TECHNISCHE UNIVERSITÄT
CHEMNITZ

Research Seminar
Computer Science
MSc Automotive Software Engineering

SEMINAR REPORT

Pedestrian Detection using HOG and SVM in Automotives

Chair of Computer Engineering
Dept. of Computer Science
TU Chemnitz

Submitted by :	Surajit Dutta
Mtr. No. :	334439
DOB :	Feb 9, 1989
Address :	Bürstergasse 16
	88131 Lindau
	Germany
Supervising tutor :	Prof.Dr. Wolfram Hardt

March 2015

Abstract

This technical report describes the problem of pedestrian detection using Histograms of Oriented Gradients in combination with linear Support Vector Machines. This particular pairing has, for years, shown beneficial performance over other methods approaching similar problems. This report aims to explain why this is the case by explaining both methods. In addition, we outline why this particular combination of algorithms is specifically useful in the automotive setting and may support function such as pedestrian detection, which eventually may lead to automatic breaking assist functionality in a car.

We provide empirical results of the described approach in form of qualitative and quantitative evaluation on two different datasets.

PREFACE

This report was mainly written focusing on one of three individual research topics during an internship at the Automotive Distance Control Systems (A.D.C. GmbH), a sub-company of the Continental AG.

While I have learned most of the concepts presented in this report at the A.D.C. GmbH, I hereby ensure that the presented material only coarsely reflects the approach the A.D.C. GmbH is taking to pedestrian detection. I want to highlight that the presented algorithms and optimization (in software and hardware), do not reflect the approach of the A.D.C. GmbH to pedestrian detection.

Where possibly, alternative approaches are discussed. Unfortunately, details on the software implementation and hardware platforms at the A.D.C. GmbH can not be provided due to the obvious reason that competing companies could “gain an edge” on such information. However, the principle ideas behind the implementation and other possibilities of realization are briefly mentioned.

CONTENTS

List of Figures

List of Tables

Chapter 1
Introduction

In the past few decades, the computer vision world has experienced an ample amount of improvement in the research area of object detection. A key-contribution in this area was made by Viola & Jones in their CVPR paper of the year 2001 [1], which was – without a doubt correctly – awarded the Longuet-Higgins Prize for fundamental contributions in computer vision. Other key contributions are listed in Chapter 2, which, within the limits of this report, covers related work in the area of object detection.

The paper of Viola & Jones very successfully coupled two methods, which when combined are known as the sliding window detector. The idea is to slide a window of fixed size over an image. Once this process is finished, the window or the image are re-scaled and the processes is repeated again up until the window does not fit the image anymore. This ensures at objects of multiple sizes can be detected.

In case of the sliding window detector, an algorithm extracts an image representation from the underlying image content. Such an image representation can be understood as translating the pure pixel values into a "language" the computer can efficiently 'work' with. This work, in the case of the sliding window detector, is done by a classifier. Viola & Jones used AdaBoost [2], an efficient cascaded classification architecture which quickly rejects windows that do not contain objects of interest (in their case faces of people) and retains true-positive detection (actual faces).

The choices regarding image representation and classification architecture are a testimonial to the time – computers were simply not as fast as they are today [3]. The image representation used by Viola & Jones are Haar features – in short such features are subtractions of the sums of grey-scale pixel values of neighboring rectangular areas. These areas can be efficiently computed in constant time using a technique called integral images [4, 5]. The fast rejection rate of true-negative windows of the AdaBoost classification scheme gave an additional boost, which allowed for fast face detection at the time. Today this algorithm is used in many digital cameras to perform face detection [6].

In 2005 Dalal & Triggs [7] introduced HOG features - Histograms of Oriented Gradient. These features gave superior performance over the previously used feature sets, i.e. SIFT descriptor [8] and Haar features [9]. The reason for this will be explained in detail in Chapter 3 of this report.

Dalal & Triggs coupled the HOG features with linear Support Vector Machines (SVM), which were mainly used for their good classification accuracy and linear complexity at classification time. Modern solvers to the linear SVM problem also enable linear or even sub-linear training time [10, 11]. Chapter 4 more closely examines these classifier architectures.

Chapter 5 present qualitative and quantitative results of "HOG+SVM" approach. We will present results on the INRIA Person dataset [7] used by Dalal & Triggs and, in addition, present results on the A.D.C. dataset (dubbed the "ADAS dataset"), which was kindly provided by Continental for this report.

Chapter 6 will summarize the report and also provide an outlook on future work by combining the inspiring thoughts of the multiple research directions object detection has taken.

Chapter 2
Related Work

Due to the constrained amount of space in this report, this chapter aims to provide an overview of the key-publications in the area of pedestrian detection and, where necessary, object detection in general. The interested reader is pointed towards Dollar *et al.* [12] for a comprehensive overview of pedestrian detection methods.

Pioneering Work. Indeed, two of the pioneering approaches to pedestrian detection originate in the automotive domain. Gavrila & Philomin [13] approached the challenging problem of real-time detection using Chamfer matching, i.e. the matching of shapes using a novel variant of the Distance Transform function, which was later employed by Felzenszwalb & Huttenlocher [14] to make inference in part-based models for object detection efficient. A wavelet representation of a window (i.e. Haar-Features) was used by Papageorgiou and Poggio [15] in combination with a Support Vector Machine [16]. Interestingly enough this approach was also aimed to be used for a pedestrian detection system in car. Clearly, the usage of two separate approaches early in the development of object detection, in general emphasizes the need of the industry for such systems. This is one is reason for this report, i.e. to highlight the aspects of the described approach in the automotive setting.

Haar-Features (and other wavelet representations) are essentially a move away from the simple grayscale image pixels, which have been used previously but will not be covered in this report due to the lack of space and the unnecessary reflection of these pieces of work w.r.t the scope of this report. Nevertheless these initial papers were the starting point of object detection and should be treated with the foremost respect – for example we refer the interested reader to Sirovich & Kirby [17] or the approach towards pose estimation of Hogg [18].

SIFT & HOG. Haar-Features grouped adjacent pixels into cells and this idea, i.e. the grouping of neighboring pixels, has eversince been a part of many approaches to object detection. Histograms of Oriented Gradient (HOG) [7] are a perfect example of that and will be explained in Chapter 3. HOG features are essentially an extension to SIFT-Features [8] which compute local histograms of gradient orientation in a set of square cells. However, they lack a block-normalization scheme as proposed by Dalal & Triggs (see Chapter 3). Such a normalization scheme can improve detection accuracy significantly by providing additional contrast invariance c.f.SIFT.

Extensions of HOG. Most outstanding work on feature engineering has at this point focused on either (i) extending the HOG feature extractor or (ii) combining it with other feature types. One piece of work in the first category was presented by Ott & Everingham [19] who combined segmentation information with the HOG feature descriptor in an efficient way. The segmentation was obtained from colored areas in the vicinity of local HOG features.

The method significantly improved detection performance. Other methods, such as the one of Felzenszwalb *et al.* [20] removed some of the original shortcomings of the HOG features (e.g. differentiating between gradient orientations, which lie on the opposite of the unit circle). The second category is primarily dominated by work which has combined other popular feature extraction schemes with the HOG descriptor, such as the work of Schwartz *et al.* [21] who have added texture and color information to the feature vector and applied Partial Least Squares analysis [22] as a dimensionality reduction technique to reduce the number of features. Similar to Ott & Everingham [19], the approach was then coupled with an non-linear SVM and yielded a significant improvement over using HOG features alone.

Covariance Matrices as Features. A rather exotic approach to the problem was proposed by Tuzel *et al.* [23]. In this approach covariance matrices are computed over squared regions of pixels and other feature channels (such as derivatives of the original image). These covariance matrices do not lie in the real number space but on a Riemannian manifold and Tuzel *et al.* propose an adapted LogitBoost [24] classifier (an adaptation of the original AdaBoost classifier [2]) to perform classification on these manifolds. The method proved successful and improved results over the most successful approach of that time [7], which is covered in this report. However, it unfortunately never opened up a new research direction.

Kernel SVMs and Approximations theorem. Rather than focusing on the feature representation, many researchers have focused on the classification scheme. As mentioned previously HOG is usually combined with a linear Support Vector Machine due to its favorable linear complexity $\mathcal{O}(n)$ – here n represents the feature dimensionality. Some work has replaced the linear SVM with a non-linear Kernel SVM using a Polynomial Kernel – as mentioned earlier this was e.g. considered by Ott & Everingham [19] [19] and Schwartz *et al.* [21]. This of course leads to a dramatic increase in computational processing due to the necessary computations of the dot-product with the support vectors. Assuming m is the number of support vectors, $\mathcal{O}(mn)$ is the lower bound on the kernel SVMs complexity (additional computations for each kernel might be necessary). A short description of the kernel SVM will be given in Chapter 4.

As a result research has also focused on enabling non-linear classification in an efficient manner. From the various pieces of work, two stand out in particular. Vedaldi & Zisserman [25] proposed a method to approximate the function that lifts the feature vector to the Reproducing Kernel Hilbert Space of SVMs with additive kernels, essentially defining an explicit mapping. Using the explicit mapping, a linear SVM is used to train on the explicitly lifted feature vectors. The method performs very well and requires small changes to the code. In addition, the dimensionality of the lifted feature vector usually only increases by a factor of three and hence computational complexity is still manageable.

Part-based Models. A promising research direction was opened up by discriminatively modeling an object as an assembly of parts. Among the first pieces of work in this area was presented by Mohan *et al.* [26] who learned separate non-linear SVM classifiers for

different body parts and combined their score using a linear SVM. Felzenszwalb *et al.* [20] proposed a method that decomposes the appearance of an object into a set of agile parts, which can move around given a spatial log-gaussian prior, which acts as a penalty term if the part moves too far away from its mean position. The parts motivation to move away from this position is given by the fact that they might score higher at another position. Hence, inference in this scheme is performed by maximizing the score of each window by moving the parts. Since the part-based model builds on a tree-based structure (one root node, each part connected to the root node and only to the root node), inference can be made very efficient due to the use of Distance Transforms, which can be efficiently computed using the method of Felzenszwalb & Huttenlocher [14]. Multiple such models for different viewpoints are usually trained in form of a mixture model [27]. The approach that combines mixture models and part decomposition is commonly known as a Deformable Part-based Model (DPM). The underlying image representation to the DPMs are HOG features. One interesting observation here is that HOG features introduce various levels of invariance to the image representation, which will be explained in Chapter 3. However, the decomposition of the model into parts c.f.the rigid model representation of Dalal & Triggs [7] actually "removes" invariance from the overall scheme while improving detection accuracy. The approach of Felzenszwalb *et al.* [20] has attracted much attention in the past, mainly due to the good performance it offers. One addition to the model was part-sharing, first introduced by Ott & Everingham [28]. This model, in particular, offers the possibility to share parts across viewpoints (mixture components) and even object classes. An interesting result of the method is that the authors showed better performing models with less shared parts than individual trained DPMs. DPMs have finally been extended to Object Grammars by Girshick *et al.* [29]. The grammar models reformulate the original DPM formulation into a scheme suited for broader usage, e.g. the possibility to add parts of parts.

A Probabilistic Approach using CRFs. Winn & Shotton propose the "Layout Consistent Rrandom field" (LayoutCRF) model [30], which is also a part-based model. By combining part-detectors in form of randomized decision trees [31] with a conditional random field (CRF) [32], pairwise potentials between the parts, which lie on a grid in the CRF formulation, ensure the proper layout of an object class. Given the CRFs potential to model the probability of an object instance y being present given the appearance of the underlying image content $\boldsymbol{x}$ (modeled by the randomised decision trees), e.g. $\mathrm{P}\left(y|\,\boldsymbol{x}\right)$ allows for explicit modeling of common expectations in object appearance, e.g. occlusion. To realize that, the CRFs unary potentials (represented by the randomized decision trees) will 'fire', i.e. return a high score, in case of the presence of a part but provide a poor score if the part is not present. The binary potentials, which couple the nodes in the CRF in the form of a grid, ensure that common combinations of parts are assigned to the same object instance, i.e. they add a fairly high score to the inference scheme. However, if, e.g. a pedestrian occludes a car and the expected part at the position of the car where the pedestrian enforces the occlusion is not present, a binary potential with a lower score will 'kick in' and allow for the placement of a pedestrian part and essentially model a car-to-pedestrian transition. The model, while being intuitive and outstandingly elegant, is very slow at inference.

Deep Learning. Finally, deep learning approaches [33–35] also form a large and growing group of approaches to the object detection problem. The term "Deep Learning" most often refers to deep neural networks, which in case of images are usually of convolutional nature. These networks are, as the name suggest, deep, i.e. consist of many layers. These do not all need to be convolution layers – common other layers are pooling layers, which combine the responses in a region of the image into an output neuron, mainly to achieve translation invariance. At a certain point the remaining convolutional responses will be 'unrolled' and form the input to the standard neural network architecture [27]. The convolutional filters in the first layer very often represent individual detectors for edges and corners while the lower layers combine these responses into parts of various sizes. Commonly such deep networks were learned by stacking multiple auto-encoder schemes [34, 36] on top of each other, giving rise to a deeper architecture and a fairly efficient learning scheme – each auto-encoder was learned on its own. The reason for this situation stemmed from the fact that the common backpropagation algorithm [37] for training convolutional neural networks (CNNs) usually fails on such deep networks due to reasons of numerical instability, many poor solutions in the search space of the learning problem as well as steep 'cliffs' in this search space. This cliffs upset the commonly used Stochastic Gradient Descent learning scheme. Very recently, methods for training deep CNN in "one go" became attractive again [38], mainly due to a handful of significant modifications of the original CNN approach. Among these are (i) the replacement of the transfer function in each neuron (previously a sigmoidal function) with rectified linear units [39] (ReLUs, e.g. $f(\boldsymbol{x}) = \max(0, \boldsymbol{x})$); (ii) local normalization to the responses of each convolution; (iii) the efficient use of pooling layers (e.g. max-pooling); (iv) harvesting the compute power of modern graphic processors and (v) the utilization of more efficient training methods such as Drop-Out [39]. Recently Girshick *et al.* [40] proposed the R-CNN. This method extracts region proposals from an image (where an object could be located), resizes them to a fixed size and aspect ratio and feeds them into a CNN. Although the method seems somewhat crude, the results are the current state-of-the-art.

Chapter 3
Histogram of Oriented Gradients

Despite of previously mentioned noticeable improvements, both in *feature representation* and *learning algorithms*, pedestrian detection, and object detection in general, remains a challenging task. In case of pedestrian detection, which is what this report is focused on, the reason be the various poses and different appearances (e.g. color of clothing). Therefore pedestrians require a robust yet adaptive feature representation that allows them to be conspicuously distinguished from the background notwithstanding those issues. In addition, the feature representation should subdue the effects caused by the poor illumination conditions or cluttered backgrounds. In such scenarios Histogram of Oriented Gradients features [7] (in short HOG) are a very common choice. Before understanding how this feature descriptor satisfies all aforementioned requirements, one shall understand the contextual definition of *features* in this report. The word 'features' is a short form for "feature representation" or "image representation". These three forms will be used throughout the report, essentially representing a similar meaning.

Image Representation. In the context of pedestrian and object detection, features form a representation of the image (or a part of the image, e.g. a window). This representation should match the above mentioned criteria, i.e. provide various forms of invariance while still being able to accurately describe the pedestrian and distinguish it from the background. This is especially crucial when such features are combined with a discriminative classifier, such as a Support Vector Machine (SVM, discussed in Chapter 4). One obvious first choice would be the pure pixel values of the image. However, using the pixel values as features performs very well at *training* (e.g. teaching) the classifier but performs poorly on unseen pedestrians (*testing*), i.e. the generalization performance is low. An ideal feature descriptor should be robust to the changes in appearance of the object of interest (the pedestrian) while also being able to detect pedestrians it has not been exposed to during learning.

Neither color nor texture are representative enough (on their own) to describe pedestrians reliably as they are exposed to a great deal of variation in appearance of the pedestrian. In addition, cluttered background might upset texture-based features and, when used alone, is often confused with a pedestrian.

Edges, most commonly and easily represented by the gradient of the image, often provide the best representation of a pedestrian as they are invariant to color and partially invariant to texture. When appropriately normalized, edges are also invariant to changes in brightness of the image. Histograms of Oriented Gradient (HOG) build on the gradients of an image and have shown very good performance at the task of pedestrian detection.

3.1 Overview of the Method

This section gives an overview of the method. The following sections will explain each step in much more detail. In combination with Chapter 4 this gives a clear picture of what HOG features are and how their expressive power combined with a classification scheme such as a Support Vector Machine, can yield very good results at the task of pedestrian detection.

Main Idea & Motivation. The central idea of this approach is that local shape or appearance of an object can be distinctively characterized by the distribution of the local intensity gradients and edge directions [7]. Pixels are grouped into cells and the orientation of the gradient of each pixel is computed via the $arctan2\left(\frac{\partial}{\partial y}, \frac{\partial}{\partial x}\right)$ function. The common size of such a cell is, in case of pedestrian detection, 8×8 pixels. Within each cell a histogram over gradient direction is computed, with the magnitude of each pixel-gradient voting into the appropriate bin. Dalal & Triggs [7] used nine bins for the histogram in each cell. In the automotive domain usually eight bins are used for two reasons: (i) it slightly reduces the dimensionality of the feature descriptor of the pedestrian and (ii) allows for efficient use of various intrinsic operations on $x86$-platforms as well as DSPs and additional co-processors.

Blocks of Cells. Adjacent cells are usually grouped into overlapping blocks of size 2×2, yielding $4 \times 8 = 32$ HOGs per block. Each histogram bin in each cell of a block is ℓ_2-normalized w.r.t the other bins of all cells in the block. While other normalization schemes have been explored (e.g. ℓ_1-normalization), this scheme has empirically proven to be the best performing. It adds contrast invariance to the feature descriptor, i.e. a block only consisting of low gradient magnitudes contributes equally to the overall feature vector c.f.a block with strong gradient magnitudes.

The concatenation of all block descriptors is known as the HOG descriptor of a window and is fed to a classification algorithm, e.g. a Support Vector Machine (SVM)) [16]. The previously described sliding window scheme executes a brute-force search at all location and sizes of a window to locate pedestrians in an image. This approach might initially sound cumbersome and time consuming. However, with the use of integral images [41], which will be explained later, and efficient usage of the provided hardware (i.e. pre-loading window information into a buffer while classifying the features of the previous window as well as making efficient use of intrinsic operations) this process can be considered very fast.

Process Chain. Initially HOG features are collected from a set of labeled pedestrians and random background windows. A classifier then learns the *model*, which represents the pedestrian, given the HOG feature information. The overall process chain, excluding the sliding window detector, is further illustrated in Figure 3.1.

3.2 Methodology

This section aims to explain the details of the process of HOG feature extraction and classification.

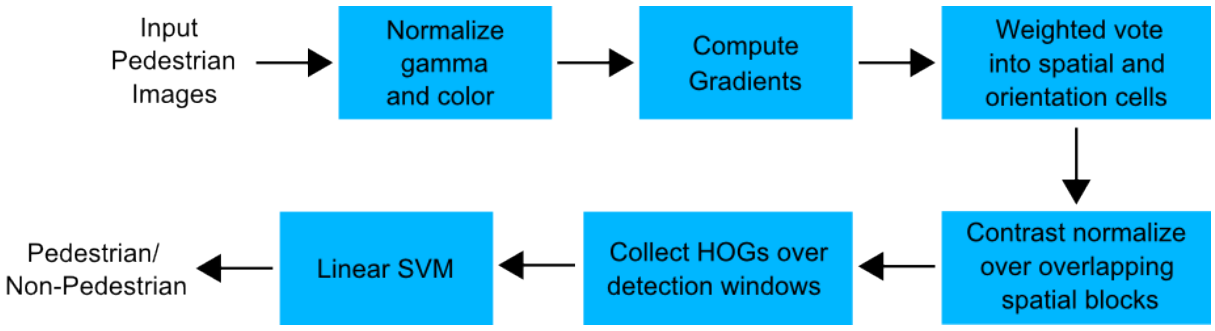


Figure 3.1: Processing pipeline of HOG features and classification. (i) The input images are pre-processed using a color and gamma normalization [7]; (ii) gradients of the image are extracted and according to their orientation vote they vote their magnitude into a bin of a histogram. These histograms are computed locally for a set of cells (normally of size 8×8); (iii) blocks of size 2×2 cells are independently ℓ_2-normalized to acquire contrast-invariance; (iv) the collection of all blocks forms the HOG feature descriptor of a window; (v) this feature vector is finally passed to a classifier, in this case a linear Support Vector Machine, to achieve classification into one of two classes (pedestrian or background).

3.2.1 Gradient Computation

Many ways for computing gradients are available in the literature, e.g. the common Sobel Filter [42]. One might even consider more sophisticated methods for edge extraction, e.g. the approach of Dollár *et al.* [43] using random-forests with structured output for extraction of edges and apply the remaining processing chain of HOG features on these outputs.

Differential Filters and Gradient Extraction. However, Dalal & Triggs determined experimentally that the differential filters $\begin{bmatrix}-1 & 0 & 1\end{bmatrix}$ and $\begin{bmatrix}-1 & 0 & 1\end{bmatrix}^T$ for x- and y- direction respectively perform best for HOG features. Their simple form makes them very fast during convolution with the image c.f.more sophisticated techniques which might convolve the image with a kernel of larger size. Let $*$ denote the convolutional operator, then the derivatives of a gray-scale image I are given by

$$\frac{\partial I}{\partial x} = \begin{bmatrix} -1 & 0 & 1 \end{bmatrix} * I \qquad \frac{\partial I}{\partial y} = \begin{bmatrix} -1 & 0 & 1 \end{bmatrix}^T * I \tag{3.1}$$

In case of color images, the gradient of the color channel with the largest magnitude is taken.

Gradient Orientation and Magnitude. As previously mentioned the gradients are then passed to the $\operatorname{atan2}(\cdot,\cdot)$ function to provide their orientation, i.e.

$$\theta = \operatorname{atan2}\left(\frac{\partial I}{\partial y}, \frac{\partial I}{\partial x}\right) \tag{3.2}$$

The $\operatorname{atan2}(\cdot,\cdot)$ function provides output on a unit circle, i.e. covering the full range of $360°$. This is often referred to as the *signed gradient* as it can distinguish between dark-to-bright and bright-to-dark transitions along an edge. For pedestrians this might not be the most useful feature, given the different types of clothing and the various backgrounds pedestrians appear

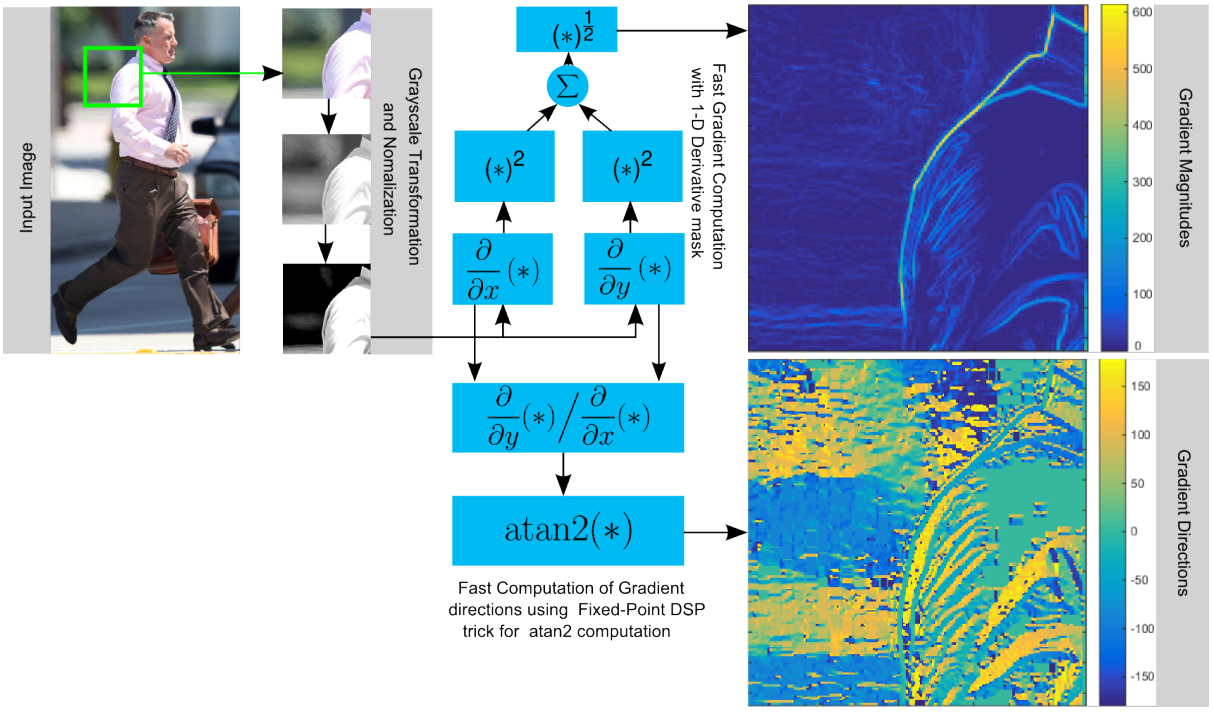


Figure 3.2: Computation of gradient, gradient magnitude and orientation: (i) 1-D centered derivative filters i.e. $[-1 \quad 0 \quad 1]$ and $[-1 \quad 0 \quad 1]^T$ are convolved with the preprocessed input image to obtain derivatives along x and y directions respectively; (ii) the magnitude of each gradient at each pixel position is computed via $|\nabla I| = \sqrt{\left(\frac{\partial I}{\partial x}\right)^2 + \left(\frac{\partial I}{\partial y}\right)^2}$. (iii) orientation of of the gradients are computed by $\theta = \text{atan2}\left(\frac{\partial I}{\partial y}, \frac{\partial I}{\partial x}\right)$.

in front of. In this case we consider the *unsigned gradient*, which "flips" the orientation of the gradient by $180°$ provided the original orientation exceeds $180°$. This form of orientation calculation introduces an additional level of invariance by not being able to distinguish edges in the way signed gradients are able to.

In addition, the magnitude of each gradient at each pixel location is computed by

$$|\nabla I| = \sqrt{\left(\frac{\partial I}{\partial x}\right)^2 + \left(\frac{\partial I}{\partial y}\right)^2} \tag{3.3}$$

The process of gradient extraction, orientation and magnitude computation is visualized in Figure 3.2.

3.2.2 Orientation Binning

Histograms over Gradient Orientation. This stage involves creating the cells of histograms with 8 bins for each cell. Every pixel within the cell casts a weighted vote for an orientation-based histogram channel depending on the gradient responses, i.e. in which direction the gradient points. The cells themselves are usually rectangular (normally 8×8 pixels) and the histogram channels are uniformly spaced over $0°$-$180°$ (for unsigned gradients) or $0°$-$360°$ (for

signed gradients). Signed gradients allow them to model the difference of a black-to-white and white-to-block transition in the image while unsigned gradients will treat them the same and add an additional level of invariance. While for pedestrians unsigned gradients are very useful, given their difference in clothing and the various backgrounds they might appear in front of, other object classes might benefit from this information.

Weighted Voting into Histogram Bins. In this research we use unsigned gradients as proposed by Dalal & Triggs [7] as it prioritizes only the edges to understand the shape, not the directions of the gradient at the edge. Since, 8 bins of the histograms are equally spaced over orientation domain $0° \leq \theta \leq 180°$, each bin is $20°$ wide (i.e. $0°$-$20°$, $20°$-$40°$, ..., $160°$-$180°$ bins construct one cellular histogram). Each pixel under each cell gives weighted votes (the weight is its gradient magnitude itself) to the corresponding orientation bins in the corresponding histogram as shown in Figure 3.3.

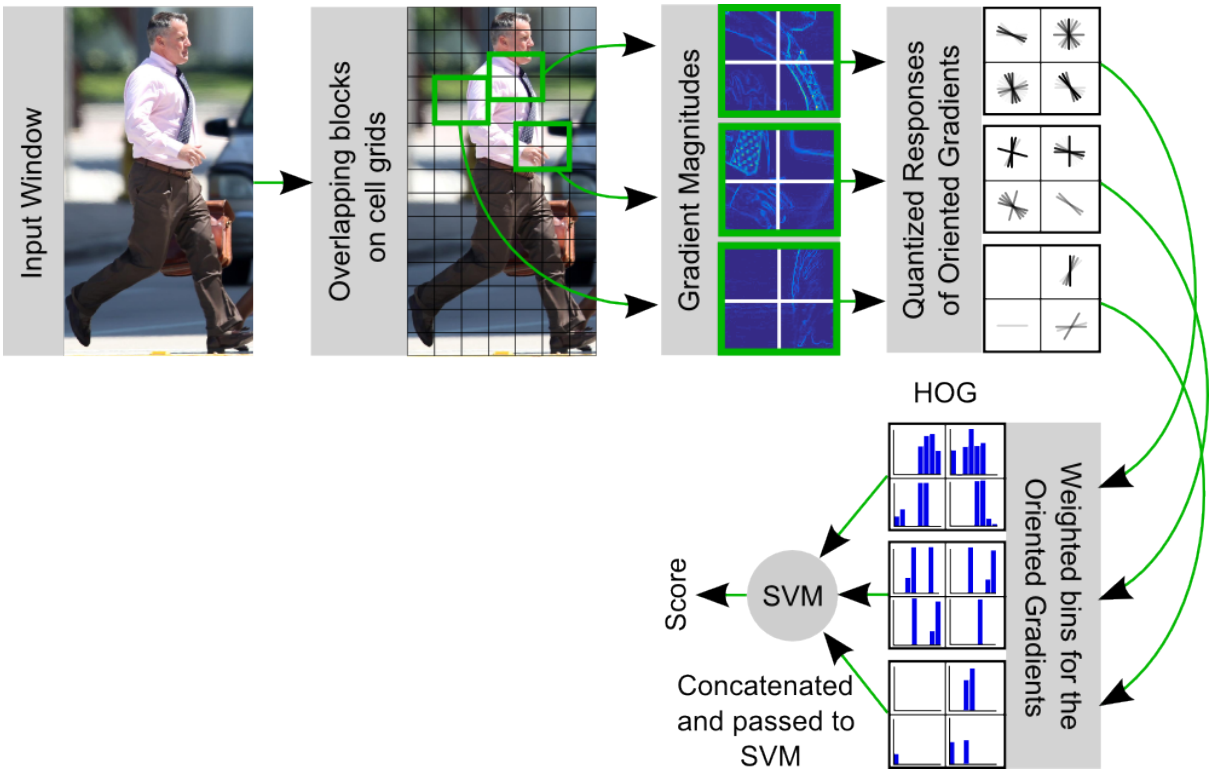


Figure 3.3: Image gradients are computed over a contrast-enhanced grayscale image. A histogram with 8 bins is produced for every cell on the basis of the orientation and magnitude distribution of the gradients. The x-axis of such histograms denotes the gradient orientation "brackets" while the y-axis denotes the gradient magnitude. For histograms with 8 bins the interval is $20°$ and the bins are spaced over $[0°, 180°]$. Binning is performed by voting the magnitude of every pixel within a 8×8 HOG cell into the appropriate histogram bin. This is visualized in the figure in form of "star plots" – the orientation of each line within these plots corresponds to the orientation of a bin in the HOG feature descriptor of one cell. Darker and bolder lines represent a strong gradient magnitude in these specific orientations. Groups of 2×2 adjacent cells forms a descriptor block (green boxes). Consequently each descriptor block consists of $2 \times 2 = 4$ histograms. In a final, step these blocks are locally normalized for contrast invariance (see the main text for more detailed explanation). All the descriptor values are then concatenated into a feature vector $\boldsymbol{x}$, which is then passed to a binary classifier e.g. a Support Vector Machine. The classifier predicts if the object is a pedestrian or not.

3.2.3 Block Normalization

Blocks of Cells. The magnitude of a gradient varies strongly in images containing pedestrians due to variable foreground-background contrast and non-uniform illumination [7]. Therefore, a local contrast normalization would pose beneficial by improving detection accuracy of the whole system. The method exploited in this research follows Dalal & Triggs [7] and groups adjacent cells into blocks of size (2×2). These blocks overlap by one cell, essentially creating an overcomplete feature representation which has shown that it has superior detection accuracy over non-overlapping schemes. However, the addition to computational complexity should not be underestimated and efficient pipelining of the code (e.g. pre-calculate one block while classifying the features of another) is of great importance.

Contrast-invariant Block Normalization. Dalal & Triggs apply an ℓ_2-normalization to each block, i.e. each histogram bin in each cell of the block is updated by the following scheme:

$$v_i \leftarrow \frac{v_i}{\sqrt{{v_1}^2 + {v_2}^2 + ... + {v_{32}}^2 + \epsilon^2}} \tag{3.4}$$

where v_i is the i'th bin of the concatenated cell-histograms of a block. ϵ is a small value (commonly set to 0.001) to ensure a non zero denominator in Equation 3.4 in case of absence of gradients. As mentioned earlier, these updates features are significantly contrast-invariant, which contributed to the good performance of the presented approach.

The Feature Vector. The histograms of all overlapping and normalized blocks are 'collapsed' into a single vector, the *feature vector*. For 64×128 pixel windows there are $(\frac{64}{8} - 1) \times (\frac{128}{8} - 1) \times 32 \ = 3360$ components of the vector, i.e. features. The space in which these features lie, which in this case is a subspace of $\mathbb{R}$, is called the *feature space*. Learning the recognition of pedestrian will take part in this feature space – Chapter 4 will explain this process in detail.

3.3 Feature Processing

Sliding Window Approach [44] is an exhaustive scanning method as it progressively visits every possible location in an image with all possible scales of the pedestrians (shown with white windows in Figure 3.4). However, it is widely believed that with growing power of GPU [3], this would no longer remain an issue. It is a common and popular approch because of its simplicity. The steps obtained by this approach are as follows:

The feature values are processed from hypotheses, the small regions (white boxes in Figure 3.4) at all possible locations of the input image, with the method explained in Section 3.2. These hypotheses are then passed to a binary classifier e.g. an SVM to compute a prediction score, which is an $\mathcal{O}(n)$ operation. This score predicts if the detected instance is a 'pedestrian' or a 'non-pedestrian'.

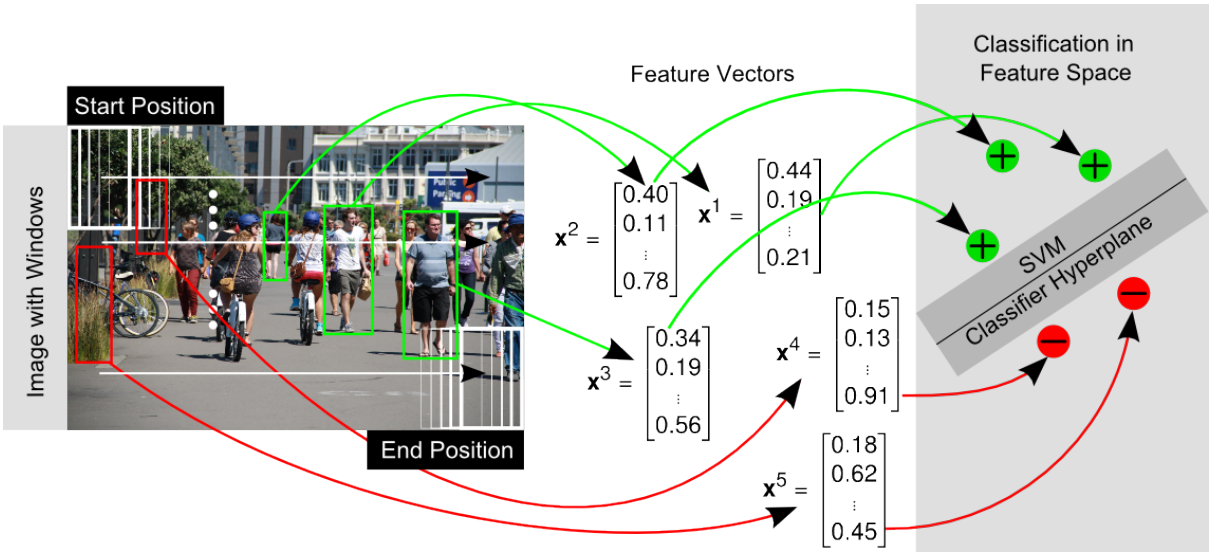


Figure 3.4: The white boxes are the slidding windows in this digram. A window of a fixed size progressively and densely scans the image at various scales [44]. The green windows contain the pedestrians i.e. the object class of interest, the red ones do not. Features are extracted from the underlying image content and represented by the feature vectors $\mathbf{x}^1, \mathbf{x}^2$ and $\mathbf{x}^3$ for the positive class and $\mathbf{x}^4, \mathbf{x}^5$ for the negative class respectively. A binary classifier e.g. SVM in feature space is used to predict that the window contains an instance of the object class.

3.4 Computational Bottlenecks

Multiple computational bottlenecks for computation of HOG features exist and have to be sidestepped to create an efficient implementation.

Speeding up Computation of $\mathrm{atan2}(\cdot,\cdot)$**.** The $\mathrm{atan2}(\cdot,\cdot)$ function is one of the most obvious ones. Without tuning, this function will execute a Taylor-Series to derive the functional values. To the best of the authors knowledge, no custom chips exist that would provide intrinsics to compute the $\mathrm{atan2}(\cdot,\cdot)$ function. One solution to remedy this situation is to realize that we are actually not interested in computing the $\mathrm{atan2}(\cdot,\cdot)$ function value itself but just the bin that the gradient is assigned to. This can be done with a set of if- and else-statements. Unfortunately, for reasons of confidentiality, we can not go into depth on this method.

If one were to compute the $\mathrm{atan2}(\cdot,\cdot)$ function one a DSP, then reasonable fast solutions exist [45]. One of these solutions is outlined in Appendix . Another solution is provided by the utilization of look-up-tables (LUTs). For the purpose of computing HOG features we would require a LUT that indicates the orientation bin and a separate LUT that indicates the magnitude of the gradient. Given 8-bit images, these LUTs would be of size 512×512. However, it should be noted that the lookups would appear random to the processor, i.e. efficient pre-fetching is not an option. It is therefore important that the LUTs remain in fast level-1 or level-2 cache. Given an 8 bit representation of each LUT entry this would require 512 KiB of storage space – a rather large number for a level-1 cache. However, modern CPUs and potentially some DSPs might offer level-2 caches that would fit this criteria.

Efficient Inverse Square Root for Block Normalization. Another bottleneck exists in the form of the calculation of the inverse square-root for block normalization as shown in Equation 3.4. A C++ function for fast computation of the inverse square root is shown in Appendix A.2. The source of this algorithm is not clear but the most probable explanation at this point is that the algorithm was developed at Silicon Graphics Inc. during the creation of the computer game Quake™. In this context it was used to efficiently calculate reflections of lightning and shading in the game. The function itself essentially performs one update of a Newton-Raphson optimization algorithm to approximate the inverse square root.

However, we must mention that most modern CPUs and DSPs have inbuilt intrinsics for the inverse square root, e.g. *rsqrtss* in SSE (Streaming SIMD Extensions, where SIMD stands for Single instruction, multiple data). Even in the newest set of computational intrinsics at the writing of this report (AVX-512, Advanced Vector Extensions), such operations exists, e.g. *vrsqrt14pd*. These outperform the above mentioned C++ function significantly.

Accelerating Dot-Products. In the next section of this report we will explain how to classify HOG features into the binary choice of pedestrian or background. This classification step essentially consists of a large dot-product of the HOG feature vector with a learned weight vector $\boldsymbol{w}$ that represents a separating hyper-plane between both classes. Dot-products can obviously be computed very efficiently by the use of widely available intrinsic operations (e.g. SSE and AVX both contain the necessary intrinsics). The FMA (Fused Multiply-Add) instruction set of the most recent CPUs is actually specifically aimed at computing dot products by offering the option of computing one multiplication and one addition within one clock cycle [46].

In addition, co-processors exist, which are solely build to perform the calculation of such dot-products and only these calculations. The process can be further sped up by representing the HOG features and the weight vector $\boldsymbol{w}$ in a (dynamic) fixed-point notation up to a desired exactness and withstandability of numerical error. Unfortunately again, due to reasons of confidentiality, we can not go into further detail at this point.

However, we believe we have pointed out that sufficient ways to counteract the bottlenecks of the proposed approach to pedestrian detection exist. One final approach would be the design of a dedicated chip (ASIC, Application-specific integrated circuit), which would combine some of the above mentioned ideas and directly integrate them into the hardware. This approach obviously lacks the flexibility of changing the feature representation (even to only small degrees) but will obviously yield a very fast chip with a low power-envelope, which should not be underestimated in the automotive domain.

CHAPTER 4
SUPPORT VECTOR MACHINES

The algorithm widely used for learning how to recognize pedestrians given HOG features is called an *Support Vector Machine* [47] (SVM). Such a scheme models an object (e.g. the pedestrian) w.r.t a set of parameters, which usually undergo an optimization step, which is the essential "learning-part" of the proposed scheme. An SVM was also successfully used by Dalal & Triggs [7].

4.1 Classification

Given the HOG features $\boldsymbol{x}$ of a window in an image, the SVM assigns a *score*, which determines how certain the algorithm is that this object is a pedestrian or not (binary classification). More formally to classify a window with a feature vector $\boldsymbol{x}$, an SVM computes the following function

$$h(\boldsymbol{x}) = \boldsymbol{w}^{\mathsf{T}}\boldsymbol{x} + b \tag{4.1}$$

where $\boldsymbol{w}$ is called the *weight vector* and b is the *bias*. These two parameters give rise to a hyperplane in feature space, which separates windows containing pedestrians from background windows in a sliding-window-fashion – see Chapter 1. A binary decision is commonly achieved by using the sign function on the output of the SVM, i.e.

$$g(\boldsymbol{x}) = sign\left(h(\boldsymbol{x})\right) = sign\left(\boldsymbol{w}^{T}\boldsymbol{x} + b\right) \tag{4.2}$$

with $g(\boldsymbol{x}) \in \{-1, 1\}$.

Linear Support Vector Machines. This work focuses on *linear* SVMs with a hyperplane as stated in the above equation. However, non-linear variants that use kernel functions to learn hyperplanes in a higher dimensional space do exist [48]. These methods are able to separate feature vectors of windows which do not appear to be separable in the original feature space by lifting them into a higher dimensional space where they can be separated. However, compared to the linear SVM used here both, training and classification, is computationally intensive for such methods and hence we lay the focus on linear SVMs. In addition, it should be mentioned that a non-linear SVM using an Radial-Basis-Function Kernel [48] was considered by Dalal & Triggs [7] and did not yield a significant performance benefit c.f.a much more simple linear SVM. On the other hand it is worth noting that Ott & Everingham [19] as well as Schwartz *et al.* [21] have shown substantial performance increases using a polynomial kernel. Unfortunately, it is beyond the scope of this report to explore such methods.

4.2 Optimal Hyperplane

As previously mentioned the primary objective of an SVM is to learn a hyperplane (which can be interpreted as an object model) that maximizes the margin between the two different classes

– see Figure 4.1. Many hyperplanes which separate the data might exist but the theory of the SVM states that the one that maximizes the margin between the two classes also yields the best generalization error, i.e. performs best on data it has not been exposed to during training [16]. Note that the data does not necessarily have to be linearly separable as will be shown in the next paragraphs (and in fact is represented in the constrained primal learning problem of the SVM [16]). However, the underlying principle – the maximization of the margin – remains and it is beyond the scope of this report to explore further formulations of the SVM learning problem.

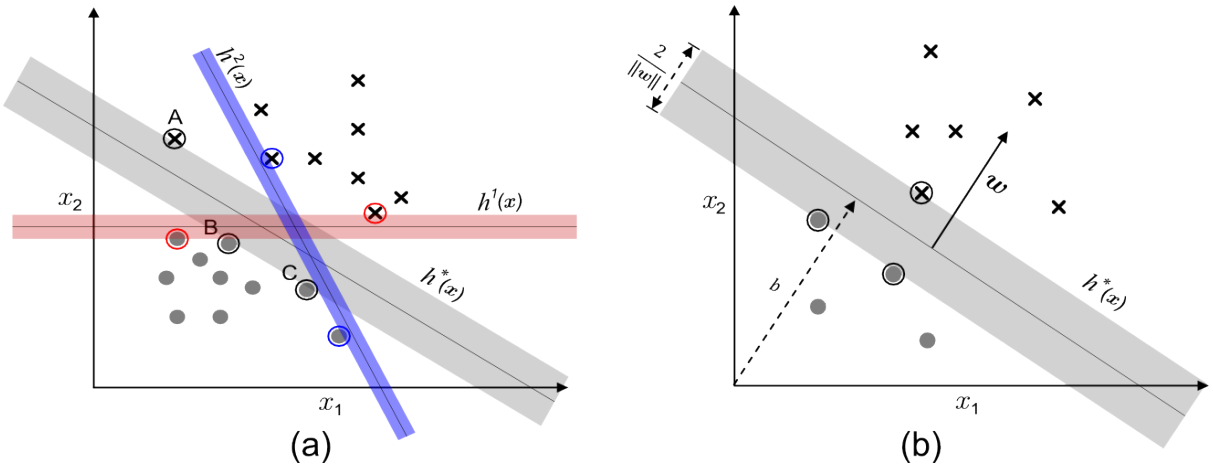


Figure 4.1: The diagrams contain examples of the positive class (marked by 'x') and examples of the negative class (marked by 'o'), which lie in a 2D feature space. An infinite number of hyperplanes exist which can separate the presented data. Two explanatory hyperplanes are shown in blue and red with their respective margins in Subfigure (a). The hyperplane that has the largest margin is drawn in gray – by maximizing the margin one hopes to increase detection accuracy on unseen data. The training examples that lie on the margin are called the *Support Vectors*. We denote this specific hyperplane as the optimal hyperplane h^*. The aim of the SVM optimization problem is to learn the parameters of this hyperplane – i.e. $\boldsymbol{w}, b$ – given a training dataset. In Subfigure (b) parameters $\boldsymbol{w}, b$ denote the normal vector of the hyperplane and the distance between the hyperplane and the origin (in feature space) respectively.

4.3 Learning Problem

The task of learning an SVM – i.e. in our case, teaching it to distinguish between pedestrian and background – can be formulated as an energy minimization problem. The energy to minimize is given by:

$$\mathcal{J}(\boldsymbol{w}, b) = \frac{\lambda}{2} \|\boldsymbol{w}\|_2^2 + \frac{1}{m} \sum_{i=1}^{m} \mathcal{L}_{\boldsymbol{w},b}\left(x^{(i)}, y^{(i)}\right) \tag{4.3}$$

and the optimal solution, which minimizes this energy, is desirable:

$$(\boldsymbol{w}^*, b^*) = \arg\min_{\boldsymbol{w},b} \mathcal{J}(\boldsymbol{w}, b) \tag{4.4}$$

The Regularization Term. The first term in Equation 4.3 is called the *regularization term*; λ being accordingly called the regularization parameter. The function of this term is to constrain the ℓ_2-norm of the solution of the learning problem. If the data is linearly separable, then there

would be nothing to stop the values of $\boldsymbol{w}$ to grow towards $\{-\infty, +\infty\}$ given the definition of $\mathcal{L}(\cdot, \cdot)$, which will be explained in the next paragraph. Essentially this term and the parameter λ control how well the learned object model fits the training data and how well it generalizes on unseen data.

The Loss Term. The second term in Equation 4.3 is known as the *loss term*. This term assigns a loss (a penalty) if an example is misclassified during learning. In fact, in case of the SVM, even slightly correctly classified examples (those for which $y^i h(x) < 1$ holds true) are penalized. An SVM uses a loss function called the hinge loss, which is given by:

$$\mathcal{L}_{\boldsymbol{w},b}(\boldsymbol{x}, y) = \max\{0, 1 - h(\boldsymbol{x})\, y\} = \max\left\{0, 1 - \left(\boldsymbol{w}^{\mathsf{T}}\boldsymbol{x} + b\right) y\right\} \tag{4.5}$$

This loss is, among other loss functions, visualized in Figure 4.2. It can be observed that the "hinge" appears at $y^i h(\boldsymbol{x}) = 1$, which enforces the learning of a maximal separating hyperplane.

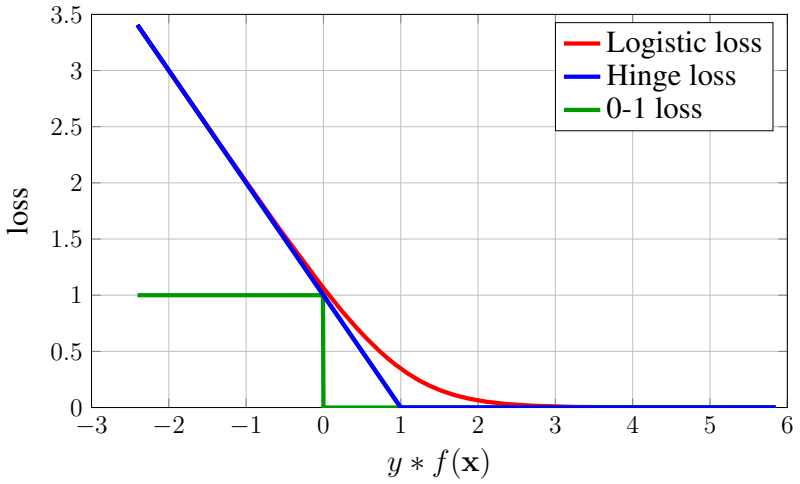


Figure 4.2: The graph shows the prediction value multiplied with the target value (pedestrian/background) on the x-axis. Note that in our case $f(\boldsymbol{x}) = h(\boldsymbol{x})$. The y-axis represents the imposed loss in case an example is misclassified or lies within the margin of the hinge loss (**blue** curve). The **green** curve shows the 0-1-loss, which is actually the foundation of Statistical Learning Theory and partially led to the discovery of Support Vector Machines and the hinge loss [16]. In case an example is misclassified, it pays a constant penalty of 1 otherwise it does not contribute to the accumulated loss. This function, due to the ambiguity of the derivatives at $y * f(\boldsymbol{x}) = 0$ is tough to optimize. The logistic loss is shown in red – it plays an important role in logistic regression [27], which provides a probabilistic output rather than a score (such as the SVM). In case this is not of much importance, an SVM usually performs slightly better. Interestingly enough the logistic loss converges to the hinge loss for $yf(\boldsymbol{x}) \to -\infty$ and $yf(\boldsymbol{x}) \to +\infty$.

Optimization of the Energy Function. The energy function is minimized by descending the weight vector multiple times towards the direction of $\left\{ \frac{\partial J}{\partial w_t} \middle| \, t = 1, ..., n \right\}$ where n is the number of features, i.e. we are performing gradient descent [49]. The method stops when the ℓ_2-norm of the gradient, i.e. $\|\nabla \mathcal{J}\|_2^2$, is smaller than a previously set ϵ-value – typically set in the vicinity of 0.01. The update equations of $\boldsymbol{w}$ and b are given by

$$w_s{}^{(t+1)} \leftarrow w_s{}^{(t)} - \eta \frac{\partial \mathcal{J}}{\partial w_s} \qquad s = 1, ..., n \tag{4.6}$$

with

$$\frac{\partial \mathcal{J}}{\partial w_s} = \lambda w_s - \sum_{y^{(k)} h\left(\boldsymbol{x}^{(k)}\right) < 1} y^k x_s^k \tag{4.7}$$

That is, only elements which violate the margin – $y * h(\mathbf{x}) < 1$ – contribute to the derivative. The update equation of the bias parameter can be obtained in a similar way:

$$b^{(t+1)} \leftarrow b^{(t)} - \eta \frac{\partial J}{\partial b} \tag{4.8}$$

with

$$\frac{\partial J}{\partial b} = -\sum_{y^{(k)} h\left(x^{(k)}\right) < 1} y^k \tag{4.9}$$

where η is the learning rate (can be variable or constant w.r.t the learning iteration) [50], $\boldsymbol{w}^{(t)}$ represents the weight vector at t^{th} iteration of learning and $\boldsymbol{w}^{(t+1)} \leftarrow \boldsymbol{w}^{(t)}$ indicates the update step of the weight vector.

This form of optimization has been considered as a part of this research due to the constant time taken by the derivative computation (as mentioned in Equation 4.6). However, due to the time and space constraints it is not possible to illustrate the complete reason in this report.

4.4 Final Overview

The feature vector explained in this chapter can be seen as the concatenated HOG descriptors over the detection window as explained in Chapter 3. These feature vectors for all the training samples are fed to the SVM (with batch or online scheme). This produces the pedestrian model as shown in Figure 4.3. During the testing the features are collected over the detection windows explained in Chapter 3, Section 3.3 and matched with the pedestrian model which is done by taking the vector multiplication between the model weight vector ('$\boldsymbol{w}$' in Equation 4.1) and feature vector ('$\boldsymbol{x}$' in Equation 4.1) from a detection window to predict if the detection window contains a pedestrian or not.

After modelling the detector, it is important to know how well this detector performs. This is done by evaluating the performance with different metrics explained in Chapter 5.

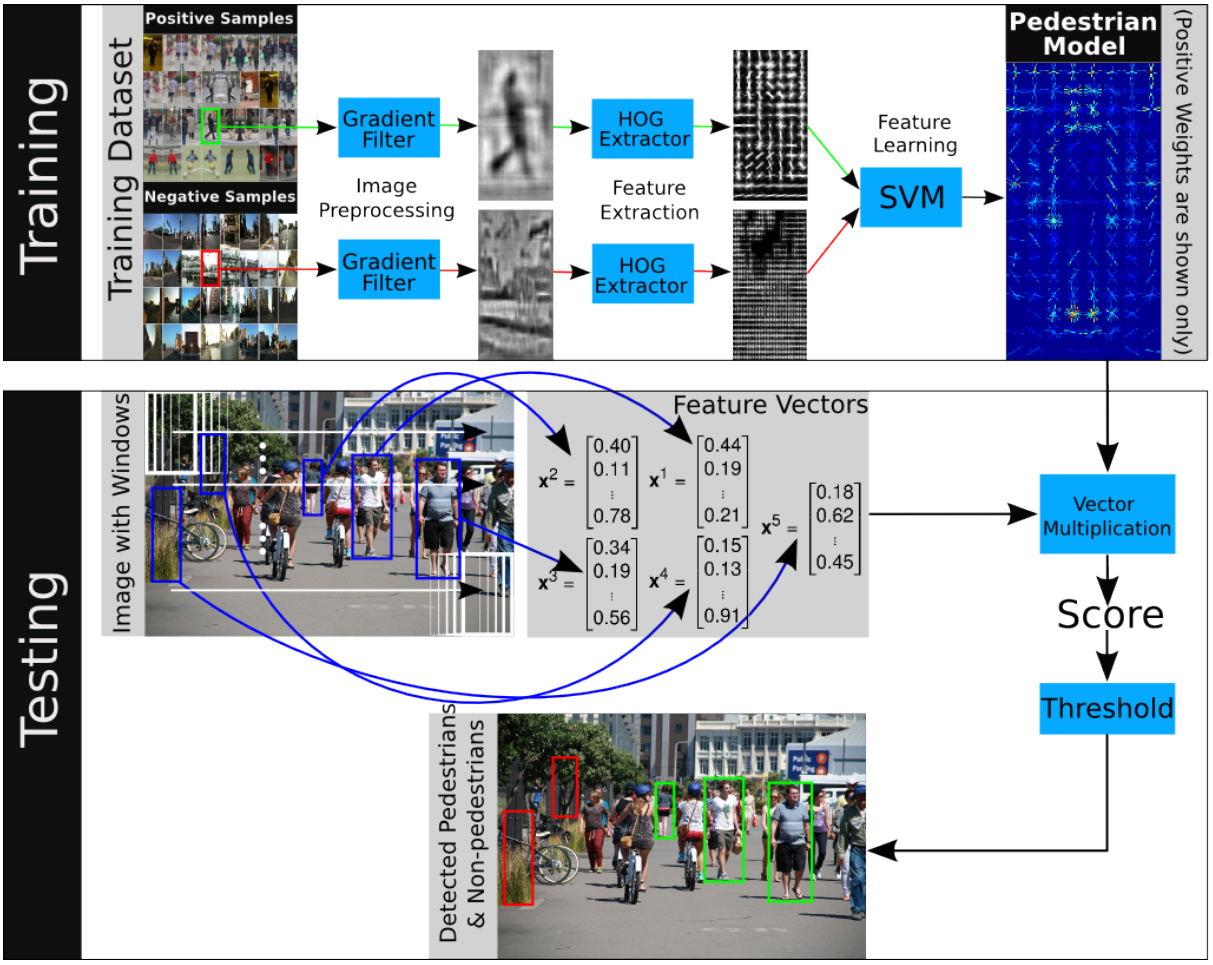


Figure 4.3: The individual training datasets used here are INRIA Person Dataset [7] and pedestrian dataset of ADAS. However in this diagram the positive and negative training data samples shown are taken from the former training dataset. Once the gradients of those sample images is computed, the HOG features are extracted from them. An SVM is used to learn these features for all the samples in the training dataset. This produces the pedestrian model (as visualized by the positive weights of the SVM in the far right of the figure). During testing (classification) a brute-force search for possible pedestrians in the image is executed in form of a sliding-window-detector – this follows the approach of Dalal & Triggs [7]. HOG features are extracted for every window and handed to the classifier to determine whether they resemble a pedestrian or not (as learned by the SVM). If the resulting score exceeds a given threshold (usually 0), the corresponding object is predicted as a pedestrian (green boxes), otherwise as background (red boxes).

Chapter 5
Evaluation

In this chapter we will present quantitative and qualitative evaluation results on two datasets, the INRIA Person dataset as well as the Continental in-house dataset (ADAS dataset).

For the INRIA Person Dataset [7] we will replicate the results of Ott & Everingham [19], since repeating the experiment again will not yield different results. We will also refrain from presenting qualitative results as they can be widely found in the literature, starting with the original work by Dalal & Triggs [7].

The INRIA Person *training dataset* consists of 2416 pre-cropped pedestrian windows. Additionally 1218 images with no pedestrians are provided – see Figure 5.1. The *testing dataset* consists of 1132 pre-cropped pedestrian windows as well as the 288 images these pedestrians were taken from. It also consists of 453 images without pedestrians.

For the ADAS dataset, gracefully provided by Continental, we will present quantitative as well as qualitative results within the boundaries of confidentiality. The ADAS dataset provides up to a million labeled pedestrians and up to multiple millions of images containing no pedestrians.

Figure 5.1: This diagram contains some of the positive and negative example windows that have been used to train the classifier. These images are taken from the INRIA Person Dataset [7]. To produce a stronger detector, the positive examples are mirrored along the y-axis. The figure gives a good insight into the different poses, clothes and backgrounds of images capturing pedestrians. Negative windows are also shown (right) and visualize the wide variety of possible background image material.

5.1 Training Protocol

If pre-cropped positive training data is given, we use these windows for training (in case of the INRIA Person dataset). In case complete images with meta-data in form of bounding box

Type	Prediction	Ground-Truth
True-Positive (TP)	yes	yes
False-Positive (FP)	yes	no
True-Negative (TN)	no	no
False-Negative (FN)	no	yes

Table 5.1: Explanation of the terms True-Positives (TP), True-Negatives (TN), False-Positives (FP), False-Negatives (FN). "yes" in the "Prediction"-tab indicates that the detector predicted an object (a pedestrian), "no" indicates no prediction has been given by the detector. "yes" in the "Ground-Truth"-tab indicates the presence of a ground-truth object, "no" indicates that there is no ground-truth object present.

coordinates is given (i.e. like in the ADAS dataset), we center a bounding box on them and extract HOG features from these bounding boxes. Negative windows are initially sampled at random from the previously mentioned sub-dataset, which does not consist of pedestrians. We then train an initial Support Vector Machine. To increase the detection accuracy of the classifier, we use bootstrapping [51], which essentially scans these images for high-ranking false-positives and adds them to the training dataset. We repeat this process 10 times.

5.2 Testing Protocol

The testing-protocol essentially consists of scanning all images in the test datasets using a sliding window detector. For the INRIA Person dataset, we additionally apply the detector to pre-cropped positive test examples and let it run on full images not containing pedestrians. These two different approaches, results in two different methods of qualitative evaluation and different measures of qualitative evaluation.

5.2.1 Qualitative Measures

We will shortly outline the qualitative measures used in this report to access the performance of the pedestrian detector. All methods build on the notion of True-Positives (TP), True-Negatives (TN), False-Positives (FP), False-Negatives (FN), which are explained in Table 5.1.

DET curve. To evaluate performance on the INRIA Person dataset the Detection Error Tradeoff (DET) curve was proposed by Dalal & Triggs [7]. DET curves are closely related to Receiver Operating Characteristic (ROC) curves [52], which plots the false positive rate against the true positive rate. However, the negative test data in the INRIA Person dataset is supplied by images not containing pedestrians and hence are scanned by a sliding window detector. On the other hand positive examples are provided in two versions, i.e. as pre-cropped image windows or as bounding boxes with full-sized images. For the first case the DET curve is usually applied as the ROC curve would present an almost perfect result, given the huge amount of negative windows (all windows that are scanned) c.f.the few available pre-cropped positive windows. DET curves plot False Positives Per Window (FPPW) on the x-axis against the miss rate, which is given by

$$MR_{\mathcal{S}}(h) = \frac{TP}{TP + FN} \tag{5.1}$$

where $\mathcal{S}$ is a test dataset and $(\cdot)$ the detector to be evaluated. The DET curve plots these measures on a log-log-scale which allows for closer observation than the standard ROC curve.

Precision/Recall curve. If for both, the positive and negative, class images with bounding boxes that indicate the location of an pedestrian are given, the precision/recall curve is commonly used to evaluate the performance of a detector. Generally recall is defined as "the proportion of all positive examples ranked above a given rank" and precision is given by "the proportion of all examples above that rank which are from the positive class" [53]. More formally, this can be stated as:

$$R_{\mathcal{S}}(h) = \frac{TP}{TP + FN} \tag{5.2}$$

and

$$P_{\mathcal{S}}(h) = \frac{TP}{TP + FP} \tag{5.3}$$

where R and P indicate recall and precision respectively.

While it shall be clear what a false negative in this context represents (i.e. an undetected pedestrian), the terminology for false-positives and true-positives requires further explanation. A window is considered a true-positive if the predicted bounding box overlaps with the ground-truth bounding box by at least 50%. Furthermore, if more than one predicted bounding box overlaps with the ground-truth object by at least 50%, the one with the highest score is considered a true-positive while all others are considered false-positives. More formally this can be stated as:

$$TP_{h(\cdot)}(B_p) \Leftrightarrow \left[ov(B_p, B_{gt}) = \frac{area(B_p \cap B_{gt})}{area(B_p \cup B_{gt})} \geqslant 50\% \right] \tag{5.4}$$

and

$$h(\boldsymbol{x}^p) = \max \{ h(\boldsymbol{x}^q) | \, \forall B_q : ov(B_p, B_{gt}) \geqslant 50\% \} \tag{5.5}$$

where B_p is the predicted bounding box, B_{gt} is the ground-truth bounding box, $B_p \cap B_{gt}$ denotes the intersection of the predicted and the ground-truth bounding box while $B_q \cup B_{gt}$ denotes their union. $\boldsymbol{x}^t$ denotes the feature vector of bounding box B_t.

To conclude the performance of a detector given the precision/recall curve of that detector into a single number, the area under the curve is considered. Obviously the goal is to push this number towards 1.

Non-Maximum Suppression. Clearly, the fact that all bounding boxes that overlap 50% with the ground-truth bounding box but do not have the highest score among them, are counted as false-positive requires a scheme that suppresses these bounding boxes. Such a scheme is commonly called a *Non-Maximum Suppression* (NMS) scheme.

In this report we rely on a greedy scheme proposed by Felzenszwalb *et al.* citeFelzenszwalb10. This schemes selects the highest ranking bounding box in an image and removes all bounding

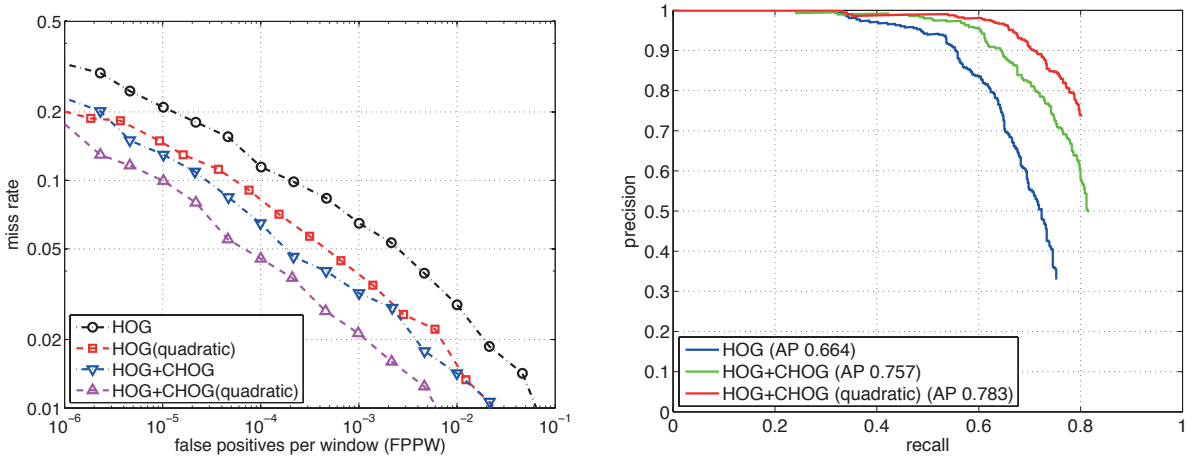
boxes that overlap up to 50%. It then repeats this process up until no bounding boxes are suppressed anymore.

5.3 Empirical results

In this section we present empirical results of the described method using the two presented datasets.

5.3.1 INRIA Person dataset

As previously mentioned, for the INRIA Person dataset, we will purely focus on the quantitative results. For qualitative results, the interested reader is pointed towards Dalal [54].



(a) Performance on DET curve

(b) Performance on Precision/Recall curve

Figure 5.2: This figure is graciously provided by Ott & Everingham [19]. (a) Performance of the described approach on the DET curve is shown by the black line. For example at 10^-4 FPPW a misrate of 0.12 is achieved. This translates into the fact that we miss on average 12% of the pedestrians every 10000 windows. The plot also shows the additional methods that were introduced by Ott & Everingham and discussed in Chapter 2. Both, the addition of a kernel ("quadratic") and the introduction of a new color-based HOG scheme ("CHOG") yield impressive performance benefits, especially when used in combination. (b) Performance of the described approach on the Precision / Recall curve. It is shown that the described approach ("HOG") provides an Average Precision (AP) of 0.664. At a recall level of 60%, the described approach is able to recognize 82% of the pedestrians. Again, the addition to the HOG+SVM-scheme presented by Ott & Everingham yield a significant performance improvement.

Figure 5.2 shows the performance of the proposed approach ("HOG") on a DET curve as well as on a precision/recall curve. In addition, the performance of the approach proposed by Ott & Everingham and discussed in Chapter 2 is shown. For details, please see the figure caption.

5.3.2 ADAS dataset

Here we will present quantitative and qualitative results on the ADAS dataset.

Quantitative results. Figure 5.3 shows results in form of a precision/recall-curve on the ADAS dataset. In addition, performance on multiple subsets of the dataset (such as for different

weather conditions) are shown. Overall the detector performance can be considered good. In combination with a strong tracking-model it produces very good results. Please see figure caption for more details.

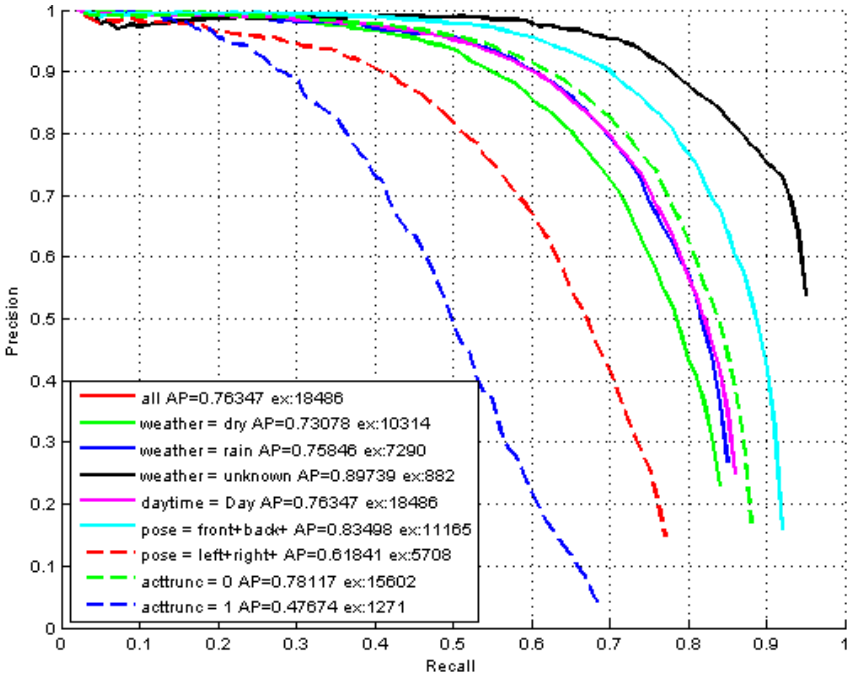


Figure 5.3: This figure illustrates the performance of the presented approach on the ADAS dataset under different image-specific conditions (weather, daytime) as well as instance specific conditions (pose, truncation). On the complete dataset ("all"), an AP of 0.76 is achieved. In addition to the AP, the legend also gives the number of examples in each subclass. Please note that the red curve can not be seen due to the magenta curve covering it (only daytime examples were used for this test). We can see that at a recall of 0.7 a precision of 80% is achieved. It can also be observed from the figure that the performance of the detector weakens for lateral pedestrians as well as for partially truncated pedestrians.

Qualitative Results. We show qualitative results in the form highest-ranking false-positive detections as well as highest-ranking true-positive detections, i.e. what the detector mistook for being a pedestrian and what the detector definitely considers to be a pedestrian.

Figure 5.4 shows a selection of highest-ranking false-positive detections. It accurately captures the distribution of the type of false-positive, i.e. the majority of the highest-ranking false-positives are indeed detections that do not overlap 50% with the ground-truth object. In other situation pedestrian-like shaped objects are often confused as pedestrians.

Figure 5.5 shows a selection of the highest-ranking true-positive detections. Generally it can be stated that front/back-facing pedestrians, independent of size, are reliably detected.

(a) (b)

(c) (d)

Figure 5.4: Highest-Ranking False-Positives (selection). (a,b,c) Very often false-positive results from insufficient overlap with the ground-truth object. This is in fact the majority of the highest-ranking false-positives. (d) Other often encountered high-ranking false-positives are on object which to some degree resemble a pedestrian, especially in HOG feature space.

(a) (b)

(c) (d)

Figure 5.5: Highest-Ranking True-Positives (selection). The system generally performs very well on front/back-facing pedestrians. In addition, even small object instances (c,d) are correctly detected and even score among the highest-ranking true-positives.

Chapter 6
Conclusion and Future work

This chapter concludes the research report by recapping on the key facts that were presented on the previous pages. In addition an outlook on possible future work is provided.

6.1 Conclusion

This report presented an approach towards pedestrian detection and object detection in general. It has been shown that by using Histograms of Oriented Gradient (HOG) features in combination with a Support Vector Machine, strong pedestrian detectors can be obtained.

6.2 Future Work

Two of the more recent theses on object detection [44, 55] have concluded that the most commonly used feature set, the here presented HOG features, may limit further development in the research area of object detection. However, multiple pieces of work (e.g. [19, 21]) have shown that careful handcrafting of features can lead to great performance improvements.

In the end the truth might lie, as it often so is, somewhere in the middle. Approaches which learn the representation of an image might require additional prior knowledge. In fact, when observing these approaches closer, it becomes obvious that prior knowledge for the task at hand, i.e. object detection, has already been provided to some degree. The fact that convolutional filter banks are used to represent an image strongly resembles the cell layout, which very often can be found in hand-crafted approaches [7]. Furthermore, the entire structure of a deep belief network can be considered prior knowledge as well, given the hand-crafted nature of it.

Hand crafted approaches, although while at the writing of this report being outperformed by methods of deep learning (e.g. the R-CNN [40]), also deserve a closer look. It has been shown that such approaches have potential when carefully designed [19, 21]. Very recently Ba & Caruana [56] proposed to mimic the output of a deep belief network with a shallow network, not very much unlike a typical SVM. The results were surprising in the fact that the shallow network was able to almost achieve the performance of the deep network. This fact shows that classical classification schemes, such as Support Vector Machines, might not have reached their full potential. The extension to kernel-based SVMs, while being slow, has shown a great leap in performance [19, 21]. Methods which approximate such SVMs [25] should be further investigated.

Meanwhile Deformable Part-based Models (DPMs) [20], their extensions [28], and their successors in the form of Object Grammars [29], have shown substantial performance increases over the classical HOG+SVM method proposed by Dalal & Triggs [7]. In addition, recent work by Girshick *et al.* [57] has shown that when replacing the underlying HOG features in a DPM with convolutional filters, a strong performance benefit can be achieved. This

furthermore shows that handcrafted schemes, i.e. providing the notion of a deformable part-based representation at the classification stage, provide a very good foundation for future research consisting of moving hand-crafted schemes and learnable schemes closer together.

APPENDIX A
CODES

A.1 DSP trick for atan2 computation

```
float arctan2(float y, float x)
{
   coeff_1 = pi/4;
   coeff_2 = 3*coeff_1;
   abs_y = fabs(y)+1e-10      // kludge to prevent 0/0 condition
   if (x>=0)
   {
      r = (x - abs_y) / (x + abs_y);
      angle = coeff_1 - coeff_1 * r;
   }
   else
   {
      r = (x + abs_y) / (abs_y - x);
      angle = coeff_2 - coeff_1 * r;
   }
   if (y < 0)
   return(-angle);     // negate if in quad III or IV
   else
   return(angle);
}
```

A.2 Fast Inverse Square Root computation

```
float Q_rsqrt( float number )
{
        long i;
        float x2, y;
        const float threehalfs = 1.5F;

        x2 = number * 0.5F;
        y  = number;
        i  = * ( long * ) &y;   // floating point bit level hacking
        i  = 0x5f3759df - ( i >> 1 );
        y  = * ( float * ) &i;
        y  = y * ( threehalfs - ( x2 * y * y ) );
        return y;
}
```

BIBLIOGRAPHY

[1] P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2001.

[2] Y. Freund and R. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. In *Proceedings of the Second European Conference on Computational Learning Theory (EuroCOLT)*, 1995.

[3] Gordon E Moore et al. Cramming more components onto integrated circuits, 1965.

[4] Franklin C. Crow. Summed-area tables for texture mapping. In *Proceedings of the 11th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '84, pages 207–212, New York, NY, USA, 1984. ACM. ISBN 0-89791-138-5. doi: 10.1145/800031.808600. URL `http://doi.acm.org/10.1145/800031.808600`.

[5] Piotr Dollár, Zhuowen Tu, Pietro Perona, and Serge Belongie. Integral channel features. In *BMVC*, volume 2, page 5, 2009.

[6] Paul Viola and Michael J Jones. Robust real-time face detection. *International journal of computer vision*, 57(2):137–154, 2004.

[7] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2005.

[8] D. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60, 2004.

[9] Phillip Ian Wilson and John Fernandez. Facial feature detection using haar classifiers. *J. Comput. Sci. Coll.*, 21(4):127–133, April 2006. ISSN 1937-4771. URL `http://dl.acm.org/citation.cfm?id=1127389.1127416`.

[10] Shai Shalev-Shwartz, Yoram Singer, Nathan Srebro, and Andrew Cotter. Pegasos: Primal estimated sub-gradient solver for svm. *Mathematical programming*, 127(1):3–30, 2011.

[11] Vojtěch Franc and Soeren Sonnenburg. Optimized cutting plane algorithm for support vector machines. In *Proceedings of the 25th international conference on Machine learning*, pages 320–327. ACM, 2008.

[12] P. Dollar, C. Wojek, C. Schiele, and P. Perona. Pedestrian detection: An evaluation of the state of the art. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 99, 2011.

[13] D. Gavrila and V. Philomin. Real-time object detection for "smart" vehicles. In *Proceedings of the IEEE International Conference on Computer Vision*, 1999.

[14] P. Felzenszwalb and D. Huttenlocher. Distance transforms of sampled functions. Technical report, Cornell Computing and Information Science, 2004.

[15] C. Papageorgiou, T. Evgeniou, and T. Poggio. A trainable pedestrian detection system. In *Proc. IEEE Intelligent Vehicles Symposium*, 1998.

[16] N. Cristianini and J. Shawe-Taylor. *An introduction to support vector machines : and other kernel-based learning methods*. Cambridge University Press, 2000.

[17] L. Sirovich and M. Kirby. Low-dimensional procedure for the characterization of human faces. *Journal of the Optical Society of America*, 4, 1987.

[18] D. C. Hogg. Model-Based Vision: A Program to See a Walking Person. *Image and Vision Computing*, (1), 1983.

[19] P. Ott and M. Everingham. Implicit color segmentation features for pedestrian and object detection. In *Proceedings of the IEEE International Conference on Computer Vision*, 2009.

[20] P. Felzenszwalb, R. Girshick, D. McAllester, and D. Ramanan. Object detection with discriminatively trained part based models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(9), 2010.

[21] W. Schwartz, A. Kembhavi, D. Harwood, and L. Davis. Human detection using partial least squares analysis. In *Proceedings of the IEEE International Conference on Computer Vision*, 2009.

[22] H. Wold. Partial least squares. *Encyclopedia of Statistical Sciences*, 6, 1985.

[23] O. Tuzel, F. Porikli, and P. Meer. Human detection via classification on Riemannian manifolds. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2007.

[24] J. Friedman, T. Hastie, and R. Tibshirani. Additive logistic regression: a statistical view of boosting. *Annals of Statistics*, 28(2), 2000.

[25] A. Vedaldi and A. Zisserman. Efficient additive kernels via explicit feature maps. *Pattern Analysis and Machine Intellingence*, 2011.

[26] A. Mohan, C. Papageorgiou, and T. Poggio. Example-based object detection in images by components. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(4), 2001.

[27] C. Bishop. *Neural networks for pattern recognition*. Oxford University Press, 1995.

[28] P. Ott and M. Everingham. Shared parts for deformable part-based models. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2011.

[29] R. Girshick, P. Felzenszwalb, and D. McAllester. Object detection with grammar models. In *Proceedings of Neural Information Processing Systems*, 2011.

[30] J. Winn and J. Shotton. The layout consistent random field for recognizing and segmenting partially occluded objects. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2006.

[31] Carolina Redondo-Cabrera, Roberto López-Sastre, and Tinne Tuytelaars. All together now: Simultaneous object detection and continuous pose estimation using a hough forest with probabilistic locally enhanced voting. *Proceedings BMVC 2014*, pages 1–12, 2014.

[32] J. Lafferty, A. McCallum, and F. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2001.

[33] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the Institute of Electrical and Electronics Engineers*, 86(11), 1998.

[34] Y. Bengio. Learning deep architectures for ai. *Foundations and Trends in Machine Learning*, 2(1), 2009.

[35] G. Hinton, S. Osindero, M. Welling, and Y. Teh. Unsupervised discovery of non-linear structure using contrastive backpropagation. *Cognitive Science*, 30(4), 2006.

[36] M. Ranzato, Y-Lan Boureau, and Y. LeCun. Sparse feature learning for deep belief networks. In *Proceedings of Neural Information Processing Systems*, 2007.

[37] D. Rumelhart, G. Hintont, and R. Williams. Learning representations by back-propagating errors. *Nature*, 323(6088), 1986.

[38] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems 25*, 2012.

[39] George E. Dahl, Tara N. Sainath, and Geoffrey E. Hinton. Improving deep neural networks for LVCSR using rectified linear units and dropout. In *ICASSP*, 2013.

[40] Ross B. Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. *CoRR*, 2013. URL `http://arxiv.org/abs/1311.2524`.

[41] Q. Zhu, M. Yeh, K. Cheng, and S. Avidan. Fast human detection using a cascade of histograms of oriented gradients. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2006.

[42] DL Bhombe and MB Ugale. A hybrid approach to edge detection.

[43] Piotr Dollar and C. Lawrence Zitnick. Structured forests for fast edge detection. In *Proceedings of the 2013 IEEE International Conference on Computer Vision*, ICCV '13, pages 1841–1848, Washington, DC, USA, 2013. IEEE Computer Society. ISBN 978-1-4799-2840-8. doi: 10.1109/ICCV.2013.231. URL http://dx.doi.org/10.1109/ICCV.2013.231.

[44] P. Ott. *Segmentation Features, Visibility Modeling and Shared Parts for Object Detection*. PhD thesis, University of Leeds, February 2012. URL http://patrick-ott.de/dl/thesis_ott.pdf.

[45] Dspguru.com. Dsp trick: Fixed-point atan2 with self normalization — dspguru.com, 2015. URL http://www.dspguru.com/dsp/tricks/fixed-point-atan2-with-self-normalization.

[46] Intel architecture instruction set extensions programming reference, 2015.

[47] Christopher J. C. Burges. A tutorial on support vector machines for pattern recognition. *Data Min. Knowl. Discov.*, 2(2):121–167, June 1998. ISSN 1384-5810. doi: 10.1023/A:1009715923555. URL http://dx.doi.org/10.1023/A:1009715923555.

[48] J. Shawe-Taylor and N. Cristianini. *Kernel Methods for Pattern Analysis*. Cambridge University Press, 2004.

[49] S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004.

[50] Léon Bottou. *Stochastic Gradient Descent Tricks*, volume 7700 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, 2012. ISBN 978-3-642-35288-1. doi: 10.1007/978-3-642-35289-8_25. URL http://dx.doi.org/10.1007/978-3-642-35289-8_25.

[51] E. Osuna, R. Freund, and F. Girosi. Support vector machines: Training and applications. Technical report, MIT, 1997.

[52] K. Spackman. Signal detection theory: Valuable tools for evaluating inductive learning. In *Proceedings of the Sixth International Workshop on Machine Learning (ML)*, 1989.

[53] M. Everingham, L. Van Gool, C. Williams, J. Winn, and A. Zisserman. The PASCAL Visual Object Classes (VOC) challenge. *International Journal of Computer Vision*, 88 (2), 2010.

[54] N. Dalal. *Finding people in images and videos*. PhD thesis, INRIA, 2006.

[55] R. Girshick. *From Rigid Templates to Grammars: Object Detection with Structured Models*. PhD thesis, University of Chicago, 2012.

[56] Jimmy Ba and Rich Caruana. Do deep nets really need to be deep? In *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014 (NIPS)*, 2014.

[57] Ross B. Girshick, Forrest N. Iandola, Trevor Darrell, and Jitendra Malik. Deformable part models are convolutional neural networks. *CoRR*, 2014.