Master's Thesis

LEARNING OBJECT DETECTORS FROM MULTIPLE CAMERAS BY CENTRALIZED INFORMATION FUSION

Armin Berger

Graz University of Technology
Erzherzog-Johann-Universität

Institute for
Computer Graphics and Vision

Supervisor:
Univ.-Prof. Dipl.-Ing. Dr.techn. Horst Bischof

Advisors:
Dipl.-Ing. Dr.techn. Peter Roth
Dipl.-Ing. Christian Leistner

Graz, March 2010

Deutsche Fassung:
Beschluss der Curricula-Kommission für Bachelor-, Master- und Diplomstudien vom 10.11.2008
Genehmigung des Senates am 1.12.2008

EIDESSTATTLICHE ERKLÄRUNG

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

Graz, am ……. ……..
(Unterschrift)

Englische Fassung:

STATUTORY DECLARATION

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

……. ……..
date (signature)

Abstract

Automated object detection is an important task in computer vision and visual surveillance in particular. It is a difficult task to train accurate detectors that have a high performance on a wide variety of scenes. For this purpose, recently, in surveillance multi-camera networks attracted interest for training scene specific detectors to improve the detection performance and decrease the false positive rate, since there are many problems that cannot be tackled with single camera approaches (e.g. occlusion handling).

This thesis introduces a novel centralized approach to simplify information fusion within a multi-camera network by learning an object detectors from multiple cameras. This approach allows to collect information form an arbitrary number of cameras. Having calibrated cameras, where the calibration has to be performed only once for each camera, the centralized approach projects each camera's detection information to a central (virtual) camera. A mean-shift algorithm extracts local maxima from the fused information. This location information is back-projected to the single camera views to extract additional examples for training. The approach is demonstrated for the task of person detection within an on-line boosting framework. A detailed analysis of the learning behavior is given and it is shown that the performance of state-of-the-art detectors can be achieved on single camera views although only a small number of labeled training examples are used.

Kurzfassung

Automatische Objektdetektion spielt eine wichtige Rolle im Bereich des maschinellen Sehens und in der optischen Überwachung. Jedoch erweist sich das Trainieren von generellen Objektdetektoren die in möglichst vielen Umgebungen einsetzbar sind als äußerst schwierig. Aus diesem Grund werden seit kurzem Konfigurationen mit mehreren Kameras zum Trainieren von szenenspezifischen Detektoren im Überwachungsbereich verwendet um die Detektionen zu verbessern und falsche Detektionen zu verhindern, da mit einzelnen Kameras einige Probleme nicht gelöst werden können (z.B. Behandlung von Verdeckungen).

Diese Arbeit präsentiert einen neuen zentralistischen Ansatz um die Fusionierung der Informationen mehrerer Kameras durch Lernen von Objektdetektoren zu vereinfachen. Der Ansatz erlaubt es Informationen von einer beliebigen Anzahl an Kameras zu sammeln. Die Verwendung von kalibrierten Kameras, die nur einmal kalibriert werden müssen, erlaubt eine Translation der Detektionen jeder einzelnen Kamera in eine zentrale Ebene. Auf diese zentrale Ebene wird ein Mean-Shift Algorithmus angewandt um alle darin enthaltenden lokale Maxima zu finden. Die Positionsinformationen der lokalen Maxima werden in die einzelnen Kameras zurück projiziert um an den resultierenden Stellen positive Trainingsbeispiele zu extrahieren. Dieser Ansatz wird anhand der Personendetektion in einem Online-Boosting-Framework demonstriert. Das Lernverhalten wird im Detail analysiert und es wird gezeigt, dass die Ergebnisse eines aktuellen Detektors auch mit einer sehr geringen Anzahl an annotierten Trainingsbeispiele auf einzelnen Kameras erzielt werden können.

Acknowledgements

There are many people who have contributed in several ways to the creation of this thesis. Some of them might not even know that they made an allowance. Hence, accept the acknowledgement of my gratitude and forgive me that I am unable to thank each of you individually.

I especially wish to thank my supervisor Horst Bischof. Special thanks go to Christian Leistner and Peter Roth who provided invaluable help and feedback during the course of my work. Additional mention goes to Peter for proof-reading and correcting this thesis. Further, special thanks go to Horst Bischof and Pascal Fua, who enabled an one-month internship in the computer vision laboratory at the École Polytechnique Fédérale de Lausanne, and to Jérôme Berclaz who influenced my work during this internship. Finally, I wish to thank my parents for giving continuing support and motivation.

Armin Berger

Graz, Austria, March 2010

Contents

List of Figures

List of Tables

Chapter 1

Introduction

In general, object detection is a challenging and important task in computer vision and visual surveillance. According to increasing visual surveillance also the amount of data that has to be analyzed increases. Thus, systems for automated detection and analysis are required. The most important approach to tackle such a detection task is to apply a sliding window technique. This means that an image is divided into a fine grid of overlapping subpatches on various scales. Then each patch is evaluated and checked if it corresponds to a previously learned model. Early approaches used background models to tackle the detection task of single moving objects in un-cluttered scenes. In principle, this model can be either generative or discriminative. Generative models directly describe the training data. As a result, these models can create new synthetic data. In contrast, discriminative models learn the relation between the data and its corresponding labels. Discriminative models can be efficiently evaluated for new data, whereas generative models probably require several iteration steps to derive a solution. As a result wrong predictions may occur if unknown examples have to be predicted. Nowadays, the objects are modeled using a learning algorithm. This means that the models are derived from numerous examples. Vaillant *et al.* [97] and Sung and Poggio [91, 92] were one of the first that used a data driven learning-based technique to tackle the object detection problem. For example, state-of-the-art approaches use Boosting [33, 84] or Support Vector Machines [99] to solve this problem.

It is a fact that recent research only focused on effectively learning models and training classifiers with high performance instead of efficiently labeling and acquiring accurate training data. Anyway, large amounts of positive and negative training examples are required to train high quality classifiers, but usually this examples have to be obtained by hand labeling, which is time consuming and thus very expensive. Semi-supervised learning [13] methods can be applied instead of using supervised learning methods. In general, semi-supervised learning uses labeled and unlabeled data to train a classifier. This means that a small num-

ber of labeled examples is processed to train an initial classifier which is improved with unlabeled data afterwards. For example, Blum and Mitchell [8] introduced co-training. Co-training is a semi-supervise learning approach, which uses redundant views on the training data to improve a pair of classifiers. In detail, co-training generates two initial classifiers using labeled examples and further it uses predicted information form one classifier to train the opposite classifier. Levin *et al.* [54] were the first that used co-training to learn an object detector for a computer vision application. They used gray-value image information to train one classifier and background/motion information to train a second classifier. These two classifiers were then improved within a co-training framework.

The reason of using multiple cameras for a detection or tracking task is diversified. For instance, it allows to address problems with occlusions, since single camera view approaches are unable to handle occlusions. Further, most single camera approaches for object detection require large amounts of training data to create as general detectors as possible. Nevertheless, most of these single camera approaches fail in practice due to lacks of representation. To overcome such problems Leistner *et al.* [52] recently proposed a co-training framework to train adaptive scene specific classifiers using information from multiple cameras. This approach was extended by Roth *et al.* [83] later on. They assumed to use different camera views as different views on the data. In detail, they used geometric relations between the cameras to share information between the camera views as proposed in [4], [47] or [48], who used rather simple object or motion detectors. Most multi-camera systems estimate homographies to transfer information between the cameras. For example, Mueller *et al.* [62] introduced a common ground plane and estimated the homographies using at least four points from within the ground plane. By contrast, Stauffer and Tieu [89] used trajectories, that were estimated from tracking data in single views, to estimate homographies between single camera views. Further, Fleuret *et al.* [31] used a central 2D occupancy map to combine information from different cameras.

The contribution of this work is to develop a centralized approach for object detection in a multi-camera setup, whereas the cameras have partially overlapping fields of views. The object detector is required to extract high-quality positive examples from the training data. This approach to automatic labeling of data has two positive side-effects. First, a high-performance object detector is trained and second, all resulting positive training data can be deployed to train other object detectors.

The remainder of this thesis is organized as follows: In Chapter 2 several feature types are discussed for a low-level image representation and methods that enable a fast feature evaluation. Then, Chapter 3 depicts the basics of visual learning including the on-line boosting approach that is required to perform learning from multiple camera views. Chapter 4 derives the geometry that is essential for combining information from several cameras

within a 3D scene. Additionally, it discusses the baseline approach on which this thesis is built on. Further, Chapter 5 proposes the centralized approach that merges information from several cameras using geometric constraints and it obtains positive examples using the merged information. Chapter 6 shows experiments using the centralized approach and the corresponding results. Finally, Chapter 7 draws the conclusion of this work and gives a short summary.

Chapter 2

Image Representation

In this chapter we discuss image features and image representation. Image features constitute the basis for complex image analysis, e.g., object detection, object recognition, or object tracking. These tasks are very challenging, since real-world objects have a huge variability in color, texture and appearance. For this reason it is essential to have descriptive image features which are easy to compute and sufficiently describe the appearance of objects by handling its variability. In specific cases it is necessary to turn an image into a special image representation to compute and evaluate certain features. Moreover, at the use of image sequences also motion information can be used to detect objects. A simple method to obtain motion information from an image sequence is to identify the image parts that belong to the background. This means that background models are derived to obtain foreground and motion information.

In short, Section 2.1 presents an overview of widely used features. Section 2.2 introduces the combination of different feature types. Section 2.3 depicts several image representations that allow a more efficient feature computation and evaluation. Finally, Section 2.4 depicts a simple background model that allows foreground and background identification in image sequences.

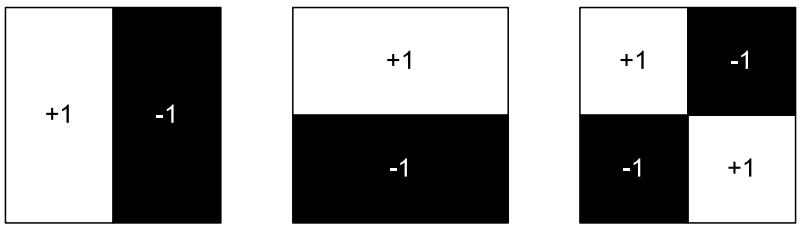
2.1 Image Features

Image features either describe global or local image information, whereas local features either represent information which is obtained through a local neighborhood operation or they describe the image structure directly. In case of neighborhood operations a feature response vector or a feature response value indicates the presence or the absence of special image structures at certain places. For instance, certain features indicate the existence of simple points, lines, edges, or even more complex structures. In general, these features

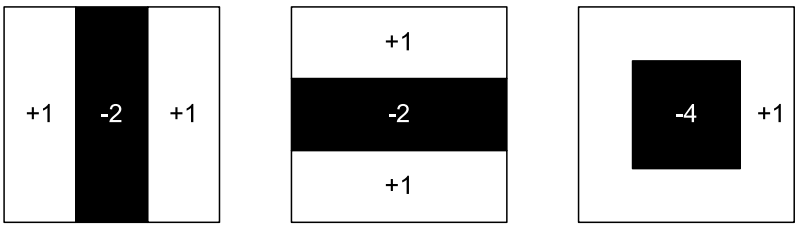
have to be descriptive enough to represent specific objects. On the other hand the features have to circumvent the object variability such that it is possible to derive an object model. Additionally, these features should be easy to compute, since huge numbers of features have to be evaluated.

2.1.1 Haar Wavelets

Oren *et al.* [69] and Papageorgiou *et al.* [76] proposed Haar wavelets in computer vision for recognition tasks. Haar wavelets compute differences of averaged intensities in certain regions. Originally there have been only three different types of Haar wavelets (see Figure 2.1a). Within this thesis an extended set of Haar-like wavelets is used similar to Viola and Jones [100] (see Figure 2.1. Haar-wavelets are closely related to Gabor Filters[1] [3], but they are much easier and faster to compute. Lienhart and Maydt [55] proposed an further extension to Haar-wavelets by using rotated versions of the original wavelets.



(a) Original Haar wavelets.



(b) Additional Haar-like wavelets.

Figure 2.1: (a) shows the original Haar wavelets used by Oren *et al.* [69] and (b) shows the wavelets that are additionally used within this thesis.

Each wavelet describes a different type of edges or bars. For example, *two-rectangle* wavelets consist of two rectangular regions of equal size that are horizontally or vertically adjacent. Their feature response value is computed by summing up all pixel values in each rectangular region and additionally computing the difference between these sums. Usually the feature response is normalized to handle features on different scales. As a result, these feature types indicate horizontal or vertical edges in the image. Another *two-rectangle*

[1] Gabor filters banks work similar to the visual image processing in the primary human visual cortex.

wavelet consists of a larger rectangle that surrounds a smaller rectangle. The feature response value is estimated as follows: the inner rectangle's sum is subtracted from the surrounding rectangle's sum. *Three-rectangle* wavelets have an additional rectangle of equal size compared to the others. In this case the sum of the center rectangle is subtracted from the sum of the outer rectangles. High response values in conjunction with this special type of features indicate horizontal or vertical lines. Finally there are *four-rectangle* wavelets that compute the difference between diagonal pairs of rectangles. They indicate the occurrence of diagonal lines. Viola and Jones [100] used integral images to speed-up the feature evaluation process. The computation of integral images is explained in Section 2.3.1.

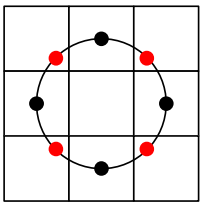
2.1.2 Local Binary Patterns

Local Binary Patterns (LBP) were proposed by Ojala *et al.* [66] for face recognition. In general, $LBP_{n,r}$ describe the texture of image patches by thresholding n neighboring points that are placed equidistant on a circle with radius r to a central pixel. The value of the central pixel defines the threshold. Usually, the value of any neighboring point is bilinearly interpolated if this point's location does not match exactly a pixel's location (see Figure 2.2a). The original LBP operator considered the $(8, 1)$ neighborhood only (see Figure 2.2b). This means that each pixel takes $n = 8$ equidistant points from a circle with radius $r = 1$. Several extensions [67] enabled the operator to use neighborhoods of different sizes, to use uniform patterns or to use patterns that are rotation invariant. Uniform patterns $LBP^{u}_{n,r}$ with $u = 2$ allow at most two bitwise transitions per pattern only. The uniform patterns are assumed to be circular: 00000000, 00110000, 11111000 or 11100011. These binary results are then converted into decimal numbers. Based on these decimal numbers histograms are computed to describe the texture within defined regions, since the number of uniform patterns is very limited.

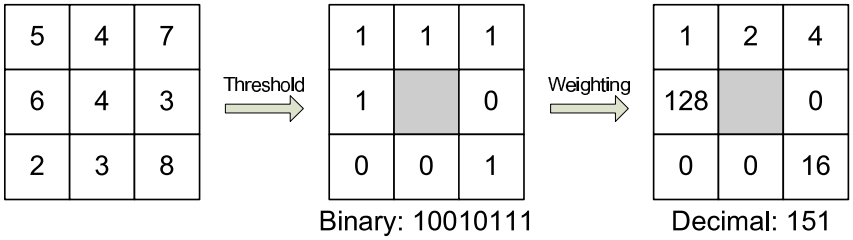
Within this thesis LBP are similarly computed as presented by Ahonen *et al.* [1]. They assigned each uniform pattern to a single histogram bin. All remaining nonuniform patterns are accumulated within one separate bin. The nonuniform patterns are merged within one bin, because there is a large number of nonuniform patterns that occur very sparsely. Each histogram bin H_i of a rectangular region R_j is computed as follows:

$$H_{i,j} = \sum_{x,y} \delta(R_{lbp,j}(x, y) = i) \quad . \tag{2.1}$$

Again using integral images significantly decrease the computation time for histograms of several rectangular regions per image. The usage of integral images with LBP is depicted in Section 2.3.2.



(a) LBP with circular (8,1) neighborhood where the values of the red dots have to be interpolated.



(b) Computing a LBP with $(8, 1)$ neighborhood: neighboring pixels are thresholded to get the binary pattern and then weights are assigned to get a decimal value.

Figure 2.2: Estimating LBP: (a) neighborhood where some pixel values have to be interpolated and (b) computation of the LBP pattern.

2.1.3 Histogram of Oriented Gradients

Histograms of oriented gradients (HOG) were originally proposed by Dalal and Triggs [20] in context of human detection. Their method evaluates normalized local histograms of image gradients in a dense grid. They divided the image into cells and computed a histogram of oriented gradients within each cell. This is achieved by accumulating pixels with equal gradient orientation per cell into one histogram bin. Either the cells are contrast normalized to improve the invariance to illumination changes or several cells are combined to blocks which are normalized (see Figure 2.3). For instance, Dalal and Triggs [20] combined four cells to blocks and they additionally shared cells between these blocks to gain robustness. In other words, each cell was assigned to at least four blocks. The derivations, which are essential to compute each pixel's gradient, are computed using $[-1, 0, +1]$ and $[-1, 0, +1]^T$ masks. Higher order filters or additional blurring would decrease the performance of the HOG features [20]. After deriving the image, the unsigned gradients are voted into 9 histogram bins that are equally spaced between 0° and 180°. The votes are based on the gradient's magnitude and are bilinearly interpolated between neighboring bin centers and neighboring locations to reduce aliasing effects.

Within this thesis each block of HOG cells is assumed to be a single feature similar to Zhu *et al.* [107]. Dalal and Triggs used cells with a fixed size and thus all blocks were of equal size. Zhu *et al.* allowed variable sized blocks with an aspect ratio of $(1 : 1)$, $(1 : 2)$

or $(2:2)$, such that each block returns a 36 dimensional histogram. The 36D histogram is created by concatenating 2×2 adjacent cells whereas each cell is described by a histogram of 9 bins. Each block response is seen as one feature response vector $\mathbf{v}$. Additionally, the response vectors are normalized using *L2-norm*

$$\mathbf{v} \rightarrow \frac{\mathbf{v}}{\sqrt{\|\mathbf{v}\|_2^2 + \epsilon^2}} \quad ,$$

which is followed by clipping. The clipping limits the maximum values of $\mathbf{v}$ to 0.2.

The HOG feature computation process is computationally much more expensive than computing simple Haar-like features. Anyway, the computation time of HOG features also can be decreased using integral images. The computation of integral images in context of HOG features is described in Section 2.3.3.

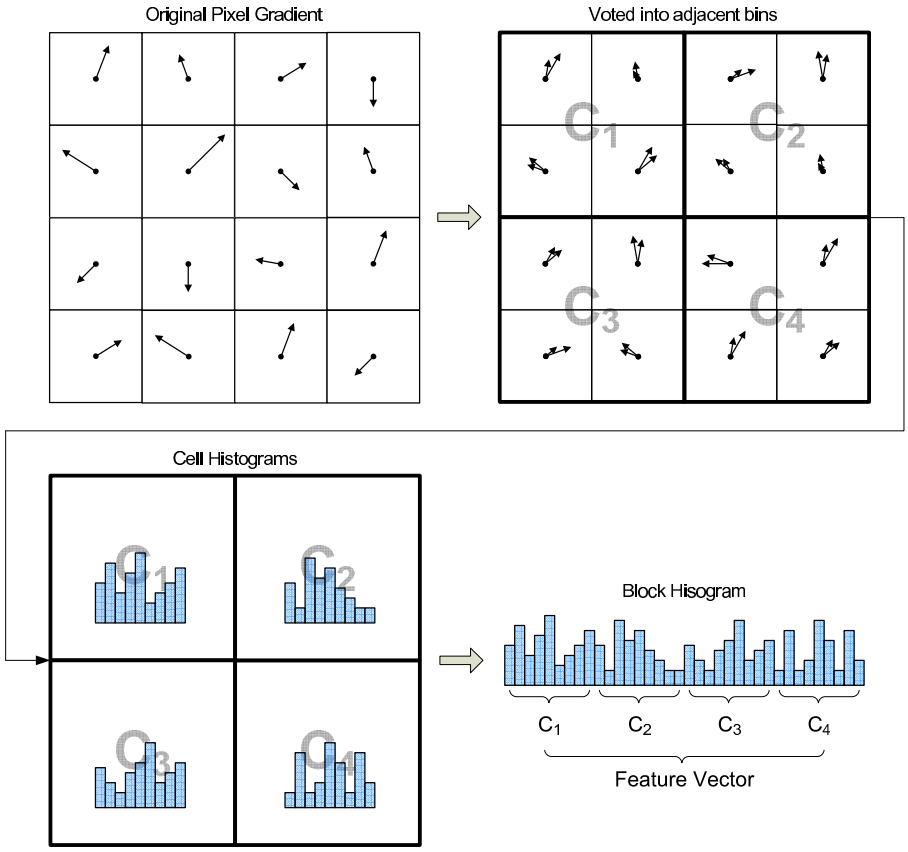


Figure 2.3: Computation of HOG feature response: compute the gradient of an image; vote the gradient into histogram bins at each pixel; combine several pixels to cells; compute histograms for each cell by accumulating the bins; compute the block histogram by concatenating several cell histograms.

2.2 Combining Multiple Feature Types

To increase a classifier's performance different features are applied in parallel. The huge advantage of using different feature types is that different local image information can be described. For example, HOG features describe the gradient distribution within image regions using histograms, while LBPs describe the local texture within rectangular image regions. There are several feature types that can be combined (e.g. [103, 63, 102, 71, 41, 82]). Wojek and Schiele [103] systematically evaluated different feature types for general people detection. They used linear Support Vector Machines (SVMs) and AdaBoost in conjunction with decision tree stumps for classification. Moreover, Wang *et al.* [102] concatenated the feature response vectors of HOG features and LBP features to combine these feature types. They used a linear SVM to learn these concatenated response vectors. Grabner *et al.* [41] or Roth *et al.* [82] combined Haar-like wavelets, orientation histograms, and LBPs to improve a classifiers performance. In this case orientation histograms correspond to Haar features that are computed on gradient images. Levin *et al.* [54] even combined background and gray-value information to improve a pair of classifiers. Negri *et al.* [63] combined Haar-features and HOG-features using AdaBoost for vehicle detection.

The approach used within this thesis is strongly related to their approach. Within this thesis different feature types are combined using a robust on-line boosting algorithm for feature selection (see Section 3.4). In detail, subsets of Haar-like wavelets, HOG features and LBPs are combined by creating a feature pool that contains these feature types with a certain probability. Then the features with the lowest training error are selected on the basis of this feature pool. Finally, on-line boosting is applied on this feature pool to select those features that represent a set of training examples best.

2.3 Speed-up Feature Computation

In general, the total time for feature evaluation is influenced by two factors. On the one hand the total number of features that have to be evaluated per example and on the other hand the total number of examples that have to be processed. Hence, the feature computation and evaluation process has to be efficiently implemented to be able to compute large classifiers which evaluate numerous features. Additionally, large amounts of training data have to be processed to obtain good classifiers. Therefore, the remainder of this section presents the integral image representation and its variations for different feature types, that finally speed-up the feature evaluation process.

2.3.1 Integral Image Representation

Crow [17] introduced this method to compute summed-area tables for texture mapping. Viola and Jones [100] used Crow's idea to compute rectangular features (Haar-wavelets) in constant time. In this case, only four image access operations are necessary to compute the sum of a rectangular region. An integral image at location x, y contains the sum of all pixels that are placed left and above this location including the pixel value at this position:

$$ii(x,y) = \sum_{x' \leq x, y' \leq y} i(x',y') \quad , \tag{2.2}$$

where $ii(x,y)$ expresses the integral image and $i(x,y)$ corresponds to the input image. By splitting up the computation step into two separate computations it is possible to create the integral image in one pass over of the input image as follows:

$$s(x,y) = s(x,y-1) + i(x,y) \tag{2.3}$$

$$ii(x,y) = ii(x-1,y) + s(x,y) \quad , \tag{2.4}$$

where $s(x,y)$ denotes the cumulative row sum. Additionally, special cases concerning the first row and the first column of the input image have to be considered:

$$s(x,-1) = 0 \quad , \tag{2.5}$$

$$ii(-1,y) = 0 \quad . \tag{2.6}$$

Based on the integral image representation the sum of pixel values within the rectangular region D is defined as

$$D = d - b - c + a \quad , \tag{2.7}$$

where location a contains the sum of all pixel values within rectangle A. Location b stores the pixel value sum of area A plus area B. Further, location c contains the sum of $A + C$ and location $d = A + B + C + D$ as illustrated in Figure 2.4.

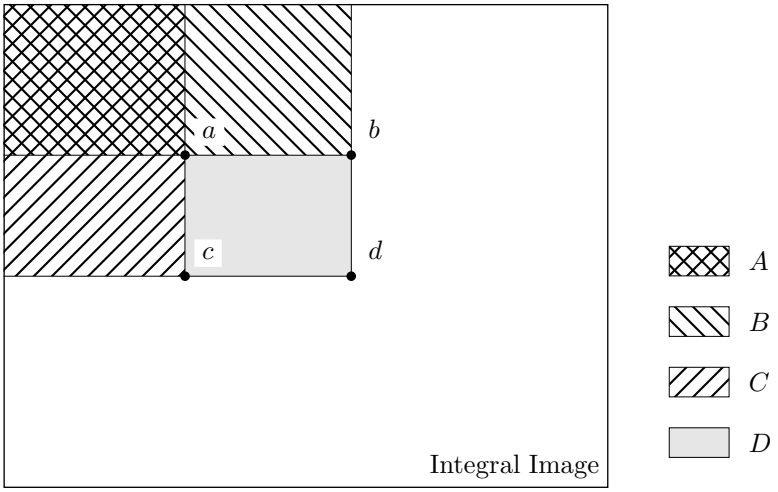


Figure 2.4: Rectangular sum computation with four access operations [100]: The sum of all pixels within rectangle A is stored at location a in the integral image. Location b equates to the sum of rectangle $A + B$, location c corresponds to the pixel value sum within $A+C$ and $d = A+B+C+D$. Therefore, the sum within rectangle $D = d - b - c + a$.

2.3.2 LBP Integral Image Representation

The idea of computing a LBP integral image is related to computing standard integral images (see Section 2.3.1). However, LBP-features describe the texture in a rectangular region. Therefore, they compute a histogram to represent the distribution of the different LBP. The standard integral image was not developed to handle multidimensional input data. Porikli [77] proposed an approach to compute integral histograms for fast histogram extraction in d dimensional data. He defined the integral histogram $H(\mathbf{x}^p, b)$ at a point $\mathbf{x}^p$, where $\mathbf{x}^p$ is a point along a sequence of points $\mathbf{x}^0, \ldots, \mathbf{x}^p$:

$$H(\mathbf{x}^p, b) = \bigcup_{j=0}^{p} Q(f(\mathbf{x}^j)) \quad , \tag{2.8}$$

where $Q(\cdot)$ corresponds to the bin at the current point, $\bigcup$ denotes the union operator, and f is any function that maps the input points to a k-dimensional tensor, for example $f([x_1, \ldots, x_d]) = [y_1, \ldots, y_k]$. Hence, $H(\mathbf{x}^p, b)$ is equal to the sum of the previously visited point's histogram bin values. In short, it defines the histogram between the origin and the current point. Finally, a histogram is extracted by

$$h(T, b) = \sum_{r=0}^{d} (-1)^r \sum_{l=1}^{C_r^d} H(\mathbf{x}_l^r, b) \quad , \tag{2.9}$$

where T is a target region. This target region is a polytope that is enclosed by a finite number of hyperplanes within a cartesian space. Its boundary points are $\mathbf{x}_l^r$ where the indices in each dimension consist of r coordinates x^- and $d-r$ coordinates x^+. If r is fixed there are C_r^d such combinations. If $r=0$ then there is one point; if $r=1$ there are already d points, and so forth. In case of $d=2$ and $k=1$ this formulation is equivalent to the integral image in Section 2.3.1.

2.3.3 HOG Integral Image Representation

Zhu *et al.* [107] explained a method to efficiently compute HOG features using integral images. However, their approach is inferior to Dalal's and Triggs' method. Dalal and Triggs additionally applied a Gaussian mask on the cells and used a spatial bilinear interpolation. Additionally, they bilinearly interpolated between the bins for the HOG block generation to avoid aliasing effects. Therefore, one separate integral image has to be computed per histogram bin as depicted in Section 2.3.2. Wang *et al.* [102] proposed to include the trilinear interpolation into the integral image approach: *Convoluted Trilinear Interpolation* (CTI). They also discretized the pixel gradient into 9 histogram bins, but then they treated the pixel value at each position as a 9 dimensional vector to overcome the problem that the gradient is a 2 dimensional vector (magnitude and angle) at each pixel. This means that they created one image per histogram bin to store the gradient magnitude of a specific gradient direction for each pixel (bin image). Hence, the value at each dimension equates to the value of the interpolated magnitude at the corresponding direction. The trilinear interpolation has to be done after the gradient discretization and before the integral images are constructed. This consists of two steps. First the magnitude of the gradients is voted into the histogram bins (bilinear interpolation) and then a 7×7 convolution kernel

$$Conv(k)_{7\times 7} = \frac{1}{256}\begin{bmatrix} 1 & 2 & 3 & 4 & 3 & 2 & 1 \\ 2 & 4 & 6 & 8 & 6 & 4 & 2 \\ 3 & 6 & 9 & 12 & 9 & 6 & 3 \\ 4 & 8 & 12 & 16 & 12 & 8 & 4 \\ 3 & 6 & 9 & 12 & 9 & 6 & 3 \\ 2 & 4 & 6 & 8 & 6 & 4 & 2 \\ 1 & 2 & 3 & 4 & 3 & 2 & 1 \end{bmatrix} \tag{2.10}$$

is applied on the orientation bin images afterwards. Figure 2.5 illustrates the HOG integral image computation process. Further, they used the Fast Fourier Transform (FFT) to accelerate the convolution with the interpolation kernel. Anyway, within this thesis FFT was not applied to speed-up the convolution process, instead the large kernel is split up into two 1D

kernels. The horizontal kernel is defined as

$$Conv_{sep}(k)_{1\times 7} = \frac{1}{16}\begin{bmatrix} 1 & 2 & 3 & 4 & 3 & 2 & 1 \end{bmatrix} \tag{2.11}$$

and the corresponding vertical kernel is defined as $Conv_{sep}(k)^T$. In fact, applying both kernels on the bin images is equivalent to deploying the large kernel. Using the small convolution kernels has the advantage that one convolution with each kernel is much faster than using the large kernel. Hence, the computation time of both kernels together is even faster than a convolution with the large kernel.

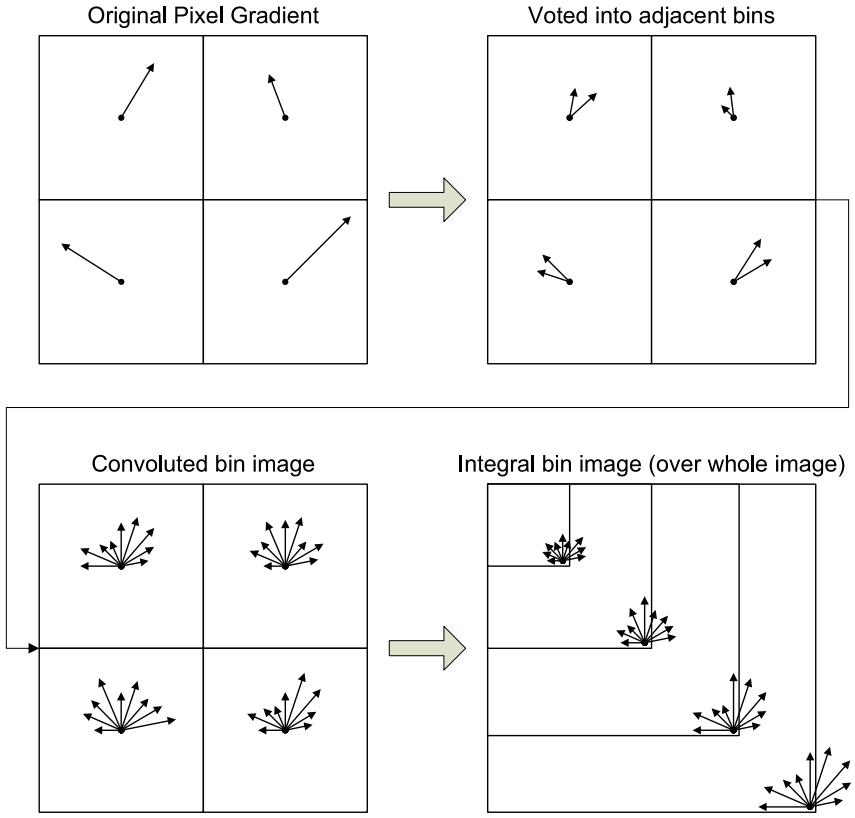


Figure 2.5: Trilinear interpolation method in context of integral images for histograms [102].

2.4 Background Models and Motion Information

In video sequences it is often helpful to use motion information for foreground/background identification. Modeling the background accurately constitutes the basis for efficient and usable background subtraction to obtain the motion information. Cheung and Chandrika [14] compared various background modeling approaches. First, background modeling methods

can be divided into two main categories: (i) non-recursive techniques and (ii) recursive techniques. The following summary of non-recursive and recursive background modeling methods discusses only highly adaptive background modeling methods and does not mention methods that require enormous resources for initialization as in [68, 42]. These approaches use either eigenimages or temporal minimum, maximum and maximum interframe differences at all identified background pixels.

Non-recursive techniques require a buffer to store a certain number of video frames. Hence, the simplest background model is to compute the difference between two successive video frames (frame differencing). However, this method is unable to identify pixels of large uniformly-colored moving objects, since only one single previous frame is considered to compute the difference image. More sophisticated methods use several images which are stored in a buffer to estimate an accurate background model (e.g. [94, 28]). Another simple and commonly used method is median filtering [19, 18]. In this case the background model is estimated by computing the median at each pixel location for all frames in a buffer.

Recursive methods estimate the background model on-the-fly which means that they do not use a buffer to store images. Therefore, they use either a Kalman filter as in [104, 50, 45], a Mixture of Gaussians method [38, 88], or an approximated median filter [60].

Within this thesis an approximated median filter is used which is similar to McFarlane's and Schofield's approach [60]. In general, the median cuts a distributions into two halves. Assume a finite list of n numbers. If n is an odd number, than the median is computed by first sorting the list $(x_i, \ldots, x_n)$ and selecting $\tilde{x}$, where

$$\tilde{x} = x_{\frac{n+1}{2}} \quad . \tag{2.12}$$

On the other hand, if n is an even number, then there is no single middle value. In this case the $\tilde{x}$ is often computed by taking the mean of the two middle values:

$$\tilde{x} = \frac{1}{2}\left(x_{\frac{n}{2}} + x_{\frac{n}{2}+1}\right) \quad . \tag{2.13}$$

Hence, the median is more robust to outliers than the arithmetic mean. For this reason, median filtering is also used in computer vision to remove salt and pepper noise in images. This noise consists of random occurring white and black pixels.

McFarlane and Schofield proposed a recursive filter to approximate the median. This approach has the advantage that no buffer is required and therefore no optimal buffer size has to be determined to compute an approximation of the median. Within their approach the approximated median is incremented by one if the input pixel value is larger than the current approximation. On the other hand, if the input pixel value is smaller than the currently approximated median, the current median's approximation is decreased by one.

Further, there are also various methods to detect the foreground pixels in images, as described by Cheung and Chandrika [14], using the background model. Commonly a pixel is assigned to the foreground, if the difference between an input image I_t at time t and the background model Bg_t at time t is smaller than a certain threshold $\mathcal{T}$:

$$|I_t(x,y) - Bg_t(x,y)| > \mathcal{T} \quad . \tag{2.14}$$

A different approach that also detects the foreground is to threshold on the basis of the normalized statistics:

$$\frac{|I_t(x,y) - Bg_t(x,y) - \mu_d|}{\sigma_d} > \mathcal{T}_s \quad , \tag{2.15}$$

where μ_d and σ_d equate to the mean and the standard deviation of $I_t(x,y) - Bg_t(x,y)$ for all spatial locations (x,y). Usually, the thresholds $\mathcal{T}$ or $\mathcal{T}_s$ are determined experimentally. Another possibility is to use a relative distance to additionally emphasizes the contrast in dark areas:

$$\frac{|I_t(x,y) - Bg_t(x,y)|}{Bg_t(x,y)} > \mathcal{T}_c \quad . \tag{2.16}$$

2.5 Summary

In this chapter local feature types are discussed. On the one hand, the basics on computing Haar-like wavelets, Local Binary Patterns and Histograms of Oriented Gradients are depicted. And on the other hand, integral images are presented to speed up the evaluation of each feature type. Additionally, a short introduction on background modeling is given, since background models also allow to detect foreground objects in video sequences. Moreover, the idea of combining multiple feature types to improve a classifier's performance is discussed shortly. In general, these image representations/features constitute the core elements of the object detector that is presented within this work, since the object detector uses feature selection (see Section 3.3) to create accurate object models.

Chapter 3

Boosting for Visual Learning

In general, learning refers to the process of improving performance which is obtained through well-directed efforts. Machine learning is closely related to statistics which means that certain behaviors or models are derived from existing data. In detail, either generative or discriminative data models are learned. Ulusoy and Bishop [96] compare both methods for object recognition. Discriminative object models describe the relation between data and the corresponding labels. Using this relation such models are able to predict labels for unknown examples, but they also interpolate between known examples, which may cause wrong predictions for novel examples. Anyway, these models can be efficiently evaluated. Alternatively generative object models can be learned, which directly model the training data. Hence, such models are able to create new synthetic data and handle partially labeled data. These models are learned incrementally which makes it easy to extend them.

In computer vision many techniques rely on machine learning or are even designed for it. Bishop [6] reflects the recent development in the area of pattern recognition and machine learning and gives introductions to various methods. Hence, there are various machine learning methods, which can be divided into several categories, for example supervised learning, unsupervised learning, or semi-supervised learning. Supervised learning methods such as Bayes classifiers, Boosting [33, 84], Nearest Neighbor Algorithm [16], Random Ferns [75], Random Forests [11] and Support Vector Machines [99] solely use training data with known labels, whereas often large amounts of training data are required to train an accurate classifier. For this purpose the training data have to be labeled previously, which is time consuming and thus very expensive.

By contrast, unsupervised learning performs either some sort of clustering or blind source separation, since this kind of algorithms use unlabeled examples only. In most cases even the number of cluster centers within a dataset is unknown. Under this directive it is very difficult to find the correct cluster centers. Duda *et al.* [26] give an overview of unsu-

pervised learning methods and an introduction to clustering. In general, there are several methods to perform efficient clustering. For instance, hierarchical clustering, which creates cluster centers in a tree structure, whereas the top level in the tree correspond to coarse clusters and the lowest level correspond to a very fine clusters. Lloyd [56] proposed the standard k-means algorithm for clustering which is a Partitional Clustering method. Another approach to clustering is conceptual clustering which uses the data structure by generating a concept description of each cluster [61, 90]. Comon [15] introduced the concept of independent component analysis for blind source separation. More methods are introduced in [23, 9, 39].

Semi-supervised learning [13] is a tradeoff between supervised and unsupervised learning. This means that these learning algorithms use labeled and unlabeled data. In general, these methods use a small amount of labeled data to train a classifier and use large amounts of unlabeled data afterwards to improve this classifier. Zhu [108] proposed an overview of familiar semi-supervised learning techniques including multi-view learning [22, 86, 10] and co-training [8, 65, 106].

Learning algorithms additionally can be separated into two main categories. On the one hand there are off-line algorithms, that have access to all training data during the training process. They create a classifier that is available after training. And on the other hand there are on-line algorithms that operate on single examples only. This means that on-line algorithms discard the example after processing. Hence, these algorithms do not need large amounts of memory to save all training examples and the classifier is available all the time. In general, off-line algorithms try to find the best possible solution to learn all training data. By contrast on-line algorithms have to adapt their solution every time a new example arrives.

In short, this chapter discusses the basics of boosting and object modeling in Section 3.1. Section 3.2 discusses the advantages of on-line boosting in contrast to off-line boosting. Section 3.3 depicts the usage of boosting for computer vision tasks and how boosting is deployed for feature selection. Further, Section 3.4 presents a robust method for off-line and on-line boosting to better handle wrongly labeled data. Finally, the concept of cascades and the WaldBoost algorithm is explained in Section 3.5 to speed-up the evaluation of large classifiers.

3.1 Boosting

Boosting is a supervised learning method that has been successfully used for many machine learning tasks such as text filtering, routing, and medical diagnosis. A more comprehensive list of machine learning tasks using boosting can be found in [85]. In general, boosting im-

proves the accuracy of any given learning algorithm by linearly combining weak classifiers to form a strong classifier. There is also a strong relation between boosting and support vector machines as discussed in [79]. For example, both methods try to maximize the margin between positive and negative training examples to find a proper decision boundary. Further, they generate discriminative models to describe objects. Freund [33] and Schapire [84] published the original boosting algorithms. Today there is a large number of different boosting algorithms. The primary divergency between most of these algorithms is the method of computing the training example weights and weighting the weak learners. Table 3.1 depicts the AdaBoost algorithm which was proposed by Freund and Schapire [35].

- Requires N labeled training examples $(\mathbf{x}_n, y_n)$, a base learning algorithm L_b and the number of base models M
- Initialize $D_1(n) = \frac{1}{N}$ for all $n \in \{1, \ldots, N\}$
- For $m = 1, \ldots, M$:
  1. Call L_b with the distribution D_m
  2. Get back a hypothesis $h_m : X \rightarrow Y$
  3. Calculate the error of $h_m : \varepsilon_m = \sum_{n:h_m(x_n) \neq y_n} D_m(n)$
  4. If $\varepsilon_m > \frac{1}{2}$ then set $M = m - 1$ and abort loop
  5. Set $\beta_m = \frac{\varepsilon_m}{1-\varepsilon_m}$
  6. Update distribution D_m:
     $D_{m+1}(n) = \frac{D_m(n)}{Z_m} \times \begin{cases} \beta_m & if h_m(x_n) = y_n \\ 1 & otherwise \end{cases}$
     where Z_m is a normalization constant chosen so that D_{m+1} is a probability distribution
- Output the final hypothesis: $H(x) = \arg\max_{y \in Y} \sum_{m:h_m(x)=y} \log \frac{1}{\beta_m}$.

Table 3.1: AdaBoost algorithm proposed by Freund and Schapire [35].

3.1.1 Weak Classifiers

In principle, a weak classifier h_t can be any kind of decision rule that has to perform slightly better than making a random decision, e.g., a Bayes classifier, a decision tree, or something superior. Hence, in case of a binary decision task the error rate has to be below 50%. For example, binary decision stumps estimate the mean and the variance of a positive and a negative class respectively. A simple threshold between the mean values is sufficient to derive a hypothesis for unknown examples. In particular, within this thesis histograms and nearest neighbor classifiers are used as weak learners.

Histograms are able to handle multi-modalities and can still be efficiently computed even in the on-line case. In the binary case there have to be two probability distributions. A weak classifier's positive feature responses are incrementally added to the corresponding positive distribution's bins $W^i_{t,+}$ during training and the negative feature responses are added to the corresponding negative distribution's bins $W^i_{t,-}$. As mentioned before, the creation is straight forward. Further, the evaluation is also easy to perform. The evaluation procedure consists of comparing the positive and the negative distributions. A confidence measure is computed using these distribution values as in [78]:

$$h_t(i) = \frac{1}{2} \log \left(\frac{W^i_{t,+}}{W^i_{t,-}} \right) \quad . \tag{3.1}$$

A distribution is more accurately modeled if more histogram bins are used, but using too many bins may cause overfitting whereas using too little bins may causes underfitting and a loss of accuracy.

Nearest neighbor classifiers model the cluster center $\mathbf{c}_+$ of the positive training data and the cluster center $\mathbf{c}_-$ of the negative training data respectively. This classifier predicts an unknown example $\mathbf{x}$ to be positive if

$$d(\mathbf{x}, \mathbf{c}_+) < d(\mathbf{x}, \mathbf{c}_-) \tag{3.2}$$

and vice versa. Therefore, a distance measure $d(\mathbf{p}, \mathbf{q})$ is essential. The Euclidean distance is a proper choice in the Euclidean space:

$$d(\mathbf{p}, \mathbf{q}) = \sqrt{(p_1 - q_1)^2 + (p_2 - q_2)^2 + \ldots + (p_n - q_n)^2} \quad . \tag{3.3}$$

Further, the nearest neighbor classifier is able to handle nD feature responses. The on-line nearest neighbor classifier has to estimate an on-line mean for feature responses computed on positive data $\mathbf{c}_+$ and an on-line mean for feature responses determined on negative examples $\mathbf{c}_-$, that were already processed.

3.1.2 Strong Classifiers

A strong classifier H is constructed by computing a linear combination of T weak classifiers h_t, where T defines the number of iterations:

$$H(\mathbf{x}) = \sum_{t=1}^{T} \alpha_t h_t(\mathbf{x}) \quad . \tag{3.4}$$

The weights α_t are inversely proportional to the training error ε_t of each weak classifier h_t.

3.2 Off-line Boosting versus On-line Boosting

An on-line learning algorithm processes an example only once as it arrives. This means that an example is discarded after it is processed and that the classifiers are learned incrementally. This approach is needed if data continuously arrives or huge data sets have to be processed, since most off-line learning algorithms have to process examples multiple times. Another advantage of on-line learning methods is that the classifier is available for classification of unknown examples all the time.

Oza and Russel [74] developed an on-line boosting algorithm (see Table 3.2) that corresponds to the off-line AdaBoost.M1 algorithm [35]. The on-line algorithm requires a set of M weak learner h_m, an incremental on-line learning strategy $OnlineBase()$, and a training example d. The on-line learning strategy takes a current hypothesis and the current example and returns an updated hypothesis. If an example is misclassified the example weight λ_d is increased for the next weak learner using a loss-function, otherwise it is decreased. In the off-line case the sum of all weights equates to one, but in the on-line case the sum of all λs equates to N, which is the number of examples that have been already processed. Hence, λ_m^{sc} and λ_m^{sw} are scaled, which correspond to the number of correctly and wrongly classified examples. Introducing the scale factors f_m^c and f_m^w leads to

$$\lambda_m^{sc} f_m^c = \frac{N}{2} \Longrightarrow f_m^c = \frac{N}{2\lambda_m^{sc}} \tag{3.5}$$

$$\lambda_m^{sw} f_m^w = \frac{N}{2} \Longrightarrow f_m^w = \frac{N}{2\lambda_m^{sw}} \quad , \tag{3.6}$$

where λ_m^{sc} and λ_m^{sw} can be treated as weights. Further, it is expected that $\lambda_m^{sc} > \frac{N}{2}$ and $\lambda_m^{sw} < \frac{N}{2}$, $f_m^c < 1$, and $f_m^w > 1$. Therefore, the weights of correctly classified examples decrease and the weights of incorrectly classified examples increase. Oza [72] also showed that on-line computed weak Naïve Bayes classifiers converge to the weak Naïve Bayes classifiers that are obtained by an off-line boosting algorithm if the number of iterations converges to infinite and if the same training examples are processed in both cases.

- Requires M on-line weak learners h_m and a training example d.
- Set example's weight $\lambda_d = 1$.
- For $m = 1, \ldots, M$:
  1. Set k according to $Poisson(\lambda_d)$.
  2. Do k times: $h_m = OnlineBase(h_m, d)$
  3. If $h_m(d)$ is correctly labeled
     (a) $\lambda_m^{sc} \longleftarrow \lambda_m^{sc} + \lambda_d$
     (b) $\lambda_d \longleftarrow \lambda_d \left(\frac{N}{2\lambda_m^{sc}}\right)$
  4. Else
     (a) $\lambda_m^{sw} \longleftarrow \lambda_m^{sw} + \lambda_d$
     (b) $\lambda_d \longleftarrow \lambda_d \left(\frac{N}{2\lambda_m^{sw}}\right)$

To classify new examples:

- For $m = 1, \ldots, M$:
  1. Calculate $\varepsilon_m = \frac{\lambda_m^{sw}}{\lambda_m^{sc} + \lambda_m^{sw}}$
  2. Calculate $\beta_m = \frac{\varepsilon_m}{1 - \varepsilon_m}$
- Return:

$$h(x) = \arg\max_{c \in C} \sum_{m: h_m(x) = y} \log \frac{1}{\beta_m} \quad . \tag{3.7}$$

Table 3.2: On-line boosting algorithm proposed by Oza and Russel [74].

3.3 Boosting for Feature Selection

For computer vision tasks like object detection or object tracking, the objects are modeled using image features to obtain robustness. In specific cases the number of possible image features can be very large. Assuming a rectangular detection window with the size 30×60, creating all possible Haar-features within this window would lead to an overcomplete set of several million features. It is impossible to perform a complete evaluation using all features because of limited resources. Thus, an optimal subset of features $\mathcal{F}_{sub}$ is extracted from the entire set of all possible features $\mathcal{F}$. For this reason, $\mathcal{F}_{sub}$ should be able to represent the designated objects as best as possible. In detail, the designated objects are modeled using several weak learners, whereas one feature is assigned to one weak learner, on the basis of feature response values. Finally, numerous weak learners are boosted to form a strong classifier. As a result, this strong classifier only depends on a small number of features.

3.3.1 Off-line AdaBoost

Tieu and Viola [93] were the first that enabled AdaBoost to solve a feature selection task. Their approach requires a set $\mathcal{X} = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_L, y_L)\}$ of labeled training images, where $y_i = 0, 1$ for negative and positive examples respectively. This training data set is divided into m negative and l positive examples. Based on this information the initial example weights are set to $w_{1,i} = \frac{1}{2m}, \frac{1}{2l}$ for $y_i = 0, 1$ respectively. The boosting algorithm trains a weak classifier h_j for each feature j using the example weights w_t at time t. Based on the resulting training errors ε_j the weak learner with the lowest error is selected and added to $\mathcal{F}_{sub}$:

$$\varepsilon_j = \Pr_i^{w_t} \left[h_j(\mathbf{x}_i) \neq y_i \right] \quad , \tag{3.8}$$

where ε_j specifies the proportion of wrongly predicted examples by the weak learner h_j. On the basis of the lowest training error $\varepsilon_t = \varepsilon_j$ the new example weights are computed as follows:

$$w_{t+1,i} = w_{t,i} \beta_t^{1-e_i} \quad , \tag{3.9}$$

where $e_i = 0, 1$ for the example $\mathbf{x}_i$ whether it is correctly or incorrectly predicted and

$$\beta_t = \frac{\varepsilon_t}{1 - \varepsilon_t} \quad . \tag{3.10}$$

Additionally all weights $w_{t+1,i}$ are normalized to form a distribution:

$$w_{t+1,i} = \frac{w_{t+1,i}}{\sum_{j=1}^{n} w_{t+1,j}} \quad . \tag{3.11}$$

Finally, a strong classifier is computed on linearly combining the selected weak learners:

$$H(\mathbf{x}) = \sum_{t=1}^{T} \alpha_t h_t(\mathbf{x}) \geq \frac{1}{2} \sum_{t=1}^{T} \alpha_t \quad , \tag{3.12}$$

where

$$\alpha_t = \log \frac{1}{\beta_t} \quad . \tag{3.13}$$

An overview of this off-line feature selection algorithm is depicted in Table 3.3.

- Given example images $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)$ with $y_i = 0, 1$.
- Initialize weights $w_{1,i} = \frac{1}{2m}, \frac{1}{2l}$ for $y_i = 0, 1$
- For $t = 1, \ldots, T$:
  1. Train one hypothesis h_j for each feature j using w_t, with error $\varepsilon_j = \Pr_i^{w_t}[h_j(\mathbf{x}_i) \neq y_i]$.
  2. Choose $h_t(\cdot) = h_k(\cdot)$ such that $\forall j \neq k, \varepsilon_k < \varepsilon_j$ and set $\varepsilon_t = \varepsilon_k$
  3. Update:
  $$w_{t+1,i} = w_{t,i}\beta_t^{1-e_i} \quad ,$$
  where $e_i = 0, 1$ for the example $\mathbf{x}_i$ and $\beta_t = \frac{\varepsilon_t}{1-\varepsilon_t}$.
  Normalize $w_{t+1,i} \leftarrow \frac{w_{t+1,i}}{\sum_{j=1}^{n} w_{t+1,j}}$
- The final hypothesis is:
$$H(\mathbf{x}) = \sum_{t=1}^{T} \alpha_t h_t(\mathbf{x}) \geq \frac{1}{2} \sum_{t=1}^{T} \alpha_t \quad ,$$
where $\alpha_t = \log \frac{1}{\beta_t}$

Table 3.3: Discrete off-line AdaBoost algorithm for feature selection [93].

3.3.2 Discrete On-line Boosting

Based on the Oza's ideas (see Section 3.2), Grabner and Bischof [40] presented an on-line boosting framework for feature selection. For this purpose they introduced a new concept called *selector*, since Oza's approach is not directly applicable to extract a subset of features $\mathcal{F}_{sub}$ from $\mathcal{F}$.

A **selector** holds a fixed set of M weak learners. Each weak learner is assigned to a feature. And the weak learner with the lowest training error corresponds to the selector's decision. Therefore, a selector acts like a weak classifier. If a selector h^{sel} is trained, all its weak classifiers are updated with respect to the importance weight λ of the current example:

$$h^{sel}(\mathbf{x}) = h_m(\mathbf{x}) \quad , \tag{3.14}$$

where

$$m = \arg\min_i e_i \quad . \tag{3.15}$$

The training error e_i for each weak classifier h_i is computed using the number of correctly

$\lambda_i^{correct}$ and wrongly λ_i^{wrong} classified examples that have already been processed:

$$e_i = \frac{\lambda_i^{wrong}}{\lambda_i^{correct} + \lambda_i^{wrong}} \quad . \tag{3.16}$$

The on-line boosting algorithm for feature selection is illustrated in Figure 3.1. It is initialized with N selectors $h_1^{sel}, \ldots, h_N^{sel}$ that contain a fixed number of weak learners which are assigned to randomly generated features. Now, each selector has its own feature pool $\mathcal{F}_{sub,n}$. If a new training example $\langle \mathbf{x}, y \rangle$ arrives, all weak learners within the first selector are updated with the example weight $\lambda = 1$ and the weak learner with the lowest training error ε_t is selected. Note, that the weak classifiers can be updated by any on-line learning algorithm. Further, λ and the update weight α_n are computed using the selector's training error. λ is passed to the next selector until all selectors are computed. The strong classifier H that results form on-line boosting is available any time and it is computed by linearly combining all selectors using α_n:

$$H(\mathbf{x}) = \text{sign}\left(\sum_{n=1}^{N} \alpha_n \cdot h_n^{sel}(\mathbf{x})\right) \quad . \tag{3.17}$$

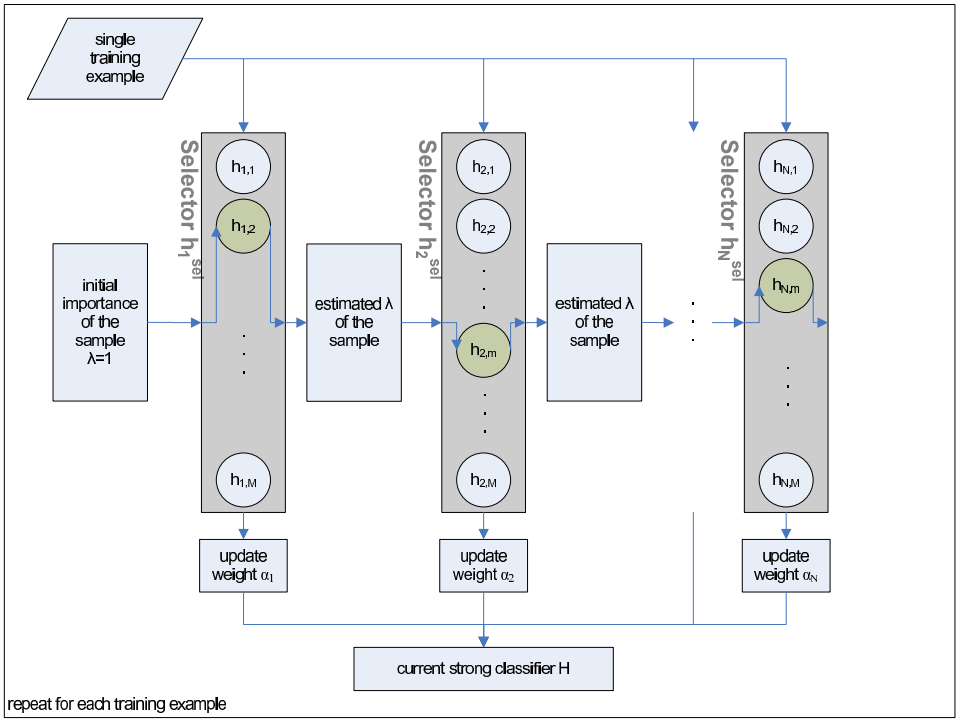


Figure 3.1: On-line boosting for feature selection proposed by Grabner and Bischof [40].

3.4 Robust Boosting

Maclin and Optiz [57] were one of the first that noticed AdaBoost's sensitivity to label noise (i.e. an example was wrongly labeled during the labeling process). Generally, AdaBoost tries to learn the training examples by re-weightening the examples. This means that high weights are assigned to hard examples and low weights are assigned to easy examples, respectively. This enables the algorithm to concentrate on the hard examples. If a weak learner predicts the actual label of the example and the actual label does not coincide with the assigned label, AdaBoost increases the weight of this specific examples to concentrate on it. Hence, the weak learners are forced to learn this noisy examples which may corrupt the learned classifier. Mason *et al.* [59] were the first who analyzed boosting algorithms using functional gradient descent that seem to be more robust in case of label noise if certain *loss functions* are used. Within this context they also proposed the *AnyBoost* framework that allows to easily change the loss functions.

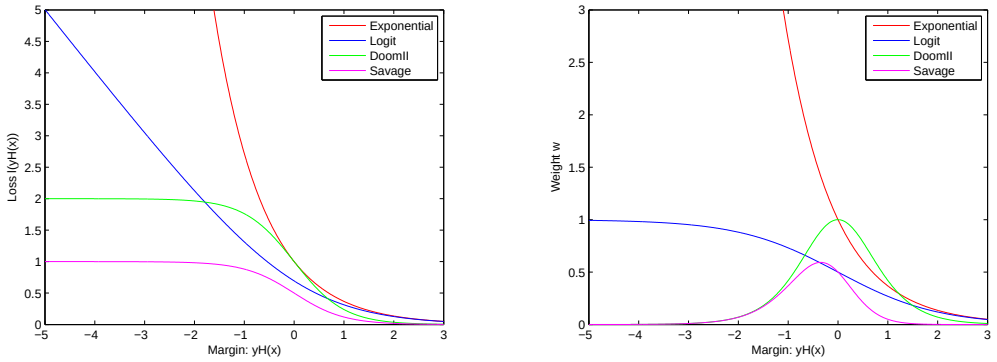
In general, a **loss function** describes the cost or the effort of specific events. In other words, it maps an event onto a real value. In case of boosting they map the classification margin to real values.

Mason *et al.* also proposed the DoomII loss function [59] that shows increased robustness to noisy data label. Further, Friedman *et al.* [36] showed the connection between boosting algorithms and stage-wise additive logistic regression methods. They proposed another loss function and the corresponding boosting algorithm (LogitBoost). Other loss functions and boosting methods are published in [25] (MadaBoost), [34] (BrownBoost) and [58] (SavageBoost). Table 3.4 and Figure 3.2a illustrate possible loss functions. Different loss functions have different effects on the training strategies if label noise is present.

	loss functions
Exponential	$\ell_{exp}(yH(x)) = \exp(-yH(x))$
DoomII	$\ell_{doom}(yH(x)) = 1 - \tanh(yH(x))$
Logit	$\ell_{log}(yH(x)) = \log(1 + \exp(-yH(x)))$
Savage	$\ell_{sav}(yH(x)) = 1/(1 + \exp(2yH(x)))^2$

Table 3.4: Most commonly used loss functions for boosting.

As already mentioned, AdaBoost strongly weights mislabeled examples. The corresponding loss function increases exponentially which originates the aggressive weighting strategy. For instance, the Logit loss function increases linearly in case of misclassification which is less aggressive and the Savage and the DoomII loss functions are even constant from a certain point on. As a result, these loss functions stop penalizing the classifier if it continuously predicts the wrong label in case of massive misclassification. Additionally, Figure 3.2b reflects the corresponding weight functions.



(a) Loss functions **(b)** Weight functions

Figure 3.2: Different loss functions used for boosting and their corresponding weight functions.

Leistner *et al.* [53] introduced an on-line boosting algorithm for feature selection that is an on-line formulation of the GradientBoost proposed by Friedman [37]. Friedman's approach performs stage-wise gradient descent at a given loss function. Leistner *et al.* derived that maximizing the weighted classification accuracy

$$f_t(\mathbf{x}) = \arg\max_{f(\mathbf{x})} \sum_{n=1}^{N} w_n y_n f(\mathbf{x}_n) \quad , \tag{3.18}$$

where $w_n = -\ell'(y_n F_{t-1}(\mathbf{x}_n))$, is equivalent to solving the optimization problem which uses a loss function ℓ to find a set of weak learners $\{f_1(\mathbf{x}), \ldots, f_M(\mathbf{x})\}$ and the corresponding boosting model

$$F(\mathbf{x}) = \sum_{m=1}^{M} f_m(\mathbf{x}) \quad , \tag{3.19}$$

that minimizes the loss. This boosting model is slightly differs form a discrete boosting model (see Equation 3.17), since this approach uses a probabilistic output of the weak learners like the RealBoost algorithm [36] does:

$$f(x) = \frac{1}{2} \log \frac{p_+(x)}{1 - p_+(x)} \quad , \tag{3.20}$$

where $p_+(x)$ constitutes the probability of the sample to be positive. This robust on-line boosting algorithm for feature selection is depicted in Table 3.5. In detail, the algorithm uses a fixed set of weak learners and performs boosting on the selectors, similar to the on-line boosting algorithm in Section 3.3. Again each selector selects its weak learner with the

lowest total weight error:

$$f_m(\mathbf{x}_n) = f_m^j(\mathbf{x}_n) \quad , \tag{3.21}$$

where

$$j = \arg\min_k e_m^k \quad . \tag{3.22}$$

The total weight error is computed for each weak learner as follows:

$$e_m^k = e_m^k + w_n \cdot \delta(\text{sign}(f_m^k(\mathbf{x}_n)) \neq y_n) \quad , \tag{3.23}$$

where

$$\delta(x) = \begin{cases} 1, \text{if } x = true \\ 0, \text{if } x = false \end{cases} \quad . \tag{3.24}$$

The optimization step, that is depicted in Equation 3.18, is iteratively performed. This means that the samples are propagated through the selectors and that the weights λ_m are estimated according to the negative derivative of the loss function (see Figure 3.2).

- Requires a training example $(\mathbf{x}_n, y_n)$, a differentiable loss function $\ell(\cdot)$, M selectors and K weak learners per selector
- Set $F_0(\mathbf{x}_n) = 0$ and set initial weight $w_n = -\ell'(0)$
- For $m = 1, \ldots, M$:
  1. For $k = 1, \ldots, K$:
     (a) Train k^{th} weak learner $f_m^k(\mathbf{x})$ with $\langle \mathbf{x}_n, y_n \rangle$ and weight w_n
     (b) Compute error: $e_m^k = e_m^k + w_n \cdot \delta(\text{sign}(f_m^k(\mathbf{x}_n)) \neq y_n)$.
  2. Find best weak learner with the least total weighted error: $j = \arg\min_k e_m^k$
  3. Set $f_m(\mathbf{x}_n) = f_m^j(\mathbf{x}_n)$
  4. Set $F_m(\mathbf{x}_n) = F_{m-1}(\mathbf{x}_n) + f_m(\mathbf{x}_n)$
  5. Set the weight $w_n = -\ell'(y_n F_m(\mathbf{x}_n))$
- Output the final model: $F(\mathbf{x})$

Table 3.5: Robust on-line boosting algorithm for feature selection introduced by Leistner *et al.* [53].

3.5 Speed-up Classifier Evaluation

In general, is the goal to maximize the performance and minimize the evaluation time of classifiers. In other words, the classifiers should have a high recall and a high precision and should be evaluable in real time. The classifier performance is mostly improved by using larger classifiers if the tasks to solve become harder.

An excellent method to decrease the evaluation time for large classifiers is the usage of cascades or WaldBoost.

3.5.1 The Concept of Cascades

Cascades of classifiers are completely degenerated decision trees (see Figure 3.3) that perform early rejection of wrongly classified image patches. In general, the idea is to use simple and small classifiers at low cascade stages to reject the majority of examples as early as possible. This means that an example is passed on to the next classifier only if the response of the current classifier is positive. If any classifier in the cascade computes a negative response for an example, this example is rejected immediately and no longer processed (see Figure 3.3). In order to get an efficient classifier, each cascade stage has to reduce the false positive rate.

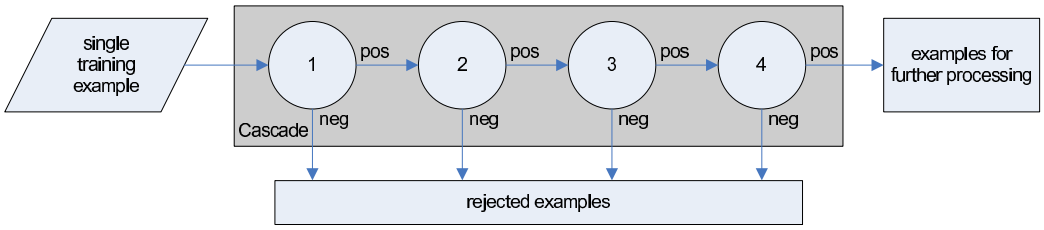


Figure 3.3: Typical structure of a classifier cascade.

Viola and Jones [100] introduced cascades to perform a fast evaluation on large classifiers for object detection. They constructed the cascade stages by training classifiers using Adaboost. Afterwards they adjusted the thresholds to minimize the false negatives. Anyway, lower thresholds not only lead to higher detection rates but also to higher false positive rates. Within their approach they added features until a defined detection rate and the false positive rate are met. Further, stages are added to the cascade until an overall detection rate and false positive rate are met.

The cascades that are used within this thesis are related to Visentini *et al.* [101]. They proposed a framework to build a cascade of on-line updated classifiers within an on-line boosting framework [73]. Each weak learner's training error constitutes the basis for the

level a weak learner belongs to. A low training error indicates a low false negative rate. Ideally the first cascade levels have a low false negative rate and a moderate false positive rate to be able to reject most of the examples in which no object is present. Additionally, they introduced a level threshold. A weak learner is assigned to a cascade level, if its training error is below this threshold θ_l. They compute each examples confidence on a certain level by adding the confidence from the previous level to the confidence of the current level. As usual an example is passed to the next cascade level if its computed confidence on the current level is positive. An algorithmic overview is depicted in Table 3.6.

- Requires randomly initialized strong classifier H, feature pool $\mathcal{F}$, confidence map CM and a base learner model L_0
- For $t = 1, \ldots, T$:
  1. For each subwindow $\mathbf{x}_{t,k}$ in $Frame_t$:
     (a) Apply cascade and fill CM: $CM \leftarrow H_t(\mathbf{x}_t)$
  2. $\mathbf{x}^+_{t,k} \leftarrow \arg\max_k(CM_{t,k})$
  3. $\lambda \leftarrow 1$
  4. $OnlineBoosting(H_t, L_0, (\mathbf{x}^+_{t,k}, 1), \lambda)$ [73]
  5. For each negative sample $\mathbf{x}^-_{t,k}$:
     (a) $\lambda \leftarrow 1$
     (b) $OnlineBoosting(H_t, L_0, (\mathbf{x}^-_{t,k}, 0), \lambda)$ [73]
  6. Build cascade with the features in $\mathcal{F}$
     For $l = 1, \ldots, L$:
     (a) For $h_m \in H, m = 1, \ldots, M$:
        – If $\varepsilon_m \leq \theta_l$ then $H_l \leftarrow h_m$

Table 3.6: Cascaded on-line boosting for feature selection [101].

3.5.2 WaldBoost

Sochman and Matas [87] proposed a method called WaldBoost. It integrates AdaBoost-based measurement selection and Wald's optimal sequential probability ratio test (SPRT) to optimize the classification time. WaldBoost executes the SPRT test using a strong classifier H_T with a sequence of thresholds $\theta_A^{(t)}$ and $\theta_B^{(t)}$. Then, either H_T exceeds a threshold and a decision is made, or the next weak classifier is taken. In the case that no decision is made within T iterations, the input is classified using H_T and user-defined threshold γ. Table 3.7 gives a short overview of the WaldBoost classification procedure. AdaBoost's output is taken as a measurement which is essential for SPRT. According to this measurement the thresholds A and B are determined.

- Given: $h^{(t)}, \theta_A^{(t)}, \theta_B^{(t)}, \gamma, (t = 1, \ldots, T)$
- Input a classified object x.
- For $t = 1, \ldots, T$:
  1. If $H_t(x) \geq \theta_B^{(t)}$ then classify x as positive and terminate
  2. If $H_t(x) \leq \theta_A^{(t)}$ then classify x as negative and terminate
- If $H_T(x) > \gamma$ then classify as positive, else classify as negative

Table 3.7: WaldBoost algorithm [87].

3.6 Summary

In this chapter basic methods on visual learning are discussed using boosting. First, a short introduction to boosting is given. Additionally, weak classifiers and strong classifiers are depicted. Second, the differences between on-line boosting and off-line boosting are presented. Then the idea of using boosting for feature selection depicted in the off-line case and in the on-line case. Further, robust on-line feature selection in context of label noise at the training data is examined, which constitutes the basis for learning object detectors within this work. Finally, the concept of cascades and the WaldBoost algorithm is presented to speed-up the evaluation of large classifiers.

Chapter 4

Multi-Camera Learning[1]

Multi-camera learning is a special variation of multi-view learning. In general, multi-view learning denotes a learning process that takes redundant information from existing data to perform learning. For example, training different classifiers such as decision trees or SVMs, etc. from the same labeled data is sufficient to perform multi-view learning. Moreover, training one classifier with one feature type and training another classifier with a different feature type using the same data would have the same effect. Co-training which was proposed by Blum and Mitchell [8] is a specific multi-view learning method, which uses redundant views on the training data to train one classifier with information that is obtained by another classifier. Leistner *et al.* [52] were one of the first who used co-training to solve a visual object detection task. They assumed different camera views as different views on the data. In addition, Roth *et al.* [83] showed that the training results can be significantly improved if additional cameras are incorporated in the training procedure by using geometric relations additionally.

For this reason, this chapter presents the geometric basics of multi-camera learning. In order to combine information from different cameras. Further, the basics on co-training are also discussed to provide a basis for learning from multiple camera views as introduced by Leistner *et al.* [52]. In brief, Section 4.1 shows a principal scene model. Then, Section 4.2 derives the camera model and it also explains the basics of radial lens distortion, since it is considered within the camera calibration algorithm. Section 4.3 gives an introduction to camera calibration. Further, Section 4.4 discusses point transfers with an arbitrary plane in the 3D scene using a homography, which is a simple method to transfer information from one camera view to another camera view. Then an overview of multi-camera approaches is given in Section 4.5. Finally, the original co-training approach is discussed in Section 4.6 to

[1] The geometric issues within this chapter are widely based on the book of Hartley and Zisserman [43] and Tsai's work [95].

depict the idea of learning from redundant views on the training data. Finally, Section 4.7 presents the baseline approach for this thesis.

4.1 Scene Model

The camera locations and the corresponding poses are kept constant in a training scene during the training process. This means that a set of n cameras with partially overlapping fields of view observe a scene as illustrated in Figure 4.1. The designated objects are assumed to move on a ground plane only. Further, the world coordinate system's origin is placed somewhere in the ground plane such that the z-axis specifies the height within the scene.

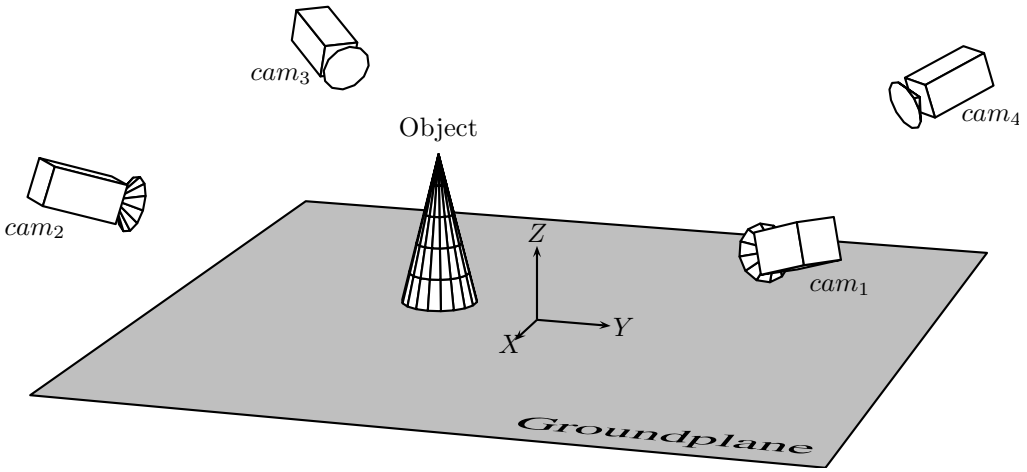


Figure 4.1: A scene containing an object on a ground plane that is observed by multiple cameras.

4.2 Camera Model

Real-world cameras have to be approximated by parametric camera models to be usable for computer vision applications that use geometry information. Hence, a model is needed that describes the camera geometry. For example, the pinhole camera model is a very simple model that expresses the mapping of 3D points onto a 2D plane.

4.2.1 Central Projection and Homogeneous Coordinates

The simple pinhole camera projects a 3D point onto a 2D plane, whereas the camera center $\mathbf{C}$ is placed in the origin of an Euclidean coordinate system. The image plane is placed parallel to the XY-plane of this coordinate system at the distance f in front of the camera.

Thus, f is called the focal length and it is measured along the positive Z-axis. This model is illustrated in Figure 4.2. A line that connects the camera center and a 3D point in front of the camera, pierces the image plane at some point. This special mapping from a 3D point $\mathbf{X}$ onto a point $\mathbf{x}$ in a 2D plane is called a central projection. If this 3D point is placed on the Z-axis of the coordinate system, the pricing point is called principal point $\mathbf{p}$.

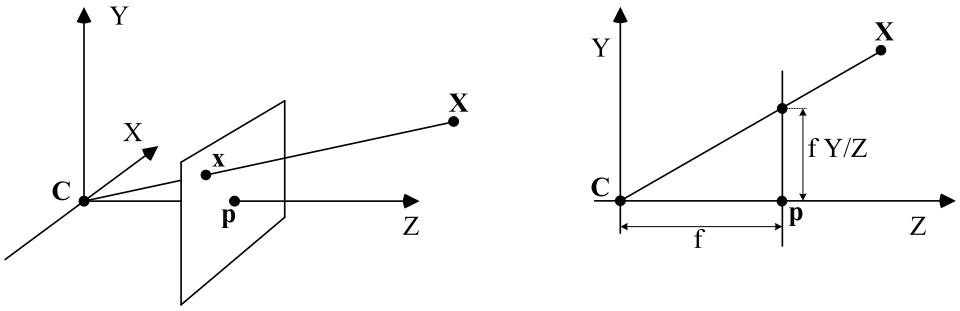


Figure 4.2: Simple pinhole camera model.

Considering homogeneous coordinates, this projective mapping can be expressed as a linear mapping. Assuming a 3D point $\mathbf{X} = (X, Y, Z, 1)^T$ that is multiplied with a 3×4 camera matrix P is a projective mapping onto a 2D point $\mathbf{x} = (x, y, 1)^T$ in the image plane. This can be written as

$$\mathbf{x} = \mathrm{P}\mathbf{X} \quad . \tag{4.1}$$

Using similar triangles $x = f\frac{X}{Z}$, $y = f\frac{Y}{Z}$ and homogeneous coordinates, the central projection for the pinhole camera can be expressed as

$$\begin{pmatrix} fX \\ fY \\ Z \end{pmatrix} = \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} \quad . \tag{4.2}$$

4.2.2 Image Coordinate System

Equation 4.2 is only valid if the origin of the image coordinate system is placed on the principal point $\mathbf{p} = (p_x, p_y)^T$ where the positive Z-axis pierces the image plane. Anyway, for practical issues it is more convenient to place the image coordinate system in an image corner. The camera coordinate system and the image coordinate system are depicted in

Figure 4.3. The mapping

$$\begin{pmatrix} fX + Zp_x \\ fY + Zp_y \\ Z \end{pmatrix} = \begin{bmatrix} f & 0 & p_x & 0 \\ 0 & f & p_y & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} \tag{4.3}$$

performs this translation using homogeneous coordinates. Rewriting this equation leads to

$$\mathbf{x} = \mathrm{K}[\ \mathrm{I}\ |\ \mathbf{0}\ ]\mathbf{X}_{cam} \quad , \tag{4.4}$$

where K defines the camera calibration matrix:

$$\mathrm{K} = \begin{bmatrix} f & 0 & p_x \\ 0 & f & p_y \\ 0 & 0 & 1 \end{bmatrix} \quad . \tag{4.5}$$

Further, I equates to the identity matrix, $\mathbf{0}$ is a zero column vector and $\mathbf{X}_{cam} = (X, Y, Z, 1)^T$, which marks a real-world point in the camera coordinate system. $\mathbf{X}$ is given in homogeneous coordinates.

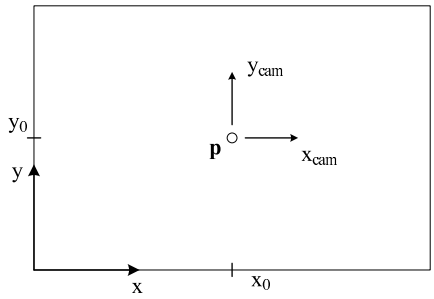


Figure 4.3: Illustration of the camera and the image coordinate systems.

4.2.3 World Coordinate System

Generally real-world points are not expressed in camera coordinates, but in world coordinates. A mapping between these two coordinate systems is performed through rotation and translation as illustrated in Figure 4.4.

A 3D point $\tilde{\mathbf{X}}_w$, that is given in inhomogeneous world coordinates, has to be mapped onto a point $\tilde{\mathbf{X}}_{cam}$ in the camera coordinate system. $\tilde{\mathbf{C}}_w$ defines the camera center in inhomogeneous world coordinates (the origin of the camera coordinate system). Thus,

$$\tilde{\mathbf{X}}_{cam} = \mathrm{R}(\tilde{\mathbf{X}}_w - \tilde{\mathbf{C}}_w) \tag{4.6}$$

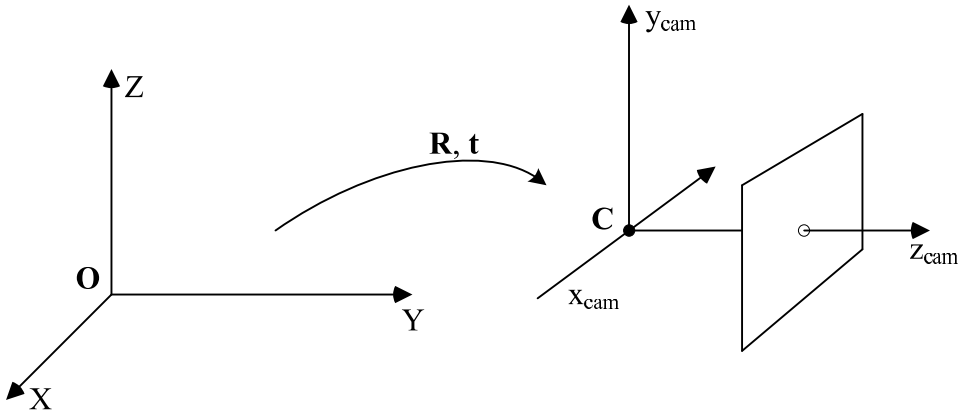


Figure 4.4: A mapping between the camera coordinate system and the world coordinate system that is performed through rotation and translation.

represents the transformation between the coordinate systems. Here R equates to a 3×3 rotation matrix that represents the camera coordinate system's orientation. Rewriting this equation in homogeneous coordinates leads to:

$$\mathbf{X}_{cam} = \begin{bmatrix} \mathrm{R} & -\mathrm{R}\tilde{\mathbf{C}}_w \\ 0 & 1 \end{bmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} \quad . \tag{4.7}$$

Combining Equation 4.4 and Equation 4.7 leads to

$$\mathbf{x} = \mathrm{KR} \left[\begin{array}{c|c} \mathrm{I} & -\tilde{\mathbf{C}}_w \end{array} \right] \mathbf{X}_w \quad , \tag{4.8}$$

where $\mathbf{X}_w$ represents a point in the world coordinate system in homogeneous coordinates. All in all, this pinhole camera model provides 9 degrees of freedom (DoF), where K has 3 DoF, R has 3 DoF and $\tilde{\mathbf{C}}$ has 3 DoF respectively.

In practice, the transformation from the world coordinate system to the image coordinate system is mostly written as $\tilde{\mathbf{X}}_{cam} = \mathrm{R}\tilde{\mathbf{X}}_w + \mathbf{t}$, where $\mathbf{t} = -\mathrm{R}\tilde{\mathbf{C}}_w$ and the camera center is not explicitly considered. Therefore, the camera matrix is computed as

$$\mathrm{P} = \mathrm{K} \left[\begin{array}{c|c} \mathrm{R} & \mathbf{t} \end{array} \right] \quad . \tag{4.9}$$

All parameters in K are called internal camera parameters (f, p_x, p_y) and the parameters R and $\tilde{\mathbf{C}}_\mathbf{w}$ are called external camera parameters respectively.

4.2.4 Finite Projective Camera

A finite projective camera is more general than the pinhole camera, since it has 11 degrees of freedom and an arbitrary scale. This extended camera model is needed for several CCD cameras. The additional degrees of freedom are reached by adding a skew parameter s, which is usually zero for normal cameras and the parameters α_x respectively α_y that represent the camera's focal length in terms of pixel dimensions in each direction. Some CCD cameras do not have squared pixels, in that case the number of pixels per unit x-direction differs from the number of pixels per unit y-direction. $\alpha_x = fm_x$ and $\alpha_y = fm_y$, whereas m_x and m_y correspond to the number of pixels per unit distance measured in image coordinates per direction and f depicts the focal length. Finally, the more general calibration matrix is written as

$$\mathrm{K} = \begin{bmatrix} \alpha_x & s & p_x \\ 0 & \alpha_y & p_y \\ 0 & 0 & 1 \end{bmatrix} \quad . \tag{4.10}$$

4.2.5 Radial Lense Distortion

In theory the camera models map 3D points linearly onto 2D points in the image plane, but in reality a projection error occurs because of imperfect lens manufacture. Figure 4.5 and Figure 4.6 depict the error caused by radial lens distortion. To receive a more exact estimation of the internal and the external camera parameters though camera calibration, this radial distortion can be considered additionally.

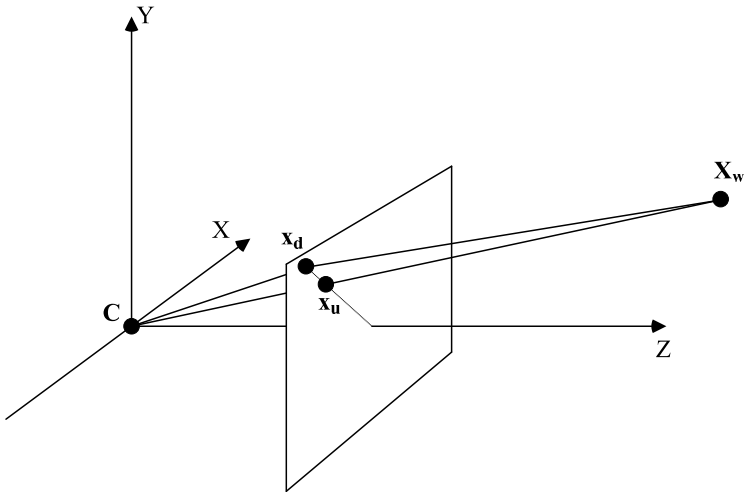


Figure 4.5: Displays the error caused by lens distortion.

The radial distortion is modeled by

$$\begin{pmatrix} x_d \\ y_d \end{pmatrix} = L(r) \begin{pmatrix} x_u \\ y_u \end{pmatrix} \quad , \tag{4.11}$$

where (x_d, y_d) represents the actual image point after radial distortion, (x_u, y_u) is the ideal position of the projected point in a pinhole camera model, $r = \sqrt{x_u^2 + y_u^2}$ that corresponds to the Euclidean distance from the center for the radial distortion and $L(r)$ defines a function that computes the radial distortion factor based on r.

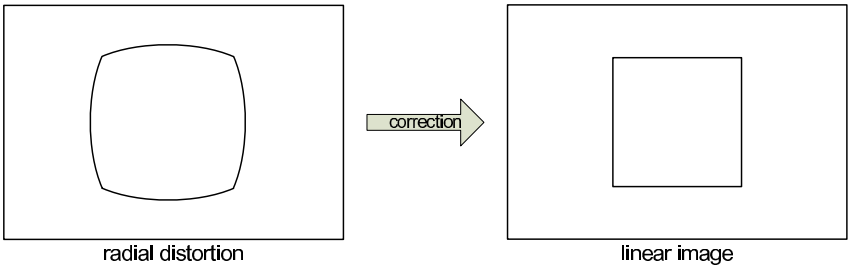


Figure 4.6: Applying a correction on a radial distorted square leads to a square that looks like obtained with a perfect lense.

The distortion is corrected by using an arbitrary function, that is assumed as a Taylor expansion $L(r) = 1 + \kappa_1 r^2 + \kappa_2 r^4 + \ldots$. The coefficients $\kappa_1, \kappa_2, \ldots$ are also considered as internal camera parameters. The following equation computes the corrected point position (x_u, y_u) in pixel coordinates:

$$x_u = x_c + L(r)(x_d - x_c) \qquad y_u = y_c + L(r)(y_d - y_c) \quad , \tag{4.12}$$

where (x_d, y_d) is the measured point in the image plane (distorted coordinates), (x_c, y_c) defines the center of the radial distortion and $r = \sqrt{(x_d - x_c)^2 + (y_d - y_c)^2}$ to take the distance to center of radial distortion into account. The principal point is often assumed to be the center of radial distortion, even though these two points do not coincide exactly. Within this thesis the radial distortion correction is performed using a first order Taylor polynomial, which is sufficient precise.

4.3 Camera Calibration

Many computer vision problems require calibrated cameras (e.g. stereo-vision, structure from motion, change detection, etc.). In general, there are many calibration algorithms available (e.g. [105, 51, 44]). However, within this thesis Tsai's camera calibration algorithm [95] is applied to calibrate the camera and estimate its pose, since it allows to estimate

the internal and external camera parameters using either coplanar or non-coplanar 3D points for calibration. It only requires 3D point to 2D pixel correspondences and does not require a specific calibration target like a chessboard plane. Further, it can estimate the internal and the external camera parameters separately. If desired the external parameters can be estimated only. This property is useful if the camera is moved to a different place in the 3D scene. Within this thesis the world coordinate system's xy-plane ($z_w = 0$) is assumed to be the ground plane to perform the calibration (see Figure 4.1). The origin of the world coordinate system is placed such that it is neither close to the camera view centers nor close to the y-axes of the camera coordinate systems, otherwise it can be simply moved to another location. Further, all points, that are needed for calibration, are taken from the ground plane. Tsai's algorithm estimates the internal and the external parameters within two steps. An algorithmic overview is given in Table 4.1. Tsai's camera model has six fixed intrinsic camera constants additionally to the eleven camera parameters: d_x, d_y, N_{cx}, N_{fx}, d_{px} and d_{py}. d_x and d_y correspond to the center to center distance between adjacent sensor elements in x and y direction respectively, N_{cx} corresponds to the number of sensor elements in x direction and N_{fx} equates to the number of pixels in the frame grabber's x direction. Further, d_{px} and d_{py} define the effective pixel dimensions in x and y direction of the frame grabber.

In the first step Tsai's algorithm requires a set of image coordinates (x_{di}, y_{di}) much larger than five and the corresponding world coordinates (x_{wi}, y_{wi}) to compute the 3D orientation and the xy-position. It uses the point correspondences to compute $\frac{r_1}{t_y}$, $\frac{r_2}{t_y}$, $\frac{t_x}{t_y}$, $\frac{r_4}{t_y}$ and $\frac{r_5}{t_y}$ by solving an overdetermined system of linear equations:

$$\begin{bmatrix} y_{di}x_{wi} & y_{di}y_{wi} & y_{di} & -x_{di}x_{wi} & -x_{di}ywi \end{bmatrix} \begin{bmatrix} \frac{r_1}{t_y} \\ \frac{r_2}{t_y} \\ \frac{t_x}{t_y} \\ \frac{r4}{t_y} \\ \frac{r_5}{t_y} \end{bmatrix} = x_{di} \quad . \tag{4.13}$$

Then $r_1, \ldots, r_9, t_x$ and t_y have to be computed from $\frac{r_1}{t_y}$, $\frac{r_2}{t_y}$, $\frac{t_x}{t_y}$, $\frac{r_4}{t_y}$ and $\frac{r_5}{t_y}$. Therefore, $|t_y|$ is computed using

$$\mathrm{C} \equiv \begin{bmatrix} r'_1 & r'_2 \\ r'_4 & r'_5 \end{bmatrix} \equiv \begin{bmatrix} \frac{r_1}{t_y} & \frac{r_2}{t_y} \\ \frac{r_4}{t_y} & \frac{r_5}{t_y} \end{bmatrix} \quad . \tag{4.14}$$

If a whole column or row does not vanish from C then

$$t_y^2 = \frac{s_r - \sqrt{s_r^2 - 4(r'_1 r'_5 - r'_4 r'_2)^2}}{2(r'_1 r'_5 - r'_4 r'_2)^2} \quad , \tag{4.15}$$

where $s_r = r_1'^2 + r_2'^2 + r_4'^2 + r_5'^2$. Otherwise,

$$t_y^2 = \frac{1}{r_i'^2 + r_j'^2} \quad , \tag{4.16}$$

where r_i' and r_j' correspond to the row or the column in C that do not vanish. At the next step the sign of $|t_y|$ has to be determined. Therefore, an object point (x_{fi}, y_{fi}) has to be picked in image coordinates that is away from the image center with the corresponding world coordinates (x_{wi}, y_{wi}, z_{wi}). Assuming the sign of t_y to be positive, compute

$$\begin{aligned}
r_1 &= \frac{r_1}{t_y} \cdot t_y \quad , && (4.17)\\
r_2 &= \frac{r_2}{t_y} \cdot t_y \quad , && (4.18)\\
r_4 &= \frac{r_4}{t_y} \cdot t_y \quad , && (4.19)\\
r_5 &= \frac{r_5}{t_y} \cdot t_y \quad , && (4.20)\\
t_x &= \frac{t_x}{t_y} \cdot t_y \quad , && (4.21)\\
x &= r_1 x_w + r_2 y_w + t_x \quad , && (4.22)\\
y &= r_4 x_w + r_5 y_w + t_y \quad . && (4.23)
\end{aligned}$$

If the sign of x equates to the sign of x_d and the sign of y equates to the sign of y_d then the sign of t_y is positive, otherwise, it is negative. Based on this result r_1, r_2, r_4, r_5 and t_x have to be recomputed if the sign of t_y is negative. Then determine R:

$$\mathrm{R} = \begin{bmatrix} r_1 & r_2 & \sqrt{1 - r_1^2 - r_2^2} \\ r_4 & r_5 & s\sqrt{1 - r_4^2 - r_5^2} \\ r_7 & r_8 & r_9 \end{bmatrix} \quad , \tag{4.24}$$

where $s = -\operatorname{sign}(r_1 r_4 + r_2 r_5)$. Further, compute r_7, r_8 and r_9 from the outer product of the first two rows using the orthonormal and right-handed property of R.

In the second step compute approximations for f and for t_z by solving an overdetermined system of linear equations without considering radial distortion:

$$\begin{bmatrix} y_i & -d_y y_{di} \end{bmatrix} \begin{bmatrix} f \\ t_z \end{bmatrix} = w_i d_y y_{di} \quad , \tag{4.25}$$

where

$$y_i = r_4 x_{wi} + r_5 y_{wi} + t_y \tag{4.26}$$

and

$$w_i = r_7 x_{wi} + r_8 y_{wi} \quad . \tag{4.27}$$

If $f < 0$ then the rotation matrix has to be corrected as follows:

$$\mathrm{R} = \begin{bmatrix} r_1 & r_2 & -\sqrt{1 - r_1^2 - r_2^2} \\ r_4 & r_5 & -s\sqrt{1 - r_4^2 - r_5^2} \\ -r_7 & -r_8 & r_9 \end{bmatrix} \quad . \tag{4.28}$$

Finally, the exact solutions for f, t_z and κ_1 have to be computed by applying a standard optimization scheme on the mapping that projects 3D points onto the image plane considering radial distortion. The already estimated values of f and t_z with $\kappa_1 = 0$ are used as the initial guess.

Tsai's camera calibration algorithm requires at least five data points due to the fixed internal camera parameters for coplanar calibration. To get an accurate estimation for the lens distortion and the image center parameters the data points have to be distributed broadly across the image. Ideally the data point positions are measured with subpixel accuracy.

1. Compute camera's position (t_x, t_y) and estimate its 3D orientation R
   (a) acquire distorted image coordinates (x_d, y_d)
   (b) acquire the corresponding world coordinates (x_w, y_w)
   (c) compute the five unknowns $\frac{r_1}{t_y}, \frac{r_2}{t_y}, \frac{t_x}{t_y}, \frac{r_4}{t_y}, \frac{r_5}{t_y}$ using point correspondences between (x_d, y_d) and (x_w, y_w, z_w)
   (d) compute $(r_1, \ldots, r_9, t_x, t_y)$ from $\frac{r_1}{t_y}, \frac{r_2}{t_y}, \frac{t_x}{t_y}, \frac{r_4}{t_y}, \frac{r_5}{t_y}$
      i. compute $|t_y|$ from $\frac{r_1}{t_y}, \frac{r_2}{t_y}, \frac{t_x}{t_y}, \frac{r_4}{t_y}, \frac{r_5}{t_y}$
      ii. determine the sign of t_y
      iii. compute 3D rotation matrix $(r_1, \ldots, r_9)$ and t_x
      iv. compute f and if $(f < 0)$ then correct rotation matrix
2. Compute the effective focal length, the distortion coefficients and the camera's t_z position
   (a) compute an approximation of f and t_z by ignoring lens distortion
   (b) compute the exact solution for f, t_z and κ_1

Table 4.1: Tsai's camera calibration algorithm [95].

4.4 Homographies – Point Transfers Using Planes

In general, homographies describe the relation of corresponding image points, that are located in a plane, between two views. For instance, a homography H_π transfers a point $\mathbf{x}$ from one camera view to a point $\mathbf{x}'$ in another camera view using an arbitrary plane π:

$$\mathbf{x}' = \mathrm{H}_\pi \mathbf{x} \quad , \tag{4.29}$$

where $\mathbf{x}$ and $\mathbf{x}'$ are the projections of any 3D point $\mathbf{X}$ on the plane π. π does not have to meet the camera centers $\mathbf{C}$ and $\mathbf{C}'$. This point correspondence is depicted in Figure 4.7.

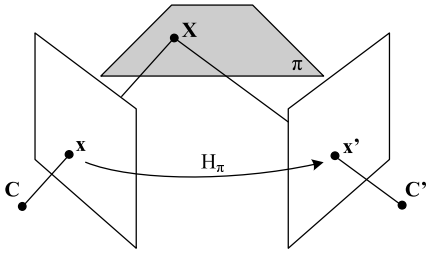


Figure 4.7: A homography is applied to transfer points, that are on the plane π, from one camera view to another camera view.

Any plane is estimated by selecting at least four point correspondences per camera view on the plane. The estimation can be computed using the Direct Linear Transformation (DLT) [43], for example.

Further, the homography between a world plane with $z = 0$ and the image plane of a calibrated camera $\mathrm{P} = \mathrm{K}[\ \mathrm{R} \ \big|\ \mathbf{t}\ ]$ is defined as:

$$\mathrm{H} = \mathrm{K} \begin{bmatrix} \mathbf{r_1} & \mathbf{r_2} & \mathbf{t} \end{bmatrix} \quad , \tag{4.30}$$

where $\mathbf{r_i}$ corresponds to the columns of R, since

$$\mathbf{x} = \mathrm{P}\mathbf{X} = \begin{bmatrix} \mathbf{p}_1 & \mathbf{p}_2 & \mathbf{p}_3 & \mathbf{p}_4 \end{bmatrix} \begin{pmatrix} X \\ Y \\ 0 \\ 1 \end{pmatrix} = \begin{bmatrix} \mathbf{p}_1 & \mathbf{p}_2 & \mathbf{p}_4 \end{bmatrix} \begin{pmatrix} X \\ Y \\ 1 \end{pmatrix} \quad . \tag{4.31}$$

4.5 Multi-Camera Approaches

Nowadays, multi-camera approaches consider classifier responses and additionally take knowledge about the scene geometry into account to address the problem of occlusions.

Using both leads to more accurate detections with less false positives. For example, Berclaz *et al.* [4] apply a fixed pretrained classifier repeatedly to each 3D position per camera view. This leads to one map of classifier responses per camera. Then, they perform a post-processing and combine all maps with a probabilistic approach to yield the final detections. Kim *et al.* [49] also use a fixed pretrained classifier to detect objects in a multi-camera configuration.

Other works use change detections to derive foreground information instead of using classifiers. For example, Khan and Shah [47] classify individual pixels as background or foreground (i.e. moving objects). They map all foreground points to the ground plane using planar homographies and finally multiply the mapped points to segment each person's feet region. Fleuret *et al.* [31] did not detect or track people in each camera view. They reformulated the problem and used a 2D occupancy grid which is placed parallel to the ground plane to gather evidences form all views about the presence of persons at all ground point locations. Then they used this information to detect and track persons using the occupancy grid information. Franco and Boyer [32] even used a 3D occupancy grid. Further, Eshel and Moses [29] used a head detector and multiple cameras to estimate person's locations in dense crowds. Mueller *et al.* [62] also project all detected objects in a common reference and then they mark the nearest objects which have the same size and center of gravity with the same label. By contrast, Caspi *et al.* [12] and Orwell *et al.* [70] track objects in the single camera views and finally they fuse estimated trajectories to identify objects.

However, there are also works which share information directly between cameras without projecting each view to a common reference plane, for example, Stauffer and Tieu [89], Black *et al.* [7] and Leistner *et al.* [52]. They use planar homographies between the camera views to transfer information. Hence, they suffer to solve the occlusion problem.

4.6 Co-Training

Co-training was originally presented by Blum and Mitchell [8]. It is a semi-supervised learning algorithm, that requires two distinct views for each example. At first two separate classifiers are trained using labeled data for both views. Afterwards each classifier has to classify unlabeled data and has to predict the examples' labels that are required to update the other classifier with the obtained information. This method is based on two assumptions that have to be satisfied:

1. Features of each view are sufficient descriptive so that the classifiers' predicted labels can be trusted.
2. Features in one view are conditionally independent from the features in the other view.

The Co-training algorithm is depicted in Table 4.2. It uses as set of L labeled examples $\mathbf{x}$ and a set U of unlabeled examples. Based on U co-training extracts u random examples to create a set U'. Then two classifiers h_1 and h_2 are trained using the x_1 and the x_2 portion of the example $\mathbf{x}$ respectively. Afterwards, h_1 and h_2 are employed to label p positive and n negative examples from U'. The self-labeled examples are added to L. Finally, $2p + 2n$ examples are randomly selected from U to replenish U'. Then h_1 and h_2 are trained again. The training and labeling steps are performed K times to generate co-trained classifiers. The two conditions proposed by Blum and Mitchell are often hard to fulfill in practice and were therefore relaxed in [2]. Anyway, the training algorithm should never provide hypotheses that are confident but wrong.

Levin *et al.* [54] proposed a visual detection system using a co-training approach to improve a pair of classifiers. They used a grey image classifier and a background difference classifier to fulfill the requirement of two independent views as proposed by Blum and Mitchell [8]. Their grey image classifier used the grey scale images directly, whereas the background difference classifier evaluates the difference between each video image and the video image's average background that is computed from the whole video clip. There are some extensions to the original co-training approach. For example, Javed *et al.* [46] extended the co-training approach for on-line learning. They also used a background model to perform classification on moving objects only. Further, they used different feature types to get two independent views on the data to process co-training. Finally, they used a combination of all features for classification. Another extension is presented by Zhou and Li [106]. They proposed tri-training that is an extension to co-training by adding a third classifier. In detail, they trained three classifiers on a small set of labeled examples and refined the classifiers by processing unlabeled data afterwards. This means that one unlabeled example receives a label, that is determined by a majority decision of all available classifiers, per tri-training iteration.

- Requires a set L of labeled training examples and a set U of unlabeled examples
- Create a pool U' of examples by choosing u examples at random from U
- For $k = 1 \ldots K$:
  1. Use L to train a classifier h_1 that considers only the x_1 portion of x
  2. Use L to train a classifier h_2 that considers only the x_2 portion of x
  3. Allow h_1 to label p positive and n negative examples from U'
  4. Allow h_2 to label p positive and n negative examples from U'
  5. Add these self-labeled examples to L
  6. Randomly choose $2p + 2n$ examples from U to replenish U'

Table 4.2: Co-training algorithm by Blum and Mitchell [8].

4.7 Baseline Approach

Leistner *et al.* [52] introduced a multi-camera learning approach within a co-training framework. They further used geometric relations to share information between the camera views. Roth *et al.* [83] extended this approach to assure more robust learning. Hence, this extended approach constitutes the baseline for this thesis (see Table 4.3). The approach which was proposed by Leistner *et al.* performed suboptimal classifier updates which were caused by projection errors due to geometric inaccuracies. To limit these errors Roth *et al.* performed an additional alignment check on projected information to prevent these suboptimal classifier updates.

The multi-camera approach uses simple homographies (see Section 4.4) to transfer points from one camera view into another camera view using the ground plane. This information transfer is illustrated in Figure 4.8 at the use of four cameras. Further, a general classifier H^P is trained using off-line AdaBoost for feature selection to initialize the multi-camera co-training approach. The off-line boosting algorithm is applied on a fixed set of positive and negative training patches. Afterwards, this initial classifier is cloned to assign one copy to each camera. Then each camera's initial classifier is improved by performing on-line classifier updates using a co-training strategy. Anyway, each camera view is considered as an independent view on the data for co-training, where the initial classifiers lead to independent observations and predictions. Hence, the independence criteria of views on the data, that is required for co-training (see Section 4.6), is fulfilled.

In the this case co-training is performed on the camera classifiers by verifying or falsifying each prediction by using information from another camera view. Further, a decentralized camera setup is assumed where only limited bandwidth is available (i.e. the data

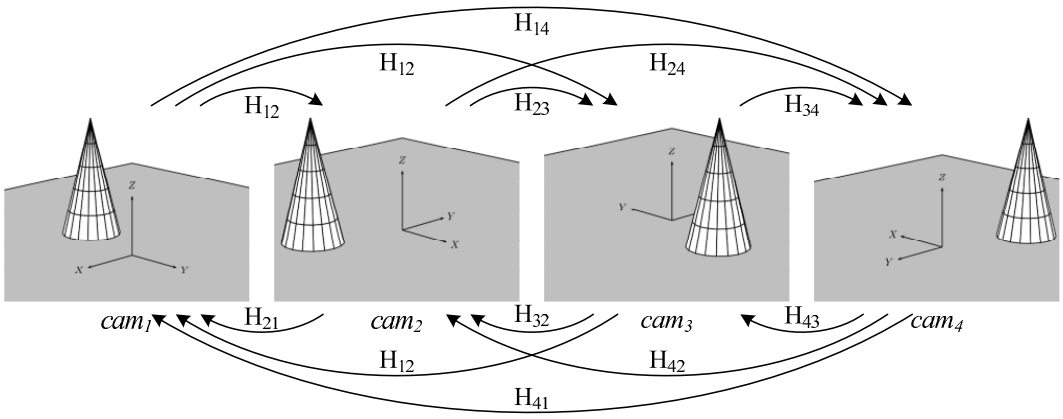


Figure 4.8: The baseline approach uses pairwise homographies to transfer information from one camera view to another camera views.

exchange between the cameras has to be minimized). Therefore, a confidence threshold Θ is introduced to come along with this restriction. Θ classifies each camera's predictions. Each prediction H_i at camera i with a confidence value larger than Θ is assumed to be *correct* and not further considered. On the other hand, if H_i predicts a confidence value that is smaller than Θ, the corresponding confidences of the other cameras are considered additionally. To avoid centralized information merging, a simple strategy is proposed: each camera has its own confidence map which is overlaid with requested confidences from other cameras. Then, non-maximum and non-minimum suppression are performed on the merged confidence map to extract patches which have the highest positive and the highest negative disagreement to perform updates. Additionally a conservative verification and falsification step is performed to increase the classifier stability. At the *verification* and *falsification* step all classifier responses are examined. If the responses of all classifiers are positive, the patch is verified and added to the positive training data set. Otherwise, if the responses of all other classifiers are negative, a negative update is performed on the classifier H_i and the example is falsified. After each negative update the positive training data set is checked for consistency. If the positive training set is no longer consistent, a positive update is performed with the corresponding example. Table 4.4 summarizes this co-training update strategy.

- Train an initial classifier H^P
- Clone H^P n times
- Assign one copy to each camera
- While t
  1. update i classifiers H_i^{t-1} (Table 4.4)

Table 4.3: Co-training with partly overlapping camera views [83].

- Requires K detections x_k and H_i^{t-1}
- For $k = 1, \ldots, K$:
  1. If $H_i^{t-1}(x_k) > \Theta$ then
     (a) For $j = 1, \ldots, J$:
        i. Project x_k onto other views: $x_j = \mathrm{H}_{ij} x_k$
        ii. Evaluate $H_j^{t-1}(x_j)$
        iii. Check alignment: $align(x_j)$
        iv. If $H_j^{t-1}(x_j) > \Theta$ & $align(x_j) = true$ then
            A. $update(H_i^{t-1}, x_k, +)$
        v. Else if $H_j^{t-1}(x_j) < 0$ then
            A. $update(H_i^{t-1}, x_k, -)$

Table 4.4: Co-training update strategy [83].

4.8 Summary

This chapter derives the geometric basics for sharing information between multiple cameras, which are widely base on the book of Hartley and Zisserman [43] and on Tsai's work [95]. First, the scene model is depicted. Then, a camera model is derived and an introduction to correction of radial lens distortion is given. Further, Tsai's camera calibration algorithm is discussed to estimate the internal and external camera parameters. The calibration is required for efficient centralized information fusion as proposed within this work. Finally, multi-camera approaches for detecting objects are presented. Further, details on co-training are given and the baseline approach for this thesis is discussed.

Chapter 5

A Centralized Approach to Multi-View Learning

This chapter introduces a centralized approach to learn scene specific classifiers in a multi-camera network. A centralized approach has several advantages compared to the baseline approach as discussed in Section 4.7. The centralized approach collects information from an arbitrary number of cameras without increasing the computational effort of merging the information. Similar to the baseline approach at least two cameras have to be used. The baseline approach uses simple pairwise homographies to transfer information from one camera view to another camera view. Anyway, the number of homographies increases quadratically if the number of cameras increases linearly. For instance, a setup consisting of two cameras requires two homographies to provide each camera's detections in the opposite camera view. Using three cameras would require six homographies to achieve the same result and so on. In general, N cameras require $N(N-1)$ homographies to share information within all cameras. Thus, the required effort to determine the homographies also increases quadratically. On the contrary, the centralized approach requires calibrated cameras only, since all information is projected to a single central map. This means that the required effort is restricted to calibrate each camera once. Further, the information merging algorithm is completely independent from the subjacent learning algorithm, insofar the learning algorithm returns confidence values for all detections and supports on-line learning.

In short, Section 5.1 gives an overview of the centralized approach. Section 5.2 discusses the object detection process for single camera views. Further, Section 5.3 derives the required geometry to transfer information from the top view map to the single camera views and vice versa. The initialization of the centralized approach is introduced in Section 5.4. Then, Section 5.5 describes the process of collecting and fusing the detection information in a central map. Section 5.6 proposes the method to extract positive examples using the

fused information. Section 5.7 discusses the process of gathering negative examples using a background model or bootstrapping. Finally, Section 5.8 presents cascades that speed-up the evaluation of large classifiers.

5.1 Centralized Approach – Overview

The centralized approach requires classifiers for each camera within the multi-camera network. These classifiers should be adaptive to assure a high detection performance even if the light or the environment itself slightly change within the training scene. Additionally, these initial classifiers have to be created with minimum labeling effort. To fulfill these properties an on-line training algorithm (see Section 3.4) has to be deployed to train the classifiers. Notice, that the initial classifiers can be also trained with an off-line learning algorithm, since Roth *et al.* [81] showed that off-line trained classifiers can be easily re-trained with on-line learning algorithms. Anyway, for simplicity the initial classifiers are trained with an on-line learning algorithm. These initial classifiers are generated by using about 20 positive and 20 negative examples, which are randomly selected from a much larger dataset, for training (see Section 5.4).

To collect information from all cameras within the multi-camera network each camera view has to perform an object detection process on its own (see Section 5.2). Thus, the base point of each detection are projected into a common top view map on the ground plane using the camera calibration (see Section 5.3.2). The next step fuses this information and extracts locations on which the designated objects are located most likely (see Section 5.5. Finally, these extracted locations are projected to the camera views to extract positive examples (see Section 5.6), with which all classifiers are updated immediately to implicitly improve the classifiers. Additionally, an occlusion check is performed in each camera view before an example is extracted to circumvent suboptimal updates. Further, negative updates have to be performed to keep the classifier statistics balanced. The negative examples are extracted by bootstrapping and evaluating foreground information of detections using a background model (see Section 5.7). Optionally, cascades can be deployed to efficiently evaluate large classifiers (see Section 5.8).

5.2 Single View Detection Process

A sliding window approach that uses a single classifier or a cascade of classifiers accomplishes the detection on various scales and locations as introduced by Viola and Jones [100]. In general, the detection window is shifted by Δx and Δy to reach different locations. Different scales are achieved by scaling the detection window itself. Note, that this method is

much more efficient than scaling the whole image. In addition, the current scale s affects the shift distance Δ. Therefore, $\mathrm{round}(s\Delta)$ computes the effective shift distance on various scales.

Within a multi-camera scene the possible sub-window scales and locations can be shortened. In this case a ground plane is estimated and the sliding window is restricted to positions on the ground plane, since it is assumed that the objects can move on the ground plane only. Further, it is assumed that the object size decreases if an object moves from the front to the back in a camera's field of view. Hence, all possible scales are restricted to the relative maximum object size that is related to the ground plane location. As a result the sliding window positions have to be computed for each camera view separately.

Another approach would be to discretize the ground plane and place sub windows at each discrete ground plane position. Then these locations are transferred to each camera view as proposed in [31]. This approach has the disadvantage that locations in the front are scanned on a more coarse level than locations in the back of a camera's field of view. This means that a camera's more flat angle of view leads to larger differences on the scanning resolution between the foreground and the background locations and varying errors due to projection and detection inaccuracies.

5.3 Information Transfer Between Cameras and a Central Map

Assume that the ground plane is observed from a point that is located orthogonally above this plane. Then a map is placed on the ground plane such that the world coordinate system's XY-plane coincides with the top view map. Further, this map stores all locations of detections that are reported by each camera classifier after a post-processing step. Hence, all classifier detections that occur on the ground plane have to be transferred from the recorded image locations to their corresponding locations in the top view map. For this purpose, all cameras have to be calibrated to measure each detection's ground plane coordinates in the world coordinate system. This means that an inverse projection from the image coordinate system to the ground plane has to be performed to consider the radial lens distortion by using the camera model which was derived in Section 4.2. If there is no need to consider the radial lens distortion a simple homography can be applied to convert the coordinates as expressed in Section 4.4. In detail, a detection is transferred from a camera view to the central map by transfusing its base point via backward projection (see Section 5.3.2). A forward projection is performed to transfer information from the top view map to the camera views (see Section 5.3.1). Figure 5.1 illustrates the information transfer using a centralized map with four cameras. The basics to transfer information between camera views and the top view map are discussed in Chapter 4.

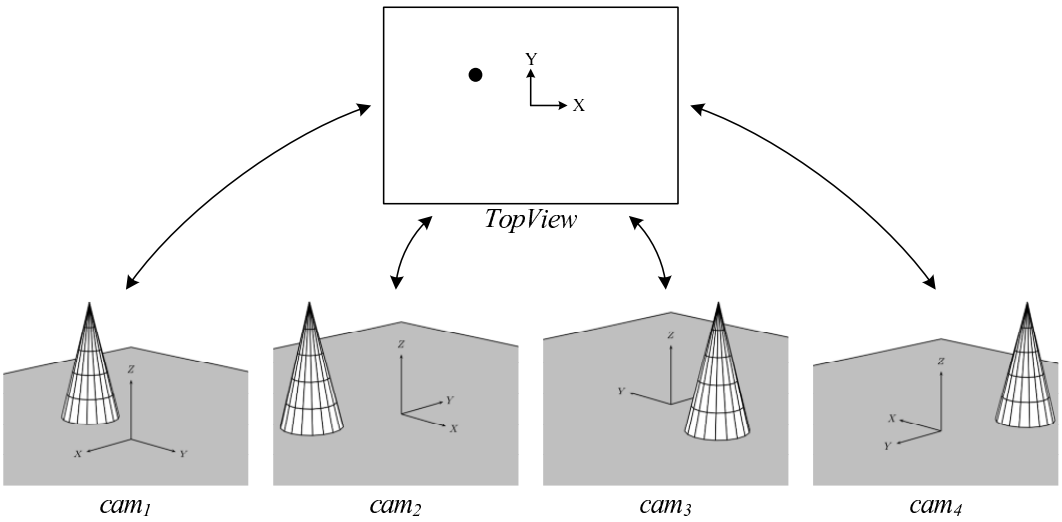


Figure 5.1: Information transfer using a centralized top view map.

5.3.1 Forward Projection Considering Radial Distortion

The internal and external camera parameters are estimated with Tsai's algorithm (see Section 4.3). Thus, the mapping from world coordinates to the pixel coordinates are computed as follows [95]. The translation from the world coordinates to camera coordinates equates to (see Section 4.2.3),

$$\mathbf{x}_{cam} = \mathrm{R}\mathbf{X}_w + \mathbf{t} \quad . \tag{5.1}$$

Then, the undistorted sensor coordinates are,

$$x_u = f\frac{x_{cam}}{z_{cam}}s_x \tag{5.2}$$

$$y_u = f\frac{y_{cam}}{z_{cam}} \quad . \tag{5.3}$$

Further, use the radial distortion model (see Section 4.2.5)

$$x_u = x_d L(r) \tag{5.4}$$

$$y_u = y_d L(r) \quad , \tag{5.5}$$

where $r = \sqrt{x_d^2 + y_d^2}$, to compute the distorted sensor coordinates. Finally, the image coordinates are,

$$x_i = \frac{s_x x_d}{d_x} + p_x \tag{5.6}$$

$$y_i = \frac{y_d}{d_y} + p_y \quad . \tag{5.7}$$

5.3.2 Backward Projection Considering Radial Distortion

The back-projection from image coordinates to the real world works as follows. First, compute the distorted sensor coordinates:

$$x_d = (x_i - p_x)\frac{d_x}{s_x} \tag{5.8}$$

$$y_d = (y_i - p_y)d_x \quad . \tag{5.9}$$

Based on the x_d and y_d the undistorted sensor coordinates have to be computed using Equation 5.4 and Equation 5.5. This undistorted point is then back-projected as a ray in the real world using the camera center. Then the ray is intersected with the ground plane ($z = 0$ in the world coordinate system). Using the relation depicted in Equation 4.31 leads to

$$\begin{bmatrix} X_w \\ Y_w \\ s_w \end{bmatrix} = \left(\begin{bmatrix} f & 0 & 0 \\ 0 & f & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & t_x \\ r_{21} & r_{22} & t_y \\ r_{31} & r_{32} & t_z \end{bmatrix} \right)^{-1} \begin{bmatrix} x_u \\ y_u \\ 1 \end{bmatrix} \quad . \tag{5.10}$$

Finally, the inhomogeneous ground plane coordinate is

$$\tilde{\mathbf{X}}_w = \begin{bmatrix} \frac{X_w}{s_w} & \frac{Y_w}{s_w} & 0 \end{bmatrix}^T \quad . \tag{5.11}$$

5.4 Initialize the Centralized Approach

The centralized learning approach requires $N \geq 2$ calibrated cameras, that observe a 3D scene from arbitrary angles of view. These calibrated cameras allow the framework to forward the detection information to a central top view map (see Section 5.3). Further, an initial classifier for each camera is required. Within this thesis the initial classifier is trained with an on-line *GradientBoost* algorithm using a logistic loss-function as depicted in Section 3.4. The initial classifier is trained with a small number of positive and negative examples only (e.g. 20 positive and 20 negative).

In general, the proposed approach is completely independent from the learning algorithm. Anyway, an off-line learning algorithm is not applicable to generate adaptive classifiers, since the classifier has to change during the training process and additionally has to be available all the time. Similar to the baseline approach, the initial classifier is trained on a desired object category using some positive and negative training examples. Then, the classifier is duplicated N times and one copy is assigned to each camera. These initial classifiers detect objects from time to time, which is essential and sufficient for merging the information to finally obtain positive updates. Using these updates the classifier improves

and finally yields excellent single camera view detection performance on the training scene. A summary of the initialization procedure is given in Table 5.1.

- Requires $N \geq 2$ calibrated cameras and a small set of positive and negative training examples ($\mathcal{X}^+$ and $\mathcal{X}^-$).
- Train an initial on-line classifier using $\mathcal{X}^+$ and $\mathcal{X}^-$.
- Clone the classifier N times.
- Assign one copy to each camera.
- Create top view map.

Table 5.1: Initializing the centralized approach.

5.5 Combining Information from Multiple Views

Each camera's classifier returns a confidence value for each classified sub-window within its own view. A threshold is applied on the confidence values to discard very weak and negative responses. Moreover, a simple but efficient non-maximum-suppression (NMS) [64] is applied on the threshold output to ensure that each object is detected only once in each camera's field of view. This aggressive post-processing leads to a very sparse list of detections at each camera view and may cause a loss of information, but it alleviates the following merging process. The remaining detections, that survive the post-processing, are transferred to the central top view map considering geometry as introduced in Section 5.3. As a result, the top view map stores the confidence values from all detections at each camera view. Additionally, it retains information about the camera views in which the detections occurred.

One problem arises from the fact that many detections in the different camera views are shifted. This leads to misaligned entries in the top view map. In addition, there is a projection error, since the camera calibration is also imperfect. The baseline approach applied an extra alignment check to handle misaligned detections, but within this approach an additional alignment check in the single camera views is not required, since the proposed approach transfers all information into a centralized map. Thus, corresponding detections from different camera views have to be fused in the top view map. To perform this task, check the distance between neighboring points in the top view map to find corresponding

detections from different views:

$$d(\mathbf{p}_i, \mathbf{p}_j) = \sqrt{(p_{ix} - p_{jx})^2 + (p_{iy} - p_{jy})^2} \quad . \tag{5.12}$$

Further, a central camera count is introduced. This camera count is incremented by one for each detection entry in the top view map, if the distance to a neighboring point, that belongs to another camera view, is smaller than a threshold Θ.

Next, a probability density map is created on the basis of this camera count map and the corresponding confidence values in the top view map. In particular, top view items are transferred to the density map only, if $c \geq 2$ at the point's location. In other words an agreement of at least two camera views is required to gain suitable locations for positive updates. Then, a mean-shift clustering algorithm is applied on the resulting density map to find all its local maxima $P_\chi = \{p_{\chi,1}, \ldots, p_{\chi,k}\}$. Finally, P_χ represents locations on the ground plane on which the desired objects are most likely located. Table 5.2 summarizes the information combining process formally.

For each input image:

- Clear top view map.
- For $n = 1, \ldots, N$:
  1. Detect objects in camera view n.
  2. Apply NMS on the detection output $\rightarrow \mathcal{D}_n$.
  3. Transfer $\mathcal{D}_n$ into the top view map.
- If ($\exists\, d(\mathbf{p}_i, \mathbf{p}_j) < \Theta \;\; \forall\, j \neq i$,) and
  (camera view of $\mathbf{p}_i \neq$ camera view of $\mathbf{p}_j$) then
  1. Increment camera count at $\mathbf{p}_i$
- Top view map $\rightarrow$ probability density map $\forall\, \mathbf{p}_i$ with camera count ≥ 2
- Apply mean-shift algorithm on probability density map $\rightarrow$ local maxima $P_\chi = \{p_{\chi,1}, \ldots, p_{\chi,k}\}$.
- Output: P_χ

Table 5.2: Combining detection information from multiple views to extract possible ground plane locations for positive updates.

5.6 Gathering Positive Updates

Eventually, it is the goal to gain positive training examples from the 3D scene that is observed by an arbitrary number of cameras. The information merging algorithm, which is introduced in Section 5.5, combines detections from all camera views in a central map and provides a set of locations P_χ. These positions depict locations on the ground plane where the desired objects are located most likely. Hence, these ground plane locations that are back-projected to the single camera views and restore its corresponding bounding box to obtain a positive example.

In detail, all locations in P_χ are projected to each camera view in which the corresponding detections originated. Then their corresponding bounding boxes (see Section 5.2) are restored. Anyway, one problem remains. Assume that there are two objects moving on the ground plane. Two cameras observe the scene. One camera detects both objects and the second camera detects only one object. Unfortunately, this object is partly occluded by the second object which is not detected in this camera view. In this case a suboptimal update would be performed on both cameras using the patch with the partly occluded object. Such suboptimal updates may corrupt the classifiers and have to be avoided.

Therefore, an additional occlusion check is performed at each camera view to ensure robust learning with fully visible objects. The occlusion check evaluates if the back projected points and its restored bounding boxes are completely visible. In detail, each camera view's occlusion check considers all detections that survived the post-processing and were added to the top view map. It computes the overlap with the back projected bounding boxes. Positive updates are not performed if the overlap between any detection and a back projected bounding box exceeds a threshold Φ. The overlap is computed as follows:

$$\text{overlap}(rect_1, rect_2) = \frac{\text{area}(rect_1 \cap rect_2)}{\min(\text{area}(rect_1), \text{area}(rect_2))} \quad . \tag{5.13}$$

A summary of this process is depicted in Table 5.3.

- Requires P_χ with K locations from N cameras.
- For $n = 1, \ldots, N$:
  1. For $k = 1, \ldots, K$:
     (a) Project $p_{\chi,k}$ into camera view n.
     (b) Compute the corresponding bbox.
     (c) Compute overlap between $p_{\chi,k}$ and $\mathcal{D}_n$.
     (d) If $\text{overlap}(\mathcal{D}_{n,j}, \text{bbox}(p_{\chi,k})) < \Phi \; \forall \; j \neq k$ then
        i. Positive update with $\text{bbox}(p_{\chi,k})$ at all classifiers

Table 5.3: Generating positive updates using locations that are determined by combining detection information from multiple views (see Table 5.2).

5.7 Gathering Negative Updates

Negative examples are much easier to obtain than positive examples. For instance, the integration of a background model to perform background subtraction or simple bootstrapping are excellent methods to extract negative examples from any external database or video sequence. Bootstrapping negative examples from an external database is a very common approach to improve a classifiers performance by generating hard negative examples. Details on bootstrapping are given in Section 5.7.1. Using an external database to train a classifier has the advantage that the classifier becomes more general, since a huge variety of negative examples can be obtained. This means that the classifier will have a high performance on arbitrary scenes and not on the training scene only. On the other hand using a background model allows to extract negative examples from the training scene. In this case the classifier does not generalize as good as using an external database, because only scene specific negative examples are processed, but it enables the classifier to specialize on one specific scene. Obtaining negative examples with background models is discussed in Section 5.7.2.

5.7.1 Bootstrapping

To obtain negative updates with a bootstrapping method any set of images that do not contain positive examples of the desired object can be processed. Notice, in statistics bootstrapping is a resampling method that randomly extracts data from already existing data. It was originally proposed by Efron [27]. Therefore, bootstrapping is performed if the number of examples is insufficient to derive accurate statistics. In computer vision bootstrapping is sometimes also referred as self-learning. In this case a classifier is trained using all available positive and negative training examples. Then this classifier is applied on a set of

images that do not contain positive examples. For this reason, all detections within this set of images are false positives. Then the classifier is retrained with this set of false positive examples to gain an improvement. In general, these steps are repeated several times to increase the classifier's performance. Further, approaches that also used a bootstrapping method to improve the classifier performance are proposed in [92] and [76].

5.7.2 Use Background Subtraction

As already mentioned, movable objects can be identified by performing background subtraction in video sequences. Section 2.4 discusses several background models and methods to identify foreground pixels by using a background model. Accordingly, this foreground information is essential to extract negative examples from a training sequence. This method has the advantage that it is very easy to obtain negative examples, since it has to be checked only if foreground pixels are present in an update patch. On the other hand, the obtained negative examples are scene specific and thus the updates lead to a more specialized classifier for a specific training scene.

Within this thesis the background model computation is very similar to McFarlane's and Schofield's approach [60]. Instead of incrementing and decrementing the approximated median by one an arbitrarily fixed step size is used. Then the foreground pixels are identified by computing the absolute difference between the approximated median and the pixel values as expressed in Equation 2.14. In detail, the negative examples are obtained by checking all detections on foreground information. Assume a detection window that contains only a few foreground pixels and all these pixels are additionally placed near the detection window's borders. In this case the training algorithm immediately performs a negative update on the classifier that has developed the detection. Anyway, if the number of foreground pixels is below a threshold compared to the patch size, then the detection is also assumed to be a false positive.

5.8 Improving Performance with Cascades

The principal concept of cascades is presented in Section 3.5.1. The cascades that are used within this thesis are related to the approach of Visentini *et al.* [101]. They introduced a concept that fits into Oza's on-line boosting approach (see Section 3.3). By contrast, within this work Grabner's and Bischof's [40] on-line boosting approach is used. Hence, to come along with the implementation of selectors, the number of cascade stages and its corresponding classifier sizes have to be defined in advance.

Generally, low cascade levels consist of small classifiers and higher levels consist of more complex classifiers, since the examples that have to be processed in higher levels are

more difficult. Thus, to initialize the cascade, each cascade stage's classifier is initialized randomly. This means that a feature pool is generated for each classifier at random using the predefined number of selectors and the predefined number of weak learners per selector. Afterwards, the cascade is trained stage-to-stage such that each stage's classifier selects its best weak learners. In detail, a negative training example is passed to the next cascade stage if the current stage is unable to learn and predict it correctly. By contrast, positive examples are always processed at all cascade stages.

5.9 Summary

In this chapter a novel centralized approach for information fusion is introduced. The differences to the baseline approach are discussed. It describes the process of object detection in single camera views and the procedure of collecting and merging this information in a central map. Based on this map, positive examples are extracted from the unlabeled training scene and immediately deployed for training the object detector. Further, methods are discussed to also extract negative examples from unlabeled training data. Finally, the usage of cascades is presented in this context to speed-up the evaluation of large classifiers.

Chapter 6

Experimental Results

This chapter illustrates several experiments with various data sets. Multi-camera learning is performed with the centralized approach, which requires calibrated cameras during training. By contrast, the evaluation procedure does not require calibrated cameras, since each camera view is evaluated on its own, if multiple camera views are available. The intrinsic and extrinsic camera parameters are determined by Tsai's camera calibration algorithm [95] (see Section 4.3). In practice, Tsai's camera calibration toolbox[1] was deployed to estimate the parameters. The performance of the centralized approach is evaluated on learning a person detector. This special scenario was taken, because there are various reference implementations and many benchmark datasets publicly available. Further, on-line boosting is selected to be the subjacent learning algorithm (see Section 3.1). However, the centralized approach is neither limited to learn person detectors nor it is restricted to a certain learning algorithm.

In brief, Section 6.1 explains the training and evaluation methodology with the appertaining performance measures. Section 6.2 presents the training and test data sets that constitute the basis for all experiments. The basic setup of all experiments is introduced in Section 6.3. Section 6.4 illustrates the learning behavior over time for each classifier within a multi-camera setup using the centralized approach. Then the implemented feature types are compared within Section 6.5. Section 6.6 compares the on-line histograms and on-line nearest neighbor classifiers as weak learners. Further, Section 6.7 shows the classifier results if the training is performed with an increasing number of cameras. Finally, Section 6.8 presents the performance of scene specific classifiers. Section 6.9 compares the results obtained with the baseline approach and the centralized approach and Section 6.10 demonstrates generalization of the classifiers.

[1] `http://www.cs.cmu.edu/~rgw/TsaiCode.html`

6.1 Evaluation Methodology

The evaluation methodology strongly influences an experiment's results. In general, there are several alternatives to evaluate a multi-camera object detection system. For instance, an evaluation method on the basis of a top view map is useful if the goal is to detect all objects in a scene whether all objects are visible in each view or not.

Within this thesis all classifiers are evaluated on single views only, since the aim is to improve each camera's classifier by using the centralized multi-camera learning approach. Hence, there is no collaboration between the cameras during evaluation.

6.1.1 Per-Window versus Per-Image Evaluation

As already discussed by Dollár *et al.* [24] there are two methods to compare detection results. On the one hand there is the per-window evaluation method and on the other hand there is the per-image evaluation method. Per-window evaluation does not consider effects caused by NMS. Using this method a binary classifier only has to decide if a cropped image patch contains an object or not. Within this thesis such an evaluation approach would not make sense, because the objects additionally have to be located in the images to obtain positive training examples. For this purpose all experiments use an per-image evaluation method to produce viable results.

Per-image evaluation considers the whole image. Therefore, rectangular detection information is processed to determine the evaluation results of the whole image. This means that the detection system additionally has to perform a NMS or any similar post processing method to avoid multiple detections of the same object. It also has to return all detected objects' bounding boxes. Additionally, ground truth data have to be available for the evaluation images. A detection counts as true positive if the overlap a_0 between the detection's bounding box B_{det} and the corresponding ground truth bounding box B_{gt} exceeds 50% as introduced in the PASCAL object detection challenges[2]:

$$a_0 = \frac{\text{area}(B_{det} \cap B_{gt})}{\text{area}(B_{det} \cup B_{gt})} \quad . \tag{6.1}$$

Multiple detections of the same object count as false positives. For example, the same object is detected three times in one image. Then one detection counts as true positive detection and the remaining two detections count as false positive.

[2] http://pascallin.ecs.soton.ac.uk/challenges/VOC/voc2009/devkit_doc_14-May-2009.pdf

6.1.2 Precision/Recall Curves

The detection results are plotted into Precision/Recall (PR) curves to determine the system's performance as proposed by Davis and Goadrich [21]. A PR curve is determined by plotting the Recall on one axis and the Precision or $1-$Precision on the other axis. Further, a classifier's predictions can be divided into four categories:

- *True Positives* (TP) are positive examples that are predicted as positives.
- *False Positives* (FP) are negative examples that are misleadingly labeled as positives.
- *True Negatives* (TN) refer to negative examples that are predicted as negatives.
- *False Negatives* (FN) correspond to positive examples that are predicted as negatives.

Based on these definitions the Recall is defined as the fraction of positive examples that are correctly predicted and the Precision measures the fraction of examples that are predicted to be positive and are truly positive:

$$Recall = \frac{TP}{TP + FN} \quad , \tag{6.2}$$

$$Precision = \frac{TP}{TP + FP} \quad . \tag{6.3}$$

This means that the performance of a classifier is high if both Recall and Precision are high.

6.1.3 F-Measure

Rijsbergen [98] introduced the F-Measure to weight the performance with respect to Precision and Recall. It computes the weighted harmonic mean between the Precision and the Recall. The general formula to compute the F-Measure is as follows:

$$F_\beta = \frac{(1 + \beta^2) \cdot (Precision \cdot Recall)}{(\beta^2 \cdot Precision + Recall)} \quad , \tag{6.4}$$

where β has to be a non-negative real value. This means that the F_β-Measure gives β times importance to the recall than to the precision. Within this thesis the traditional F-Measure is used as an additional performance measure. The traditional F-Measure is computed as follows:

$$F = \frac{2 \cdot Precision \cdot Recall}{Precision + Recall} \quad . \tag{6.5}$$

6.2 Training and Test Data

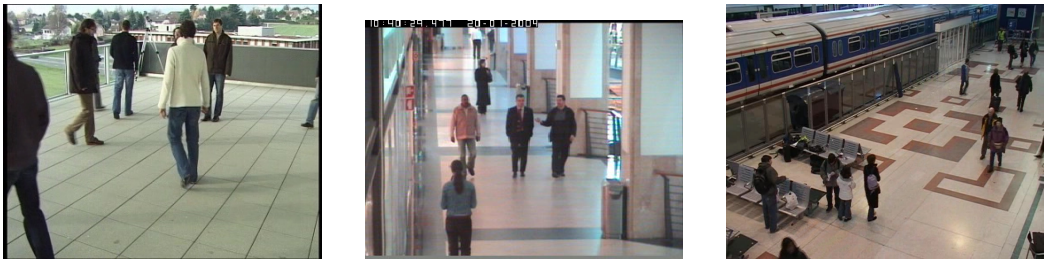
The required datasets for the experiments have to be divided into two categories: (i) training data and (ii) evaluation data. The difference between training and evaluation data is that the main training data have to be recorded with a multi-camera setup with synchronized cameras. This property is essential for the centralized learning approach. Further, single view datasets are required to train the initial classifiers and to additionally bootstrap negative examples during training. By contrast, there are no restrictions for the evaluation data, since there is no camera collaboration during evaluation. Figure 6.1 contains gaps from the training and evaluation image sequences. Examples from the INRIA Person Dataset are not shown, since this dataset is used for bootstrapping negative training examples only.

- **DT2 - Set4**[3]: This synchronized sequence shows an indoor scene with moving persons. It is observed by three static cameras. The video sequence contains 2589 frames. The frame size is 384×288 pixels.
- **BC06**[4]: Is a dataset consisting of four synchronized cameras observing an outdoor scene. This video sequence consists of 5000 frames with an image size of 360×288 pixels.
- **INRIA Person Dataset**[5]: This dataset contains a large amount of positive and negative examples [20]. Within this thesis this dataset is predominantly used to obtain additional negative examples via bootstrapping for classifier updates. The dataset contains 1832 training images with varying resolution whereas 1218 images do not contain positive examples and 614 images contain positives.
- **Caviar dataset**[6]: The evaluation image sequence was recorded in a shopping center showing various persons moving on a corridor. The image size equates to 384×288 pixels. The whole sequence contains 370 images.
- **PETS2006 dataset**[7]: This evaluation dataset observes an indoor public place with various moving and stalling persons. This image sequence contains 307 images, whereas each image has the size of 720×576 pixels.

[3] Provided by the Institute for Computer Graphics and Vision, University of Technology Graz
[4] Provided by the Computer Vision Laboratory, École Polytechnique Fédéral de Lausanne
[5] `http://pascal.inrialpes.fr/data/human/`
[6] `http://homepages.inf.ed.ac.uk/rbf/CAVIARDATA1/`
[7] `http://pets2006.net/data.html`

(a) DT2 Set4 - Camera 1 **(b)** DT2 Set4 - Camera 2 **(c)** DT2 Set4 - Camera 3

(d) BC06 - Camera 1 **(e)** BC06 - Camera 2 **(f)** BC06 - Camera 3



(g) BC06 - Camera 4 **(h)** Caviar **(i)** PETS2006

Figure 6.1: Gaps of training and evaluation datasets. (a), (b) and (c) show the same frame from the DT2 Set4 dataset for all three cameras. (d), (e), (f) and (g) show one frame from the BC05 dataset for all four cameras. (h) shows one frame from the Caviar dataset and (i) shows one frame from the PETS2006 dataset.

6.3 System Setup Description

All experiments in conjunction with the centralized approach are performed with a slightly varying system configuration. The on-line GradientBoost algorithm, which was discussed in Section 3.4, is used as the basic learning algorithm at each experiment. This GradientBoost uses a logistic loss function to determine the example weights. The principle detection process is performed as described in Section 5.2. According to this the classifier evaluates features within a detection window. The corresponding number of features is defined by the number of used selectors (see Section 3.3). The size of the overall feature pool that is available for the classifier is defined by the number of weak learners per selector, since one

feature is assigned to one weak learner. All selectable image feature types are discussed in Chapter 2. To handle the nD feature response vectors of HOG- and LBP-features the Euclidean distance to nD reference vectors is computed. The reference vectors are determined in context of the classifiers' initialization by computing a mean response vector of 20 randomly selected training examples. Incremental on-line histograms are deployed as weak learners (see Section 3.1.1). The centralized approach for learning requires a camera setup with at least two stationary cameras (see Section 5.5). Thus, either the *DT2 - Set4* dataset or the *BC06* dataset are required to perform the centralized approach. To enable the use of geometry these scenes have to be calibrated. Therefore, Tsai's camera calibration algorithm is deployed (see Section 4.3). All initial classifiers are trained with 20 positive and 20 negative examples that are randomly selected from the *INRIA Person Dataset*. The initial classifiers are cloned and assigned to single cameras in the multi-camera setup (see Section 5.4). The training process is illustrated in Figure 6.2. Finally, the classifiers are either evaluated on the sequence they are trained on or they are evaluated on the *PETS2006* and *Caviar* dataset respectively. Notice, that there is no collaboration between the camera views during evaluation!

6.4 Classifier Improvement Over Time

This experiment demonstrates the progress of the classifier performance through on-line learning using the centralized approach. The setup for this experiment consists of three calibrated cameras that observe an indoor scene whereas people walk on the floor (*DT2 - Set4* dataset). All frames are processed to gain positive and negative updates, whereas each tenth frame is taken to finally evaluate each camera's classifier. The initial classifier is trained as described in Section 6.3. In detail, the initial classifier consists of 50 selectors and 50 weak learners per selector. Each weak learner is assigned to a randomly generated Haar-like-wavelet. An on-line GradientBoost algorithm with a logistic loss-function is employed to learn the examples (see Section 3.4). The positive examples are obtained using the centralized approach which is introduced in Section 5.6. The negative examples are extracted by the usage of an approximated median background model (see Section 5.7.2). Additionally, negative examples are bootstrapped from an the *INRIA Person Dataset*. The classifier is saved at fixed times during training. Then these stored classifiers are evaluated on the whole training set separately to measure its Precision and Recall. Finally, these measures are assigned to the corresponding saving times and plotted into Recall/time and Precision/time curves respectively (see Figure 6.3).

Figure 6.3 clearly shows that the performance of the final classifiers is almost reached after 500 training frames. The more important property is, that the classifier's Recall and

(a) Camera 1, $t = 0$ **(b)** Camera 2, $t = 0$ **(c)** Camera 3, $t = 0$

(d) Camera 1, $t = 250$ **(e)** Camera 2, $t = 250$ **(f)** Camera 3, $t = 250$

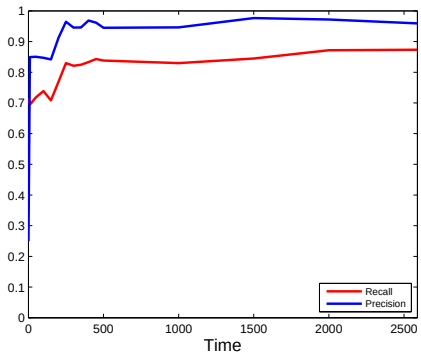
(g) Camera 1, $t = 500$ **(h)** Camera 2, $t = 500$ **(i)** Camera 3, $t = 500$

(j) Camera 1, $t = 750$ **(k)** Camera 2, $t = 750$ **(l)** Camera 3, $t = 750$

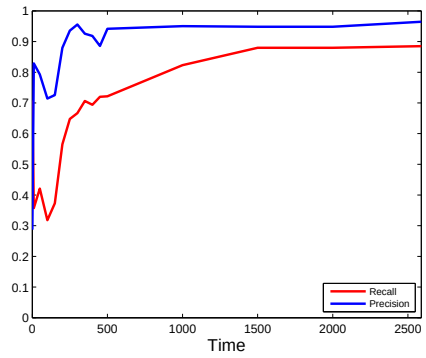
(m) Camera 1, $t = 1500$ **(n)** Camera 2, $t = 1500$ **(o)** Camera 3, $t = 1500$

(p) Camera 1, $t = 2000$ **(q)** Camera 2, $t = 2000$ **(r)** Camera 3, $t = 2000$

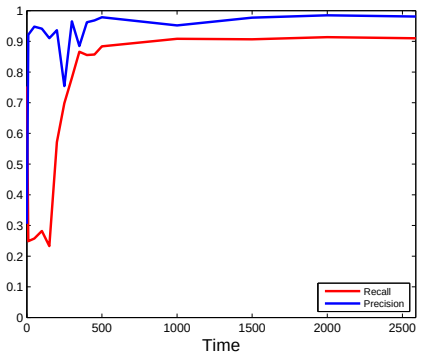
Figure 6.2: Training process using three cameras on the *DT2 - Set4* dataset. White bounding boxes are detections in the single camera views. Red bounding boxes correspond to negative updates that are obtained with a background model and green bounding boxes correspond to positive updates that are obtained using the centralized approach. Grey bounding boxes indicate suppressed updates.



(a) First camera



(b) Second camera



(c) Third camera

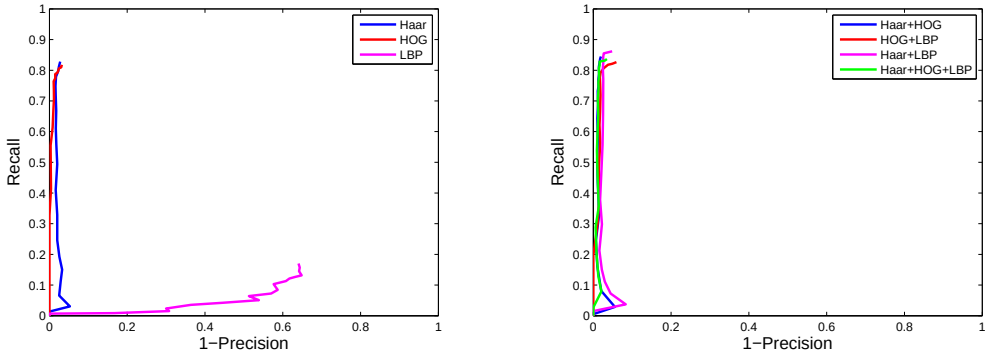
Figure 6.3: Illustration of the classifier performance at fixed training times for each camera view that is evaluated on the whole training set.

its Precision keep stable over time after 500 training frames. This reasons that the positive updates do not loose quality after some time. Further, it can be seen that the Recall increases dramatically after about 100 processed training frames. This results from the fact, that positive updates are extracted and processed starting from this time. In other words, no persons are visible beforehand within this dataset. Moreover, an extreme drop of the recall is recorded in the second and the third camera view within the first few frames. As mentioned above, there are no persons visible within the first few frames of the training scene. Additionally, negative updates are obtained using motion information, which is computed by means of a background model. Thus, negative updates are performed on the classifiers without performing positive updates, since there are no persons visible yet (false positives). According to these negative updates the Recall drops drastically, but stabilizes after a few positive updates.

6.5 Feature Performance Evaluation

In Chapter 2 several low-level image representations are discussed. This experiment compares the performance of the implemented feature types. All classifiers are trained using the centralized approach. Further, one initial classifier is trained per feature type using 20 positive and 20 negative training examples that are randomly selected from a much larger dataset. Then the classifiers are on-line trained using the centralized approach with the *DT2 - Set4* dataset. Each classifier consists of 50 selectors and 10 weak learners per selector. Details on the basic setup are depicted in Section 6.3. Different feature types are boosted together to a final classifier. In the case of using multiple feature types the classifiers are initialized with an even number of features per feature type.

Figure 6.4 and Table 6.1 show that Haar-like-wavelets and HOG-features have almost the same performance on the indoor scene. By contrast, using only LBP-features performs poorly, since these features may be too descriptive to generate high-quality updates in the first place. Moreover, the PR-curves show that classifiers which combine multiple feature types have a higher performance. Especially the combination of Haar-like-wavelets with LBP-features reach the highest Recall within this scenario, but it has even a lower precision than Haar-like-wavelets alone. On the other hand, a combination of Haar-like-wavelets and HOG-features have a rather high Recall and the highest Precision. Table 6.2 gives a more detailed distribution of the feature types that the classifiers deployed within this scenario. It is shown that usage of all feature types is warranted. Additionally, the performance of the different initial classifiers is evaluated on the *Caviar* dataset. The initial classifiers are taken to check their performance on external data without performing on-line updates. This evaluation clearly shows that combining multiple feature types increases the performance for some combinations. The corresponding PR-curves are depicted in Figure 6.5 and Table 6.3.



(a) Comparing Haar-, HOG- and LBP-features. **(b)** Comparing combinations of different feature-types.

Figure 6.4: Performance of the final classifiers using different low-level image representations.

	Recall	Precision	F-Measure
Haar	0.83	0.97	0.89
HOG	0.82	0.97	0.89
LBP	0.17	0.36	0.23

(a) Comparing Haar-, HOG- and LBP-features.

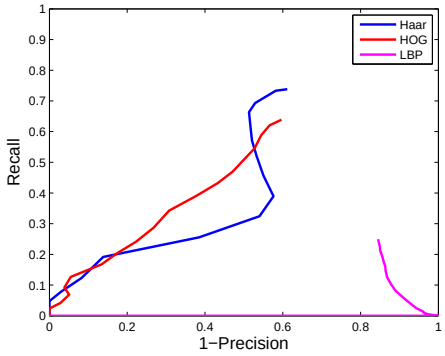
	Recall	Precision	F-Measure
Haar+HOG	0.84	0.98	0.91
HOG+LBP	0.83	0.94	0.88
Haar+LBP	0.86	0.95	0.90
Haar+HOG+LBP	0.84	0.96	0.90

(b) Comparing boosted feature-types.

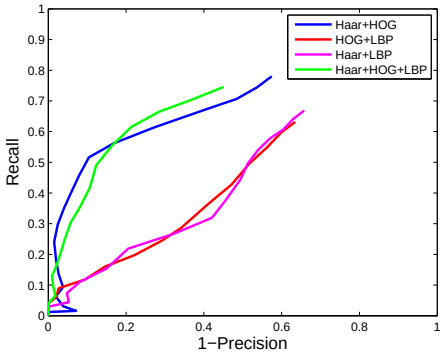
Table 6.1: Performance of the final classifiers using different low-level image representations.

	No. Features	Haar [%]	HOG [%]	LBP [%]
Haar+HOG	50	42.0	58.0	0.0
HOG+LBP	50	0.0	76.0	24.0
Haar+LBP	50	52.0	0.0	48.0
Haar+HOG+LBP	50	26.0	54.0	20.0

Table 6.2: Distribution of feature types at the classifiers that combine multiple feature types.



(a) Comparing Haar-, HOG- and LBP-features.



(b) Comparing boosted feature-types.

Figure 6.5: Performance of the initial classifiers that are evaluated on the *Caviar* dataset.

	Recall	Precision	F-Measure
Haar	0.74	0.39	0.51
HOG	0.64	0.40	0.49
LBP	0.25	0.16	0.19

(a) Comparing Haar-, HOG- and LBP-features.

	Recall	Precision	F-Measure
Haar+HOG	0.78	0.43	0.55
HOG+LBP	0.63	0.37	0.46
Haar+LBP	0.67	0.34	0.45
Haar+HOG+LBP	0.75	0.55	0.63

(b) Comparing boosted feature-types.

Table 6.3: Performance of the initial classifiers on the *Caviar* dataset.

6.6 Weak Learner Performance Evaluation

In Section 3.1.1 two different weak learners are discussed. The first weak learner computes an incremental histogram to approximate a probabilistic distributions of the positive and the negative object class, whereas the second weak learner is a nearest neighbor classifier that estimates the cluster centers of the positive and the negative object class in an on-line manner. This classifier computes the distance to each cluster center within the feature space for new examples and assigns the new example to the class with the lower distance to. By contrast, the on-line histogram compares the bins of the positive an the negative distribution to decide whether a new example contains an object or not. This experiment

compares these two weak learners at the use of multidimensional feature response vectors. The performance of HOG features is evaluated, where each HOG feature return feature vectors with 36 dimensions. The on-line nearest neighbor classifier can directly handle this multidimensional feature vectors, whereas on-line histograms have to compute the distance to a reference vector for each feature to deliver competitive results. Section 6.3 introduces further details on handling multidimensional feature response vectors at the use of on-line histograms.

The performance of these two weak learners is evaluated on the *DT2 - Set4* dataset. The on-line histograms use 16 bins to approximate the positive and the negative object class distributions. The initial classifier for this experiment uses 50 selectors and 50 weak learners per selector. The initial classifier is again trained with 20 positive and 20 negative examples only. Finally, Figure 6.6 and Table 6.4 depict the performance of both final classifiers. These curves clearly show, that the on-line histogram delivers much better results overall at the use of HOG features. As a result, the on-line histograms are deployed as weak classifiers for all experiments within this thesis.

	Recall	Precision	F-Measure
Histograms	0.84	0.93	0.89
Nearest Neighbor	0.86	0.91	0.89

(a) Camera 1

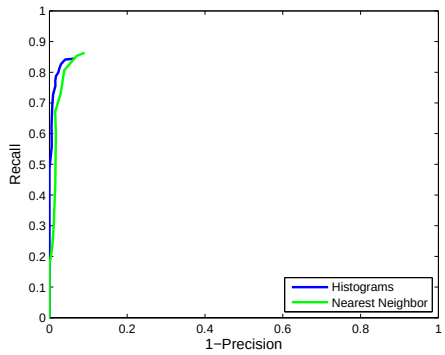
	Recall	Precision	F-Measure
Histograms	0.84	0.96	0.90
Nearest Neighbor	0.67	0.94	0.78

(b) Camera 2

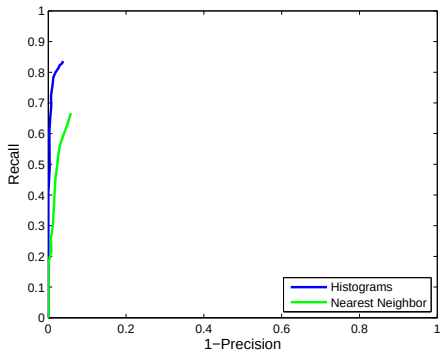
	Recall	Precision	F-Measure
Histograms	0.87	0.92	0.89
Nearest Neighbor	0.82	0.87	0.84

(c) Camera 3

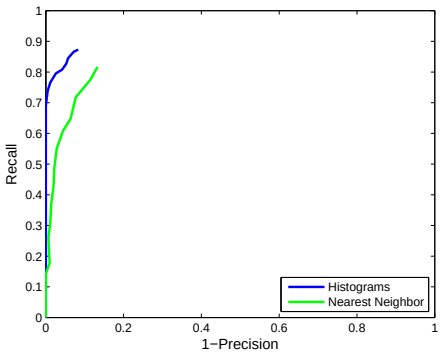
Table 6.4: Performance evaluation of on-line histograms and on-line nearest neighbor classifiers as weak learners.



(a) Camera 1



(b) Camera 2



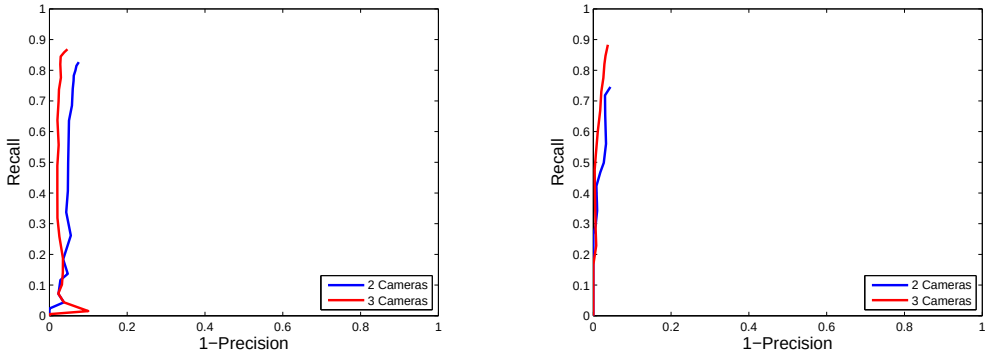
(c) Camera 3

Figure 6.6: Performance evaluation of on-line histograms and on-line nearest neighbor classifiers as weak learners.

6.7 Increasing Number of Camera Views

This experiment compares classifiers that are trained with two views and three views respectively. The basic setup is set as depicted in Section 6.3. The initial classifier is trained using 20 positive and 20 negative example and it consists of 50 selectors and 50 weak learners per selector. For this experiment Haar-like-wavelets are used only, since they perform well on the *DT2 - Set4* dataset if the centralized approach is deployed to on-line update the classifiers. Finally, the classifiers are evaluated on a subset of the training sequence. For this purpose, each tenth frame is evaluate.

Figure 6.7 and Table 6.5 depict the performance of the final classifiers that are trained either with two cameras or with three cameras. They clearly show, that the performance increases if more camera views are deployed for training. This fact is feasible, since higher performance can be obtained if more information is available to update the classifiers.



(a) Camera 1 **(b)** Camera 2

Figure 6.7: Performance of the final classifiers using 2 cameras and 3 cameras during training on the *DT2 - Set4* dataset respectively.

	Recall	Precision	F-Measure
2 Cameras	0.83	0.92	0.87
3 Cameras	0.87	0.95	0.91

(a) Camera 1

	Recall	Precision	F-Measure
2 Cameras	0.75	0.96	0.84
3 Cameras	0.88	0.96	0.92

(b) Camera 2

Table 6.5: Performance of the final classifiers using 2 cameras and 3 cameras during training on the *DT2 - Set4* dataset respectively.

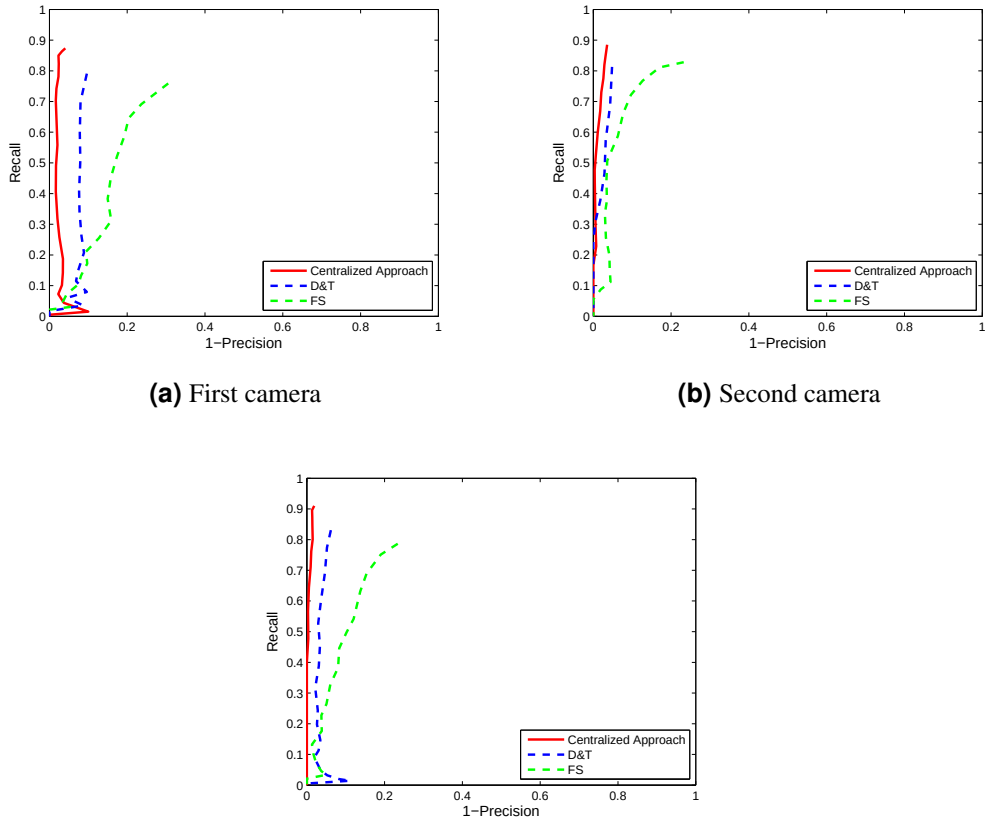
6.8 Scene Specific Classifiers

Scene specific classifiers are a byproduct if positive updates are gained from an arbitrary training scene. Hence, the classifiers are immediately updated with positive and negative examples as they are obtained. This experiment shows the detection results of the final classifiers that were obtained within the experiment, that shows the classifiers' performance improvement over time (see Section 6.4). Additionally, this experiment shows the performance of two static state-of-the-art person detectors on the same dataset (*DT2 - Set4*). The first state-of-the-art person detector within this experiment was proposed by Dalal and Triggs [20] (D&T), who used the HOG-descriptor with Support Vector Machines. And the second state-of-the-art person detector was proposed by Felzenszwalb *et al.* [30] (FS), who used a deformable part based model to identify persons within images. Each state-of-the-art detector is applied to the single camera views and compared with the corresponding on-line

trained classifiers that are obtained using the centralized approach. Remember, that there is no collaboration between the cameras during evaluation.

The results that are depicted in Figure 6.8 and Table 6.6. They show that the scene specific classifiers outperform the static state-of-the-art classifiers as expected. Remember that the scene specific initial classifier was trained with 20 positive and 20 negative examples only, whereas the static classifiers have been trained with $10,000$s of positive and negative training examples. As a result, the centralized approach establishes opportunities to train scene specific classifiers with a modicum of effort.

A further classifier is trained on the *BC06* dataset. This classifier has the same specifications as the classifier that was trained on the *DT2 - Set4* dataset. Finally, the classifier is evaluated on a subset of the training sequence. The evaluation sequence contains 250 frames, which corresponds to each twentieth frame of the training sequence. Figure 6.9 Table 6.7 shows the performance for both cameras.



(a) First camera

(b) Second camera

(c) Third camera

Figure 6.8: Scene specific classifiers' performance using *DT2 - Set4* dataset

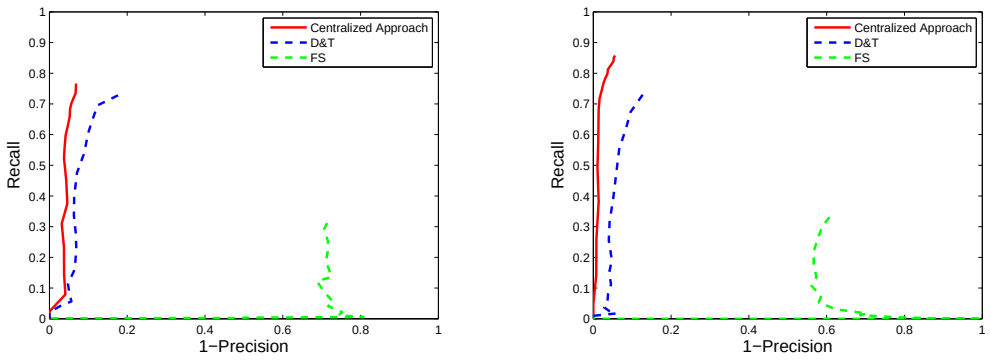
	Recall	Precision	F-Measure
Camera 1	0.72	0.25	0.37
Camera 2	0.71	0.29	0.41
Camera 3	0.75	0.31	0.43

(a) initial classifiers

	Recall	Precision	F-Measure
Camera 1	0.87	0.96	0.91
Camera 2	0.88	0.96	0.92
Camera 3	0.91	0.98	0.94

(b) final classifiers

Table 6.6: Performance measures for all three cameras that are trained with the proposed approach on the *DT2 - Set4* dataset.



(a) First camera **(b)** Second camera

Figure 6.9: Scene specific classifiers using *BC06* data.

	Recall	Precision	F-Measure
Camera 1	0.76	0.39	0.51
Camera 2	0.81	0.64	0.71

(a) initial classifiers

	Recall	Precision	F-Measure
Camera 1	0.77	0.93	0.84
Camera 2	0.86	0.94	0.90

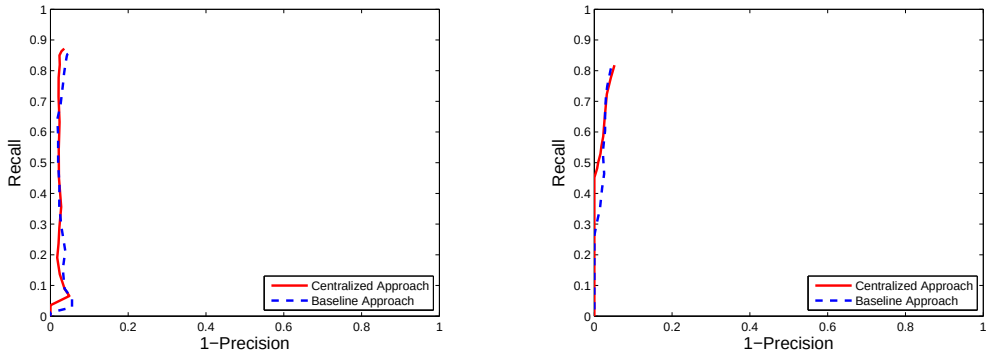
(b) final classifiers

Table 6.7: Performance measures for all three cameras that are trained with the proposed approach on the *BC06* dataset.

6.9 Centralized Approach versus Baseline Approach

This experiment compares the baseline approach (see Section 4.7) and the centralized approach which is proposed within this thesis. Both approaches are applied on the *DT2 - Set4* dataset using two cameras only. The initial classifiers are trained with 20 positive and 20 negative examples. Section 6.3 introduces additional details on the basic setup. The classifiers consist of 50 selectors whereas each selector contains 50 weak learners.

Figure 6.10 and Table 6.8 depict the performance of the final classifiers at each camera view. Hence, the PR-curves clearly show, that the baseline approach and the centralized approach deliver almost the same performance.



(a) Camera 1 **(b)** Camera 2

Figure 6.10: Performance of the final classifiers using either the baseline or the centralized approach with the *DT2 - Set4* dataset.

	Recall	Precision	F-Measure
Centralized Approach	0.87	0.96	0.92
Baseline Approach	0.85	0.95	0.90

(a) Camera 1

	Recall	Precision	F-Measure
Centralized Approach	0.82	0.95	0.88
Baseline Approach	0.81	0.96	0.88

(b) Camera 2

Table 6.8: Performance of the final classifiers using either the baseline or the centralized approach with the *DT2 - Set4* dataset.

6.10 Classifier Generalization

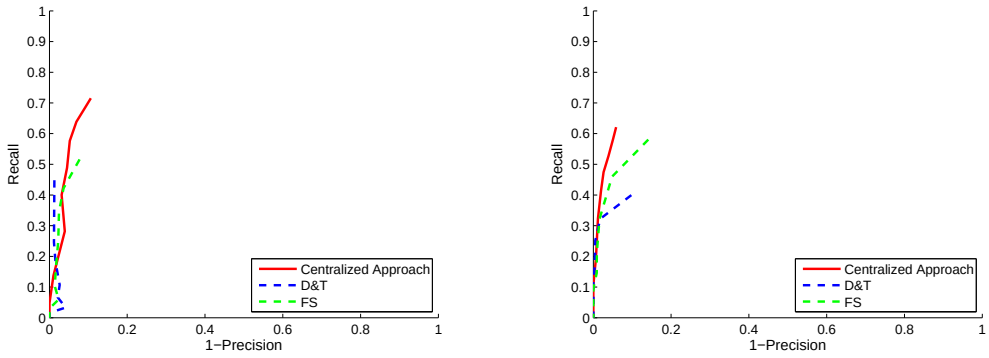
Section 6.8 shows that the centralized approach produces excellent scene specific classifiers. However, it is the goal the produce classifiers that have a high performance on a large variety of scenes. Therefore, this experiment evaluates the classifier performance on a much more general setup to proof its generalization. To perform this task, a much larger classifier is trained. Hence, cascades are deployed to speed-up the classifier's evaluation time. The cascade configuration is depicted in Table 6.9, since the cascade has to be parametrized in advance (see Section 5.8). Further, Haar-like-features and HOG-features are combined

	Stage 1	Stage 2	Stage 3	Stage 4
# Selectors	10	20	40	80
# Weak Learners per Selector	100	100	100	100

Table 6.9: Cascade configuration for generalizing classifier.

using the on-line Gradientboost algorithm with a logistic loss-function (see Section 3.4). The initial classifier contains 50% Haar-like-features and 50% HOG-features. Moreover, 20 positive and 20 negative examples are picked at random from the *INRIA Person Dataset* to train the initial classifier. This classifier is retrained by deploying the centralized approach on the *DT2 - Set4* dataset using three cameras. After training, the final classifier of one camera is evaluated on the *PETS2006* and the *Caviar* dataset. The D&T detector and the FS detector are evaluated on the same datasets to obtain reference results by using state-of-the-art person detectors.

Figure 6.11 shows the classifiers' PR-curves and Table 6.10 depicts the corresponding performance values. In both cases the centralized learning approach achieves excellent results compared to the static state-of-the-art detectors. Especially the Recall is much better on both datasets. This shows that the centralized approach is able to produce general classifiers that work on scenes with similar geometric conditions concerning the objects that have to be detected.



(a) Caviar Data **(b)** PETS2006 Data

Figure 6.11: Classifier performance on *PETS2006* and *Caviar* data.

	Recall	Precision	F-Measure
Centralized Approach	0.72	0.89	0.79
Dalal and Triggs	0.45	0.99	0.62
Felzenszwalb	0.52	0.92	0.66

(a) Caviar Data

	Recall	Precision	F-Measure
Centralized Approach	0.62	0.94	0.75
Dalal and Triggs	0.40	0.90	0.55
Felzenszwalb	0.58	0.86	0.69

(b) PETS2006 Data

Table 6.10: Classifier performance on *PETS2006* and *Caviar* data.

6.11 Summary

This chapter illustrates all experiments and the corresponding results which show the learning behavior of the centralized approach. The approach is applied to various datasets and it is shown that this approach performs stable learning which means that the classifier performance improves and finally saturates at a certain degree. Further, the centralized approach is compared to the baseline approach and to several state-of-the-art detectors. Finally, it is shown that scene specific classifiers which are trained with the novel approach outperform the state-of-the art detectors and that they yield competitive results on various other datasets, since this classifiers also generalize.

Chapter 7

Summary and Conclusion

This thesis presents a novel centralized approach to learn single-view classifiers from multi-camera scenes. This approach combines information from several camera views exploiting geometric constraints to obtain positive examples from an unlabeled training scene. Further, a method is introduced, that allows to train an object detector from a limited number of labeled data only. Hence, the approach implicitly labels data during training the detector. Compared to the baseline approach, which was proposed by Roth *et al.* [83], the centralized approach is able to use an arbitrary number of cameras without increasing the computational complexity of merging information. Moreover, the detector that is trained using a multi-camera setup, can be simply applied to any single-camera setup, which is important especially at surveillance applications, since there is mostly a single-camera setup available. In other words, there are hardly overlapping fields of camera views due to reduce the costs of surveillance systems. Finally, the centralized approach reduces the complexity of setting up a multi-camera training scene, since it requires calibrated cameras only. This means that each camera has to be calibrated once, after it is placed within the 3D scene. In detail, the calibrated cameras are deployed to transfer information from each camera view to a centralized top view map. In contrast, the baseline approach requires numerous homographies that have to be estimated for each pair of cameras. In this case, the number of required homographies increases quadratically if the number of cameras increases linearly. Another advantage of the centralized approach is the property, that it is completely independent from the subjacent learning algorithm and the single-view classifiers, since they provide confidence measures for all detections.

The centralized approach requires single-view detections with corresponding confidence measures. This novel approach collects all single-view detection information and projects them into a common top view map using each camera's calibration information. In the next step, the information fusion algorithm creates a density map on the basis of

the top view map, whereas only detections are transferred to the density map if the detection occurred in at least two different camera views. Due to calibration inaccuracies neighboring detections from different camera views in the top view map are regarded as one, if the euclidean distance between these detections is smaller than a certain threshold. To finally extract positive examples from the training data, a mean-shift algorithm is applied on the density map to find all local maxima. These local maxima correspond to top view locations of positive examples within the training data. Furthermore, these locations are back-projected into the single camera views to extract positive examples. Additionally, occlusion checks are performed before the examples are extracted, since possible overlapping examples may cause suboptimal positive updates and thus may corrupt the classifiers. Hence, scene specific classifiers are a byproduct of gathering positive examples. To finally obtain balanced classifiers negative updates are performed using negative examples that are bootstrapped from any external dataset that does not contain positive examples and negative examples that are extracted from the training scene using an approximated median background model.

Within the experimental section an on-line GradientBoost algorithm using a logistic loss-function is deployed to obtain classifiers via feature selection. Additionally, various feature types and weak learner types are evaluated. The initial classifiers are trained using 20 positive and 20 negative examples from a much larger external dataset. These classifiers are then retrained in an on-line manner to improve their performance using the centralized approach. The experiments showed that the centralized approach delivers equivalent results to the baseline approach without increasing the computational complexity if the number of cameras increases. Further, the experiments show that using more cameras during training leads to better classification results during single-view evaluation, since more cameras provide more information to train the classifiers. It is also shown that the classifiers' performance saturates after some time and keeps stable. Additionally, this indicates that the quality of the positive updates remains at a high level. Further, it is shown that the scene specific classifiers outperform state-of-the-art detectors on the training scenes as it is expected even though only 20 positive and 20 negative examples are taken to train the initial classifier. Finally, it is shown that classifiers that are trained using the centralized approach also generalize well. Such classifiers obtained more than competitive results compared to the state-of-the-art detectors' performance on publicly available benchmark datasets.

Future work will focus on integrating a part based detector to better handle partly occlusions in the single camera views. The current detector only detects objects if they are almost completely visible in single camera views to guarantee reliable detections. Further, more accurate weak learners have to be developed, that fit into the on-line boosting framework, to additionally increase the detection performance especially at the use of features that deliver

heigh-dimensional response vectors like HOG features or LBP features. Moreover, it is the goal to get rid of the background model (e.g. integrate a conservative learning strategy [80]). Currently the background model delivers negative examples from the multi-camera scene during training by evaluating motion information at detections. A negative update is triggered immediately if a detection occurs and there is no motion information present within the detection rectangle. Within this thesis the background model works fine, but it has several disadvantages. Assume an outdoor training scenario with multiple cameras, where a person is walking along a wall. The person causes a shadow on the wall and at least person detectors from two different cameras detect the person itself and the shadow during training. In this case positive updates are performed using both detections and the background model does not trigger a negative update on the shadow, since there is also motion information present.

Appendix A

Publications

For the sake of completeness the remainder of this chapter lists all publications that were published during the course of this work.

1. A. Berger, P.M. Roth, C. Leistner, H. Bischof, *Centralized Information Fusion for Learning Object Detectors in Multi-Camera Networks*, Austrian Association for Pattern Recognition, 2010 (submitted) [5]

Bibliography

[1] T. Ahonen, A. Hadid, and M. Pietikainen. Face recognition with local binary patterns. In *Proc. European Conference on Computer Vision*, pages 469–481, 2004.

[2] M.-F. Balcan, A. Blum, and K. Yang. Co-training and expansion: Towards bridging theory and practice. In *Advances in Neural Information Processing Systems*, pages 89–96, 2005.

[3] M. S. Bartlett. A comparison of gabor filter methods for automatic detection of facial landmarks. In *Proc. International Conference on Automatic Face and Gesture Recognition*, pages 242–246, 2002.

[4] J. Berclaz, F. Fleuret, and P. Fua. Principled detection-by-classification from multiple views. In *Proc. International Conference on Computer Vision Theory and Applications*, pages 375–382, 2008.

[5] A. Berger, P. M. Roth, C. Leistner, and H. Bischof. Centralized information fusion for learning object detectors in multi-camera networks. In *Proc. Workshop of the Austrian Association for Pattern Recognition*, 2010. submitted.

[6] C. M. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2008.

[7] J. Black, T. Ellis, and P. Rosin. Multi view image surveillance and tracking. In *Proc. of the Workshop on Motion and Video Computing*, 2002.

[8] A. Blum and T. Mitchell. Combining labeled and unlabeled data with co-training. In *Proc. Conference on Computational Learning Theory*, pages 92–100, 1998.

[9] I. Borg and P. Groenen. *Modern Multidimensional Scaling: Theory and Applications*. Springer, 1996.

[10] U. Brefeld, T. Gärtner, T. Scheffer, and S. Wrobel. Efficient co-regularised least squares regression. In *Proc. International Conference on Machine Learning*, pages 137–144, 2006.

[11] L. Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001.

[12] Y. Caspi, D. Simakov, and M. Irani. Feature-based sequence-to-sequence matching. *International Journal of Computer Vision*, 68(1):53–64, 2006.

[13] O. Chapelle, B. Schölkopf, and A. Zien, editors. *Semi-Supervised Learning (Adaptive Computation and Machine Learning)*. The MIT Press, 2006.

[14] S.-C. Cheung and C. Kamath. Robust techniques for background subtraction in urban traffic video. In *Proc. on Visual Communications and Image Processing*, pages 881–892. SPIE Electronic Imaging, 2004.

[15] P. Comon. Independent component analysis, a new concept? *Signal Processing*, 36(3):287–314, 1994.

[16] T. M. Cover and P. E. Hart. Nearest neighbor pattern classification. In *IEEE Transactions on Information Theory*, volume 13, pages 21–27, 1967.

[17] F. C. Crow. Summed-area tables for texture mapping. In *Proc. Special Interest Group on Graphics and Interactive Techniques*, pages 207–212, 1984.

[18] R. Cucchiara, M. Piccardi, and A. Prati. Detecting moving objects, ghosts, and shadows in video streams. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(10):1337–1342, 2003.

[19] R. Cutler and L. Davis. View-based detection and analysis of periodic motion. In *Proc. International Conference on Pattern Recognition*, volume 1, page 495, 1998.

[20] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, volume 1, pages 886–893, 2005.

[21] J. Davis and M. Goadrich. The relationship between precision-recall and ROC curves. In *Proc. International Conference on Machine Learning*, pages 233–240, 2006.

[22] V. R. de Sa. Learning classification with unlabeled data. In *Advances in Neural Information Processing Systems*, pages 112–119, 1993.

[23] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society, Series B*, 39(1):1–38, 1977.

[24] P. Dollár, C. Wojek, B. Schiele, and P. Perona. Pedestrian detection: A benchmark. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, pages 304–311, 2009.

[25] C. Domingo and O. Watanabe. Madaboost: A modification of adaboost. In *Proc. Conference on Computational Learning Theory*, pages 180–189, 2000.

[26] R. Duda, P. Hart, and D. Stork. *Pattern Classification*. John Wiley & Sons Inc., 2001.

[27] B. Efron. Bootstrap methods: Another look at the jackknife. *Annals of Statistics*, 14(1):1–26, 1979.

[28] A. M. Elgammal, D. Harwood, and L. S. Davis. Non-parametric model for background subtraction. In *Proc. European Conference on Computer Vision*, pages 751–767, 2000.

[29] R. Eshel and Y. Moses. Homography based multiple camera detection and tracking of people in a dense crowd. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8, 2008.

[30] P. Felzenszwalb, D. McAllester, and D. Ramanan. A discriminatively trained, multiscale, deformable part model. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, 2008.

[31] F. Fleuret, J. Berclaz, R. Lengagne, and P. Fua. Multi-camera people tracking with a probabilistic occupancy map. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30(2):267–282, 2008.

[32] J.-S. Franco and E. Boyer. Fusion of multi-view silhouette cues using a space occupancy grid. In *Proc. IEEE International Conference on Computer Vision*, pages 1747–1753, 2005.

[33] Y. Freund. Boosting a weak learning algorithm by majority. In *Proc. Conference on Computational Learning Theory*, pages 202–216, 1990.

[34] Y. Freund. An adaptive version of the boost by majority algorithm. In *Proc. Conference on Computational Learning Theory*, pages 102–113, 2000.

[35] Y. Freund and R. E. Schapire. Experiments with a new boosting algorithm. In *Machine Learning*, pages 148–156, 1996.

[36] J. Friedman, T. Hastie, and R. Tibshirani. Additive logistic regression: a statistical view of boosting. *Annals of Statistics*, 28(2):337–407, 2000.

[37] J. H. Friedman. Greedy function approximation: A gradient boosting machine. *Annals of Statistics*, 29:1189–1232, 2001.

[38] N. Friedman and S. Russell. Image segmentation in video sequences: A probabilistic approach. In *Proc. of the Annual Conference on Uncertainty in Artificial Intelligence*, pages 175–181, 1997.

[39] J. Goldberger, S. Roweis, G. Hinton, and R. Salakhutdinov. Neighbourhood components analysis. In *Advances in Neural Information Processing Systems*, pages 513–520, 2005.

[40] H. Grabner and H. Bischof. On-line boosting and vision. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, volume 1, pages 260–267, 2006.

[41] H. Grabner, P. M. Roth, M. Grabner, and H. Bischof. Autonomous learning of a robust background model for change detection. In *Proc. IEEE International Workshop on Performance Evaluation of Tracking and Surveillance*, pages 39–46, 2006.

[42] I. Haritaoglu, D. Harwood, and L. S. David. W4: Real-time surveillance of people and their activities. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8):809–830, 2000.

[43] R. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, 2003.

[44] J. Heikkila and O. Silven. A four-step camera calibration procedure with implicit image correction. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, page 1106, 1997.

[45] J. Heikkilä and O. Silvén. A real-time system for monitoring of cyclists and pedestrians. In *Proc. IEEE Workshop on Visual Surveillance*, pages 74–81, 1999.

[46] O. Javed, S. Ali, and M. Shah. Online detection and classification of moving objects using progressively improving detectors. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, volume 1, pages 696–701, 2005.

[47] S. M. Khan and M. Shah. A multiview approach to tracking people in crowded scenes using a planar homography constraint. In *Proc. European Conference on Computer Vision*, volume 4, pages 133–146, 2006.

[48] S. M. Khan and M. Shah. Tracking multiple occluding people by localizing on multiple scene planes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 31(3):505–519, 2009.

[49] H. Kim, E. Murphy-Chutorian, and J. Triesch. Semi-autonomous learning of objects. In *Proc. IEEE Workshop on Vision for Human-Computer Interaction*, 2006.

[50] D. Koller, J. Weber, and J. Malik. Robust multiple car tracking with occlusion reasoning. In *Proc. European Conference on Computer Vision*, pages 189–196, 1994.

[51] R. Kumar and A. R. Hanson. Robust methods for estimating pose and a sensitivity analysis. *Computer Vison, Graphics, and Image Processing: Graphical Models and Image Processing*, 60(3):313–342, 1994.

[52] C. Leistner, P. M. Roth, H. Grabner, H. Bischof, A. Starzacher, and B. Rinner. Visual on-line learning in distributed camera networks. In *Proc. ACM/IEEE International Conference on Distributed Smart Cameras*, pages 1–10, 2008.

[53] C. Leistner, A. Saffari, P. M. Roth, and H. Bischof. On robustness of on-line boosting - a competitive study. In *Proc. IEEE Workshop on On-line Computer Vision (ICCV)*, 2009.

[54] A. Levin, P. Viola, and Y. Freund. Unsupervised improvement of visual detectors using co-training. In *Proc. IEEE International Conference on Computer Vision*, volume 1, pages 626–633, 2003.

[55] R. Lienhart and J. Maydt. An extended set of haar-like features for rapid object detection. In *Proc. IEEE International Conference on Image Processing*, pages 900–903, 2002.

[56] S. Lloyd. Least squares quantization in PCM. In *IEEE Transactions on Information Theory*, volume 28, pages 129–137, 1982.

[57] R. Maclin and D. Opitz. An empirical evaluation of bagging and boosting. In *Proc. of National Conference on Artificial Intelligence*, pages 546–551, 1997.

[58] H. Masnadi-Shirazi and N. Vasconcelos. On the design of loss functions for classification: theory, robustness to outliers, and savageboost. In *Advances in Neural Information Processing Systems*, pages 1049–1056, 2008.

[59] L. Mason, J. Baxter, P. Bartlett, and M. Frean. *Advances in Large Margin Classifiers*. MIT Press, 2000.

[60] N. McFarlane and C. P. Schofield. Segmentation and tracking of piglets in images. *Machine Vision and Applications*, 8:187–193, 1995.

[61] R. S. Michalski. Knowledge acquisition through conceptual clustering: A theoretical framework and an algorithm for partitioning data into conjunctive concepts. *International Journal of Policy Analysis and Information Systems*, 4:219–244, 1980.

[62] K. Mueller, A. Smolic, M. Droese, P. Voigt, and T. Wienand. Multi-texture modeling of 3d traffic scenes. In *Proc. International Conference on Multimedia and Expo*, pages 657–660, 2003.

[63] P. Negri, X. Clady, and L. Prevost. Benchmarking haar and histograms of oriented gradients features applied to vehicle detection. In *Proc. International Conference on Informatics in Control, Automation and Robotics*, pages 359–364, 2007.

[64] A. Neubeck and L. Van Gool. Efficient non-maximum suppression. In *Proc. International Conference on Pattern Recognition*, volume 3, pages 850–855, 2006.

[65] K. Nigam and R. Ghani. Analyzing the effectiveness and applicability of co-training. In *Proc. International Conference on Information and Knowledge Management*, pages 86–93, 2000.

[66] T. Ojala, M. Pietikäinen, and D. Harwood. A comparative study of texture measures with classification based on feature distributions. *Pattern Recognition*, 29(1):51–59, 1996.

[67] T. Ojala, M. Pietikäinen, and T. Maenpaa. Multiresolution gray-scale and rotation invariant texture classification with local binary patterns. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(7):971–987, 2002.

[68] N. M. Oliver, B. Rosario, and A. P. Pentland. A bayesian computer vision system for modeling human interactions. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8):831–843, 2000.

[69] M. Oren, C. Papageorgiou, P. Sinha, E. Osuna, and T. Poggio. Pedestrian detection using wavelet templates. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, pages 193–199, 1997.

[70] J. Orwell, S. Massey, P. Remagnino, D. Greenhill, and G. A. Jones. A multi-agent framework for visual surveillance. In *Proc. International Conference on Image Analysis and Processing*, 1999.

[71] G. Overett, L. Petersson, L. Andersson, and N. Pettersson. Boosting a heterogeneous pool of fast HOG features for pedestrian and sign detection. In *IEEE Intelligent Vehicles Symposium*, pages 584 –590, 2009.

[72] N. C. Oza. *Online Ensemble Learning*. PhD thesis, The University of California, Berkely, Faculty of Electrical Engineering and Computer Science, September 2001.

[73] N. C. Oza. Online bagging and boosting. In *Proc. IEEE International Conference on Systems, Man and Cybernetics*, volume 3, pages 2340–2345, 2005.

[74] N. C. Oza and S. Russell. Online bagging and boosting. In *Proc. Workshop on Artificial Intelligence and Statistics*, pages 105–112, 2001.

[75] M. Ozuysal, P. Fua, and V. Lepetit. Fast keypoint recognition in ten lines of code. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8, 2007.

[76] C. Papageorgiou, M. Oren, and T. Poggio. A general framework for object detection. In *Proc. IEEE International Conference on Computer Vision*, pages 555–562, 1998.

[77] F. Porikli. Integral histogram: A fast way to extract histograms in cartesian spaces integral histogram. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, volume 1, pages 829–836, 2005.

[78] B. Rasolzadeh, L. Petersson, and N. Pettersson. Response binning: Improved weak classifiers for boosting. In *IEEE Intelligent Vehicles Symposium*, pages 344–349, 2006.

[79] G. Rätsch, B. Schökopf, S. Mika, and K. R. Müller. SVM and boosting: One class. Technical Report 119, GMD FIRST, 2000.

[80] P. M. Roth and H. Bischof. *Machine Learning Techniques for Multimedia*, chapter Conservative Learning for Object Detectors. Springer, 2008.

[81] P. M. Roth, H. Grabner, C. Leistner, M. Winter, and H. Bischof. Interactive learning a person detector: Fewer clicks - less frustration. In *Proc. Workshop of the Austrian Association for Pattern Recognition*, 2008.

[82] P. M. Roth, H. Grabner, D. Skočaj, H. Bischof, and A. Leonardis. On-line conservative learning for person detection. In *Proc. IEEE International Workshop on Visual Surveillance and Performance Evaluation of Tracking and Surveillance*, pages 223–230, 2005.

[83] P. M. Roth, C. Leistner, H. Grabner, and H. Bischof. *Multi-Camera Networks, Principles and Applications*, chapter Online Learning of Person Detectors by Co-Training from Multiple Cameras, pages 313–334. Academic Press, 2009.

[84] R. Schapire. The strength of weak learnability. *Machine Learning*, 5(2):197–227, 1990.

[85] R. Schapire. The boosting approach to machine learning: An overview. In *Proc. MSRI Workshop on Nonlinear Estimation and Classification*, 2001.

[86] V. Sindhwani, P. Niyogi, M. Belkin, and S. Keerthi. A co-regularized approach to semi-supervised learning with multiple views. In *Proc. ICML Workshop on Learning with Multiple Views*, 2005.

[87] J. Sochman and J. Matas. Waldboost " learning for time constrained sequential detection. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, volume 2, pages 150–156, 2005.

[88] C. Stauffer and W. E. L. Grimson. Learning patterns of activity using real-time tracking. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8):747–757, 2000.

[89] K. Stauffer, A. Smolic, M. Droese, P. Voigt, and T. Wienand. Automated multi-camera planar tracking correspondence modeling. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, pages 259–266, 2003.

[90] R. E. Stepp. Concepts in conceptual clustering. In *Proc. International Joint Conference on Artificial Intelligence*, pages 211–213, 1987.

[91] K.-K. Sung and T. Poggio. Example-based learning for view-based human face detection. Technical Report 1521, Artificial Intelligence Laboratory, Massachusetts Institute of Technology, 1994.

[92] K.-K. Sung and T. Poggio. Example-based learning for view-based human face detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(1):39–51, 1998.

[93] K. Tieu and P. Viola. Boosting image retrieval. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, volume 1, pages 228–235, 2000.

[94] K. Toyama, J. Krumm, B. Brumitt, and B. Meyers. Wallflower: Principles and practice of background maintenance. In *Proc. IEEE International Conference on Computer Vision*, pages 255–261, 1999.

[95] R. Tsai. A versatile camera calibration technique for high-accuracy 3D machine vision metrology using off-the-shelf TV cameras and lenses. *IEEE Journal of Robotics and Automation*, 3:323–344, 1987.

[96] I. Ulusoy and C. M. Bishop. Generative versus discriminative methods for object recognition. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, volume 2, pages 258–265, 2005.

[97] R. Vaillant, C. Monrocq, and Y. Le Chun. Original approach for the localisation of objects in images. In *IEEE Proc. - Vision, Image and Signal Processing*, volume 141, pages 254–250, 1994.

[98] C. J. Van Rijsbergen. *Information Retrieval*. Butterworth-Heinemann, 1979.

[99] V. Vapnik and A. Chervonenkis. *Theory of Pattern Recognition*. Nauka, 1974. (German Translation: W. Wapnik & A. Tscherwonenkis, *Theorie der Zeichenerkennung*, Akademie–Verlag, Berlin, 1979).

[100] P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, volume 1, pages 511–518, 2001.

[101] I. Visentini, L. Snidaro, and G. L. Foresti. On-line boosted cascade for object detection. In *Proc. International Conference on Pattern Recognition*, pages 1–4, 2008.

[102] X. Wang, T. Han, and S. Yan. An HOG-LBP human detector with partial occlusion handling. In *Proc. IEEE International Conference on Computer Vision*, pages 32–39, 2009.

[103] C. Wojek and B. Schiele. A performance evaluation of single and multi-feature people detection. In *Proc. DAGM symposium on Pattern Recognition*, pages 82–91, 2008.

[104] C. Wren, A. Azarbayejani, T. Darrell, and A. Pentland. Pfinder: Real-time tracking of the human body. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(7):780–785, 1997.

[105] Z. Zhang. A flexible new technique for camera calibration. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(11):1330–1334, 2000.

[106] Z.-H. Zhou and M. Li. Tri-training: Exploiting unlabeled data using three classifiers. *IEEE Transactions on Knowledge and Data Engineering*, 17(11):1529–1541, 2005.

[107] Q. Zhu, S. Avidan, M. C. Yeh, and K. T. Cheng. Fast human detection using a cascade of histograms of oriented gradients. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, pages 1491–1498, 2006.

[108] X. Zhu. Semi-supervised learning literature survey. Technical Report 1530, Computer Sciences, University of Wisconsin-Madison, 2005.