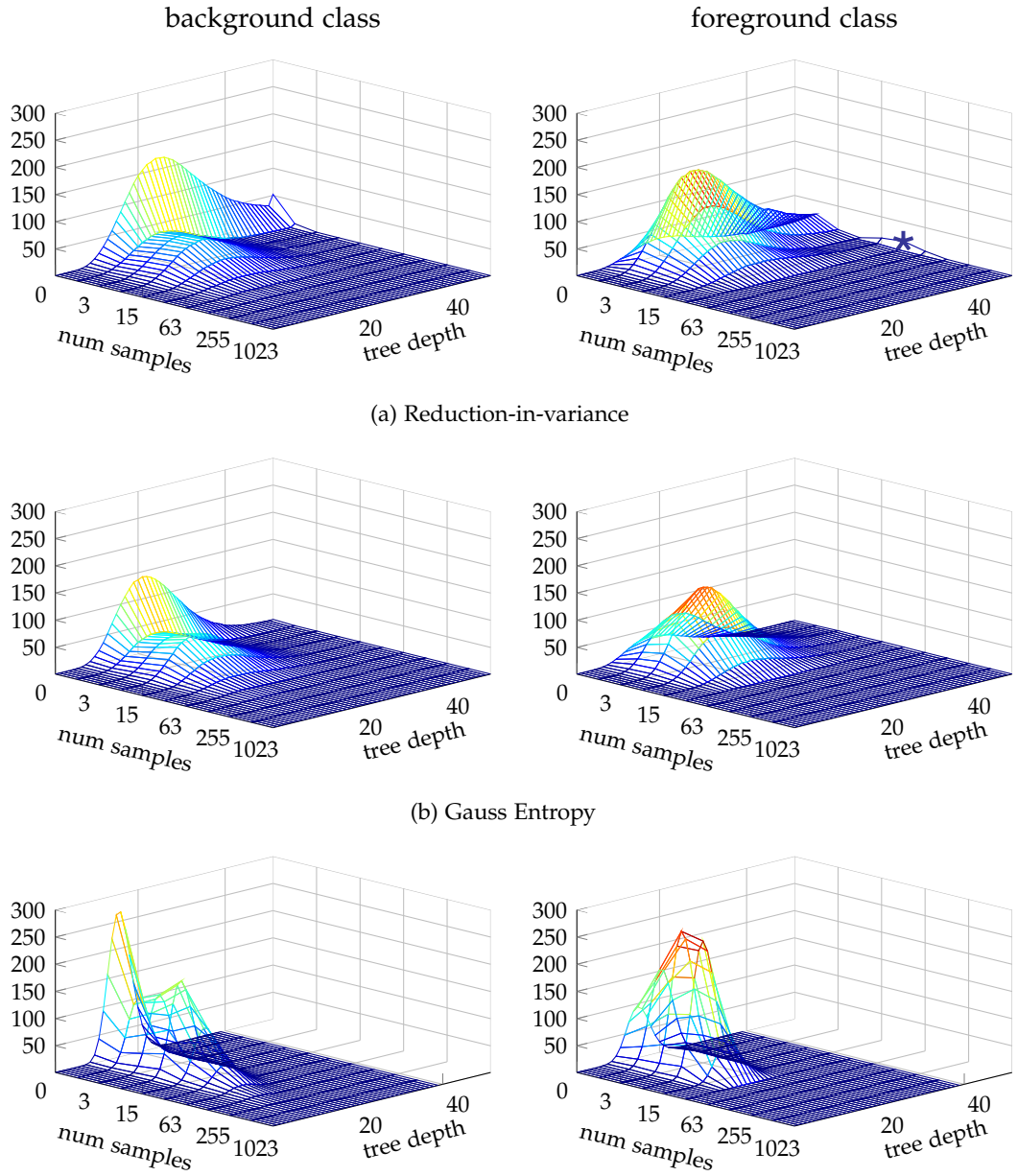


Figure 3.11: Leaf node count of forests trained with different split test evaluation criteria. With the histogram entropy criterion many more leaf nodes can be created higher up in the tree. None of the leafs extends beyond a depth of 39, while with reduction-in-variance trees are still not finished at the maximal depth of 50.



(a) Reduction-in-variance

(b) Gauss Entropy

(c) Histogram Entropy

Figure 3.12: Leaf node count of forests trained with different split test evaluation criteria. The z-value of each entry in the mesh represents the mean number of leaf nodes at the corresponding depth in the trees and the corresponding number of samples in the node. For instance, the position marked with a little star in the right plot of (a) indicates that there are on average 15.12 nodes at the final depth of 50 that contain between 31 and 47 samples. The columns show distributions of negative (left) and positive (right) samples. Main observation: With the *histogram entropy* criterion the splits are much more balanced, such that leaf nodes are created much earlier. Actually, no branch extends above level 39. For further details, see the text in Section 3.6.1.6.

the best overall performance, the gain in test time performance becomes quite small, while the performance is basically unchanged.

These findings are different from those in [139, 140], where meanshift in the leaf nodes always improves the results over taking all offset vectors in the node. This might be due to the chosen split test evaluation criterion, which assumes a unimodal distribution and, in contrast to the original formulation in Hough Forests, ignores samples far from the mean, treating them as outliers. Those outliers are then removed by the meanshift clustering.

The results above, in contrast to the related work, were created with the novel *histogram entropy* criterion, which has a big influence on the distribution of positive samples in the leaf nodes. This observation motivates to also compare the behaviour of the meanshifting on trees that were created using the *reduction-in-variance* or the *Gauss entropy* criterion. Evaluations for these criteria are shown in Figures 3.14 and 3.15.

Generally, the trends are the same with *reduction-in-variance*. However, shallow trees without meanshifting get even slower by a factor of more than 10. With meanshifting a lot more performance is lost up to a maximum tree depth of 50, compared to the trees trained with the *histogram entropy* criterion, for which a depth of 20 is sufficient to reach the maximal performance. This indicates that the *reduction-in-variance* does not manage to split up the positive samples early and creates huge positive leaf nodes if stopped in shallow depths. Additionally, the distributions of samples in those large leaf nodes seem not to be suitable for meanshifting, most probably because of the unimodality assumption, without outlier handling, in the split criterion. With *Gauss entropy* the increase in runtime for shallow trees is by a factor of 5 to 7. Thus, its better than *reduction-in-variance* in finding good splits early, but still a lot worse than *histogram entropy*. However, with increasing depth, for both criteria the speedup for non-meanshifted trees is considerable. At a depth of 30 not-meanshifted forests already reach the speed of those with meanshifting and the performance is still considerably better.

To conclude, in the setup presented here, with mean shifting post-processing there is little to gain in detection time and much to lose in precision. Training deeper trees and keeping all offset vectors is generally the better option. The increase in evaluation time caused by deeper trees is insignificant compared to voting with all offset vectors in a large leaf node stopped at a low tree depth. Mean shifting the offset vectors in those large leaf nodes generally leads to a quite significant drop in performance. Trees trained with *histogram entropy* suffer the least in this setting. For the best performing configurations, however, mean shifting in general does not have a large influence.

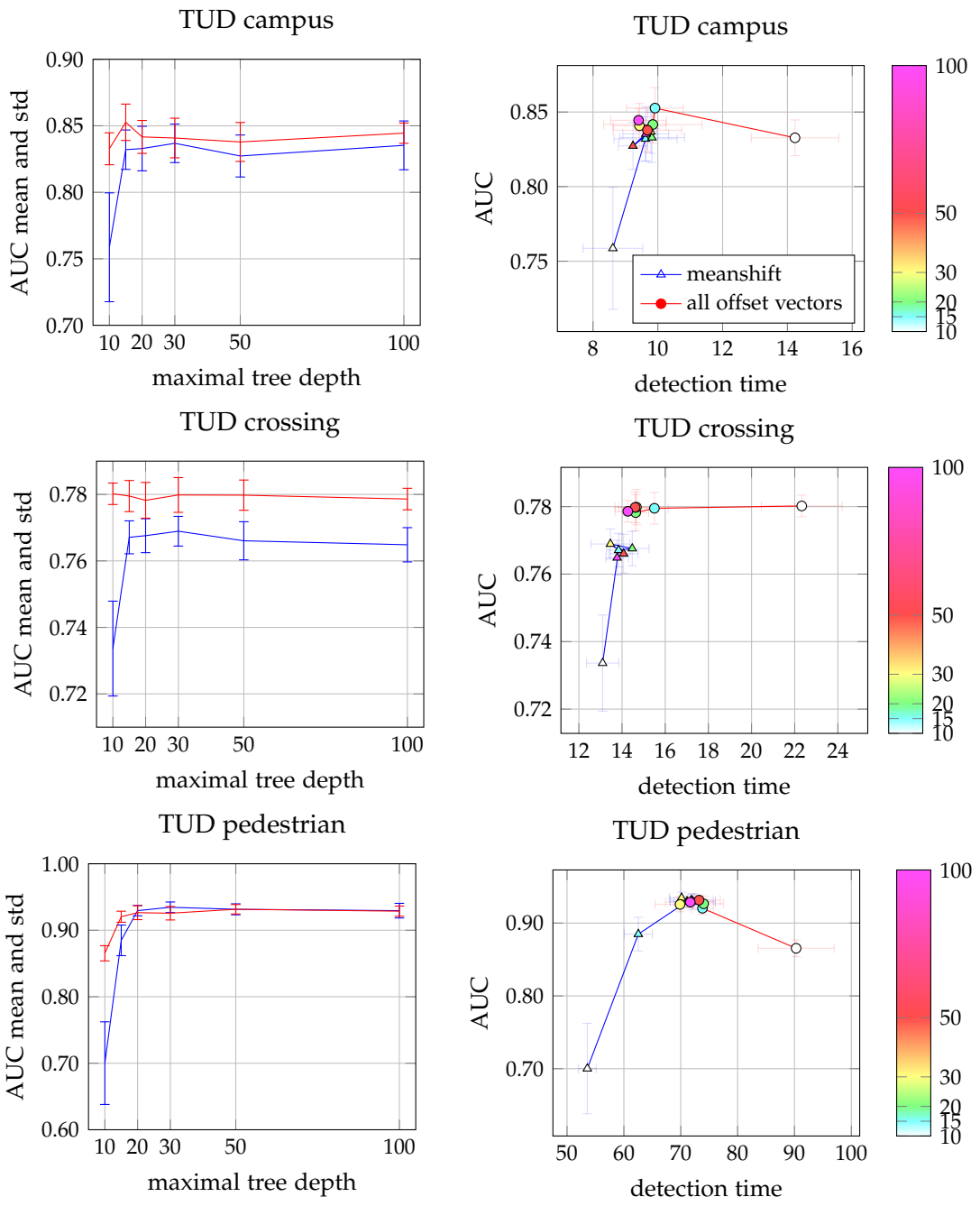


Figure 3.13: Effect of meanshifting the offset vectors in the leaf nodes on the detector performance for forests trained with the *histogram entropy* criterion (see Sec. 3.3). Left column: Area Under the Curve (AUC) depending on the maximal depth of the trees. Right column: AUC vs. detection runtime per image, with different maximal tree depths indicated by the color of the markers. When using the meanshifting, performance is traded off vs. detection accuracy. See text in Section 3.6.1.7 for more details.

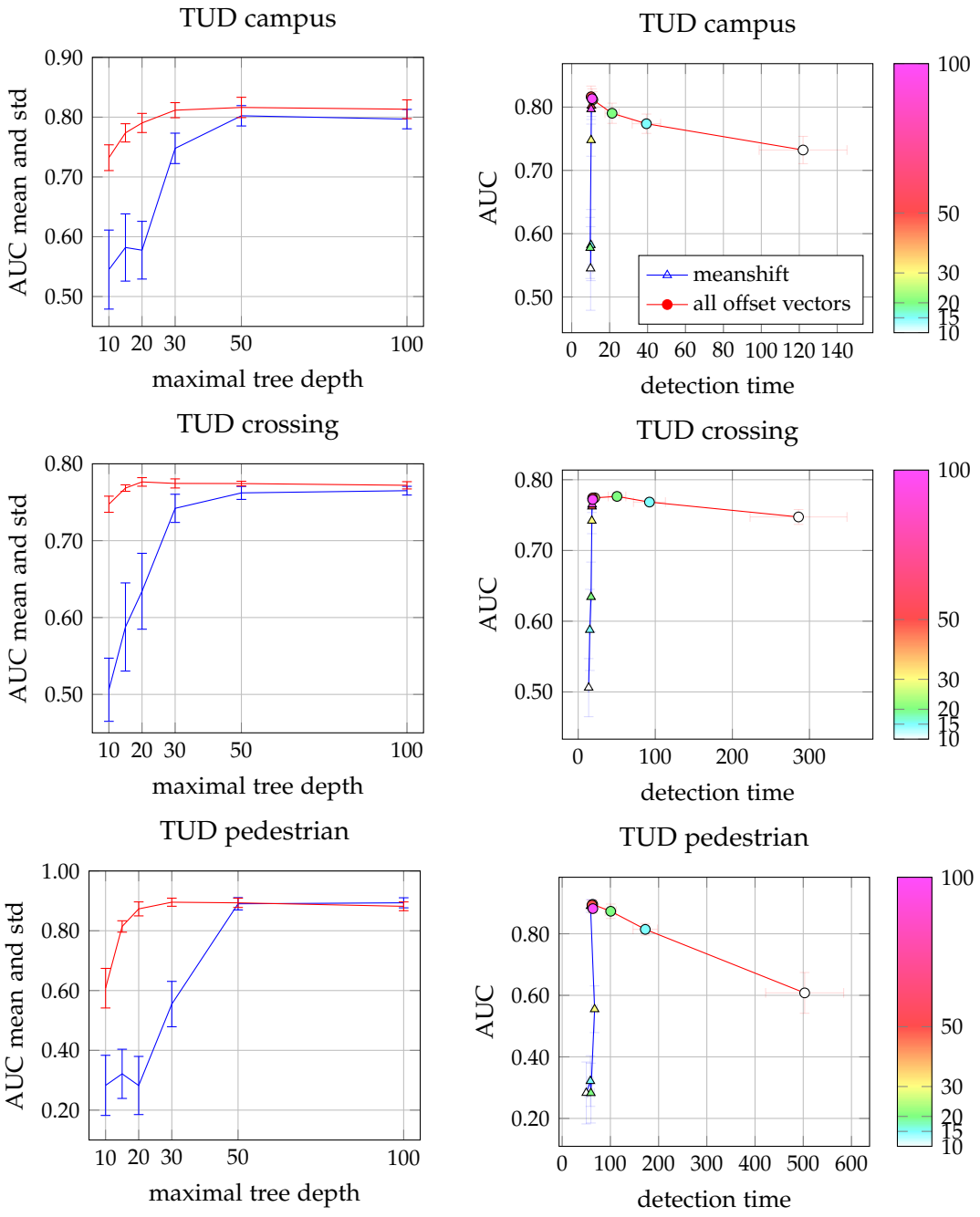


Figure 3.14: Effect of meanshifting the offset vectors in the leaf nodes on the detector performance, depending on the maximal depth of the trees. In contrast to Figure 3.13 the forests were trained with the *reduction-in-variance* split node evaluation criterion. The trends are the same as in Figure 3.13. For more details, see the text (Sec. 3.6.1.7).

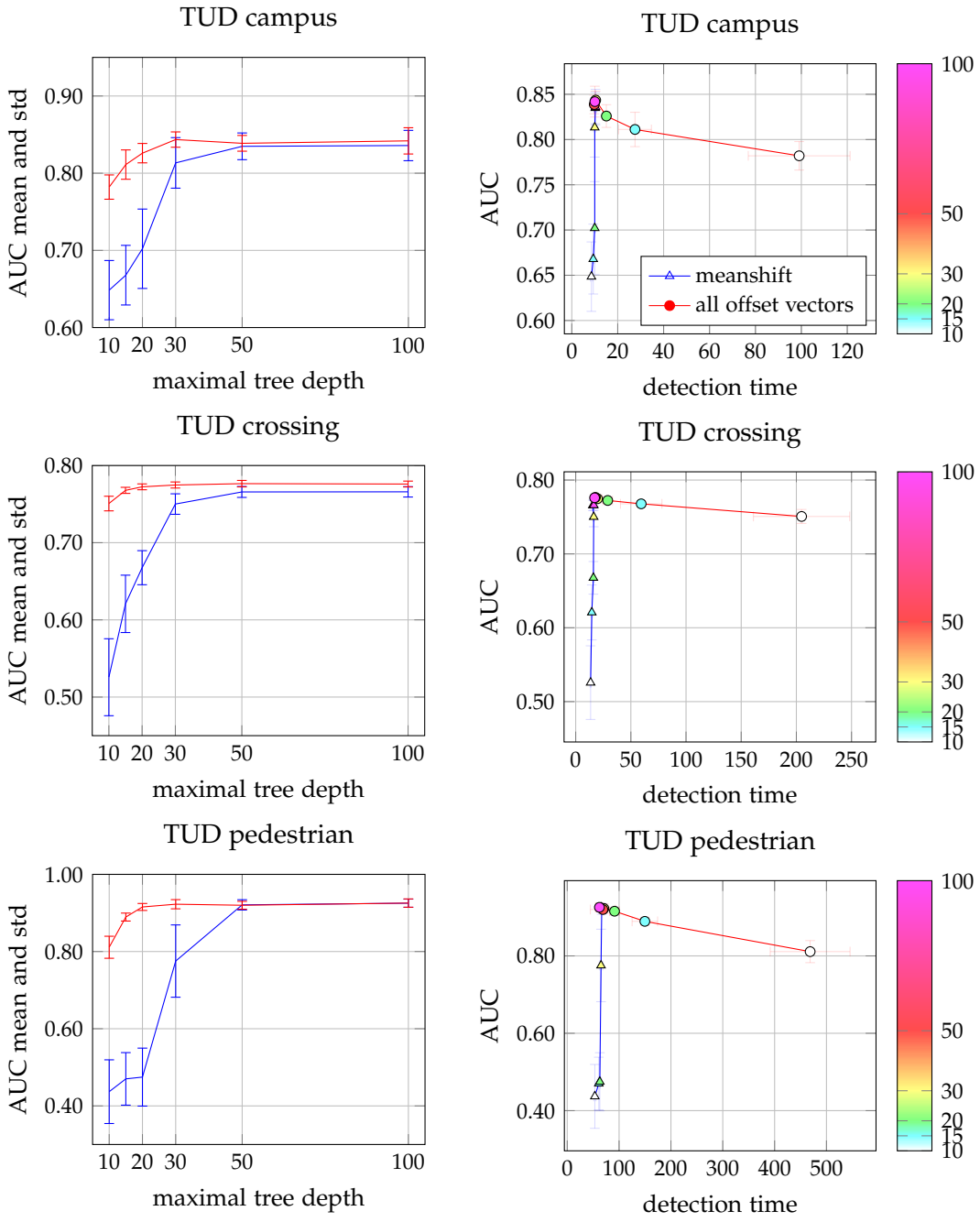


Figure 3.15: Effect of meanshifting the offset vectors in the leaf nodes on the detector performance. In contrast to Figure 3.13 the forests were trained with the *Gauss entropy* split node evaluation criterion. Here, shallow trees without meanshifting get much slower (by a factor of 5 to 7), because the Gauss entropy criterion does not manage to split up the samples early and, thus, creates huge positive leaf nodes. With increasing depth the speedup is considerable. At a depth of 30 not-meanshifted forests already reach the speed of those with meanshifting, at a much higher level of performance.

3.6.1.8 Evaluation Grid

Another way to speed up the detection, which has not been mentioned so far, is to evaluate the forest on a coarser grid, instead of every patch, densely sampled from the image. Figure 3.16 shows the results for different evaluation grids ranging from dense to a stride of 12 in horizontal as well as vertical direction.

The performance continuously goes down the coarser the grid gets. However, up to a grid size of 4×4 the drop is quite moderate (about 1% of mean AUC) and not statistically significant. Thus, if runtime is crucial trading-off 1% performance against a speedup of 16 might be a good deal.

The performance drop is a little more pronounced than reported in related work of [42] and [34], where practically no loss is observed for a grid of 4×4, probably because there the evaluation is over a longer pipeline with more post-processing, such that the precision of the voting itself is not that important. Also, the task is a little different, because no foreground/background estimation is performed, but only the regression of landmark locations in a relatively small neighborhood of the real location. However, other factors may influence the performance of different grid sizes, such as the resolution at which objects are detected, the size of the patches or the smoothing and filtering applied to the input image.

3.6.1.9 Non-Maxima Suppression

Section 3.5 discusses two different versions of non-maxima suppression in the Hough spaces. It was argued that performing the non-maxima suppression in a stack of Hough spaces resized to a common reference scale, instead of a pyramid of scales, is only viable for a small range of scales. Obviously, this raises the question, why the method of fusing Hough spaces into a common reference resolution was chosen in [66]. Thus, in the following experiment the effect of choosing on of the two aforementioned methods is demonstrated.

Figure 3.17 shows results for the TUD sequences. On *TUD-pedestrian* the NMS in the Hough space cube, as proposed in [66], outperforms the NMS in the Hough space pyramid. The main reason for this result is that there is a lot of noisy background in the images. On high resolution scales the fine structures and details result in quite some noise in the Hough space, leading to many false positives. Downsampling to a common, low resolution reference scale simply smooths this evidence out. Additionally, being conservative about what to suppress around a maximum is not a good choice on

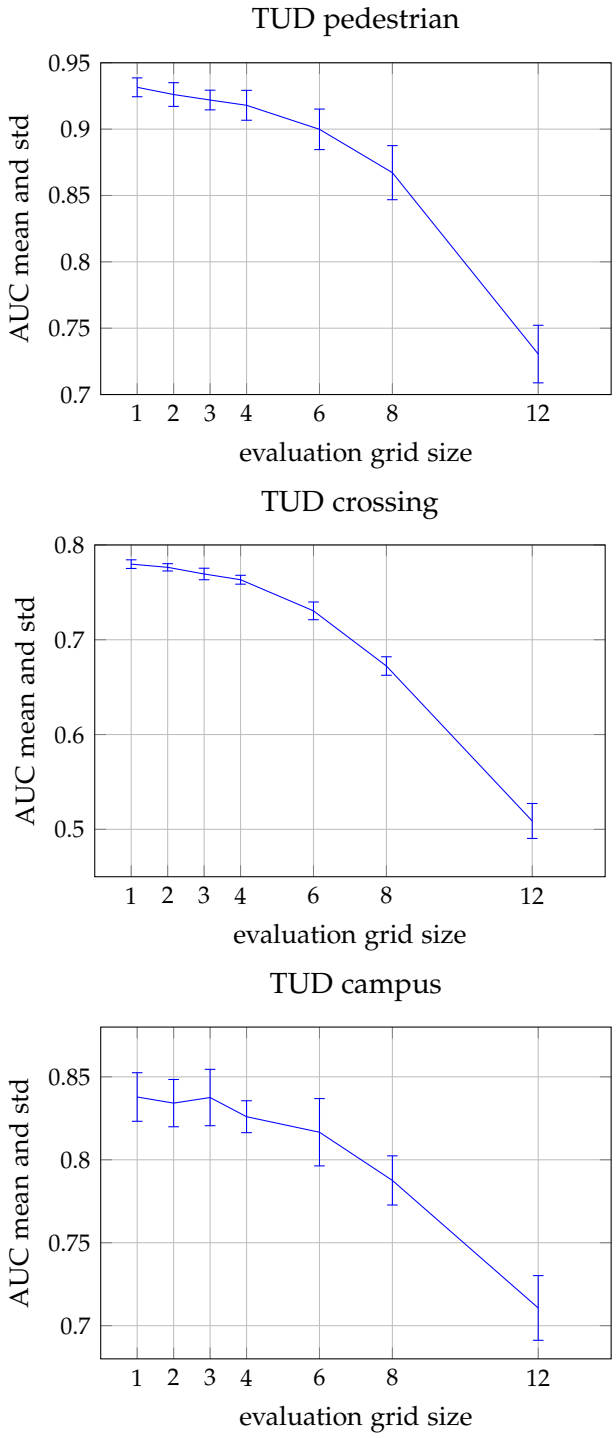


Figure 3.16: Performance of different **evaluation grid** sizes. The forest is evaluated only on every x-th pixel. Thus, the speedup of the evaluation and voting phase is x^2. The performance drops continuously. However, up to a grid of 4×4 the loss is quite moderate (approx. 1%).

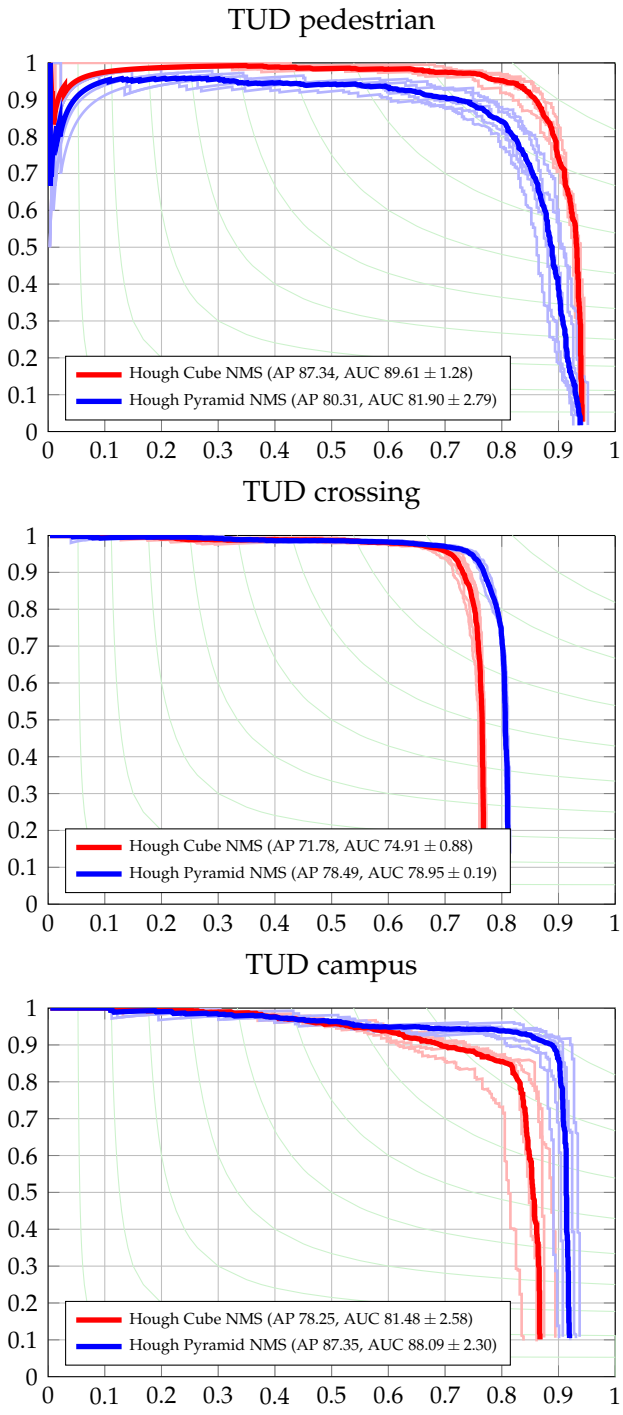


Figure 3.17: Comparison of simple Non-Maxima Supression variants. Hough spaces for different scales are either resized to a common reference scale (Cube NMS) or processed as a pyramid of scales (Pyramid NMS). On the TUD pedestrian sequence, the former works better. For further details see the text.

this database, because none of the depicted pedestrians appear close together. Thus, it is always safe to erase a little more of the neighborhood, which also helps to remove spurious false positives. On the other two sequences the background is much cleaner and the pedestrians appear very close to each other, even overlapping each other heavily. Thus, the non-maxima suppression in the full Hough space pyramid increases the recall by erasing smaller portions without increasing the false positive rate.

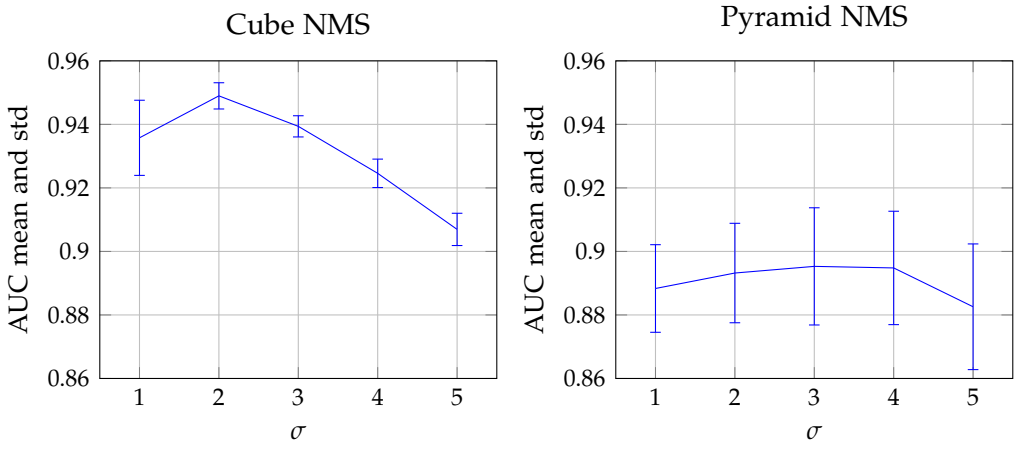
To conclude, the approach to resample the Hough spaces of each individual scale to a common reference resolution can in some cases produce considerably better results. However, this depends on the specifics of the test data. Generally, the more generic method of dealing with the full scale space pyramid of Hough spaces performs better and is more flexible in the number of scales that need to be tested for each individual image.

One parameter that turns out to be quite essential is the width of the Gaussian kernel used to filter the raw Hough maps, introducing the tolerance to slight translations and scale changes of the local patches. Figure 3.18 shows an evaluation of this parameter for the three TUD test sequences and both NMS variants. Interestingly the performance varies on the different test sequences and is also different for the two methods. For Cube NMS the resizing to a common reference frame effectively constitutes a second smoothing process. Thus, the width of the kernel for the final smoothing can be smaller ($\sigma = 2$ delivers the best result). For the NMS in the full pyramid the smoothing must be stronger. Additionally, the smoothing of the resized Hough spaces in the Cube NMS actually results in a different smoothing of the different scales. Higher resolution scales virtually get smoothed with a larger kernel. This, again – although not theoretically underpinned – leads to better results. Also notice, the smoothing was only performed in x and y of the Hough maps, not in scale. The training data was already sampled from slightly different scales. Thus, the evidence already gets spread over several scales during the testing and smoothing in scale does not improve the performance.

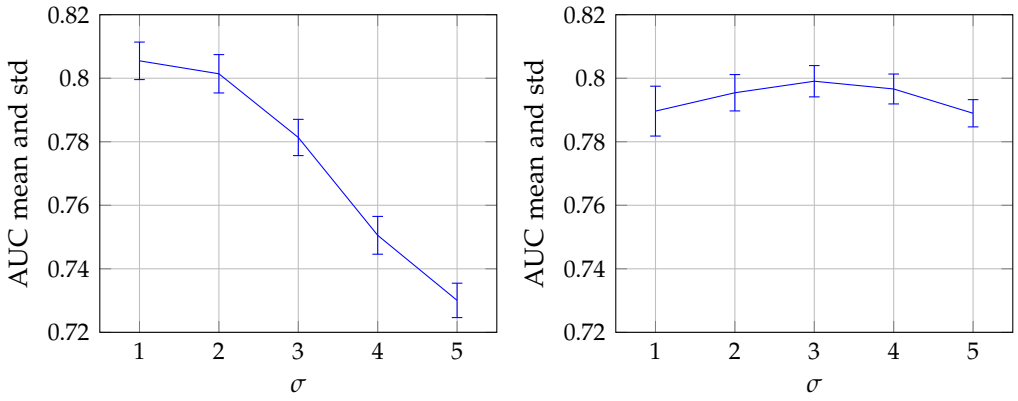
3.6.1.10 Comparison to the State-of-the-Art

To conclude this experimental section, the best setup determined in the evaluations above is compared to the state-of-the-art on the tasks of pedestrian detection on *TUD-pedestrian* and face detection on FDDB. Results on *ETHZ-cars* are presented in Section 4.3.

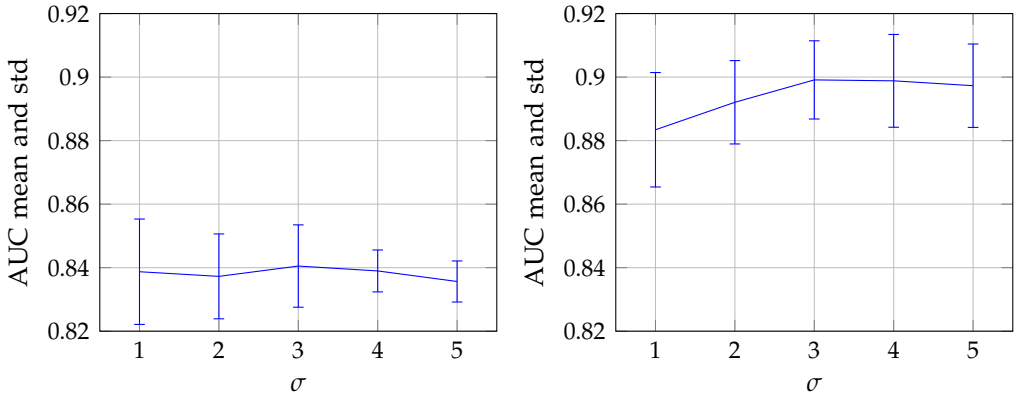
Figure 3.19 shows results on pedestrian detection on the three test sets of the *TUD-pedestrian* dataset. Hough Forests trained as proposed in this chapter are compared to



(a) TUD pedestrian

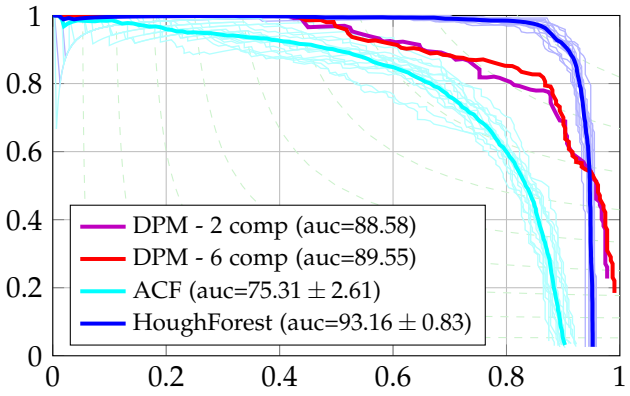


(b) TUD crossing

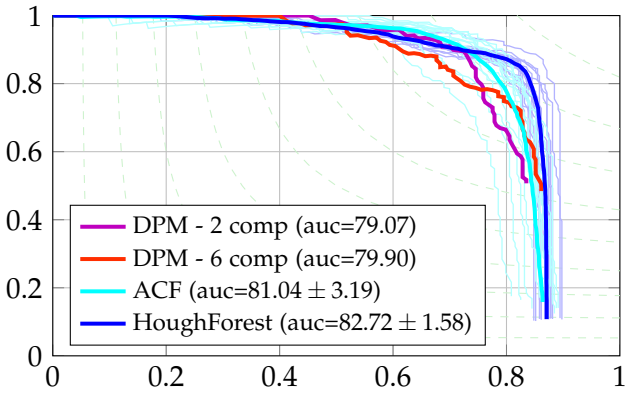


(c) TUD campus

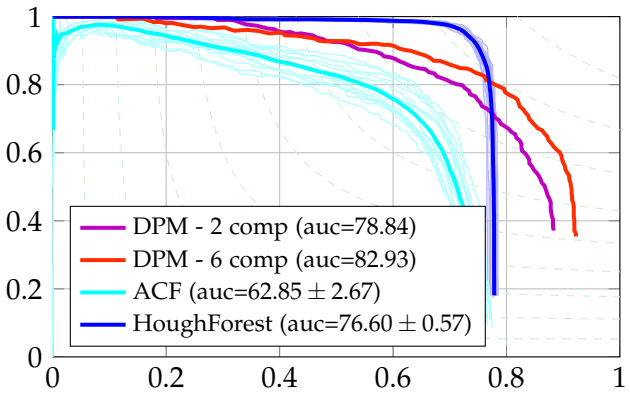
Figure 3.18: Variations of the Gaussian smoothing of the raw Hough maps. Left column: non maxima suppression in a stack/cube of Hough spaces. Right column: NMS in the pyramid of Hough spaces. The stacking of the Hough spaces into a common reference, already performs some smoothing. Thus the subsequent smoothing can have a smaller radius.



(a) TUD pedestrian



(b) TUD campus



(c) TUD crossing

Figure 3.19: Detection performance on the TUD test sets. Hough Forests are compared to the publicly available implementations of Aggreated Channel Features (ACF) [43] and the Deformable Part Model (DPM) [56]. Light thin lines are individual runs, thicker solid lines are mean curves. The Hough Forests are trained as presented in this thesis, with *histogram entropy* and the best determined parameter settings. The plot shows, that the Hough Forests are competitive with the state-of-the-art on this dataset.

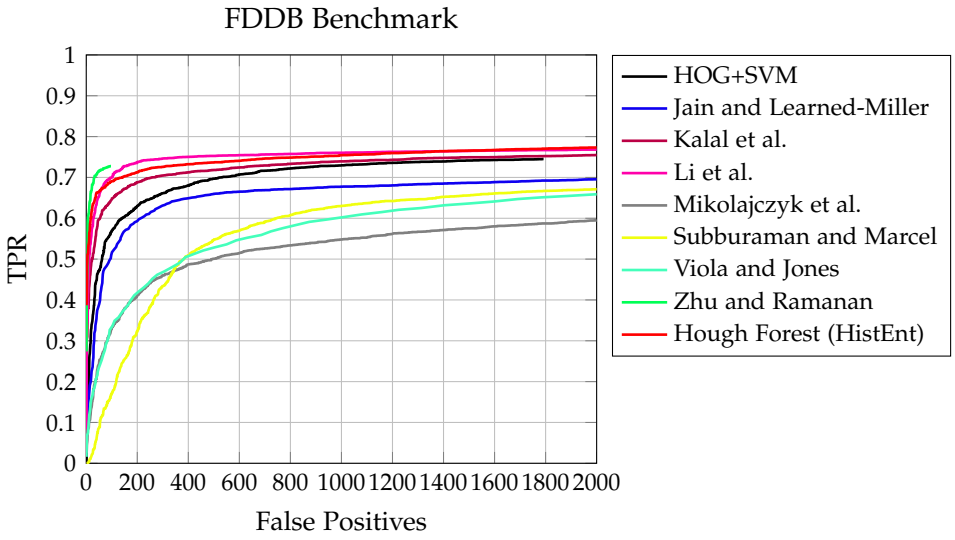


Figure 3.20: Detection performance on FDDB. Compared methods are HOG+SVM [85], Jain and Learned-Miller [76], Kalal *et al.* [81], Li *et al.* [97], Mikolajczyk *et al.* [107], Subburaman and Marcel [144], Viola and Jones [156], Zhu and Ramanan [162]. The Hough Forests are trained as presented in this thesis, with *histogram entropy* and the best determined parameter settings and match the state-of-the-art.

Aggregate Channel Features (ACF) [43][1] and the Deformable Part Model [56][2]. For both methods the current publicly available code was used and trained on the same data as the Hough Forests. The evaluation shows that Hough Forests are competitive with the two state-of-the-art methods on these test sets. For the ACF, which is considerably worse on TUD pedestrian and TUD crossing, it has to be mentioned that the parametrizations are taken from the example shipped with the code for pedestrian detection on the Inria dataset. It is completely tuned towards speed and thus training and testing are really fast (training in about 3 minutes, testing with 4fps for 720×567 images on a single core). Different parametrizations, thus, might make it slower but achieve better detection performance. The Deformable Part Model achieves good recall on this dataset, however, at the cost of accuracy.

The running example in all experiments above was pedestrian detection, as most related approaches were demonstrated for this task. Hough Forests, however, are by no means tuned to this task. To show the generality of the method and also of the determined parameter setup Figure 3.20 shows results for face detection. Hough Forests trained on mostly frontal views ($\pm 35°$) achieve a performance that is comparable to the

[1] `http://vision.ucsd.edu/~pdollar/toolbox/doc/`
[2] `http://www.cs.berkeley.edu/~rbg/latent/`

state-of-the-art. It comes close to the work of Li *et al.* [97], even slightly outperforming it for low values of tolerated false positives. Although being a generic object detection approach, it even gets close to the performance of the work of Zhu and Ramanan [162], which explicitly targets multi-view face detection, with hand designed models and quite a large set of components for individual views from frontal to profile.

3.6.2 Conclusion

This section presented an extensive evaluation of different setups of Hough Forests on benchmark data. The influence of parameters and choices of different variants of the algorithm were discussed and shown in experiments. To conclude, the following summarizes the most important findings.

- In the tested scenarios 10 to 15 trees are enough to achieve good performance.
- Setting a minimal number of samples per leaf node generally results in slightly more robust results.
- Setting the maximal depth of the trees too low severely degrades the performance. However, the optimal tree depth depends of several factors such as the size of the database and the split node evaluation criteria, influencing how balanced the trees are. If no validation set is available, also looking at the distribution of samples over nodes can give a good indication for a good tree depth.
- The results of aggregating the information in the leaf nodes by extracting only modes of the distribution by meanshift were not convincing in the tested settings. The slight speedup during voting does not justify the drop in performance and almost vanishes for the best performing setups with deeper trees. It probably would be worth revisiting the concept if really huge databases are used for which training deep enough trees is not an option. Another possible direction for future investigations also are other ways to assign a weight to the extracted modes.
- Using *histogram entropy* as the regression node evaluation criterion generally delivers the best results over all tested setups. When trees are trained deep enough and the number of samples per leaf node gets low, the assumption that the distribution of the offset vectors is unimodal holds. Thus, the influence of the evaluation criterion is reduced. However, *reduction-in-variance* still performs slightly worse than the two other criteria. Additionally, with *histogram entropy* shallower trees, which are faster to evaluate, are sufficient to reach top performance.

4 Discriminative Hough Forests

The learning procedure of Implicit Shape Models considers each of the many sub-parts extracted from the target objects separately, as discussed in Section 2.5. Occurrences of parts of every type on the training images (visual codebook entries in ISM, leaf nodes of decision trees in Hough Forests) are recorded and models of their spatial distribution with respect to the object center are calculated. Additionally, the probability for each element to stem from the object as opposed to somewhere in the background is estimated. During testing, each patch from the input image is matched to the codebook. The information stored there is used to cast votes for potential object center, again independently from all other patches. This completely independent treatment of individual parts leads to a very flexible model, allowing detection of new objects in configurations that have not been seen in the training data. However, on the other hand, not looking at the whole picture and analyzing which parts appear jointly in a valid object configuration essentially leads to two problems: (a) The flexibility to combine arbitrary local features from all training images to new object configurations might (and does) also result in invalid constellations producing highly confident detections in random noisy backgrounds, and (b) it ignores the relative importance of individual voting element contributions.

To address these problems, in this chapter, we investigate the additive fusion of evidence in the generalized Hough voting process and the constellation of local object parts that jointly vote for an object. We formulate the contribution of each individual voting element to the score of an object hypothesis as a new descriptor for the hypothesis. These descriptors can be used in a discriminative classification framework in order to distinguish correct from wrong constellations. Several kernels are evaluated for the classification. Additionally, a linear classification model allows for using the discriminative weights learned during training directly in the voting process. This approach brings

back the holistic view on the whole object, without completely sacrificing the robustness to occlusions, gained from the local, bottom-up method. The experimental evaluations demonstrate significantly improved results for both approaches.

4.1 Related Work

In [101] a formulation of ISM was presented, with a generative codebook of features extracted on image contours. A max-margin formulation was introduced to learn discriminative weights for each voting element. For the calculation of the activation of each codebook entry the spatial configuration was considered. However, weights were learned only per codebook entry, disregarding the relative position on the object. Additionally, the weights were constrained to be positive.

The Principled Implicit Shape model (PRISM) [91] introduces a weighting scheme for individual words. It was argued that the weighting could be defined by an arbitrary function, which would thus also allow for discriminative training. However, the specific model employed for the voting then defines the weights by a Gaussian mixtures model, which again only casts positive votes. The flexibility in choosing a weighting function is only exploited by globally scaling each mixture model, such that the maximum vote over the space is set to the probability of the corresponding visual word to belong to the foreground.

Furthermore, Hough Forests do not only assign a test patch to different generative codebook entries, but, as we will explain in more detail in Section 4.2.1, estimate a similarity to each individual training patch. In contrast to the two methods mentioned above, we explicitly make use of this fact by calculating a feature vector expressing the contribution of each single training sample to a detection and thus giving the classifier more fine grained control.

In [124] the activation of individual voting vectors by a detection hypothesis was determined. A Support Intersection Kernel was introduced to compare such descriptors. As will been shown in Section 4.2.2, this definition is suitable for finding the most similar training example, but not to discriminate between correct and false detections.

4.2 Method

To facilitate understanding of the following derivations, only the essential parts of the notation used to describe Hough Forests, presented in Section 3.1, is repeated here.

Hough Forests model an object as a collection of a large number of local features (image patches) $\mathcal{P}_i = \{\mathcal{I}_i, \mathbf{d}_i, y_i\}$. $\mathcal{I}_i$ is the appearance description of the patch. The label of the patch, given by y_i, specifies whether a patch was extracted from a positive or negative training image. Each of the positive local patches additionally stores an offset vector $\mathbf{d}_i$ pointing to the center of the object. Over this collection of input training patches Hough Forests build an ensemble of hierarchical codebook structures $\mathcal{F} = \{\mathcal{T}_t\}_{t=1}^{T}$, where $\mathcal{T}_t$ is a randomized decision tree [23], and T is the number of trees. The training of Hough Forests proceeds as described in Section 3.1.3.

During testing, for each test image $\mathcal{I}$, small local patches $\mathcal{P}(\mathbf{y})$ with appearance $\mathcal{I}(\mathbf{y})$ are densely extracted at all locations $\mathbf{y}$ and mapped onto the codebook entries, *i.e.*, the leaf nodes. Each leaf L of tree $\mathcal{T}_t$ stores the set of offset vectors D_L of all the training patches that were routed to that node. In the following, $D_t(\mathbf{y})$ denotes the set of offset vectors stored in the leaf node of tree $\mathcal{T}_t$ that the test patch $\mathcal{P}(\mathbf{y})$ reaches, and $C_t(\mathbf{y})$ is the ratio of positive to negative training samples that reached this leaf node during training (*i.e.*, the probability foreground for the test patch at $\mathbf{y}$, according to tree $\mathcal{T}_t$). Following the generalized Hough transformation procedure, each local patch in the test image casts votes for an object center. These votes are accumulated in the Hough space and local maxima indicate prospective object centers.

4.2.1 Activation Vector

Considering only one input location $\mathbf{y}$ and one decision tree $\mathcal{T}_t$ the score of an object hypothesis at location $\mathbf{x}$ is defined as (compare Equation (6) in [66])

$$S(\mathbf{x}|\mathbf{y};\mathcal{T}_t) = \left[\frac{1}{|D_t(\mathbf{y})|}\sum_{\mathbf{d}\in D_t(\mathbf{y})}\frac{1}{2\pi\sigma^2}\exp\left(-\frac{||(\mathbf{y}-\mathbf{x})-\mathbf{d}||^2}{2\sigma^2}\right)\right]\cdot C_t(\mathbf{y})\ . \tag{4.1}$$

By setting

$$w_t(\mathbf{y}) = \frac{1}{2\pi\sigma^2}\frac{C_t(\mathbf{y})}{|D_t(\mathbf{y})|}\ , \tag{4.2}$$

this can be simplified to

$$S(\mathbf{x}|\mathbf{y};\mathcal{T}_t) = w_t(\mathbf{y})\sum_{\mathbf{d}\in D_t(\mathbf{y})}\exp\left(-\frac{||(\mathbf{y}-\mathbf{x})-\mathbf{d}||^2}{2\sigma^2}\right)\ . \tag{4.3}$$

To further simplify the notation later on, instead of summing over all offset vectors in the set stored with one leaf node, we can just sum over all training patches and check if the according offset vector is contained in that set:

$$S(\mathbf{x}|\mathbf{y};\mathcal{T}_t) = w_t(\mathbf{y}) \sum_{i=1}^{N} [\mathbf{d}_i \in D_t(\mathbf{y})] \exp\left(-\frac{||(\mathbf{y}-\mathbf{x})-\mathbf{d}_i||^2}{2\sigma^2}\right) \quad , \tag{4.4}$$

where $[\cdot]$ is the Iverson bracket (*i.e.*, the expression is 1 if vector $\mathbf{d}_i$ is in the set of vectors $D_t(\mathbf{y})$ and 0 otherwise).

Integrating the information of all decision trees $\{\mathcal{T}_t\}_{t=1}^{T}$ and of all patch locations $\mathbf{y}$ in the test image gives the total score for an object hypothesis at location $\mathbf{x}$:

$$S(\mathbf{x}) = \sum_{\mathbf{y}} \frac{1}{T} \sum_{t=1}^{T} S(\mathbf{x}|\mathbf{y};\mathcal{T}_t) \quad . \tag{4.5}$$

By combining Equations (4.4) and (4.5) and rearranging the sums, the contribution of a single offset vector $\mathbf{d}_i$ of patch $\mathcal{P}_i$ to this total score can be determined as

$$a_i(\mathbf{x}) = \frac{1}{T} \sum_{\mathbf{y}} \sum_{t=1}^{T} [\mathbf{d}_i \in D_t(\mathbf{y})]\, w_t(\mathbf{y}) \exp\left(-\frac{||(\mathbf{y}-\mathbf{x})-\mathbf{d}_i||^2}{2\sigma^2}\right) \quad . \tag{4.6}$$

We will refer to $a_i(\mathbf{x})$ as *activation* of the offset vector $\mathbf{d}_i$ for the hypothesis $\mathbf{x}$. The total *activation vector* recording the activations of all offset vectors for hypothesis $\mathbf{x}$ is then given by $\mathbf{A}(\mathbf{x}) = [a_1(\mathbf{x}), \ldots, a_N(\mathbf{x})]^T$. We additionally introduce the short notations $\mathbf{A}_j = \mathbf{A}(\mathbf{x}_j)$ and $a_{j,i} = a_i(\mathbf{x}_j)$.

Thus, the score $S(\mathbf{x})$ of a hypothesis $\mathbf{x}$ can be expressed in terms of the activations of individual training patch offset vectors:

$$S(\mathbf{x}) = \sum_{i=1}^{N} a_i(\mathbf{x}) \quad . \tag{4.7}$$

This formulation, in which each of the offset vectors from training is treated individually, also allows for a different interpretation of Hough Forests in the context of ISMs. As stated above, in an ISM local features of the test image are mapped to codebook entries; each of them either to only one codebook entry or via a weighting to a sparse set of entries (soft assignment, *e.g.*, [101]). Along with each codebook entry (generative prototype) information of all associated training samples is summarized and then used

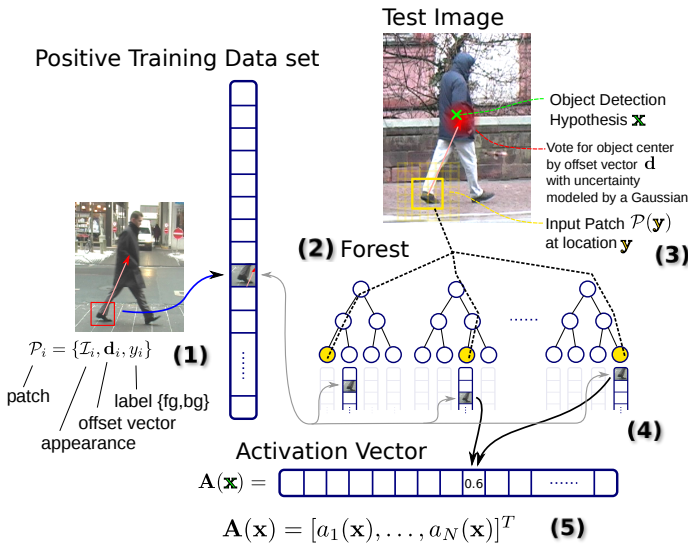


Figure 4.1: Formation of an *activation vector*. (1) Patches are cropped from the training images. (2) A Hough Forest is trained. Each patch from the training set ends up in one leaf node of each tree. (3) Each patch of the input image is passed down all trees. All training patches stored with that leaf node are used to cast votes for the object centers weighted by the probability of being foreground. (4) The total amount of weight one voting element (offset vector) contributes to a detection hypothesis (the total activation) can be calculated by summing over all input patches and trees in the forest, according to Equation (4.8). (5) Stacking the activations of all elements gives the *activation vector* for hypothesis $\mathbf{x}$. This vector can be seen as a descriptor of the detection hypothesis and be used in further analysis.

for the voting. In a Hough Forest, a test patch is assigned a codebook entry (leaf node) by each tree. This way the RF can be seen as uniform soft assignment to a fixed number of entries from individual sub-codebooks (one per tree). However, the leaf nodes of all trees are just different partitions of the same set of all training patches. Thus, by counting in how many of the trees the test patch ends up in the same leaf node as a training patch, the Random Forest gives an estimate of the similarity to each individual training patch, not only to the collection of training patches forming a codebook entry. This is explicitly encoded by the activation of one offset vector by one test input patch at $\mathbf{y}$ relative to the object hypothesis $\mathbf{x}$:

$$a_i(\mathbf{x}, \mathbf{y}) = \frac{1}{T} \sum_{t=1}^{T} [\mathbf{d}_i \in D_t(\mathbf{y})] \, w_t(\mathbf{y}) \exp\left(-\frac{||(\mathbf{y} - \mathbf{x}) - \mathbf{d}_i||^2}{2\sigma^2} \right) . \tag{4.8}$$

The whole process of the formation of an activation vector is illustrated in Figure 4.1.

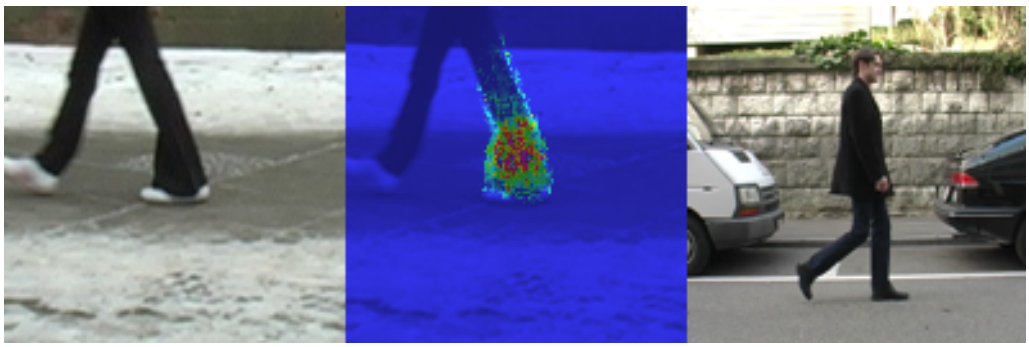


Figure 4.2: Matching the activations (middle) of a low scoring false positive detection (left) using the support intersection kernel [124]. Since the small area captured by the back-projection is a perfect match, the distance to the training sample on the right is smaller than many distances between true positive detections and their respective closest training sample.

4.2.2 Classifying Valid Object Constellations

The activation vector can be seen as a descriptor of an object hypothesis. We can collect descriptors of positive and negative examples in order to train classifiers to discriminate between correct and incorrect detections.

In [124] a *support intersection kernel* was introduced. It induces a distance measure by which the closest training sample for a given detection can be identified. This information can, for instance, be used to transfer metadata such as the object's pose from the annotated training sample to the detection. However, this measure is not suitable to discriminate between correct and incorrect constellations, since it is specifically designed to also match detections with only a few active voting elements in order to handle strong occlusions. Thus, also for low scoring false positive detections often perfect matches are found, as illustrated in Figure 4.2.

We therefore propose to use a histogram intersection kernel. Given two activations $\mathbf{A}_1$ and $\mathbf{A}_2$, it is defined as

$$K(\mathbf{A}_1, \mathbf{A}_2) = \frac{\sum_i \min\{a_{1,i}, a_{2,i}\}}{\max\left\{\sum_i a_{1,i}, \sum_i a_{2,i}\right\}} \quad . \tag{4.9}$$

In the literature (*e.g.*, [153]) intersection kernels are often defined without the normalizing sum in the denominator of Equation (4.9). Thus, either the range of the values in the histogram vectors has to be limited intrinsically to reasonable values, or the individual histograms have to be normalized (usually L_1 normalization). Otherwise no meaningful results can be obtained.

In the case at hand, normalizing each activation vector individually is avoided. In fact, normalization would increase the noise in low scoring detection hypotheses, with very low activations of only a few offset vectors, and thus lead to similar problems as with the support intersection kernel. On the other hand, without any normalization no activation vector would ever be considered to be similar to a low scoring true positive example. Normalization with the sum over the larger of the two activation vectors returns similarity values in the well defined range of $[0, 1]$, a similarity value of close to 1 also for similar low scoring true positives, and low similarity for activations of the same distribution but very different magnitude.

In the experiments training of Support Vector Machines on histogram intersection kernels as well as linear SVMs to discriminate between valid and invalid object constellations is explored. We collect a set $\{(A_j, l_j)\}_{j=1}^{M}$ of activation vectors $\mathbf{A}_j = \mathbf{A}(\mathbf{x}_j)$ from locations $\mathbf{x}_j$ in the set of training images and assign them a label $l_j = 1$ if there is an object in the ground truth at $\mathbf{x}_j$ and $l_j = -1$ for locations $\mathbf{x_j}$ in the background or on negative training images. The output of the classifier learned on this training set then defines the new score of the detection hypothesis.

4.2.3 Using Discriminative Weights in the Voting Process

In the last sections we have shown how to form activation vectors and learn classifiers in order to improve the total score of a hypothesis. To get the activation vector $\mathbf{A}(\mathbf{x})$ for one object hypothesis location $\mathbf{x}$ we need to sum over all patch locations $\mathbf{y}$. Repeating this for each location $\mathbf{x}$ in the input image would be very inefficient. Therefore, the Hough Forests algorithm, as presented in [66], calculates the score for all hypotheses jointly in one run over all input patches. Each patch is passed down the trees and for each offset vector in the resulting leaf nodes its weight is added to exactly the relative hypothesis it points to (*i.e.*, $\mathbf{y} - \mathbf{d}$). The full Hough map is calculated by filtering the result once with a Gaussian.

Since each offset vector contributes individually to the total score, instead of calculating an unweighted sum discriminative weights can be introduced:

$$\hat{S}(\mathbf{x}) = \sum_{i=1}^{N} w_i a_i(\mathbf{x}) = \mathbf{w}^T \mathbf{A}(\mathbf{x}) \quad . \tag{4.10}$$

Learning these weights from training data in a max-margin setup in order to optimize the final detection scores leads exactly to the linear SVM formulation as proposed

in Section 4.2.2. However, as confirmed by experiments, to obtain good classification performance, we cannot train directly on A_j, but need to normalize each dimension of the data to zero mean and unit variance. Thus, the weight vector resulting from the SVM training (which is denoted as $\tilde{\mathbf{w}}$) cannot directly be used in Equation (4.10) but leads to a score defined as

$$S_{\text{SVM}}(\mathbf{x}) = \sum_{i=1}^{N} \tilde{w}_i \frac{a_i(\mathbf{x}) - m_i}{s_i} \ , \tag{4.11}$$

where m_i and s_i are the mean and standard deviation of the activation of each vector over the training data. In order to use the discriminative weights learned in the last section directly in the voting process, such that the Hough map reflects the final score, we need to associate a weight with each individual offset vector $\mathbf{d}_i$. Looking at Equation (4.11) we see that

$$S_{\text{SVM}}(\mathbf{x}) = \sum_{i=1}^{N} \tilde{w}_i \frac{a_i(\mathbf{x}) - m_i}{s_i} = \sum_{i=1}^{N} \frac{\tilde{w}_i}{s_i} a_i(\mathbf{x}) - \sum_{i=1}^{N} \frac{\tilde{w}_i}{s_i} m_i \ . \tag{4.12}$$

The last term is a constant offset that can be added after the voting. Thus, we can set $w_i = \frac{\tilde{w}_i}{s_i}$ to receive the discriminative weights as defined in Equation (4.10). Using the definition of $a_i(\mathbf{x})$ from Equation (4.6) and rearranging the terms, we see that

$$\hat{S}(\mathbf{x}) = \sum_{i=1}^{N} w_i a_i(\mathbf{x}) = \sum_{\mathbf{y}} \sum_{t=1}^{T} \sum_{\mathbf{d}_i \in D_t(\mathbf{y})} w_i w_t(\mathbf{y}) \exp\left(-\frac{||(\mathbf{y} - \mathbf{x}) - \mathbf{d}_i||^2}{2\sigma^2} \right) \ . \tag{4.13}$$

Thus, the final score map can be obtained during the voting process by adding a weight of $w_i w_t(\mathbf{y})$ to the hypothesis at $\mathbf{x} = \mathbf{y} - \mathbf{d}_i$ for each activation of $\mathbf{d}_i$ by a patch at $\mathbf{y}$ and Gaussian filtering of the result.

4.3 Experiments

The performance of the proposed method is demonstrated on two different object detection data sets, namely *TUD-pedestrian* [6] and *ETHZ-cars* [92], introduced in Section 3.6.1.1. Both contain quite challenging images, where objects are captured under different poses and lighting conditions and also show some occlusions. The goal of the experiments is to show the relative improvement of the approach compared to the standard Hough Forest [66] without the proposed discriminative weighting of the voting vectors.

Training

In all experiments the basis is an implementation of Hough Forests as presented in Chapter 3, using the same patch representation, split tests and parametrization as in the original Hough Forest [66]: 15 trees, with a maximal depth of 15, *reduction-in-variance* offset vector regression criterion, minimal number of samples per node = 20, splitting is stopped if nodes are purely negative, whereas nodes that only contain positive patches are split on the offset vector regression criterion until the maximal depth, or the minimal number of samples per node is reached.

In order to collect activation vectors of positive and negative examples, standard Hough Forest with uniform weights are evaluated on the training images. The evaluation of the resulting detection hypotheses gives the label of the activation vector. For the evaluation of the performance of the Hough Forest we use the PASCAL overlap criterion (see Section 3.6.1.9) with a threshold of 0.5. For the labeling of the activation vectors however, not the most confident detection with acceptable overlap is considered as positive, but the one with biggest overlap, since that is the one that should get the maximum confidence in the end.

As in [66] and described in Section 3.5, for handling of different scales, the feature extraction and voting is run on a series of scaled versions of the input and the resulting Hough maps are stacked into one 3D Hough space on which local maxima detection and non-maxima suppression is performed. Due to the interpolation, the resulting detection might lie between two of the tested scales. Thus, to collect the activation vectors the input image is resized to the respective scale and the voting procedure is repeated, now only recording activations for the target location.

As with all max-margin classification systems, proper bootstrapping is crucial to obtain good test performance. Thus, several rounds of bootstrapping are run on a validation set during learning of the SVMs. Additionally, a preliminary version of the discriminative weights are used in a second run of the voting process, after which again activation vectors are collected and several more rounds of bootstrapping are performed.

Bounding Box Estimation from Back-projection

The standard Hough Forest [66] delivers an estimate for the position and scale of objects in a test image. The aspect ratio of the reported detections is the mean aspect ratio of the bounding boxes from the training. If the aspect ratio varies heavily (such as for front-/back- vs. side-views of cars) this parameter has to be estimated as well. The

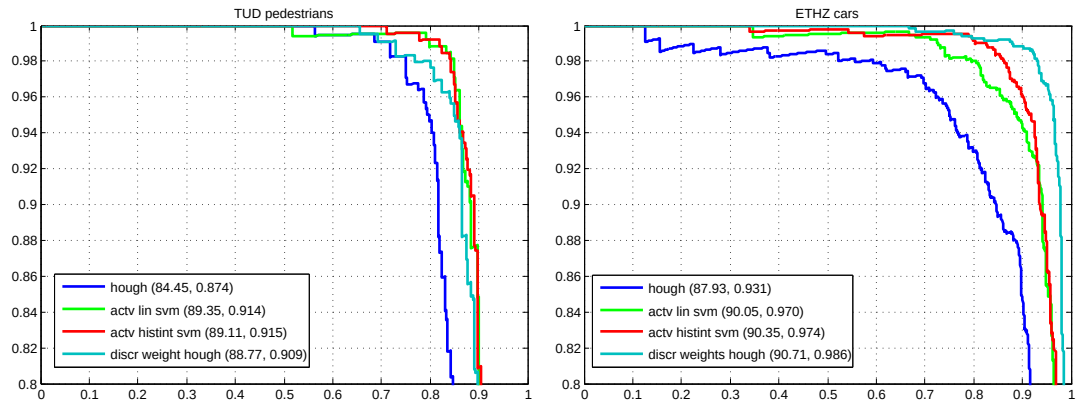


Figure 4.3: Precision/Recall curves for *TUD pedestrians* and *ETHZ cars* dataset. The curves represent the performance of standard Hough Forests [66] (*hough*), linear SVM on the activation vector (*actv lin svm*), histogram intersection kernel SVM on the activation vector (*actv histint svm*), and Hough voting with learned discriminative weights (*discr weights hough*).

activation vectors that are being collected already deliver the information from which a better estimate of the bounding box can be derived. Similarly to [124], the bounding box of the detection is estimated by determining where votes for the object came from (in [124] this is called the *back-projection*). To increase the robustness to noisy votes from the background, only pixels that contribute more than a minimum of 5% of the total score to the detection are considered to be inside the object. The reported bounding box then tightly captures everything above this threshold.

Results

Figure 4.3 shows the resulting Precision/Recall curves for all methods for the task of pedestrian detection and car detection on the *TUD-pedestrian* and *ETHZ-cars* datasets. All three proposed methods show significant improvements over the baseline. The post-processing of detections with a histogram intersection kernel SVM on the activation vector achieves an Area Under the Curve (AUC) of 91.5% on *TUD-pedestrian* and 97.4% on *ETHZ-cars*, compared to 87.4% and 93.1% for the baseline. The linear classifier performs on par with the histogram intersection kernel on *TUD-pedestrian* (91.4%) and only slightly worse on *ETHZ-cars* (97.0%).

The Hough voting with discriminative weights also shows clear improvements over the baseline (90.9% and 98.6%). On the *ETHZ-cars* it even outperforms the two post-processing variants. The difference in the performance on the two datasets relative to

Figure 4.4: Hough maps for example test images (left) from the TUD pedestrian dataset, with discriminative weights (middle) and standard uniform weights (right). Note how in the case of discriminative voting correct detection peaks are clearly pronounced while background noise is significantly reduced.

the other methods may be attributed to the more complex background on the *TUD-pedestrian* dataset. This could probably still be improved by further rounds of bootstrapping. Notice, that the post-processing methods work on the output of the non-maxima suppression after standard Hough voting and thus, can only increase the precision but not the recall.

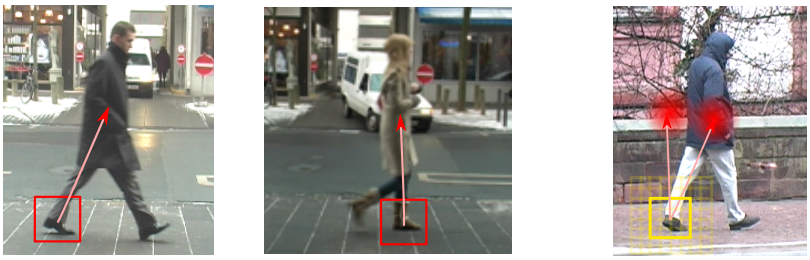
Figure 4.4 additionally visualizes some examples of Hough maps obtained from voting with the learned discriminative weights and compares them to the results of the baseline.

4.4 Conclusion

In this chapter it was shown how to determine how much each voting element in a Hough Forest contributes to a detection. This information is collected in an *activation vector* of the detection. It can be seen as a description of the detection hypothesis. It was shown how to collect sets of activation vectors and train classifiers to discriminate between correct and incorrect hypotheses, based on this descriptor. Additionally, the weights for each voting element, learned for a linear classification model, can be incorporated directly into the generalized Hough voting process. Thus, improved detection hypotheses can be calculated at the same expense as in the standard Hough Forest framework, without the need to extract and re-score activation vectors. The experiments on two different object classes, namely pedestrians and cars, show significant improvements over the baseline. Visual inspection of the voting maps created with discriminatively learned voting weights shows much cleaner backgrounds and clearly sharpened and pronounced peaks for correct object locations. This is also reflected in the detection scores.

5 Implicit Shape Model Random Field

The goal of this chapter is to provide a framework for object detection that explicitly addresses the detection of overlapping objects. Implicit Shape Models (ISMs) and Hough Forests particularly lend themselves for detailed reasoning about occlusions, because the image is processed in a bottom-up manner and evidence for objects is gathered on a per pixel level. Analyzing the inference process of ISMs, the central observation is that every local element that looks like a part of an object can vote for multiple hypotheses of where the center of the corresponding object could be. However, in a specific image each local feature can only be part of one particular object and, thus, only the votes agreeing on this one true can be correct. Thus, this chapter presents a novel method to perform inference on ISMs that specifically aims at resolving those inconsistencies and, thus, arrives at better detection results. For this purpose, a probabilistic approach in a general random field setting is developed, which effectively detects object instances and additionally identifies all local patches contributing to the different instances. A sparse graph structure and a semantic label space are defined, specifically tuned to the task of localizing objects. The design of the graph structure then allows for defining a novel inference process that efficiently returns a good local minimum of the associated energy minimization problem. A key benefit of the method is that it removes the need to set a fixed range for local neighborhood suppression, as necessary for instance in related non-maxima suppression approaches. The inference process implicitly is capable to separate even strongly overlapping object instances. Experimental evaluation compares the method to state-of-the-art in this field on challenging sequences showing competitive and improved results.



(a) Training images (b) Test image

Figure 5.1: Problem Statement: During training patches are extracted from the positive images. For each one, additionally, its relative offset to the center of the object is stored. From those patches a codebook of visual features is created by grouping similar looking ones, coming from similar places on the object. At run-time patches are densely extracted from the test image and compared to the codebook entries. The yellow patch matches a visual word that contains both of the red patches from the left. Thus, both offset vectors are used to vote for potential object centers. However, only one of the votes can be correct, since the central pixel of the yellow patch can only lie on one object. The other vote is wrong and creates additional noise in the Hough space.

5.1 A Random Field for Object Detection

As starting point, we assume that we are given a codebook consisting of several visual words, and that we can assign local features of a test image (*e.g.*, a dense set of patches) to the individual visual words. This functionality is provided by a Hough Forest, trained as described in Chapter 3, although the method developed in this chapter is not limited to this approach, but applicable to any kind of ISM. Additionally, each visual word stores a set of training samples (patches) that were assigned to it. These patches carry a label indicating if they appeared somewhere in the background (negative training set) or somewhere on a positive training sample. Those from positive training samples additionally store a relative offset vector to the corresponding object centroid. Given this information, the ISM is able to provide pixel-wise probabilities $p(y|x_i)$ for having a part of an object of category y at location i and a list of relative offset vectors to the object centroid.

The overall goal of the method is to fuse the provided information of the ISM in a probabilistically meaningful way, which jointly decides where in a test image instances of the learned category are depicted and which local features are part of the individual detections. The method is based on a random field formulation.

The core idea of the method presented in this chapter is to take the probabilistic formulation of [11], and reformulate it to better fit the special case of object detection with

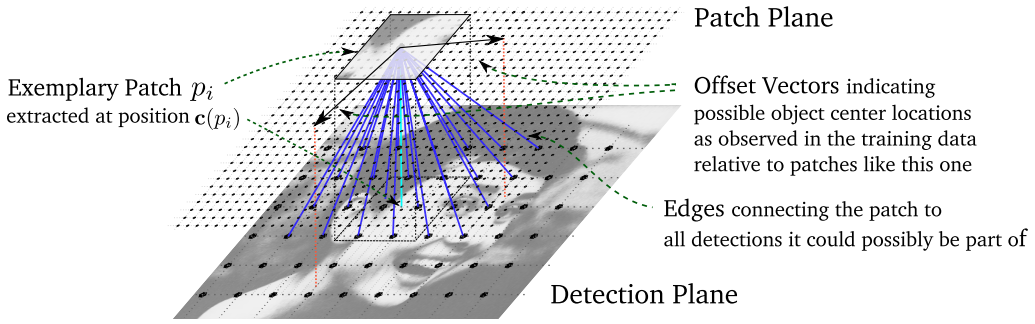


Figure 5.2: Constructing a random field for object detection: The graph consists of a set of nodes positioned at patches extracted at each pixel in the image (the patch plane) and a coarser grid of nodes defining possible locations of detection centroids (detection plane). Each patch node is connected to the detection nodes it could be part of, where offset vectors stored in the implicit shape model define the connections' likelihood. The novel inference process jointly solves the problems of detecting all objects and uniquely assigning contributing patch nodes to them.

Implicit Shape Models. One of the key insights of [11] was the following: An element in the ISM can vote for multiple objects at different positions in the image, because it was seen in training images on different locations relative to the object centroid. In one particular input image, however, each of the pixels is only part of exactly one object. This situation is illustrated in Figure 5.1. Thus, when solving the detection task we ultimately have to decide for each patch to which detection it belongs (or implicitly do so).

This chapter introduces an algorithm to efficiently solve this problem. Section 5.1.1 introduces the underlying graph structure. An important part is the novel definition of a semantic label space tuned to the specific task of localizing objects based on an ISM, which is described in Section 5.1.2. Finally, in Section 5.1.3, the random field energy minimization problem is defined and solved in Section 5.2.

5.1.1 Two-layer Graph Structure

Contrary to the generic formulation in [11], we make use of the fact that in an ISM an element cannot vote for every detection hypothesis, but only for those that are reachable with an offset vector. The offset vectors define a fixed set of detection nodes a patch can interact with, relative to its position.

We thus define a two-layer graph structure, as illustrated in Figure 5.2. The Graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ is defined by a set of nodes $\mathcal{V}$ and edges $\mathcal{E}$ connecting the nodes. The set of nodes $\mathcal{V}$ consists of patch nodes $\mathcal{P}$ at an image layer and detection nodes $\mathcal{D}$ at

a detection layer (*i.e.*, $\mathcal{V} = \mathcal{P} \cup \mathcal{D}$). The patch nodes $\mathcal{P} = \{p_0, \ldots, p_{wh}\}$ form a grid spanning the whole input image, with one node per extracted local feature for the ISM. In our case, this is the dense grid of pixels of the input image; w, h are width and height of the image. The set of detection nodes $\mathcal{D} = \{d_0, \ldots, d_{uv}\}$ defines a coarser grid of size $u \times v$, where each d_j specifies the center of a potential object detection.

Each patch node p_i is connected by an edge $e_{p_i,d_j} \in \mathcal{E}$ to every node d_j that defines a detection that p_i could potentially be part of. This means, if the pixel coordinates of patch p_i, which we will denote as $\mathbf{c}(p_i)$, lie within a hypothetical detection bounding box centered at $\mathbf{c}(d_j)$, then there is an edge e_{p_i,d_j} connecting them. This is illustrated in Figure 5.2 for one exemplary patch node.

Note that there are no connections between detection nodes in this graph. Such relations could additionally, explicitly implement local neighborhood suppression, but experiments showed that this is not required in this framework. Since patch nodes are not allowed to contribute to more than one detection in the inference process, stronger detections pull away evidence from nearby detections automatically. Thus, the method does not require to fix a range for local neighborhood suppression as necessary in non-maxima suppression methods, but is implicitly capable to separate even strongly overlapping object instances.

Using the graph $\mathcal{G}$, the random field is defined, by associating a random variable with each node (which will also be denote as p_i and d_j for simpler notation). Each random variable can be assigned one of the labels of the label set $\mathcal{L} = \{l_{\text{bg}}, l_{\text{fg}}, l_0, \ldots, l_n\}$. The label currently assigned to node v will be denoted as l_v and the set of assignments to all patch and detection nodes as $l_{\mathbf{p}}$ and $l_{\mathbf{d}}$, respectively.

5.1.2 Defining the Label Set

The semantics of assigning one of the labels to a node, which is the essential characteristic of the formulation presented here, is defined in the following way. Assigning the background label l_{bg} to a detection node (*i.e.*, $d_j = l_{\text{bg}}$) means that there is no detection at this position. Likewise, a configuration having $d_j = l_{\text{fg}}$ specifies that there is an object centered at $\mathbf{c}(d_j)$. For a patch node $p_i = l_{\text{bg}}$ signifies that at the center of the patch $\mathbf{c}(p_i)$ there is no object, but background. This does not imply that none of the detection nodes connected to p_i can be set to l_{fg}, since the bounding box of a detection might well contain some background pixels.

The crucial point of our framework is the meaning of the labels $l_0, \ldots, l_n$. Assigning one of these labels to a patch node indicates that this patch is part of a detection centered

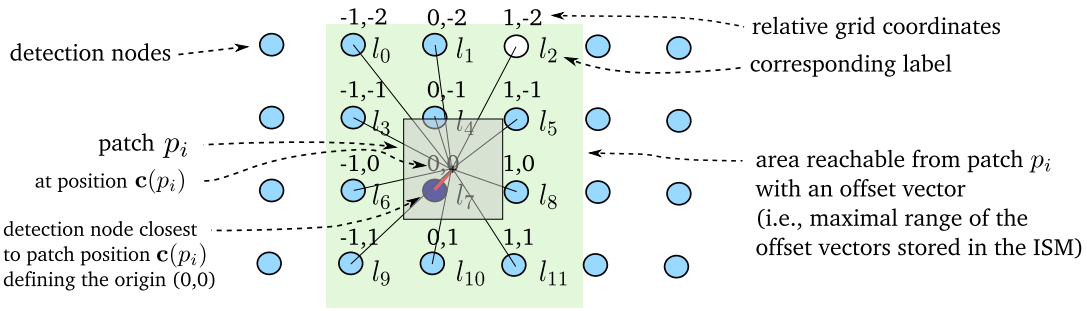


Figure 5.3: A patch p_i centered at pixel $\mathbf{c}(p_i)$ is connected to all detections it could potentially be part of. The relative position on the grid defines the semantics of the labels for this patch. For example, assigning label l_2 to the patch would mean that it is part of (votes for) a detection centered at the detection node (white) at position (1,-2) relative to its closest detection node (dark blue).

on a specific detection node, specified as follows. As shown in Figure 5.3, the detection node with the closest pixel coordinates to the patch (printed in dark blue) defines the origin of a coordinate system of relative offsets in the detection grid. From the training data, the maximal range of the offset vectors, stored with the codebook entries, can be determined. This range defines a fixed rectangular area of detection nodes that a patch could potentially vote for with its offset vectors. Within this area we reserve a separate label for each detection node. Assigning this label to the patch means that it is part of the corresponding detection. For an example, see Figure 5.3. Note that the set of labels is the same for all patch nodes. However, the semantic meaning of label assignments is spatially varying, since the label implicitly defines an assignment to different object hypotheses depending on the location of the patch. For notational convenience, we will denote the label that specifies that the patch at p_i is part of the object centered on detection node d_j as $\hat{l}_{i,j}$.

5.1.3 Energy Function

Given an input image I and the graph structure as defined in Section 5.1.1, the probability of an assignment of labels to all nodes, *i.e.*, a total configuration of the random field, can be written as

$$p(l_{\mathbf{p}}, l_{\mathbf{d}}|I) = \prod_{p_i \in \mathcal{P}} p(l_{p_i}|I) \prod_{d_j \in \mathcal{D}} p(l_{d_j}|I) \prod_{e_{p_i,d_j} \in \mathcal{E}} p(l_{p_i}, l_{d_j}|I)\,. \tag{5.1}$$

Taking the log of Equation (5.1) leads to the formulation of the energy function to be minimized:

$$E(l_{\mathbf{p}}, l_{\mathbf{d}}) = \sum_{p_i \in \mathcal{P}} \psi_{p_i}(l_{p_i}) + \sum_{d_j \in \mathcal{D}} \psi_{d_j}(l_{d_j}) + \sum_{e_{p_i,d_j}} \psi_{i,j}(l_{p_i}, l_{d_j}) \,, \tag{5.2}$$

where $\psi_{p_i}(l_{p_i}) = -\log(p(l_{p_i}|I))$ is the unary cost of assigning the label l_{p_i} to node p_i, $\psi_{d_j}(l_{d_j})$ is the equivalent for detection node d_j and $\psi_{i,j}(l_{p_i}, l_{d_j})$ is the resulting pairwise cost. With these definitions, finding the objects in the image amounts to finding the assignment of labels to all nodes that minimizes Equation (5.2).

5.1.3.1 Definition of Unary Potentials

Starting with the first term in Equation (5.1), $p(l_{p_i}|I)$ represents the probability of assigning the label l_{p_i} to node p_i, given the image data. Let x_i be the appearance of the local feature extracted around $\mathbf{c}(p_i)$. By making the same independence assumption as in [11], namely that the probability of a label on a patch only depends on its appearance x_i, we can define the posterior probability of the labeling of a patch node by $p(l_{p_i}|I) = p(l_{p_i}|x_i)$. This probability can be derived from the statistics collected in the ISM as follows.

In order to get an estimate of how likely a detection at a certain position is, given one patch, we have to sum up the voting weight cast by offset vectors that point from the patch to that detection. As in [66], the patches should be allowed to move slightly around their original offset position, accounting for small deformations of the object. This is modeled by aggregating the weigths of all voting vectors by a Gaussian centered at the detection.

This summing up of evidence for an object center around a detection node also has a different interpretation. In order to achieve tolerance for small shifts of the patch, we could also resample the training set and insert additional offset vectors pointing to positions around the original centroid location, giving the same effect as the smoothing with a Gaussian. Unfortunately, this smoothing or resampling introduces additional virtual samples that change the ratio of positive to negative samples in the ISM statistics. Thus, it is not possible to directly take the summed up voting weights at each detection node as probabilities for the labels. Correcting for this bias would be a tedious task since the amount of virtually introduced samples depends on the density of the detection grid and the distribution of offset vectors.

Additionally, the statistics stored with the codebook are not completely reliable, as for instance an entry with no single negative training patch would indicate zero prob-

ability for a patch with this appearance to appear somewhere in the background. This almost certainly does not reflect truth but is an artifact of insufficient training data.

Barinova *et al.* [11] bypass these problems, by setting the probability for assigning background to a patch node to a constant chosen on a validation set. We take a different approach, trying to make more use of the inexact but nonetheless valuable information stored with the codebook entries. We take the probability of being foreground (pfg_{p_i}) estimated from the original ratio of training samples stored in the ISM and estimate $p(p_i = l_{\text{bg}}|x_i)$ by taking it as input to the shifted sigmoidal function:

$$p(p_i = l_{\text{bg}}|x_i) = 1 - \frac{pfg_{\max}}{1 + \exp(-\alpha(pfg_{p_i} - \beta))}\,. \tag{5.3}$$

All parameters of this function can be estimated once on a validation set and are kept fixed at $pfg_{\max} = 0.95, \alpha = 10, \beta = 0.4$. This procedure of limiting the foreground probability to a maximum value of $pfg_{\max}$ can also be seen as combining the estimated distribution with a uniform Dirichlet prior. The probabilities for the labels $l_0, \ldots, l_n$ are then defined by taking the evidence gathered above for each detection node and scaling it such that the maximum reaches $1 - p(p_i = l_{\text{bg}}|x_i)$.

The second term of Equation (5.1), $p(l_{d_j}|I)$, encodes the probability for a label on a detection node. This can be used to express a prior probability for a detection. However, in practice by setting $p(d_j = l_{\text{bg}}) = p(d_j = l_{\text{fg}}) = 0.5$ no assumptions are made about the distribution or frequency of detections. Detection nodes can thus be seen as auxiliary variables, collecting the information of its connected patch nodes via the pairwise relations. All other labels are invalid for detection nodes, so their probability is set to 0.

5.1.3.2 Definition of Pairwise Potentials

The pairwise costs $\psi_{i,j}(l_{p_i}, l_{d_j})$ reflect the semantics of the labels for the relationship between patch nodes and detection nodes. Figure 5.4 shows a simple example with one exemplary patch node connected to three detection nodes. The tables on the left list the costs for all different kinds of label configurations for one patch and its neighboring detection nodes. The first row contains the unary cost for assigning each label to the patch node $\psi_{p_i}(l_{p_i})$, as defined above. The three separate tables below show the pairwise costs for combinations of label assignments to the patch and each detection node. Each has one row for the costs of assigning l_{bg} and l_{fg} to the detection node respectively. The blue frames mark the column with the costs for assigning label $\hat{l}_{i,j}$ to the patch, which means that the patch p_i is part of the corresponding detection d_j.

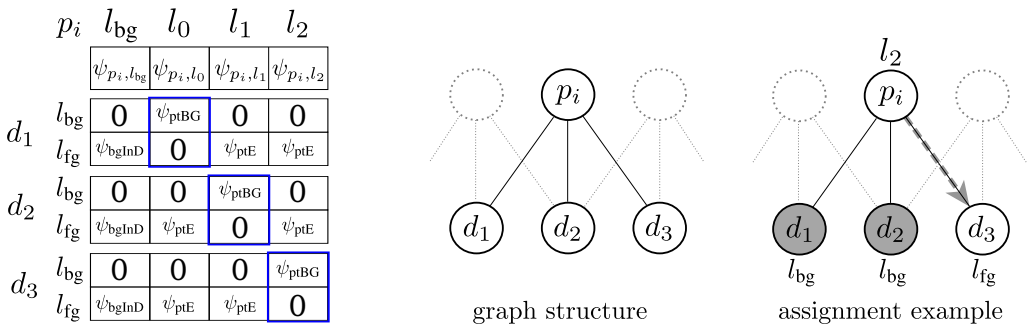


Figure 5.4: A simple 2D example: each patch node connects to three detection nodes. The table on the left shows all unary (first row) and binary costs for all possible labeling combinations for one patch node p_i and its associated detection nodes. Example shown on the right: Let $p_i = l_2$ (*i.e.*, patch p_i votes for detection d_3) and the detection nodes are set to $d_1 = l_{\text{bg}}, d_2 = l_{\text{bg}}, d_3 = l_{\text{fg}}$; the total cost of the configuration is $\psi_{p_i}(l_2) + \psi_{i,1}(l_2, l_{\text{bg}}) + \psi_{i,2}(l_2, l_{\text{bg}}) + \psi_{i,3}(l_2, l_{\text{fg}}) = \psi_{p_i,l_2} + 0 + 0 + 0$.

If detection and patch are both assigned to background, this is a valid combination and the cost is 0. The same is true if a detection is set to background ($d_j = l_{\text{bg}}$) and the patch is set to anything else but $\hat{l}_{i,j}$, meaning that it is not part of this detection. Furthermore, switching the detection on ($d_j = l_{\text{fg}}$) and setting the patch to be part of it ()$p_i = \hat{l}_{i,j}$ also results in 0 cost. A patch being part of a detection at an inactive detection node (*i.e.*, $p_i = \hat{l}_{i,j} \wedge d_j = l_{\text{bg}}$) is an invalid configuration resulting in a cost of ψ_{ptBG}, which we can set to ∞ (or in practical implementations to a very high cost). Conversely, a patch assigned to background $p_i = l_{\text{bg}}$, in the range of an active detection $d_j = l_{\text{fg}}$, adds a fixed cost ψ_{bgInDet}, derived from the probability that a pixel inside a detection rectangle might be background, which can be estimated from the training data. This expresses the fact that objects in the training and test data do not completely fill the bounding box they are annotated with. This parameter also controls how much of an object must be visible (not occluded) for a valid detection. Finally, from the point of view of the detection, there is no difference if the patch is assigned to background or to any other detection close by, so $\psi_{\text{ptE}} = \psi_{\text{bgInDet}}$.

5.2 Inference

Since the pairwise costs, as defined above, fulfill the conditions of *regularity* [86], we could, *e.g.*, apply standard graphcut-based inference methods such as alpha expansion or alpha/beta swap [21] to solve the labeling problem. However, generic solving algorithms fail for the particular graph structure and definition of potentials. The main

problem is that trying to change a single node, or even all nodes, to exactly one new label can almost never result in a lower energy.

For example, as can be seen from Figure 5.4, if all nodes are assigned the background label l_{bg}, switching a patch node to any different label will result in adding the very high cost ψ_{ptBG} at one binary relation. Switching a single detection node to l_{fg} will not change the unary cost for this node but increase the total energy of each pairwise edge that connects this detection node to any patch node by the cost ψ_{bgInDet}. Thus, setting every node to l_{bg} results in a strong local minimum of the energy and thus, inference approaches like alpha-expansion that only consider changing nodes to a single, new label per iteration, immediately fail.

For this reason, we propose an inference approach tuned to the specific graph structure and label semantics. The core idea is a novel move making strategy, which is described in detail in Section 5.2.1. The corresponding inference process is outlined in Section 5.2.2, while Section 5.2.3 discusses the overall characteristics of our inference approach.

5.2.1 Moves

We propose to use a different kind of move, specialized for the problem setup at hand, that changes the labels of several nodes simultaneously. The central observation, that was also already pointed out in [11], is that given a labeling of the detection nodes, the optimal label for each patch can be determined independently, since the graph is bipartite. Careful inspection of the setup reveals that the new optimal assignment can efficiently be computed for each patch node when a single detection node changes its label in $O(1)$, if the previously optimal assignment and cost is known. Thus, the following efficient inference algorithm can be designed.

The prerequisite of a starting point with known optimal assignments of the patch nodes and total costs is easily fulfilled by setting all detection nodes to l_{bg}. The optimal label for each patch is then also l_{bg}, because any other label would add ψ_{ptBG} to the total cost. The total energy of this configuration amounts to the sum of unary costs for l_{bg} of all detection and patch nodes (all pairwise costs are 0). The sum of costs for each label at each patch node, that we will need later in the process, is its unary cost plus, for each label other than l_{bg}, one binary cost of ψ_{ptBG}.

Then, we consecutively turn on one detection d_j after the other, and find the optimal configuration of patch node labels for the new situation, to discover which one lowers the total energy most. To compute the total energy of a new configuration we need

to keep track of the change of energy ΔE for each node that changes its label during the process, and affected edges. The change in unary cost for the detection node is $-\psi_{d_j}(l_{\text{bg}}) + \psi_{d_j}(l_{\text{fg}})$. Since none of the connected patch nodes can have pointed to d_j before (since it was background), all pairwise relations switch from 0 cost to ψ_{bgInDet} or ψ_{ptE} (which are equal).

Now, we have to check for each patch node p_i connected to the currently tested detection node d_j, for the new best label. The optimal label for a patch node depends on the patch's unary cost for the label, plus the pairwise to all detection nodes it is connected to. To find the label with lowest energy in a brute force manner, one would have to go over all labels and for each of them sum up the binary costs of all the edges of the patch. Despite the sparse graph structure in which a patch is not connected to all detection nodes, this would require $O(|L|^2)$. But, in fact, for every label the only change in energy is in the pairwise connection between the patch and the changed detection d_j, so we can update them incrementally. Additionally, not all the labels have to be checked to find the new best one, since it is known that the label currently assigned to p_i was the best one before the current move. Looking at Figure 5.4 it can be seen that only switching to $\hat{l}_{i,j}$ can possibly result in a decrease of the energy. So the only possibility we have to check is, if the total cost of the patch's old label is now bigger than the cost for $\hat{l}_{i,j}$. We keep track of the total change of costs for the better of those two possibilities. This is an $O(1)$ operation for each patch.

5.2.2 Overall Inference Process

Thus, in total, calculating the change in energy for switching on a single detection hypothesis and finding the optimal configuration of patch labels is a fast operation. Therefore, we can afford to test every single detection hypothesis and take the best one, without having to rely on a heuristic to propose potentially good hypotheses. After the new best detection hypothesis is found, the corresponding detection node d_j is switched to l_{fg} and each patch node, for which this results in a better energy, is set to $\hat{l}_{i,j}$, to associate it with the new detection Then the costs for each label are updated. This is only done once per newly found detection and only for the patches connected to the new detection. The whole process is repeated until no move lowers the total energy.

5.2.3 Discussion

Note that the decision of finally taking the most probable detection in each iteration is greedy. However, the greedy decision is based on the evaluation of every single possible move that switches on one detection node and finds the new optimal configuration of all patch nodes. Even after the move is taken, the patch nodes that were switched to the new detection in this iteration are not fixed to this decision, but can switch to a different detection found later, if this again decreases the total energy.

Additionally, the order of configurations checked by the algorithm assures fast convergence to a good minimum of the energy by making use of domain knowledge. For instance, in a generic solver it would be hard to exploit the fact that switching on lots of detection nodes at once is very unlikely to give a low energy, or encode the knowledge which set of labels to apply to the corresponding patch nodes.

Furthermore, note that after the first iteration not all possible remaining hypotheses have to be checked again to find the next best detection. The total benefit (reduction of cost) for each hypothesis can only become smaller with the new detection from the last iteration now switched on and adding pairwise costs to every patch node not pointing towards it. Thus, we do not have to check those hypotheses that already did not have a negative ΔE in the last run. Since even for a crowded scene the number of objects is way lower than the number of detection nodes, this again dramatically reduces the search space.

5.3 Experiments

As stated above, the codebook is created by training a Hough Forest, as presented in Chapter 3. The smoothing kernel's sigma is set to $\sigma = 3.0$ to allow for small shifts of the patches with respect to the object center. Derived from this, the resolution of the detection grid is set to 8×8. A coarser grid would miss detections, because the patches can only vote for detection nodes within the range of the Gaussian. A denser grid would linearly increase computation time with the number of detection nodes. The only parameter left to set is ψ_{bgInDet}. Basically, it defines how much of an object must be visible in order to create a positive detection. Since we want to detect highly overlapping instances, we set it quite low, to a value of 0.4. Conversely this implies a high probability of about $e^{-0.4} \approx 67\%$ of a patch to be background within a valid detection.

To get multi-scale detection results, each scale is first processed individually. Subsequently, according to our localization principle, we ensure that also over scales each

patch only votes for a single detection. From the final configurations of the random fields for each scale, we obtain all detections and the corresponding set of patches assigned to them, which defines a pixel-wise voting mask per detection (see Figures 5.6 and 5.7). We collect these masks over all scales and resize them to a reference frame. Then we sort all detections by their confidences and, starting with the most confident, accept only detections that do not overlap (considering the voting masks) with those already taken. Thus, we again ensure that also over scales each patch is only assigned to a single detection. Thereby, we effectively suppress lower scoring redetections in nearby scales and obtain a unified solution for multi-scale analysis.

Similar to [66], rectangles of mean aspect ratio (estimated from the training data) centered at each active detection node are reported. The confidence of each reported detection is set to the absolute value of the decrease in energy that was recorded during testing the corresponding detection node.

5.3.1 Datasets

The choice of evaluation datasets is motivated by several factors. First we want to have a direct comparison to the most closely related approach [11]. The publicly available implementation comes with its own set of random forests, trained for detection of side views of pedestrians. Thus, we also focus on this task, although our method is not specifically tailored towards it and potentially handles arbitrary object categories.

Another aspect is the resolution of the objects in the images. Part-based approaches, like ISMs, can only capitalize on their strengths if the objects are depicted at a resolution where the parts are distinguishable. Thus we require the smallest category instances to have at least about 100 pixels in height.

Since for non overlapping object instances our proposed method reaches the same decisions as standard NMS (as was tested and assured in evaluations on single scale datasets like UIUC cars), we are especially interested in testing the capability of our algorithm to resolve detections of strongly overlapping objects. Thus, we evaluate it on the *TUD crossing* and *TUD campus* sequences, presented in Section 3.6.1.1 and also used in [11]. Both datasets require the ability to locally decide for each patch to which detection it belongs in a reasonable manner, in order to identify heavily overlapped pedestrians. Additionally, we evaluate all approaches on the PETS 2009 dataset, also featuring close to side views of a large number of pedestrians with heavy overlaps.

5.3.2 Results

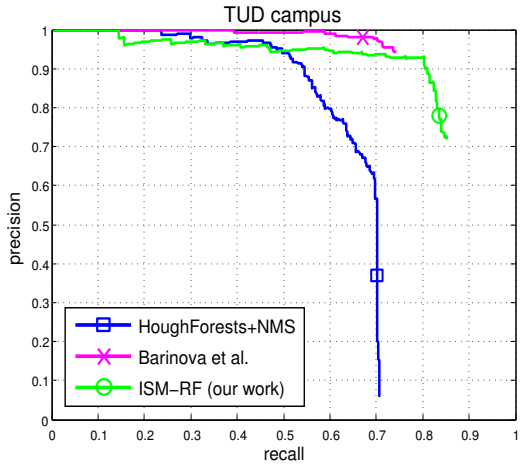
The results are directly compared to the two most related approaches: Hough Forests using standard non maximum suppression [66] and the probabilistic framework of Barinova *et al.* [11]. Detections are considered as valid analyzing the standard PASCAL-VOC overlap criterion, with the threshold set to 50%. For both methods compared, we used the publicly available source codes and associated configuration files as published by the respective authors.

Figure 5.5 shows Precision/Recall Curves (PRCs) for all three methods on all three databases. As can be seen, the method presented here significantly improves over [66] and also outperforms [11] on all three datasets. At precision levels above 90% the recall is improved by over 10% and extending way further for lower values of precision. For PETS and TUD crossing, where very accurate ground-truth annotation is available, the precision also stays close to 100% for up to 60% recall. The small earlier breakdown on TUD campus can partly be attributed to incorrectly identified false positives, because the ground-truth does not include persons of which only a few pixels are visible.

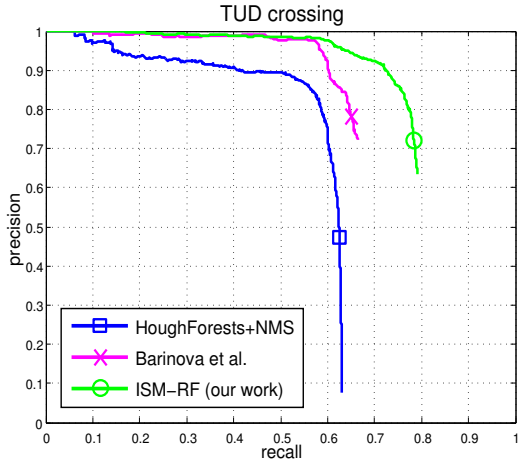
Figures 5.6 and 5.7 additionally visualize detection results. The local patches that were assigned to each detected instance by the inference process are shown in different colors. Note, how the assignments of patches to detections provide accurate segmentations of each individual object instance and even strongly overlapping pedestrians are correctly separated from each other.

5.4 Conclusion

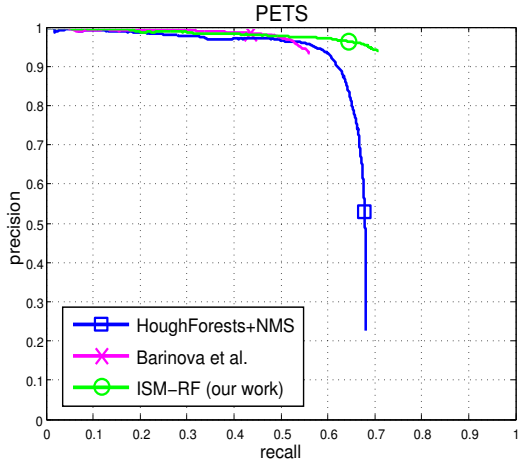
In this chapter, it was observed that the classical inference approach for ISMs sums over the evidence coming from local features, ignoring mutually inconsistent votes. To address this problem, the dual problem of detecting valid object hypotheses and assigning local patches to the detections was formulated in a random field, with a significantly sparser graph structure than in related approaches. Furthermore, the specific graph structure facilitates the definition a novel, fast inference algorithm to solve the energy minimization problem defined by the random field formulation. As an additional benefit, the approach does not require to fix a range for local neighborhood suppression as it is necessary in related methods, but is implicitly capable to separate even strongly overlapping object instances. Experiments demonstrated that object hypotheses and their local support patches can be detected accurately on challenging data sets, achieving competitive or even improved results in comparison to the state-of-the-art.



(a)



(b)



(c)

Figure 5.5: Precision/Recall curves on (a) *TUD campus*, (b) *TUD crossing* and (c) PETS 2009 S1.L1 sequence for all three methods.

Figure 5.6: Sample detections on the TUD crossing. For each detection the uniquely assigned patches are plotted in a different color. Note how closely walking pedestrians, overlapping each other, are correctly separated.

Figure 5.7: Sample detections on PETS 2009. For each detection the uniquely assigned patches are plotted in a different color. Note how each person is accurately segmented and closely walking pedestrians, overlapping each other, are correctly separated.

6 Conclusion

This thesis addressed detection of objects of specific classes. The main focus was on robustness to occlusions and the ability to identify object instances that appear very close to each other, potentially overlapping each other. Additionally, handling of object classes with high intra-class variance, stemming from variations in pose, the flexibility of individual objects and differences between object instances, was an essential requirement.

In Chapter 2 a general overview over object detection methods was given. Implicit Shape Models and, in particular, Hough Forests were identified as methods that are particularly suited for the targeted situations. The overview and discussion additionally clarifies the relations of this approach to other methods in the field. Essentially, although the linear formation of the confidence score in the voting process makes it formally equivalent to holistic models, as discussed in Section 2.5, the extraction of an over-complete set of parts and implicit definition of the shape gives a different view on the object detection process and led to different classification and non maxima suppression schemes.

Chapter 3 introduced the Hough Forest framework as the basis for all extensions and improvements in the rest of the thesis. As part of it, a novel method to evaluate split test candidates for the nodes of the decision trees that try to improve the regression of the object's center was introduced. The chapter was concluded with an extensive discussion and evaluation of the parameters of Hough Forests and their interdependence. Among other insights, it showed the superiority of the new regression node evaluation criterion, leading to improved detection performance with shallower decision trees. Additionally, concluding experiments showed the performance of the best performing setup in comparison to other state-of-the-art methods on typical benchmarks.

The remaining two chapters each addressed one particular enhancement, dealing with a specific aspect of the approach. Chapter 4 introduced discriminative Hough Forests. The essential observation was that the training and testing procedures of Hough Forests treats all local parts completely independently. This, on the one hand, creates a very flexible model, that makes it possible to form constellations of sub-parts from arbitrary training data. Thus, new object instance can be detected that were not seen like that in any of the individual training images. On the other hand, this flexibility also makes it vulnerable to false positives in backgrounds that feature random collections of elements that look somewhat similar to sub-parts of the object.

With increasing amounts of training data, this flexibility becomes less important and, thus, can be traded off against more accurate detections by enforcing more consistency in the constellations of parts. Thus, an approach was formulated to classify detections in order to reject wrong constellations. The analysis showed that the voting process with the object model gives a distinctive description of a detection hypothesis that was termed activation vector, recording the contribution of each voting element to the score of the detection. By collecting such descriptions from correct and wrong detections classifiers can be built to re-score detections and decrease the score of false ones. Additionally, when using a linear classifier, the weights which are learned for each activation vector entry can be used directly in the voting process. This eliminates the need to first do normal voting, then explicitly calculate activation vectors and use a classifier only for post-processing. The improved evidence maps (Hough spaces), thus, not only lead to better precision but can also lead to better recall.

Finally, in Chapter 5 the problem of non-maxima suppression in case of strongly overlapping objects was addressed. As a starting point, it was pointed out that the ISM voting process inherently creates sets of mutually inconsistent votes. Each voting element can vote for multiple object centers, but in fact, the central pixel of each of them can only be part of one specific object in the image. Thus only one the votes agreeing on the correct object center are valid. The others contribute to clutter in the background and if enough of them by chance agree on a hypothesis, this creates false positive detections. Thus a random field formulation and a novel inference algorithm were developed to resolve these conflicting assignments of image patches to detection hypothesis. It allows for identifying most likely configurations and, consequently, removing evidence for conflicting hypotheses, effectively cleaning up the Hough space. The evaluation showed improved detection performance, especially on images with larger crowds of pedestrians, overlapping and occluding each other.

The results also indicate some new directions for future work. As an extension of the ideas pursued in the discriminative Hough Forests, it would be interesting not only to optimize the weights but also the location of the votes. This could be achieved by shifting the votes according to a gradient descent on a loss function. Further, each voting element could have a full map of voting directions which, however, would make the problem quite big and slow down the inference. Again, the reduction to a sparse grid of detection hypotheses, as proposed in Chapter 5, could be used to reduce the complexity.

Another promising direction could be to fuse the approaches of the last two chapters and learn discriminative voting weights directly for good inference in the random field. It would effectively lead to a (quite complex) bi-level optimization problem. The main benefit would be to get rid of the good, but nevertheless heuristic formulation of the unary potentials in the random field.

Additionally, the equivalence of the score formation process, pointed out in Section 2.5, suggests that the non-maxima suppression method introduced in Chapter 5 would also be applicable in holistically trained object detection models, for which the descriptor can be subdivided into individually scored local sub-fields, such as the cells in a HOG descriptor.

To conclude, object detection with ISMs, Hough Forests and related methods and the contributions presented in this thesis have shown success in a series of applications, but also room for improvement in comparison to other methods in certain other fields and some open questions and, thus, still remains an interesting field for further research.

A

List of Publications

2013

Optimizing 1-nearest prototype classifiers

Paul Wohlhart, Martin Koestinger, Michael Donoser, Peter M. Roth and Horst Bischof
In: *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2013

Alternating Decision Forests

Samuel Schulter, Paul Wohlhart, Christian Leistner, Amir Saffari, Peter M. Roth and Horst Bischof
In: *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2013

Alternating Regression Forests for Object Detection and Pose Estimation

Samuel Schulter, Christian Leistner, Paul Wohlhart, Peter M. Roth and Horst Bischof
In: *Proc. International Conference on Computer Vision (ICCV)*, 2013

Joint Learning of Discriminative Prototypes and Large Margin Nearest Neighbor Classifiers

Martin Koestinger, Paul Wohlhart, Peter M. Roth and Horst Bischof
In: *Proc. International Conference on Computer Vision (ICCV)*, 2013

2012

Discriminative Hough Forests for Object Detection

Paul Wohlhart, Samuel Schulter, Martin Koestinger, Peter M. Roth and Horst Bischof
In: *Proc. British Machine Vision Conf. (BMVC)*, 2012

Detecting Partially Occluded Objects with an Implicit Shape Model Random Field

Paul Wohlhart, Michael Donoser, Peter M. Roth and Horst Bischof
In: *Proc. Asian Conf. on Computer Vision (ACCV)*, 2012, (*Winner Best Paper Award*)

Robust Face Detection by Simple Means

Martin Koestinger, Paul Wohlhart, Peter M. Roth and Horst Bischof
In: *Computer Vision in Applications Workshop (DAGM)*, 2012

Large Scale Metric Learning from Equivalence Constraints

Martin Koestinger, Martin Hirzer, Paul Wohlhart, Peter M. Rotha and Horst Bischof
In: *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2012

2011

Multiple Instance Boosting for Face Recognition in Videos

Paul Wohlhart, Martin Koestinger, Peter M. Roth and Horst Bischof
In: *Proc. DAGM Symposium*, 2011

Learning Face Recognition in Videos from Associated Information Sources

Paul Wohlhart, Martin Koestinger, Peter M. Roth and Horst Bischof
In: *Proc. Workshop of the Austrian Association for Pattern Recognition (AAPR)*, 2011

Learning to Recognize Faces from Videos and Weakly Related Information Cues

Martin Koestinger, Paul Wohlhart, Peter M. Roth and Horst Bischof
In: *Proc. IEEE Int'l Conf. on Advanced Video and Signal-Based Surveillance (AVSS)*, 2011

Annotated Facial Landmarks in the Wild: A Large-scale, Real-world Database for Facial Landmark Localization

Martin Koestinger, Paul Wohlhart, Peter M. Roth and Horst Bischof
In: *First IEEE International Workshop on Benchmarking Facial Image Analysis Technologies (BeFIT)*, 2011

2010

3D Camera Tracking in Unknown Environments using On-line Keypoint Learning

Paul Wohlhart, Peter M. Roth and Horst Bischof
In: *Proc. Computer Vision Winter Workshop (CVWW)*, 2010

Video Detection of Dangerous Goods Vehicles in Road Tunnels

Josef Birchbauer, Martin Koestinger, Paul Wohlhart, Peter M. Roth, Horst Bischof and Claudia Windisch
In: *Proc. Tunnel Safety and Ventilation*, 2010

Automatic Detection and Reading of Dangerous Goods Plates

Peter M. Roth, Martin Koestinger, Paul Wohlhart, Horst Bischof and Josef Birchbauer
In: *Proc. IEEE Int'l Conf. on Advanced Video and Signal-Based Surveillance (AVSS)*, 2010

Bibliography

[1] Achanta, R., Shaji, A., Smith, K., Lucchi, A., Fua, P., and Suesstrunk, S. (2012). SLIC superpixels compared to state-of-the-art superpixel methods. *IEEE Trans. on PAMI*, 34(11). (cited on page 7)

[2] Agarwal, S., Awan, A., and Roth, D. (2004). Learning to detect objects in images via a sparse, part-based representation. *IEEE Trans. on PAMI*, 26(11):1475–1490. (cited on page 50)

[3] Alexe, B., Deselaers, T., and Ferrari, V. (2012). Measuring the objectness of image windows. *IEEE Trans. on PAMI*, 34(11):2189–2202. (cited on page 7)

[4] Alexe, B., Deselares, T., and Ferrari, V. (2010). What is an object? In *Proc. IEEE Intern. Conf. on Computer Vision and Pattern Recognition*. (cited on page 7)

[5] Amit, Y. and Geman, D. (1997). Shape quantization and recognition with randomized trees. *Neural Computation*, 9:1545–1588. (cited on page 28)

[6] Andriluka, M., Roth, S., and Schiele, B. (2008). People-tracking-by-detection and people-detection-by-tracking. In *Proc. IEEE Intern. Conf. on Computer Vision and Pattern Recognition*. (cited on pages 52 and 86)

[7] Arandjelović, R. and Zisserman, A. (2012). Three things everyone should know to improve object retrieval. In *Proc. IEEE Intern. Conf. on Computer Vision and Pattern Recognition*. (cited on page 9)

[8] Arandjelović, R. and Zisserman, A. (2013). All about VLAD. In *Proc. IEEE Intern. Conf. on Computer Vision and Pattern Recognition*. (cited on page 10)

[9] Arbelaez, P., Maire, M., Fowlkes, C., and Malik, J. (2011). Contour detection and hierarchical image segmentation. *IEEE Trans. on PAMI*, 33(5):898–916. (cited on page 7)

[10] Ballard, D. H. (1981). Generalizing the hough transform to detect arbitrary shapes. *Pattern Recognition*, 13(2):111–122. (cited on page 36)

[11] Barinova, O., Lempitsky, V., and Kohli, P. (2010). On the detection of multiple object instances using Hough transforms. In *Proc. IEEE Intern. Conf. on Computer Vision and Pattern Recognition*. (cited on pages 92, 93, 96, 97, 99, 102, and 103)

[12] Bay, H., Ess, A., Tuytelaars, T., and Van Gool, L. (2008). Speeded-up robust features (SURF). *Computer Vision and Image Understanding*, 110(3):346–359. (cited on pages 7, 8, and 9)

[13] Benenson, R., Mathias, M., Tuytelaars, T., and Van Gool, L. (2013). Seeking the strongest rigid detector. In *Proc. IEEE Intern. Conf. on Computer Vision and Pattern Recognition*. (cited on pages 8, 9, 29, and 54)

[14] Berg, A., Berg, T., and Malik, J. (2005). Shape matching and object recognition using low distortion correspondences. In *Proc. IEEE Intern. Conf. on Computer Vision and Pattern Recognition*. (cited on page 8)

[15] Berg, T. L., Berg, A. C., Edwards, J., and Forsyth, D. A. (2004). Who's in the picture. In *Advances in Neural Information Processing Systems*. (cited on page 52)

[16] Blaschko, M. and Lampert, C. (2008). Learning to localize objects with structured output regression. In *Proc. European Conf. on Computer Vision*. (cited on page 19)

[17] Bosch, A., Zisserman, A., and Muñoz, X. (2011). Image classification using random forests and ferns. In *Proc. IEEE Intern. Conf. on Computer Vision*. (cited on pages 8 and 12)

[18] Bouchard, G. and Triggs, B. (2005). Hierarchical part-based visual object categorization. In *Proc. IEEE Intern. Conf. on Computer Vision and Pattern Recognition*. (cited on page 14)

[19] Bourdev, L. and Brandt, J. (2005). Robust object detection via soft cascade. In *Proc. IEEE Intern. Conf. on Computer Vision and Pattern Recognition*. (cited on page 20)

[20] Bourdev, L. and Malik, J. (2009). Poselets: Body part detectors trained using 3d human pose annotations. In *Proc. IEEE Intern. Conf. on Computer Vision*. (cited on page 13)

[21] Boykov, Y., Veksler, O., and Zabih, R. (2001). Fast approximate energy minimization via graph cuts. *IEEE Trans. on PAMI*, 23:1222–1239. (cited on page 98)

[22] Bradski, G. (2000). The OpenCV Library. *Dr. Dobb's Journal of Software Tools*. (cited on pages 44 and 54)

[23] Breiman, L. (2001). Random forests. *Machine Learning*, 45(1):5–32. (cited on pages 28, 29, 47, 48, and 81)

[24] Brown, M., Hua, G., and Winder, S. (2011). Discriminative learning of local image descriptors. *IEEE Trans. on PAMI*, 33(1):43–57. (cited on page 10)

[25] Bruce, N. and Tsotsos, J. (2006). Saliency based on information maximization. In *Advances in Neural Information Processing Systems*. (cited on page 6)

[26] Calonder, M., Lepetit, V., Ozuysal, M., Trzcinski, T., Strecha, C., and Fua, P. (2012). BRIEF: Computing a local binary descriptor very fast. *IEEE Trans. on PAMI*, 34(7):1281–1298. (cited on page 9)

[27] Carneiro, G. and Lowe, D. (2006). Sparse flexible models of local features. In *Proc. European Conf. on Computer Vision*. (cited on page 14)

[28] Carreira, J. and Sminchisescu, C. (2012). CPMC: Automatic object segmentation using constrained parametric min-cuts. *IEEE Trans. on PAMI*, 34(7):1312–1328. (cited on page 7)

[29] Chang, K. Y., Liu, T. L., Chen, H. T., and Lai, S. H. (2011). Fusing generic objectness and visual saliency for salient oject detection. In *Proc. IEEE Intern. Conf. on Computer Vision*. (cited on page 7)

[30] Chen, H.-T., Chang, H.-W., and Liu, T.-L. (2005). Local discriminant embedding and its variants. In *Proc. IEEE Intern. Conf. on Computer Vision and Pattern Recognition*. (cited on page 10)

[31] Chow, C. and Liu, C. (1968). Approximating discrete probability distributions with dependence trees. *IEEE Trans. on Information Theory*, 14(3):462–467. (cited on page 21)

[32] Coates, A. and Ng, A. Y. (2011). The importance of encoding versus training with sparse coding and vector quantization. In *Proc. IEEE Intern. Conf. on Machine Learning*. (cited on page 11)

[33] Comaniciu, D. and Meer, P. (2002). Mean shift: A robust approach toward feature space analysis. *IEEE Trans. on PAMI*, 24(5):603–619. (cited on page 38)

[34] Cootes, T., Ionita, M. C., Lindner, C., and Sauer, P. (2012). Robust and accurate shape model fitting using random forest regression voting. In *Proc. European Conf. on Computer Vision*. (cited on pages 31, 42, and 70)

[35] Cootes, T. F., Taylor, C. J., Cooper, D. H., and Graham, J. (1995). Active shape models - their training and application. *Computer Vision and Image Understanding*, 61(1):38–59. (cited on page 14)

[36] Cover, T. and Thomas, J. (2006). *Elements of Information Theory*. Wiley. (cited on page 31)

[37] Crandall, D., Felzenszwalb, P., and Huttenlocher, D. (2005). Spatial priors for part-based recognition using statistical models. In *Proc. IEEE Intern. Conf. on Computer Vision and Pattern Recognition*. (cited on page 14)

[38] Criminisi, A. and Shotton, J. (2013). *Decision Forests for Computer Vision and Medical Image Analysis*. Advances in Computer Vision and Pattern Recognition. Springer. (cited on page 31)

[39] Csurka, G., Bray, C., Dance, C., and Fan, L. (2004). Visual categorization with bags of keypoints. *Proc. European Conf. on Computer Vision Workshops*. (cited on pages 8 and 14)

[40] Csurka, G. and Perronnin, F. (2011). Fisher vectors: Beyond bag-of-visual-words image representations. In *Int. Joint Conf. on Computer Vision, Imaging and Computer Graphics Theory and Applications*. (cited on page 10)

[41] Dalal, N. and Triggs, B. (2005). Histograms of oriented gradients for human detection. In *Proc. IEEE Intern. Conf. on Computer Vision and Pattern Recognition*. (cited on pages 8, 9, 19, 29, and 44)

[42] Dantone, M., Gall, J., Fanelli, G., and van Gool, L. (2012). Real-time facial feature detection using conditional regression forests. In *Proc. IEEE Intern. Conf. on Computer Vision and Pattern Recognition*. (cited on pages 18 and 70)

[43] Dollár, P., Appel, R., Belongie, S., and Perona, P. (2014). Fast feature pyramids for object detection. *IEEE Trans. on PAMI*. (cited on pages 75 and 76)

[44] Dollár, P., Appel, R., and Kienzle, W. (2012). Crosstalk cascades for frame-rate pedestrian detection. In *Proc. European Conf. on Computer Vision*. (cited on pages 8, 9, 20, and 30)

[45] Dollár, P., Belongie, S., and Perona, P. (2010). The fastest pedestrian detector in the west. In *Proc. British Machine Vision Conf.* (cited on pages 8, 9, 10, and 30)

[46] Dollár, P., Tu, Z., Perona, P., and Belongie, S. (2009). Integral channel features. In *Proc. British Machine Vision Conf.* (cited on pages 8, 9, 29, 30, and 54)

[47] Donoser, M., Riemenschneider, H., and Bischof, H. (2010). Efficient partial shape matching of outer contours. In *Proc. Asian Conf. on Computer Vision*. (cited on page 8)

[48] Endres, I. and Hoiem, D. (2010). Category independent object proposals. In *Proc. European Conf. on Computer Vision*. (cited on page 7)

[49] Everingham, M., Van Gool, L., Williams, C. K. I., Winn, J., and Zisserman, A. (2011). The PASCAL Visual Object Classes Challenge 2011 (VOC2011) Results. http://www.pascal-network.org/challenges/VOC/voc2011/workshop/index.html. (cited on page 44)

[50] Fanelli, G., Dantone, M., Gall, J., Fossati, A., and Van Gool, L. (2013). Random forests for real time 3d face analysis. *Intern. Journal of Computer Vision*, 101(3):437–458. (cited on page 59)

[51] Fanelli, G., Gall, J., and Van Gool, L. (2011a). Real time head pose estimation with random regression forests. In *Proc. IEEE Intern. Conf. on Computer Vision and Pattern Recognition*. (cited on page 59)

[52] Fanelli, G., Weise, T., Gall, J., and Van Gool, L. (2011b). Real time head pose estimation from consumer depth cameras. In *Proc. DAGM Symposium*. (cited on page 59)

[53] Fei-Fei, L., Fergus, R., and Perona, P. (2003). A bayesian approach to unsupervised one-shot learning of object categories. In *Proc. IEEE Intern. Conf. on Computer Vision*. (cited on page 14)

[54] Felzenszwalb, P. and Huttenlocher, D. (2004a). Distance transforms of sampled functions. Technical Report 2004-1963, Cornell University CIS. (cited on page 22)

[55] Felzenszwalb, P., McAllester, D., and Ramanan, D. (2008). A discriminatively trained, multiscale, deformable part model. In *Proc. IEEE Intern. Conf. on Computer Vision and Pattern Recognition*. (cited on pages 13, 14, 16, 17, and 22)

[56] Felzenszwalb, P. F., Girshick, R. B., McAllester, D., and Ramanan, D. (2010). Object detection with discriminatively trained part-based models. *IEEE Trans. on PAMI*, 32:1627–1645. (cited on pages 9, 11, 29, 44, 75, and 76)

[57] Felzenszwalb, P. F. and Huttenlocher, D. P. (2004b). Efficient graph-based image segmentation. *Intern. Journal of Computer Vision*, 59(2). (cited on pages 7, 16, and 22)

[58] Felzenszwalb, P. F. and Huttenlocher, D. P. (2005). Pictorial structures for object recognition. *Intern. Journal of Computer Vision*, 61:55–79. (cited on pages 14, 15, and 21)

[59] Fergus, R., Perona, P., and Zisserman, A. (2003). Object class recognition by unsupervised scale-invariant learning. In *Proc. IEEE Intern. Conf. on Computer Vision and Pattern Recognition*. (cited on page 14)

[60] Fergus, R., Perona, P., and Zisserman, A. (2005). A sparse object category model for efficient learning and exhaustive recognition. In *Proc. IEEE Intern. Conf. on Computer Vision and Pattern Recognition*. (cited on page 14)

[61] Ferrari, V., Fevrier, L., Jurie, F., and Schmid, C. (2008). Groups of adjacent contour segments for object detection. *IEEE Trans. on PAMI*, 30(1):36–51. (cited on page 8)

[62] Ferrari, V., Jurie, F., and Schmid, C. (2010). From images to shape models for object detection. *Intern. Journal of Computer Vision*, 87(3):284–300. (cited on page 8)

[63] Ferrari, V., Tuytelaars, T., and Gool, L. (2006). Object detection by contour segment networks. In *Proc. European Conf. on Computer Vision*. (cited on page 8)

[64] Ferryman, J. and Shahrokni, A. (2009). PETS2009: Dataset and challenge. In *Proc. IEEE Workshop on PETS*. (cited on page 52)

[65] Fischler, M. and Elschlager, R. (1973). The representation and matching of pictorial structures. *IEEE Trans. on Computers*, C-22(1):67 – 92. (cited on pages 14, 15, and 20)

[66] Gall, J. and Lempitsky, V. (2009). Class-specific Hough forests for object detection. In *Proc. IEEE Intern. Conf. on Computer Vision and Pattern Recognition*. (cited on pages 3, 9, 14, 16, 20, 27, 29, 39, 45, 47, 59, 70, 81, 85, 86, 87, 88, 96, 102, and 103)

[67] Guillaumin, M., Verbeek, J., and Schmid, C. (2009). Is that you? metric learning approaches for face identification. In *Proc. IEEE Intern. Conf. on Computer Vision*. (cited on page 10)

[68] Harris, C. and Stephens, M. (1988). A combined corner and edge detector. In *Proc. Alvey Vision Conf.* (cited on page 7)

[69] Ho, T. K. (1995). Random decision forests. In *Proc. Int. Conf. on Document Analysis and Recognition*, volume 1, pages 278–282. (cited on page 28)

[70] Ho, T. K. (1998). The random subspace method for constructing decision forests. *IEEE Trans. on PAMI*, 20(8):832–844. (cited on page 28)

[71] Hua, G., Brown, M., and Winder, S. (2007). Discriminant embedding for local image descriptors. In *Proc. IEEE Intern. Conf. on Computer Vision*. (cited on page 10)

[72] Huang, C., Ai, H., Lao, S., and Li, Y. (2007). High-performance rotation invariant multiview face detection. *IEEE Trans. on PAMI*, 29(4):671–686. (cited on page 18)

[73] Huang, J., Zhang, T., and Metaxas, D. (2011). Learning with structured sparsity. *Journal of Machine Learning Research*, 12:3371–3412. (cited on page 11)

[74] Hubel, D. H. and Wiesel, T. N. (1959). Receptive fields of single neurones in the cat's striate cortex. *Journal of Physiology*, 148(3):574–591. (cited on page 9)

[75] Jain, V. and Learned-Miller, E. (2010). FDDB: A benchmark for face detection in unconstrained settings. Technical Report UM-CS-2010-009, University of Massachusetts, Amherst. (cited on page 52)

[76] Jain, V. and Learned-Miller, E. (2011). Online domain adaptation of a pre-trained cascade of classifiers. In *Proc. IEEE Intern. Conf. on Computer Vision and Pattern Recognition*. (cited on page 76)

[77] Jegou, H., Douze, M., Schmid, C., and Perez, P. (2010). Aggregating local descriptors into a compact image representation. In *Proc. IEEE Intern. Conf. on Computer Vision and Pattern Recognition*. (cited on page 10)

[78] Jenatton, R., Mairal, J., Obozinski, G., and Bach, F. (2011). Proximal methods for hierarchical sparse coding. *Journal of Machine Learning Research*, 12:2297–2334. (cited on page 11)

[79] Joachims, T., Finley, T., and Yu, C.-N. J. (2009). Cutting-plane training of structural svms. *Machine Learning*, 77(1):27–59. (cited on page 19)

[80] Kadir, T. and Brady, M. (2003). Scale saliency: A novel approach to salient feature and scale selection. In *Proc. Int'l Conf. on Visual Information Engineering*, pages 25–28. (cited on page 6)

[81] Kalal, Z., Mikolajczyk, K., and Matas, J. (2010). Face-TLD: Tracking-learning-detection applied to faces. In *Proc. Intern. Conf. on Image Processing*. (cited on page 76)

[82] Kirby, M. and Sirovich, L. (1990). Application of the karhunen-loeve procedure for the characterization of human faces. *IEEE Trans. on PAMI*, 12(1):103–108. (cited on page 19)

[83] Koestinger, M., Hirzer, M., Wohlhart, P., Roth, P. M., and Bischof, H. (2012a). Large scale metric learning from equivalence constraints. In *Proc. IEEE Intern. Conf. on Computer Vision and Pattern Recognition*. (cited on page 10)

[84] Koestinger, M., Wohlhart, P., Roth, P. M., and Bischof, H. (2011). Annotated facial landmarks in the wild: A large-scale, real-world database for facial landmark localization. In *Proc. IEEE Intern. Conf. on Computer Vision Workshops*. (cited on page 53)

[85] Koestinger, M., Wohlhart, P., Roth, P. M., and Bischof, H. (2012b). Robust face detection by simple means. In *Proc. DAGM Symposium Workshops*. (cited on pages 54 and 76)

[86] Kolmogorov, V. and Zabin, R. (2004). What energy functions can be minimized via graph cuts? *IEEE Trans. on PAMI*, 26(2):147–159. (cited on page 98)

[87] Kontschieder, P., Riemenschneider, H., Donoser, M., and Bischof, H. (2011). Discriminative learning of contour fragments for object detection. In *Proc. British Machine Vision Conf.* (cited on page 8)

[88] Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems*, pages 1106–1114. (cited on pages 10 and 20)

[89] Lampert, C. H., Blaschko, M. B., and Hofmann, T. (2009). Efficient subwindow search: A branch and bound framework for object localization. *IEEE Trans. on PAMI*, 31(12):2129–2142. (cited on page 15)

[90] Lazebnik, S., Schmid, C., and Ponce, J. (2006). Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In *Proc. IEEE Intern. Conf. on Computer Vision and Pattern Recognition*. (cited on pages 8 and 12)

[91] Lehmann, A., Leibe, B., and van Gool, L. (2011). Fast PRISM: Branch and bound hough transform for object class detection. *Int. J. Computer Vision*, 94(2):175–197. (cited on pages 16, 21, 23, 24, and 80)

[92] Leibe, B., Cornelis, N., Cornelis, K., and van Gool, L. (2007). Dynamic 3D scene analysis from a moving vehicle. In *Proc. IEEE Intern. Conf. on Computer Vision and Pattern Recognition*. (cited on pages 40, 50, and 86)

[93] Leibe, B., Leonardis, A., and Schiele, B. (2004). Combined object categorization and segmentation with an implicit shape model. In *Proc. European Conf. on Computer Vision*. (cited on pages 3, 14, 16, and 27)

[94] Leibe, B., Leonardis, A., and Schiele, B. (2008). Robust object detection with interleaved categorization and segmentation. *Intern. Journal of Computer Vision*, 77(1–3):259–289. (cited on pages 14 and 16)

[95] Lepetit, V. and Fua, P. (2006). Keypoint recognition using randomized trees. *IEEE Trans. on PAMI*, 28(9):1465–1479. (cited on page 9)

[96] Leutenegger, S., Chli, M., and Siegwart, R. (2011). BRISK: Binary robust invariant scalable keypoints. In *Proc. IEEE Intern. Conf. on Computer Vision*. (cited on page 9)

[97] Li, J., Wang, T., and Zhang, Y. (2011). Face detection using surf cascade. In *Proc. IEEE Intern. Conf. on Computer Vision Workshops*. (cited on pages 76 and 77)

[98] Lim, J. J., Zitnick, L. C., and Dollár, P. (2013). Sketch tokens: A learned mid-level representation for contour and object detection. In *Proc. IEEE Intern. Conf. on Computer Vision and Pattern Recognition*. (cited on pages 7 and 41)

[99] Lowe, D. G. (2004). Distinctive image features from scale-invariant keypoints. *Intern. Journal of Computer Vision*, 60(2):91–110. (cited on pages 7 and 9)

[100] Mairal, J., Bach, F., Ponce, J., Sapiro, G., and Zisserman, A. (2008). Discriminative learned dictionaries for local image analysis. In *Proc. IEEE Intern. Conf. on Computer Vision and Pattern Recognition*. (cited on page 11)

[101] Maji, S. and Berg, A. C. (2009). Max-margin additive classifiers for detection. In *Proc. IEEE Intern. Conf. on Computer Vision*. (cited on pages 8, 80, and 82)

[102] Maji, S. and Malik, J. (2009). Object detection using a max-margin hough transform. In *Proc. IEEE Intern. Conf. on Computer Vision and Pattern Recognition*. (cited on page 25)

[103] Malisiewicz, T., Gupta, A., and Efros, A. A. (2011). Ensemble of exemplar-svms for object detection and beyond. In *Proc. IEEE Intern. Conf. on Computer Vision*. (cited on page 18)

[104] Matas, J., Chum, O., Urban, M., and Pajdla, T. (2002). Robust wide baseline stereo from maximally stable extremal regions. In *Proc. British Machine Vision Conf.* (cited on page 8)

[105] Mensink, T., Verbeek, J., Perronnin, F., and Csurka, G. (2012). Metric Learning for Large Scale Image Classification: Generalizing to New Classes at Near-Zero Cost. In *Proc. European Conf. on Computer Vision*. (cited on page 10)

[106] Menze, B., Kelm, B., Splitthoff, D., Koethe, U., and Hamprecht, F. (2011). On oblique random forests. In *Proc. European Conf. on Machine Learning and Knowledge Discovery in Databases*. (cited on pages 29 and 30)

[107] Mikolajczyk, K., Schmid, C., and Zisserman, A. (2004). Human detection based on a probabilistic assembly of robust detectors. In *Proc. European Conf. on Computer Vision*. (cited on page 76)

[108] Mikolajczyk, K., Tuytelaars, T., Schmid, C., Zisserman, A., Matas, J., Schaffalitzky, F., Kadir, T., and Gool, L. V. (2005). A comparison of affine region detectors. *Intern. Journal of Computer Vision*, 65(1/2):43–72. (cited on page 7)

[109] Mutch, J. and Lowe, D. G. (2008). Object class recognition and localization using sparse features with limited receptive fields. *Intern. Journal of Computer Vision*, 80(1):45–57. (cited on page 9)

[110] Nister, D. and Stewenius, H. (2006). Scalable recognition with a vocabulary tree. In *Proc. IEEE Intern. Conf. on Computer Vision and Pattern Recognition*. (cited on page 10)

[111] Nowozin, S. (2012). Improved information gain estimates for decision tree induction. In *Proc. IEEE Intern. Conf. on Machine Learning*. (cited on page 31)

[112] Ojala, T., Pietikäinen, M., and Mäenpää, T. (2002). Multiresolution gray-scale and rotation invariant texture classification ith local binary patterns. *IEEE Trans. on PAMI*, 24(7):971–987. (cited on page 9)

[113] Okada, R. (2009). Discriminative generalized hough transform for object dectection. In *Proc. IEEE Intern. Conf. on Computer Vision*. (cited on page 30)

[114] Opelt, A., Pinz, A., and Zisserman, A. (2006). A boundary-fragment-model for object detection. In *Proc. European Conf. on Computer Vision*. (cited on pages 8 and 16)

[115] Oren, M., Papageorgiou, C., Sinha, P., Osuna, E., and Poggio, T. (1997). Pedestrian detection using wavelet templates. In *Proc. IEEE Intern. Conf. on Computer Vision and Pattern Recognition*. (cited on page 30)

[116] Papageorgiou, C., Oren, M., and Poggio, T. (1998). A general framework for object detection. In *Proc. IEEE Intern. Conf. on Computer Vision*. (cited on page 30)

[117] Perronnin, F. and Dance, C. (2007). Fisher kernels on visual vocabularies for image categorization. In *Proc. IEEE Intern. Conf. on Computer Vision and Pattern Recognition*. (cited on page 10)

[118] Philbin, J., Isard, M., Sivic, J., and Zisserman, A. (2010). Descriptor learning for efficient retrieval. In *Proc. European Conf. on Computer Vision*. (cited on page 10)

[119] Pietikainen, M., Hadid, A., Zhao, G., and Ahonen, T. (2011). *Computer Vision Using Local Binary Patterns*. Springer. (cited on page 9)

[120] Pirsiavash, H. and Ramanan, D. (2012). Steerable part models. In *Proc. IEEE Intern. Conf. on Computer Vision and Pattern Recognition*. (cited on page 11)

[121] Rahtu, E., Kannala, J., and Blaschko, M. (2011). Learning a category independent object detection cascade. In *Proc. IEEE Intern. Conf. on Computer Vision*. (cited on page 7)

[122] Ravishankar, S., Jain, A., and Mittal, A. (2008). Multi-stage contour based detection of deformable objects. In *Proc. European Conf. on Computer Vision*. (cited on page 8)

[123] Razavi, N., Gall, J., and Kohli, Pushmeet Gool, L. v. (2012). Latent hough transform for object detection. In *Proc. European Conf. on Computer Vision*. (cited on pages 18 and 25)

[124] Razavi, N., Gall, J., and van Gool, L. (2010). Backprojection revisited: Scalable multi-view object detection and similarity metrics for detections. In *Proc. European Conf. on Computer Vision*. (cited on pages 18, 80, 84, and 88)

[125] Riemenschneider, H., Donoser, M., and Bischof, H. (2010). Using partial edge contour matches for efficient object category localization. In *Proc. European Conf. on Computer Vision*. (cited on page 8)

[126] Riemenschneider, H., Sternig, S., Donoser, M., Roth, P. M., and Bischof, H. (2012). Hough regions for joining instance localization and segmentation. In *Proc. European Conf. on Computer Vision*. (cited on page 52)

[127] Rosten, E. and Drummond, T. (2005). Fusing points and lines for high performance tracking. In *Proc. IEEE Intern. Conf. on Computer Vision*. (cited on page 7)

[128] Rosten, E. and Drummond, T. (2006). Machine learning for high-speed corner detection. In *Proc. European Conf. on Computer Vision*. (cited on page 7)

[129] Schölkopf, B. and Smola, A. J. (2002). *Learning with Kernels*. MIT Press. (cited on page 19)

[130] Schulter, S., Leistner, C., Roth, P. M., Van Gool, L., and Bischof, H. (2011). On-line hough forests. In *Proc. British Machine Vision Conf.* (cited on page 42)

[131] Schulter, S., Leistner, C., Wohlhart, P., Roth, P. M., and Bischof, H. (2013a). Alternating regression forests for object detection and pose estimation. In *Proc. IEEE Intern. Conf. on Computer Vision*. (cited on page 20)

[132] Schulter, S., Roth, P. M., and Bischof, H. (2013b). Ordinal random forests for object detection. In *Proc. German Conf. on Pattern Recognition*. (cited on page 30)

[133] Sermanet, P., Kavukcuoglu, K., Chintala, S., and Lecun, Y. (2013). Pedestrian detection with unsupervised multi-stage feature learning. In *Proc. IEEE Intern. Conf. on Computer Vision and Pattern Recognition*. (cited on page 10)

[134] Serre, T., Oliva, A., and Poggio, T. (2007). A feedforward architecture accounts for rapid categorization. *Proceedings of the National Academy of Sciences*, 104(15):6424–6429. (cited on page 9)

[135] Shakhnarovich, G. (2006). *Learning Task-Specific Similarity*. PhD thesis, Massachusetts Institute of Technology. (cited on page 10)

[136] Shi, J. and Malik, J. (2000). Normalized cuts and image segmentation. *IEEE Trans. on PAMI*, 22(8):888–905. (cited on page 7)

[137] Shi, J. and Tomasi, C. (1994). Good features to track. In *Proc. IEEE Intern. Conf. on Computer Vision and Pattern Recognition*. (cited on page 7)

[138] Shotton, J., Blake, A., and Cipolla, R. (2008). Multi-scale categorical object recognition using contour fragments. *IEEE Trans. on PAMI*, 30(7):1270–1281. (cited on page 8)

[139] Shotton, J., Fitzgibbon, A. W., Cook, M., Sharp, T., Finocchio, M., Moore, R., Kipman, A., and Blake, A. (2011). Real-time human pose recognition in parts from single depth images. In *Proc. IEEE Intern. Conf. on Computer Vision and Pattern Recognition*. (cited on page 66)

[140] Shotton, J., Girshick, R., Fitzgibbon, A., Sharp, T., Cook, M., Finocchio, M., Moore, R., Kohli, P., Criminisi, A., Kipman, A., and Blake, A. (2013). Efficient human pose estimation from single depth images. *IEEE Trans. on PAMI*, 35(12):2821–2840. (cited on pages 38, 49, 59, and 66)

[141] Sivic, J. and Zisserman, A. (2003). Video Google: A text retrieval approach to object matching in videos. In *Proc. IEEE Intern. Conf. on Computer Vision*. (cited on page 10)

[142] Song, H. O., Zickler, S., Althoff, T., Girshick, R. B., Fritz, M., Geyer, C., Felzenszwalb, P. F., and Darrell, T. (2012). Sparselet models for efficient multiclass object detection. In *Proc. European Conf. on Computer Vision*. (cited on page 11)

[143] Strecha, C., Bronstein, A. M., Bronstein, M. M., and Fua, P. (2012). LDAHash: Improved matching with smaller descriptors. *IEEE Trans. on PAMI*, 34(1). (cited on page 10)

[144] Subburaman, V. B. and Marcel, S. (2010). Fast bounding box estimation based face detection. In *Proc. European Conf. on Computer Vision Workshops*. (cited on page 76)

[145] Tieu, K. and Viola, P. (2000). Boosting image retrieval. In *Proc. IEEE Intern. Conf. on Computer Vision and Pattern Recognition*, pages 228–235. (cited on page 20)

[146] Trzcinski, T., Christoudias, C. M., Lepetit, V., and Fua, P. (2012). Learning Image Descriptors with the Boosting-Trick. In *Advances in Neural Information Processing Systems*. (cited on page 10)

[147] Trzcinski, T., Christoudias, M., Lepetit, V., and Fua, P. (2013). Boosting binary keypoint descriptors. In *Proc. IEEE Intern. Conf. on Computer Vision and Pattern Recognition*. (cited on pages 9 and 10)

[148] Tsochantaridis, I., Hofmann, T., Joachims, T., and Altun, Y. (2004). Support vector machine learning for interdependent and structured output spaces. In *Proc. IEEE Intern. Conf. on Machine Learning*. (cited on page 19)

[149] Turk, M. and Pentland, A. (1991). Face recognition using eigenfaces. In *Proc. IEEE Intern. Conf. on Computer Vision and Pattern Recognition*. (cited on page 19)

[150] Tuytelaars, T. and Mikolajczyk, K. (2008). Local invariant feature detectors: A survey. *Found. Trends. Comput. Graph. Vis.*, 3(3):177–280. (cited on page 7)

[151] Uijlings, J. R. R., van de Sande, K. E. A., Gevers, T., and Smeulders, A. W. M. (2013). Selective search for object recognition. *International Journal of Computer Vision*, 104(2):154–171. (cited on page 7)

[152] Vapnik, V. N. (1995). *The Nature of Statistical Learning Theory*. Springer. (cited on page 19)

[153] Vedaldi, A. and Zisserman, A. (2010). Efficient additive kernels via explicit feature maps. In *Proc. IEEE Intern. Conf. on Computer Vision and Pattern Recognition*. (cited on page 84)

[154] Veksler, O., Boykov, Y., and Mehrani, P. (2010). Superpixels and supervoxels in an energy optimization framework. In *Proc. European Conf. on Computer Vision*. (cited on page 7)

[155] Viola, P. and Jones, M. (2001). Rapid object detection using a boosted cascade of simple features. In *Proc. IEEE Intern. Conf. on Computer Vision and Pattern Recognition*. (cited on pages 8, 18, 20, and 44)

[156] Viola, P. and Jones, M. (2004). Robust real-time face detection. *Intern. Journal of Computer Vision*, 57:137–154. (cited on pages 8, 20, 30, and 76)

[157] Walk, S., Majer, N., Schindler, K., and Schiele, B. (2010). New features and insights for pedestrian detection. In *Proc. IEEE Intern. Conf. on Computer Vision and Pattern Recognition*. (cited on page 54)

[158] Wang, T., Wu, D., Coates, A., and Ng, A. (2012). End-to-end text recognition with convolutional neural networks. In *Proc. Intern. Conf. on Pattern Recognition*. (cited on page 10)

[159] Yang, J., Yu, K., Gong, Y., and Huang, T. (2009). Linear spatial pyramid matching using sparse coding for image classification. In *Proc. IEEE Intern. Conf. on Computer Vision and Pattern Recognition*. (cited on page 11)

[160] Zabih, R. and Woodfill, J. (1994). Non-parametric local transforms for computing visual correspondence. In *Proc. European Conf. on Computer Vision*. Springer Berlin Heidelberg. (cited on page 9)

[161] Zhang, J., Marszalek, M., Lazebnik, S., and Schmid, C. (2007). Local features and kernels for classification of texture and object categories: A comprehensive study. *Intern. Journal of Computer Vision*, 73(2):213–238. (cited on page 9)

[162] Zhu, X. and Ramanan, D. (2012). Face detection, pose estimation and landmark estimation in the wild. In *Proc. IEEE Intern. Conf. on Computer Vision and Pattern Recognition*. (cited on pages 21, 76, and 77)