Evaluation of Feature Representations for Object Detection with Boosted Classifiers

Bachelor's Thesis

Horst Possegger
0730874

Software Development and Business Management
F 033 524

Institute for Computer Graphics and Vision
Graz University of Technology, Austria

Supervisor
Sabine Sternig

Graz, January 23, 2011

Abstract

Object detection is one of the most difficult computer vision tasks. On the one hand, appearance and pose of the objects of interest may vary considerably. On the other hand, illumination changes, bad lighting conditions, and other external influences impose additional challenges. In order to overcome these problems, robust detectors cannot rely on the information extracted by raw pixel values alone. Thus, different feature types are used, which encode characteristical image information.

This thesis evaluates common feature representations for pedestrian and car detection. We focus on appearance-based features, which encode static image information, such as differences of grey values, edge information, as well as higher statistical moments. Additionally, we evaluate motion-based features, which operate on the information provided by the optical flow estimated throughout consecutive image frames. In our experiments, we use boosting to select the best-performing features for the corresponding detection task. For comparison, we evaluate the different feature types on several surveillance sequences with a single static camera.

Our experiments show that motion-based features alone cannot achieve good detection performance. Nevertheless, a combination of motion features and appearance features allows to build robust detectors with good performance.

Keywords:
Computer Vision, Detection, Boosting for Feature Selection, Evaluation

Zusammenfassung

Objekterkennung stellt aufgrund unterschiedlichster Herausforderungen eine der anspruchsvollsten Aufgaben im Bereich des Maschinellen Sehens dar. Einerseits kann das Erscheinungsbild der zu detektierenden Objekte, wie zum Beispiel Personen, mitunter stark variieren. Andererseits erschweren wechselnde Belichtungsverhältnisse, Schatteneffekte und andere externe Einflüsse die Detektion zusätzlich. Robuste Detektoren verwenden Abstraktionen, so genannte Features (Bildmerkmale), da eine Objektklasse nicht effizient mit reinen Pixelwerten der Farbkanäle modelliert werden kann. Diese Arbeit evaluiert unterschiedliche Feature Repräsentationen für die Klassifikation von Personen und Autos. Die behandelten Features extrahieren unterschiedliche Bildinformationen, wie zum Beispiel Grauwertdifferenzen, Gradienten als auch statistische Momente höherer Ordnung.
Um die Qualität der trainierten Objektdetektoren zu verbessern, verwenden wir zusätzliche Bewegungsfeatures, die in ein bestehendes Framework integriert werden. Diese Features abstrahieren Informationen aus dem optischen Fluss, der aus aufeinanderfolgenden Bildsequenzen bestimmt werden kann. Da die Bewegung eines Objektes innerhalb der Szene allein meist nicht ausreichend ist um dieses zu klassifizieren, werden die Bewegungsfeatures mit anderen Repräsentationen kombiniert, um so die Ergebnisse der Detektoren zu verbessern. Wir verwenden Boosting, um die besten Features zur Detektion zu selektieren. Die Features werden anhand der Ergebnisse auf unterschiedlichen Szenen verglichen. Jede Szene wird von einer statischen Kamera überwacht.
Die experimentellen Resultate zeigen, dass Bewegungsfeatures alleine nicht ausreichen, um einen robusten Detektor zu trainieren. Allerdings ermöglicht die Kombination von statischen und Bewegungsfeatures die Konstruktion von Detektoren mit guten Ergebnissen auf den Testsequenzen.

Schlüsselwörter:
Computer Vision, Objekterkennung, Feature Selektion, Evaluation

Contents

List of Figures

List of Tables

1 Introduction

Object detection, in general, is an important but challenging task. On the one hand, appearance and pose of an object may vary considerably, *e.g.* considering human clothing and shape. On the other hand, illumination effects, shadows, and dynamic backgrounds impose scene-specific challenges. The goal would be to overcome all of these challenges and to provide robust results. However, collecting large datasets of representative object samples is time-consuming and intricate. Thus, training data is limited in general, *i.e.* the dataset cannot contain a sample for every possible variation of an object's appearance. Hence, the complete knowledge about an object class cannot be modelled by pure pixel values alone [VJ02]. Therefore, feature representations are used, which encode several kinds of image content, such as gradient information, grey value differences, various filter respones, and the like.

This thesis evaluates feature representations which are commonly used for object detection, *i.e.* Haar features, Local Binary Patterns, covariance features, and Histograms of Oriented Gradients. Additionally, we implement motion-based Oriented Histograms of Optical Flow as proposed by Dalal *et al.* [DTS06], which operate on flow components extracted from consecutive image frames. Since motion can be characteristic for an object class, motion-based features can be used to improve the detection performance in combination with appearance-based features. Our evaluation will focus on pedestrian and car detection. Pedestrians are known to be one of the most challenging classes for object detection, mainly because appearances and poses vary considerably [WS08]. Cars on the other hand have less variations in appearance and shape. Thus, we will compare popular feature types on two very diverse object classes.

Since boosting can be used for efficient feature selection, we apply it to train detectors for evaluation [GB06]. The feature representations are evaluated on surveillance scenarios with a single static camera. Thus, camera motion is cancelled out, which simplifies the detection challenge. Therefore, our experimental comparison will show which features encode pedestrians and cars reasonably well.

This thesis is organized as follows. First, section 2 discusses commonly used feature types for detection tasks. Second, we will describe the application of boosting in order to build the detectors for evaluation in section 3. Next, section 4 will review implementation details and present results on pedestrian and car detection. Finally, section 5 concludes our work.

2 Features

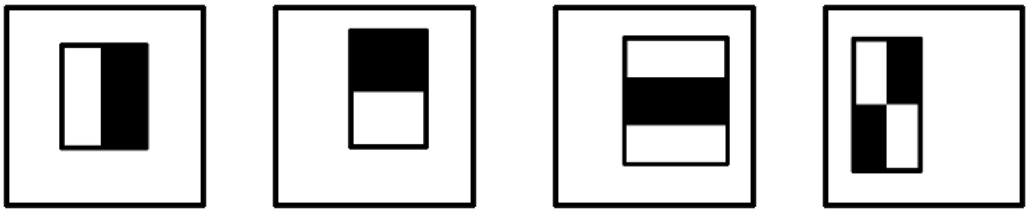
Features are an abstraction of image information crucial for any kind of image-based analysis. A lot of different feature types are commonly used to encode domain knowledge such as gradient information, variations of local illumination, and so forth. Since the training dataset has a limited size, these cues cannot be modelled directly by raw pixel values [VJ02]. On the other hand, features also simplify the detection task by reducing the intra-class variability and increasing the inter-class variability of objects within an image [LM02].

In this section we will discuss both appearance-based and motion-based features. Appearance-based features consider object shape or texture information, whereas motion-based features operate on the flow information provided by consecutive image frames. In the following we discuss Haar features, Local Binary Patterns covariance-based features, and Histograms of Oriented Gradients. All these features are based on object appearance. The experiments by Dalal *et al.* [DTS06] and Viola *et al.* [VJS03] show that the combination of static appearance and dynamic motion information improves detection performance. Therefore, we will describe the motion-based Oriented Histograms of Optical Flow features in order to introduce an additional cue.

2.1 Haar Features

Haar features have been introduced by Papageorgiou *et al.* [POP98]. These features can be used to model simple image structure such as edges, bars, and lines [VJ02]. Calculation of Haar features can be done very efficiently and thus allows to build fast object detectors which can handle real time detection. Haar features compute differences of sums of pixels between rectangles within the input image. The rectangular regions can be of arbitrary size. However, Viola and Jones propose to use adjacent rectangles of equal size [VJ02]. Unequally sized regions can be used if each rectangle is normalised before [SHB07]. Figure 1 illustrates the different rectangle types described by Viola and Jones. Two-rectangle features, as illustrated in figure 1(a) and 1(b), mostly encode edges, while three-rectangle features, illustrated in figure 1(c), can be used to model lines [LM02].

To compute Haar features efficiently, Viola and Jones [VJ02] introduced an auxiliary image representation, called *integral image*. The value of the integral image ii at position (x, y) is the sum of all pixels of the original image $\mathcal{I}$ that are located above and to the left of (x, y). The integral image



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

Figure 1: Samples of two-, three-, and four-rectangle Haar features. The value of each Haar feature is the difference between the sum of the pixels within the black rectangles and the sum of the pixels within the white rectangles. Figures 1(a) and 1(b) show vertical and horizontal two-rectangle features. Figure 1(c) shows a vertical three-rectangle feature, and figure 1(d) a four-rectangle feature. Illustrations taken from [VJ02].

at position (x, y) is defined as

$$ii(x, y) = \sum_{x' \leq x, y' \leq y} \mathcal{I}(x', y').$$

The integral image can be created by a single row-by-row pass over the original image and allows fast computation of any type of rectangular features [VJ02]. Once the representation has been computed, the sum of all pixels within any rectangle can be calculated by a constant number of lookups on the integral image. Figure 2 illustrates the computation of the sum of all pixels within a rectangle. The integral image representation is very similar to the *Summed Area Table* defined by Crow in [Cro84].

Since Haar features are defined by vertically and horizontally oriented rectangles, they have a limited flexibility. However, the efficient computation compensates for this drawback [VJ02]. Lienhart and Maydt propose an extended set of Haar features that additionally consists of 45° rotated rectangles in [LM02]. Figure 3 illustrates a few examples of these features. The experiments by Lienhart and Maydt show that detectors using the extended feature set achieve 10% less false positives than detectors using the basic set without rotated Haar features. The hit rates at both experiments were almost the same [LM02]. In order to compute the rotated Haar features efficiently, Lienhart and Maydt defined the *Rotated Summed Area Table* as an extension to the integral image representation. This representation can be calculated by two passes over the input image. After this representation is computed, all sums of pixels within 45° rotated rectangles can be computed by a constant number of array references. The better false positive rate ob-

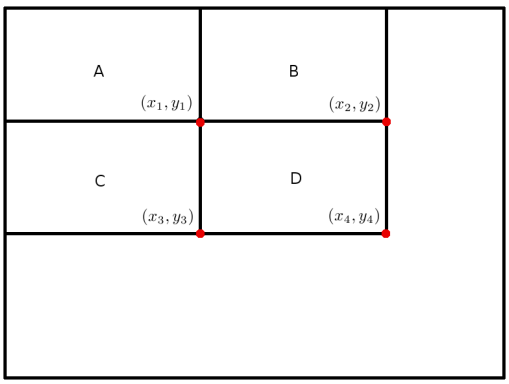
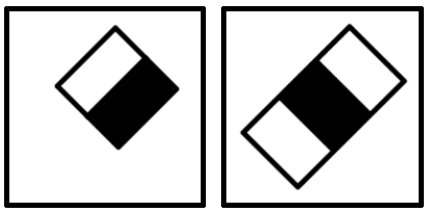


Figure 2: The integral image representation. The value $ii(x_1, y_1)$ is the sum of all pixels inside rectangle A. The value $ii(x_2, y_2)$ is the sum of all pixels inside the rectangle A + B. Thus, the sum of all pixels within D can be computed as $ii(x_4, y_4) - ii(x_3, y_3) - ii(x_2, y_2) + ii(x_1, y_1)$. Illustration based on [VJ02].



(a) (b)

Figure 3: Samples of 45° rotated Haar features. Figure 3(a) shows two-rectangle edge features, while 3(b) shows three-rectangle line features. Illustrations taken from [LM02].

viously compensates for the additional computation effort required to create the auxiliary representation.

2.2 Local Binary Patterns

Local Binary Patterns (LBP) were introduced by Ojala *et al.* [OPH96] in order to describe image textures. These features provide good performance and are resistant against monotonic changes of illumination. Ojala *et al.* show that LBP features achieve better classification results on textures than grey-level difference methods, covariance measures, and *Laws'* texture measures.

The LBP calculation thresholds the grey values within a predetermined neighborhood with the value of the center pixel. The definition by Ojala *et*

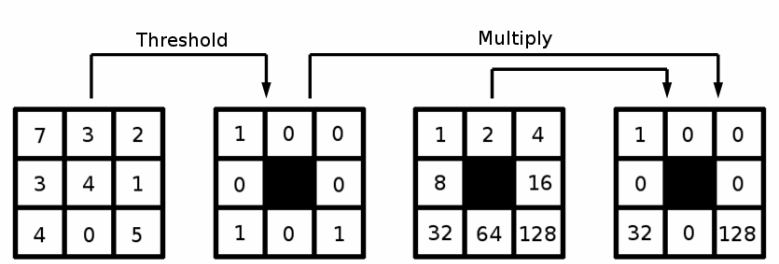


Figure 4: Schema of calculating the LBP code of a pixel using a 3×3 neighborhood. In the illustrated example, the value of the LBP feature would be $1 + 32 + 128 = 161$. Illustration based on [HPH04].

al. [OPH96] proposes a 3×3 neighborhood. A more general version of the LBP feature uses a circularly symmetric neighborhood [HPH04]. The result is encoded as a binary number, so called *LBP code*. More formally, the value of the LBP at position (x_c, y_c) can be defined as

$$LBP(x_c, y_c) = \sum_{p=0}^{P-1} s(g_p - g_c)2^p,$$

where g_c represents the grey value of the center pixel at (x_c, y_c) and g_p is the grey value of the P neighboring pixels. The function $s(g)$ is defined to threshold the grey value difference as follows

$$s(g) = \begin{cases} 1 & g \geq 0 \\ 0 & g < 0 \end{cases} .$$

Figure 4 illustrates the computation of the LBP value for one pixel. The LBP codes of all pixels within an image block are used to vote into the bins of a histogram. This histogram can be used as a texture descriptor for detection tasks. [HPH04]

Heikkilä *et al.* [HPH04] show that LBP features can be successfully applied to detect moving objects. Therefore, they introduce an adaptive background subtraction method. Their algorithm using LBP features clearly outperforms other histogram based background subtraction methods.

2.3 Covariance Features

Covariance features were introduced by Tuzel *et al.* [TPM06]. These features can be used to efficiently combine raw pixel information, *e.g.* RGB or intensity values, and higher order image moments, *e.g.* first and second

order gradients. This feature type encodes spatial and statistical image information as well as correlation relations between feature values. Tuzel *et al.* propose that a single covariance matrix computed over a region of interest is enough to match it in several poses and views.

Given an input image $\mathcal{I}$ of dimension $w \times h$, a d-dimensional feature representation F of dimension $w \times h \times d$ can be extracted. The covariance matrix $\Sigma_r \in \mathbb{R}^{d \times d}$ for an arbitrary $N \times M$ sized rectangular region $R \subset F$ can thus be calculated by

$$\Sigma_R = \frac{1}{NM-1} \sum_{x=1}^{N} \sum_{y=1}^{M} \left(F(x,y) - \mu\right) \left(F(x,y) - \mu\right)^T,$$

where $\mu \in \mathbb{R}^d$ is the mean vector. The subtraction of this mean vector leads to zero-mean normalisation and provides resistance against changes in illumination [KMB09]. The mean vector of a region of interest is defined as

$$\mu = \frac{1}{NM} \sum_{x=1}^{N} \sum_{y=1}^{M} F(x,y).$$

The diagonal elements of the covariance matrix Σ_R contain the variance of each source channel, the off-diagonal elements represent the correlation between the involved image statistics [KMB09]. Tuzel *et al.* propose to use a 9-dimensional feature vector containing the pixel location (x, y), RGB values, and the norm of the first and second order intensity derivatives with respect to x and y [TPM06]. Thus, the covariance matrix describing a region of interest would be $\Sigma_R \in \mathbb{R}^{9 \times 9}$.

Covariance matrices do not lie on Euclidean space. Thus, the originally proposed features cannot be used with standard machine learning algorithms [TPM06]. Operations such as distance computation have to be carried out using Riemannian geometry, which requires increased computational effort. Therefore, Kluckner *et al.* [KMB09] propose a method for computing an efficient approximation on Euclidean space which can be directly used in machine learning strategies. The concept of their approximation is to choose a representative set of vectors from two given distributions. This set of vectors allows to apply the Euclidean distance measure. The covariance matrix Σ'_R of this representative set must be equivalent to Σ_R. This means that the representative set and the region of interest R have the same second order statistics. The set of columns of a matrix $\mathbf{A}$ that satisfies $\Sigma_R = \mathbf{A}\mathbf{A}^T$ is equivalent to R in terms of second order statistics [HCS+09]. A matrix can be factorized if it is nonsingular. Since most covariance matrices are semi-positive and definite, factorization can be applied [HCS+09]. Singular

covariance matrices only occur as an extremal case where a feature has a constant value for each pixel inside a region of interest [HCS+09]. Kluckner *et al.* propose to apply the *Cholesky* factorization to decompose the covariance matrix Σ_R into $\mathbf{L}\mathbf{L}^T$, where $\mathbf{L}$ is a lower triangular matrix. Although there exist several methods for matrix factorization, the *Cholesky* decomposition requires the lowest number of operations on symmetric and semi-positive definite matrices [HCS+09, KMB09]. We use the approximation proposed by Kluckner *et al.* throughout our experiments.

A similar approximation of covariance matrices on Euclidean space has been proposed by Hong *et al.* [HCS+09]. Their basic idea is to create a set of points within the Euclidean space, which covariance matrix is equivalent to the covariance matrix of the region of interest within the image. The set of points should be of low dimensionality and is called *Sigma Set*. As well as Kluckner *et al.*, they propose to apply the *Cholesky* decomposition for better efficiency.

2.4 Histograms of Oriented Gradients

Histograms of Oriented Gradients (HOG) were introduced by Dalal and Triggs [DT05]. These features are based on the distribution of local edge direction and gradient magnitudes. This distribution describes the object shape and appearance without considering the precise location of the gradients. However, the knowledge of the exact edge positions is not relevant for good detection performance. HOG features reduce sensitivity to illumination variations and can also detect small changes in an object's appearance. Although these features provide good detection performance, they cannot be used for fast real time applications on arbitrary hardware. [DT05]

The first step in calculating HOGs is to normalise the input image and compute first order gradients. Although the normalisation of the input image is optional, Dalal and Triggs propose to apply it to achieve improved resistance to illumination variations. This can be explained by the fact that texture strength is typically proportional to surface illumination [DT05]. Furthermore, Dalal and Triggs suggest to use simple $[-1, 0, 1]$ derivative masks to calculate the image gradients. Compared to other masks, *e.g.* 3×3 Sobel masks, these simple ones perform best with the proposed HOG features. Next, the image window is divided into small spatial regions, so called *cells*. Each cell accumulates a histogram of local edge orientations over all pixels within the cell. The allowed gradient angle range can be predetermined to either $[0^\circ, 180^\circ]$ or $[0^\circ, 360^\circ]$. The choice depends on the objects to classify. The limited range $[0^\circ, 180^\circ]$ yields better performance for pedestrian detection whereas the full 360° range achieves better results on the detection

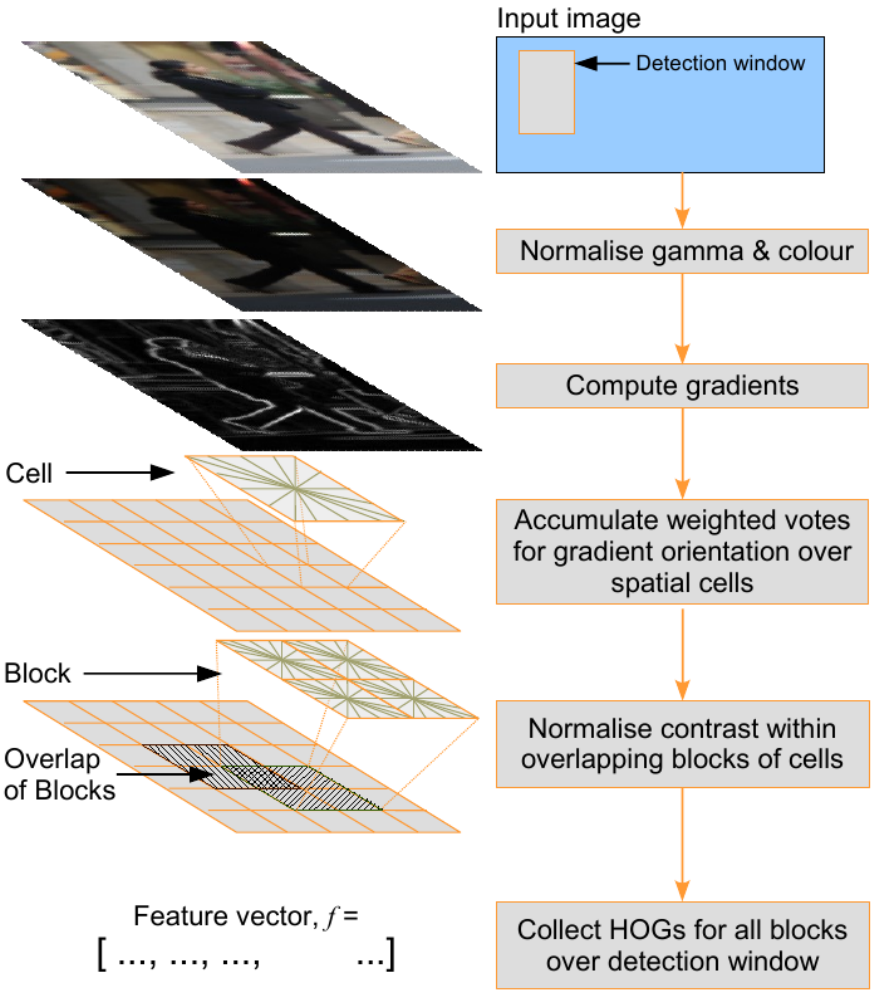


Figure 5: Overview of the HOG feature extraction process. The first order gradients are computed based on the gamma normalised input image. Gradient orientations and magnitudes are then used to vote into cell histograms. Next, overlapping cells are normalised and combined into blocks. The final feature vector is built by combining the block histograms. Illustration taken from [Dal06].

of cars and motorbikes [DT05]. The chosen angle range is divided by the number of histogram bins into equally sized regions and each pixel calculates a weighted vote into the corresponding bin. The vote is based on the orientation and magnitude of the edge that is centred on it. The number of histogram bins must be predetermined. Since more bins do not improve the descriptor's performance, Dalal and Triggs [DT05] propose that 9 histogram bins should suffice. Since gradient magnitudes vary due to local changes of illumination, the next step is to provide extended invariance to illumination effects through histogram normalisation. Therefore, cells are combined into larger spatial regions, so called *blocks*. Each block calculates a histogram measure over all cells within the block which is used to normalise all cells within the corresponding block. Dalal and Triggs [DT05] propose that a cell should be shared between several blocks. Since cell normalisation depends on the block's histogram measure, the normalised cell is different for each block it appears in. Finally, a dense overlapping grid of blocks is created and the normalised histograms of all blocks are concatenated to build the resulting feature vector. The experiments by Dalal [Dal06] show that highly overlapping grids provide better performance than grids with low overlap. Figure 5 illustrates the extraction of HOG feature vectors.

Dalal and Triggs [DT05] conclude that good performance can be achieved by using fine scale derivatives without smoothing with at most 9 orientation bins and strongly normalised, highly overlapping blocks. The proposed cell size is 8×8 pixels with 2×2 cells per block. The block overlap with these parameters should be one cell in each direction. In their experiments, HOG features clearly out-perform other feature types like Haar features for person detection.

2.5 Oriented Histograms of Optical Flow

Dalal *et al.* [DTS06] introduced the Oriented Histograms of Optical Flow (HOF) features. These features model object movements and are resistant against background or camera motion. HOF features can be computed in almost the same way as the Histograms of Oriented Gradients. The only difference in computation is that HOF features are calculated over gradients of the optical flow components instead of the input image gradients. Figure 6 illustrates the feature extraction of both HOG and HOF. As with the HOG feature, the HOF computation also consists of dividing a window of interest into small cells, grouping them into blocks, and strong normalisation. Although the motion cues allow to improve detection performance, building real time detectors requires fast hardware.

In contrast to gradients of the appearance channel, gradients of the flow

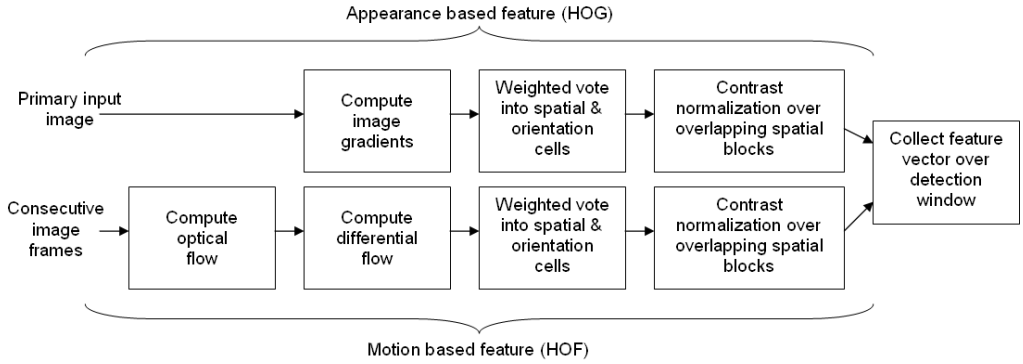


Figure 6: Comparison of the extraction process of HOG and HOF features. Illustration taken from [DTS06].

components are more blurred. This is caused by errors in flow estimation, which lead to imprecise location of the motion edges [Dal06]. Dalal *et al.* propose to estimate the optical flow based on the constant brightness assumption. Since optical flow is the distribution of motion velocities of brightness patterns within an image, this allows to calculate the flow estimates [HS81]. Although the flow estimation proposed by Dalal *et al.* [DTS06] is less accurate than other methods, their experimental results show that this estimation provides better performance for HOG features than the implementations of Galvin *et al.* [GMN+98].

We will use the following notation throughout this section. $\mathcal{I}^x, \mathcal{I}^y$ define images holding the horizontal (x) and vertical (y) flow components. Subscripts denote x- and y-derivations of an image. *E.g.*, $\mathcal{I}^y_x = \frac{\partial}{\partial x}\mathcal{I}^y$ is the x-derivative of the vertical flow component. [DTS06]

Considering motion information, we can differ between an object's motion boundaries and relative motion, *e.g.* human limbs that move relative to the torso. Therefore, Dalal *et al.* define two types of HOF features: *Motion Boundary Histograms* and *Internal Motion Histograms*.

Motion Boundary Histograms (MBH) consider the local edge orientations of each flow component independently. These are the simplest type of HOF features. To calculate the feature vector, the first order image gradients of the flow components are computed. This results in two separate image representation pairs $(\mathcal{I}^x_x, \mathcal{I}^x_y)$ and $(\mathcal{I}^y_x, \mathcal{I}^y_y)$. Next, each flow component is divided into small cells. Afterwards, the corresponding gradients vote into local histogram bins using their magnitude. Analogical to the HOG features, local cells are grouped into blocks. This results in two separate feature vectors that are

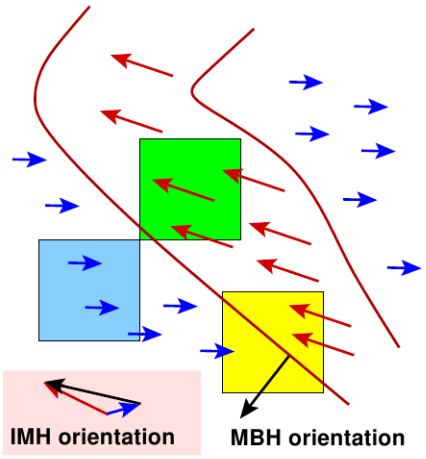


Figure 7: Comparison of MBH and IMH features. The bottom right cell illustrates the motion cue encoded by MBH features. IMH features encode the motion of a cell relative to its neighbors as illustrated by the two adjacent cells top left. Illustration taken from [Dal06].

based on the information provided by the pairs $(\mathcal{I}_x^x, \mathcal{I}_y^x)$ and $(\mathcal{I}_x^y, \mathcal{I}_y^y)$. Please note that calculating the HOG representation results in just one feature vector as only the first order gradients of the input image $(\mathcal{I}_x, \mathcal{I}_y)$ are required for angular voting. The two feature vectors provided by the separate flow components must be combined. This can be done by various means, *e.g.* simple concatenation which forms a large feature vector or combination by applying a winner-takes-all voting, where the largest values out of both components build a smaller feature vector. Dalal *et al.* conclude that treating the resulting vectors separately achieves the best performance. Thus, concatenation of the vectors of both flow components gives the MBH feature vector. [DTS06]

The silhouette information used by MBH features seems to suffice for detection of objects without internal motion, *e.g.* cars. However, detecting humans imposes a more challenging task due to the possible movement of limbs. Thus, Dalal *et al.* [DTS06] introduced *Internal Motion Histograms* (IMH). In order to encode additional information about relative motions, the angular voting is based on the direction of the flow difference vector, instead of using the spatial derivative displacement as within the MBH features. Therefore, IMH features use the vector pairs $(\mathcal{I}_x^x, \mathcal{I}_x^y)$ and $(\mathcal{I}_y^x, \mathcal{I}_y^y)$ to vote into the histograms. Dalal *et al.* propose several types of IMH features. The simplest feature type is called *IMHdiff*. This type is calculated in the

same way as MBH features. The only difference as mentioned above is to use the pairs $(\mathcal{I}_x^x, \mathcal{I}_x^y)$ and $(\mathcal{I}_y^x, \mathcal{I}_y^y)$ for angular voting. Other variations of IMH features use the blocks-of-cell structure as applied for HOG and MBH features in a different way. Figure 7 illustrates the difference between MBH and IMH features. Dalal *et al.* [DTS06] conclude that IMH features provide better performance than MBH features for pedestrian detection. However, we implement MBH features since we want to evaluate the effect of these features for both pedestrian and car detection.

3 Boosting

There exist several suitable machine learning algorithms for object detection. Popular classifiers for pedestrian and car detection are for example Support Vector Machines (SVM) and boosting [WWS09]. Since boosting proved to achieve good performance in a wide variety of machine learning tasks [GB06], we use it to build our classifiers.

This section describes the boosting algorithm and its application for feature selection. First, we review the standard offline boosting algorithm. Second, we detail the feature selection process using both offline and online boosting. Finally, we discuss the challenges for selecting a suitable classifier for the different feature representations.

3.1 Offline Boosting

Boosting is an efficient method for combining multiple classifiers to form a classifier that performs better than any of the simple classifiers alone. Since the simple classifiers only have to perform better than chance in order to achieve good performance after boosting, they are also called *weak* classifiers. Thus, if for a two-class classification problem the error rate of any weak classifier is less than 50%, boosting can be applied to build a performant detector. The classifier formed by the boosting algorithm is called a *strong* classifier. Boosting can be applied to improve the performance of any learning algorithm [FS96]. The implementation used in our experiments is based on the adaptive boosting algorithm (*AdaBoost*) introduced by Freund and Schapire [FS96]. This algorithm is one of the most widely used [Bis07].

More formally, the strong classifier $H(\mathbf{x})$ is defined as

$$H(\mathbf{x}) = \text{sign}\left(\sum_{j=1}^{N} \alpha_j h_j(\mathbf{x})\right),$$

where α_j denotes the weight of the corresponding weak classifier $h_j(\mathbf{x})$. The basic steps of the boosting algorithm are as follows. All samples within a given training set of positive and negative labeled samples are assigned a weight out of an initially uniform distribution. Thus, a sample $\mathbf{x}_i$,

$$\mathbf{x}_i \in \mathcal{X} = \{\langle \mathsf{x}_1, \mathsf{y}_1 \rangle, \ldots, \langle \mathsf{x}_L, \mathsf{y}_L \rangle | \mathsf{x}_i \in \mathbb{R}^m, \mathsf{y}_i \in \{-1, +1\}\},$$

is initialised with

$$p(\mathbf{x}_i) = \frac{1}{L}.$$

At each boosting iteration, a new weak classifier h_j is added to the strong classifier. Based on its error ϵ_j on the training samples, the weak classifier gets a weight α_j, which is defined as

$$\alpha_j = \frac{1}{2} \ln \frac{1 - \epsilon_j}{\epsilon_j}.$$

The probability $p(\mathbf{x}_i)$ for each sample is then increased for misclassified samples and reduced for correctly classified samples. Thus, the algorithm concentrates on hard samples which were still misclassified. These steps are repeated until a stopping condition is met, *e.g.* a given number of weak classifiers has been trained, or the classification error drops below a predetermined threshold. Since the training samples must be available at every stage of the algorithm, it is called *offline*. [FS96]

3.2 Boosting for Feature Selection

Tieu and Viola [TV00] introduced boosting for feature selection. The basic concept is that each weak classifier h_j corresponds to a single feature f_j. The features are drawn from a pool of possible features $\mathcal{F}$. For computational efficiency, the algorithm operates on a subset of all possible features.

The feature selection process works analogical to the standard boosting algorithm. In each iteration j, all features within the pool are evaluated and the best one is selected to form the weak hypothesis h_j. The weight α_j is set accordingly to the error of the hypothesis. Finally, a strong classifier is built by a weighted linear combination of all weak classifiers. [TV00]

The disadvantage of the method proposed by Tieu and Viola is that all samples must be available during each step of the boosting algorithm. Thus, Grabner and Bischof [GB06] introduced an online boosting algorithm for feature selection, which is able to operate on a single sample which can be discarded after updating. They introduce the concept of selectors, where each selector holds a set of M weak classifier. The task of a selector is to choose

a weak classifier out of its set. Similar to the offline method as proposed by Tieu and Viola, each weak classifier corresponds to a single feature from a feature pool. The actual boosting step is performed on the selectors instead of the weak classifiers. Contrary to the offline selection process, the online boosting algorithm allows to extract a strong classifier at any time. [GB06]

In our experiments, we combine the two approaches of offline and online boosting for feature selection. Therefore, the final classifier consists of M selectors with N features per selector. In each iteration of the offline training, the N best-performing weak classifiers are selected out of the feature pool. Additionally, we ensure that the selected features encode different regions of interest. Otherwise, the selected features might be too similar and additional information would be lost. After the training, each selector holds N features, which perform best on the training dataset. However, the best feature of each selector might not perform best on the test sequence. Thus, the classifier adapts to the scene using online feature selection. Therefore, the classifier is updated with hard negative samples out of the scene. During this step, each selector chooses the best-performing feature out of its N offline selected features. Afterwards, the final classifier uses those features which perform best on the scene. [RSGB09]

3.3 Choosing Suitable Weak Classifiers

The performance of boosting algorithms strongly depends on the choice of the weak classifiers. Although effective weak classifiers increase the performance of the final strong classifier, not every classifier can be used due to runtime limitations. For scalar features $f \in \mathbb{R}$, such as Haar features, an optimal decision threshold can be efficiently computed in $O(n \log n)$ time, where n is the number of training samples. However, finding optimal discriminative thresholds for multi-dimensional feature vectors $f \in \mathbb{R}^m$ requires $O(\binom{n}{m})$ time. Thus, finding optimal decision thresholds of complex classifiers such as SVMs or Neural Networks can be inefficient due to the large dimensionality of the feature vectors. [Lap06]

There are several approaches to deal with multi-dimensional feature vectors for boosting. The simplest approach is to use a pre-defined set of functions $g_j : \mathbb{R}^m \to \mathbb{R}$, which project the feature vectors into a set of 1-dimensional manifolds. Although this approach provides an efficient solution, the choice of suitable functions g_j is crucial. Functions which are not well suited for the particular problem cause poor generalization of the final boosted classifier. Hence, Laptev [Lap06] proposes to use *Fisher Linear Discriminant* as an efficient weak classifier for multi-dimensional vectors. This method allows optimal classification of normally distributed samples using a

linear projection based on the class means and covariance matrices. [Lap06]

We concentrate on *K-Means* and the *Nearest Neighbor* approach to boost higher dimensional feature vectors. Additionally, we evaluate an algorithm which performs class separation by calculating the Euclidean distance of the feature vector to a randomly generated vector. The basic idea is that feature vectors of positive samples have a different distance measure to a random vector than negative samples. We will refer to this algorithm as *Distance-to-Random-Vector*. *K-Means* on the other hand separates training samples into K different clusters [Bis07]. Each cluster is defined by the mean vector computed over all samples within the cluster. For classification of a sample, its Euclidean distance to all mean vectors is computed. The algorithm then assigns the label of the cluster with the shortest distance. The advantage of *K-Means* is that it stores only the mean vectors for further classification. In contrast, the *Nearest Neighbor* algorithm stores each training sample [Bis07]. To classify a new sample, its Euclidean distance to every stored sample is computed. The label of the sample with the shortest distance is assigned to the sample to be classified. An evaluation of the different weak classifier types used in our experiments will be discussed in section 4.3.

4 Experiments

We evaluate our classifiers on three pedestrian sequences and two car sequences. For the purpose of feature comparison, we focus on typical surveillance scenarios, where a single stationary camera overviews the whole scene. We compare the feature types using recall, precision and F-measure. Recall indicates how many of the true positive objects are detected. Precision denotes the fraction of correctly classified objects. F-measure is the harmonic mean of recall and precision. More formally, recall, precision, and F-measure are defined as

$$\begin{aligned} Recall &= \frac{\#tp}{\#objects}, \\ Precision &= \frac{\#tp}{\#tp + \#fp}, \\ F\text{-}measure &= 2\frac{Recall \cdot Precision}{Recall + Precision}, \end{aligned}$$

where $\#tp$ denotes the number of correctly detected objects (true positives), $\#fp$ the number of misclassified detections (false positives), and $\#objects$ indicates the number of annotated objects within the scene.

Figure 8: Samples within the pedestrian dataset. This dataset contains patches of upright standing or walking humans with varying poses (left), as well as optical flow patches to train motion-based features (right).

Figure 9: Sample patches within the car dataset. The patches have been extracted from image frames provided by highway surveillance cameras. The dataset contains patches showing the rear view of cars on highways (left), as well as pre-computed optical flow patches (right).

The remainder of this section is organised as follows. First, we briefly review the training datasets, followed by a discussion of implementation details. Next, we evaluate different classification algorithms. Afterwards, we visualise and discuss the evaluated feature representations. Next, we present the results for pedestrian and car detection. Finally, we analyse the runtime performance of all evaluated feature types.

4.1 Training Datasets

We use two datasets for training the classifiers throughout our experiments. The first dataset is used for training pedestrian detectors, while the second dataset contains samples of cars.

The pedestrian dataset contains 1276 positive and 6946 negative samples of size 32×64. Additionally, this dataset holds 610 positive and 966 negative optical flow patches. These patches have been pre-computed in order to train motion-based features efficiently. Figure 8 shows examples of positive patches within the pedestrian dataset.

The car dataset consists of 1752 positive and 13188 negative examples of size 50×50, as well as 1330 positive and 7946 negative optical flow patches used to train motion-based features. All patches show the rear view of cars on highways. Figure 9 shows examples of positive car patches within this dataset.

4.2 Implementation Details

The feature selection during classifier training operates on a pool of randomly generated features. In each iteration of the boosting algorithm, 15,000 random features of the pre-determined types are generated. Out of this feature pool, each selector chooses the best-performing features. We find that in practice a pool size of 15,000 is enough for our classifiers, as the patch size is only 32×64 for pedestrian detection, respectively 50×50 for car detection.

To reduce the chance of overfitting, we bootstrap the trained features in each offline training iteration. After training all features within the pool using positive and negative samples, the best-performing features are chosen by the selectors. Next, these features are evaluated on all negative samples within the dataset. Misclassified samples are used in the next iteration as hard samples. Thus, the bootstrapping concentrates on hard negative samples.

In order to adapt the classifiers to the scene, we bootstrap our offline trained classifiers on frames of the current sequence using online feature selection. Therefore, a classifier is evaluated on scene-specific hard negative samples. Misclassified samples are used as negative updates to re-train the classifier in order to learn a scene-specific model.

We increase the detection speed by taking advantage of *a priori* scene knowledge, *i.e.* the ground plane calibration. If no specific scene knowledge is available, the general approach is to scan each image location at multiple scales for the objects to classify. However, by knowing the ground plane calibration, the required patch size at each image location can be predicted. This allows to build a grid of highly overlapping patches of the correct size. Since the detector doesn't have to evaluate each image location at different scales, this obviously increases its performance. Thus, our detectors use the scale information of each scene to create a scene-specific patch grid. Figure 10 illustrates the advantage of available scene calibration data. [SRGB09]

In order to find a suitable detector setup, we compare classifiers with 100 to 200 selectors against smaller ones. For this comparison, all classifiers are trained with Haar features and evaluated on the corridor sequence, which will be detailed in section 4.5.1. The grid overlap is kept fixed at 92%. It is noteworthy that all tested classifiers provided similar results. Table 1 lists some results of the comparison. Training large classifiers requires additional computational effort. However, since the provided results are almost the same as for smaller classifiers, we will compare the feature representations using a classifier consisting of 30 selectors and 20 weak classifiers within each selector. Additionally, the grid overlap is kept fixed at 92% throughout all experiments.

For optical flow estimation, we use an implementation based on the

(a) (b)

Figure 10: Gaining speed-up using scene knowledge. As shown in figure 10(a), a detector without knowledge of the scene calibration has to scan the image at several scales at every position. If this calibration information is available, the detector can estimate the correct patch size for each position in the input frame as shown in figure 10(b). Illustration based on [SRGB09].

anisotropic *Huber-L^1* optical flow estimation proposed by Werlberger *et al.* [WTP+00]. This method uses the robust *Huber-L^1* norm for regularization of the flow estimates. The resulting flow gradients are more distinct than flow estimates computed by isotropic regularization techniques [WTP+00].

The covariance features used throughout the experiments are calculated over the correlation of pixel locations and first order image gradients. Thus, the information encoded by these features can be compared to that encoded by HOG features to some extend. Our implementation of covariance features does not include the information provided by the different color channels, since the training datasets only contain grey value images.

4.3 Evaluation of Weak Classifiers

Most of the evaluated feature representation provide multi-dimensional vector results. In fact, only the value of Haar features is scalar. Since boosting of multi-dimensional feature vectors is a challenging task, we evaluate different algorithms to be used as weak classifiers.

In our experiments we compare *K-Means*, *Nearest Neighbor*, and the *Distance-to-Random-Vector* approach. To guarantee a fair comparison, each strong classifier contains 30 selectors, with 20 weak classifiers of the corresponding type per selector. All classifiers are evaluated on the corridor sequence, which will be discussed in section 4.5.1. The grid overlap is kept fixed at 92%. We select covariance features to represent multi-dimensional

Table 1: Performance of several detectors with different numbers of selectors and weak classifiers based on Haar features. The abbreviations are as follows. S stands for the number of selectors of the strong classifier and H_{weak} gives the number of weak classifiers per selector. B indicates whether or not bootstrapping has been applied.

S	H_{weak}	B	Recall	Precision	F-measure
30	20	×	88	96	92
30	40	×	85	98	91
40	25	×	85	95	90
100	10	×	90	90	90
150	20	×	80	93	86
150	50	×	78	81	80
200	20	×	77	92	84
20	50		91	43	58
100	50		85	40	54
150	20		89	41	57
200	20		86	42	56

Table 2: Comparison of classification algorithms. For this evaluation, all classifiers were trained with covariance features.

Weak Classifier Type	Recall	Precision	F-measure
K-Means	77	87	82
Distance to Random Vector	79	82	80
Nearest Neighbor	54	81	65

vectors. Table 2 lists results of the different weak classifiers. *K-Means* performs slightly better than the other algorithms. Thus, we use *K-Means* as weak classifiers throughout the evaluation of the feature representations.

4.4 Feature Visualisation

For a better understanding of the image information encoded by the different feature types, we will now discuss the location and form of the best-performing features of the corresponding types. Haar features encode simple image structures like edges and bars. This can be seen in Figure 11. For the task of human detection, both two-rectangle and three-rectangle Haar features are selected. However, three-rectangle Haar features are not drawn from the feature pool for car detection. This can be explained by the fact, that three-rectangle features mostly encode bars, *e.g.* human limbs. Silhou-

Figure 11: Visualisation of selected Haar features. The top row shows features for pedestrian detection, while features for car detection are depicted in the bottom row. Three-rectangle Haar features encode line structure like human limbs. However, this feature type is not drawn from the feature pool for car detection. Two-rectangle Haar features on the other hand are used to model edges along the silhouettes of the persons and the contours of the cars.

ettes on the other hand are better modelled by two-rectangle Haar features.

Figure 12 illustrates the locations of highly discriminative LBP features. For this visualisation, the best-performing feature of each selector has been chosen. We calculate the feature value by computing the LBP codes for each pixel inside the illustrated regions. Afterwards, these codes are used to vote into the bins of a histogram, which is used as feature response. LBPs encode texture information. Thus, the features are located near areas with distinctive textures such as legs. The human's upper torso also forms a very distinctive shape. Thus, several LBP features concentrate on blocks that are located there. A closer look upon the location of the LBP features for car detection shows that bumpers in combination with the rear wheels, the rear area containing the back lights, and the rear window provide good features. These elements are also the most characteristic ones for describing the rear view of a car.

Locations of covariance features are illustrated in Figure 13. For pedestrian classification, these features concentrate on areas containing the upper torso, shoulders, and head contours. Regions containing the feet don't seem to be distinctive enough to be selected as good cues. Covariance features for car detection mostly encode regions containing the roof.

Figure 14 shows the locations of HOG features that achieved good performance for classification. As can be seen, the HOG features encode gradient information over areas containing strong edges. This feature type clearly concentrates on characteristical silhouette information for both pedestrian and car patches.

The location of HOF features is illustrated in Figure 15. The illustrated

Figure 12: Visualisation of regions of interest for selected LBP features. The texture measures are spread all over the objects of interest. Highly discriminative features for pedestrian detection are computed across blocks that contain the upper torso, the change from body to feet, and contours. Strong features for car detection contain textures like rear lights, bumpers, and back windows.

Figure 13: Locations of regions of interest for selected covariance features. These features seem to concentrate on areas with transitions from background to the objects of interest.

Figure 14: Visualisation of locations of selected HOG features. These features concentrate on areas with highly discriminative edges, such as feet and contour information of the torso for human detection and distinctive silhouette edge structures for car detection.

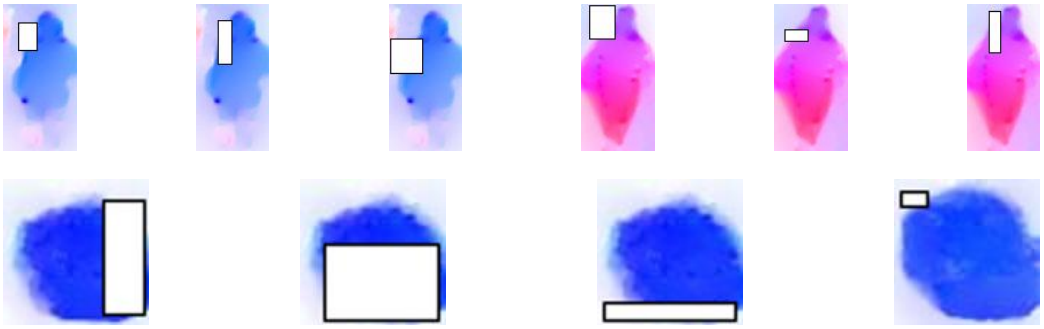


Figure 15: Locations of selected HOF features for pedestrian detection (top row) and car detection (bottom row). For pedestrian detection, the features are located on the upper half of the patches, since the motion boundaries at the bottom are more blurred. For car detection, the features are located at the right hand side and bottom area where the motion boundaries are less blurred due to the camera angle.

locations show Motion Boundary Histograms. These features seem to operate well on strong motion boundaries. Thus, areas with blurred motion edges do not provide good classification cues. If a person is walking, the motion boundaries within the feet area gets blurred. However, the upper torso usually remains reasonably steady and thus provides strong edges within the flow components. Therefore, MBH features at the upper torso of a person provide stronger cues for pedestrian matching. For the task of car detection, the MBH features also concentrate on strong motion boundaries. As can be seen from the illustrated locations of good feature regions, the motion edges at the bottom and at the right side of the patches provide better information than the edges at the top. This can be explained by the fact that the cars move away from the camera and thus, the motion boundaries closer to the camera provide stronger cues.

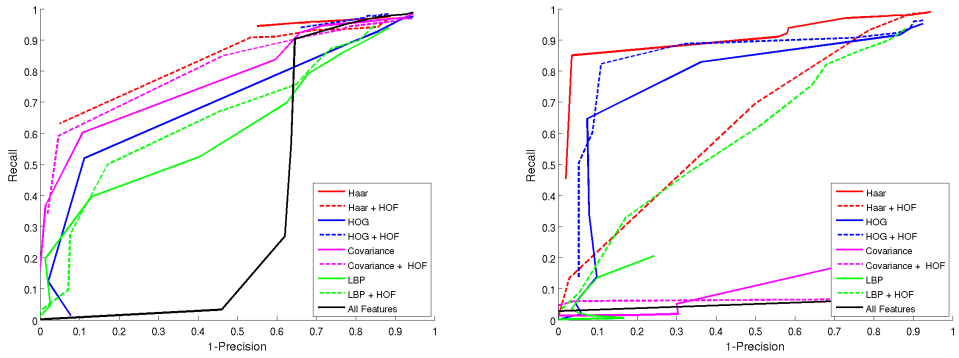
4.5 Pedestrian Detection Results

This section presents the results of our classifiers on three pedestrian sequences.

4.5.1 Corridor Sequence

The first dataset for evaluation is the corridor sequence as used by Sternig *et al.* [SRB10]. This scene shows pedestrians walking across a hallway in

Figure 16: Sample detections on the corridor sequence.



(a) Without bootstrapping.

(b) With bootstrapping.

Figure 17: RPC plot of classifiers on the corridor sequence. Values located top left are best.

a public building. It contains 901 frames of dimension 480×640. Figure 16 shows sample detections of our classifiers. Table 3 lists the results of the evaluated classifiers. The plot of the Recall-Precision-Curves (RPC) of the different feature types can be found in Figure 17.

Since optical flow estimation operates on consecutive image frames, we extracted the optical flow patches contained within the pedestrian training dataset from this sequence. As the last 300 frames show pedestrians moving in arbitrary directions, we used this part of the sequence to extract the optical flow patches. Thus, the classifiers are evaluated on the first 600 frames in order to guarantee a fair comparison.

Haar features clearly outperform all other feature types on this sequence. Bootstrapping provides good improvement for Haar and HOG features, while

Table 3: Results on the corridor sequence. B indicates whether or not the classifier has been adapted to the scene using bootstrapping.

Feature Types	B	Recall	Precision	F-measure
Haar	×	85	97	90
HOG + HOF	×	82	89	86
Haar + HOF		63	95	76
HOG	×	65	93	76
Covariance + HOF		59	95	73
Covariance		60	89	72
HOG		52	89	66
LBP + HOF		50	83	62
Haar		94	45	61
Haar + HOF	×	70	50	58
LBP + HOF	×	48	66	56
LBP		52	60	56
All Features		90	35	51
HOG + HOF		94	34	50
LBP	×	21	76	32
HOF		62	17	27
All Features	×	27	19	23
Covariance	×	24	21	22
Covariance + HOF	×	6	96	12

covariance and LBP features perform better without bootstrapping. Since the corridor sequence is similar to the training samples, LBP features cannot be improved by additional bootstrapping. The results verify that motion features alone cannot be used to build a classifier which performs reasonably well. However, combining other feature types with HOF representations allows to build detectors with good performance. The combination of HOG and HOF features provides better improvement than combining HOF features with other feature types on this sequence.

We train a detector based on all feature types in order to analyse which features are selected after the training and bootstrapping process. Therefore, we create a feature pool of $50,000$ randomly generated features, where all feature types are uniformly distributed. After offline training, we analyse the distribution of all feature types within the classifier, as well as the 30 selected features, which are used for detection. Thus, we can identify the best-performing feature types on the training dataset. Afterwards, we bootstrap the classifier with hard negative samples from the scene and evaluate

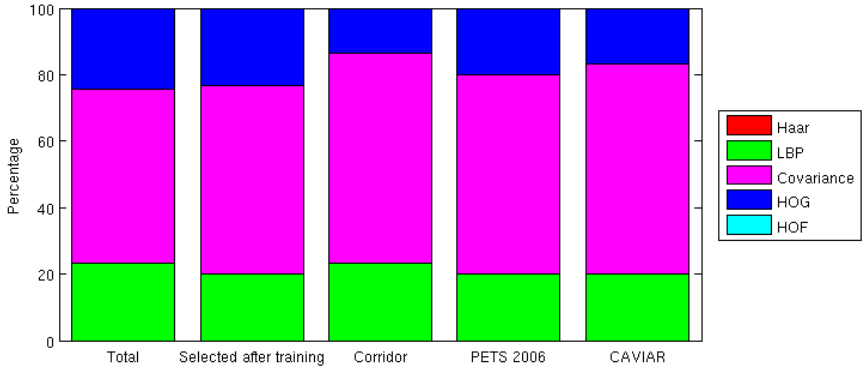


Figure 18: Feature distribution of pedestrian detectors. The first bar shows the percentage of features within the classifier. The second bar shows the percentage of feature types among the 30 best-performing features after the offline training. The remaining bars indicate the percentage among the 30 best-performing features after bootstrapping on the corresponding pedestrian sequence.

the distribution of the selected feature types again. This allows to determine whether the selected features also perform best on the scene. Figure 18 shows the distribution of the feature types. As can be seen, LBP, HOG, and covariance features perform best throughout the training. In contrast, features with good performance on the scene, *i.e.* Haar, are not present in the trained classifier, as these perform worse on the training dataset. Hence, the bootstrapped classifier consists mostly of covariance and LBP features, which are obviously not the best features for the corridor sequence.

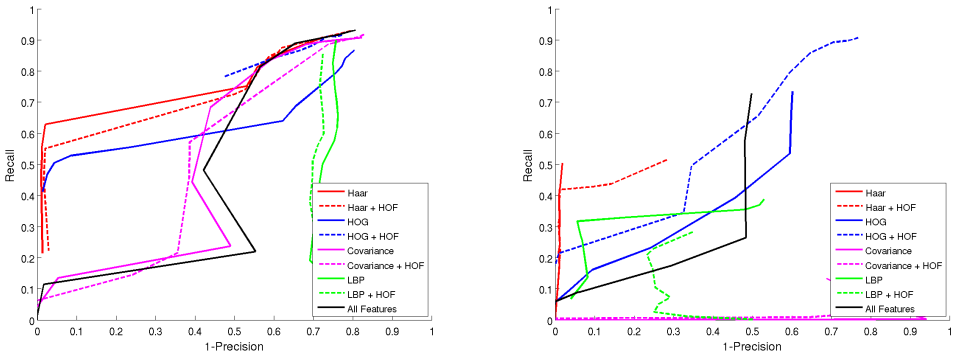
4.5.2 PETS 2006

We use an image sequence from the publicly available PETS 2006 dataset[1] as standard benchmark. The dataset contains 308 frames of dimension 720×576. The scene shows people moving across the platform of a railway station. Figure 19 shows sample detections of the tested classifiers. Table 4 lists the classification results. RPC plots are shown in Figure 20.

As can be seen from the results, this scene is more complex than the corridor sequence. There are many additional structures in the background such as patterns on the floor, seats, and bars at the barrier which separates the platform from the trains. Additionally, seats and crowds cause occlu-

[1] http://www.pets2006.net, June 1, 2010.

Figure 19: Sample detections on the PETS 2006 sequence. The camera overviews a platform of a railway station.



(a) Without bootstrapping.

(b) With bootstrapping.

Figure 20: RPCs of classifiers on the PETS 2006 dataset.

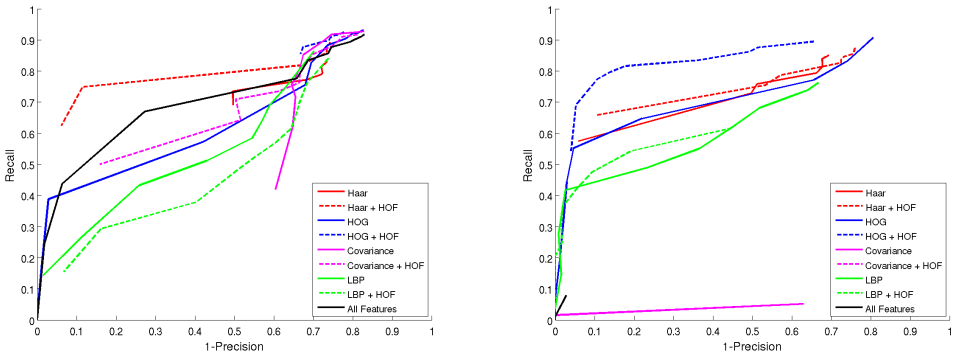
sions, which impose additional challenges. Haar and HOG features again perform best on this scene. Covariance and LBP features cannot achieve good performance. Motion cues in combination with other features achieve less improvement than on the corridor sequence. This can be explained by the fact that the persons move in different directions, which were not covered completely by the flow patches within the training dataset. Bootstrapping does not seem to improve the detection performance on this sequence. Instead, the confidence drops and thus the detectors achieve less recall, which leads to lower performance than without bootstrapping.

The feature distribution of the classifier trained on all feature types is illustrated in Figure 18. Since the selected features after training also outperform the other features available within the classifier, the distribution of the feature types does not change significantly after bootstrapping. The detector performs similar to the detector trained with covariance features only. This can be explained by the high percentage of selected covariance features.

Table 4: Results on the PETS 2006 sequence.

Feature Types	B	Recall	Precision	F-measure
Haar		63	98	77
Haar + HOF		55	98	71
Haar	×	50	98	67
HOG		53	91	67
HOG + HOF		78	52	63
Covariance		68	56	62
Haar + HOF	×	52	71	60
Covariance + HOF		57	61	59
All Features	×	73	50	59
All Features		67	50	57
HOG + HOF	×	50	65	56
HOG	×	74	40	52
LBP	×	32	94	48
LBP + HOF		86	28	42
LBP + HOF	×	29	65	40
LBP		83	25	38
HOF		69	22	33
Covariance	×	13	31	19
Covariance + HOF	×	11	17	13

Figure 21: Sample detections on the CAVIAR sequence. The pedestrians within the region above the white line are smaller than the trained patches. Thus, this region has not been used for evaluation.



(a) Without bootstrapping.

(b) With bootstrapping.

Figure 22: RPC plot of classifiers on the CAVIAR sequence.

4.5.3 CAVIAR

We use a sequence from the publicly available CAVIAR dataset[2] as an additional standard benchmark. The scene shows the hallway of a shopping centre in Lisbon. The image sequence contains 370 frames of dimension 384×288. Figure 21 shows sample detections of our classifiers. The results are summarized in Table 5. Figure 22 shows the RPCs of the evaluated classifiers.

On this dataset, HOG features outperform the other feature types, followed by Haar features. The major challenges of this sequence are the varying scale and the camera angle. On the one hand, pedestrians at the far end of

[2] http://homepages.inf.ed.ac.uk/rbf/CAVIARDATA1, June 1, 2010.

Table 5: Results on the CAVIAR sequence.

Feature Types	B	Recall	Precision	F-measure
HOG + HOF	×	77	89	83
Haar + HOF		75	90	82
Haar + HOF	×	66	89	76
Haar	×	58	94	71
HOG	×	65	78	71
All Features		66	71	68
LBP + HOF	×	54	81	65
HOG		56	72	63
Covariance + HOF		49	83	62
Haar		81	50	62
LBP	×	49	77	60
LBP + HOF		69	43	53
LBP		40	71	51
HOG + HOF		85	36	51
Covariance		75	34	51
HOF		63	28	39
Covariance	×	5	37	9
Covariance + HOF	×	1	100	2
All Features	×	8	97	15

the hallway are very small and show almost no distinctive structures. On the other hand, the camera angle in combination with the frequency of pedestrians causes people in the foreground to occlude people in the background. If the overlap is too large, the detectors cannot distinguish between all individuals and thus, some detections are missed. Similar to the other evaluated pedestrian sequences, covariance and LBP features achieve a lower performance than Haar and HOG features. Bootstrapping allows to improve the performance of HOG, LBP, and Haar features. However, the performance of covariance features drops dramatically after scene-specific negative updates. The combination of motion-based and appearance-based features achieves the best results for HOG and HOF features. It is noteworthy that the optical flow estimates are less accurate than on the other sequences, since the dataset contains only every tenth frame of the surveillance camera. However, this does not cause a lower performance of the motion-based features. This can be explained by the fact that HOF features do not require precise optical flow estimates, as shown by Dalal *et al.* [DTS06]. Also, the movement of pedestrians within the CAVIAR sequence resembles the motion of pedestri-

Figure 23: Sample detections on the first highway sequence. Since we trained our detectors with patches showing the rear end of cars, we evaluate our classifiers only on the right lane. The white box shows the boundaries of the detection region.

ans within the corridor sequence, from which the training patches for the motion-based features have been extracted.

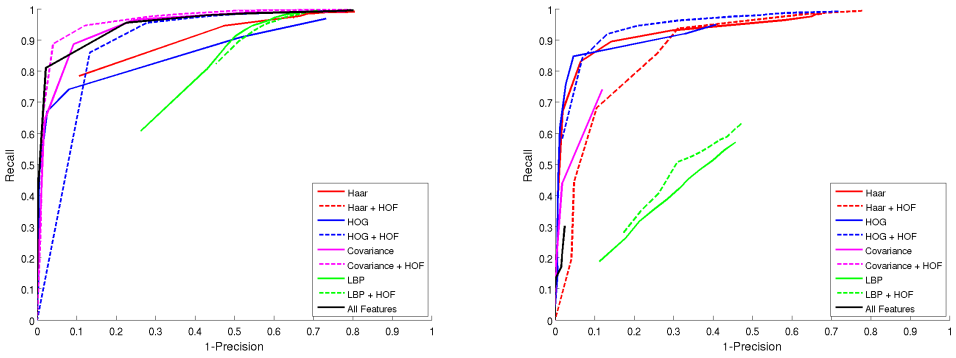
4.6 Car Detection Results

This section presents the results of our classifiers on two highway sequences.

4.6.1 Highway 1

The first image sequence for evaluation of car classifiers consists of $2,000$ frames of dimension 704×576. This scene has been taken from a highway surveillance camera. Rapidly changing illumination and shadowing effects make up the challenge of this scene. Figure 23 shows sample detections, whereas Table 6 lists the results of the evaluated classifiers. Figure 24 shows the RPCs of the different feature types on this sequence.

The optical flow patches contained in the dataset of car samples have been extracted from the first 450 frames of this sequence. Thus, evaluation starts at frame 450 to guarantee a fair comparison. The results show that car detection is an easier task than pedestrian detection. This can be explained by several facts. First, there is almost no variation of the shape of cars. Given a type of cars, *e.g.* passenger cars or estate cars, all objects of this type have the same characteristic shape. Second, the cars within a highway surveillance sequence are moving in the same direction. Thus, there is only monotonic motion, whereas pedestrians can move arbitrarily across the scene.



(a) Without bootstrapping.

(b) With bootstrapping.

Figure 24: RPC plot of classifiers on the first highway sequence.

Table 6: Results on the first highway sequence.

Feature Types	B	Recall	Precision	F-measure
Covariance + HOF		89	96	92
HOG	×	85	95	90
Covariance		89	91	90
All Features		81	98	89
HOG + HOF	×	92	87	89
Haar	×	83	94	88
HOG + HOF		86	87	86
Haar		79	89	84
HOG		74	92	82
Covariance	×	74	88	80
Haar + HOF	×	86	74	80
LBP		80	57	67
LBP + HOF		82	55	66
LBP + HOF	×	51	69	58
LBP	×	55	57	56
HOF		74	44	55
Haar + HOF		97	35	52
All Features	×	30	98	46
Covariance + HOF	×	7	100	13

Almost all feature types achieve good performance on this sequence. Even illumination changes and moving shadows do not reduce the accuracy of the detectors. However, LBP features perform worst compared to the other feature representations. A possible explanation is that in general, the texture of cars is weaker than the texture of other object classes such as pedestrians. Bootstrapping can be applied to improve the performance of Haar and HOG features. However, as it is also the case at pedestrian detection, bootstrapping of LBP and covariance features leads to lower performance. It is also noteworthy that HOF features perform much better on car detection compared to pedestrian detection. This is mainly caused by the fact that the cars move in the same direction with almost constant speed. However, the results verify that only motion-based features are not enough to build a well performing car detector.

Figure 25 illustrates the distribution of feature types within a detector trained on all feature types. Since MBH motion features perform better on car detection tasks [DTS06], these features are also among the best-performing features after training and bootstrapping. Similar to the pedestrian detector with all feature types, Haar features are not selected due to the larger error on the training set. Nevertheless, contrary to the pedestrian detector trained on all feature types, the car detector achieves a better performance, since covariance and LBP features, as can be seen from the results, perform better on car sequences than on pedestrian scenes.

4.6.2 Highway 2

The second highway surveillance scenario contains $1,000$ frames of dimension 380×324. Sample detections are illustrated in Figure 26. Table 7 lists the detection results. Figure 27 shows the RPCs of the different feature types.

The classifiers were trained on patches showing the rear view of cars. Thus, the slightly different camera angle causes additional challenges. However, HOG and Haar features perform reasonably well. On this scene, Haar features in combination with motion-based features perform best. Bootstrapping works best for the combination of Haar and HOF features and Haar features alone. Contrary to the first highway sequence, bootstrapping improves the performance of LBP features. However, the overall performance of LBP features is still not enough to build a robust car detector.

The feature distribution of the detector trained on all feature types is illustrated in Figure 25. As can be seen, bootstrapping has only little effect on the selection of the best-performing feature types. The bootstrapped detector still holds a large number of LBP features, which is also a reason for the lower performance of the detector on this dataset.

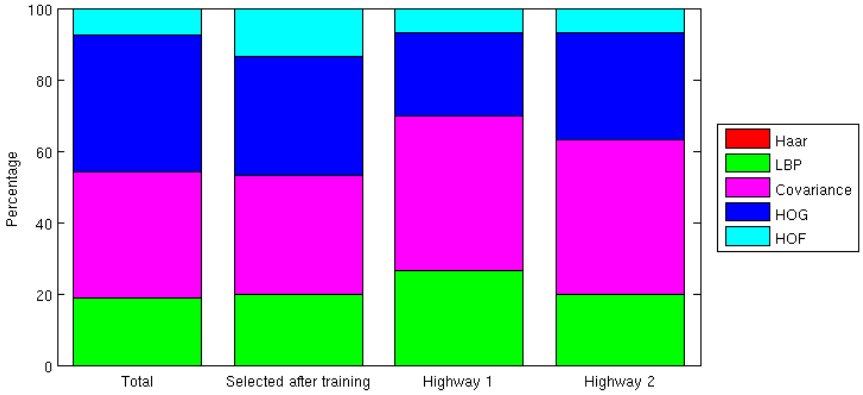
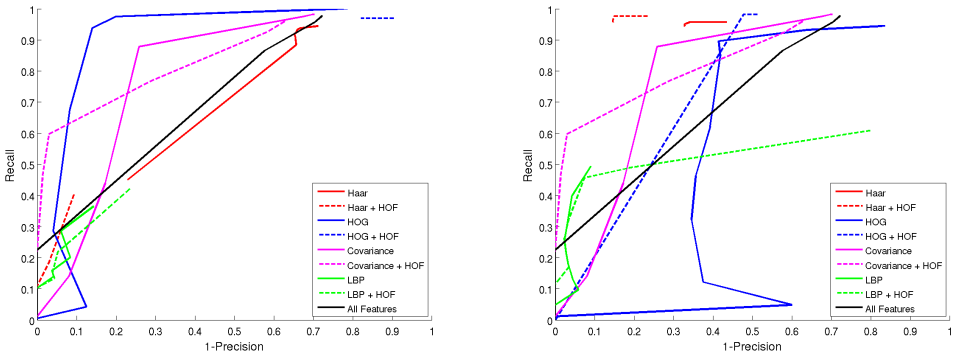


Figure 25: Feature distribution of car detectors. The first bar shows the percentage of features within the classifier. The second bar shows the percentage of feature types among the 30 best-performing features after the offline training. The remaining bars indicate the percentage among the 30 best-performing features after bootstrapping on the corresponding car sequence.

Figure 26: Sample detections on the second highway sequence. The white box shows the boundaries of the detection region.



(a) Without bootstrapping.

(b) With bootstrapping.

Figure 27: RPC plot showing the different classifiers on the second highway sequence.

Table 7: Results on the second highway sequence.

Feature Types	B	Recall	Precision	F-measure
Haar + HOF	×	98	85	91
HOG		94	86	90
Covariance	×	88	74	80
Covariance		88	74	80
Haar	×	95	67	79
Covariance + HOF	×	60	97	74
Covariance + HOF		60	97	74
HOG	×	90	59	71
HOG + HOF	×	98	52	68
LBP	×	49	91	64
LBP + HOF	×	46	93	61
Haar		45	77	57
All Features	×	87	42	57
All Features		87	42	57
Haar + HOF		41	91	56
LBP + HOF		43	76	55
LBP		37	86	51
HOF		30	41	35
HOG + HOF		97	18	30

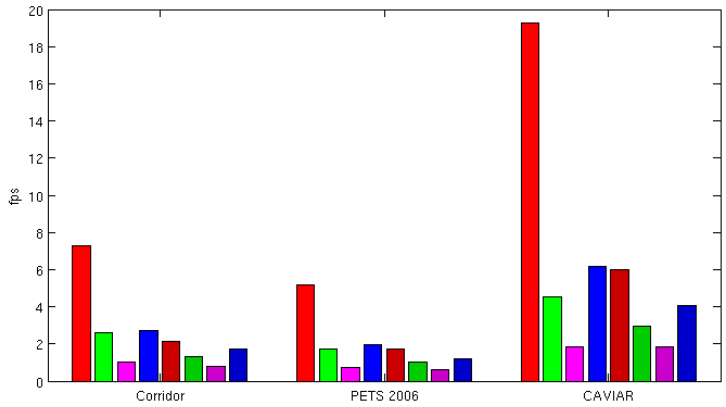
Table 8: Frame rates achieved with the different feature types on the evaluated sequences. The runtime measure is frames per second (fps). The abbreviation *HW* stands for "highway".

Feature Type	Corridor	PETS 2006	CAVIAR	*HW* 1	*HW* 2
Haar	7.27	5.17	19.28	5.45	18.84
LBP	2.59	1.75	4.54	1.32	4.33
Covariance	1.00	0.71	1.87	0.92	2.87
HOG	2.70	1.95	6.19	2.56	8.51
Haar + HOF	2.14	1.70	5.97	1.28	4.13
LBP + HOF	1.34	1.01	2.94	0.90	2.98
Covariance + HOF	0.81	0.60	1.83	0.56	1.77
HOG + HOF	1.71	1.22	4.07	1.07	3.31
Number of patches	756	1,110	525	2,276	703

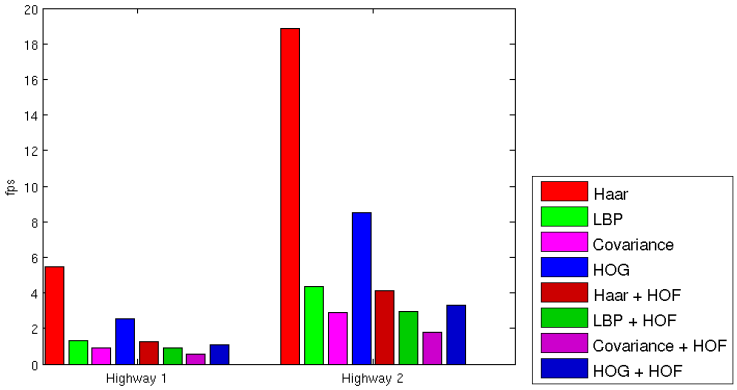
4.7 Runtime Evaluation

To compare the feature representations in terms of runtime efficiency, we calculate frame rates of each feature type on the evaluation datasets. Since the complexity of the feature type, as well as the dimensions of the input images and the chosen grid overlay cause the frame rates to vary, we compare the frame rates on each image sequence. The tests were run on an Intel®Core™2 Duo T9550 CPU at 2.66 GHz with 2 GB RAM. Since the setup does not contain a GPU, we pre-compute the optical flow estimates in order to achieve competitive frame rates for motion-based features. To guarantee a fair comparison, we train a classifier with 30 selectors and 20 features of a specific type per selector. The grid overlap is kept fixed at 80%. The achieved frame rates on the pedestrian and car sequences are listed in Table 8 and illustrated in Figure 28.

Haar features are obviously the most efficient feature types. More complex features, which operate on histograms such as HOGs and LBPs, require much more computational effort. Thus, the frame rates of detectors using these features is significantly lower. The results show that the computation of covariance features is the most time-consuming due to the complex handling of covariance matrices and the approximation on Euclidean space. It is noteworthy, that the additional computational effort for combining motion-based features with complex feature types is quite low, *i.e.* there is only a small decrease of the frame rate. Higher frame rates can be achieved when using the combination of motion-based and appearance-based features on fast hardware.



(a) Frame rates on pedestrian sequences.



(b) Frame rates on car sequences. (c) Color key.

Figure 28: Frame rates.

5 Conclusion

In this thesis we evaluated common feature representations for pedestrian and car detection, *i.e.* Haar, LBP, covariance, and HOG features. We also combined these appearance-based features with motion-based HOF features to analyse the effect of additional motion cues. For pedestrian detection, the features were evaluated on a corridor sequence, a standard PETS 2006 sequence and a sequence of the CAVIAR dataset. Car detection results were compared on two highway surveillance scenarios.

We used boosting to build the detectors for comparison. In order to select the best-performing features, we applied boosting for feature selection [GB06]. Since all feature types except for Haar features provide multi-dimensional feature responses, we evaluated various algorithms as weak classifiers. In our experiments, *K-Means* performs slightly better than the *Near-*

est Neighbor and *Distance-to-Random-Vector* approach. In order to determine a suitable classifier size, we compared small, moderately sized, and large detectors. Since large detectors with 100 to 200 selectors achieve almost the same performance as the smaller detectors with only 20 to 50 selectors. Thus, the experimental results do not encourage the use of very large classifiers since they require additional computational effort during training and take much more time to evaluate on the datasets. The best-performing classifier in our experiments uses 30 selectors with 20 weak classifiers per selector.

Our results show that Haar and HOG features outperform LBP and covariance features for both pedestrian and car detection. Haar features achieve the best performance on most pedestrian sequences, while HOG features perform best on car sequences. Covariance features computed over location and first order image gradients do not perform well on pedestrian sequences, while these features achieve considerably good performance on car sequences similar to HOGs. LBP features seem to generalize worse than the other feature types, although the error of LBP features on the training set is considerably low. Bootstrapping with hard negative samples taken from a scene can obviously improve the performance of Haar and HOG features. However, bootstrapping detectors which already achieve a very good performance does not improve the results further. LBP and covariance features on the other hand seem to perform best without being bootstrapped on the scene.

We evaluated *Motion Boundary Histograms*, which are a subtype of the motion-based HOF features. The experimental results show that motion-based features alone cannot achieve a good detection performance. However, the combination of motion-based and appearance-based features can improve the overall performance. Detectors based on such a combination perform obviously best on sequences where the motion of the objects resembles the motion of the trained patches to a large extent. Since our dataset contains only a limited number of different motion directions, motion-based features achieve a lower performance on more complex sequences, where the objects may move arbitrarily across the scene. Although *Motion Boundary Histograms* are suited better for car detection [DTS06], the results on pedestrian sequences are notably well. The results so far encourage further progress at combining appearance-based features with additional motion cues in order to build robust object detectors.

6 Acknowledgments

I would like to express my gratitude to my supervisor, Sabine Sternig, for her help and guidance. Her expertise and technical support added considerably to my experience. I also appreciate her patience and assistance at writing this thesis very much.

I would also like to thank Peter Roth for taking time to proofread my thesis and to provide helpful annotation. Finally, I want to thank Martin Hirzer for helpful discussion and his support at customizing the framework.

References

[Bis07] Christopher M. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2007. 12, 15

[Cro84] Franklin Crow. Summed Area Tables for Texture Mapping. In *International Conference on Computer Graphics and Interactive Techniques*, 1984. 3

[Dal06] Navneet Dalal. *Finding People in Images and Videos*. PhD thesis, Institut National Polytechnique de Grenoble, 2006. 8, 9, 10, 11

[DT05] Navneet Dalal and Bill Triggs. Histograms of Oriented Gradients for Human Detection. In *International Conference on Computer Vision and Pattern Recognition*, 2005. 7, 9

[DTS06] Navneet Dalal, Bill Triggs, and Cordelia Schmid. Human Detection using Oriented Histograms of Flow and Appearance. In *European Conference on Computer Vision*, 2006. 1, 2, 9, 10, 11, 12, 29, 32, 37

[FS96] Yoav Freund and Robert E. Schapire. Experiments with a New Boosting Algorithm. In *International Conference on Machine Learning*, 1996. 12, 13

[GB06] Helmut Grabner and Horst Bischof. On-line Boosting and Vision. In *International Conference on Computer Vision and Pattern Recognition*, 2006. 1, 12, 13, 14, 36

[GMN+98] Ben Galvin, Brendan McCane, Kevin Novins, David Mason, and Steven Mills. Recovering Motion Fields: An Evaluation of Eight Optical Flow Algorithms. In *British Machine Vision Conference*, 1998. 10

[HCS+09] Xiaopeng Hong, Hong Chang, Shiguang Shan, Xilin Chen, and Wen Gao. Sigma Set: A Small Second Order Statistical Region Descriptor. In *International Conference on Computer Vision and Pattern Recognition*, 2009. 6, 7

[HPH04] Marko Heikkilä, Matti Pietikäinen, and Janne Heikkilä. A Texture-Based Method for Detecting Moving Objects. In *British Machine Vision Conference*, 2004. 5

[HS81] Berthold K.P. Horn and Brian G. Schunck. Determining Optical Flow. *Artificial Intelligence*, 1981. 10

[KMB09] Stefan Kluckner, Thomas Mauthner, and Horst Bischof. A Covariance Approximation on Euclidean Space for Visual Tracking. In *Workshop of the Austrian Association for Pattern Recognition*, 2009. 6, 7

[Lap06] Ivan Laptev. Improvements of Object Detection Using Boosted Histograms. In *British Machine Vision Conference*, 2006. 14, 15

[LM02] Rainer Lienhart and Jochen Maydt. An Extended Set of Haar-Like Features for Rapid Object Detection. In *International Conference on Image Processing*, 2002. 2, 3, 4

[OPH96] Timo Ojala, Matti Pietikäinen, and David Harwood. A Comparative Study of Texture Measures with Classification based on Feature Distributions. *Pattern Recognition*, 1996. 4, 5

[POP98] Constantine Papageorgiou, Michael Oren, and Tomaso Poggio. A General Framework for Object Detection. In *International Conference on Computer Vision*, 1998. 2

[RSGB09] Peter M. Roth, Sabine Sternig, Helmut Grabner, and Horst Bischof. Classifier Grids for Robust Adaptive Object Detection. In *International Conference on Computer Vision and Pattern Recognition*, 2009. 14

[SHB07] Milan Sonka, Vaclav Hlavac, and Roger Boyle. *Image Processing, Analysis, and Machine Vision*. Cengage Learning, 3rd edition, 2007. 2

[SRB10] Sabine Sternig, Peter M. Roth, and Horst Bischof. Inverse Multiple Instance Learning for Classifier Grids. In *International Conference on Pattern Recognition*, 2010. 22

[SRGB09] Sabine Sternig, Peter M. Roth, Helmut Grabner, and Horst Bischof. Robust Adaptive Classifier Grids for Object Detection from Static Camera. In *Computer Vision Winter Workshop*, 2009. 17, 18

[TPM06] Oncel Tuzel, Fatih Porikli, and Peter Meer. Region Covariance: A Fast Descriptor for Detection and Classification. In *European Conference on Computer Vision*, 2006. 5, 6

[TV00] Kinh Tieu and Paul Viola. Boosting Image Retrieval. In *International Conference on Computer Vision and Pattern Recognition*, 2000. 13

[VJ02] Paul Viola and Michael Jones. Robust Real-time Object Detection. *International Journal of Computer Vision*, 2002. 1, 2, 3, 4

[VJS03] Paul Viola, Michael Jones, and Daniel Snow. Detecting Pedestrians Using Patterns of Motion and Appearance. In *International Conference on Computer Vision*, 2003. 2

[WS08] Christian Wojek and Bernt Schiele. A Performance Evaluation of Single and Multi-feature People Detection. In *Symposium of the German Association for Pattern Recognition*, 2008. 1

[WTP+00] Manuel Werlberger, Werner Trobin, Thomas Pock, Andreas Wedel, Daniel Cremers, and Horst Bischof. Anisotropic Huber-L1 Optical Flow. In *British Machine Vision Conference*, 200. 18

[WWS09] Christian Wojek, Stefan Walk, and Bernt Schiele. Multi-Cue Onboard Pedestrian Detection. In *International Conference on Computer Vision and Pattern Recognition*, 2009. 12