FAKULTÄT FÜR !NFORMATIK

Face detection in historic documentaries with a cascaded classifier

DIPLOMARBEIT

zur Erlangung des akademischen Grades

Diplom-Ingenieur

im Rahmen des Studiums

Computergraphik/Digitale Bildverarbeitung

eingereicht von

Tobias Schleser
Matrikelnummer 0225349

an der
Fakultät für Informatik der Technischen Universität Wien

Betreuung:
Betreuer: Univ.-Prof. Dr. Christian Breiteneder
Mitwirkung: Dipl.-Ing. Matthias Zeppelzauer

Wien, 20.04.2009 _____ _____
(Unterschrift Verfasser) (Unterschrift Betreuer)

Erklärung zur Verfassung der Arbeit

Tobias Schleser
Alseggerstraße 38/9, 1180 Wien, Österreich

Hiermit erkläre ich, dass ich diese Arbeit selbstständig verfasst habe, dass ich die verwendeten Quellen und Hilfsmittel vollständig angegeben habe und dass ich die Stellen der Arbeit – einschließlich Tabellen, Karten und Abbildungen –, die anderen Werken oder dem Internet im Wortlaut oder dem Sinn nach entnommen sind, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe.

Wien, am 20. April 2009

Abstract

Face detection aims at detecting and localizing an unknown number of faces in a still image or video frame. The challenges are to detect all faces while keeping the false positive rate small and to minimize the detection time per frame.

We study face detection in the context of historic documentaries. The source material for this work are films of the Soviet film maker Dziga Vertov that date back to the 1920's. The digitally available material bears major image deficiencies including flicker, scratches, dirt, bad lighting and contrast and visible frame lines. Naturally, the material is monochromatic and silent.

Based on a literature survey on different approaches for face detection, we select a method introduced by Viola and Jones for this investigation. Their approach employs a cascaded classifier, i.e. a sequence of nodes, that distinguishes faces from non-faces. These nodes are organized as a hierarchy of classifiers that are built from simple, Haar-like features. The main advantage of using a cascade is that only a moderate false-positive rate is needed for individual nodes as the individual rates multiply up to the overall false-positive rate.

We describe how the detection framework is set up for and adapted to the historic material and how it is implemented. Additionally, we suggest several post-processing steps to ameliorate the false-positive rate. Finally, we provide detailed results for several sample scenes from the documentaries, and analyze the performance of the training and detection stages.

Deutsche Zusammenfassung

Bei der Gesichtserkennung wird versucht, eine unbekannte Anzahl an Gesichtern in einem Bild oder Video zu erkennen. Dabei geht es darum, möglichst alle Gesichter zu erkennen und gleichzeitig eine geringe Fehlerrate aufrechtzuerhalten. Weiters soll die Zeit für die Analyse eines Frames minimiert werden.

In dieser Arbeit wird Gesichtserkennung im Kontext historischer Dokumentarfilme untersucht. Das Material sind aus den 1920er Jahren stammende Filme des sowjetischen Regisseurs Dziga Vertov. Das (digital) verfügbare Material weist zum Teil schwere Mängel wie beispielsweise Flimmern, Kratzer, Schmutz, schlechte Beleuchtung und Kontrast auf. Außerdem handelt es sich um schwarz-weiß Filme ohne Ton.

Ausgehend von einer Literaturrecherche über die verschiedenen Ansätze zur Gesichtserkennung wurde eine Methode von Viola und Jones als Basis dieser Arbeit ausgewählt. Der Ansatz verwendet eine Kaskade von Klassifikatoren um Gesichter von nicht-Gesichtern zu unterscheiden. Die einzelnen Stufen dieser Kaskade bestehen aus einer Hierarchie von Klassifikatoren, welche aus einfachen, Haar-ähnlichen Features erstellt werden. Der Vorteil der Kaskade ist, dass für die einzelnen Stufen nur eine mittelmäßig gute Fehlerrate erreicht werden muss, da sich die individuellen Raten zur Gesamtfehlerrate aufmultiplizieren.

Diese Arbeit beschreibt, wie die Methode für die Gesichtserkennung aufgebaut ist, wie sie an das alte Material angepasst wird und wie die konkrete Implementierung aussieht. Außerdem werden einige Nachverarbeitungsschritte vorgeschlagen, welche die Erkennungs- und Fehlerrate verbessern. Es werden detailierte Ergebnisse für einige Beispielszenen aus den Dokumentationen präsentiert und der Rechenaufwand für Training und Erkennung analysiert.

Contents

1 Introduction

Face detection is one of the visual tasks that humans can do effortlessly. However, the goal of automatically detecting faces by computer systems is hard to achieve. Hjelmås defines the face detection problem as follows: *"Given a still or video image, detect and localize an unknown number (if any) of faces."* [11] Thus, we aim at developing a detector that is able to localize any number of faces in an image while producing no false detections (i.e. detected locations where no face is present).

The localization of faces is difficult because faces vary not only in location but also in scale, orientation and pose. Furthermore, faces are non-rigid, thus a face may change appearance over time. Additionally there may be varying lighting conditions and occlusions. The challenges in face detection are presented in more detail in Section 1.4.

The different approaches for automatic face detection rely for example on pattern recognition and statistics. Other methods are based on learning algorithms. These methods rely heavily on training from sample (face-)images. We discuss several already available face databases in Section 1.5. Up to now there exist no standardized data sets for detector evaluation. We introduce important evaluation metrics and databases in Section 1.6.

1.1 Organization

This thesis is organized as follows. Section 1 gives an introduction to face detection, its challenges and application. An overview of different approaches is presented in Section 2. Section 3 summarizes the novel domain of historic documentaries, along with the special challenges in this domain. The chosen face detection approach of Viola and Jones and adaptions of their method are presented in Section 4. Further, the experimental setup for training of the face detector and evaluation of sample scenes is described in Section 5. Results from the experiments are presented in Section 6 and we conclude in Section 7.

1.2 Motivation

Face detection has become an important task not only in the area of surveillance but also for image and video indexing and retrieval. The world wide web gives access to thousands of media files, but also private media collections get more and more difficult to index with increasing size. Thus, robust face detection, among other object detection methods, is preferrable to support a user's search.

We study face detection in the domain of historic documentaries as face detection is regarded to be an important basis for tools that aim at automated understanding and indexing of films and video material. Such tools might analyze dialogue sequences or aim at recognizing certain persons like politicians. Additionally, a detected face may be used as reference object for determining the shot size and camera geometry. Other technologies, that are built upon face detection, are presented in Section 1.3.

A major challenge for face detection in the context of historic documentaries is the monochromaticity of the film material. Although faces may still be detected, the missing color information is a major drawback as color is an important modality to detect faces. However, monochromatic sources are common, also in the area of visual surveillance. Monochromatic cameras are cheaper and night shots feature monochromatic light as well. Thus, a method that works robustly on monochromatic material is needed.

1.3 Applications

As mentioned before, traditionally computer vision has been applied in repetitive, well defined tasks such as assembly line inspection. However, recent developments lead to broader fields of applications. Decreasing costs for image acquisition hardware and computing power allow for the deployment of computer vision and image understanding software in large-scale visual systems as well as in personal tools on PDAs and pocket cameras. As faster hardware and algorithms for face detection become available, a broad range of applications, in the areas of *face recognition*, *face authentication*, *face tracking* and *expression recognition* is able to solve real-world problems. *Face detection* is the basis for all these applications.

Tasks range from surveillance to multimedia processing, human computer interaction and others. Figure 1 shows the diverse connections of related research fields in computer vision and summarizes their applications.

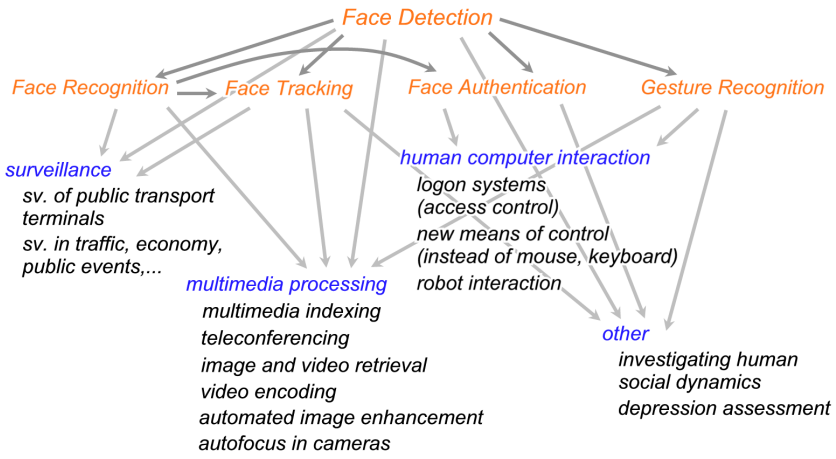

Figure 1: Applications of face detection. Various technologies (orange) are basis for fields of applications (blue) with specific tasks (black).

In the field of security and surveillance, Tsala et al. presented a system for face authentication in [45]. They use both, 3D-range data and color images to robustly detect and localize faces. They employ synthetically generated views to deal with difficult pose and varying lighting.

While surveillance in the field of public transport systems and monitoring of facilities is important, there also exist various applications in multimedia processing: the indexing and retrieval of video and image databases are examples. Other applications include teleconferencing and automated image enhancement [46]. Recently, face detectors have also been implemented in consumer electronics like digital pocket cameras for enhanced autofocussing [56].

Human computer interaction (HCI) is a new field of computer vision where reliable face recognition systems can be used for access control. Pnevmatikakis states that face recognition systems would rank among the most desirable systems because of two main properties: First, they are non-intrusive and second, the needed infrastructure is simple and cheap [32]. New means of computer interaction alternatively to using a mouse and keyboard

are also developed with regards to face detection [56]. Other applications include the investigation of human social dynamics and depression assessment, i.e. the observation and timing of facial expression, head motion and gaze that serve to identify nonverbal indicators of depression [18].

Some methods on face detection may be used for general object detection as well. Typical examples are the detection and classification of cars, text or cells in biomedical images [13].

1.4 Challenges

Human faces are non-rigid, meaning that they are subject to steady deformation. Although the face detection problem can be clearly formulated and the present task is a two-class decision, the nature of faces, their appearance in images and videos and certain properties of general object detection bear major challenges. The most important challenges are:

- *Facial expression.* Affective states or emotions (happy, sad, surprised, scared, etc.) have significant impact on the appearance of a human face.

- *Pose.* In images and videos, people appear in all different poses, ranging from frontal to profile view and anything in between. Unfortunately, the image of a face is completely different for a frontal and a profile face.

- *Human race.* People of different races feature not only different skin color. The shape of lips, eyes and nose differ significantly among people from different parts of the world (while a certain variability is present among individuals of a single race as well).

- *Occlusions and facial features.* Parts of faces may be occluded by both, facial features such as beards, mustaches or glasses (which themselves occur in a multitude of shapes, sizes and colors) and by other objects in the scene like other people, scarfs, headwear and so on. The degree of occlusion and the occluded parts of the face vary randomly.

- *Orientation, 3D position and camera distance.* Additionally to the pose of a face, it may be translated and rotated in three dimensional space with respect to the camera coordinate system.

- *Uncontrolled background.* Video and images generally feature uncontrolled, scattered background. A multitude of other objects may appear in the background (including other faces).

- *Illumination.* The brightness and color temperature of different light sources, as well as their number, influence the color appearance of facial parts. Additionally, the shape of objects (i.e. head, nose, mouth) can be distorted, for instance by sharp shadows or reflections.

A face detection system has to take these issues into account. Some of them can be met by broadening the training set of face images (appearance-based approaches, see Section 2.4) or by extending the set of rules, features or templates (see Sections 2.1 - 2.3). However, these extensions have direct impact on performance and usually not all challenges can be met satisfactorily. For example, strong in-plane and out-of-plane rotated faces can not be detected by traditional face detection algorithms that work on 2D data without training completely independent detectors for the different poses.

1.5 Face Databases

As there has been a strong development in the area of learning algorithms for face detection, the collection and supply of face databases, which constitute the source material for these algorithms, has boosted. While most of these databases are available online and are free for scientific use, they differ significantly in size and quality. Table 1 summarizes some of the high-quality datasets and their properties. Most of these data sets are suited for both, training face detectors (given an abundance of face images with varying lighting conditions and a lot of different individuals) and for the evaluation of face recognition systems (given several images per individual, for example with varying expression, lighting and pose). The *UMIST Database*, the *Yale Face Database B* and the *AR Face Database* are well known and offer a broad range of illumination conditions, facial expressions and poses. The

FERET Database and the *Essex Database* have been compiled specifically for the purpose of evaluating face recognition systems (but can be used for training purposes as well). Some datasets provide additional information like eye- or mouth coordinates (*Yale Face Database B*, *BioID Face Set* presented in Table 2). The *CMU PIE Dataset* is the largest face collection in this summary, providing 41,368 images of 68 individuals in an abundance of lighting conditions, with different poses and facial expressions. The set is not available online but the database may be ordered on several DVDs on the project website. An unofficial but very large training set with over 3,000 face images (though at the low resolution of 24×24 pixels) has been collected by Viola and Jones and has been used for the implementation in this work [48].

Table 1: Face databases for training of face detectors.

Database and Location	Description
FERET Database [29, 30] National Institute of Standards and Technology[1]	2,413 images of 865 individuals with different facial expressions; developed for evaluation of face recognition systems
UMIST Face Database [8] University of Manchester	564 images of 20 individuals covering a range of poses from profile to frontal view
Yale Face Database [47] Yale University	165 images of 15 individuals; images with different facial expressions and/or configuration (direction of light, glasses)
Yale Face Database B [7] Yale University	5,460 images of 10 individuals; 576 viewing conditions (9 poses x 64 illumination conditions); ground truth with coordinates of eyes and mouth available
AR Face Database [22] Purdue University	Over 4,000 images of 126 individuals; different facial expressions, illumination conditions, and occlusions (sun glasses and scarf); images taken in 2 sessions; configuration encoded in file name
Essex Database [42] Essex University	7,900 images of 395 individuals; varying expression and background, divided into 4 sets (increasing difficulty); developed for evaluation of face recognition systems
CMU PIE Database [37] Carnegie Mellon University[2]	41,368 images of 68 individuals, one face per image; 13 different poses, 43 different illumination conditions and 4 different expressions per person
Viola and Jones Database [48]	3,900 face images of indivduals at the resolution of 24×24 pixels; extracted from a random web crawl

1.6 Detector Evaluation

The evaluation of the accuracy of face detectors is an important but difficult task. Scientists use different metrics, for example the execution time and the ratio between detection rate and false-positive rate. These rates take the

[1] FERET Database not available for direct download
[2] CMU PIE Database not available for direct download

two types of errors into account that classifiers can generally make: *false-negatives* are positive instances (faces) that are not classified to be positive and *false-positives* are negative instances (non-face regions) that are declared to be positive (i.e. a false detection). Definitions for the detection- and false positive rate vary. In this work, the *detection rate* (also called recall) is the ratio of the number of faces classified correctly by the detector and the number of faces detected by a human:

$$detectionRate = \frac{detectedFaces}{totalNumberOfFaces} \tag{1}$$

The *false positive rate* (also called fallout) is the ratio of the number of false positives and the total number of image regions that are classified by the detector:

$$falsePositiveRate = \frac{falsePositives}{totalNumberOfImageRegions} \tag{2}$$

The rates can also be calculated in terms of type 1 and type 2 errors. The following confusion matrix summarizes the results that a two-class classifier can generally return. True positives (TP) and true negatives (TN) are good results, false positives (FP) and false negatives (FN) indicate errors of the classifier.

	correct: 1	**correct: 0**
classified: 1	true positive (TP)	false positive (FP)
classified: 0	false negative (FN)	true negative (TN)

By using this table the calculation of *recall* (r) and *fallout* (f) can be reformulated. Additionally, another value called *precision* (p) can be deduced. The calculation of the three values is presented in Equations 3-5.

$$r = \frac{TP}{TP + FN} \tag{3} \qquad p = \frac{TP}{TP + FP} \tag{4} \qquad f = \frac{FP}{FP + TN} \tag{5}$$

These values may be used to build diagrams. In a *ROC (receiver operating characteristic) Curve*, the false positive rate (fallout) is plotted against the detection rate (recall). The *Recall-Precision Graph* plots recall rates against precision. In both cases, this is done for different parameter settings of the detector, yielding diagrams that show interpolated lines of ROC- and Recall-Precision value pairs.

For learning algorithms, two additional matrics are used: the *learning time* and the *number of samples required* in training.

There is a lack of standardized test scenarios for face detectors which adds to the difficulty of a comparable evaluation. However, several image databases exist for the purpose of evaluating detectors and face recognition systems. Some of these provide images of single persons with a controlled background (see Table 1), others contain images with several faces and complex backgrounds and lighting. Table 2 presents some of these test sets and their properties. The *CMU data sets* have been collected by Henry Schneiderman and Takeo Kanade and are widely used in the scientific face detection community. A part of the set contains faces with out-of-plane rotations. The *BioID Face Set* contains a large set (1,521) of gray level images of 23 individuals with natural background. Additionally, data files are available that contain 20 manually labeled facial feature points per face (for example left eye pupil, right mouth corner and others). Thus, this database can be used to train face detectors and to evaluate detection and gesture recognition systems in a comparable way.

Table 2: Face databases for evaluation of face detectors.

Database and Location	Description
CMU Frontal Test Sets [43] Carnegie Mellon University Massachusetts Institute of Technology	3 sets of upright frontal face images (42 and 65 images); the database includes face images collected at CMU and MIT; all sets contain 1 or more human faces, cartoons or drawings; ground truth available
CMU Rotated Test Sets [33] Carnegie Mellon University	1 set of face images (50 images), each containing one or more human faces; faces appear rotated about the first (i.e. roll) camera axis; ground truth available
BioID Face Set[14] FAU Erlangen AG	1,521 images, multiple faces per image, natural conditions (background, lighting), manually labeled eye positions in data file

2 Approaches

In the following sections several approaches to face detection are discussed. Section 2.1 covers knowledge-based approaches and Section 2.2 describes methods that use invariant features. In Section 2.3 we present approaches with template matching and Section 2.4 covers appearance-based methods.

2.1 Knowledge-based approaches

Generally, knowledge-based approaches to face detection try to encode human knowledge of faces in rules. They match face candidate regions against these rules to classify them as being a face or not. The rules are based on the results of research in the fields of pattern recognition and mathematics but also biology and medicine. After all, rules are often derived from image material directly. The rules mainly encode *ratios*, *similarities* and *symmetries* of human face features and their relationships, for example relative distances and positions. For example, a simple principle could be: 'faces often have two eyes, a nose and a mouth and the direction of the nose constitutes an axis of symmetry'. Thus, the challenge is 1) to construct a broad set of well-defined rules (and their mathematical representation) and 2) to combine them in a meaningful way in order to construct a classifier.

Representative work on the subject has been performed by Yang and Huang [54]. Their approach is a hierarchical system with three levels of image resolution and respective rules. An image is subsampled to three resolutions allowing to apply different sets of coarse to fine rules. While at the highest level (coarse image representation) general descriptions of face patterns and homogeneous patches are present, finer levels rely on more detailed descriptions of facial features.

As basis for most rules serves an abstract face as presented in Figure 2a. At the top-most level, coded rules are derived, for example 'the center part of a face has four cells with basically uniform intensity' (the dark shaded area in Figure 2b) or 'the difference in gray scale intensity between the center part and the upper round part of the face is significant' as described in [56]. Positively classified face candidates are then analyzed in a next stage. Here, the images are represented with better resolutions and after local histogram

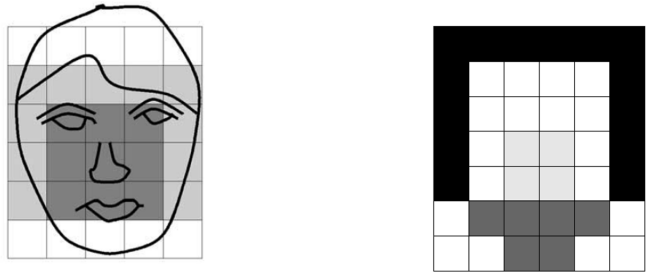

Figure 2: Abstract faces used in knowledge based methods. Rules, derived from human knowledge on characteristics of facial regions are derived from abstract face models in [54] (a) and [16] (b).

equalizations, edge detection is performed to detect the contour of a face. The last step operates on the original resolution of the image where rules are applied that take eyes, mouth and other facial features into account.

Interestingly, even in this early work, a coarse-to-fine or focus-of-attention strategy was chosen. This is done by Viola and Jones years later as well [49]. We show the great impact that this approach can have on speed and accuracy of a detection system in Section 4.2. Although the design of the stages is quite different, the cascaded detection framework that is used in this work uses a similar focus-of-attention strategy.

In 1997, Kotropoulos and Pitas picked up the ideas of Yang and Huang in [16], retaining the rule- and hierarchy-based concept. Their approach is, to first locate facial features with a projection method. The horizontal profile H_I and vertical profile V_I of an $m \times n$ image I are calculated as $H_I(x) = \sum_{y=1}^{n} I(x,y)$ and $V_I(y) = \sum_{x=1}^{m} I(x,y)$. From H_I, the left and right boundaries of a face can be extracted by searching for local minima. Similarly, the local minima in V_I correspond to eye and mouth region and the maxima between them to the nose area. These facial features are then evaluated with several rules. The approach performed well on a training set that comprises 37 video sequences showing single persons in front of a uniform background (86.5 percent detection rate of facial features). However,

the approach is not suited for more complex backgrounds or multiple persons in one image. Thus, it is rather a face localization than a face detection method.

2.2 Invariant features approaches

The second group of face detection methods are feature-based approaches. Here, efforts are put on detecting features such as *edges*, *texture*, *color* or generally *facial features* (lips, eyes, eyebrows, hairline), among others. Some of these features are assumed to be potentially invariant for different poses, facial expressions and lighting conditions. The underlying assumption is that humans can detect faces independently of these variables.

Some methods that aim at detecting *facial features* (mainly early works in face detection) are built upon edge detectors. Commonly used edge detectors are the Canny detector [1], the Sobel [40], Prewitt (similar to Sobel without weight setting) and Roberts (fast, poor performance) operators and the Marr-Hildreth-Operator (Laplacian of Gaussian, LoG) [21]. Once several facial features are extracted, statistical models and known proportions in human face geometry are used to verify the existence of faces. Problems arise from difficult illumination conditions (shadows might introduce new edges) or occlusions (corrupted feature boundaries). An example for a face localization method based on edge detection is the work by Sirohey [39]. Here, an edge map is calculated first and then processed by some heuristics to group or remove edges in order to obtain a contour image of the face. Further, Sirohey fits ellipses to the obtained outline to segment the face candidate. Then, Gabor wavelet decomposition and graph matching is performed to extract facial feature points.

There exist other methods than edge detection to extract facial features. Chetverikov and Lerch use *blobs* and *streaks* instead of edges for their detection system in [2]. Two dark and three light blobs correspond to the eyes, cheekbone and nose region in their model. The detection process comprises a search for triangular configurations of these blobs in candidate images. The streaks are used to represent outlines of the faces and to verify face

candidates. Operations in other approaches include morphological transformations and various graph algorithms [56].

More recent work on facial feature extraction assumes a ready-to-use face detection system and regards feature extraction as a second step, for example in the area of facial expression analysis. Cristinacce et al. proposed a multi-stage feature extraction system in [4]. The authors first utilise the cascaded face detector of Viola and Jones [51] and then apply multiple feature detectors using a method called Pairwise Reinforcement of Feature Responses (PRFR). Extracted feature points are then verified using a variant of the Active Appearance Model (AAM) search which is tuned to edge and corner features. They report to have achieved the best results among many other feature detector combinations. A comparison of shape constrained facial feature detectors can be found in [3].

While the before mentioned algorithms work on intensity images, *color* adds a powerful clue to detection. The main discussion among researchers in the field is about the selection of a suitable color space. A popular color space is the *normalized RGB space*. Based on the RGB color model with the three primary color components R (red), G (green) and B (blue), the components r, g and b of the normalized RGB space are defined as:

$$r = \frac{R}{R+G+B} \quad (6) \qquad g = \frac{G}{R+G+B} \quad (7) \qquad b = \frac{B}{R+G+B} \quad (8)$$

The three values sum up to 1, thus b is redundant and need not to be stored as it can be calculated as $b = 1 - r - g$. Besides the effective representation as (r, g), an important advantage of the color space is the possibility to filter out luminance (brightness). This is a strong feature as it has been shown, that human skin tones vary mainly in luminance [55].

The *YCrCb color space* can be used to segment skin from background pixels by simple thresholding. The color space has been used by Tsalakanidou et al. for an authentication system. They utilize 3D range data and color images to robustly identify people regardless of certain amounts of rotation and occlusion [45].

The strength of the *HSI (hue, saturation, intensity) color space* is its non-linear transformation of the RGB space. Thus, unlike normalized RGB or YCrCb, the extraction of facial-features (e.g. lips) need not to be based on exhaustive training as the segmentation is easier with this angularly transformed color space [19]. A drawback is the need for high quality source material, as noise of mono-CCD material is problematic for angular color transformations.

A summary of the before mentioned commonly used color spaces, their advantages and examples for applications is presented in Table 3.

Table 3: Color spaces for face detection.

Color space	**Advantages**	**Sample work**
normalized RGB	fast calculation, effective representation, luminance (brightness) can be filtered out	[55]
YCrCb	simple skin tone extraction with thresholds	[45]
HSI	large variance among facial feature color clusters	[41]

Although color is a powerful feature that is supportive in face detection if the color model can be adapted properly for different lighting conditions, Yang et al. note that it is not effective when the spectrum of the light source varies significantly [56]. Thus, skin color alone is often not suitable to detect faces. Recent methods utilize skin color as one of several features to better detect faces. Hota et al. [12] use the cascaded face classifier and AdaBoost as proposed by Viola and Jones [49] for face detection in video and extended the system by a skin color detector. Consequently, they were able to track faces over several frames of video material even if the face got rotated during the shot. A detection by the cascade served as basis for continued tracking with color segmentation. An approach of Foresti et al. in 2004 integrated color segmentation based on the YCrCb color space and the eigenface method (see Section 2.4) to detect faces. Their segmentation method is depicted in Figure 3.

There exist various comparisons of color models for the purpose of skin segmentation, some of which introduce new performance metrics. For example Phung et al. compiled a comparative study in [31].

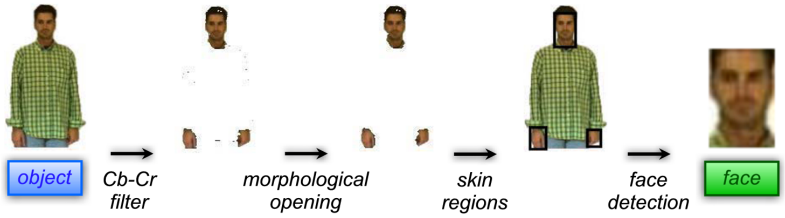

Figure 3: Segmentation based on YCrCb color space as performed by Foresti et al. [5]. The focus of attention strategy uses color information to extract potential face candidates. These are verified with a face detection method (eigenfaces).

2.3 Template Matching

Template matching makes use of predefined standard face patterns that are matched against face candidate sub-windows. Templates encode knowledge about the human face, for example the size and approximate shading and color of different parts of faces and their relation. The definition of templates is either performed manually or parameterized by a function. The standard approach is to calculate the correlation of the predefined template with different parts of a face candidate region [56]. Mouth, eyes and nose positions and the contour of a face are typical features that are matched against the template. The obtained correlation values are refined, combined and thresholded in order to classify a face candidate as face or non-face. Similarly, some approaches use the Hough transform to match candidate regions against templates [27].

A problem of the first template approaches was the strong response to changes in scale and shape of faces (faces are non-rigid and subject to steady deformation) which lead to poor detection rates in more challenging setups, for example when tested against scenes with unsteady background, different sizes of faces and changing illumination. However, newer approaches use multi-resolution or deformable templates to achieve scale and shape invariance.

Sinha proposed a template matching method where the spatial template defines areas of the face [38]. These areas correspond roughly to facial regions such as eyes, mouth, nose and chin. The approach compares a selection of

relative brightnesses of face regions to each another. The underlying assumption is that these relations remain unchanged even if the overall illumination of the face changes. Thus, the method defines several *brighter-darker constraints* (e.g. "an *eye* region is darker than an *upper-cheek* region") and a face is detected if a candidate satisfies all pairwise brightness relations.

Scassellati [35] enhanced the system of Sinha. The author uses the template approach with brightness relations to implement a robot vision system. First, potential face locations are extracted with a motion-based method, then a refined version of Sinha's template matching method is applied. Figure 4 shows the enhanced template of Scassellati. It features 16 areas and a total of 23 brightness relations. It is obvious that the areas that correspond to the eye regions are subject to many of these brightness constraints, which is typical for face detection. The eye region and its relative brightness to surrounding areas of the face is a distinctive feature in face detection, which is also exploited by Viola and Jones' feature approach (see Section 4). Scassellati reports a detection rate of over 94% for their actual goal of eye detection (the approach aimed at reacting on social cues like eye contact or gaze direction). The author further states that Sinha's method alone performed poorly on a set of standard static test images.

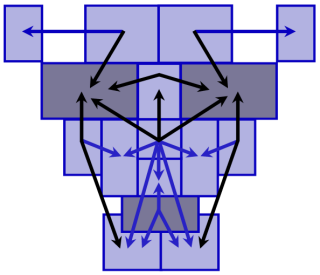

Figure 4: Spatial face template, encoding 16 face regions that correspond to facial features and 23 brightness relations. Black arrows indicate essential relations (necessary for detection), blue arrows confirming relations (not necessary). The template is an enhanced version of Sinha's original template and was proposed by Scassellati in [35].

The method of brighter-darker constraints could be extended with a training stage, where the brightness relations are determined empirically. For example, Murai et al. calculated templates from training images of faces

in [23]. After a preprocessing stage, the samples are iris-aligned and averaged to obtain face templates. Additionally, the templates were scaled to detect faces of all possible sizes in the detection domain.

Perez et al. suggested a more recent template approach [27]. Their method is divided into three steps. The first step is a rough face detection based on Hough transform to obtain a directional image. Consequently, elliptical templates of different sizes are scanned over the directional image, i.e. elliptical forms are detected in the Hough transformed image and the centers are estimated roughly. The second stage of the algorithm works with anthropometric templates (i.e. templates constructed from known proportions and measurements of human faces). These templates basically encode eye, nose and mouth region and the lower half of the facial contour as ellipse. The templates are rotated in the range $[-40°, 40°]$ and are matched against the face candidates from stage 1. The last stage deals with iris detection. An array of iris templates (containing different occlusions by eye lids) is used to determine eye positions and to track the iris.

Jain et al. compiled a review of different template approaches for object detection [13]. They concentrate on deformable templates and present approaches not only for face detection but for general object detection in different domains, for example the segmentation of cars, birds, road, microscopic forms or parts of the brain.

2.4 Appearance-based approaches

Appearance-based approaches apply eigenfaces, support vector machines, hidden markov models, naive bayes classifiers and neural networks for face detection [56]. These approaches could also be labeled learning-based as they have in common their need for exhaustive training with sample images. From the training databases, the models needed to classify potential face windows are derived by statistical analysis and with machine learning methods (though in very different ways in the individual approaches). The main goal of most approaches is the reduction of dimensionality to enhance computation time and increase robustness to noise. Appearance-based ap-

proaches have been widely used within the last years. Some methods and representative work is presented in the following.

The use of *eigenfaces* was first proposed by Kirby and Sirovich in [15] (however, the name was established later). The method is based on the Karhunen-Loève transform, which is also known as principal component analysis (PCA) or Hotelling transform. The basic idea is to reduce the dimensionality of the data (sample images) in a way that preserves a maximum of variance (which corresponds to the information content). Thus, the eigenvectors of the covariance matrix (i.e. vectors, that span an optimal subspace) are calculated from the set of vectorized training images. The dimensionality of this subspace can be reduced effectively. Eigenvectors with little information content are discarded which produces a space that is less complex than the original space of face samples while keeping most of the information.

Sung and Poggio present a method that uses eigenvectors in a *distribution based* setup [44]. They first model face prototypes from training images that are scaled to a size of 19×19 pixels and masked with a circle such that near-boundary pixels not belonging to the face are culled. Similarly, non-face prototypes are modeled. They form several clusters of face- and non-face samples in a predefined space and as input to the detection stage compute a Mahalanobis-like distance of the 75 first eigenvectors of each cluster and the candidate window which is projected on the space spanned by these eigenvectors. Similarly, the Euclidian distance is calculated and the obtained values are analyzed by a Multi-Layer-Perceptron (MLP) with 24 hidden units. Sung and Poggio report a 80% detection rate on a test database of images of average quality. Their approach is well known and has been widely used for eigenvector- or distribution-based object detection.

Eigenfaces are often used for face recognition as well and in more recent approaches in combination with other methods. Concerning face recognition, it has been shown, that the eigenface method is robust against local changes but not against global changes. Thus, it is well suited to recognize faces of the same person with different expression, clothing or hair cut while problems arise when the same person is viewed under different lighting conditions. Foresti et al. suggest to combine a neural network and color segmentation

with eigenfaces for face detection [5]. First, they perform outline analysis with an MLP neural network. The outline of a moving object in the scene is represented as a B-Spline. The upper part of the shape is analyzed by the neural net in order to find the head region. The second part consists of color segmentation in the YCrCb color space (see Figure 3 in Section 2.2). The third part utilizes eigenfaces in a similar way as Sung and Poggio as described above. They perform data fusion with the different possible head regions with a weighted vote. The weights are determined experimentally.

One important advantage of eigenface approaches is that they need less training images than other learning-based approaches. Most authors report a database size of approximately 50 to 300 face images for their methods.

Support Vector Machines (SVM) have been used in appearance-based approaches as well. In contrast to eigenface/PCA methods, patterns are projected explicitely to a higher dimensional space (the *feature space*). Consequently, the goal is to find an optimal decision surface (called *hyperplane*) in the n-dimensional feature space to discriminate the projected face and non-face patterns. The hyperplane is defined by a weighted combination of a small subset of training vectors, called the *support vectors*. The optimal hyperplane is linear in feature space and has maximum distance to the closest points (i.e. support vectors).

Heisele et al. propose a cascaded architecture of Support Vector Machines in 2003 [10]. They construct five increasingly complex SVMs that are arranged in a cascade for detection. This is done in a similar way as proposed by Viola and Jones which was used in this work and is described in Section 4.2 [49]. Heisele et al. explicitly reduce the complexity of their non-linear top layer SVM to reduce computation time. Therefore, they rank features that are derived from training samples and perform PCA. In the face detection domain, they report to have removed 99% of the *PCA gray features* without loosing information. Thus, by using a hierarchy (i.e. cascaded architecture) and feature reduction at the top level, they are able to speed-up the system by two orders of magnitude while retaining a similar classification performance.

One of the most significant works in the area of *Neural Networks (NN)* in face detection has been performed by Rowley et al. in [34]. The structure

of artificial neural networks is inspired by the biological neural network, i.e. interconnections of neurons in the human brain. An artificial neural network basically consists of simple processing units (also called neurons) that are organised in layers and their (weighted) connections. A neural network is usually designed as an adaptive system that is trained by updating its weights. Thus, the network changes its structure based on the inputs during the learning phase.

Figure 5 shows a simple neural network with one input and one output unit. These units are connected with a network of hidden units (here organized in two layers), that perform functions on their respective inputs. A probabilistic view on this setup could be stated as: a random variable $F = f(G)$, for example the classification result in face detection, depends upon the random variable $G = g(H)$ which depends upon $H = h(X)$ which depends upon random variable X, for example some representation of a candidate image region. Thus, the network can mathematically be seen as a tool for non-linear statistical data modeling. Nodes of a single layer are independant from each other which allows parallelization.

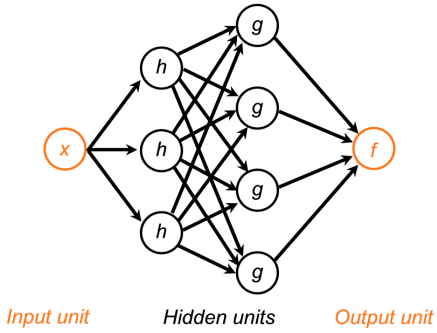

Figure 5: A simple artificial neural network (NN). The value of an input unit x is used by two layers of hidden units that perform functions h and g with different sets of parameters. The result is an output f. The arrows represent connections between the nodes which are weighted in networks that are used for learning.

Rowley et al. propose a system that consists of two stages [34]. First, a neural network-based filter is used to classify candidate image regions, then they apply a function that merges detected regions from different filter-sizes (i.e. outputs of different NNs) and eliminates ovelapping regions. They

use a manually labeled and normalized set of over 1000 positive training images. Negative (non-face) training images are collected by bootstrapping, see Section 4.2.7 for details. For preprocessing, they match a linear function to an image region and consequently subtract it. Thus, lighting conditions can be compensated. Furthermore, they perform histogram equalization to achieve better contrast in candidate images. Their neural network classifies 20×20 pixel subwindows and uses three types of hidden units that work directly on the pixels of the candidate subwindow: 4 which look at 10x10 pixel subregions, 16 which look at 5x5 pixel subregions, and 6 which look at overlapping 20x5 horizontal stripes of pixels. The units are chosen in this way to be sensitive to different facial features. The stripes, for example, are sensitive to the mouth region or the pair of eyes.

Viola and Jones propose an approach to face detection that aims at constructing a detection cascade [51]. Their method is a learning algorithm, depending on thousands of positive and negative training images and is based on primitive Haar-like rectangular features. The approach is described in detail in Section 4.

Other appearance-based approaches to face detection include Hidden Markov Models (HMM) [24], Genetic Algorithms (GA) [57] and Naive Bayes Classifiers [28]. In the latter class of methods, an image is regarded as random variable x which is characterized by the class-conditional density functions $p(x|face)$ and $p(x|nonface)$. Consequently, Bayes' theorem and, for example, the maximum a posteriori decision rule, are used to calculate $p(C|X)$, i.e. the probability of a class C (*face* or *non-face*) given a new example image X.

There exist various combinations of the presented and other methods. For example, Li et al. proposed an approach for multi-view face detection in 2004 [20]. They first use support vector regression to estimate the pose of an object in three dimensional space. They choose one of several face detectors that suits the pose and classify the image as being a face or not with a combination of support vector machine and eigenface approaches.

3 Source material

In the work at hand, the source material for face detection are old films of Dziga Vertov. The project that constitutes the context for this work is called *Digital Formalism: The Vienna Vertov Collection* and is described in Section 3.1. Special challenges of the material are summarized in Section 3.2. We provide information on the digitization process and properties of the digitized films in Section 3.3.

3.1 The Vienna Vertov Collection

Dziga Vertov (born David Abelevich Kaufman) was a Soviet avant-garde film-maker and film-director. He was a pioneer in the area of documentary films and newsreels (short documentaries that were presented publicly and covered news stories and other topics of interest of the time). His first productions date back to 1919 (the newsreel *Kino Nedelya*, Cinema Week). His best-known production today is the film *Man with a Movie Camera* which was produced in 1929.

Digital Formalism: The Vienna Vertov Collection is a project that focuses on the digital analysis of the "senses of cinema". The project is carried out by three project partners: The *Austrian Film Museum* (OeFM), the *Department for Theatre, Film and Media Studies* (TFM) at University of Vienna and the *Interactive Media Systems Group* (IMS) at Vienna University of Technology. It is funded by the WWTF (Vienna Science and Technology Fund) as a three-year project and started in 2007. The research aims at developing methods to analyze principal cinematic elements in the films [17]. Computer scientists develop methods to analyze high-level film-elements like rhythm, contrast and dialogue sequences in the films.

This work focuses on face detection, which forms the basis for extracting people, recognizing reappearing characters and to enhance understanding of peoples' interactions in different sequences. The films that are used in this work are listed in Table 4.

Table 4: Films of Dziga Vertov used in the work at hand.

Year	Original title	English title	Length
1924	Кино-глаз	Kino Glaz (Cinema eye)	78 minutes
1925	Киноправда	Kino Pravda (Film Truth)	32 minutes
1926	Шестая часть мира	A Sixth of the World	64 minutes
1928	Одиннадцатый	The Eleventh	58 minutes
1929	Чеповек с киноаппаратом	Man with a Movie Camera	16 minutes

3.2 Material properties

As described above, the material at hand dates back to the 1920's. The silent monochrome films (see Table 4) were originally recorded on 35 mm film strips. The films were copied for backup reasons. They were poorly stored and treated because a presentation or other use of the material - years later - was not intended. However, meanwhile the original film strips are not traceable anymore and were possibly destroyed. Thus, the available bad-quality backup copies are the only remaining source for the films. They were digitized recently by the Austrian Film Museum (see Section 3.3) to preserve the films and make them accessable to automatic analysis.

The material has suffered from poor storage and repeated analog copying. The most disturbing artifacts in the digitized film copies include:

- **Flicker**, which is mainly caused by the cameras used for recording. The manual film transport produces uneven exposure and flicker.
- **Scratches**, which result from repeated playback of the analog films.
- **Coarsely grained and blurred scenes**, which are possibly caused by poor focusing and bad lighting during recording and repeated analog copying.
- **Dirt**, which is a result of poor storage, transport and copying under suboptimal conditions (dust, humidity, liquids).

- **Image vibration**, caused by shrunken film strips. The biochemical material shrinks with time and changing temperature horizontally and vertically, resulting in misaligned frames and image vibrations.
- **Bad lighting with over- and underexposure** as a result from recording difficulties in very dark and light scenes (for example night shots or scenes that show underground mining) and problems with back light.
- **Bad contrast** in some scenes which is caused by repeated copying of the analog films.
- **Border and frame lines** become visible due to copying of shrinked filmstrips.

Note that most of the above mentioned artifacts accumulate during analog copying. Figure 6 shows sample frames from the material that bear some of the mentioned image deficienies.

We confront these challenges with various techniques. First of all, the chosen algorithm for face detection is robust against *minor scratches* and *varying brightness* of a face. Second, we use variance normalization to account for *varying lighting conditions within a face* and *flicker*. Concerning *scratches* and *coarse grained images* the use of filters (for example median or mean filters) of different sizes is suggested. We use a Wiener filter to deblur the images after smoothing with median or mean filters. A Wiener filter is an adaptive linear filter, that tailors itself to the local image variance. Thus, it is able to preserve edges while removing *noise* and *blurring* to some extend. Additionally, the active region where faces are detected is limited at the frame borders to avoid problems with the *border artifacts and frame lines*.

We use a base resolution of 24×24 pixels for the training of the face detection system, but due to the coarse grained source images, faces at this size are hardly detected by the program. In fact, in most cases, it would be even difficult for humans to recognize a face of this size in the films without contextual information (for example a human corpus or a priori knowledge from preceding film frames). Thus, the minimum detection size is set to a

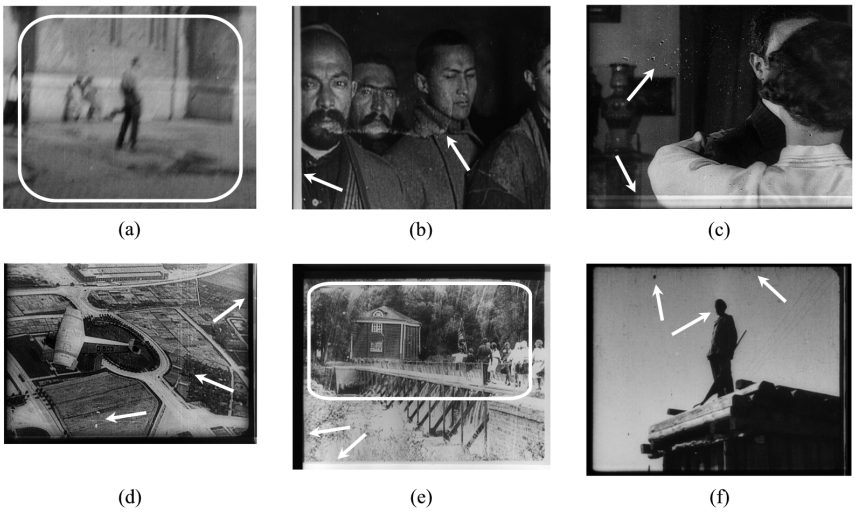

Figure 6: Six sample frames from the analyzed films (artifacts marked with white borders and arrows). (a) blurred frame 2,444 from *Man with a Movie Camera*; (b) frame 25,190 from *Kino Pravda* with an artifact from spoiled liquid through the faces and difficult lighting; (c) frame 3,015 from *A Sixth of the World* with scratches in the upper left area of the image, possibly introducing false-positive detections in this area and visible frame lines at the bottom; (d) frame 845 from *A Sixth of the World* with border and frame line artifacts, vertical lines and dirt; (e) frame 3,629 from *Kino-glaz* with dissolve artifacts in the marked area, border and frame line artifacts; (f) frame 12,199 from *The Eleventh* with dirt and scratches and underexposure of the object of interest because of back light.

larger value (twice the base resolution). Section 4 provides more insight into the face detection algorithm.

3.3 Digitization

The Austrian Film Museum provides the source material for this investigation by digitizing the films of Dziga Vertov. The films are scanned frame by frame from the 35mm film strips. Optical scanning is performed in real time at 25 frames per second. Table 5 gives an overview of the properties of the scanned films. All films are silent and monochromatic.

Table 5: Properties of the digitized films of Dziga Vertov.

English Title	**Broadcast system**	**Number of frames**	**Resolution**	**Total memory**	**Image format**
Kino-glaz	PAL	84,133	720×576	23.4 GB	TIF (LZW)
Kino Pravda	PAL	35,060	720×576	9.5 GB	TIF (LZW)
A Sixth of the World	PAL	69,181	720×514	23.2 GB	TIF (LZW)
The Eleventh	PAL	63,123	720×576	18.4 GB	TIF (LZW)
Man with a Movie Camera	PAL	17,577	720×576	5.0 GB	TIF (LZW)

Note that the resolution of *A Sixth of the World* differs from the other films. This is because the film is already cut to the area that contains the image content (i.e. borders are cropped). Note also that *Man with a Movie Camera* is not available completely for this work. Only the last reel, containing the film's last 16 minutes (17,577 frames) was available during this investigation.

4 Face detection method

This section covers a detailed description of the face detection algorithm we implemented. It is based on Viola and Jones' method [49] with the Forward Feature Selection of Wu et al. [53]. Additionally, we present suggestions for post processing the results.

4.1 Preprocessing

Both, training samples and candidate subwindows during detection are variance normalized to compensate for different lighting conditions. This is suggested by various authors, including Viola and Jones in [49] and necessary because of the flicker in the old material that is analyzed. The statistical background is presented here, the efficient calculation using so called integral images is described in Section 4.2.3.

Basically, the process is a *Z-normalization* of individual images X, i.e. the calculation of $X_{norm} = \frac{X-\overline{x}}{s}$, without a shift of the negative mean. Thus, the normalization is given as

$$X = \frac{X}{s} \tag{9}$$

where X is the input image (subwindow) and s is the standard deviation of X. The variance s^2 is defined as

$$s^2 = \frac{1}{n-1}\sum_{i=1}^{n}(x_i - \overline{x})^2 \tag{10}$$

where $\overline{x}$ is the mean of the values x_i, i.e. $\overline{x} = \frac{1}{n}\sum_{i=1}^{n} x_i$. This can be computed more efficiently by

$$Var(X) = E(X^2) - (E(X))^2 \tag{11}$$

where $Var(X)$ is the variance of random variable X and $E(X)$ is its expected value. Thus, s^2 can be calculated as

$$s^2 = \frac{1}{n-1}\sum_{i=1}^{n} x_i^2 - \frac{n}{n-1}\overline{x}^2. \tag{12}$$

where $\overline{x}$ is the expected value of X. Note, that we normalize the variance by the widely used factor $\frac{1}{n-1}$ (another used normalization factor is $\frac{1}{n}$).

Utilizing Equation 11 for the computation of the variance is especially useful when using the integral image representation as the expected value can be computed very efficiently. An outline of the calculation is presented in Section 4.2.3.

4.2 Training

In the following we describe the training process for the face detection method proposed by Viola and Jones in [49, 51] with certain changes that better suit the purpose of analyzing archive films (i.e. the set of training images) and improvements, addressing a faster feature selection process. We start by presenting an outline of the process in Figure 7.

In a first step the positive and an initial set of negative image samples are read in and a large set of rectangular features is calculated. These images are then converted to their integral image representation which allows for efficient area calculation (needed for feature evaluation). The training process starts with finding the optimal thresholds for the features. This is done by applying each feature to each of the training images, followed by a combination of median functions and lokal searching to obtain the best fitting value. A feature with its corresponding threshold is called weak classifier. The next part is the selection of a combination of weak classifiers to form an ensemble which is done by forward feature selection in this work, following the suggestions of Wu et al. in [53, 52]. Forward feature selection leads to significant improvements in the overall training runtime. After weight setting for the ensemble classifier, a bootstrapping step completes the execution of one training cycle, which is repeated to create a cascade of classifiers. The algorithm terminates when the learning goal or the maximum depth of the cascade is achieved.

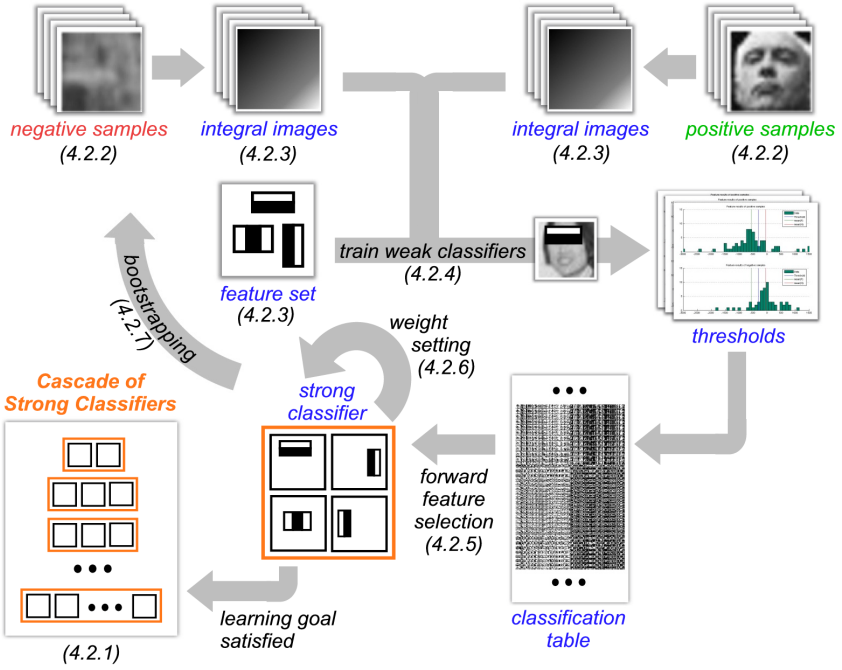

Figure 7: Schema of the training process. Input are positive and negative training samples, output is a cascaded classifier. Numbers in brackets correspond to the section numbers where the subprocesses are discussed.

The above described procedure will be depicted in more detail in the following sections. We start by presenting the concept of a *cascaded classifier* which will be created as a result from training in Section 4.2.1, go on by describing the set of *training images* in Section 4.2.2 and explain the *rectangular feature set* and its efficient calculation with the *integral image representation* in Section 4.2.3. The training of a weak classifier, the smallest single part in the detection system, is presented in Section 4.2.4. In Section 4.2.5 we will focus on the concept of *majority voting* (by FFS). The calculation of a weighted ensemble of selected features can be done with a variant of AdaBoost [52] or any other weight setting algorithm. The procedure is presented in Section 4.2.6. The description of the training process is completed by a short presentation of the bootstrapping concept which provides the basis for achieving the goal of encouraging independant errors among the classification (cascade-) stages in Section 4.2.7.

4.2.1 The idea of a cascaded classifier

One of the important contributions of Viola and Jones in [49] and core feature of their face detection system is the cascaded architecture of their detector. The key idea is to construct a detector that successively becomes more distinctive. Early detection stages aim at rejecting a majority of the non-face candidate images that are scanned by the detector while detecting almost all positive instances. The detection rate of the stage is far more crucial than its false-positive rate. Later stages become more complex and try to achieve a low overall false-positive rate. The approach is motivated by the fact that face detection is a typical example for *rare event detection*, i.e. a highly asymmetric classification task. Millions of sub-windows need to be processed by the system and only few contain targets, i.e. faces. This adds to the difficulty of the face detection problem: while the detection rate (as in most detection systems) is supposed to be very high, the false-positive rate must be minimal, i.e. in the order of 10^{-7} to avoid a flood of non-events to be passed through.

The "attentional cascade" [49] is illustrated in Figure 8. Note that the complexity of the cascade's stages increases. While a large amount of sub-windows is rejected during the first stages, later stages consist of more features and are therefore more precise while computation time also increases. The rejection of a majority of sub-windows during the efficient first stages (practically, each stage is supposed to filter 50% of the remaining false-positives) adds to the efficiency of the cascaded decision structure. The classifiers themselves are built from *rectangular features* are described in Section 4.2.3.

The exhaustive set of sub-windows is processed completely by the first stage. While this stage rejects a large number of windows, it is supposed to pass almost all positive instances to the next detection stage and so on. Overall, a sub-window is rejected and classified as non-face if it is rejected by *any* of the classifiers (cascade stages). Thus, a sub-window is classified as face if and only if it passes *all* combined classifiers.

Mathematically, this approach requires some prerequisites in order to work as intended. The individual classifiers (stages) are tuned to a false-

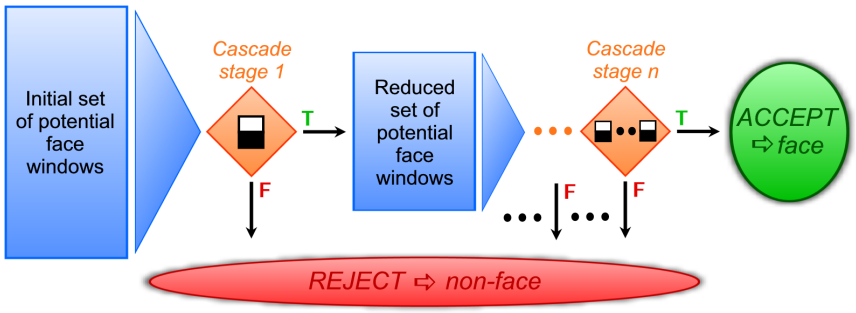

Figure 8: Face detection with a cascaded classifier with n stages. A sub-window's classification is *non-face* if it is *rejected by any* of the stages, thus it is classified as *face* if it is *accepted by all* stages.

positive rate of only 50%. This allows high indiviual detection rates of over 99% which are necessary for the cascaded detection approach. Remember, a sub-window that is only once (by only one stage) classified as non-face is rejected by the system and cannot be brought back, neither will it be classified again. When we assume independent errors of the different cascade stages (it will be shown how we achieve this by bootstrapping in Section 4.2.7), we can define the overall detection- and false-positive-rate d and f, respectively, by means of d_i and f_i, the individual rates:

$$d = \prod_{i=1}^{n} d_i \qquad (13) \qquad\qquad f = \prod_{i=1}^{n} f_i \qquad (14)$$

Thus, a hypothetical system with a cascade of 20 stages and individual rates of $d_i = 0.99$ and $f_i = 0.5$ would yield a detection rate d of 81.78% and a false-positive rate f of $9.6 * 10^{-5}\%$.

4.2.2 Training Images

The proposed algorithm needs, as it is typically for appearance-based detection methods, exhaustive training. Larger training sets constitute a broader and more general basis for training and therefore better detection and false-positive rates. Several thousands of images have been used in this approach, divided in a set of positive (face) and negative (non-face) sample images.

The set of negative training samples needs to be even larger due to the unequal complexity of the two classes *face* and *non-face*. While the face class is well defined (i.e. a frontal face features two eyes and a mouth, we regard faces from the hairline to the chin, etc.), the non-face class consists of images that can virtually be everything, thus featuring an extensive set of patterns. We describe how we generate and process the large set of negative samples in Section 4.2.7. In Section 5.3 we describe the data sets employed for our experiments.

4.2.3 Rectangular Features and Integral Images

An important concept of Viola and Jones' approach is the simplicity of the classifiers that are used. The stages of the cascade mentioned in the previous section are set up with an increasing number of individual classifiers. These *weak classifiers* consist of a single feature which is easy and fast to calculate, and a threshold which is determined during training. The feature itself consists of two or more rectangular regions. The selection of classifiers is also done during the training process.

Figure 9 shows visualizations of the different shapes of features. The key idea is, that these features are moved over a sub-window of an image and the sum of intensities under the black and white area of each feature is calculated, respectively. Then, one of the values is substracted from the other one, yielding an integer value of some magnitude. The calculation process of pixel areas can, mathematically, be seen as a *matrix convolution*. The use of simple features that are evaluated with, basically, a matrix convolution, is inspired by the work of C. Papageorgious et al. on Haar basis functions in [26].

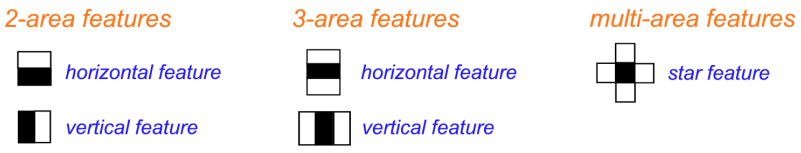

Figure 9: Different types of used features. Note that all feature types are build from rectangular regions.

The idea is, that the calculation process mentioned above yields characteristic integer values for face- and non-face sub-windows, respectively. An analysis of several hundred face- and non-face windows shows, that this hypothesis is true for some features. For example a horizontal two-rectangle feature evaluated over the region where eyes normally appear in a sub-window that contains a face, yields high negative numbers for face windows, while the same feature evaluated over non-face windows produces values around zero. The evaluation of a feature on top of a face and a non-face image is depicted in Figure 10. The feature scores (sum of intensities of white area subtracted from sum of intensities of black area) differ significantly for the two images.

Figure 10: Horizontal 2-rectangle feature on top of a face and a non-face.

This induces a simple classification process for a single feature: once a threshold is determined (by training with thousands of positive and negative samples), classify a test window as face if the calculated value is above/below the threshold and as non-face otherwise. The information of positive instances scoring above or below the threshold (we call this information the *parity* of the threshold) needs to be stored along with the threshold and feature chracteristics. A classifier therefore consists of: a *threshold*, the *parity information* and the particular *type*, *x-offset*, *y-offset*, *width* and *height* of the feature. The process for obtaining the threshold for each classifier is described in the next section.

Said, that each feature type appears in different sizes (width and hight) and positions (x- and y- offsets), it is obvious that we deal with a large number of classifiers. The number n of 2-area-features of a single type with respect to width w and height h of the sub-window and width w_f and height

h_f of the feature (which is scaled) is computed as follows (assuming a base resolution of 2×2 pixel for the feature):

$$n = \sum_{w_f=2}^{w} \sum_{h_f=2}^{h} (w - w_f + 1)(h - h_f + 1) \tag{15}$$

For a resolution of 24×24 pixel for the test- and detection sub-windows, as it was used in this implementation, the equation evaluates to a sum of $76,176$ different features of a single type. Practically, on one hand we use a larger step size for x- and y-offsets which reduces the feature set, on the other hand we have different types of features, as illustrated in Figure 9, which boosts the set again. All together, $68,000$ features are created in our experiments. This set is, in terms of linear dependencies, many times overcomplete - a difference to the linearly independent Haar basis in [26]. Although this exhaustive set is subject to a tight selection process which is described in consequent sections, an efficient calculation of the scores of features for the large set of test images is crucial.

We mentioned before, that the calculation of the scores of features described above could be done by matrix convolution. However, Viola and Jones introduce a faster and more efficient method in [49] which makes use of what is called an *integral image*. An integral image is a representation, where each pixel's value is the sum of all pixels above and to the left from the original image. An integral image ii therefore is computed as

$$ii(x, y) = \sum_{x'<=x, y'<=y} i(x', y') \tag{16}$$

where x and y are the pixel position in the integral image $ii(x, y)$ and the original image $i(x, y)$. The computation of an integral image can be done in one pass over the original image, for details see [49].

Figure 11 illustrates the calculation of an integral image for two example pixels $A(4, 8)$ and $B(10, 7)$. The computation produces a new image, as illustrated in Figure 12, where an original image showing a typical face-sample is transformed to its integral image representation. The resulting

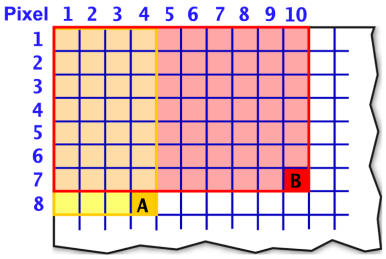

Figure 11: An integral image's pixel values are the sums of the pixel at this position, and all pixel values on the left and above. Thus, pixel A in the integral image representation gets the sum of all values that are included in the yellow rectangle, pixel B is the sum of the values in the red rectangle.

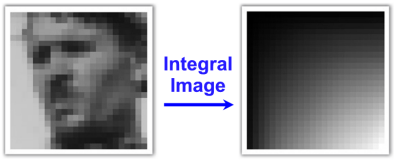

Figure 12: An Example showing the transformation of a normal gray scale portrait of size 24×24 pixels to its integral image representation. Note that the integral image's pixel values are mapped to the [0-255] gray scale color set for presentation purposes.

representation is similar to a gradient image: the lower right pixel is the sum of all pixels in the original image (all pixels are above and to the left of this position) whereas the upper left pixel has the lowest value, which is its original value (no other pixel is above or to the left of this pixel).

The benefit of the integral image representation is the possibility to calculate the sum of all pixel values in a rectangular area very fast and efficiently. Figure 13 demonstrates the calculation process.

With 4 array references we gather the values of pixels X, W, Y and Z. Since these pixels in the integral image represent areas in the original image (see Equations 1-4 in Figure 13), we can calculate area D as $D = Z+X-(Y+W)$. Consequently, we need 6 array references for the calculation of 2 adjacent areas and 8 array references for 3 adjacent areas, which are present in the above described feature types. Star-shaped features need 12 references.

Efficient calculation of the variance of an image for z-normalization. As described in Section 4.1, we preprocess the face candidate images by normalizing their variance. This can be done efficiently using integral images. We utilize the equation

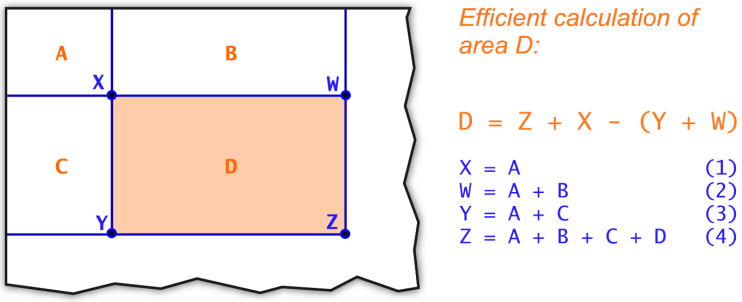

Figure 13: Efficient area calculation by only 4 array references and 3 additions, assuming an image representation as *integral image*. Pixels X, W, Y, Z need to be referenced and hold the values of area A, $A+B$, $A+C$ and $A+B+C+D$, respectively. Therefore, area D can be calculated by combining these pixel values to $D = Z + X - (Y + W)$.

$$s^2 = \frac{1}{n-1} \sum_{i=1}^{n} x_i^2 - \frac{n}{n-1} \overline{x}^2. \tag{17}$$

to compute the variance for z-normalization during preprocessing. We exploit the property of integral images, that the last value is the sum of all pixel values. Thus, the mean can be calculated by the single additional operation of dividing the sum by the number of pixels. Consequently, we utilize the squared image in the same way and use Equation 17 to calculate the variance from the two values with the appropriate coefficients.

Listing 1 shows an efficient implementation of the variance normalization process. It makes use of the integral image representation and Equation 17.

```
1  im is the input image (array) of size m × n
2  imSq is the input image squared element-by-element of size m × n
3  ii(im) is a function that computes the integral image represen-
4    tation of im
5  [] is the indexing operator, i.e. im[m,n] references the value in
6    row n and column m of image im
7  sqrt(v) is the standard square root function for a scalar v
8
9  Compute the integral image representations of im and imSq:
10   imI = ii(im)
11   imSqI = ii(imSq)
12 Compute the expected values of im and imSq, normalized by 1/(m*n-1):
13   exp = imI[m,n]/(m*n-1)
14   expSq = imSqI[m,n]/(m*n-1)
15 Compute the variance of im utilizing Equations 11 and 17:
16   variance = sqrt(expSq-(m*n)/(m*n-1)*exp*exp)
17 Divide im element-by-element by variance:
18   imVar = im / variance
19
20 save imVar which is the variance normalized image
```

Listing 1: Efficient variance normalization with integral images.

The presented algorithm is about 200 times faster than a standard implementation using the function std() in Matlab[3]. Additionally, during detection the image is not variance normalized directly. Instead, the variance is calculated and stored per subwindow. The calculated feature value (see Section 4.2.3) is divided by the variance in order to perform normalization. This yields equivalent results as normalizing subwindows directly. The proposed normalization of feature values instead of normalizing images leads to additional performance gains (1 floating point operation instead of $24 \times 24 = 576$ per subwindow).

[3] Tested on Matlab 2007b on a 2.6Ghz Pentium IV CPU with 24×24 pixel input images

4.2.4 Training a weak classifier

The first learning step in the proposed algorithm is the training of so called *weak classifiers*. The above described rectangular features are somewhat primitive in comparison to other features for face detection in literature, like steerable filters in [9]. Nevertheless, the training process shows, that even a very small number of those features - if selected carefully - can achieve a high detection rate (while few features cause bad false-positive rates, too). Thus, the weak classifiers, i.e. rectangular features and their threshold (and parity), are combined to form *strong classifiers* as described in the following section.

Weak classifiers are calculated once every iteration of the algorithm, serving as basis for a strong classifier that constitutes one stage in the detection cascade. Because of bootstrapping[4], which encourages independant errors among the classifier stages, a recalculation of all weak classifiers needs to be done every iteration. The calculation process is performed as depicted in Listing 2:

```
1  for every feature f_i
2    for every positive sample image P_j
3      calculate feature value v^P_{i,j} of the image
4    end
5    for every negative sample image N_j
6      calculate feature value v^N_{i,j} of the image
7    end
8
9    find median medP_i of positive feature values v^P_i
10   find median medN_i of negative feature values v^N_i
11
12   from = min(medP_i, medN_i)
13   to   = max(medP_i, medN_i)
14   minerr = Inf
15
16   for all i in [from, to]
17     calculate error of v^P_i, v^N_i and threshold i
```

[4] Bootstrapping means that the set of negative samples is updated in every iteration; for details see Section 4.2.7

```
18      if error < minerror
19        minerror = error
20        thresh = i
21      end
22    end
23
24    save thresh as threshold_i
25    save parity information
26  end
```

Listing 2: Calculation of thresholds for single features.

We first calculate the feature value for all positive and negative sample images, respectively, and compute the median $medP_i$ and $medN_i$. The optimal threshold lies between these boundaries, thus the error of each value between them is calculated. Consequently, the value i that makes the error minimal is selected as threshold for this feature. The parity information indicates, whether the positive sample's mean lies above or below the threshold.

Note, that a perfect separation is impossible for most setups, while not necessary either, since the best weak classifiers are combined with each other to minimize the ensemble error in a later step of training. We visualize the threshold determination with two histograms showing the values that positive and negative sample images score, respectively. The histograms of some of the best features are presented in Section 6.1 (Figure 24).

The final weak classifier h_j is given as

$$h_j(x) = \begin{cases} 1 & \text{if } p_j f_j(x) < p_j \theta_j \\ 0 & \text{otherwise} \end{cases} \tag{18}$$

where f_j is the classifier's feature, θ_j its threshold and p_j the parity, indicating the direction of the inequality sign. Here, x may be a 24×24 pixel sub-window of an image in the detection process or a 24×24 pixel sample image in training, respectively.

4.2.5 Forward Feature Selection versus AdaBoost

Given the large feature set $F = \{f_i\}$ and the corresponding set of weak classifiers $\{h_i\}$, the next step is to select a reasonable number of weak clas-

sifiers $\{h_t\}$ and combine them to an ensemble H in a way that maximizes detection- and false-positive-performance. While each feature can be computed very efficiently, the computation of all features for each possible face window is prohibitively expensive in terms of calculation time (remember, we are dealing with tens of thousands of features and even larger potential face window sets in films). Thus, a greedy feature selection method is needed that selects a small (compared to the initial size) set of weak classifiers where individual classifiers are nevertheless significantly different from each other.

Viola and Jones employ the AdaBoost algorithm proposed by Freund and Schapire in [6] in their original work for feature selection. They utilize a variant of AdaBoost for both, selecting the features and training the ensemble classifier. Based on the weak classifiers, where even the best one is not expected to classify the training images well, AdaBoost tries to combine them into a *strong* classifier which meets the learning goal. The approach is as follows. Each weak classifier solves a sequence of classification problems and, after each round, the samples are reweighted such that false classified images have a higher priority in further iterations. The final strong classifier is a weighted combination of weak classifiers. While being effective in selecting good features and weighing them, this approach has a major drawback. The calculation is extremely computational expensive, as demonstrated in Section 6.3. This is because it maintains and evolves a distribution of weights over the complete training data set which is not only computational expensive itself, but requires retraining for the complete set of classifiers at each iteration.

Thus, another method has been chosen in this work. The combination of Viola and Jones' algorithm with the *forward feature selection* (FFS) heuristic was proposed by Wu et al. in [53]. It benefits from the idea to decouple the feature selection process from the design of an ensemble strong classifier. We employ forward feature selection as in [52], which is a more recent version of the original method. The ensemble classifier design, i.e. the weight setting for selected features, is presented in Section 4.2.6.

Additionally to focusing only on the selection of the best features, a major advantage of FFS is that it doesn't need to evolve a learning distribution over the complete training set. This leads to dramatic improvement in

computation time as a retraining of weak classifiers for each selection is not necessary any more. However, this comes at the cost of calculating a large binary classification table V_{ij} once for the selection of T features for each stage of the cascade of classifiers. The table gives the score of each weak classifier h_i for each of the N sample images x_j, i.e. $V_{ij} = h_i(x_j)$. Once calculated, the table is stored for utilization during the process of feature selection and discarded when the selection process of features for the particular stage has completed.

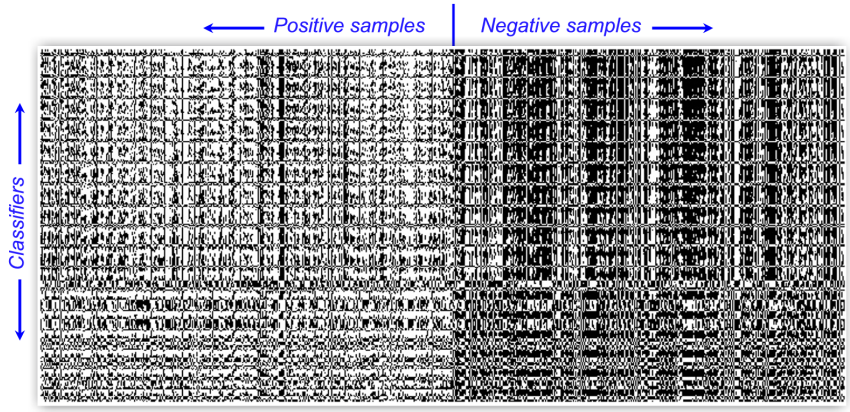

Figure 14: Fraction of binary classification table V showing results of weak classifiers. Training images are in x-, classifiers in y-direction. Thus, $V_{ij} = h_i(x_j)$.

Figure 14 presents a small fraction of the table where each pixel corresponds to a table cell V_{ij}. Each classifier (rows) classifies each input sample (columns). Positive sample images are located on the left half of the table, negative ones on the right side. Black pixels indicate that the classifier classified the sample as non-face while white pixels indicate that the classifier detects a face. While it is obvious, that single classifiers cannot separate the input images perfectly, it is intuitively clear, that the left side of the table has an overall lighter appearance (the positive sample images that show faces are located here). Also, the evolving patterns when scanning classifier lines are interesting to analyze. Remember, a *perfect* classifier would score all images on the left half *true* (white) and all on the right half *false* (black). Furthermore, problematic training samples can be detected (vertical lines in the table).

The selection of T features assumes a feature set F of size M with corresponding thresholds θ_i such that each classifier has the smallest error on the training set. During selection of the ensemble set S (initialised to $S = \emptyset$), the addition of each possible classifier to the set is considered. Therefor, a vector $v_{1\times N}$ is maintained, that is initiatlised to a row vector of zeros. When the addition of classifier h_i is considered, a temporary v' is calculated as $v' = v + V_i$. Thus, v' contains integers in the range 0 to t, i.e. the round index. By building a histogram of the temporary vector v', an optimal value τ for the ensemble threshold which minimizes the ensemble's error rate is easily found. The value τ serves as threshold for the vote of all up to now selected classifiers. Thus, an image is classified as *face* if more than τ of the classifiers in this stage vote positively. Note, that instead of θ which is used in [52], we use the letter τ for the ensemble's threshold to avoid confusion with the individual classifier thresholds θ.

The error rate ϵ_i of each temporary set of different weak classifiers is stored and once all classifiers have been analyzed, the one that takes the minimum value is selected and added to set S. If the desired number of selected classifiers is not yet reached, the process is taken into another round with an updated v and S. Once the method has finished, the ensemble's threshold τ is adjusted such that the ensemble classifier H has a 50% false-positive rate. Summarizing, we present the above described procedure in Listing 3:

```
1  h_i are the classifiers
2  x_j are the n sample images
3  T is the number of features of this cascade stage
4  V_ij = h_i(x_j) is a large binary classification table
5  v = 0_{1×N} is a row vector of zeros
6
7  for t = 1 to T
8    for every classifier h_i
9      consider adding V_i to v: v' = v + V_i
10     the ensemble's value H'(x_i) can be computed as H'(x_i) = sgn(v'_i − τ)
11     find τ ∈ {0,...,t} that minimizes the set's error rate ε_i
12     save ε_i
13   end
14   select classifier h_k with smallest ε
```

```
    add h_k to S and update v: S = S ∪ h_k and v = v + V_k
end
save ensemble classifier H(x)
adjust value of τ such that H has a false-positive rate of 50%
save ensemble threshold τ
```

Listing 3: Forward feature selection as suggested by Wu et al. [52].

The increase in training performance by forward feature selection is described in Section 6.3, the mathematical representation of the combined (strong) classifier is presented in the following section.

4.2.6 Weight setting for the strong classifier

Weight setting is the last step in forming a strong classifier and is performed after the features have been selected. The idea is to give the best classifiers in an ensemble more importance, thus assigning a larger weight. There exist a large number of weight setting algorithms but only few are suited for the asymmetric classification task of face detection. Remember, we are dealing with three kinds of asymmetries as described in Section 1: first, we are dealing with *uneven class priors* (from the large number of input windows during detection, i.e. potential face windows, only few contain faces). Second, there exist a significant *goal asymmetry*. A very high detection rate is needed (no face should be missed), while the overall false-positive rate needs to be extremely low (e.g. 10^{-7}) to avoid a flood of non-events. Third, there is an *unequal complexity* between the two classes *face* and *non-face* as the positive class is well defined ('face') while negatives could virtually be everything, except a face (e.g. nature, buildings, animals, patterns,...).

While bootstrapping takes care of the unequal class complexity and uneven class priors (see Section 4.2.7), the nature of the used classification cascade alleviates the goal asymmetries (i.e. goals for single classifiers of the cascade are much easier to achieve than the overall rates mentioned above). However, within a cascade stage i that features learning goals for detection rate $d_i \approx 99.9\%$ and false-positive rate of $f_i \approx 50\%$, an asymmetric weight setting function, that gives more importance to positive (face) samples, is favored. While AdaBoost - decoupled from feature selection -

could be used for the task, it does not explicitely take the asymmetries into account. Therefore, other methods like assigning higher initial weights to positive samples with a variation of an algorithm called Rocchio have been proposed by Schapire et al. for text filtering in [36]. However, it has been argued by Viola and Jones, that these weights get quickly absorbed by the boosting algorithm. They proposed AsymBoost instead in [50] but here, again, the problem of selecting features and designing ensemble classifiers (i.e. weight setting) are conflated which leads to unnecessary performance losses.

In [52], Wu et al. present a variation of Linear Asymmetric Classifier (LAC) for the weight setting task, that can efficiently be decoupled from the feature selection process and takes care of the goal asymmetries. When the sign function sgn is used to label positive and negative classifications, the ensemble classifier $H(x)$ that was calculated by FFS can be written as

$$H(x) = sgn(\sum_{t=1}^{T} a_t h_t(x) - b) = sgn(a^T h(x) - b) \tag{19}$$

where T is the number of weak classifiers in this stage and $h_t(x)$ is a weak classifier's output. Thus, the ensemble classifier is defined by its weak classifiers and the value pair (a, b), a being a $1 \times T$ weight vector and b being a scalar (offset). However, there is no guarantee, that the values (a, b) selected by FFS (here, $a = 1_{1\times T}$ and $b = \tau$) or AdaBoost satisfies the learning goal of maximizing the detection rate given a false-positive rate of 0.5.

Wu et al. argue, that this guarantee can be achieved by applying a set of equations on the mathematical expression of the above described learning goal. By projecting the standardized positive and negative class labels, expressed as a vector, onto the direction of weight vector a and under certain assumptions (i.e. the projection of positive samples is Gaussian and the mean and median of the projection of negative samples are similar), they conclude that efficient calculation of (a, b) can be performed as follows. First, mean and covariance of positive samples x and negative samples y are estimated (n_x and n_y being the number of samples). The experimental means $\overline{x}$ and $\overline{y}$ are given as

$$\overline{x} = \frac{\sum_{i=1}^{n_x} h(x_i)}{n_x} \qquad (20) \qquad\qquad \overline{y} = \frac{\sum_{i=1}^{n_y} h(y_i)}{n_y} \qquad (21)$$

and the experimental covariance matrices Σ_x and Σ_y can be calculated as

$$\Sigma_x = \frac{\sum_{i=1}^{n_x}(h(x_i) - \overline{x})(h(x_i) - \overline{x})^T}{n_x} \qquad (22)$$

$$\Sigma_y = \frac{\sum_{i=1}^{n_y}(h(y_i) - \overline{y})(h(y_i) - \overline{y})^T}{n_y} \qquad (23)$$

Now, under the condition that Σ_x is positive definite[5] (i.e. all eigenvalues λ_i of the matrix are positive and its inverse exists), the value pair (a, b) can be calculated as

$$a = \Sigma_x^{-1}(\overline{x} - \overline{y}), \qquad (24) \qquad\qquad b = a^T\overline{y} \qquad (25)$$

where Σ_x^{-1} is the inverse of the covariance matrix Σ_x.

In cases where the matrix is not positive definite, we induce this property by adding a small λ to all matrix elements on the diagonal, thus making the matrix invertible. This has been suggested by Wu and can be performed when handling a positiv semidefinite matrix. As this is a property of all covariance matrices, we replace the $n \times n$ covariance matrix Σ_x with $\Sigma_x + \lambda Diag_n$ where $Diag_n$ is an $n \times n$ matrix with elements on the main diagonal 1 and 0 otherwise.

In this work, the Fisher Discriminant Analysis (FDA) was also implemented. Here, the weight vector a is calculated as $a = (\Sigma_x + \Sigma_y)^{-1}(\overline{x} - \overline{y})$. However, Wu et al. showed, that LAC, as it takes the goal asymmetries into account, provides in most cases better values for a and b than AdaBoost,

[5] Σ_x is positive definite if the training set is sufficiently large, i.e. the number of variables (selected classifiers) must be significantly smaller than the number of positive and negative sample images and the (classification) vectors must be linearly independent from each other.

FFS, FDA and other linear weight setting methods. For details on statistical and mathematical background see [52].

We complete the weight setting step with a novel control process of the achieved false-positive and detection rates. The rates after weight setting (with LAC or FDA) are determined with an independent test image set of 1000 positive test images and the set of negative training images. The set of negative images for the calculation of the false-positive rate is the training set itself (otherwise, the goal of each stage to achieve a 50% false-positive rate on the training set could be corrupted). The control mechanism works as described in Listing 4.

```
1  Detection rates are obtained from independent test set (1000 test
2  images), false-positive rates from set of negative training images
3
4  det1, fp1 are detection- and false-positive rate after FFS
5  det2, fp2 are detection- and false-positive rate after weight setting
6  a1, b1 are weight and threshold after FFS
7  a2, b2 are weight and threshold after weight setting
8
9  a2, b2 are currently selected as weights by default
10
11 if (fp1 <= fp2) or (fp2 <= fp1 <= 50)
12   if (det1 >= det2)
13     save a1, b1 as weights, discard a2, b2
14   end
15 end
```

Listing 4: Control mechanism of detection- and false-positive rate after weight setting.

As shown above, we select the weights $a1$, $b1$ both if the first false-positive rate is smaller than the second and the first detection rate is larger than second and in the case that the first false-positive rate is larger than the second but smaller than 50% (after all, that was the training goal) while the first detection rate is larger than the second. Thus, the two rates are in the first case both *better* (the false-positive rate is lower, the detection rate larger) and in the second case we maximize the detection rate while keeping a false-positive rate of less than 50%.

This monitoring is especially important as we cannot guarantee optimal weights in terms of resulting detection and false-positive rate when the covariance matrix was positive semidefinite (here, as described above, we added a small positive number to the diagonal elements of the matrix to make it invertible).

4.2.7 Bootstrapping

Boostrapping is the process of updating the set of negative training samples. It was introduced by Sung and Poggio to find difficult non-face samples that are hard to separate from faces [44]. The process of bootstrapping is described in Listing 5: Once a new cascade stage and the respective weights are calculated, run the current cascade on the set of negative sample images. Keep only the false-positives and *extend* the set with new non-face samples that, also, are classified incorrectly as *face* until the original set size is attained.

```
1  D is a set of video frames that contain no faces
2  N = {N_i} is the set of n negative samples
3  H = {H_j} is the current cascade of m stages
4
5  run H on N:
6    remove all correctly classified samples from N
7  while (N is not complete)
8    generate new negative sample N_new from D
9    if N_new is not classified correctly by H
10     add new sample to N: N = N ∪ {N_new}
11   end
12 end
```

Listing 5: Boostrapping after a new cascade stage has been calculated.

There are three reasons for the necessity of this updating procedure. First, we are dealing with a very large and rich set of non-face samples. Virtually anything but a face can be in this class. Thus, the use of a broad training set is adequat. Second, as mentioned earlier, we want the cascade stages to make independent errors in order to be able to achieve the training goal with the cascaded architecture. And third, as the cascade stages become

more complex (i.e. ensemble classifiers consist of more weak classifiers), the training set has to become more difficult to classify, in order to maximize the detection and false-positive performance of the ensembles. Note that the first two reasons comply with the earlier mentioned asymmetries: we have uneven class priors (thus, we keep the hardest negative samples, i.e. samples that are false-positives) and unequal complexity of the training sets (we try to gather as many new negative samples as possible).

Once bootstrapping has been performed, the training process for one stage has finished and the algorithm is taken into another round until the termination goal is met.

4.3 Detection

The detection process works at a base resolution of 24×24 pixels. When scanning a video frame or still image, the goal is to classify subwindows of this size individually by the cascade. The detector scans over the input frame, thus creating subwindows and locating faces at different positions. The cascade is evaluated at all positions and if the subwindow is classified as being a face, its coordinates are stored. Once the frame is scanned at the starting resolution, the detector is scaled and scans the image again to detect faces of different sizes. The scaling process works in constant time and is repeatedly performed up to a maximum resolution.

Once detection has finished, the process is concluded by merging overlapping detections. Because of small x- and y-offsets, similar scales and some variability in the positive training images, one face might be detected several times at slightly different positions. The following two sections present the scanning and scaling of detectors and the merging of overlapping detections.

4.3.1 Scaling and scanning the detector

The detection cascade works, as indicated before, on 24×24 pixel subwindows. Thus, the features that are stored in the cascade with their appropriate individual thresholds τ, weights and ensemble threshold θ per stage, may only be evaluated on subwindows that have the same size as the training samples. Remember, a feature consists basically of a set of coordinates

indicating its position and size and these coordinates are relative to the size of the training subwindows.

Clearly, during detection the goal is to detect faces at all scales and positions. The latter is achieved by shifting the detection subwindow over all possible positions in the image. As this is somewhat computation intensive (for small detection windows, the number of possible positions roughly corresponds to the number of pixels in the image which generally is large), we introduce step sizes dx and dy to reduce the number of subwindow positions. When dx and dy are small relative to the subwindow, this doesn't affect the detection accuracy as there is some variability of the face locations in the training samples, as well.

The detection of faces of different sizes is more complex. One straight forward approach is to make use of image pyramids. Here, the original image is subsampled repeatedly by a scaling factor. Consequently, the detector in its original resolution of 24×24 pixels is scanned among the subsampled images. As the subwindows become smaller, the relative size of the detector increases, thus it can detect larger faces at the cost of creating the image pyramid and scanning the detector repeatedly.

However, due to the nature of the features, they can be resized and evaluated at constant cost. While repeated scanning of the differently sized detectors is still necessary, the calculation of an image pyramid is not required. Thus, the approach is necessarily more efficient than the image pyramid method.

The scaling of the detector is shown for a 2-area horizontal feature (top-bottom split, see Figure 9 in Section 4.2.3) in Listing 6.

```
1  The feature is characterized by the following variables:
2  x and y are the coordinates of the upper left corner
3  w is the width of a feature cell
4  h is the height of a feature cell
5  θ is the feature's threshold
6
7  s is the current scaling factor
8
9  The values x, xw = x + w, y, yh = y + h and yhh = y + 1 + h + h are used
10 during feature evaluation. These are scaled in the following way:
```

11 $x = x \cdot s - (s-1)$

12 $xw = xw \cdot s$

13 $y = y \cdot s - (s-1)$

14 $yh = yh \cdot s$

15 $yhh = yhh \cdot s$

16

17 `The threshold is scaled by the factor` s^2`:`

18 $\theta = \theta \cdot s^2$

Listing 6: Scaling a 2-area horizontal feature (top-bottom split).

Note, that the scaling is different for values containing initial positions (x, y) and values evolved from these positions (xw, yh, yhh). Thus, x and y are scaled by the factor s and offset $-(s-1)$ while xw, yh and yhh are scaled by the factor s without offset. Note also, that the threshold is scaled by s^2. This is necessary, as s is applied to the feature size in two dimensions (x direction and y direction), making a feature s^2 times larger (i.e. a feature scaled by $s = 2$ is 4 times larger than the original one).

The scaling process is depicted in Figure 15 for a sample scale factor $s = 2$ and the sample type-1 feature with $x = y = 2$, $xw = 3$, $yh = 2$ and $yhh = 3$. The left side of the figure shows the original and the right side the scaled feature. Note that similar calculations allow for scaling of features of other types (e.g. 3-area features or the star-feature), as well.

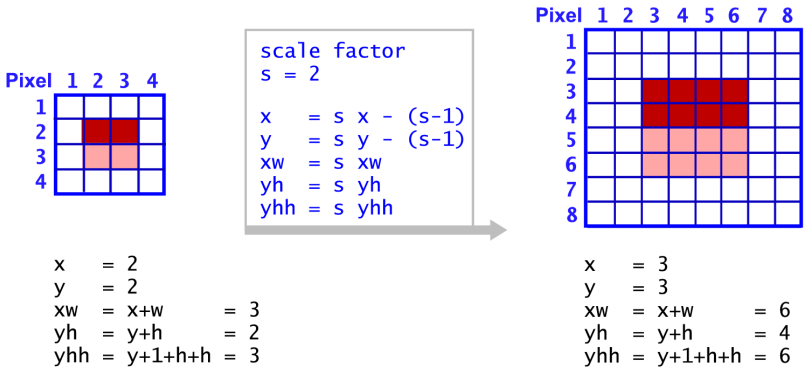

Figure 15: Scaling a type 1 feature. The small feature on the left is scaled by the factor 2. x, y, xw, yh and yhh of the larger feature are updated according to the scaling rules (blue).

4.3.2 Merging detections

The before described scanning and scaling of the detection window may result in multiple detections of one face. This is due to some variability in the training images. For example, a face might be detected at positions (x, y) and at $(x + \epsilon_1, y + \epsilon_2)$ where ϵ_1 and ϵ_2 are small offsets in the x- and y-direction, respectively.

Thus, we introduce a method that merges detection rectangles with the goal to output results of higher quality compared to the naive output of all detections. The procedure works at two levels. The first level operates on all detected face candidates that have the same size (i.e. detected at the same scale factor). Here, rectangles are merged that overlap in x- and y-direction for a certain amount. The resulting window is defined by averaging the corners of merged detections. The second level operates on the results of level one and merges rectangles of different scales. Again, the rectangles need to overlap for a certain amount in either direction. Smaller rectangles are merged if they lie inside a bigger detection rectangle. However, small shifts to either side is also accepted. The corners of the resulting rectangle are not averaged but weighted in favor of bigger rectangles. This is done due to the observation that bigger detections tend to be more precise.

Figure 16 illustrates the merging operations. The merging operation at the same scale is depicted on the left side (green original detections, blue merged detections) and the operation at different scales on the right side (blue input rectangles from the first merging stage, red results). The red results are returned as final detections.

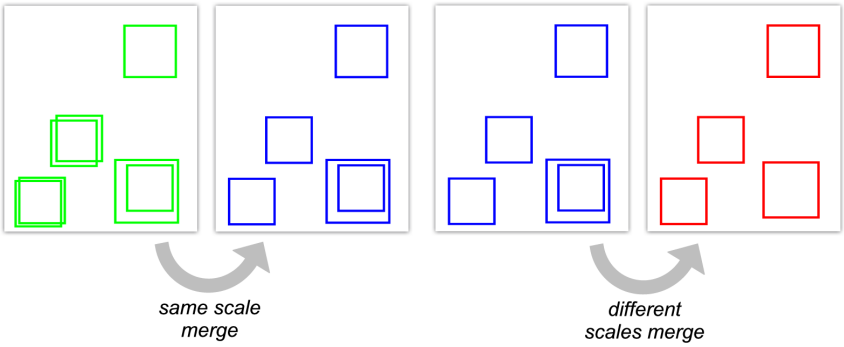

Figure 16: Merging detections on two levels. First (left side), two or more overlapping detections of the same scale are merged. Second (right side), detections at different scales are merged. Thus, the initial detections (green) are first transformed to intermediary (blue) and then to final results (red).

4.3.3 Post processing with a symmetry filter

Due to the partly bad quality of the material at hand, the detection algorithm yields more false-positives than expected. We introduce a postprocessing step that is built upon a symmetry filter. The filter is inspired by the observation that while frontal faces are generally symmetric, a large number of false detections is not.

The algorithm first reduces the number of gray values of a candidate image and removes the background (i.e. the value with the most pixels). Then, three vertical symmetry axes are constructed: one through the center of the image, one slightly shifted to the left and one slightly shifted to the right. The idea of multiple symmetry axes is motivated by the obervation, that not all candidate face windows are perfectly fitted around a centered face. Thus, three slightly shifted axes are constructed and the maximum value of symmetric pixels with respect to either axis is the symmetry score of that face candidate. Consequently, an image is discarded if it has less symmetric pixels than a threshold demands. With symmetric pixels we mean pixels, that have a pendant pixel mirrored with respect to the symmetry axis, having the same gray value. Listing 7 summarizes the filter.

```
1  H is a candidate face image of size m × n
2  c is the number of gray values to reduce complexity
3  f is a percentage of the image width
4  θ is a threshold (minimum percentage of symmetric pixels)
5
6  reduce the number of gray values of H to c levels
7  remove the intensity with the most pixels
8
9  for each gray value i
10   s1 is the number of i-pixels that are symmetric with respect to
11     a vertical symmetry axis through the center of the image
12   s2 is the number of pixels that are symmetric w.r.t. a vertical
13     symmetry axis through the center + f · width of the image
14   s3 is the number of pixels that are symmetric w.r.t. a vertical
15     symmetry axis through the center − f · width of the image
16   sum1 = sum1 + s1
17   sum2 = sum2 + s2
18   sum3 = sum3 + s3
19 end
20
21 score =max(sum1, sum2, sum3)/(m · n)
22
23 if score > θ
24   do not filter the image (assume a face):
25   return 1
26 else
27   filter the image (assume a false-positive):
28   return 0
29 end
```

Listing 7: Post processing with a symmetry filter.

The value for the threshold has been determined empirically. Thus, a test set of 3500 positive (face) and 2000 negative (non-face) images has been evaluated with the symmetry filter. The scores of the images are depicted in two histograms in Figure 17. The histogram at the top shows the scores of the positive samples, the histogram at the bottom the scores of negative samples. It shows that a significant number of negative samples has summetry values at around zero which is a result of homogeneous samples where almost the entire image gets removed by the background culling procedure.

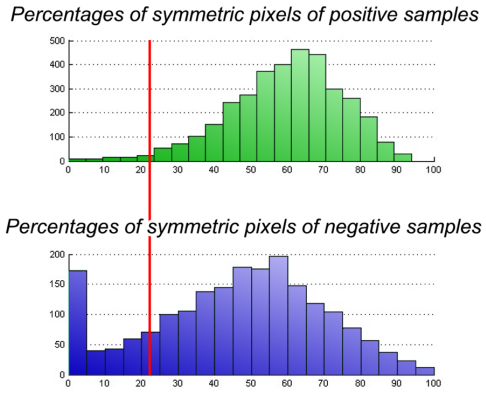

Figure 17: Results from symmetry filter test setup. The histogram at the top shows the distribution of symmetric pixel per image in per cent for 3500 positive samples, the histogram at the bottom for 2000 negative samples. The red line indicates a possible value for the threshold at 22.

Obviously, a threshold somewhere between 20 and 35 percent results in a significant reduction of false-positives while only few faces get removed. Table 6 depicts the number of discarded positive and negative samples in absolut values and in percent for different sample thresholds.

Table 6: Sample thresholds (percentage of symmetric pixels) for the symmetry filter. Test setup: 3500 positive samples, 2000 negative samples, 8 intensity values, 3 symmetry axes ($center - 1/10 \cdot width$, $center$, $center + 1/10 \cdot width$).

Threshold	**Discarded positives**	**Discarded negatives**
0.20	55 of 3500 (1.57%)	313 of 2000 (15.65%)
0.23	71 of 3500 (2.03%)	355 of 2000 (17.75%)
0.28	123 of 3500 (3.51%)	440 of 2000 (22%)
0.32	185 of 3500 (5.29%)	524 of 2000 (26.2%)
0.35	229 of 3500 (6.54%)	590 of 2000 (29.5%)

In applications where the reduction of false-positives is the main goal, the threshold can be set to a higher value (for example, a threshold of 0.35 reduces the number of false-positives by almost 30 percent). However, high thresholds also reduce the detection rate significantly. A value of 0.20 keeps

most of the true detections while still reducing the false-positive rate by over 15 percent.

4.3.4 Post processing with a connectivity filter

As an additional postprocessing step, we introduce a connectivity filter. The use of a connectivity filter is motivated by the observation that faces show significant texture while some false-positives have little or no texture. The filter calculates connectivity scores for pixels of different intensities of a candidate image. It deduces two values from the scores: median and maximum values of the connectivity scores of different intensities. The measures have been analyzed and are matched against a threshold area during the filtering process.

The algorithm first reduces the number of intensities of the candidate image to c intensities. Then, the connectivity score is calculated for each gray value independently. The score is the sum of all 8-neighborhoods of each pixel. After a normalization with the image size, the connectivity scores of all intensities are stored and the *maximum* and *median* of these values are derived. These two measures are then matched against threshold boundaries. The algorithm is summarized in Listing 8.

```
1  H is a candidate face image of size m × n
2  N is the connectivity kernel [1,1,1;1,0,1;1,1,1]
3  c is the number of intensities to reduce complexity
4  med1, med2 are the median thresholds
5  max1, max2 are the maximum thresholds
6
7  reduce the number of intensities of H to c levels
8
9  for each gray value i:
10   Htemp is the image with all pixels of intensity i set to 1, all
11     others to 0
12   C is the convolution of Htemp and N
13   multiply C and Htemp elementwise
14   connectivity(i) is the sum of all values in C × Htemp
15   normalize the connectivity of intensity i:
16     connectivity(i) = connectivity(i)/(m · n)
17 end
```

```
18
19 med = median(connectivity)
20 max = maximum(connectivity)
21
22 if (med1 < med < med2) and (max1 < max < max2)
23   do not filter the image (assume a face):
24   return 1
25 else
26   filter the image (assume a false-positive):
27   return 0
28 end
```

Listing 8: Postprocessing with a connectivity filter.

Values for the thresholds have been determined empirically. Table 7 presents two sample pairs for upper and lower bounds for the median and maximum values. The table also summarizes the quality of the filter with either of the threshold pairs. Again, the test setup consists of 3500 face images and 2000 negative sample images.

Table 7: Sample thresholds (lower and upper bounds) for connectivity filter. Test setup: 3500 positive samples, 2000 negative samples, 8 intensity values.

Thresholds for median	**Thresholds for maximum**	**Discarded positives**	**Discarded negatives**
$0.0 < \text{med} < 1.7$	$0.5 < \text{max} < 4.0$	80 of 3500 (2.29%)	674 of 2000 (33.7%)
$0.1 < \text{med} < 1.4$	$0.5 < \text{max} < 3.2$	285 of 3500 (8.14%)	953 of 2000 (47.65%)

The table shows that by reducing the detection rate by only 2.29%, the false-positive rate can be reduced significantly by over 33%.

Figure 18 shows a plot of *median* and *maximum* plotted against each other for the set of test images. Green dots indicate the (median, maximum) value pairs of positive sample images, blue dots the values for negative samples. The threshold pairs suggested in Table 7 are represented as two red rectangles where the smaller constitutes a tighter selection (all images that score outside a threshold area are classified as being a non-face).

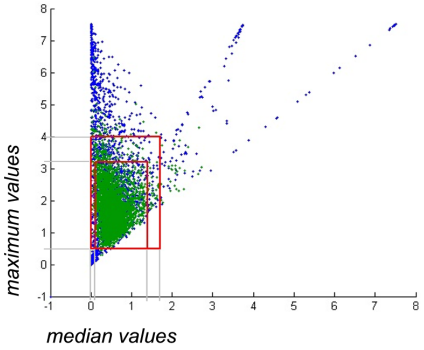

Figure 18: Results from connectivity filter test setup. Green dots indicate (median, maximum) value pairs for positive samples, blue dots for negative samples. The two red rectangles are the sample threshold areas given in Table 7. Thus, all candidates that score outside a specific threshold area are discarded by the filter.

4.3.5 Combined post processing filters

We combine the two previously described filters (symmetry filter and connectivity filter) to achieve a better trade off between detection rate and false-psoitive rate. The thresholds are selected after combined testing of the filters. We provide experimental results in Table 8.

Table 8: Sample thresholds for combined filters. Test setup: 3500 positive samples, 2000 negative samples, 8 intensities. Symmetry and connectivity filter as described in Section 4.3.3 and 4.3.4. The filters are sequentially combined.

Symmetry threshold	**Connectivity threshold *max***	**Connectivity threshold *med***	**Discarded positives**	**Discarded negatives**
0.20	[0.5;4.0]	[0.0;1.7]	131 (3.74%)	774 (38.7%)
0.20	[0.5;3.2]	[0.0;1.5]	326 (9.31%)	1007 (50.35%)
0.23	[0.5;4.0]	[0.0;1.7]	147 (4.20%)	794 (39.7%)
0.23	[0.5;3.2]	[0.0;1.5]	341 (9.74%)	1023 (51.15%)
0.28	[0.5;4.0]	[0.0;1.7]	199 (5.69%)	844 (42.2%)
0.28	[0.5;3.2]	[0.0;1.5]	390 (11.14%)	1058 (52.9%)

Table 8 shows, that the two filters operate mainly independent for the set of positive images. The scores 0 ('non-face') and 1 ('face') for the set

of negative images overlap to some extent. Thus, the number of discarded positive images of the combined filter is roughly the sum of the discarded positive images of the individual filters with appropriate thresholds. However, the combined results for the negative samples are better than for the individual filters. The number of discarded negative images is significantly less than the sum for the individual filters.

For example, the scores for discarded positives and negatives for a symmetry filter with threshold 0.20 are 1.57 and 15.65, respectively. The scores for a connectivity filter with maximum thresholds $[0.5; 4.0]$ and median threshold $[0.0; 1.7]$ are 2.29 and 33.7 percent. The combined filters yield scores of 3.74 percent of discarded positives (roughly the sum of the individual filter's scores) and 38.7 percent of discarded negatives (less than the sum).

Note, that the filter results are obtained from random sample image sets. The filter performance may vary for a particular setup, possibly being lower than indicated above. However, the filters still ameliorate the false-positive rate as we will see in Section 6.

5 Experimental setup and training

This section presents the setup for the training of the detector and for the detection and evaluation process. Section 5.1 summarizes the employed hardware and software, Section 5.2 describes the setup and data structure for the detection cascade and how to access it once it is calculated. Section 5.3 gives an overview of the training sample images and Section 5.4 describes the ground truth for the detector evaluation. The setup is summarized in Section 5.5.

5.1 Platform

The system used for training and detection has a Pentium IV CPU with 2.4 Ghz. The main memory is 1024 MB. The installed operating system is *Windows XP Professional* with *Service Pack 3*.

The code is completely written in *Matlab 2007b* (Version 7.5.0). Additionally to the standard Matlab installation, we use the *Image Processing Toolbox* and the *XML Toolbox* [25] which is used to write and load the data structure that holds the cascade.

Because of the large memory requirements, especially for table V in forward feature selection (see Section 4.2.5), two memory tweaks have been performed prior to training the cascade. First, Windows' swap file has been set to larger maximum values. Second, Windows has been booted with the 3GB switch enabled. Therefore, the "/3gb" switch has been added in the boot.ini. The new line in boot.ini should read something like:

```
multi(0)disk(0)rdisk(0)partition(1)
WINDOWS="XP with 3GB switch" /noexecute=optin /fastdetect /3gb
```

After a restart of the system, the new boot option should be available, giving Matlab more memory for workspace variables (this can be confirmed with a call of 'system_dependent memstats' in Matlab). The tweaks might not be necessary if enough random access memory is available or if the training set is sufficiently small. The performance evaluations that are described in Section 6.3 are based upon the here described system.

5.2 Data structure of the cascaded classifier

This section gives an overview of the internal structure of the cascaded classifier. First, the design of the cascade is described, then we present an example xml file that contains a cascade. Finally, a few lines of Matlab code show how to load the cascade and access different fields.

The classification cascade as described in Section 4.2 consists of a number of stages that contain classifiers and an ensemble threshold. Each classifier consists of various fields that describe its features, its weight, its threshold and its parity information. Figure 19 shows the basic structure of the classification cascade.

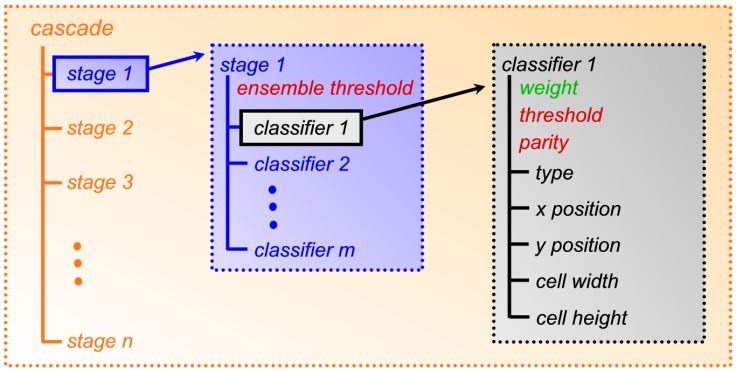

Figure 19: Structure of the cascade classifier. The cascade consists of n stages. Each stage has an ensemble threshold and m weak classifiers. Each classifier consists of a weight and a threshold with parity. Additionally, it has fields for the specification of the underlying feature: the type, x- and y position, width and height of a feature cell.

Internally, the cascade is represented as a Matlab structure during the training. It is exportet to an xml-file once a new stage is calculated. Listing 9 shows the xml-representation of the first two classifiers of cascade stage 2. Note that most of the structure shown in Figure 19 is represented explicitely in this xml representation. Only the weights for each classifier are stored as a vector per stage rather than a field per classifier for performance reasons. The two values stored in the <threshold> entities are the actual threshold (first value) and the parity of this classifier (second value).

```
...
<item idx="2" type="struct" size="1 15">
  <weights idx="1" type="double" size="15 1">3.582770825619097
    0.4396331992625959 0.296199974237249 2.372890570201271
    1.256582464366535 2.855441663672872 0.6264357607983953
    0.559260965261162 2.446557310591112 0.2263569285381752
    1.940773745846374 -0.05608546586182456 2.051127126815692
    0.497754834200101 0.3310193786694287</weights>
  <thresh_b idx="1" type="double" size="1 1">8.11317009927721</thresh_b>
  <classifiers idx="1" type="struct" size="1 1">
    <feature idx="1" type="struct" size="1 1">
      <type idx="1" type="double" size="1 1">4</type>
      <posx idx="1" type="double" size="1 1">5</posx>
      <posy idx="1" type="double" size="1 1">2</posy>
      <cellwidth idx="1" type="double" size="1 1">5</cellwidth>
      <cellheight idx="1" type="double" size="1 1">3</cellheight>
    </feature>
    <threshold idx="1" type="double" size="1 2">-8 1</threshold>
  </classifiers>
  <classifiers idx="2" type="struct" size="1 1">
    <feature idx="1" type="struct" size="1 1">
      <type idx="1" type="double" size="1 1">3</type>
      <posx idx="1" type="double" size="1 1">7</posx>
      <posy idx="1" type="double" size="1 1">2</posy>
      <cellwidth idx="1" type="double" size="1 1">3</cellwidth>
      <cellheight idx="1" type="double" size="1 1">7</cellheight>
    </feature>
    <threshold idx="1" type="double" size="1 2">-33 0</threshold>
  </classifiers>
  ...
```

Listing 9: Snippet of final classification cascade which covers the first two classifiers of stage 2.

Before detection starts, the structure is converted into a three-dimensional array for performance reasons. This *simplified cascade* is stored as xml-file, too. Thus, the access to different parts of the cascade in Matlab are standard array operations. We show how to load the cascade, determine its size, extract a certain classifier and access its fields and how to access the ensemble threshold of a certain stage in Listing 10.

```
% load the simplified cascade from xml-file
cascade = xml_load('simple-cascade.xml');

% number of classifiers per stage, fields per classifier and stages
no_classifiers = size(cascade,1);
no_fields = size(cascade,2);
no_stages = size(cascade,3);

% load classifier 3 of stage 5
classifier = cascade(3,:,5);

% load the fields of this classifier
weight = classifier(1);
theta  = classifier(2);
parity = classifier(3);
ftype  = classifier(4);
posx   = classifier(5);
posy   = classifier(6);
width  = classifier(7);
height = classifier(8);

% load ensemble threshold of stage 5
threshold = cascade(1,1,5);
```

Listing 10: Accessing parts of the cascade.

5.3 Training Images

The described algorithm relies heavily on training images. Two sets are required: *face images* (positive samples) and *non-face images* (negative samples). Both sets have to be sufficiently large to produce a useful cascade. The detector presented in this work has been trained with approximately 2×3900 sample images per cascade stage. While the set of faces remains the same for subsequent stages, the set of non-faces is updated after each stage by a bootstrapping procedure (see Section 4.2.7). All training images have a resolution of 24×24 pixels. While the decision on the training set of negative sample images was straight forward, the training set of positive samples caused some discussion.

Positive Sample Images: The sample database has been constructed from both, the exhaustive set (about 4,000 images) of the Viola and Jones samples, see [48], and of hand-picked samples from the Vertov film material. An exclusive use of the latter was discussed but considered as not being suitable due to the time-consuming selection process. Using only faces of the films of Vertov would have resulted in a much smaller face database. 200 sample faces were selected from the films by hand and stored along with their flipped counterparts, adding up to a set of 400 faces. Together wih 3,500 images from the Viola-Jones database, this sums up to 3,900 positive training images. Figure 20 presents examples of face images. Note that the blurred appearance is a result of the enlargement of the 24×24 pixel training samples for presentation purposes.

Figure 20: Samples of positive training images. The top row shows randomly chosen face samples from the films "The Eleventh" and "Kino Pravda" while the bottom row is a random set of faces from the Viola and Jones database.

Negative Sample Images: The set of negative samples has been constructed from frames of the analyzed films of Vertov exclusively. A total of 870 frames containing scenes without faces (i.e. houses, villages, water, a quarry,...) have been selected from the material. Out of those, non-face sample images have been extracted randomly at arbitrary sizes in the range of 24×24 to 200×200 pixels. The square samples have then been resized to the needed base resolution of 24×24 pixels and stored in the training database. The generation is repeated several times during training, as needed for the

bootstrapping algorithm. Figure 21 shows some sample images from the non-face set.

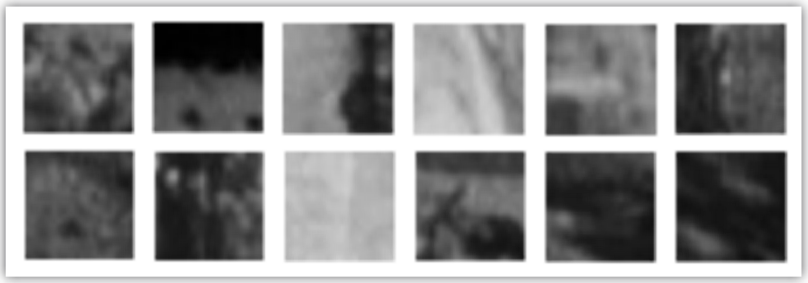

Figure 21: Samples of negative instances. The images have been randomly generated from frames containing no faces from *The Eleventh* and *Kino Pravda*.

5.4 Ground truth

The ground truth for the detector evaluation has been generated with subjective measures. While detection of all faces within a scene, regardless of pose, size and degree of dirt would be desirable, this is not feasible at the time, especially not for archive films. We thus introduce several restrictions:

- **Pose**. The detector is trained on frontal faces. While some amount of rotation (around a vertical axis) is accepted, profile or near-profile faces are excluded.
- **Rotation**. There exist in-plane rotations (around the view axis) and out-of-plane rotations (around a horizontal axis). Faces that are rotated too much in either direction are excluded.
- **Size**. While the detector is trained with 24×24 pixel faces, the minimum size for detection is 50×50 pixels. This is necessary due to the coarse grained source material.
- **Dirt and Occlusions**. Faces that are occluded significantly by objects, other people or dirt are excluded from the ground truth (see sample image in Figure 22).

- **Illumination**. While different illumination conditions are accepted, we exclude faces from the ground truth that show virtually no facial features because of over- or underexposure and backlight.

As mentioned before, the reasons for exclusion are decided by a subjective measure. As a rule of thumb, we used the following: if the contextual information would be missing, could a human decide quickly on whether or not a subwindow is a frontal face? If so, it is added to the ground truth, if not, it is excluded.

Figure 22 shows several examples for faces that are excluded from the ground truth for various reasons.

Figure 22: Restriction of the ground truth of the face detector (problems and reasons for exclusions are marked with borders and symbols).
Top row from left to right: Strong in-plane rotated face, strong out-of-plane rotated face, too small face. Bottom row from left to right: Too small and partly occluded faces, heavy dirt, face in profile pose and with strong backlight.

The decision on whether a detection is a match with the ground truth or not is, again, made on a subjective basis. We count a face as detected, if most of the face is captured by a detection window. Mouth and eyes have to be inside the detection window and it should roughly be centered around these facial features. We manually count the detections as suggested by various authors, including Wu et al. [53]. For examples of frames with highlighted ground truth, see Section 6.2.

5.5 Setup Summary

This section provides a summary of the setup for face detection. Table 9 presents information on the used platform, training images, training setup, preprocessing and filter setup for post processing.

Table 9: Training setup.

Platform	Pentium IV, 2.4GHz 1024 MB RAM, 3GB virtual memory, 3GB switch enabled Windows XP/SP3, Matlab 2007b with Image Processing Toolbox and XML Toolbox [25]
Training images (face samples)	Size: 24×24 pixels Number: 3,900 Source: Viola-Jones database (3,500), films of Vertov (2×200)
Training images (non-face samples)	Size: 24×24 pixels Number: 93,081,550 Source: generated from 870 frames from films of Vertov
Preprocessing of training images	Variance normalization as described in Section 4.1.
Features	Minimum width and height: 6/6 pixels Types: 1, 2, 3, 4, 5 (see Section 4.2.3) Number: 68,000, subsampled to 8,500
Cascade Size (features per stage)	[7, 15, 29, 29, 49, 49, 49, 99, 119, 139, 159, 179, 199, 199, 199, 199, 199, 199, 199, 199]
Weight setting	FFS (equal weights) or LAC; performance decides
Preprocessing of candidate frames	Wiener filtering with 5-pixel kernel Average filtering with 3-pixel kernel
Symmetry filter (see Section 4.3.3)	Symmetry axis shift: $\pm(15/100) \cdot imagewidth$ Threshold (minimum symmetric pixels): 20%
Connectivity filter (see Section 4.3.4)	Threshold 1 (median borders): [0.0 ; 1.7] Threshold 2 (maximum borders): [0.5 ; 4.0]

6 Results

The results are divided into three main parts. First, we describe the resulting classification cascade from training in Section 6.1. Then, we present detection results for different selected scenes in Section 6.2. Finally, Section 6.3 gives an overview over performance in terms of training and detection time.

6.1 Classification Cascade

This section presents some general results from training. First, the size of the calculated classification cascade is described. Then, we give an overview of the first cascade stage and present the selected classifiers and their weights. Additionally, the determination of threshold values and some results from bootstrapping are presented.

Size of the cascade. As described in Section 5.2, the classification cascade is internally represented as a three dimensional array. The cascade calculated for this work has 13 stages with a rising number of classifiers per stage, presented in Equation 26.

$$features = [7, 15, 29, 29, 49, 49, 49, 99, 119, 139, 159, 179, 199] \tag{26}$$

Each additional stage would have 199 features, too, as indicated in the setup summary in Table 9.

The first cascade stage. The first stage consists of 7 features. They are shown on top of a typical training sample from the analyzed films in Figure 23. As expected, the training algorithm (see Section 4.2.5) selected features that encode the eye region (second and sixth). Other features encode only one eye (1 and 7), the top right corner of a candidate window (3) and face edges (4 and 5). The graph beneath the images shows the weights that were selected for the different features by the weight setting procedure (see Section 4.2.6). The orange line indicates the ensemble threshold. Remember, a candidate subwindow is classified as being a face by the stage, if the sum of

all weights of positively scoring features is larger than the threshold. Note, that the largest weights are assigned to features encoding the eye-region. Remember, that the classifiers and weights are selected *automatically* by the training procedure by applying complex statistics that rely *only* on the training images.

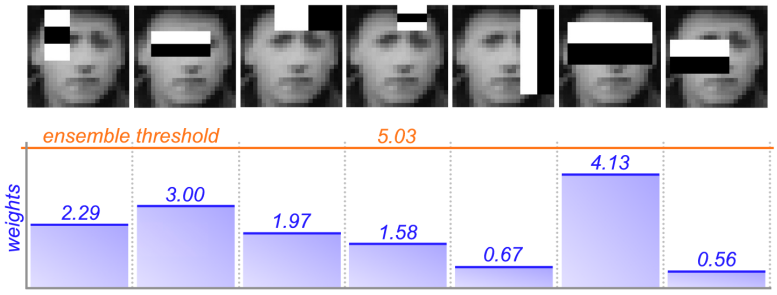

Figure 23: Features of stage 1 with weights. The images show the 7 selected features for the first stage. Beneath is a graph that shows the individual weights of the features, the orange line indicates the enemble threshold of the stage.

Because of the large weight, feature 5 of this cascade stage is assumed to be the strongest feature. We visualize the quality of a single feature by two histograms. Figure 24 shows the histograms for this feature. The histogram at the top summarizes the feature scores for the set of 3900 positive training images (faces). The scores of negative images are summarized by the second histogram. The threshold position is indicated by the red line. It is always placed between the positive and negative images' median score and is determined in a way to have the minimum over all error (see Section 4.2.4). In this setup, the threshold is chosen at about -60 and the parity information indicates, that all candidate windows that score below this threshold are classified as being a face by this single feature.

Bootstrapping results. As described earlier, the negative sample images for training are extracted randomly from frames of the films that contain no faces. Section 4.2.7 describes how the bootstrapping process produces tens of thousands of images and filters out the false-positives as training images for the next cascade stage. The repeated process for a cascade of 13 stages produces a total of $93,081,550$ negative samples out of which approximately

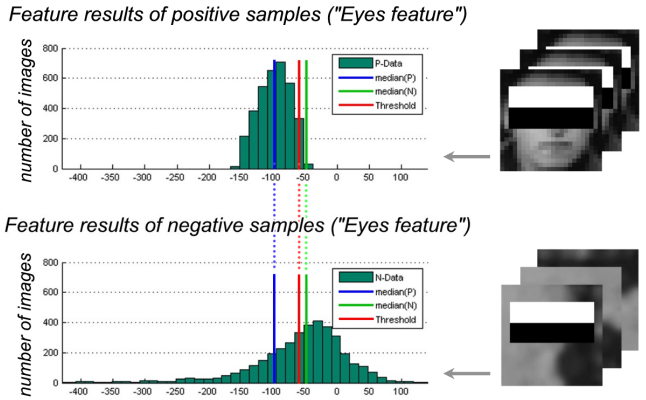

Figure 24: Feature results for *Eyes* feature. The histograms summarize the scores of 3900 positive (top) and negative (bottom) images for the feature, respectively. The blue line indicates the median score of positive samples, the green line the median scores of negative samples. The red line shows the position of the error minimizing threshold.

$3900+12\cdot1950 = 27,500$ samples were used for training (3900 initial samples and $3900/2$ samples per cascade stage). As bootstrapping selects the hardest negative samples we can present some of these samples in Figure 25.

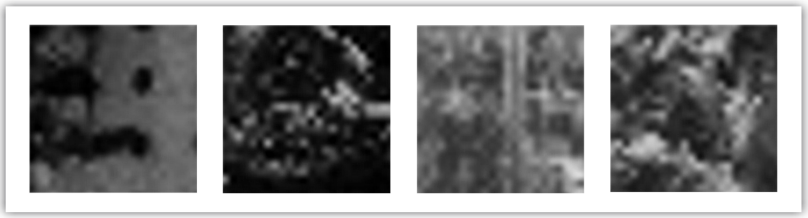

Figure 25: Complex negative samples, accumulated by bootstrapping. These samples were among the hardest to classify for the cascade.

6.2 Accuracy

This section presents a detailed evaluation of the detector for several scenes from the analyzed films. Ground truth and setup are described in Section 5. Section 6.2.6 summarizes the results for all evaluated scenes.

6.2.1 Scene "Marching women" (The Eleventh)

The scene was selected from the film *The Eleventh*. It has 58 frames and covers frames 51,160 to 51,217. The scene shows three women, marching in the direction of the camera position which is dollying away in front of the protagonists. All faces are visible and in acceptable pose for the complete scene, so the ground truth for the scene is $58 \cdot 3 = 174$ faces. The faces differ slightly in size, both within one frame and from frame to frame. By windowing over each frame and scaling the detector, a total of 27,784 subwindows per frame is examined. Thus, the sequence covers $58 \cdot 27,784 = 1,611,472$ subwindows that are classified as containing a face or not.

Figure 26 shows 3 key frames of the scene with green squares which highlight the ground truth.

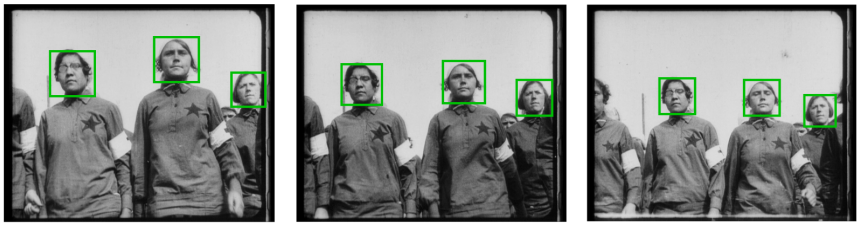

Figure 26: Keyframes of scene *Marching women* with highlighted ground truth (green squares). Frames from left to right: 51,160; 51,187; 51,217.

The achieved recall, precision and fallout for the complete sequence are summarized in Table 10 for 13, 12, 11 and 10 cascade stages. Due to the nature of the cascade, reducing the number of stages results in a higher detection rate but also in a higher false-positive rate and reduced precision. Post processing ameliorates the precision scores by reducing the false-positive rate.

Table 10: Performance for scene *Marching women* with and without post processing.

Stages with and without post processing	**Recall (detection rate)**	**Precision**	**Fallout (false-positive rate)**
13 w/ post processing	69.45 %	69.94 %	0.0032 %
13 w/o post processing	69.45 %	69.14 %	0.0034 %
12 w/ post processing	71.84 %	64.77 %	0.0042 %
12 w/o post procesing	71.84 %	63.13 %	0.0045 %
11 w/ post processing	74.14 %	54.66 %	0.0066 %
11 w/o post procesing	74.14 %	52.44 %	0.0073 %
10 w/ post processing	75.29 %	43.95 %	0.0104 %
10 w/o post procesing	75.29 %	42.26 %	0.0111 %

A detailed evaluation shows, that the most precise detection (13 stages with post processing) allows a maximum of 3 false-positives per frame in 4 frames (6.9%) while there are 20 frames (34.5%) with no false-positives. In each frame, at least one of the faces is detected and all faces are detected in 12 frames (20.7%). The face that is most often missed belongs to the woman in the middle, the face that was detected most often belongs to the woman on the left.

Figure 27 shows the Receiver Operating Characteristic (ROC) for the scene without (red) and with (green) post processing. The variable parameter is the size of the detection cascade (13 to 8 stages). The best false-positive rate but lowest detection rate is obtained by the largest cascade (13 stages). The point with the best detection but highest false-positive rate is the result for an 8-stages cascade.

Figure 28 shows the Recall-Precision Rate for the scene, again for 13 to 8 stages, without (red) and with (green) post processing. Here, the point with the highest precision represents the values for the 13-stages cascade. The point with lowest precision but highest recall is the 8-stages cascade score.

The two figures demonstrate, that post processing (symmetry and connectivity filter) ameliorates the results. In the ROC plot, the false-positive rate is lower for the filtered results while the detection rate remains unchanged. Thus, the recall remains unchanged while the precision rises as

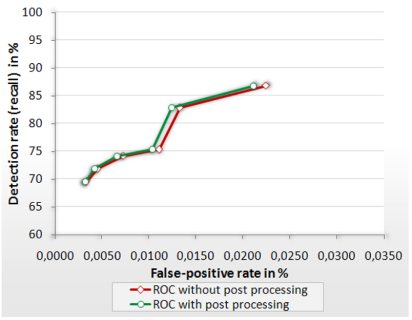

Figure 27: ROC Curve of scene *Marching women* for a cascade of 13 to 8 stages.

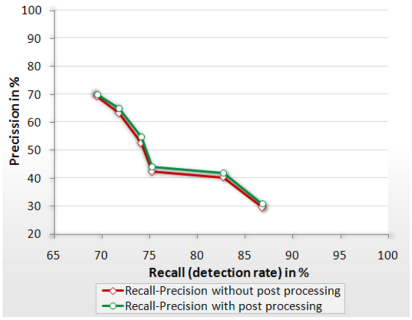

Figure 28: Recall-Precision Rate of scene *Marching women* for a cascade of 13 to 8 stages.

depicted by the Recall-Precision plot. The reduction of false-positives by the post processing procedure is not significant but noticeable. Figure 29 shows 3 representations of a sample frame from the sequence where post processing reduces the number of false-positives from 1 to 0. The left image shows the ground truth, the centered image the results without post processing and the right image the final results with post processing.

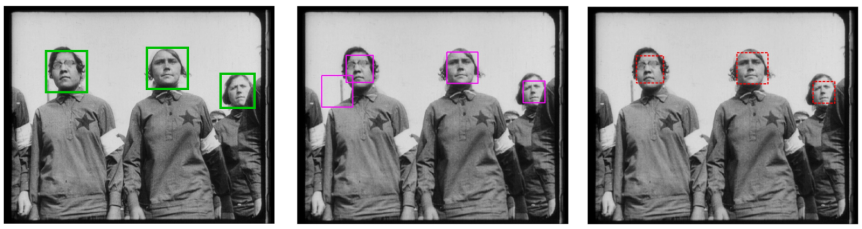

Figure 29: Impact of post processing on detection results for frame 51,169 of scene *Marching women*. From left to right: frame with manually placed ground truth (green squares); frame with detection results from the 13-stages cascade, including one false-positive (purple squares); frame with detections after symmetry and connectivity post processing filtering with no false-positives (red squares).

We present three additional frames with post processed results in Figure 30. While the left and the right frame show detections of all three faces, one face is missing in the centered image. The centered and right frame each show one false-positive.

Figure 30: Detection results in scene *Marching women*. From left to right: frame 51,165 showing correct detection and no false-positives; frame 51,188 which misses one face and has one false-positive (possibly because of the horizontal edge at the top of the subwindow which corresponds to the region where eyes appear in faces); frame 51,215 which shows all three detections but one false-positive.

6.2.2 Scene "Crowd" (The Eleventh)

The scene was selected from the film *The Eleventh*. It has 35 frames (frames 49,788 to 49,822). The scene shows a transition of two separate shots of a crowd, one in the upper half of the image and one in the lower half. Both crowds are moving right in front of the camera. Thus, many different faces are present. The ground truth for the complete scene contains 84 faces. The scene covers a total of $1,515,220$ subwindows.

Figure 31 shows 3 key frames of the scene with green squares which highlight the ground truth. Note that many faces are rotated or profile faces and that the ground truth only covers frontal faces as described in Section 5.4.

Figure 31: Keyframes of scene *Crowd* with highlighted ground truth (green squares). Frames from left to right: 49,788; 49,805; 49,822.

The scene shows the limitations of the detector. The results for false-positive rates are worse than in other scenes. This is a result from the

strong movement in the scene, along with a lot of structure from many different individual objects. There exist no homogeneous areas in the frames. While the false-positive rate is not satisfactory, the detection rate is high in comparison to other scenes.

Table 11: Performance for scene *Crowd* with and without post processing.

Stages with and without post processing	**Recall (detection rate)**	**Precision**	**Fallout (false-positive rate)**
13 w/ post processing	74.33 %	28.28 %	0.0141 %
13 w/o post processing	74.33 %	27.27 %	0.0148 %

As there are a lot of false-positives even for the most precise detector (13 stages with post processing), a ROC Curve is not calculated for the scene. The detector with 13 stages does not produce one frame without false-positives, while 5 result frames (14.7%) had the maximum of 9 false-positives. Possibly, some of the false-positives are a result of too strict exclusion criterias for the ground truth.

Figure 32 shows three sample result frames of the sequence. As the ground truth is not self-explanatory in this scene, a representation of the frame with highlighted ground truth is placed above the detection results for each of the sample frames. Note, that in the left-most frame 2 faces are detected that are excluded from the ground truth because of strong occlusion.

6.2.3 Scene "Dancing women" (Kino-glaz)

The scene from the film *Kino-glaz* has 63 frames (frames 1,561 to 1,623). The scene shows several women dancing in front of the camera. The camera moves slightly as it is focused on a main protagonist. Thus, different faces are visible throughout the scene. There are a few partial occlusions and many faces are slightly rotated in plane. As the protagonists have different distance to the camera, the faces vary in size. The frames show 0 to 4 faces for the ground truth and includes a total of 118 faces. The sequence covers a total of 1,314,317 subwindows that are classified. Figure 33 shows 3 key frames with ground truth.

Figure 32: Detection results in scene *Crowd*. The top row shows the ground truth of each frame (green squares), the bottom row the detection results (red squares). From left to right: frame 49,802 with a ground truth of 2 faces, perfect detection but 3 false-positives; frame 49,812 with 3 of 5 detections but 4 false-positives; frame 49,813 with 4 of 5 detections but again 4 false-positives.

Figure 34 shows the Receiver Operating Characteristic (ROC) for the scene without (red) and with (green) post processing. Again, the variable parameter is the size of the detection cascade (13 to 10 stages). Note that the ROC is almost linear in this scene. Table 12 summarizes the detection results for a cascade with 13 stages.

Table 12: Performance for scene *Dancing women* with and without post processing.

Stages with and without post processing	**Recall (detection rate)**	**Precision**	**Fallout (false-positive rate)**
13 w/ post processing	65.25 %	39.29 %	0.0091 %
13 w/o post processing	66.10 %	38.24 %	0.0096 %

The results are similar to those of other scenes. The detection rate is slightly lower due to the partly complex material. There are small in-plane and out-of-plane rotations of faces and one of the protagonists is very old. The face is missed several times because the training database does not contain a sufficiently large set of elder people. Only one of 78 correct detections is removed by post processing. Detection produces a maximum of 4 false

Figure 33: Keyframes of scene *Dancing women* with highlighted ground truth (green squares). Frames from left to right: 1,561; 1,583; 1,623.

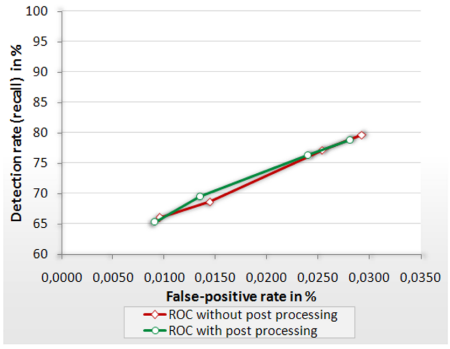

Figure 34: ROC Curve of scene *Dancing women* for a cascade of 13 to 10 stages.

detections per frame in 6 of 58 frames (10.3%). All faces are detected in 28 frames (48.3%) while in one frame none of the 3 faces is detected. Figure 35 presents some results for the scene. Note the face rotations in all three frames.

6.2.4 Scene "Rifle instructor" (Man with a Movie Camera)

The scene has 21 frames (frames 3,550 to 3,570) and was selected from *Man with a Movie Camera*. The scene shows a rifle instructor who passes a rifle to a woman. The woman turns her back to the camera so the only visible face belongs to the instructor. The man's face is partly occluded by the woman in some frames. The camera is steady and ground truth is 20 faces because in the last frame of the sequence, the man's face is occluded too much to be counted as face for the ground truth. A total of 438,039 subwindows is classified in the scene. Figure 36 shows 3 key frames of the scene with highlighted ground truth.

Figure 35: Detection results in scene *Dancing women*. The top row shows the ground truth of each frame (green squares), the bottom row the detection results (red squares). From left to right: frame 1,596 with a ground truth of 3 faces, two slightly rotated, perfect detection but 2 false-positives; frame 1,612 with a ground truth of 1 face with slight out of plane rotation which was detected; frame 1,623 with 2 of 2 detections, one slightly rotated face and 1 false-positives.

Table 13 summarizes the detection results for a cascade with 13 stages. Note the high detection rate of 85% to 90%. In fact, the face is missed by the detector only two times and one time it is discarded due to post processing. Post processing reduces the number of false-positives from 36 to 32 for the complete scene (correct detections are reduced from 18 to 17).

Table 13: Performance for scene *Rifle instructor* with and without post processing.

Stages with and without post processing	**Recall (detection rate)**	**Precision**	**Fallout (false-positive rate)**
13 w/ post processing	85.0 %	34.69 %	0.0073 %
13 w/o post processing	90.0 %	33.33 %	0.0082 %

Figure 37 shows results for three sample frames. Note the correct detection despite partial occlusions of the face.

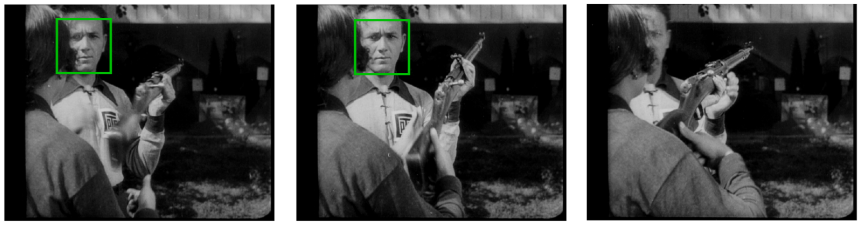

Figure 36: Keyframes of scene *Rifle instructor* with highlighted ground truth (green squares). Frames: 3,550; 3,557; 3,570.

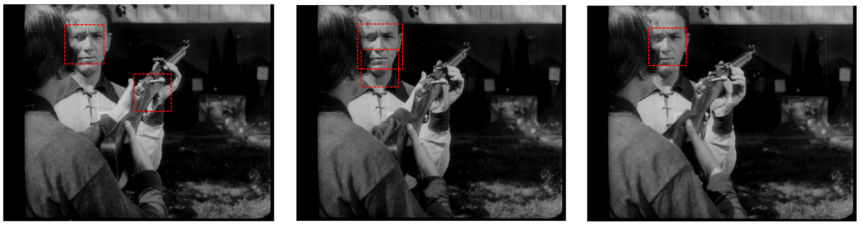

Figure 37: Detection results in scene *Rifle instructor* (frames 3,560, 3,566 and 3,566).

6.2.5 Scene "Cinema Audience" (Man with a Movie Camera)

The scene was again selected from the film *Man with a Movie Camera* and has 50 frames (frames 9,351 to 9,400). The scene shows several people that are watching a movie in the cinema. The camera moves slightly throughout the scene and shows up to 6 frontal faces of varying size. There are a few partial occlusions and the overall illumination changes significantly in some frames. The ground truth includes a total of 253 faces. The sequence covers a total of 1,475,150 subwindows. Some examples from the ground truth are shown in Figure 38.

Figure 39 shows the Receiver Operating Characteristic (ROC) for the scene without (red) and with (green) post processing. Again, the variable parameter is the size of the detection cascade (13 to 10 stages). Note that while the false-positive rate rises as expected, the ROC shows that detection results are not significantly better for a cascade with fewer stages except for the step of 13 to 12 cascade stages (left-most points in the graph). Performance values are given in Table 14.

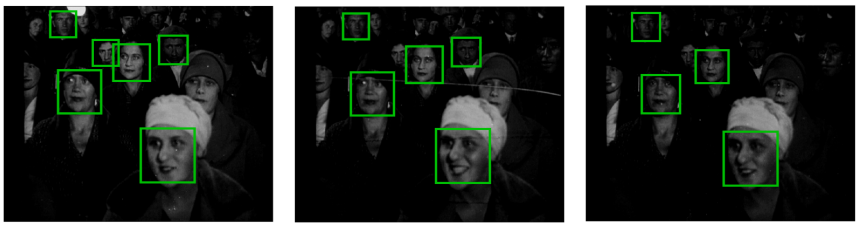

Figure 38: Keyframes of scene *Cinema audience* with highlighted ground truth (green squares). Frames from left to right: 9,352; 9,370; 9,400.

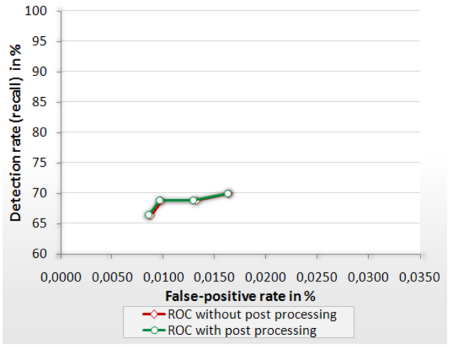

Figure 39: ROC Curve of scene *Cinema audience* for a cascade of 13 to 10 stages.

Table 14: Performance for scene *Cinema audience* with and without post processing.

Stages with and without post processing	**Recall (detection rate)**	**Precision**	**Fallout (false-positive rate)**
13 w/ post processsing	66.40 %	57.14 %	0.0085 %
13 w/o post processing	66.40 %	56.56 %	0.0087 %

The detection rate is slightly lower than in other scenes mainly due to bad illumination. The face of the man at the top right is included in the ground truth in most of the frames. However, it is missed most of the times by the detector. Post processing is not effective in this scene, only 3 false detections are removed. However, no correct detections are removed by post processing. Figure 40 presents results for the scene. Note the restricted ground truth due to partial occlusions, pose, lighting and size of faces.

Figure 40: Detection results in scene *Cinema audience*. The top row shows the ground truth of each frame (green squares), the bottom row the detection results (red squares). From left to right: frame 9,372, 9,388 and 9,391. Note the dark face (top right) in the last two frames that is missed by the detector.

6.2.6 Summary

We presented the detection results of five scenes from films of Dziga Vertov in Sections 6.2.1 to 6.2.5. The scenes have varying length and complexity. The ground truth for the scenes contains 678 faces. Some of these faces are partly occluded, slightly rotated and bad illuminated. A total number of 227 frames are analyzed which include 6,353,998 subwindows to classify. Table 15 summarizes the selected scenes. The column *Evaluation* indicates, with which cascade sizes the scene is evaluated.

Table 15: Summary of the five evaluated scenes.

Scene	Origin	# frames	# faces	Evaluation
Marching women	The Eleventh	58	174	13, 12, 11, 10, 9, 8
Crowd	The Eleventh	35	113	13, 12, 11, 10
Dancing women	Kino-glaz	63	118	13, 12, 11, 10
Rifle instructor	Man with a Movie Camera	21	20	13, 12, 11, 10
Cinema Audience	Man with a Movie Camera	50	253	13, 12, 11, 10
		227	**678**	

The achieved values for recall, precision and fallout over all five sequences are summarized in Table 16 (for a cascade with 13 stages). The values are presented with and without post processing. Note that post processing ameliorates the precision scores by reducing the false-positive rate. However, few faces were removed from the results by post processing too, so the detection rate is slightly lower with post processing.

Table 16: Performance for the five evaluated scenes with and without post processing.

Stages with and without post processing	Recall (detection rate)	Precision	Fallout (false-positive rate)
13 w/o post processing	69.17 %	45.18 %	0.0089 %
13 w/ post processing	68.88 % (-0.4 %)	46.28 % (+2.4 %)	0.0085 % (-4.5 %)

Figure 41 shows the Receiver Operating Characteristic (ROC) for all scenes without (red) and with (green) post processing. As before, the variable parameter in the ROC is the size of the detection cascade (13 to 10 stages). Figure 42 shows the Recall-Precision Rate for all scenes without (red) and with (green) post processing. Again, precision grows with the number of cascade stages.

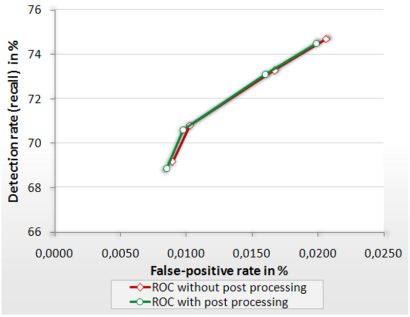

Figure 41: ROC Curve for all scenes for a cascade of 13 to 10 stages.

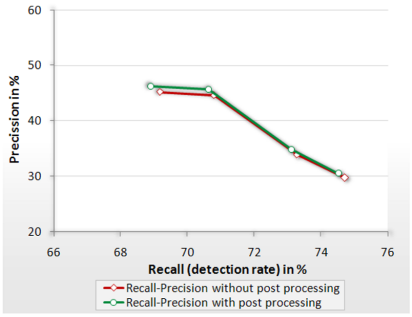

Figure 42: Recall-Precision Rate for all scenes for a cascade of 13 to 10 stages.

6.3 Performance

This section covers the observed training and detection times. Basis for this evaluation is the setup as presented in Section 5.5. We start by summarizing the performance of the training of the cascade in Section 6.3.1 together with a comparison of training with Forward Feature Selection and AdaBoost. Section 6.3.2 gives an overview on the performance of the detector.

6.3.1 Training Time

The presented algorithm needs exhaustive training. This is mainly due to the large database of training images (2×3900 per stage) and the architecture of the learning algorithm. As described in Section 4.2.1, the individual stages of the cascade have to make independent errors. This is achieved by updating the set of negative sample images for each cascade stage by the bootstrapping process. However, this makes recalculation of weak classifiers and repeated classification necessary which is computational expensive.

Bootstrapping itself is a dominating factor for computation costs. Remember that during bootstrapping, new negative sample images are generated that are classified as false-positives by the current cascade. Consequently, when the false-positive rate of the cascade gets better exponentially (as in this algorithm), the bootstrapping costs rise exponentially because it gets more and more difficult to create negative samples that are false-positives.

Figure 43 and 44 show the computation time that is spent for the first 11 (12) cascade stages with normal and logarithmic scaling of computation time, respectively. Reading in images, calculating the weak classifier and precomputing of classification results in table V (for forward feature selection) runs in constant time for all cascade stages. The feature selection is linear in the number of features (later cascade stages select more features). Bootstrapping time rises exponentially. After stage 10, for example, the false positive rate is 0.092 percent and the bootstrapping goal is to generate 1,946 new negative samples, which is about one half of the complete set. The algorithm needed to generate 30,780,434 samples in this setup to collect the roughly 2,000 false-positives. Naturally, this generation is costly.

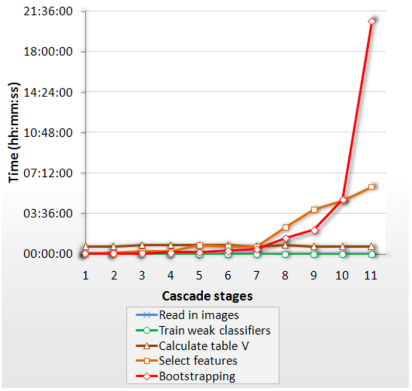

Figure 43: Runtime of training stages 1 to 11. Bootstrapping becomes dominant at stage 10.

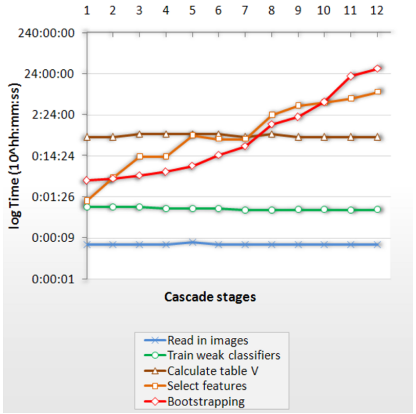

Figure 44: Logarithmic Runtime of training stages 1 to 12.

Figure 45 presents the total time spent for the different parts of the algorithm cumulated over the first 11 stages. The dominating character of bootstrapping becomes obvious here, too. Possibly, a larger set of source images for sample calculation may reduce the time spent for bootstrapping because of less similarity of the training images.

Note that we plot the results for the first 11 stages, omitting stages 12 and 13 due to presentation purposes (the logarithmic plot of the runtime in Figure 44 covers 12 stages).

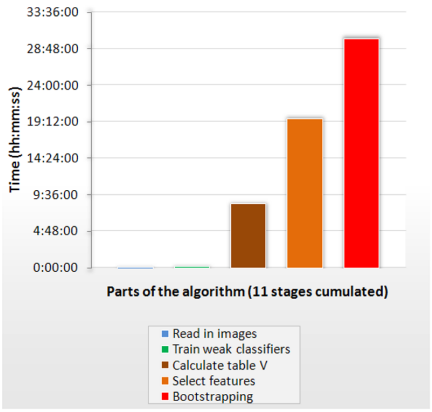

Figure 45: Runtime of different parts of the training algorithm. The values are the cumulated sum of each part for the first 11 cascade stages.

Forward Feature Selection versus AdaBoost. We employ the feature selection method of Wu et al [53] called Forward Feature Selection (FFS), see Section 4.2.5. Compared to the original suggestion of Viola and Jones in [49] (AdaBoost for feature selection) the main advantage of FFS is the shorter computation time. While the two methods do not select the same features, it has been shown by Wu that the detection and false-positive performance of features selected by both methods are similar. Thus, the more efficient algorithm should be selected.

The reason for the performance advantages of FFS over AdaBoost is that the latter needs recalculation of all weak classifiers for every single selected feature. The FFS algorithm needs this recalculation only once per stage. Thus, especially for later stages that have many features, FFS is by far more efficient. Additionally, the weight setting procedure in AdaBoost is coupled to the feature selection and computation of weak classifiers which is slower than calculating it only once per stage as in the algorithm of Wu et al. where weight setting is a separate process after feature selection in each stage.

Figure 46 shows a comparison of the training time of both algorithms for the first 11 cascade stages. While bootstrapping becomes dominant for both approaches in later stages, the plot shows a stronger dependency between computation time and the number of selected features for AdaBoost. The

number of features in the first 11 stages are [7, 15, 29, 29, 49, 49, 49, 99, 119, 139, 159].

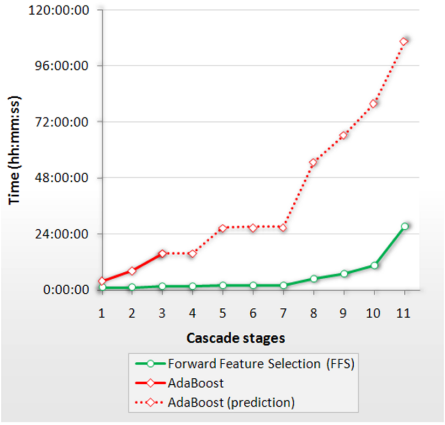

Figure 46: Runtime of the algorithm based on FFS compared to AdaBoost.

Viola and Jones report a training time of several weeks for the AdaBoost algorithm for a 32-stages cascade. While this was not tested empirically in this work, a similar value is estimated based on the performance values measured for the first stages with AdaBoost. Further stages of AdaBoost were not tested because of time restrictions but predicted. Note, for example, that the calculation of stage 10 needs about 3 days and 8 hours with AdaBoost compared to 10 hours for FFS.

6.3.2 Detection Time

For face detection in films, algorithms that detect faces in real time or faster are desired (i.e. detect faces in fast forward mode). However, this is not possible with the approach presented here.

There are several parameters that affect detection time. These parameters are the frame size and time spent for pre-processing as well as the depth of the cascade (i.e. the number of stages). While a cascade with more stages is more precise, detection with a smaller cascade is faster. An important point, however, is the selection of the programming language.

In this work 720×576 pixel frames are analyzed and we employ a cascade with 13 stages. Matlab 2007b is used as programming language and platform, for details see Section 5.5. The detection time for the implementation was

10 to 14 seconds per frame. Figure 47 shows that detection time does not depend on the size of the cascade. This is because later stages are evaluated less often, while most of the work is done by the first few stages. The diagram shows average detection timess from the scene *Marching women* (other scenes have similar detection times).

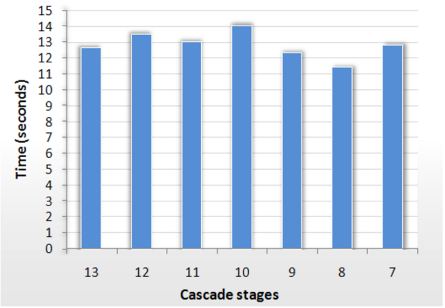

Figure 47: Detection time for cascades of different sizes. The runtime does not depend on the size of the cascade.

An aspect of the algorithm is that the detection cascade can be represented as xml file. Thus, it can easily be used by an optimized detector. This is a strong feature as Viola and Jones [51] and Wu [52] report realtime detection for detectors based on precompiled programming languages.

7 Conclusion

We have presented a study on face detection in the context of historic documentaries and started by giving an overview on the challenges of face detection and popular approaches.

A novel type of source material for face detection was introduced. Artifacts like flicker, scratches, bad contrast and illumination as well as the monochromaticity make this material challenging for automated object detection. We have chosen the face detection method of Viola and Jones for this work. Their method aims at constructing a classification cascade that is built from simple, fast to calculate features. Wu's Forward Feature Selection was employed which allows faster training. Additionally, two post-processing filters have been introduced because of the high false-positive rate of the original approach.

The face detection system has been evaluated with several sample scenes from three films of Dziga Vertov. The overall detection rate (recall) is 68.88 %, the false-positive rate (fallout) is 0.0085 % and the precision 46.28 %. While training of the detector needs several days, the detection time per frame is around 13 seconds.

The presented algorithm is suited for general object detection because of the generality of the features. Changing the set of positive sample images results in the calculation of a cascade which can be used to detect arbitrary objects.

In future, two issues may be addressed. First, the performance of the detection system does not reach realtime by far. Implementing the algorithm in a precompiled programming language may solve this problem. Second, additional post-processing filters are needed to further reduce the high false-positive rate.

APPENDIX

List of Tables

List of Figures

Listings

References

[1] John F. Canny. A computational approach to edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 8(6):679–698, June 1986.

[2] Dmitry Chetverikov and Attila Lerch. Multiresolution face detection. *Theoretical foundations of computer vision*, 69:131–140, 1993.

[3] David Cristinacce and Tim F. Cootes. A comparison of shape constrained facial feature detectors. *IEEE International Conference on Automatic Face and Gesture Recognition*, pages 375–380, 2004.

[4] David Cristinacce, Tim F. Cootes, and Ian Scott. A multi-stage approach to facial feature detection. *Proceedings of the 15th British Machine Vision Conference*, pages 277–286, 2004.

[5] Gian L. Foresti, Christian Micheloni, Lauro Snidaro, and Christian Marchiol. Face detection for visual surveillance. *Proceedings of the 12th In-*

ternational Conference on Image Analysis and Processing, 22:115–120, 2003.

[6] Yoav Freund and Robert E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Computational Learning Theory: Eurocolt '95*, pages 23–37, 1995.

[7] Athos S. Georghiades, Peter N. Belhumeur, and David J. Kriegman. From few to many: Illumination cone models for face recognition under variable lighting and pose. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(6):643–660, 2001. http://cvc.yale.edu/projects/yalefacesB/yalefacesB.html.

[8] Daniel B. Graham and Nigel M. Allinson. Characterizing virtual eigensignatures for general purpose face recognition. *Face Recognition: From Theory to Applications*, 163:446–456, 1998. http://images.ee.umist.ac.uk/danny/database.html.

[9] Hayid Greenspan, Serge Belongie, and Rodney Goodman et al. Overcomplete steerable pyramid filters and rotation invariance. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 222–228, 1994.

[10] Bernd Heisele, Thomas Serre, Sam Prentice, and Tomaso Poggio. Hierarchical classification and feature reduction for fast face detection with support vector machines. *Elsevier Pattern Recognition*, 36:2007–2017, 2003.

[11] Erik Hjelmås and Boon Kee Low. Face detection: A survey. *Computer Vision and Image Understanding*, 83:236–274, 2001.

[12] Rudra N. Hota, Vijendran Venkoparao, and Saad Bedros. Face detection by using skin color model based on one class classifier. *9th International Conference on Information Technology (ICIT 06)*, pages 15–16, 2006.

[13] Anil K. Jain, Yu Zhong, and Marie-Pierre Dubuisson-Jolly. Deformable template models: a review. *Elsevier Signal Processing*, 71(2):109–129, 1998.

[14] Oliver Jesorsky, Klaus J. Kirchberg, and Robert W. Frischholz. Robust face detection using the hausdorff distance. *Third International Conference on Audio- und Video-based Biometric Person Authentication*, pages 90–95, 2001. http://www.bioid.com/downloads/facedb/index.php.

[15] Michael Kirby and Lawrence Sirowich. Application of the karhunen-loève procedure for the characterization of human faces. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 12(1):103–108, January 1990.

[16] Constantine Kotropoulos and Ioannis Pitas. Rule-based face detection in frontal views. *Proceedings of IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP 97)*, 4:2537–2540, 1997.

[17] Vera Kropf, Matthias Zeppelzauer, Stefan Hahn, and Dalibor Mitrovic. First steps towards digital formalism: The vienna vertov collection. *Proceedings of the International Workshop on Digital Tools in Film Studies*, 2007.

[18] Fernando De la Torre Frade and Jeffrey Cohn. Depression assessment. http://www.ri.cmu.edu/projects/project_633.html, last visited: October 2008.

[19] Marc Líevin and Franck Luthon. Nonlinear color space and spatiotemporal mrf for hierarchical segmentation of face features in video. *IEEE Transactions on Image Processing*, 13(1):63–71, January 2004.

[20] Yongmin Li, Shaogang Gong, Jamie Sherrah, and Heather Liddell. Support vector machine based multi-view face detection and recognition. *Image and Vision Computing*, 22:413–427, 2004.

[21] David Marr and Ellen C. Hildreth. Theory of edge detection. *Proceedings of the Royal Society of London. Series B, Biological Sciences*, 207:187–217, 1980.

[22] Aleix M. Martinez and Robert Benavente. The ar face database. CVC Technical Report #24, June 1998. http:// cobweb.ecn.purdue.edu/ãleix/aleix_face_DB.html.

[23] Kazumasa Murai and Satoshi Nakamura. Deformable template models: a review. *IEEE International Conference on Multimedia and Expo (ICME)*, 2:373–376, 2002.

[24] Ara V. Nefian and Monson H. Hayes. Face detection and recognition using hidden markov models. In *International Conference on Image Processing (ICIP)*, pages 141–145, 1998.

[25] University of Southampton. Xml toolbox, geodise project, 2004. http:// www.geodise.org/toolboxes/generic/xml_toolbox.htm, last visited: January 2009.

[26] Constantine P. Papageorgiou, Michael Oren, and Tomaso Poggio. A general framework for object detection. *International Conference on Computer Vision*, 00:555, 1998.

[27] Claudio Perez. Real-time iris detection on coronal-axis-rotated faces. *IEEE Transactions on Systems, Man and Cybernetics*, 37(5):971–978, 2007.

[28] Thang V. Pham, Marcel Worring, and Arnold W. M. Smeulders. Face detection by aggregated bayesian network classifiers. *Pattern Recognition Letters*, 23(4):451–461, 2002.

[29] P. Jonathon Phillips, Hyeonjoon Moon, Syed A. Rizvi, and Patrick J. Rauss. The feret database and evaluation procedure for face recognition algorithms. *Image and Vision Computing J*, 16(5):295–306, 1998. http://face.nist.gov/colorferet/.

[30] P. Jonathon Phillips, Hyeonjoon Moon, Syed A. Rizvi, and Patrick J. Rauss. The feret evaluation methodology for face recognition algorithms. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22:1090–1104, December 2000. http://face.nist.gov/colorferet/.

[31] Son L. Phung, Abdesselam Bouzerdoum, and Douglas Chai. Skin segmentation using color pixel classification: Analysis and comparison. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(1):148–154, January 2005.

[32] Aristodemos Pnevmatikakis and Lazaros Polymenakos. Comparison of eigenface-based feature vectors under different impairments. *Proceedings of the 17th International Conference on Pattern Recognition*, 1:296–299, 2004.

[33] Henry A. Rowley, Shumeet Baluja, and Takeo Kanade. Cmu frontal test set a, c and rotated. http://vasc.ri.cmu.edu/idb/html/face/frontal_images/, last visited: June 2008.

[34] Henry A. Rowley, Shumeet Baluja, and Takeo Kanade. Neural network-based face detection. *Transactions on Pattern Analysis and Machine Intelligence*, 20(1):23–38, 1998.

[35] Brian Scassellati. Eye finding via face detection for a foveated, active vision system. *Proceedings of the 15th National Conference on Artificial Intelligence*, pages 969–976, 1998.

[36] Robert E. Schapire, Yoram Singer, and Amit Singhal. Boosting and rocchio applied to text filtering. *Proceedings of ACM SIGIR*, pages 215–223, 1998.

[37] Terence Sim, Simon Baker, and Maan Bsat. The cmu pose, illumination, and expression database. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(12):1615–1618, December 2003. http://www.ri.cmu.edu/projects/project_418.html.

[38] Pawan Sinha. Object recognition via image invariants: A case study. *Investigative Ophthalmology and Visual Science*, 35(4):1735–1740, 1994.

[39] Saad A. Sirohey. Human face segmentation and identification. University of Maryland: Technical Report CS-TR-3176, CS Department, 1998.

[40] Irwin Sobel and Jerome A. Feldman. A 3x3 isotropic gradient operator for image processing. Talk at the Stanford Artificial Project, 1968.

[41] Karin Sobottka and Ioannis Pitas. Segmentation and tracking of faces in color images. *Proceedings of the 2nd International Conference on Automatic Face and Gesture Recognition*, pages 236–241, 1996.

[42] Libor Spacek. The essex university face database. http://cswww.essex.ac.uk/mv/allfaces/, last visited: June 2008.

[43] Kah-Kay Sung and Tomaso Poggio. Cmu frontal test set b. http://vasc.ri.cmu.edu/idb/html/face/frontal_images/, last visited: June 2008.

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

[45] Filareti Tsalakanidou, Sotiris Malassiotis, and Michael G. Strintzis. Face localization and authentication using color and depth images. *IEEE Transactions on Image Processing*, 14(2):152–168, February 2005.

[46] Dzmitry Tsishkou, Liming Chen, and Eugeny Bovbel. Mobile face detection and tracking for media streaming applications. *International Journal of Wireless and Mobile Computing*, 2(2):192–211, 2007.

[47] Yale University. The yale university face database. http://cvc.yale.edu/projects/yalefaces/yalefaces.html, last visited: June 2008.

[48] Paul Viola and Michael J. Jones. Face training data. http://www.cmucam.org/wiki/viola-jones, last visited: September 2008.

[49] Paul Viola and Michael J. Jones. Robust real-time object detection. *ICCV Workshop on Statistical and Computation Theories of Vision*, 2:747–747, July 2001.

[50] Paul Viola and Michael J. Jones. Fast and robust classification using asymmetric adaboost and a detector cascade. *Advances in Neural Information Processing Systems 14*, pages 1311–1318, 2002.

[51] Paul Viola and Michael J. Jones. Robust real-time face detection. *International Journal of Computer Vision*, 57(2):137–154, 2004.

[52] Jianxin Wu, S. Charles Brubaker, Matthew D. Mullin, and James M. Rehg. Fast asymmetric learning for cascade face detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, pages 369–382, 2008.

[53] Jianxin Wu, James M. Rehg, and Matthew D. Mullin. Learning a rare event detection cascade by direct feature selection. *Advances in Neural Information Processing Systems 16*, pages 1523–1530, 2004.

[54] Guangzheng Yang and Thomas S. Huang. Human face detection in a complex background. *Pattern Recognition*, 27(1):53–63, 1994.

[55] Jie Yang and A. Waibel. A real-time face tracker. *IEEE Proceedings of the 3rd Workshop on Applications of Computer Vision (WACV)*, pages 142–147, 1996.

[56] Ming-Hsuan Yang, David J. Kriegman, and Narendra Ahuja. Detecting faces in images: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 00:34–58, 2002.

[57] Yuuki Yokoo and Masafumi Hagiwara. Human faces detection method using genetic algorithm. pages 113–118, 1996.