Masarykova univerzita
Fakulta informatiky

Face detection in images

Bakalářská práce

Jiří Jánoš

2008

Masarykova univerzita
Fakulta informatiky

Face detection in images

Bakalářská práce

Jiří Jánoš

2008

Masarykova univerzita
Fakulta informatiky

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Datum: 16.4.2008

Student(ka): Jiří Jánoš

Program: FI B-AP Aplikovaná informatika, bakalářský studijní program

Vedoucí práce: RNDr. Stanislav Bartoň, Ph.D.

Katedra (pracoviště): Centrum zpracování přirozeného jazyka - Fakulta informatiky

Název práce: Face Detection in Images

Zadání: Cílem bakalářské práce bude rešerše v oblasti detekce obličejů v obrazových datech a následná implementace nejlepšího nalezeného algoritmu. Implementace bude provedena v Javě. Součástí práce bude vytvoření porovnání studovaných technik v textové části práce.

Základní literatura: *Handbook of face recognition.* Edited by Stan Z. Li - Anil K. Jain. New York : Springer, 2005. x, 395 s. ISBN 038740595X.

Souhlas se zadáním (podpis, datum)

student(ka)	vedoucí bakalářské práce	vedoucí odpovědného pracoviště

Declaration

By my signature below, I certify that this thesis I am presenting is entirely my own work. In my thesis I have cited all the sources (printed, electronic or oral), I have used them faithfully and have always indicated their origin.

Date: _____ Signature: _____

Acknowledgement

I would like to express my gratitude to my thesis supervisor, RNDr. Stanislav Bartoň, Ph.D., for guiding me during my thesis. Exchanges of ideas and discussions were a source of inspiration and the success of my work.

I wish to thank Mike Jones from Mitsubishi Electric Research Laboratories and Hamed Masnadi-Shirazi from Department of Electrical and Computer Engineering of University of California for their kind assistance with writing emails, giving wise advice and helping with implementation of AdaBoost.

I want to express my gratitude to METACentrum project, that focuses on activities concerning super-computing, maintaining of current computational resources and broadening of available computational capacity of the largest academic centres in the Czech republic. Thanks go to them for their computational support during implementation of AdaBoost.

Lastly, I wish to thank my family, friends and all those who supported me and gave me the opportunity to achieve this thesis.

Abstract

In recent years great research effort was made to capture, recognise and analyse objects in our constantly changing environment and much progress has been achieved. Growing performance of wide available and low-cost desktop computers created an enormous interest in the sphere of face detection and recognition and enabled developing of more sophisticated methods processing images in real time. Over the past ten years lot of strategies have been thoroughly studied in computer vision research and the efficiency of face detection techniques has improved significantly since the first face detection systems. Nowadays, collection of the most widely known strategies includes detectors based on Template matching, Neural networks, Support Vector Machines or AdaBoost-based learning. These and other suggested systems focus on reaching high reliability, efficiency and speed. In other words, they aim to detect all faces present in the scene while reaching as low false alarm rate as possible.

Although there are dozens of various methods and algorithms generally, they can be classified by common features into several categories, from which Appearance-based methods are the most significant ones comprising very efficient approach based on machine learning called AdaBoost (abbreviation for Adaptive Boosting). Since the AdaBoost-based detection belongs to the most successful and favoured ones, the extra attention is paid to this approach. Its authors claim that their real-time system is approximately 15 faster than any previous approach. Moreover, detector can deal with various poses and rotation classes of face and is independent on color information of image. Thus, faces present in grayscale images can be detected too. Finally, the implementation of detector based on Adaptive Boosting for Java platform is suggested in the second part of the thesis.

Keywords

Face detection, Face localization, Template matching, Neural Network, Support Vector Machines, Machine learning, AdaBoost, Haar-like features, Integral image, Cascade of classifiers

Contents

1 INTRODUCTION

Face detection in images is a computer technology that examines the problem of detecting the locations in arbitrary (digital) images where faces are present. In other words, given a single image, the goal is to determine whether or not there are any faces in the scene and, if present, return the specific location and fit each face into the bounding box defined by the image coordinates of the corners. Similarly, *face localization* in images is a technology whose task is to find the locations and sizes of a known number of faces in image (usually one). Face detection is considered to be the first task performed while processing scenes for varied purposes and its results are important for subsequent steps of automated human *face recognition*. Therefore the whole process should work predictably and quite reliably. Designed systems are then used in biometrics (facial recognition systems for automatically identifying or verifying a person, including applications for visual surveillance and security), vision-based human-computer interfaces, image database management or camcorder sphere (developing digital cameras or cell phones using autofocus in so-called Face Priority Mode to better capture people in still photography and video). To build these fully automated systems that analyse the information contained in face images, robust, fast and quality face detection algorithms are required.

In recent years great research effort was made to capture, recognise and analyse objects in our constantly changing environment and much progress has been achieved. Also automatic face detection and localization in a single intensity or color image became a quite well explored problem. Over the past ten years a lot of strategies (more than a hundred reported approaches for face detection) have been thoroughly studied in computer vision research and the efficiency of face detection techniques has improved significantly since the first face detection systems. Growing performance of wide available desktop computers created an enormous interest in this sphere and enabled and sped up developing of more sophisticated methods processing images in real time. Early face detection systems focused on the detection of *frontal human faces*, whereas newer approaches attempt to solve the more general and difficult problem of *multiview face detection*. The algorithms should detect all faces in a single image (or video) independently of face position, scale, in-plane rotations (rotations along the axis from the face to the observer) and out-of-plane face rotation (rotations along the vertical or left-right axis), facial features such as beards, moustaches and glasses, shape, face-color (ethnic groups), age, facial expression, texture, imaging conditions such as multiple shadows and complex lighting (spectra, source distribution and intensity), camera characteristics (sensor response, lenses) and the other factors that affect the image content. Although many of these non-trivial problems have almost be solved, there is still much work needed to improve accuracy and performance of proposed techniques.

All suggested systems focus on reaching high reliability, efficiency and speed. Each of these algorithms tries to detect face in a scene while reaching the lowest *false alarm (false-positive) rate* combined with the best *detection rate*. The terms true-positive, false-positive, true-negative, and false-negative make reference to whether the classification was correct (true/correct or false/incorrect), and the classification return value (positive image or negative image). An ideal face detection system should reach a hit rate of 100% with false alarm rate of 0. But none of contemporary systems can achieve this generally because increasing detection rate is usually connected with the simultaneous false alarm rate increment. Finding

the best ratio between these two critical components and suggesting highly accurate system dealing with complexity of face manifolds will keep challenging for many students, scientists and engineers working in image processing, computer vision, animation and biometrics.

Subsequent chapters of this bachelor thesis are devoted to survey of the most common and successful algorithms and different methods used in domain of Face detection. Specifically, Chapter 2 describes in details sphere of various approaches. Performance evaluation and comparison of suggested methods is summarised in Chapter 3. Sources of face images are listed in Chapter 4, which is devoted to face image databases. The goal of this thesis is to find and implement the best solution to detect faces in images. Therefore, in Chapter 5, extra attention is paid to *AdaBoost-based learning* and its implementation in Java. Subsequently, Chapter 6 reviews all the methods mentioned in the survey, gives the reader an overall comparison of them and also summarises the implementation of a face detector. Finally, Chapter 7 provides references to many outstanding works published on the topic of Face detection in images. In the end of this thesis annexes can be found.

2 FACE DETECTION METHODS

There is a large number of different strategies dealing with a problem of face detection and localization. As suggested in [1], they can be classified by common features into several categories: *Knowledge-Based Methods*, *Feature-Invariant Approaches*, *Template Matching* containing facial/head shape techniques and *Appearance-Based Methods*, or possibly, a mixture of all of the previous. Below, subsequent four paragraphs give short description and general introduction to each category, and *Table 1* summarises representative algorithms for face detection in a single image within these categories.

Knowledge-Based Methods are rule-based methods that use researcher's prior knowledge of what features a typical human face is formed. They come up with the knowledge of simple rules to describe the features of a face and capture their relationships by their relative distances and locations (for example, they commonly use distance between the eyes, top of the head to eyes, eyes to the nose or eyes to the mouth). Facial features in an input image are extracted and all possible face candidates are identified based on the coded rules. Then a verification process is often started to decrease false detection rate. Crucial drawback of these approaches is that they are difficult to apply on faces in different poses. For that reason, proposed methods can be successfully applied on sets of faces in frontal views. They are used mainly for face localization.

Feature-Invariant Approaches in contrast to the knowledge-based approach, aim to find invariant structural features (eyes, nose, eyebrows, mouth, hair-line) that exist even when the pose, viewpoint, or lighting conditions vary. These features are commonly extracted using edge detectors. Based on the extracted features, a statistical model is built to describe their relationships and to verify the existence of a face. One advantage of these approaches is that they have better tolerance toward variations of pose and facial expression. On the other hand, the major disadvantage of these feature-based algorithms is that the image features can be severely corrupted due to illumination, noise and occlusion. Feature-based algorithms focus on facial details and require higher resolution images than template-based approaches. Within this category we can find representative solutions based on facial features, texture, skin color or multiple features (combination of skin color, shape and size). These methods are designed mainly for face localization.

Template Matching is a simple task of performing a normalised correlation between a template image (object in large training set) and a new image to classify. It uses several standard patterns of a face that are stored to describe the face as a whole or the facial features separately. The goal is to loop through all the pixels in the search image and compare them to the pattern. The correlations between an input image and the stored patterns are computed for detection. As the execution time is proportional to the size of the training set, the problem is solved by reduction of the size of the set against which new images are compared. While this method is simple to implement and understand, it is one of the slowest methods. This category contains strategies using *Predefined Face Templates* (Shape Template) and *Deformable Templates* (e.g. Active Shape Model, ASM) and has been used for both face localization and detection.

Appearance-Based Methods solve a problem of classifying each scanned subwindow as a face or a nonface using classifiers that are learned from a set of training images. These training sets are (in contrast to template matching) composed of manifold face examples and face/nonface classifier is

built to deal with this variability of facial appearance. These learned models are then used mainly for face detection and include representative approaches as *Neural Networks (NN), Eigenfaces, Distribution-based methods, Support Vector Machines (SVM), AdaBoost-based learning,* etc. Appearance-based methods using learning algorithms have attracted much attention recently and have demonstrated excellent results. Nowadays they are the most successful ones among the face detection methods.

Category	*Approach*	*Representative work*
Knowledge-Based Methods	Multiresolution rule-based method	Yang and Huang [2]
	Vertical and horizontal profiles	Kotropoulos and Pitas [3]
Feature-Invariant Approaches	Skin-color detection	Chupeau et al. [4]
	Texture-based detection	Dai and Nakano [5]
	Facial features	Yow and Cipolla [6]
	Multiple features	Zhang and Izquierdo [7]
Template Matching	Predefined face templates	Sakai et al. [8]
	Deformable templates	Cootes and Taylor [9]
Appearance-Based Methods	Support Vector Machines	Osuna et al. [10]
	Neural Networks	Rowley et al. [11]
	Eigenfaces	Turk and Pentland [12]
	Distribution-based methods	Sung and Poggio [13]
	Local appearance	Schneiderman and Kanade [14]
	AdaBoost-based detector	Viola and Jones [15]

Table 1. Face detection approaches divided into four categories by their common features

2.1 Knowledge-Based Top-Down Methods

Knowledge-based methods, as mentioned above, are built on some a-priori knowledge of simple rules capturing the relationship of facial features (eyes, nose, lips, mouth) by their relative distances and positions. For efficiency, the image is searched at the coarsest scale first. Once a match is found, the image is searched at the next finer scale until the finest scale is reached. Illustrative face detector can easily explore an image and look for a nose with mouth and two oval eyes above situated in symmetrical positions. Subsequently, eyebrows, nostrils and mouth detection rules are used to validate those facial candidates above. All of these facial features are found on a basis of coded rules and forwarded to verification process to reduce false alarms rate. The main problem of this approach is that human knowledge of facial features has to be well-formed and precisely defined. Too general rules cause higher false positives occurrence, whereas too detailed and strict rules not all of faces can pass

through. Moreover, it's not easy to evolve this approach to discover faces in miscellaneous poses.

2.1.1 Multiresolution Rule-Based Method

Yang and Huang [2] based their detection system on *Three-level architecture* (shown in *Figure 1*) implementing hierarchical knowledge-based method to locate faces. The multiresolution hierarchy of images is created by averaging and subsampling and each level works with images of different resolution (the higher level, the more detailed image is used). Corresponding low resolution images are built-up of square cells where each cell consists of *n x n* pixels (n=1 for original image, 4, 8, 16 for the lower-resolution ones) in which the intensity of each pixel is replaced by the average intensity of the pixels in that cell. This coarse-to-fine or focus-of-attention strategy is the most significant feature of Yang-Huang's approach and is used to reduce the required complexity of the algorithm.

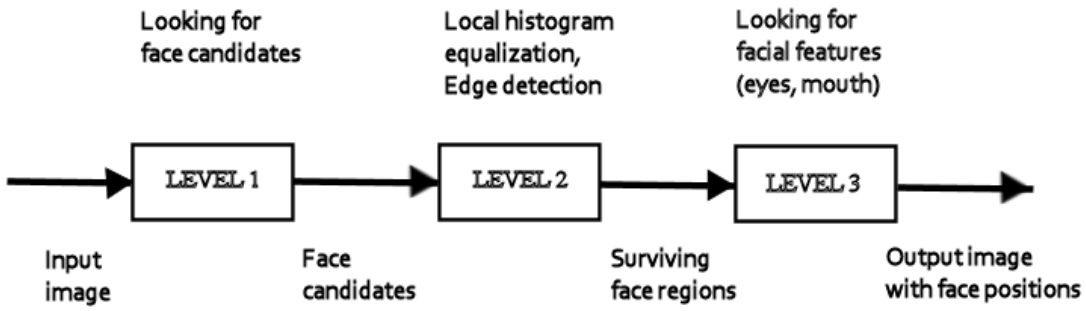

Figure 1. Yang-Huang's Three-Level Detection System

At the Level 1, an input image (transformed to the lowest resolution) is scanned at all positions for face candidates and elemental rules based on general face description are applied here, e.g. intensity distribution and difference of a typical face (central positions of the face, where nose, eyes and mouth are located, are usually darker while the upper round part of a face is light shaded and has a basically uniform intensity, and the general difference between the average gray values of the center part and the upper round part is significant, see *Figure 2*).

Figure 2. Visible dark-shaded areas of eyes, nose and mouth (image taken from [1])

Face candidates continue to Level 2, where more precise and detailed rules are used to

confirm these areas to be faces or not. At this level, local histogram equalization is performed followed by edge detection and finally surviving candidate regions are then examined at Level 3 with another set of rules that respond to facial features such as the eyes and mouth. The whole detection process is summarised in *Figure 3*.

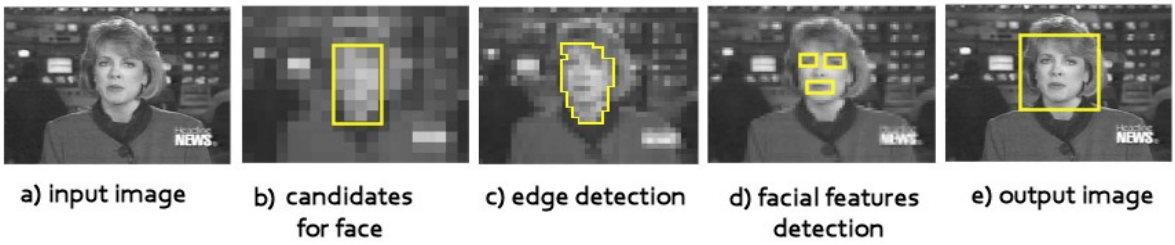

Figure 3. Processing of multiresolution images of cells of n x n pixels using course-to-fine strategy reducing the required computation a) original image, n = 1 b) n = 16 c) n = 8 d) n = 4 e) output image, n = 1

Even though this approach does not achieve a high detection rate, the crucial ideas of using a multiresolution hierarchy and rules to reduce the required computation of the algorithm have been used in later more sophisticated face detection systems, e.g. in the Kotropoulos and Pitas one [3] described in subsequent section.

2.1.2 Vertical and Horizontal Profiles

Kotropoulos and Pitas [3] introduced a very attractive approach for face detection that is also based on *multiresolution images* (also known as *mosaic images*). Motivated by the simplicity of this approach, a rule-based face detection algorithm in frontal views is developed that extends the work of Yang and Huang [2]. In order to avoid the iterative nature of Yang's method, Kotropoulos and Pitas propose to estimate the cell dimension *n* in the quartet image by processing the *horizontal* and *vertical profiles* with a projection method (proposed by Kanade [16] in order to easily detect facial bounds). Horizontal and vertical projections of the image are defined as

$$HI(x,y)=\sum_{y=1}^{n} I(x,y) \qquad VI(x,y)=\sum_{x=1}^{m} I(x,y)$$

where *I(x,y)* is the intensity value of *m x n* image at position *(x,y)*.

At the first step the horizontal image profile is got and then the left and the right sides of a head are located in terms of the two local minima of *HI(x,y)* projection function. Likewise the second step, the vertical image profile is obtained by *VI(x,y)* projection and the local gray-level minima can indicate the positions of mouth lips, nose tip and eyes. These detected features then form a facial candidate. Subsequently, eyebrow/eyes, nostrils/ nose and the mouth detection rules are used to validate or exclude these candidates.

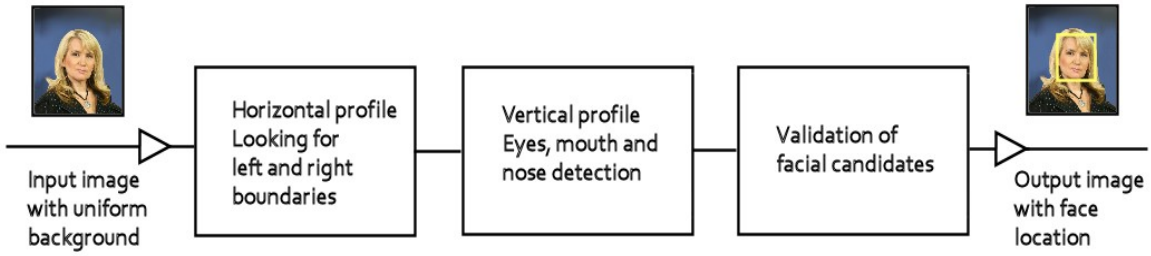

Figure 4. Kotropoulos and Pitas's detection system

Kotropoulos and Pitas's method works perfectly with images that contain only one face in a uniform background. On the other hand, the main drawback of this method appears while processing images with a complex background, where locations of faces are too difficult to determine using the horizontal and vertical profiles (see *Figure 5b*). Furthermore, this method cannot readily detect multiple faces as illustrated in *Figure 5c*.

The proposed algorithm has been applied to frontal views extracted from the European ACTS M2VTS (Multi Modal Verification for Teleservices and Security applications) database that offers synchronized video (and speech data) as well as image sequences of 37 subjects allowing to access multiple views of a face with the uniform background.

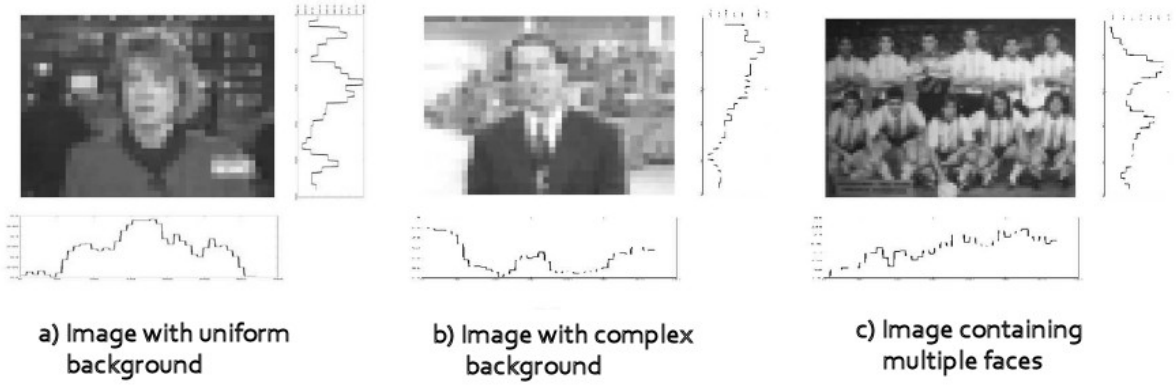

Figure 5. Images with peaks and minima in their horizontal and vertical profiles Cell dimensions a) n = 8 b) n = 8 c) n = 4

2.2 Feature-Invariant Approaches

Feature-Invariant Approaches (also called *Bottom-Up Feature-Based Methods*) search the image for a set of facial features such as eyebrows, nose, hair-line, eyes, mouth and groups them into face candidates based on their geometric relationship to verify the existence of a face. The advantage of these approaches is that they have better tolerance toward variations of pose and facial expression. Though this approach can be easily extended to multiple views, it is unable to work well under different image conditions (illumination, noise, occlusion)

because the image structure of the facial features vary too much to be robustly detected by the feature detectors. By come to intensive light, some features can be suppressed and become indistinctive, whereas shadows form strong edges and make the grouping algorithm unusable.

2.2.1 Skin Color Detection

Bertrand Chupeau et al. [4] described in their paper a new algorithm for automatic face detection in color images. This feature-based method is suitable for real-time image processing when color (preferably with motion) are available (while for gray-level pictures, some learning-based approach is the most effective one). It is based on human skin color distribution that differs from that of most of nonface objects and is assumed to be a sufficiently effective feature for face detection. Even though different people have different skin color, several researches have shown that the principal difference lies largely between their intensity rather than their chrominance (several color spaces have been used to classify pixels as skin including RGB, normalized RGB, HSV, YCbCr, CIE).

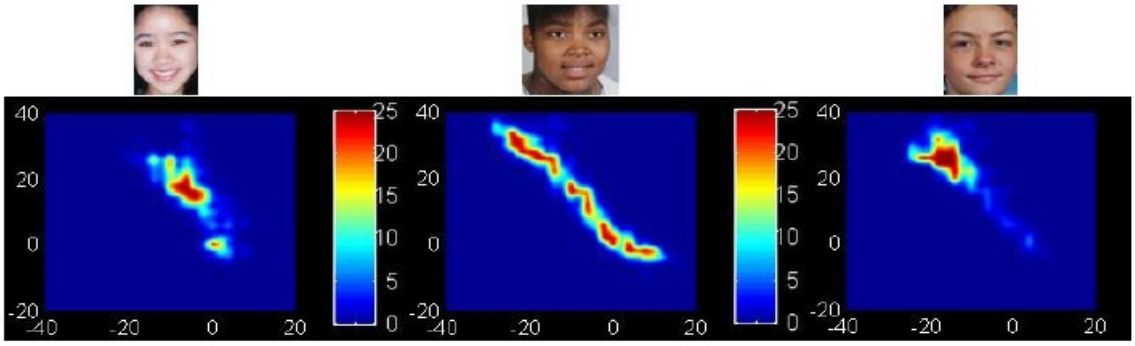

Figure 6. Skin color distribution in YCbCr color space (image taken from [4])

Indeed, under white lighting conditions, with different orientation or view angles, color doesn't vary significantly. In a normalized chrominance space, skin color (no matter the ethnic group) was shown to fall into small cluster (as seen in *Figure 6* above) that can be approximated with a single Gaussian distribution.

The whole underlying algorithm based on human skin color distribution consists of several (generally four) steps. The first task is *detection of skin-color pixels*, then region segmentation and *selection of skin-color regions* are performed, followed by *shape-based merge* of selected regions and finally *discarding of false positives* is proceeded. All steps are summarized in *Figure 7* and described in details in the following paragraphs.

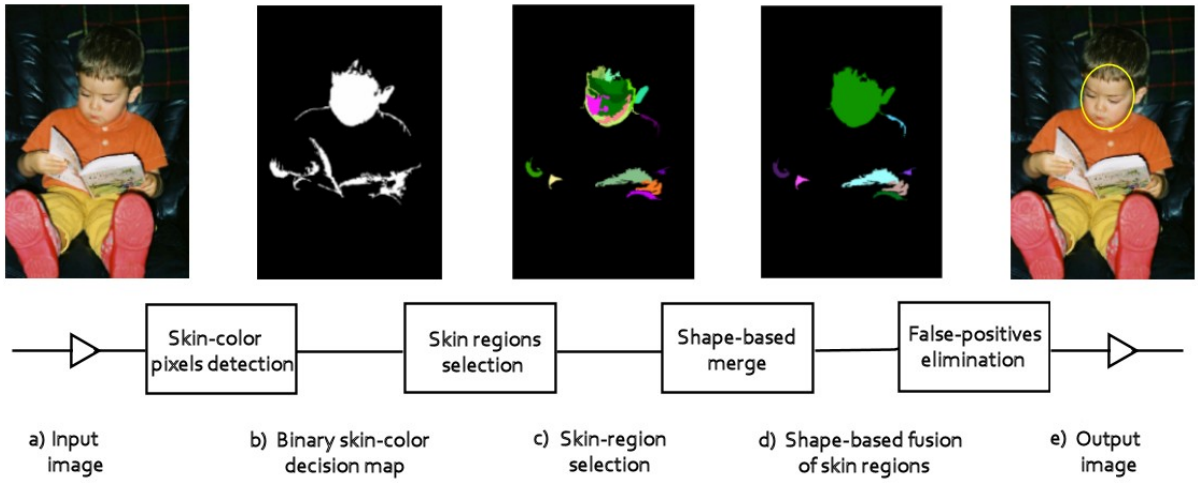

Figure 7. Four stages of skin color detection system with their input and output images

1) Skin-color pixels detection is the first task performed while detecting faces in terms of this approach. Many methods have been proposed to build a skin color model and last studies shown that *YCbCr color space* [17] (sometimes abbreviated to YCC) is one of the simplest and the most successful color spaces in segmenting skin color. In YCbCr chrominance components are represented by *Cb* (blue chroma) and *Cr* (red chroma) values and *Y* luminance component is naturally separated from them. This separation allows skin color detection system to ignore the luminance component of the color space to avoid difficulties of lighting conditions.

As mentioned above and seen in *Figure 6*, human skin color (independently of ethnic group) falls within a quite small segment of chrominance plane (the full scale ranges from –128 to 127). Using the learning database of skin-color pixels, assembled by manual cropping of skin segments from the images, a skin-color probability model can be built in the form of a bi-dimensional Gaussian function (model demonstrated in *Figure 8*).

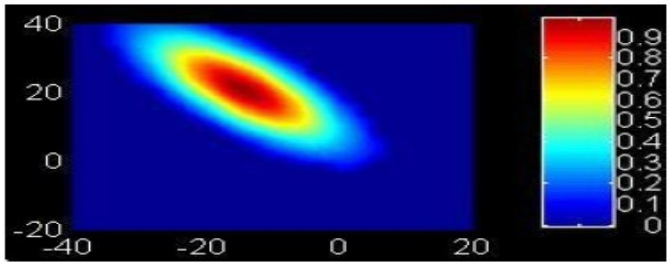

Figure 8. Skin-color probability model

With carefully chosen thresholds $R_{Cr} = [Cr_1 , Cr_2]$ and $R_{Cb} = [Cb_1 , Cb_2]$ that are set on the probability values, a pixel is classified to have skin tone if its values (*Cr*, *Cb*) fall within the ranges

$$Cr_1 \le Cr \le Cr_2 \qquad Cb_1 \le Cb \le Cb_2$$

Otherwise, the pixel is classified as non skin color. This way, a raw binary map is created and then cleaned up by morphological filters (opening and closing by reconstruction) that have the property of simplify the images while preserving contours and subsequently. Thus, a more compact skin-color decision map is obtained (*Figure 7b*) and further processing can be accomplished.

2) Skin-region selection: At this stage, independently from results of skin color pixels detection, all pixels in image are formed to regions of common color characteristics. For region-merging purposes, morphological *watershed algorithm* [18] or RSST (Recursive Shortest Spanning Tree) algorithm [19] can be implemented. This way, the whole image is divided into different color-homogeneous regions and consequently, areas with a minority (less than 50%) of skin-labelled pixels are removed while skin-region selection remains (as seen in *Figure 7c* that demonstrates color-coded surviving regions).

3) Shape-based merge: As the shapes of human faces are elliptic, merging process of skin-color regions continues, until the shape of this new area is approximately elliptic. For that purpose the modified RSST algorithm is used, but now, in addition to color property, it also takes into account shape. Results of shape-based fusion are given in *Figure 7d*, where the face merged into a single region can be observed. But unsophisticated defining a face as an elliptical skin-color region is not sufficient rule because this definition can be applied without any problem to a hand, part of a leg and many other objects of very different nature. That's why the false-positives discarding process follows to decrease the number of false alarms.

4) Rule-based false positives exclusion is the last step of skin color detection method and eliminates false positives by using rules derived from prior knowledge of what features a typical face is formed. Initially, too large and too small candidate ellipses are discarded, as well as those, whose ratio of major axis to minor axis is less than a threshold (elimination of too thin or too wide ellipses). Further, face candidates orientation can be checked (but not necessarily), so ellipses with the major axis orientation, which falls within the range *(-45º, 45º)* are selected only (unfortunately, this rule excludes faces of lying persons). Then, if the facial features are included in the skin-color region, the color of facial features should be relatively darker than the rest of the region, or if the facial features aren't classified as skin-color regions, some empty areas should exist inside the candidate region. All these regions are classified as a human face too. A last discarding rule is based on the analysis of the luminance variance of face, that tries to find a significant luminance distribution of facial surface along the nose axis in comparison with more uniform luminance distribution of background. But, this rule can easily exclude profile faces classified as faces so far.

Generally, there is a great number of similar methods, e.g. Brand and Mason's one [20] using RGB colorspace, or Yang and Ahuja's method [21] could be named. The obvious advantage of skin-color approach is the simplicity of skin detection rules that leads to construction of a very rapid classifier. Although a color-based face detection system may be computationally attractive, the color constraint alone is insufficient for achieving high accuracy face detection. Skin color detection as a model-based

approach is often combined with pattern recognition algorithms, learning from examples (without an explicit formulation of face knowledge) to solve more difficult problems such as detecting multiple faces in complex background. Skin color is frequently used as a pre-processor of the learning-based algorithms that require a computationally expensive multiresolution window scanning process (process inspecting an image with different scales and positions of window). To avoid this expensive task, skin-color detection is applied to find a small subset of face candidates.

2.2.2 Texture-Based Approach

As a human facial surface differs from that of most of nonface objects, detection of faces in the image can be successfully founded on texture-based feature approach. The very texture, as structural property of surfaces, makes the detection system more robust to arbitrary view, color, scale, lighting conditions and facial appearance. Texture features can effortlessly represent and capture regularity, randomness, directionality and graininess properties of patterns. So, the crucial goal of this approach is to discriminate facial texture patterns from the others.

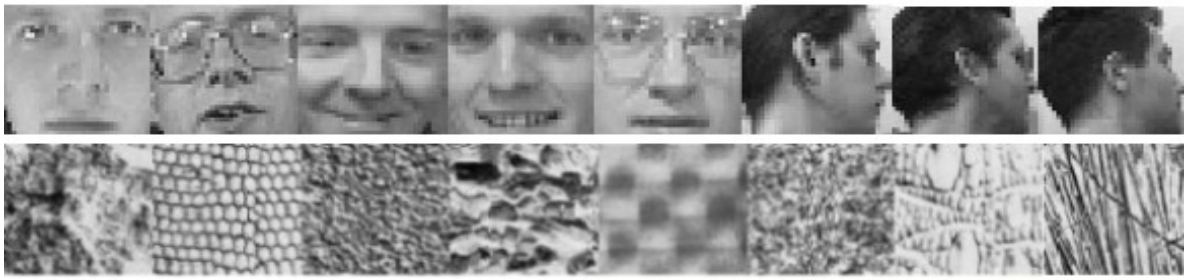

Figure 9. Face and non-face textures

The work carried out by Augusteijn and Skufca [22] has employed the textural features to classification of human faces. Concretely, the facial texture model was based on SGLD (Space Gray Level Dependence) matrices [23] (sometimes also referred to as co-occurrence matrices) that are one of the most popular sources of texture features. They are used for analyses of 16x16 grayscale sub-images to extract information from them. These statistical features can capture the coarseness, randomness and the high frequency edge information inherent in face images. Using a neural network, they were able to train the classifier for texture classification and through Kohonen's Self-Organizing Map (SOM) [24] form clusters for different texture classes. Decision on presence of face in image is based on votes of the occurrence of hair and skin texture.

Japanese researchers Y. Dai and Y. Nakano [5] proposed the similar method for the face detection, also based on SGLD technique. But in contrast to Augusteijn and Skufca, they don't use grayscale images only – furthermore, they tried to incorporate color information to their face-texture model that is used to detect faces in the color complex backgrounds. By transforming color images from RGB color representation to *YIQ color* one [25] and utilizing its I-component, the orange-like regions, where the faces are present, are enhanced in the

original images. The facial texture model based on the space gray level dependence matrices is applied to these images. Using this model, facial parts are detected as those regions which satisfy a set of inequalities. The weight coefficients in the inequalities are decided by the conventional method of learning. Using this texture model, they designed a kind of scanning scheme for face detection in the complex backgrounds and the experiments show that this method can locate faces effectively (60 images containing 150 faces used, the error rate 0% with the false alarm rate 5.4%).

One attractive aspect of texture-based approach is that it can detect faces which are not upright or have special features such as beard and glasses. It can also find faces in complex images with manifold background. Also, the algorithms use fewer number of features and require less training samples than other learning based methods. On the other hand, smoothed images may confuse the algorithms, consequently faces couldn't be detected properly. Finally, texture-based methods don't successfully operate on face sketches or cartoon images.

2.2.3 Facial Features

K. C. Yow and R. Cipolla [26] [6] presented a feature-based face detection algorithm that works in a bottom-up fashion from low-level image features to high-level model features. The algorithm extracts feature points using spatial filtering techniques, then groups these feature points into face candidates by using perceptual grouping principles and finally, applies a probabilistic framework for the selection of candidates.

1) The face model: Yow and Cipolla modelled face as a plane with 6 oriented facial features (two eyebrows, two eyes, nose, and mouth), but due to missing features (e.g. eyebrows), the face model is decomposed into components consisting of 4 features only, which are common occurrences of faces under different viewpoints. These groups are called *Partial Face Groups (PFGs)* and are further subdivided into components consisting horizontal and vertical pairs (Hpair and Vpair) for purposes of perceptual grouping. Different component groups are illustrated in *Figure 10*.

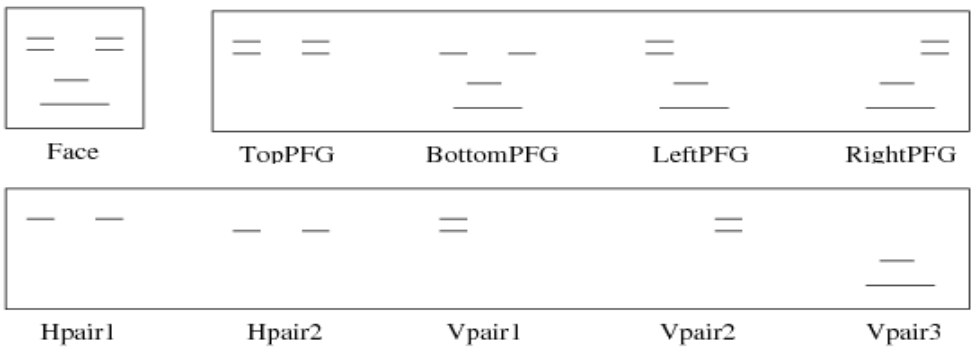

Figure 10. Yow and Cipolla model a face as a plane with 6 oriented facial features (eyebrows, eyes, nose, and mouth)

Using low resolutions, all the six oriented facial features appears as dark oblong spots. As the

edges are illumination unchangeable, each facial feature can be modelled as pairs of oriented edges (as seen in *Figure 11*). Horizontal edges are more important criteria in the detection of the feature than vertical edges.

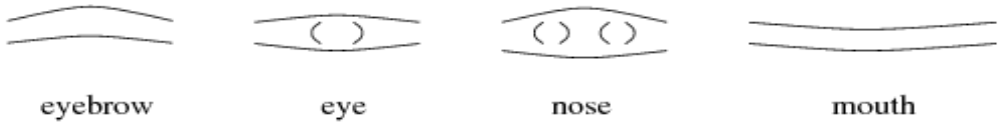

Figure 11. The facial feature model
(each feature is modelled as pairs of oriented edges)

2) Perceptual grouping: For purposes of decreasing false positive feature candidates, a perceptual grouping framework is used. At the first stage, an image is scanned to detect points and regions of interest (*Figure 12*).

Figure 12. Feature Selection Process

Subsequently, grouping, evaluation and reasoning activities based on detection of meaningful object groups are performed. Thus, feature candidates are grouped into faces (*Figure 13*) using geometrical, gray-level and spatial information. Feature candidates that can't be grouped will be excluded.

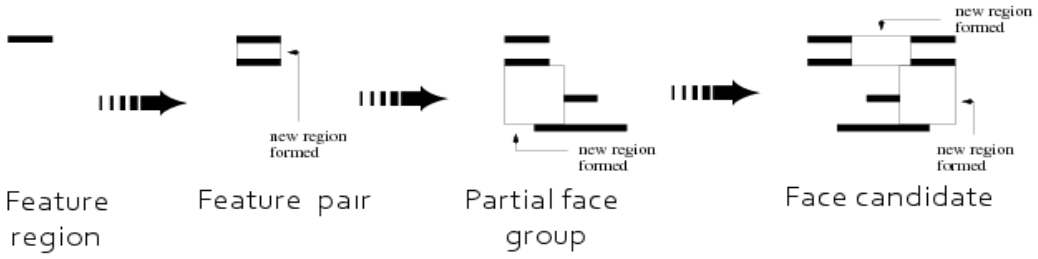

Figure 13. Attentive Feature grouping Process

3) Evidence propagation: Each grouping is evaluated using *Bayesian Networks* [27] to discard still a high number of false positives. These belief networks were proposed for defining the probability of each face candidate.
One advantage is that this method can detect faces at different orientations and poses. The overall detection rate on a test set of 110 images of faces with different scales, orientations and viewpoints is 85 %, however, the reported false detection rate is 28%. Indeed, there is much great number of other

successful approaches based on facial features.

Leung et al. [28] developed a probabilistic method based on local feature detectors and random graph matching. Their goal is to find the constellation of certain facial features (two eyes, two nostrils, and nose/lip junction) that are most likely to be a face pattern. Finding the best constellation is formulated as a random graph matching problem in which the nodes of the graph correspond to features on a face and the arcs represent the distances between different features. Classification of the constellation (whether it was face or non-face) is based on a probability density function. This system is able to achieve a correct localization rate of 86 %.

Sirohey [29] incorporated *Canny detector* to the detection system to build-up edge map. Consequently, heuristics were used to discard redundant edges while keeping only the ones that form the face contour. Then, an ellipse is fit to the boundary between the head region and the background (method applied for face identification in a cluttered background). This algorithm achieves 80% accuracy on a database of 48 images with cluttered backgrounds.

Amit et al. [30] suggested a system for shape detection that incorporated a simple edge detector to scan an image for spatial arrangements of edge fragments. A rich family of such spatial arrangements, invariant over a range of photometric and geometric transformations, is defined. Then, intensive classification is performed by using training face images and particular spatial arrangements of edges in typical faces. This method for shape detection was successfully applied to detect frontal-view faces in still intensity images.

Graf et al. [31] implemented band pass filtering and morphological operations on gray-scale images to enhance regions of interest. Through evaluation of image histogram, threshold values are obtained and two binary images generated. In both binary images, connected components are identified to find the locations of candidate facial features. Combinations of such locations are forwarded to classifiers and consequently evaluated here to determine whether and where a face is present.

2.2.4 Multiple Features

There is a group of methods combining several facial features (skin-color, shape, size) to detect faces. Through the first step, most of them try to find facial region candidates based on human color segmentation (methods similar to that one described in section 2.2.1). In the second step, detailed facial features are found and used for verification of facial candidates. The representative solution by Zhang and Izquierdo is described in the sequel.

Q. Zhang and E. Izquierdo [7] represented *multi-feature based face detection* comprising of two steps. In the first step, a-priori knowledge about human skin colors is used to extract regions of potential faces (the most common ethnic groups were included - Caucasian, Asian and African). Skin detection is done by using a statistical approach in which skin-colors are approximated by a Gaussian mixture model (GMM) [32] in the YCbCr color space.

Figure 14. Selected skin regions detected by skin detection module

After the first processing step, faces are detected among a large number of candidate regions (as shown in *Figure 14*) by discarding false positives from the skin color detection module. This second step is based on the definition of a suitable metric in multi-feature space. The primitives including the Edge Histogram Descriptor (EHD) [33], the texture descriptor defined by the Grey Level Co-occurrence Matrix (GLCM) and the Hue-Saturation-Value Histogram (HSV) were taken into consideration. For each descriptor space *i* and feature vectors v_1, v_2 in this descriptor space, a distance function $d_i(v_1, v_2)$ is available. Then, the most straightforward candidate of possible metrics combining all single distances *d,* is their linear combination:

$$D(A) = \alpha_1 d_1 + \alpha_2 d_2 + \alpha_3 d_3$$

where $A = \{\alpha_1, \alpha_2, \alpha_3\}$ is the set of weighting factors and d_1, d_2, d_3 are normalized distances. Further, the optimization of parameters of the equation above is performed. Thus, the metric is derived from combination of these three low-level descriptors and thereafter optimized for face detection. Finally, the K-Nearest Neighbourhood (KNN) search is employed as a simple binary classification to test the introduced metric. In addition the popular Support Vector Machines (SVMs) are also used [34]. Despite the reduced training set the improved SVM deliver very promising results and outperforms the KNN based classification. More details can be found in [7].

Another successful method was proposed by Yachida et al. [35]. This time, skin and hair color distributions are modelled in CIE XYZ color space and described by *fuzzy models*. Then, the appearance of faces in images is simulated by 5 head-shape models (1 frontal, 4 side-view models) that are represented by a 2D pattern of *m x n* square cells. Each cell may contain several pixels and each pixel is described by two values: the skin proportion and the hair proportion (ratios of the skin or hair area within the cell to total area of the cell. On the basis of distribution models, pixels in test images are classified as face/hair/background and thus, skin-like and hair-like regions are extracted and subsequently compared with head-shape models. In case the similarity found, the detected region becomes a face candidate. Furthermore, detection of horizontal edges is performed to extract facial features (eyes/eyebrows, nose/mouth) to confirm or reject face candidates.

2.3 Template Matching

Among face detection techniques, template matching is well known to be an expensive

operation when classifying against large training sets of images. For this purpose, it uses several standard templates of a face that are stored to describe the face as a whole or the facial features and their relations separately (subtemplates for the eyes, nose, mouth). The goal is to loop through all the pixels in the search image and compare them to the *patterns*. The correlations between an input image and the stored patterns are computed for detection. As the execution time is directly proportional to the size of the training set, the problem can be solved by reduction of the size of the set against which new images are compared, or by reducing the number of image windows that need to be compared against the set. Also, decrease in number of operations associated with correlation is possible by using alternatives such as multiresolution-pyramids that manage with lower resolutions of both template and image during the first iterations of calculation. While the entire approach is quite simple to implement and understand, it is one of the slowest methods. Also, it's difficult to enumerate templates for different poses (similar to knowledge-based methods).

2.3.1 Predefined Templates

A general framework for face detection using predefined templates can be based on template matching of an average face representing a training set of images. The proposed algorithm consists of mainly three parts, of which one is a skin color segmentation for color images, template generation and the last step is template-based face detection in gray level images.

1) Skin color segmentation: In this step, fast algorithms for skin color segmentation (for more details, see section 2.2.1) are performed to group pixels into regions in which faces are to be searched and reject as much non-face objects as possible. Mainly, there are two common ways of segmenting the image based on skin color: converting the RGB picture to YCbCr space or to HSV space. A YCbCr space segments the image into a luminosity component and color components, whereas an HSV space divides the image into the three components of hue, saturation and color value.

$$
\begin{aligned}
Y' &= 0.257 * R + 0.504 * G + 0.098 * B + 16 \\
Cb &= 0.148 * R - 0.291 * G + 0.439 * B + 128 \\
Cr &= 0.439 * R - 0.368 * G - 0.071 * B + 128
\end{aligned}
$$

YCbCr conversion from digital 8-bit RGB (R, G, B in {0, 1, 2, ..., 255})

After skin color classification, the binary mask is refined through morphological operations to reduce the background contribution and remove holes within faces and other artefacts in the image. A few thresholding techniques are applied to reduce number of face candidates based on standard variation of the face color. Thus, face candidates are separated and prepared for further processing.

2) Template generation: Once the face candidates are separated within windows, template generation is performed. For purposes of template matching that computes cross-correlation, both a subimage and the template (acting as a representative face) are needed. Therefore, an average face of all faces in the training set can be created. Diverse faces (full, frontal and upright faces chosen only) are cropped out from each training image and resized to average

size, and then the mean of faces is calculated, using the standard definition of average value

$$mean = \langle m \rangle = \frac{1}{N} \sum_{n=1}^{N} m^n$$

where N is number of samples. Some examples of cropped faces and mean face are shown bellow in *Figure 15*. Besides using the average images, the template set can be comprised of *eigenfaces* based on the Sirovich-Kirby method [36] [37] (eigenfaces mentioned and described in details later in subsequent sections).

Figure 15. The average face derived from face examples

With a view to increasing a detection rate, the face detection system should deal with different head poses (including different scales of the face and its rotations in image plane). That's why several groups of templates should be generated via affine transformations with rotation and stretch (shown in *Figure 16)* to incorporate the majority of such like cases.

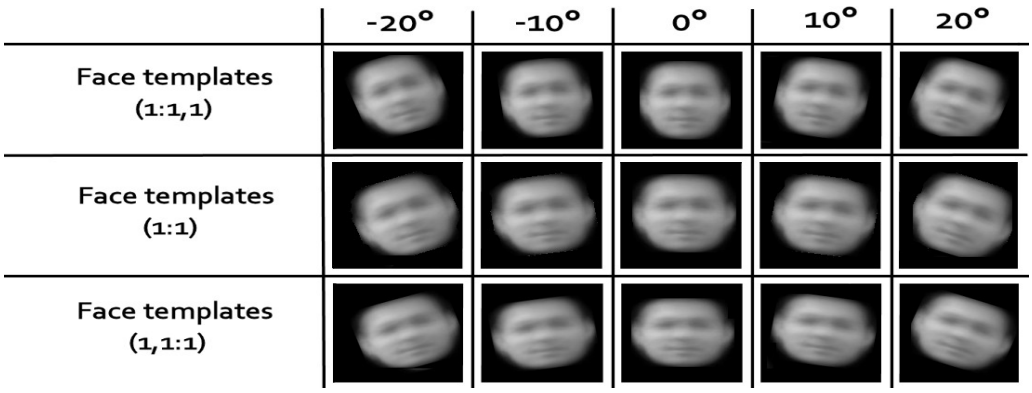

Figure 16. Face templates of various poses and scales

3) Template matching: Template matching is used as not only a final detection scheme for faces, but also for locating the centroid of the face. As the brightness of the image and template can vary due to lighting end exposure conditions, the images can be first normalized. The algorithm applies region based similarity metric such as normalized cross-correlation [38] of a subimage with the scaled template that should be a representative face: given a template *T[M][N]* with the intensity average μ_T and squared difference σ_T, an image region *R[M][N]* with μ_R and σ_R, the correlation coefficient *r(T,R)* is

$$r(T,R)=\frac{\sum_{i=0}^{M-1}\sum_{j=0}^{N-1}(T[i][j]-\mu_T)(R[i][j]-\mu_R)}{M*N*\sigma_T*\sigma_R}$$

The matching algorithm is composed of several procedures. First of all, for each subimage a face candidate list is initialized. At each point of subimage, face templates of various poses are matched, selecting the maximum one from those over a threshold and passed verification. The maximum one is then compared with those in the face candidate list, and if there is no overlapping region found, it's put in the list, otherwise the bigger is replaced by the smaller. The subimage is subsampled by specific ratio and the process is repeated with the same candidate list until a specific size is reached.

4) Verification and arbitration: Moreover, the detection system could employ a different strategies in parallel, such as the implementation of a neural network classification or a linear classifier as a secondary detection scheme to achieve higher accuracy. The whole system illustrated in *Figure 17* bellow.

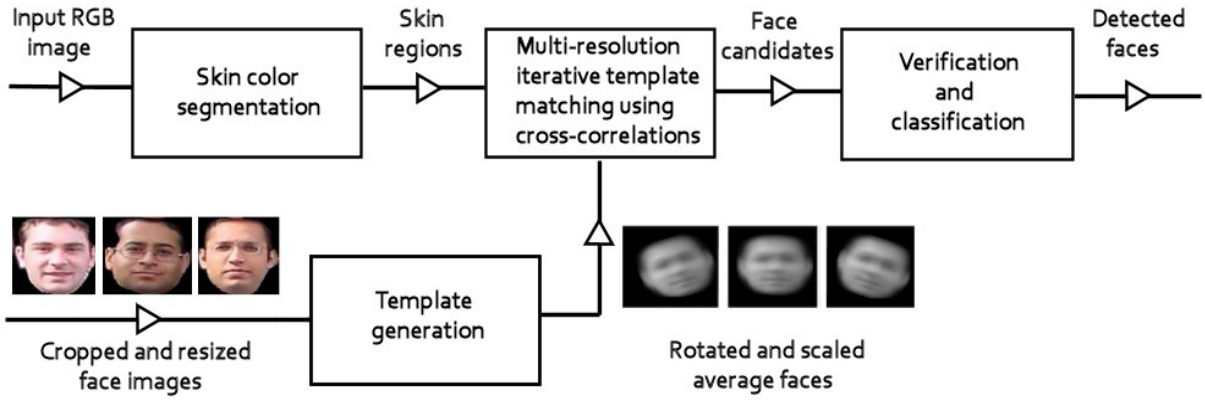

Figure 17. Detection system implementing template-based approach

In experiments [39] the detection rate of the similar systems can achieve about 95% with 4% false positives. The result demonstrate that the systems work extremely well for the images whose faces are full, upright, and facing towards the front. The false positives primarily occurred on large non-face body parts such as arms and legs. The false negatives were typically due to an obstructed face or variations that caused separations in the face such as sunglasses.

There is a great number of other template-based methods and great research effort was made in the sphere of template matching. An early attempt to find frontal faces is reported by Sakai et al. [8]. They implemented several subtemplates for the eyes, nose, mouth, and face contour to model a face. Each subtemplate is defined by using line segments. Lines in the input image are extracted based on greatest gradient change and then matched against the subtemplates.

Miao et al. [40] presented a hierarchical template matching method for face detection. Initially, input images are rotated within the range (-20º, 20º) to deal with faces with diverse rotations. Subsequently, by using the Laplacian operator, a multiresolution image hierarchy is

formed and edges are extracted. The face template is composed of the edges produced by six facial features: two eyes, two eyebrows, one nose and mouth. Finally, heuristics are applied to determine the existence of a face.

Craw et al. [41] [42] suggested a method based on a shape template of a frontal-view face. He implemented a Sobel filter to extract edges that were grouped together to search for the template of a face based on several constraints. First of all, the head contour is located, then the same process is repeated at different scales to find other features such as eyes, eyebrows, and lips.

Samal et al [43] reported a human face detection method using silhouettes. They implemented system based on Principal Component Analysis (PCA) of training images to get eigen-silhouettes that are used with a generalized Hough transformation in order to locate faces.

2.3.2 Deformable Templates

Though face detection based on predefined templates is quite simple, the use of this approach is restricted to detection of frontal faces. The limitation is due to rigidity of templates, so a parametric shape-model known as deformable template was proposed to model facial features more precisely. Goal is to deform the predefined templates, and subsequently try to match deformed template and a given image. These operations are iteratively executed until the best fit of the template is found.

One of the earlier approaches to deformable template analysis was introduced by Yuille et al. [44]. This was aimed to fit the feature of interest (eye or mouth) described by parametrized template to a human face using a deformable template model. For example, the approach considered an eye to be comprised of an iris that sits within the sclera and which can be modelled as a combination of circle that lies within a parabola (described by parametric curves). Next, an appropriate metric – an energy function is defined which links edges, peaks, and valleys in the image intensity to corresponding properties of the template. The template then interacts dynamically with the image by altering its parameter values to minimize the energy function, thereby deforming itself to find the best fit. Thus, curves try to follow the outline of the facial features and final shapes can be used as descriptor for the feature and verification factor whether the observed object is an eye, lip, or face. Problem with this technique is its relative dependency on position and lighting.

Besides the work done by Yuille et al., variations on the theme of deformable templates have quite rapidly emerged in recent years, including the work of Cootes et al. [9]. To overcome variations of the shape instances, they proposed active learning of shape models. This is called *Active Shape Model (ASM)* [45]. For shape modelling the polygonal representations are used: initially, in several training images contours of facial features such as eyes, eyebrows, chin, cheeks, nose and mouth are manually labelled with landmark points - nodes (shown in *Figure 18* below). Correspondences between the landmark points of training set are established by manually aligning the training set. Then the mean position and variation of each node from the training shapes is evaluated. A mean shape is used as the generic template (prototype) of the class of shapes and eigenvectors of the covariance matrix provide the deformation from the generic shape. This approach is sensitive to partial occlusion and unable to handle large scale and orientation changes.

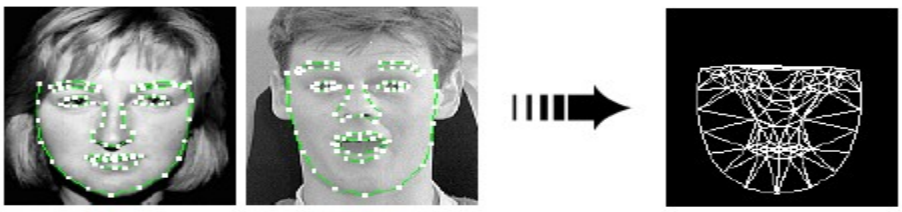

Figure 18. Training images manually labelled with landmarks and a final mean shape

Shuicheng et al. [46] extended ASM in their paper and proposed a novel shape model, called Texture-Constrained Active Shape model (TC-ASM) to localize a face in an image. TC-ASM effectively incorporates not only the shape prior and local appearance around each landmark, but also the global texture constraint over the shape. Therefore, it performs stable to initialization, accurate in shape localization and robust to illumination variation, with low computational cost. Extensive experiments show that TC-ASM outperforms ASM in facial shape localization.

Besides, detection system can be grounded on *snakes* [47] (the seminal research done by Kass et al.) and template. In their proposed model, a snake is an elastic contour that is fitted to the features detected in an image. Another example of this is a work of Lam and Yan [48] using snakes to locate the head boundaries with a greedy algorithm in minimizing the energy function.

2.4 Appearance-Based Approaches

In contrast to preceding methods, appearance-based methods use no predefined templates or a priori knowledge of characteristics present in image. Instead, usually two phases are involved: a *training phase* and a *classification phase*. Initially, in the training phase they incorporate the statistical analysis of the available dataset (image database composed of manifold face examples) and learning procedure to extract crucial characteristics from the database. For modelling these learned characteristics, either a probabilistic framework using class-conditional density functions $p(x|face)$ and $p(x|nonface)$ (where x is random variable representing high dimensional feature vector derived from image), or discriminant functions (i.e. decision surface, threshold function) between face and nonface classes are used. One of these two models is incorporated in the classifier that can deal with the variability of facial appearance and then, classification procedure is performed. Thus, each scanned window is arbitrated to be a face or nonface. Appearance-based methods are fast and robust approaches with good empirical results, and can detect faces of various poses and rotations. But they need large sets of positive and negative examples and need to search over the space and scales.

2.4.1 Support Vector Machines

As a subset of Kernel-based techniques, the S*upport Vector Machines (SVM)* [49] [50] is a

group of supervised learning methods that can be applied to classification or regression and are based on the principle of *Structural Risk Minimization* [51], which aims to minimize an upper bound on the expected generalization error.
This new statistical learning technique was developed by Vladimir Vapnik [51] and it was originally worked out for linear two-class classification with margin, where margin means the minimal distance from the separating hyperplane to the closest data points. SVM learning machine seeks for an optimal separating hyperplane, where the margin is maximal. An important and unique feature of this approach is that the solution is based only on those data points, which are at the margin. This small subset of the points is called support vectors. Firstly, training data (*Figure 19a*) are mapped by using a set of nonlinear basis functions into a higher-dimensional feature space (*Figure 19b*) where the data points can be linearly separated - a separating hyperplane with maximum margin is constructed there. This yields a nonlinear decision boundary in input space (see *Figure 19c*). An important advantage of the SVM is the use of the kernel functions to compute the separating hyperplane without explicitly carrying out the map into the possibly very-high dimensional feature space.

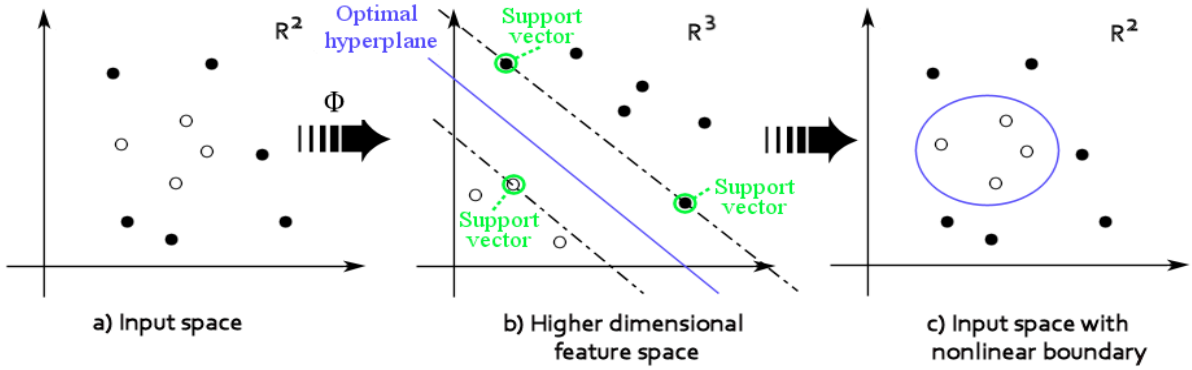

Figure 19. a) The training data are mapped into a higher-dimensional space via Φ b) separating hyperplane is construct c) and non-linear boundary is found

The first attempt to apply Support Vector Machines for face detection is reported by Osuna et al. [10]. They introduced an SMV application for detecting vertically oriented and unoccluded frontal views of human faces in gray-level images. Their application handles faces over a wide range of scales and works under different lighting conditions, even with moderately strong shadows.

In the training phase, Osuna et al. used a database of face and nonface *19x19* pixel patterns, assigned to classes +1 and –1, respectively, and they applied the support vector algorithm that employs a second-degree homogeneous polynomial kernel function and upper bound $C = 200$ to obtain a perfect training error. To compensate for certain sources of image variation, they performed some preprocessing of the data. First, an oval binary mask is applied to remove some pixels close to the window-pattern boundary, and thus, perform a reduction in the dimensionality of the input space from 19x19 = 361 to 283. Then, illumination gradient correction is performed correcting for some extreme lighting conditions. Finally, histogram equalization is applied to improve contrast.

As the negative examples are very difficult to characterize and define, and the nonface class is much broader and richer than face class, more negative examples are needed to get an accurate definition that separates these two classes in terms of binary pattern classification. For this purpose, once the process obtains a decision surface through training, *Bootstrapping procedure*, which is very important in the context of a face detector learning from examples, is performed: Images that do not contain faces (images of landscapes, trees, buildings, rocks etc.) are great sources of false positives (an example can be seen in *Figure 20* below) and in subsequent training phase they are stored as negative examples. Therefore, within a small set of the support vectors, there is a much higher proportion of those corresponding to nonfaces (illustrated in *Figure 20*).

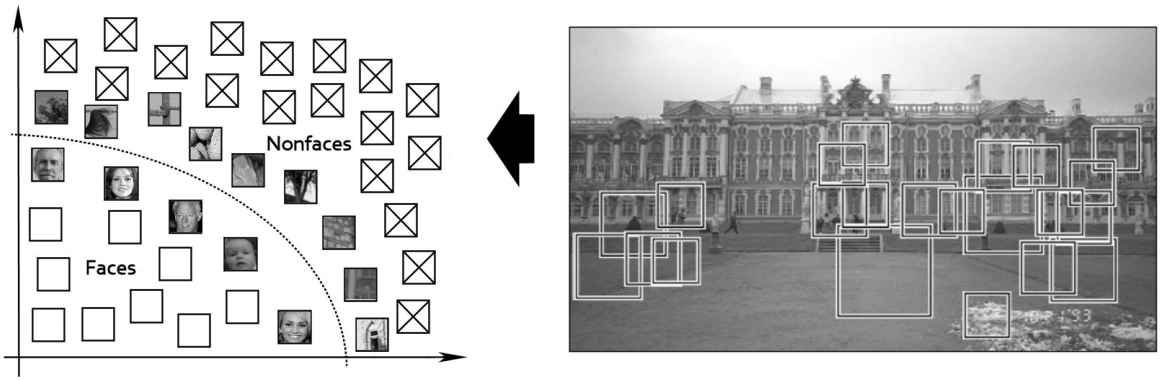

Figure 20. Bootstrapping: In the training phase, detector is run on images that do not contain faces to obtain more nonface examples

After the training SVM, classification phase is performed. The system detects faces by exhaustively scanning an image for face-like patterns at many possible scales, by dividing the original image into overlapping subimages and classifying them using an SVM as a face or nonface. The input image is rescaled several times through subsampling, then,19x19 window patterns are cut out of the scaled image and preprocessed using masking, light correction and histogram equalization. Finally, the pattern is classified by using SVM. System illustrated below.

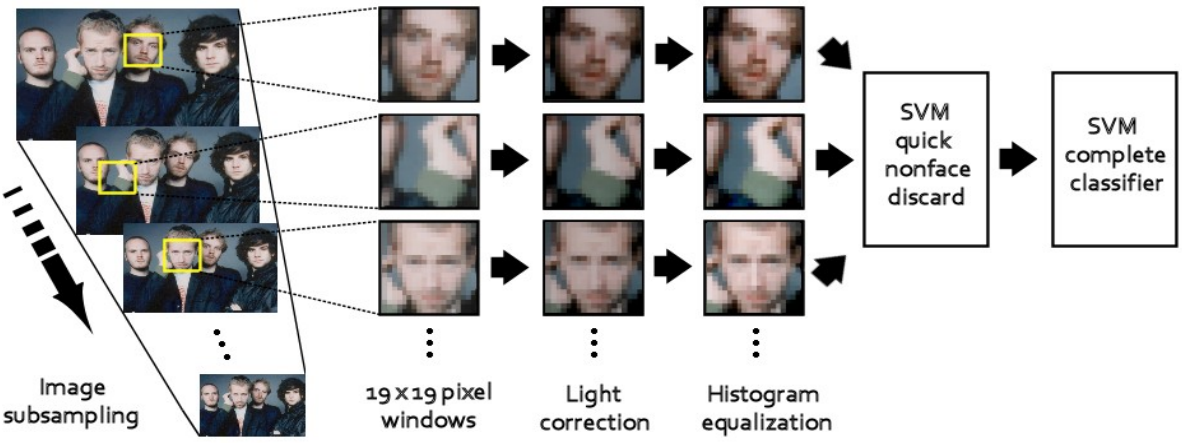

Figure 21. Architecture of the system based on SVM

To speed up the whole detection system, training the SVM classifier involving the skin-color is possible. Experimental results show a great performance of the SVM face-detection system. When a set A of 313 high-quality images with the same number of faces was classified, system reached 97% detection rate with 4 false positives. By using a set B containing 23 images of mixed quality with a total of 155 faces, system achieved 74% detection rate with 20 false positives.

Yongmin Li et al. [52] presented a novel approach using SVM to detect faces across multiple views. The view sphere is separated into several small segments. On each segment, a face detector is constructed. Initially, they explicitly estimate the pose of an image regardless of whether or not it is a face. A pose estimator is constructed using SVM Regression. Then, the pose information is used to choose the appropriate face detector to determine if it is a face. With this pose-estimation based method, considerable computational efficiency is achieved. Meanwhile, the detection accuracy is also improved since each detector is constructed on a small range of views. Finally, they developed a hybrid algorithm combining the Eigenface and SVM methods for face detection which provides improved performance in terms of accuracy and speed.

2.4.2 Neural Networks

In terms of face detection, an *Artificial Neural Network (ANN)* [53] [54], also abbreviated to neural network (NN), belongs to very popular and successful approaches and is implemented in numerous systems. ANN is an information processing paradigm that is inspired by the way biological nervous systems process information. It can be used to model complex relationships between inputs and outputs or to find patterns in data. The information processing system is composed of a large number of highly interconnected processing elements, *neurones*, working in parallel to solve specific problems (example of NN with interconnected group of nodes illustrated in *Figure 22*). The ANN can be configured for a specific application, such as pattern recognition or data classification, through a learning process. Learning in ANNs, as well in biological systems like the human brain, causes changes in their structure and involves adjustments to the connections that exist between the neurones.

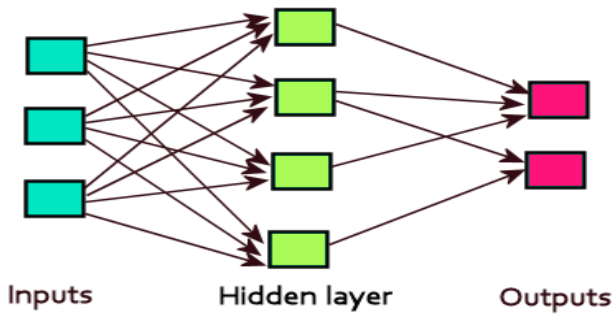

Figure 22. An example of a simple network, commonly comprising of three groups (layers)

The major advantage of using neural networks for face detection is the feasibility of training a system to capture the complex class conditional density of face patterns. If the model and learning algorithm are selected appropriately, the resulting ANN can be extremely robust. On the other hand, the network architecture has to be extensively tuned (number of layers, number of nodes, learning rates, etc.) to get the correct implementation with exceptional performance. Selecting and tuning an algorithm for training requires a significant amount of experimentation. As the neural networks learn by example, examples must be selected carefully otherwise useful time is wasted or even worse the network might be functioning incorrectly. Also, too complex models may cause problems with learning.

In terms of neural networks used in face detection, the most significant work was done by Rowley, Baluja and Kanade [11]. Their system is able to detect upright frontal faces in gray-scale images and applies one or more neural networks directly to portions of the input image. The use of arbitration between multiple networks significantly improves the accuracy of their detector. The system operates in two stages:

1) A neural network-based filter examines each position in the image at several scales in order to find all possible locations that might contain a face. Firstly, the input 20x20 pixel region of the image is preprocessed; the process is similar to the one [10] described in section 2.4.1: initially, an oval binary mask is applied to ignore the pixels close to the boundary that are representing background and then, brightness correction and histogram equalization can be performed, using only pixels inside an oval region in the window (intensity values of pixels outside the oval region are ignored during computation). Subsequently, the preprocessed window is passed through a neural network. The network has retinal connections to its input layer. There are three types of hidden units: 4 which look at 10x10 pixel subregions and 16 which look at 5x5 pixel subregions (these square fields allow to detect such features as eyes, the nose, or corners of the mouth), and 6 which look at overlapping 20x5 pixel horizontal stripes of pixels (for detection of mouth or pairs of eyes). The network has a single, real-valued output - a score within the range <-1, 1>, indicating whether or not the face is present in the window. The whole system illustrated in *Figure 23* below.

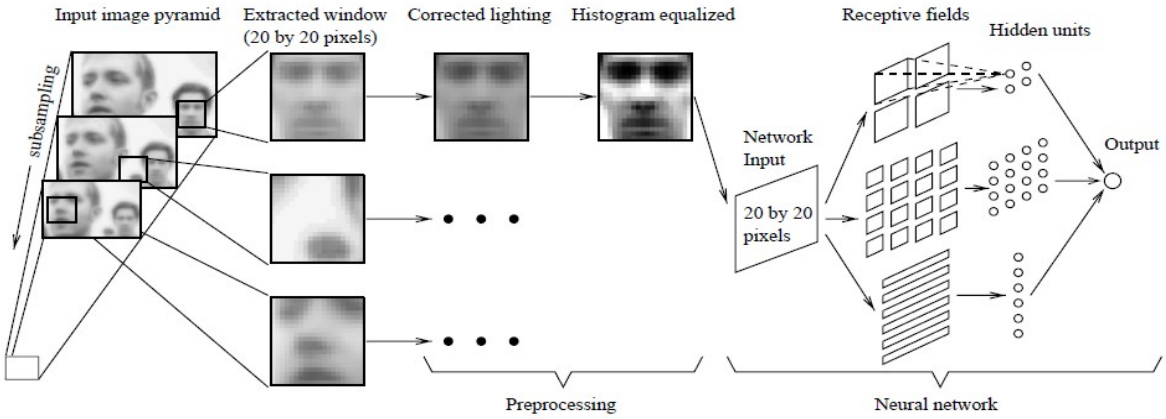

Figure 23. A neural network-based upright frontal face detection system (image taken from [54])

As mentioned in section 2.4.1 above, since the space of nonface images is much larger and more difficult to characterize than the space of face images, the *bootstrapping method* is used to collect a representative set of nonfaces and build an accurate filter. Instead of collecting the images before training is started, the images are collected during training. The system is run on an images of scenery which contains no faces and up to 250 of subimages, in which the network incorrectly identifies a face, are selected. Then, the preprocessing steps are applied and subimages are add into the training set as negative examples.

2) Merging overlapping detections and arbitration among multiple networks improves the reliability of the detector. Each network is trained in a similar manner, but with random initial weights and nonface images, and also uses self-selection of negative training examples. Because of these varying training conditions, the networks will have different biases and will make different errors. Therefore, simple arbitration schemes such as logic operators (AND/OR) and voting are used to reduce the number of false positives and improve the performance. To further eliminate many false positives, merging overlapping detections can be performed. If the number of detections within a specified neighbourhood is above a threshold, then that location is classified as a face.

Experiments shown the system built by Rowley et al. is able to detect 90% of the faces over a test set of 130 complex images, with an acceptable number of false positives. Although the system was trained only on real faces, it can detect some face sketches or cartoon images too. Later, Rowley et al. extended their method to detect rotated faces using a router network [55]. However, detection rate on upright faces had decreased, compared to the previous upright detector.

There is a great number of other neural network-based systems. An early attempt using neural networks was proposed by Agui et al. [56]. Also, Juell et al. [57] presented a hierarchical neural network for human face detection and Lin et al. [58] suggested a detection system using network with modified learning rules and probabilistic interpretation.

2.4.3 Eigenfaces

Eigenfaces [59] are the set of *eigenvectors* [60] of the covariance matrix computed from the face images in the training set. The idea of employing eigenfaces within this sphere was motivated by an early method reported by Kirby and Sirovich [37]. They involved a mathematical tool called *Principal Component Analysis* [61] (PCA, a technique used to reduce multidimensional data sets to lower dimensions for analysis) in order to efficiently represent the face images. They claimed that images of faces can be linearly encoded and then, approximately reconstructed by using a small collection of weights and modest number of standard images. Later, their work was expanded, and the method using eigenfaces for face recognition was developed by Turk and Pentland [12]. Thus, the first successful example of facial recognition technology was implemented.

Let the face image $I(x,y)$ is two dimensional N x N array of intensity values. The image corresponds to a vector Γ of dimension N^2, or equivalently, a point in high-dimensional (N^2)

space. Images of faces will not be randomly distributed in this huge image space. Therefore, they can be described into lower-dimensionality subspace. For that purposes, Turk and Pentland employed PCA to find the "best" vectors that define the subspace of all face images, called "face space". Each vector is of length N^2 (describes an $N \times N$ image), and is a linear combination of the original face images. These vectors are eigenvectors [60] of the covariance matrix corresponding to the original face images and can be displayed as a sort of "ghostly" faces, called eigenfaces [59] (shown in *Figure 24* below).

Figure 24. Examples of eigenfaces
(taken from AT&T Laboratories Cambridge)

1) Representing face images: Each face in the training set can be accurately represented in terms of a linear combination of up to M eigenfaces. Because of computational efficiency, faces can be also approximated using only M' best eigenfaces (illustrated in *Figure 25*):

Figure 25. Linear combination of best eigenfaces

2) Computation of the eigenfaces: The training set of preprocessed face images $I_1, I_2, ..., I_M$ (the same lighting conditions, faces centred and of the same size) is represented as vectors $\Gamma_1, \Gamma_2, ..., \Gamma_M$. The average vector Ψ of the set is:

$$\Psi = \frac{1}{M}\sum_{n=1}^{M}\Gamma_n$$

Each face differs from average by the vector $\Phi_i = \Gamma_i - \Psi$. Consequently, the covariance matrix C can be calculated:

$$C = \frac{1}{M}\sum_{n=1}^{M}\Phi_n\Phi_n^T = AA^T$$

where $A = [\Phi_1\ \Phi_2\ ...\ \Phi_M]$. As the matrix AA^T is very large ($N^2 x\ N^2$), which is not practical (AA^T can have up to N^2 eigenvectors with the corresponding eigenvalues), first of all, eigenvectors v_i of A^TA ($M \times M$ matrix, $M << N^2$) are computed. Here, the M eigenvalues (for ranking eigenvectors according to their usefulness) of A^TA correspond to the M largest eigenvalues of AA^T. Thus, M best eigenvectors u_i of AA^T are subsequently computed: $u_i = Av_i$. Finally, only M' best eigenvectors (the number chosen heuristically) with M' largest

eigenvalues are kept to define an M'-dimensional face space. In many of the test cases made by Turk and Pentland [12], based on $M = 16$ face images, only $M' = 7$ eigenfaces were usually used.

3) Face detection using eigenfaces: Knowledge of the face space can be used for detection of locations of the faces in images. Initially, given an input image represented as vector Γ, mean-adjusted image $\Phi = \Gamma - \Psi$ is obtained. Then, Φ must be transformed into its eigenface components, i.e. projected into face space. Projection is equivalent to creating the vector $\Omega^T = [\omega_1, \omega_2, \ldots \omega_{M'}]$ of weights ω_i that describes the contribution of each eigenface u_i in representing the input face image. The projection of Φ onto face space is:

$$\Phi_f = \sum_{i=1}^{M'} \omega_i u_i$$

Images of faces do not change radically when projected onto the face space, while the projection of nonface images appear quite different. To detect the presence of face, distance ε_d between the local mean-adjusted subimage Φ and its projection onto face space Φ_f is calculated: $\varepsilon_d = ||\Phi - \Phi_f||$. The distance ε_d is used as a measure of “faceness” and the results of calculating the distances at every location in image form a “face map”. After the face map of the whole image is created, a faces can be detected from the local minima of the face map. An example of projecting images into face space is illustrated in *Figure 26* (face space is simply modelled by 2 eigenfaces u1, u2; in cases (1) and (2) projections are near, so faces are detected, in cases (3) and (4) faces are undetected).

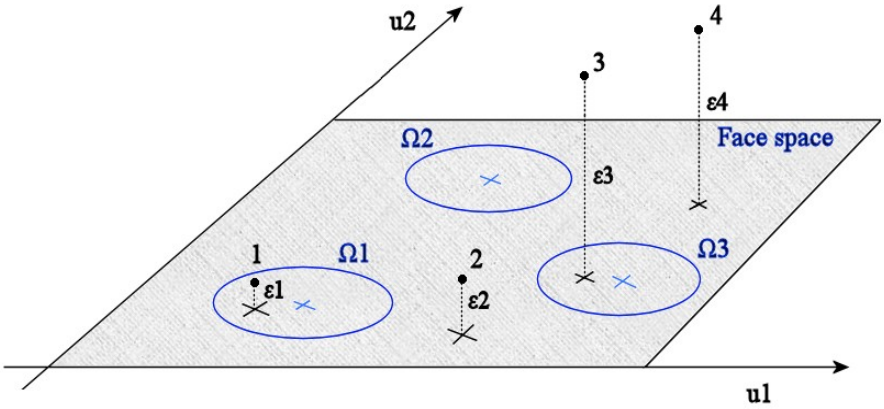

Figure 26. Four possibilities of projections onto the face space:
(1) near face space and face class Ω_1 (2) near face space, but not near a known face class
(3) distant from face space but near a face class Ω_3(4) distant from face space and face class

Eigenfaces happen to be a fairly easy, computationally economical and successful method to determine the presence of faces in images. Using eigenfaces is very fast and able to operate on lots of faces in very little time. Furthermore, face recognition using eigenfaces has been shown to be quite accurate. Unfortunately, this approach can hardly deal with faces that are taken under varying light conditions or angles of view (plane rotations can be handled, out-of-plane rotations are more difficult). For the system to work well, the faces need to be frontal and upright and taken under similar lighting. In spite of that, there are many works on face detection and recognition too that have adopted the idea of eigenvectors and eigenfaces.

2.4.4 Distribution-Based Methods

Distribution-based methods try to fit a distribution model to examples. In general, the examples are projected into reduced dimensional space and then, classifier is built to decide face/ nonface. In terms of this approach, the most significant work was done by Sung and Poggio [13]. They presented an example-based learning approach for locating vertical frontal views of human faces in complex scenes. For purposes of classification, they built a distribution-based face model which was trained using 4150 positive examples of face patterns and 6189 face like non-face patterns. As the negative examples are very difficult to characterize and define, and the nonface class is much broader and richer than face class, more negative examples are needed to get an accurate definition that separates these two classes in terms of binary pattern classification. For that purposes, the samples of nonface patterns are collected by a *bootstrap method* (later used in other works that have been already mentioned in sections 2.4.1 and 2.4.2 above) which selectively adds the negative examples to the training set during the training phase.

First of all, each face and nonface example is normalized and processed to a 19x19 pixel image region that can be represented as a 361-dimensional vector or pattern (a reduction in the dimensionality of the input patterns from 361 to 283 can be achieved again by applying an oval binary mask removing pixels close to the window-pattern boundary). Then, all the

patterns are clustered into 6 face and 6 non-face clusters using the elliptical *k*-means clustering algorithm, and each cluster is approximated by using a Gaussian function with a prototype pattern and a covariance matrix (a set of six Gaussians and their six centroid-patterns are shown in *Figure 27* below).

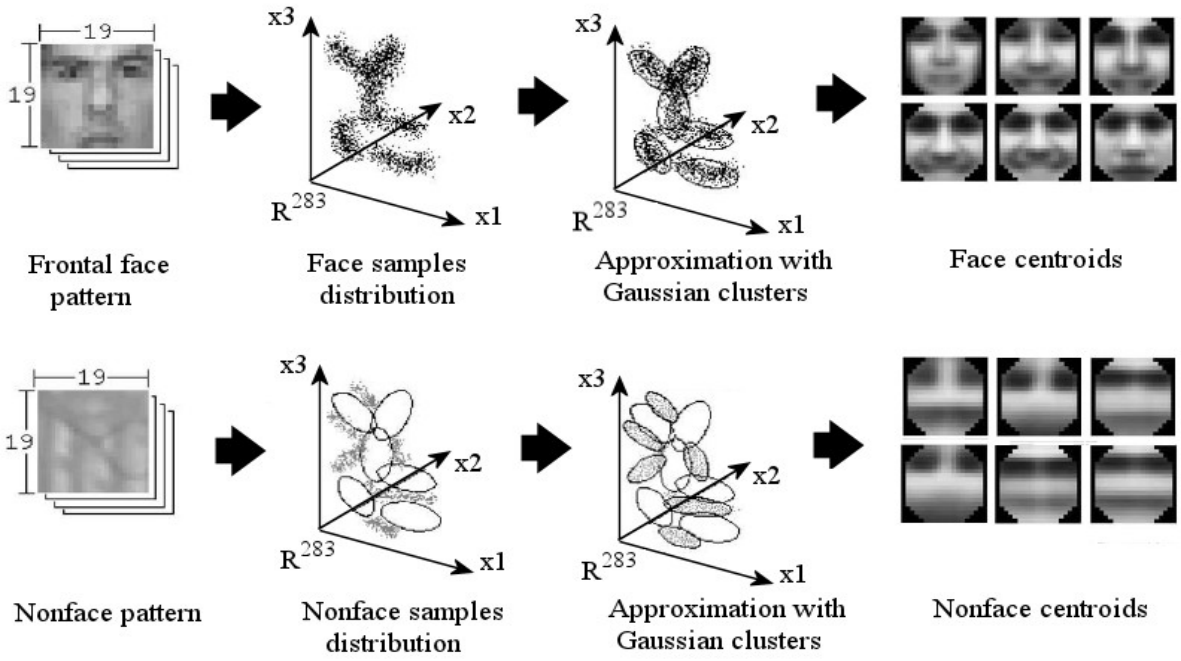

Figure 27. Face and nonface patterns are resized to 19x19 pixel images and represented as a point in 283-dimensional space. Density functions for patterns are estimated by a set of Gaussians.

For testing a new image pattern, Sung and Poggio computed the distances between that test pattern and each face and nonface cluster centroid (*Figure 28*a). This way, a set of 12 distances is obtained. Each distance measurement between the test pattern and a cluster centroid is a two-value distance metric: the first distance component is a *Mahalanobis distance* D_1 between the projection of test pattern and the cluster centroid in a lower-dimensional subspace spanned by 75 largest eigenvectors of the cluster, and the second component is an *Euclidean distance* D_2 between the test pattern and its projection in the subspace (demonstrated in *Figure 28b*).

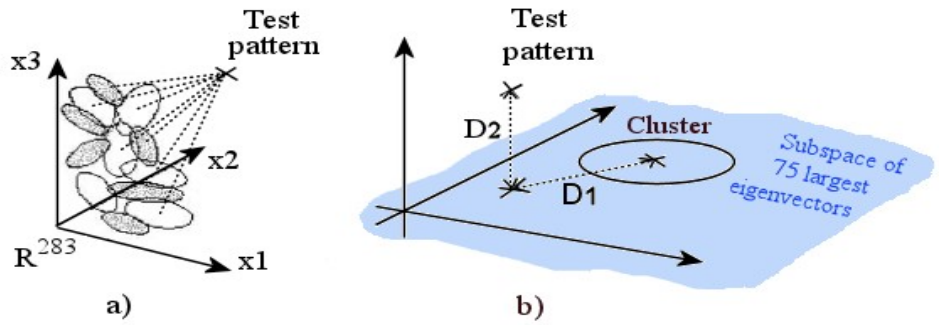

Figure 28. a) Twelve distances have to be computed for test pattern b) Each distance has two components: Mahalanobis distance D_1 and Euclidean distance D_2 computed in a subspace spanned by 75 largest eigenvectors of the cluster

This twelve pairs of distances form a distance vector of 24 values for each test pattern. Finally, the vector is used by a multilayer perceptron (MLP) [62] network that is trained to determine whether the input pattern belongs to the face class or not. This Sung and Poggio's method detects faces by exhaustively scanning an image and is motivating for many other later works.

2.4.5 Probabilistic Modelling of Local Appearance

Schneiderman and Kanade [63] presented the first reliable algorithm that can reliably detect human faces with out-of-plane rotation (besides, they also developed a system for car detection over a range of viewpoints). To deal with variation in pose, they used a view-based approach with multiple detectors that are each specialized to a specific orientation of the face. Then, a statistical modelling within each of these detectors is employed to account for the remaining variation. In contrast to other approaches (SVM, ANN, Distribution-based methods) that model the global appearance of a face, Schneiderman and Kanade applied a *naive Bayes classifier* to estimate the joint probability of local appearance and position of facial subregions (face patterns).

1) View-based detectors: As for face detection, two separate detectors have been developed by Schneiderman and Kanade (for car detection, eight view-based detectors were incorporated). Each of the detectors is specialized to a specific orientation of the object: the first view-based detector is specialized to frontal faces, whilst the second to right profile views of faces (to detect left profiles they take advantage of facial symmetry and apply the right profile detector to a mirrored input images). Detectors are used in parallel, and then their results are combined. In the case of multiple detections at the same or neighbouring locations, the system chooses the strongest detection (see *Figure 29* below, take notice of the woman's face that was detected by two separate detectors).

As described in many other works, to detect faces of various sizes and at any position, the original input image is iteratively rescaled and detector is reapplied for all possible positions of rectangular window. To reduce the execution time of this exhausting process, a heuristic coarse-to-fine strategy is used to speed up this task. Thus, the scanning operation can be done with surprising efficiency. Or, color-heuristics involvement (in case of color images are

available) is very useful for excluding unpromising candidates in the preprocessing stage. However, some faces can be removed by detector if the images are poorly color balanced.

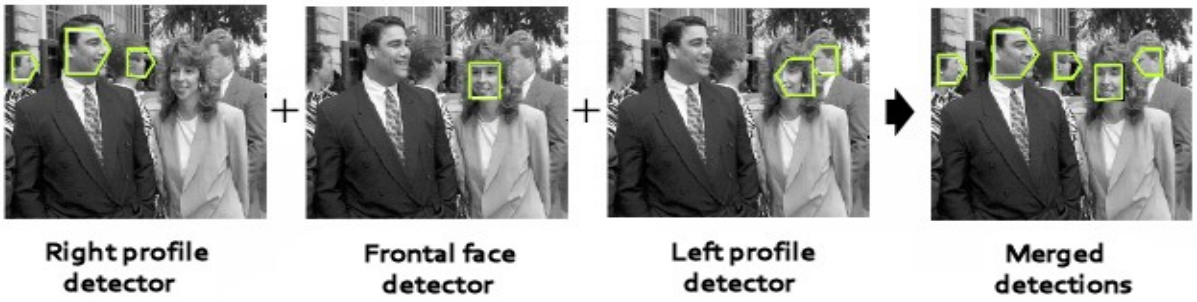

Figure 29. The results from multiple view-based detectors and the final combination

2) Functional form of decision rule: Each of the detectors uses the same underlying form for the statistical decision rule, they differ only in that they use statistics gathered from different sets of images. Two statistics *P(image|object)* and *P(image|non-object)* are modelled and the detection decision is computed using a *Bayesian decision rule* [64]:

$$\frac{P(image|\text{object})}{P(image|\text{non-object})} > \lambda \qquad \left(\lambda = \frac{P(\text{non-object})}{P(\text{object})}\right)$$

If the likelihood ratio is larger than λ, the object (face, car) is decided to be present. For purposes of modelling of those two statistics *P(image|object)* and *P(image|non-object)* Schneiderman and Kanade chose to use histograms. Because of small number of discrete values of histograms to describe appearance, multiple histograms are employed to overcome the problem. For each histogram, $P_k(pattern|object)$ represents the probability of appearance over some specified visual attribute, $pattern_k$. Then, the probabilities from different attributes are combined, so each class-conditional probability function can be approximated as a product of histograms:

$$P(image|\text{face}) \approx \prod_k P_k(pattern_k \mid \text{face})$$
$$P(image|\text{non-face}) \approx \prod_k P_k(pattern_k \mid \text{non-face})$$

In terms of this approach, decomposition of appearance in space, frequency, and orientation is performed to jointly model visual information. By decomposing the appearance of the object into several rectangular subregions, limited modelling power of each histogram is concentrated over a smaller amount of visual information. When decomposing the object spatially, the positions of each attribute sample is represented with respect to a coordinate frame affixed to the object in order to keep all relationships between the various parts. Thus, each histogram now becomes a joint distribution of attribute and attribute position, and the final form of the detector is*:*

$$\frac{\prod_{x,y\in region}\prod_{k} P_k(pattern_k(x,y),x,y \mid \text{face})}{\prod_{x,y\in region}\prod_{k} P_k(pattern_k(x,y),x,y \mid \text{non-face})} > \lambda$$

One example of using local appearance and combining the statistics of subregions to get decision, whether the whole image is face or nonface, is illustrated in *Figure 30* below.

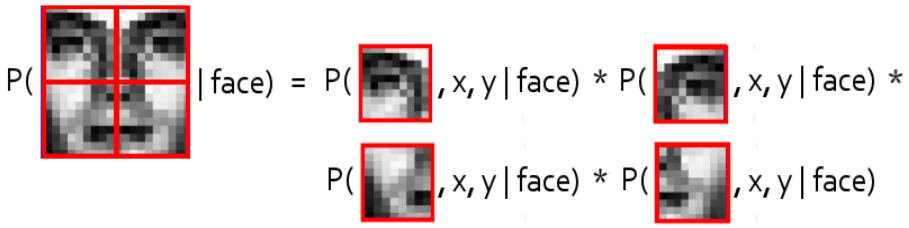

Figure 30. The statistics of each subregion encode local appearance

Next, in *Figure 31* below, an illustrative case of the classification algorithm is presented. To create visual attributes that are localized in space, frequency, and orientation, Schneiderman and Kanade needed to be able to easily select information that is localized along these dimensions, and for that purposes, wavelet transform of the image is performed.

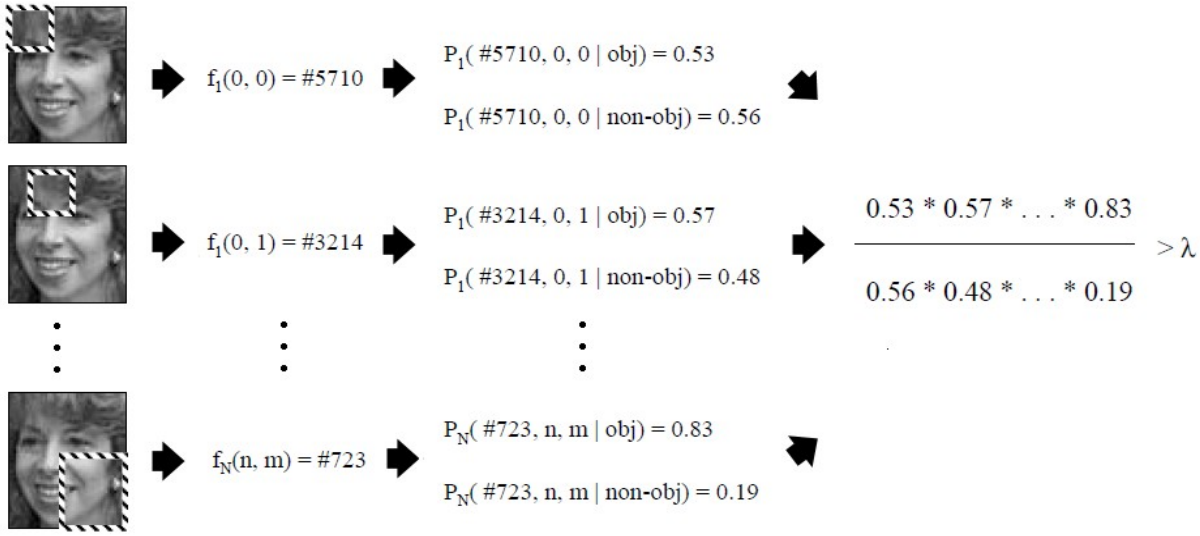

Figure 31. Classification algorithm: N local operators evaluate the image window at n x m locations, class-conditional probabilities are retrieved for each output and combined in a likelihood ratio test (image taken from [68])

The Schneiderman and Kanade's system can detect human faces quite accurately and reliably [14] and it is one of the few successful systems (analogous to e.g. Neural network-based method of Rowley et al. [55], section 2.4.2, or multi-view AdaBoost-based approach of Viola and Jones [75], section 2.4.6) that can properly deal with both frontal and rotated or profile faces.

2.4.6 AdaBoost-Based Detector

Viola and Jones [65] [66] presented a successful machine learning approach for frontal face detection that is able to process images quite rapidly and can achieve high detection rates comparable to the best previous systems. They claimed that their real-time system is approximately 15 faster than any previous approach. Viola and Jones introduced a new image representation called the *Integral Image* through which the rectangle features used by face detector can be computed very quickly. Learning algorithm is based on *AdaBoost* [67] that selects a small number of critical visual features from a larger set and yields extremely efficient classifiers. Besides, more complex classifiers are combined in a cascade in order to quickly discard background regions that are unlikely to contain the object of interest - a face. Viola and Jones made lot of improvements and research and finally, their detection system is based on several key ideas.

1) Integral image was originally used in the early work of Papageorgiou et al. [68] and allows very fast computation of rectangle features used by face detector. The system does not work directly with image intensities. The integral image at location *x, y* contains the sum of the pixels above and to the left of *x, y* inclusive:

$$ii(x,y)=\sum_{x'\le x,y'\le y} i(x',y')$$

where *ii(x,y)* is the integral image and *i(x, y)* is the original image. The integral image can be easily and efficiently computed in one pass over the original image through the following pair of recurrences:

$$s(x,y) = s(x,y-1)+i(x,y)$$
$$ii(x,y) = ii(x-1,y)+s(x,y)$$

where *s(x, y)* is the cumulative row sum, *s(x, -1) = 0*, and *ii(-1, y) = 0*. This way, any rectangular sum can be computed effectively in four array references only (see *Figure 32*), or difference between two rectangular sums can be computed in eight references, etc.

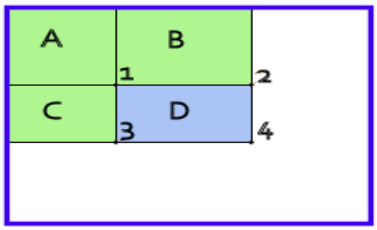

Figure 32. The sum of the pixels within rectangle D can be computed with four array references. The value of the integral image at location 1 is the sum of the pixels in rectangle A.
The value at location 2 is A+B, at location 3 is A+C, and at location 4 is A+B+C+D.
The sum within D can be computed as 4+1-(2+3).

2) Haar-like features: System presented by Viola and Jones classifies images based on the value of simple filters called *Haar-like features* [69] that are effective for face analysis. Once the integral image is computed, any one of these features can be computed at any scale or location in constant time. They used these features rather than the pixels directly, because they can encode ad-hoc domain knowledge, and the feature-based system operates much faster than pixel-based one. Three kinds of basic features have been suggested and used: *two-rectangle feature* (see *Figure 33a,b*), *three-rectangle feature* (*Figure 33c*) and *four-rectangle feature* (*Figure 33d*). The value of each of these features is the sum of the pixels within the dark gray shaded region(s) subtracted from the sum of pixels in the light gray shaded region(s). Using the base resolution *24x24* of the detector, the exhaustive set of rectangle features is quite large (contains over 140,000 features, details in *Table 5* in section 5.3).

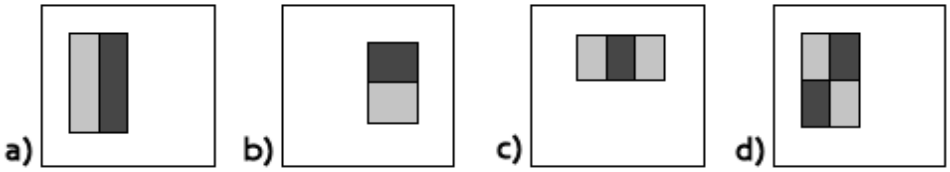

Figure 33. Examples of basic rectangle features, whose values can be computed effectively in constant time using integral image (the value of features (a) and (b) is computed by using 6 references only, value of feature (c) by 8 references, and 9 in the case of (d))

3) AdaBoost learning algorithm (abbreviation for *Adaptive Boosting*) [67] is a machine learning algorithm whose goal is to select a relatively small number (a few hundred or thousand) of simple *weak classifiers* (restricted to using exactly one Haar-like feature) and combine them to form an efficient and accurate *strong classifier*. This technique is called *Boosting*. Every weak classifier $h_j(x)$, composed of a single feature $f_j(x)$, can be written as

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

where θ_j is an optimal threshold and p_j is parity (indicating the direction of the inequality sign), such that the minimum number of examples is misclassified; variable x represents a *24x24* pixel subwindow of image. In each round of boosting, weak learning algorithm selects exactly one weak classifier, which best (with the lowest weighted error) separates the given training set containing labelled positive (faces) and negative (nonfaces) examples. In subsequent rounds, incorrectly labelled examples are given a higher weight (or alternatively, correctly labelled examples are given a lower weight), so the new classifier focuses more on those examples (demonstrated in *Figure 34* below). That's why the learning process is called *Adaptive*. AdaBoost is run for T (in practice, a few hundred) rounds, and the final strong classifier $H(x)$ is given by linear combination of weak classifiers:

$$H(x) = \begin{cases} 1 & \text{if } \sum_{t=1}^{T} \alpha_t h_t(x) \geq \lambda \sum_{t=1}^{T} \alpha_t \\ 0 & \text{otherwise} \end{cases}$$

where α_t encodes the weight of each weak classifier $h_t()$, and λ represents the threshold of strong classifier (number of false positives and faces detected in image can be adjusted by this threshold).

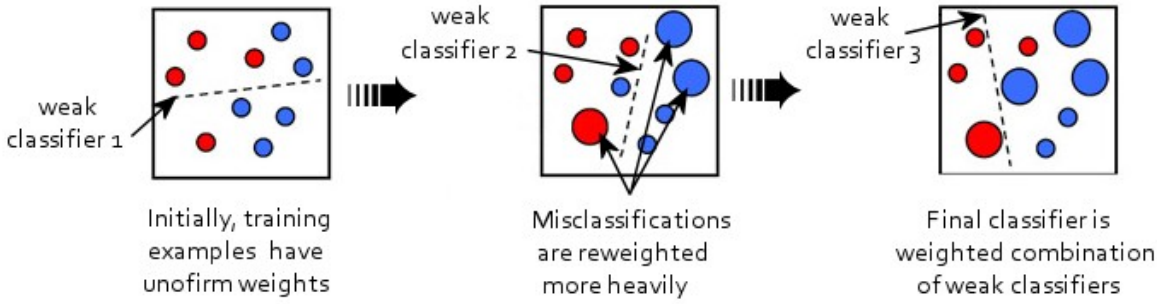

Figure 34. New weak classifiers are chosen during the boosting, weights of labelled examples are always updated in subsequent rounds of boosting

In practice, features that are selected in early rounds of the boosting process had error rates between 0.3 and 0.4, while features selected in later rounds, as the task becomes more difficult, yield error rates above 0.4 [65]. The whole pseudocode of AdaBoost can be found in *Table 6* in section 5.3.

4) The cascade architecture is an object specific focus-of-attention mechanism, which discards regions that are unlikely to contain the object of interest. It is based on combining increasingly more complex classifiers in a series. This technique allows a large number of nonface subregions of image to be quickly eliminated with very little processing by the initial classifiers, while spending more computation on promising face-like subregions by the subsequent, more complex classifiers. This method achieves increased detection performance while radically reducing computation time.

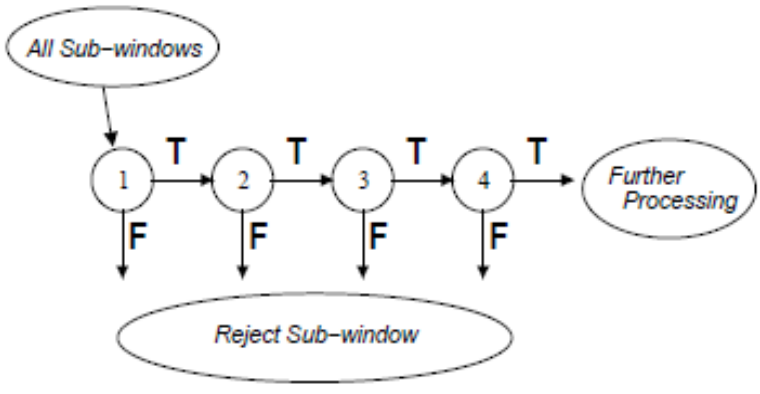

Figure 35. Detection cascade of four classifiers

AdaBoost-based detector for frontal faces of Viola and Jones is very fast and accurate method, and has a simple implementation. On the other hand, the crucial disadvantage is that it needs a very long training time and uses a greedy search through feature space. Later, Jones and Viola extended their work a bit [70], and recently [15] have presented a new multi-view face detection framework (as in the works of Rowley et al. [55], section 2.4.2, and Schneiderman et al. [63] [14], section 2.4.5, different view-based detectors were built). Compared with Rowley et al. results were very similar.

Besides the AdaBoost learning algorithm, similar approach called *FloatBoost* can be employed to detect faces [71]. FloatBoost-based system has a simple implementation too and achieves a very fast detection with a fairly high accuracy. In contrast to the AdaBoost, it allows to find a more potent set of weak classifiers through a less greedy search, and result in a faster, more accurate classifier. On the other hand, the training phase of FloatBoost detection system requires about five times longer training times (a few weeks), as compared with AdaBoost. Performance comparison of both approaches can be seen in *Figure 36* below.

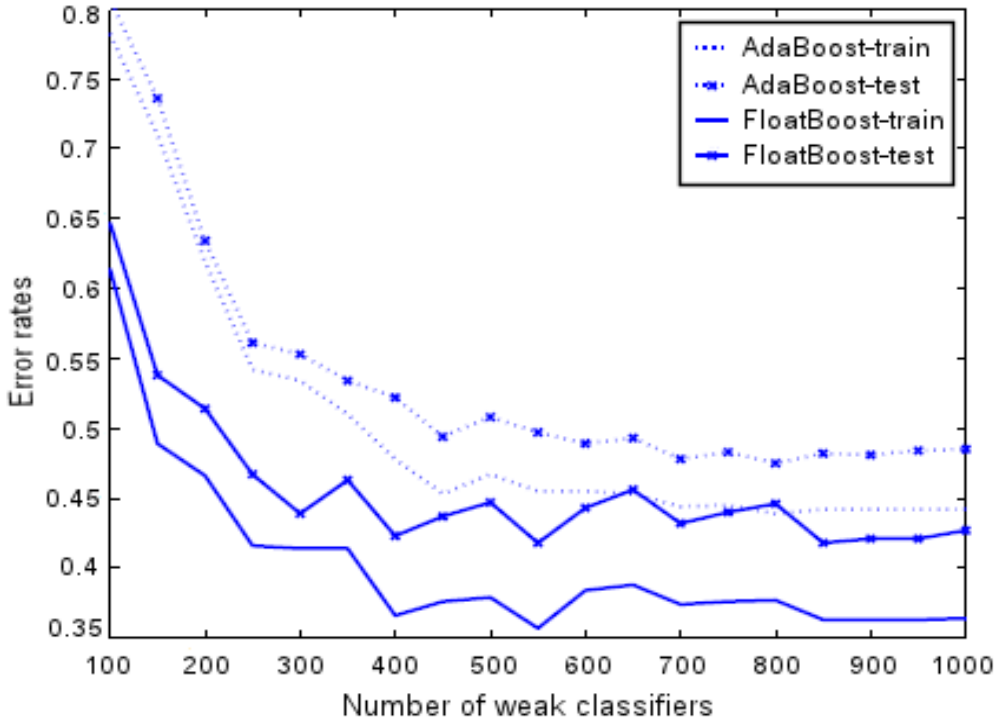

Figure 36. The training and testing error curves for FloatBoost and AdaBoost, the detection rate fixed at 99.5% (graph taken from [72])

2.4.7 Other Appearance-Based Methods

There is a great number of other successful appearance-based methods used in the sphere of face detection. Yang et al. [73] proposed a method that applies *SNoW (Sparse Network of Winnows)* learning architecture to detect faces with different features and expressions, in different poses, and under different lighting conditions. Samaria and Young implemented *Hidden Markov Model (HMM)* to facial feature extraction and face location and recognition [74] [75]. In some works [76] *Information-theoretical approach* is used to discriminate between two classes of faces and nonfaces. Last but not least, *Inductive-learning algorithms* have also been applied to detect faces [77] [78].

3 PERFORMANCE EVALUATION AND COMPARISON

All approaches mentioned in preceding chapters focus on reaching high reliability, efficiency and speed. Each of these algorithms tries to detect face in a scene while reaching the lowest false alarm rate combined with the best detection rate. An ideal face detection system should reach a hit rate of 100% with false alarm rate of 0. But none of contemporary systems can achieve this generally because increasing detection rate is usually connected with the simultaneous false alarm rate increment. In order to obtain a fair empirical evaluation of the face detection methods, it is necessary to test them on the same standard test sets (e.g. CMU/MIT test set [79]). Comparison of several representative methods can be found below in *Table 2* and *Table 3*.

Method	*Implementation*	*Test Set 1*		*Test Set 2*		*Test Set 3*	
		DR (%)	FA	DR (%)	FA	DR (%)	FA
Neural Networks	**Rowley et al. [54]**	**87.1**	**15**	**92.5**	**862**	**90.5**	**570**
Distribution-Based	**Sung and Poggio [64]**	**81.9**	**13**	**–**	**–**	**–**	**–**
Local Appearance	**Schneiderman and Kanade [66]**	**–**	**–**	**93.0**	**88**	**94.4**	**65**
AdaBoost	**Viola and Jones [69]**	**–**	**–**	**–**	**–**	**91.4**	**50**

Table 2. Performances on the CMU/MIT test set: Test Set 1 (155 faces), Test Set 2 (483 faces), Test Set 3 (507 faces) (FA = number of false alarms, DR = detection rate, results taken from [80])

Method	*Implementation*	*Test Set 1*		*Test Set 2*	
		DR (%)	FA	DR (%)	FA
Distribution-Based	**Sung and Poggio [64]**	**–**	**–**	**81.9**	**13**
Neural Network	**Rowley et al. [54]**	**92.5**	**862**	**90.3**	**42**
SNoW with primitive features	**Yang et al. [78]**	**94.2**	**84**	**93.6**	**3**
Inductive learning	**Duta and Jain [83]**	**90.0**	**–**	**–**	**–**
Local Appearance	**Schneiderman and Kanade [66]**	**93.0**	**88**	**91.2**	**12**
Support Vector Machines	**Osuna et al. [50]**	**–**	**–**	**74.2**	**20**

Table 3. Experimental results on images from Test Set 1 (125 Images with 483 Faces) and Test Set 2 (23 Images with 136 Faces)

In face detection theory, for all the algorithms the detection rate versus false positive rate can be plotted using a tool called *Receiver Operating Characteristic* (ROC) curves [81]. ROC curves are a way of visualizing the trade-offs between detection and false positives (some example of ROC curve shown in *Figure 37*). ROC analysis allows comparison with previous results (illustrated in *Figure 38*) and provides tools to select possibly optimal models and to discard suboptimal ones. Finding the best ratio between detection rate and false alarm rate and suggesting highly accurate system dealing with complexity of face manifolds will keep challenging for many students, scientists and engineers working in image processing, computer vision, animation and biometrics.

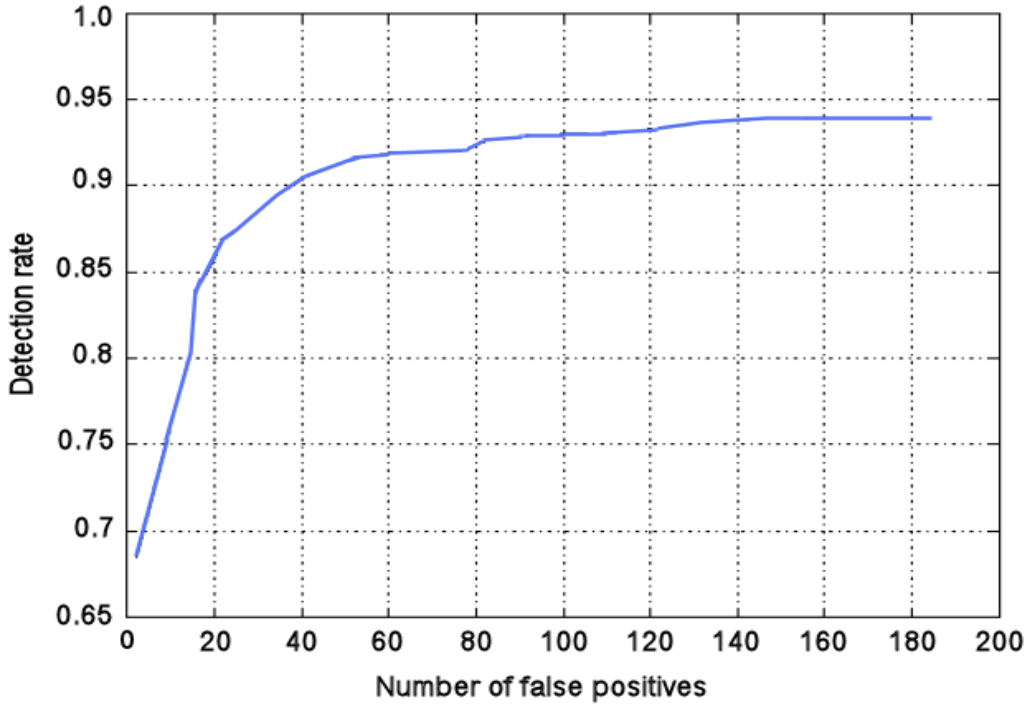

Figure 37. ROC curve showing the performance of AdaBoost-based face detector of Viola and Jones on the MIT+CMU test set (graph taken from Viola and Jones [65])

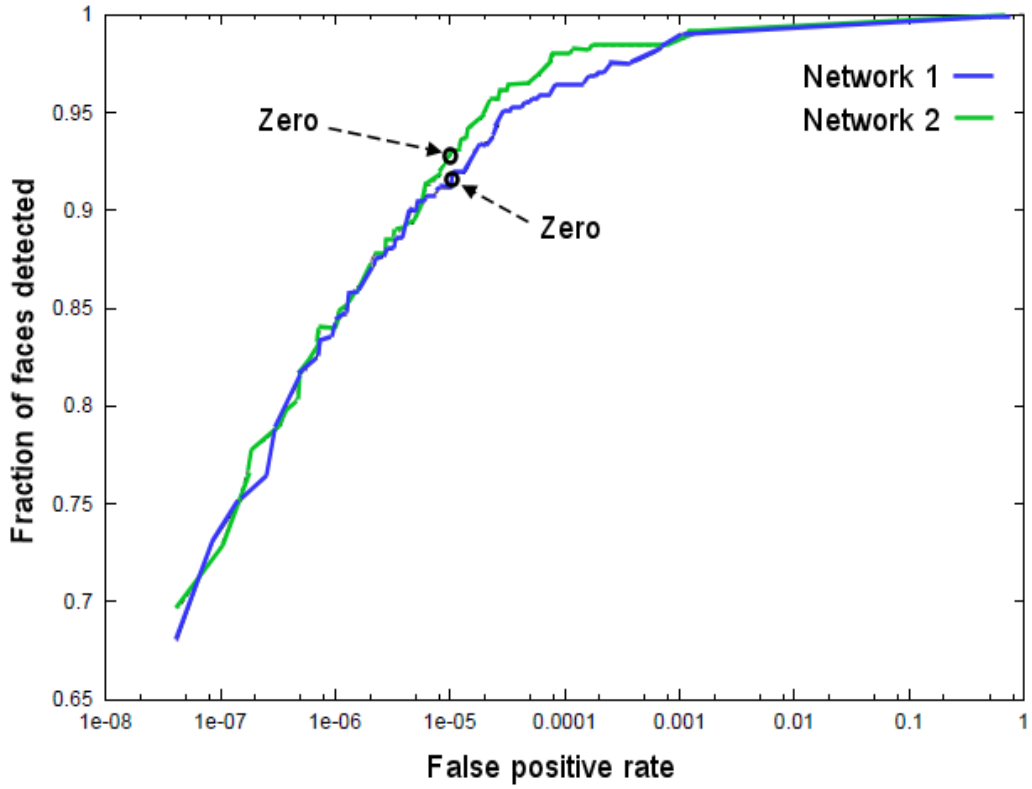

Figure 38. Comparison of two ROC curves. The detection rates are plotted against false positive rates as the detection threshold is varied for two neural networks. Zero points are the zero threshold points which are used for all other experiments (graph taken from Rowley et al. [11])

4 FACE IMAGE DATABASE

In recent years, dozens of methods for face detection have been developed. Most of them require a training data set of color or grayscale face images. Some of them are able to deal with both frontal and profile or rotated faces, others aren't. Browsing the Web galleries and and looking for appropriate images can be often quite annoying and time-consuming. Therefore, in terms of face detection and recognition, several face image databases were constructed for purposes of experiments. *Table 4* summarises a few well-known face image databases.

Data Set	*URL*
Grayscale FERET Database	http://www.itl.nist.gov/iad/humanid/feret/
Color FERET Database	http://face.nist.gov/colorferet/
MIT Database	http://www.ai.mit.edu/projects/cbcl.old/software-datasets/faces.tar.gz
AT&T Database	http://www.cl.cam.ac.uk/research/dtg/attarchive/facedatabase.html
M2VTS Database	http://www.tele.ucl.ac.be/PROJECTS/M2VTS/m2fdb.html
Extended M2VTS Database	http://www.ee.surrey.ac.uk/CVSSP/xm2vtsdb/
UMIST	http://images.ee.umist.ac.uk/danny/database.html
Yale Database	http://cvc.yale.edu/
CalTech Dataset	http://www.vision.caltech.edu/archive.html

Table 4. Face image databases

For example, from homepage of CalTech Dataset, the “Caltech 10,000 Web Faces” gallery (139 MB) can be downloaded for free. Except of the gallery of human faces, galleries of other specific objects (motorcycles, cars, airplanes, leaves, home objects, toys, and other background images) can be found there too. Images from these “nonface” galleries can be then used e.g. as negative examples (examples of images that contain no faces) during training or testing a new face detector. Indeed, the CalTech Dataset was used as source of training examples in the course of implementation of AdaBoost-based detector described in the subsequent chapter.

5 IMPLEMENTATION OF ADABOOST-BASED SYSTEM

All approaches mentioned and described in Chapter 2 focus on reaching high reliability, efficiency and speed. Although they can be classified by common features into several categories (see *Table 1*), each of them may apply other (quite different) principles, generally. For that reason, they differ from each other by robustness, overall performance, and last but not least, complexity of implementation. Moreover, each method can be determined for a specific type of images (either for color or grayscale images, or for both), and they can deal with different collections of faces (either with frontal faces only, or besides with profile face, or even with various in-plane-rotated faces). Also, their training times (if any) are significant. If we take all the preceding aspects into account, AdaBoost-based detection system seems to be the most promising solution. Detector learned by this method can deal with both frontal and profile face, and it uses no color information, that's why it's possible to be applied for both color and grayscale images. System can deal with various rotations too and is considered to be a quite fast and robust. On the other hand, a major disadvantage is that AdaBoost needs a very long training time. But as compared with FloatBoost, training times are still several times shorter.

Representative AdaBoost-based system was developed by Viola and Jones [15] [65]. They claimed that their real-time application is approximately 15 faster than any previous approach and is able to achieve high detection rates. Viola and Jones have introduced new ideas such as image representation called *Integral image*, they used *Haar-like features* for classification of subimages and suggested the *Cascade architecture* of strong classifiers. Thus, subsequent sections are devoted to implementation of AdaBoost-based detector for Java SE 6 platform, including further explanation of the key ideas and techniques employed by this approach.

5.1 Haar-like features

As suggested in [65] simple and effective filters called Haar-like features are used for face analysis. They are able to encode ad-hoc knowledge of rules that describe a typical face. Viola and Jones introduced the system based on *basic Haar-like features* comprehensive of three types of features: *two-rectangle feature* (see *Figure 33a,b* in section 2.4.6), *three-rectangle feature* (*Figure 33c*) and *four-rectangle feature* (*Figure 33d*). Experiments showed that these basic rectangle filters are not sufficient to detect in-plane-rotated and profile faces with high accuracy. That's why this basic set of features has to be extended by an efficient collection of *additional Haar-like features* [15] (illustrated in *Figure 39* below). Since these novel filters focuses on diagonal structures present in the image, they are able to deal with rotated and profile face more properly, thus they enhance the expressional power of the learning system. The value of each of these features is calculated in the same way - the sum of the pixels within the dark gray shaded regions subtracted from the sum of pixels in the light gray shaded

regions (*Figure 41*). Using an *integral image* representation for images (*Figure 32*), value of novel filters can be computed rapidly at all scales in constant time by only 16 references.

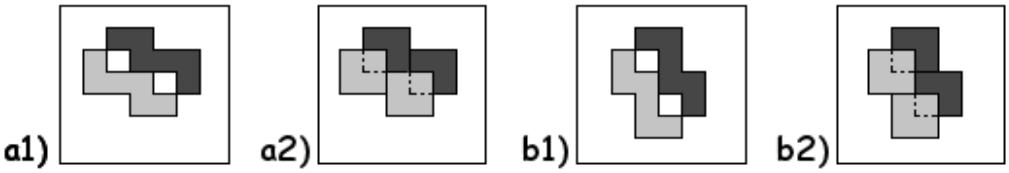

Figure 39. Examples of two additional types of diagonal features. Using an integral image, they can be computed effectively by 16 references only. Diagonal feature (a1) is constructed from 4 overlapping rectangles as illustrated in (a2), analogously feature (b1) from 4 rectangles in (b2).

Every feature is determined by its type, relative position within the enclosing window of detector (fields `x` and `y` in the exemplary source code below) and its dimensions (total `width` and `height`). For that purposes an abstract `HaarLikeFeature` superclass was created. Fields `xInit`, `yInit`, `heightInit` and `widthInit` keep the initial coordinates and dimensions before scaling the feature.

```
 public abstract class HaarLikeFeature {

   protected int x, y;     // current positions of the top left corner of
feature
   protected int height, width;              // current dimensions of the
feature

protected final int xInit, yInit;             // initial coordinates of
feature
protected final int heightInit, widthInit;     // initial dimensions of
feature

protected abstract float getValueOfFeature (IntegralImage im);

// constructors and other methods here
 }
```

Since the calculation of value of feature of one type differs from the calculation of value of feature of the other type, special child class extending `HaarLikeFeature` superclass needs to be created for each type of features. For instance, `Haar3HorizontallyAdjecentCells` class represents all three-rectangle features (*Figure 33c*). In this case, value can be calculated by 8 references (8 corner pixels). Coordinates of these crucial points are represented by fields `Ax`, `Bx`, `Cx`, `Dx`, `Ay` and `Ey`.

```
 public final class Haar3HorizontallyAdjecentCells extends HaarLikeFeature {

   private int Ax, Bx, Cx, Dx, Ay, Ey;  // coords of crucial points of
feature

   @Override
   public float getValueOfFeature(IntegralImage im) {
```

```
    // calculation of value of feature by using crucial points is done here
  }

  // constructors and other methods here
}
```

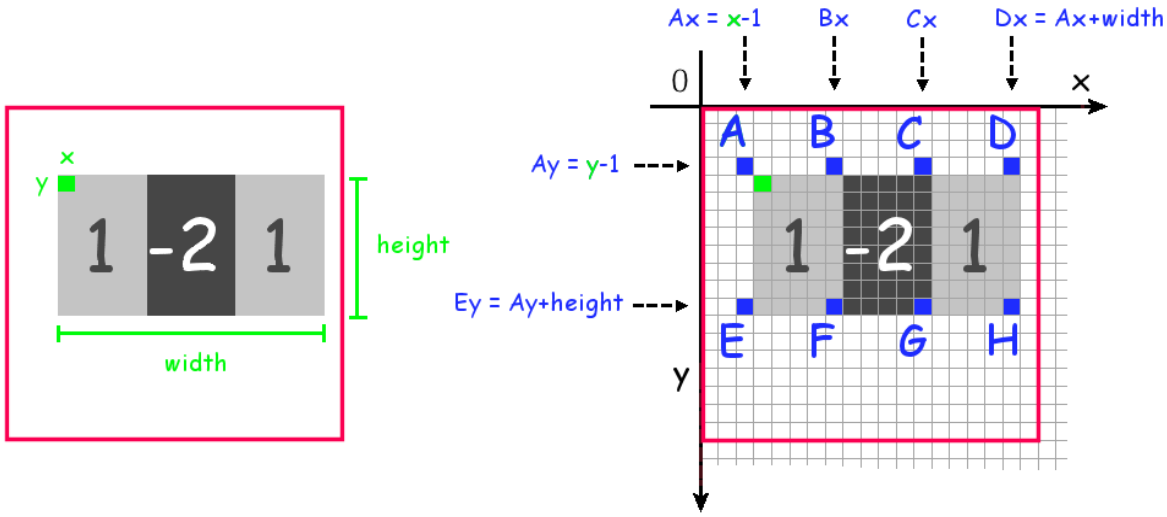

Figure 40. Value returned by three-rectangle feature can be calculated by 8 references (points A-H).
Grid represents pixels of integral image, red square is enclosing window of face detector.

In *Figure 40* above, calculation of value returned by three-rectangle feature is illustrated. Dark gray shaded area in the middle is doubled and subtracted from the sum of pixels in light gray shaded border areas. This can be written as

$$value = S_{ABFE} + S_{CDHG} - 2 * S_{BCGF}$$

that can be by using the integral image formulated as

$$value = (A + F - B - E) + (C + H - D - G) - 2 * (B + G - C - F)$$

and consequently rearranged as

$$value = A + H - D - E + 3 * (C + F - B - G)$$

Values returned by features of other types can be calculated in a similar manner. Prototypes and their regions with corresponding factors are summarised in *Figure 41* below.

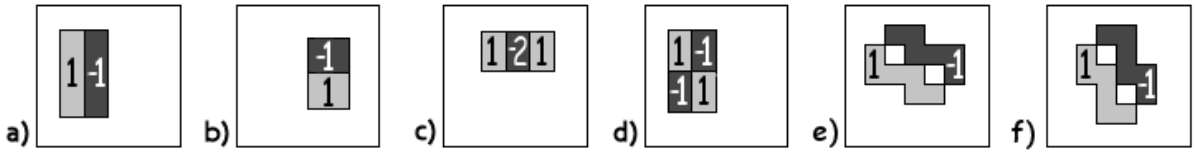

Figure 41. Prototypes of Haar-like features and factors of their keyregions during calculation of return value

5.2 Integral image

Detection system based on Haar-like features doesn't work directly with image intensities, but with new image representation called *Integral image* (see section 2.4.6 for more details, or *Figure 32*). General integral image is represented by abstract `IntegralImage` superclass.

```
public abstract class IntegralImage {

   protected final int height;                                    // height of
image
   protected final int width;                                      // width of
image
   protected final int[][] data;                        // integral image
pixel-data

   public IntegralImage(BufferedImage image) {                        //
constructor
     height = image.getHeight();
     width = image.getWidth();
     data = new int[height][width];
     // computation of pixel data of integral image is done in child classes
   }

   public abstract float getVariation();      // method to get variation of
image

   // other methods here
 }
```

Since the detection system works with images that are taken under different lighting conditions, *variance normalization* of both training data and each window of testing data is necessary to be performed. Computational formula for variance is

$$\sigma^2 = \left(\frac{1}{N}\sum_{i=1}^{N} x_i^{\,2}\right) - \bar{x}^2$$

where σ is standard deviation and $\bar{x}$ is the mean. The effect of image normalization can be achieved by dividing the return value of each feature by square root of variance. However, for training data and testing data, variance is calculated in different ways. For that reason, `getVariation()` method is abstract.

As regards the training images (that are supposed to be cropped to basic resolution of detector, e.g. *24x24* pixels), they are represented by a child `IntegralImageForTraining`

class extending the abstract `IntegralImage` superclass. Variance is computed only once during initialization of training example (in constructor of `IntegralImageForTraining` class) and is constant for each training example.

```
public final class IntegralImageForTraining extends IntegralImage {

   private final float variation;          // overall variation of training
image

   public IntegralImageForTraining(BufferedImage image) {          //
constructor
     super(image);
     // computation of pixel data and variation is done here
   }

   @Override
   public float getVariation() {
     return variation;
   }

   // other methods here
}
```

On the other hand, input images given to face detection (images of any resolution) are represented by a child `IntegralImageForDetection` class, as well extending `IntegralImage` superclass. During scanning an input image for faces, current local variance of scanning window needs to be calculated rapidly: the mean $\bar{x}$ of window can be computed using integral image, and the sum of squared pixels is computed using an auxiliary integral image – integral image of the image squared. Pixel data of both integral image and auxiliary integral image are calculated once in constructor of `IntegralImageForDetection` class and are permanently available during detection phase. Thus, local variation of scanning window can be calculated effectively and in constant time for any position of the window.

```
public final class IntegralImageForDetection extends IntegralImage {

   private final long[][] dataSquared;         // pixel data of the image
squared

   public IntegralImageForDetection(BufferedImage image) {         //
constructor
     super(image);
     // computation of pixel data and data squared is done here
   }

   @Override
   public float getVariation() {
     // local variation is computed here by using pixel data and data
squared
```

```
    }

    // other methods here
}
```

5.3 Weak classifiers

All prototypes of Haar-like features can be shifted within the enclosing window of detector and scaled in vertical and horizontal direction in order to generate an *overcomplete set of features*. The number of features derived from each prototype is quite large and differs from type to type. *Table 5* lists the number of features for a window size of *24x24* pixels.

Feature type	***Calculation of count***	***Count***
	$\sum_{i=1}^{24}\sum_{j=1}^{12}(i*(2\,j-1))$	**43,200**
	$\sum_{i=1}^{24}\sum_{j=1}^{12}(i*(2\,j-1))$	**43,200**
	$\sum_{i=1}^{24}\sum_{j=1}^{8}(i*(3\,j-2))$	**27,600**
	$\left(\sum_{i=1}^{12}(2i-1)\right)^2$	**20,736**
	$\sum_{i=1}^{6}\sum_{j=1}^{4}(5\,j*(4\,i-3))$	**3,300**
	$\sum_{i=1}^{6}\sum_{j=1}^{4}(5\,j*(4\,i-3))$	**3,300**
Total sum		**141,336**

Table 5. Prototypes of Haar-like features and their quantities within the window of basic resolution of 24x24 pixels

During initialization of AdaBoost (pseudocode of AdaBoost can be found in *Table 6* in the end of the section), *pool of weak classifiers* is initialized: each feature $f_i(x)$ from the over complete set of features, is assigned to exactly one *weak classifier* $h_i(x)$

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

with default threshold θ_j and default parity p_i. Afterwards, in every round of boosting, all weak classifiers in weak-classifiers pool are relearned – optimal threshold θ_j and parity p_j is found, considering current weights of examples. Then, exactly one weak classifier is selected, which best (with the lowest weighted error) separates the given training set of positive and negative examples (illustrated in *Figure 34* in section 2.4.6). The optimal threshold θ_j and the parity p_j for each weak classifier $h_i(x)$ can be found by a linear search over all possible thresholds and testing the weighted error for each. Since there are only as many relevant thresholds as there are examples in the training set (plus 1), there aren't too many thresholds to try. This also requires sorting all the feature values, so that the error for a threshold could be calculated efficiently. Two modelling examples are captured in *Figure 42* below.

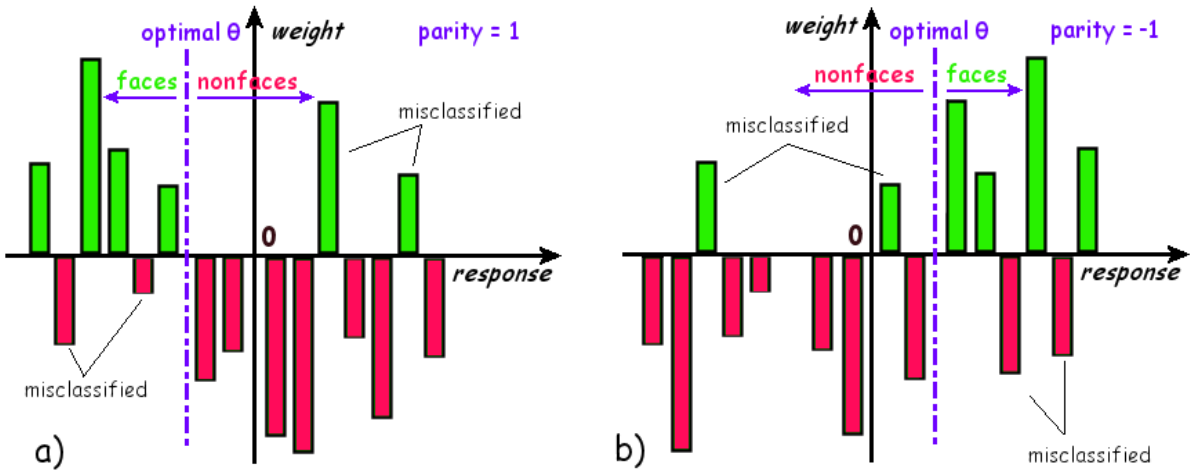

Figure 42. Searching for optimal threshold and parity of weak classifier. First of all, histogram of return values (responses to training images) is built. That means that the feature of weak classifier has to be evaluated for every positive (green column) and negative (red column) example. Weights of negative training examples are diagrammatized in histogram with opposite (i.e. negative) sign. All responses have to be sorted, and then the optimal threshold can be found effectively by linear search. There are as many relevant thresholds as there are examples in the training set. In illustration (a) the parity is set to 1, because in this case responses to faces (positive examples) are usually less then the optimal threshold, whereas in (b) the parity is set to -1.

Consequently, relearning of all 140,000 weak classifiers in the pool is quite time-consuming, because every weak classifier (rather its feature) has to be evaluated on the whole training set, in order to find optimal threshold and parity of the classifier. On a 1.8 GHz AMD Sempron 3000+ it takes about 472 seconds to relearn all weak classifiers in the pool by using 1000 positive and 3000 negative examples. That's why *multi-thread paradigm* was newly implemented in this phase of boosting to get better performance on multi-core or multi-processor systems. In `WeakClassifierLearner` class as many relearning threads

(represented by inner `WeakLearnerThread` class) are executed as there is number of cores (processors) detected by application (or manually set by user). Every running thread selects a single weak classifier that hasn't been relearned so far, and executes learning of the selected one. In practice, on a 2.0 GHz AMD Turion X2 it takes about 225 seconds now to relearn all weak classifiers in the pool with the same training examples. After relearning phase, the best weak classifier is selected, weights of misclassified training examples are increased (or weights of correctly classified examples are decreased) and normalized, and the best weak classifier is removed from the pool. Thus, in next iteration of boosting, only weak classifiers remaining in the pool are relearned.

- Given example images (x_1, y_1), ... (x_n, y_n) where $y_i = 0, 1$ for negative and positive examples, respectively.

- Initialize weights $w_{1,i} = \frac{1}{2m}, \frac{1}{2l}$ for $y_i = 0, 1$ respectively, where m and l are the number of negatives and positives, respectively.

- For $t = 1$ to T do:

1. Normalize the weights of examples:

$$w_{t,i} \leftarrow \frac{w_{t,i}}{\sum_{j=1}^{n} w_{t,j}}$$

so that w_t is probability distribution.

2. For each f_j train a weak classifier h_j (i.e. find optimal threshold θ_j and parity p_i) which is restricted to using a single feature. The error ε_j is evaluated with respect to w_t:

$$\varepsilon_j = \sum_i w_i \left| h_j(x_i) - y_i \right|$$

3. Choose the classifier, h_t,with the lowest error ε_t

4. Update the weights:

$$w_{t+1,i} = \begin{cases} \beta_t w_{t,i} & \text{if example } x_i \text{ is classified correctly} \\ w_{t,i} & \text{otherwise} \end{cases}$$

where $\beta_t = \frac{\varepsilon_t}{1 - \varepsilon_t}$

- The final strong classifier is:

$$H(x) = \begin{cases} 1 & \text{if } \sum_{t=1}^{T} \alpha_t h_t(x) \geqslant \lambda \sum_{t=1}^{T} \alpha_t \\ 0 & \text{otherwise} \end{cases}$$

where $\alpha_t = -\log \beta_t$ and λ is optimal threshold of strong classifier

Table 6. Pseudocode for the AdaBoost algorithm (adapted from [65])

5.4 Strong classifier

During learning phase, the best weak classifiers are removed from the pool and collected to form an accurate *strong classifier*. AdaBoost is run for *T* (in practice, even a few hundred or thousands) rounds until both target *false alarm rate (FAR)* and *face detection rate* (DR) of the strong classifier are reached. The final strong classifier *H(x)* is given by linear combination of weak classifiers:

$$H(x) = \begin{cases} 1 & \text{if } \sum_{t=1}^{T} \alpha_t h_t(x) \geqslant \lambda \sum_{t=1}^{T} \alpha_t \\ 0 & \text{otherwise} \end{cases}$$

where α_t encodes the weight of each weak classifier $h_t()$. Parameter λ represents an overall threshold of strong classifier and exerts influence on the number of both false positives and faces detected in image: increasing this value reduces the FAR, but also decreases DR of strong classifier, and vice versa.

For purposes of finding the optimal λ, a *binary search algorithm* can be employed. Initially, a lower bound is set to 0.0 and the upper to 1.0. At each iteration, both FAR and DR are calculated by using a special *validation set* of face and nonface examples – using the same training set as a validation set is not the best course of action, because the strong classifier was trained on it, so the calculation of FAR and DR wouldn't be objective (however as a last resort, it's possible to share positive validating examples with those in the training set). Then, λ is adjusted up or down depending on whether DR is too low or not. Algorithm continues iterating until the optimal threshold is found.

If results on the validation set aren't sufficient after finding the optimal λ threshold (the λ couldn't be adjusted so that both FAR and DR could meet their target rates), another new best weak classifier has to be learned and added, in order to form even better and more accurate strong classifier. After that, binary search for the optimal λ can be executed again and new detection rates may be calculated.

5.5 Cascade of strong classifiers

The AdaBoost-based system detects faces by exhaustively scanning an input image for faces. Evaluating all subwindows becomes time-consuming, if the strong classifier is composed several thousand weak classifiers. In order to overcome this problem, a *cascade of strong classifiers* was proposed in the form of a degenerate decision tree (see *Figure 35* in section 2.4.6). Cascade is based on combining increasingly more complex strong classifiers in a series. An input window of detector is evaluated on the first, and the simplest strong classifier, and if that classifier returns false, then computation of the window ends and detector returns false. Otherwise, if the classifier returns true, window is passed to the next, more complex strong classifier in the cascade. This technique allows a large number of nonface subregions

of image to be quickly eliminated with very little processing by the initial classifiers, while spending more computation on promising face-like subregions by the subsequent, more complex strong classifiers. In this way, the cascade of strong classifiers achieves increased detection performance while radically reducing computation time.
During training, a very simple framework was implemented to produce an effective cascade. Each stage (i.e. strong classifier) of the cascade is trained by adding the best weak classifiers until the specific fraction of nonface examples is successfully eliminated while detecting certain portion (nearly 100%) of faces. For instance, each stage could be trained to eliminate at least 50% of the nonfaces (it means that each stage has to reach the false alarm rate less than or equal to 0.5) while detecting at least 99,5% of the faces. Consequently after several, e.g. 10 stages, a DR about $0.995^{10} \approx 0.95$ and FAR about $0.5^{10} \approx 9.8e\text{-}04$ can be expected for the whole cascade, in the optimal case.

After training of each stage, it's crucial to update the set of negative examples and train the subsequent classifiers using those false positives only, witch still pass through all previous stages. As a result, the next classifier faces more difficult task than the previous one. Also with increasing stage, the number of weak classifiers, which are needed to achieve the desired FAR at given DR, increases. Since some new examples with undefined weights may appear in the training set after updating the negative subset, weights of all examples are reset to default uniform values.

In terms of building an optimal cascade of strong classifiers, many optimization frameworks were defined and lots of papers have been published on this topic (for example [82]). Unfortunately, forming such a cascade is a quite difficult problem that is still a relevant research topic.

5.6 Training phase of the detector

Building a face detection system requires obtaining a set of labelled both face and nonface examples. As the base resolution of detector is 24x24 pixels, all examples has to be cropped and resized to these dimensions. The main drawback of AdaBoost (and other algorithms based on machine learning) is that a large set of manually selected and cropped face examples is necessary for training a detector. Therefore, gathering and preprocessing of the positive training images may be quite time-consuming and annoying. On the other hand, nonface examples for the initial stage of cascade can be easily collected by selecting random subwindows from any nonface images. The nonface examples used to train subsequent stages can be obtained by scanning the partial cascade across the nonface images and gathering false positives only, until the maximum capacity of the training set is reached.

Considering that the frontal detector from Viola and Jones [65] handles approximately ± 15 degrees of in-plane rotation, generally 12 different cascades for frontal faces of 12 different rotation classes have to be trained to cover the full 360-degree range of possible rotations. Moreover, in order to deal with both frontal and profile faces, 3 stand-alone cascades for right-profile, frontal and left-profile poses are needed for each of 12 rotation classes.

Consequently, there are a total of 3*12 = 36 cascades required for full-range detection, theoretically. But in practice, only 6 basic cascades are trained: *0-degree, 30-degree and 60-degree frontal cascades*, and *0-degree, 30-degree and 60-degree right-profile cascades*. Since the Haar-like features can be flipped horizontally, any left-profile cascade can be created from the corresponding right-profile one (and vice versa). For example, by flipping a 30-degree right-profile cascade horizontally, new 330-degree left-profile cascade can be obtained (illustrated in *Figure 43a* below). Besides, all Haar-like feature can be also flipped vertically, as well as rotated 90 degrees (*Figure 43b*), which means that any of 30 remaining cascades can be easily and rapidly derived from the 6 basic ones listed above.

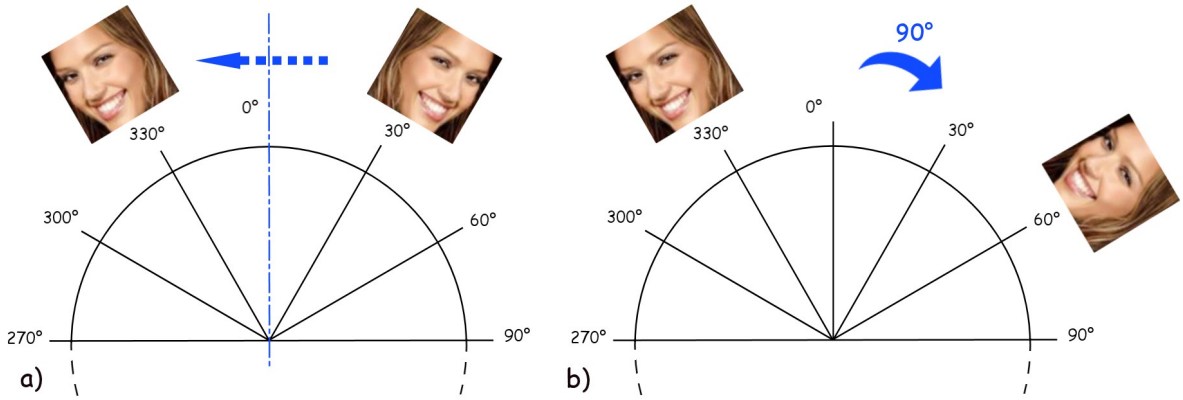

Figure 43. Derivation of remaining cascades. By flipping a 30-degree right-profile cascade horizontally, new 330-degree left-profile cascade can be obtained (a). Subsequently,by rotating this 330-degree cascade 90 degrees, another 60-degree left-profile cascade can be newly derived (b).

During flipping or rotating any feature, coordinates and dimensions of new transformed feature need to be calculated. All the computations are summarised in *Table 7* below. Besides, these transformations may often cause a change of feature type as illustrated in *Figure 44*.

Transformation	***Vertical flip***	***Horizontal flip***	***90-degree rotation***
Example	*Figure 44a*	*Figure 44b*	*Figure 44c*
Calculation of transformed coordinates and dimensions	$x' \leftarrow x$ $y' \leftarrow res - y - height$ $width' \leftarrow width$ $height' \leftarrow height$	$x' \leftarrow res - x - width$ $y' \leftarrow y$ $width' \leftarrow width$ $height' \leftarrow height$	$x' \leftarrow res - y - height$ $y' \leftarrow x$ $width' \leftarrow height$ $height' \leftarrow width$

Table 7. Calculations of dimensions (width', height') and coordinates (x', y') of newly transformed feature f'().Variable res represents current resolution of detection window.

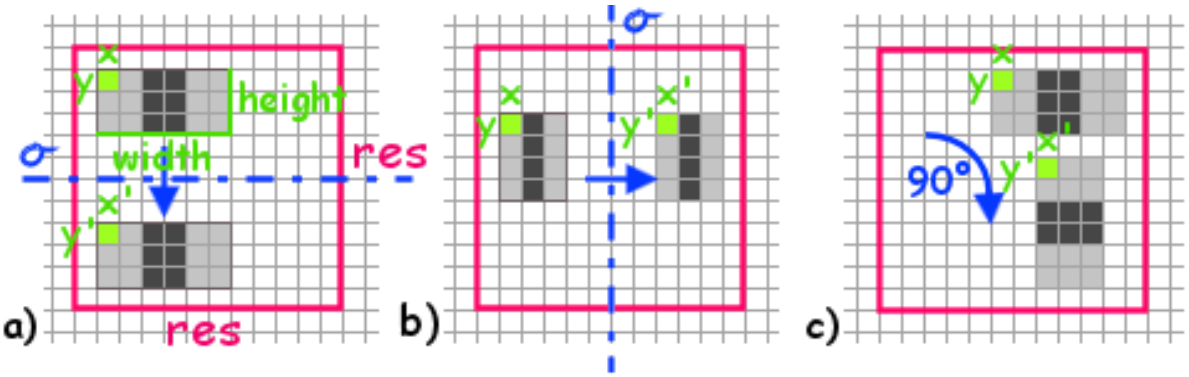

Figure 44. Vertical flip (a), horizontal flip (b) and 90-degree rotation (c) of a three-rectangle feature within a window of detector. In (c) rotation caused a change of feature type.

In some cases (see *Figure 45*), factors of two regions of a feature $f_j()$ are necessary to be swapped. This problem can be overcome by changing the sign of both threshold θ_j and parity p_j of correspondent weak classifier $h_t()$ (i.e. by multiplying both threshold and parity by -1). Moreover, during derivation of remaining cascades, *3 new additional prototypes* (*Figure 46*) of features may appear due to combination and application of various transformations.

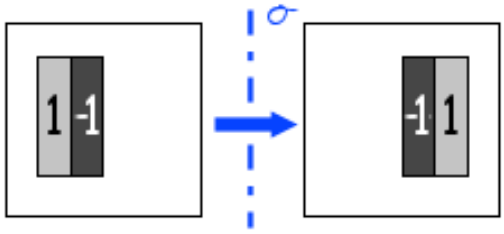

Figure 45. Flipping this prototype vertically requires changing the sign of both threshold and parity of correspondent weak classifier, so that dark and light areas can be swapped.

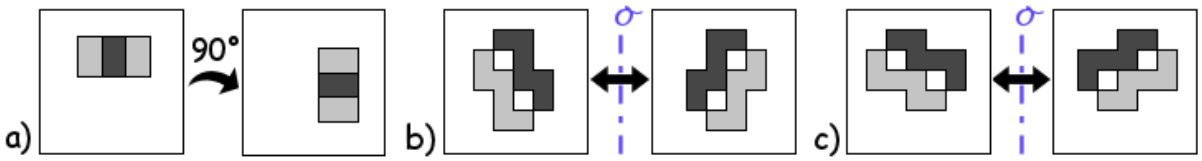

Figure 46. Three new prototypes of Haar-like features (those at the right in (a), (b) and (c)) may appear by applying of various transformations on current types of features.

Before initializing training phase of detector, some key parameters need to be set: *base resolution of detector, scale factor of detection window, shift base of window, number of strong classifiers per each cascade, maximum false alarm rate per stage, minimum face detection rate per stage, and maximum number of negative examples within the training set.* All these settings are saved in a property file, thus they can be easily edited. Mostly, the number of strong classifiers per cascade and the size of the training set exercise decisive influence on the training time of a cascade. Generally, it takes about several tens of hours to train a single cascade on a high performance supercomputer (see (*Table 9*), or about several

days on conventional desktop computer. Final detector is then saved into XML file, so that learned cascades and classifiers can be adjusted any time.

5.7 Detection phase

After loading the 6 basic cascades from the XML file and building-up the remaining ones (as suggested in previous section), the AdaBoost-based system tries to detect faces by scanning an input image at multiple scales and locations. Scaling is achieved by scaling the window of detector, rather than scaling the image. Good results can be obtained by using a scale factor of 1.25. During scanning, window of detector is shifted by $[s\Delta]$ pixels, where s is the current scale, Δ is a shift base, and [] represents the rounding operation. The choice of Δ affects both the speed and accuracy of the detector. Good results can be achieved for Δ within the range <1.0, 1.5>.
As for classification of subwindows of the input image, Viola and Jones introduced a *two-stage detector for rotated faces* [15], that at first runs the decision tree classifier on each subwindow to estimate its rotation class, and then one of 12 rotation specific cascades is used to classify the subwindow. By applying this detector, they reported about 4.7 times speed-up as well as significant decrease of the false positives (from 1345 to 221) as compared with try-all-rotations approach. On the other hand, using this two-stage detector lowered the detection rate (from 95% to 89,7%). Besides, Viola and Jones also tried to implement a *profile estimator*, that classifies an image window as right profile or left profile. Since there are only two classes in this case, this innovation didn't significantly improve the overall speed of detector, so they continued using the try-both-profiles detector instead.

Taking all the previous reported results into account, neither the two-stage detector for rotated faces nor the profile estimator were finally implemented, so the detector is based on try-all-rotations and try-all-profiles approach. However, the range of rotations, that will be inspected for presence of a face, can be manually adjusted by using the command-line arguments when executing the application. This improvement can significantly speed-up a detection phase (see *Table 10*) as well as decrease the false alarm rate, while faces are searched within the specific range of angles only.

During scanning the input image for faces, multiple detections may occur (see *Figure 47a* in Annexes section), so they need to be merged together. If the bounding regions of two detections overlap, they are put into the same subset. Finally, regions within the same subset are merged into a single detection (*Figure 47b*) – the corners of the final bounding region are the average of the corners of all regions in this subset. All the final detections are then saved into a text file, and every face is also fit within the bounding box that is highlighted within an output image.

5.8 Results and evaluation

For purposes of training, CalTech Dataset (see *Table 4*) was used. Five hundred examples of frontal faces were manually selected, cropped and scaled down to basic resolution of 24x24 pixels. Mirror

images of these examples were also used, resulting in a total of 1000 frontal positive examples. Besides, 500 right-profile faces (faces between frontal view and 45-degree profile view) were collected to train right-profile cascade. By rotating images of these two collections in-plane, positive examples for 30-degree and 60-degree cascades were generated. As for negative examples, 1000 random nonface images of any resolution were collected. Since there are about 150 million subwindows within these nonface images, a maximum of 3000 such subwindows were selected and used as negative training examples for each stage of the cascades. According to the type of example (whether it is face or nonface), its pose and rotation class, all the training examples were separated into several directories, that were finally packed into a single ZIP file. Concerning validation set, positive validating examples are shared with those 1000 in the training set, but the sets of the negative ones are always disjoint. Consequently, training of a detector can be executed. All basic settings of the detector for the training phase are summarised in *Table 8*. Using these settings, we can expect a detection rate about $0.995^{20} \approx 0.90$ and false alarm rate about $0.5^{20} \approx 9.5\text{e-}07$ for each cascade, in the optimal case.

Parameter	***Value***
Start (base) resolution of detector	24
Scale factor of detection window	1.0
Shift base of detection window	1.0
Number of strong classifiers per each cascade	20
Maximum false alarm rate per stage	0.50
Minimum face detection rate per stage	0.995
Maximum number of negative examples within the training set	3,000

Table 8. Basic parameters of the face detector for a training phase

The initial face detector was successfully trained on multi-processor machines, that are available via METACentrum project. Because of limitations on maximum run time of a job, all cascades had to be trained separately. For instance, on Manwe (8x dual-core AMD Opteron 885, 2.6 GHz) it took about 12 hours of computing time to get desired 0-degree frontal cascade. The resulting cascade had 20 stages of strong classifiers, with the first six comprising of 5, 5, 7, 13, 19 and 27 weak classifiers, respectively. More details about learned cascades can be found in *Table 9*. Finally, by editing the resulting XML files, all cascades were put together to form the complete face detector.

Cascade		***Number of weak classifiers per first 10 stages***										***Number of all weak classifiers***	***Training time***
		1	***2***	***3***	***4***	***5***	***6***	***7***	***8***	***9***	***10***		
	0-degree	5	5	7	13	19	27	34	46	39	38	357	12h 20m

frontal	30-degree	5	6	6	11	15	22	27	28	23	13	173	8h 45m
	60-degree	5	6	7	11	15	23	27	32	25	17	205	9h 26m
profile	0-degree	6	8	9	16	22	34	40	58	55	45	461	15h 05m
	30-degree	6	6	9	13	18	22	28	35	25	15	211	11h 03m
	60-degree	6	6	7	11	16	26	32	32	17	7	190	8h 31m

Table 9. Learned cascades of the face detector

Usually, after reaching the eighth stage, the number of weak classifiers began decreasing. This phenomenon is caused by repetitively selecting the same random subwindows as new negative examples while updating the set of training examples. Consequently, a new strong classifier is able to reach the target false alarm rate much sooner and needs a smaller number of weak classifier. As it is impossible to remember all previously visited subwindows (there are about 150 million negative subwindows), another more sophisticated method of selecting the new nonface examples should be implemented to overcome this problem.

The final face detector supports JPG/JPEG, GIF, BMP, WBMP and PNG input images. Detection times for images of various resolutions are listed in *Table 10* below. For instance, on a 320 x 240 pixel image, the 360-degree try-all-profiles detector takes about 21 seconds on a 2.4 GHz Intel Xeon, which is about 32 times longer than a try-all-rotations detector of Viola and Jones, and about 150 times longer than their two stage detector for rotated faces (both their detectors were tested on a 2.4 GHz Pentium 4). Mostly, this wide difference in speed is caused by building a more optimal and effective cascade – their first strong classifier in a cascade is comprised of one or two weak classifiers only, which are able to detect 100% of the faces as well as discard 60% of false positives [65]. Therefore, any extra stage post-optimization procedure [82] [83] should be implemented in future in order to reduce the number of weak classifiers per each stage of a cascade and increase the overall detection performance. Besides, Viola and Jones use two-stage detector for rotated faces, so they don't need to run all 36 cascades on each subwindow.

Image resolution	***Megapixels***	***Average detection time with try-both-profiles detector***	
		using all cascades	***using frontal cascade only***
320 x 240	0.08	21 seconds	1.99 seconds
640 x 480	0.3	1 minute 38 seconds	9.49 seconds
1,024 x 768	0.8	4 minutes 37 seconds	27.75 seconds
1,280 x 960	1.3	7 minutes 36 seconds	44.62 seconds

1,600 x 1,200	2	12 minutes 39 seconds	1 minute 13.65 seconds

Table 10. Average detection times for the input images of various resolutions on a 2.4 GHz Intel Xeon

As for results on the testing sets, the CMU Image Database collected by Schneiderman and Kanade was used. This image dataset is provided for evaluating algorithms for detecting frontal and profile views of human faces. The test set contains 208 images and according to Schneiderman and Kanade it has 347 profile faces (faces between ¾ view and full profile). Using the 360-degree try-both-profiles detector, a total of 39.0% of the profile faces were detected with 194 false positives. Since the detector wasn't trained on a full-profile training set, this results are quite sufficient. As compared with other methods, Schneiderman and Kanade [63] got a detection rate of 86.4% with 91 false positives, while Viola and Jones detected 70.4% of profile faces with 98 false positives [15].

On the second test set, which contains 33 images with 210 both frontal and profile faces (faces between frontal view and 45-degree profile view), the detector reached a detection rate of about 80% with 301 false positives.

Experiments showed, that the complexity of a background within the input image has a decisive influence on the total number of false positives – the more complex and manifold background is, the more false positives may occur (compare *Figure 48* with *Figure 49*). Furthermore, by increasing the resolution of the input image, the overall false alarm rate increases as well (*Figure 50*). More examples of face detection in images (*Figure 51*, *Figure 52*) can be found at the end of this thesis in Annexes section.

6 CONCLUSION

This bachelor thesis attempts to provide a survey of the best-known and the most successful methods for face detection in images and gives an overall comparison of them. Generally, all suggested systems focus on reaching high reliability and efficiency, and try to work quite predictably. A robust face detection system should detect all faces in a single image independently of many factors such as face position and scale, in-plane or out-of-plane face rotation, features such as beards, moustaches and glasses, facial shape and color (ethnic group), age and facial expression, and imaging conditions such as shadows and lighting. Although many of these non-trivial problems have almost be solved, there is still much work needed to improve accuracy and performance of proposed techniques.

6.1 Summarization of methods

In the last two decades a great research effort was made, that's why the automatic face detection in images is a quite well explored problem. Growing performance of wide available desktop computers created an enormous interest in this sphere and enabled and sped up developing of more sophisticated methods processing images in real time. More than a hundred strategies have been studied and reported, and can be divided into four categories: Knowledge-based methods, Feature-Invariant approaches, Template matching and Appearance-based methods.

Nowadays, appearance-based methods are the most successful and robust ones among face detection methods. Furthermore, some of them can deal with different poses, for example, rotation invariant neural network-based method of Rowley et al. [55], or Schneiderman and Kanade's method [63], or multi-view AdaBoost-based approach of Viola and Jones [15]. Actually, AdaBoost and FloatBoost [71] are said to be ones of the fastest learning algorithms, but they need quite long training times. Anyway, there is still much work needed, and highly accurate system with a high detection rate and low false alarm rate has been challenging for many students, scientists and engineers.

6.2 Conclusion of AdaBoost-based implementation

All major advantages and drawbacks of the methods mentioned in this thesis were taken into account, and finally AdaBoost-based detector was implemented for Java SE 6 platform. The detector was based on the key ideas and techniques introduced by Viola and Jones [15] [65], such as the image representation called Integral image, simple filters represented by Haar-like features, and the Cascade architecture of strong classifiers.

For purposes of training, a multi-thread paradigm was newly employed to get better performance on multi-core or multi-processor systems. Training set contained 1000 examples of frontal faces and 500 examples of right-profile faces. Both subsets were manually cropped and scaled down to the base resolution of 24x24 pixels. By rotating the images of these two collections, positive examples for 30-degree and 60-degree cascades were generated. As for negative

examples, 1000 random nonface images of any resolution were collected. The initial face detector was successfully trained on multi-processor machines, that are available via METACentrum project. On Manwe (8x dual-core AMD Opteron 885, 2.6 GHz) it took several hours to get a single cascade. After training, all cascades were put together to form a complete face detector, that was finally save into a XML file.

As for the face detection, on a 320 x 240 pixel image, the 360-degree try-all-profiles detector takes about 21 seconds to scan all subwindows by using a 2.4 GHz Intel Xeon, which is about 32 times longer than a try-all-rotations detector of Viola and Jones that was tested on a 2.4 GHz Pentium 4. Mostly, this wide difference in speed is caused by building a non-optimal cascade. Therefore, an extra stage post-optimization procedure [82] [83] should be implemented in future as well as any rotation-class estimator in order to significantly increase the overall performance of the detection system.

The final face detector was tested on the CMU Image Database that contains 208 images with 347 profile faces (faces between ¾ view and full profile). The detector reached a detection rate of 39.0% with 194 false positives. On the second test set, which contains 33 images with 210 both frontal and profile faces (faces between frontal view and 45-degree profile view), the system detected about 80% of the faces with 301 false positives.

As mentioned above, this implementation of AdaBoost-based detector can be extended by employing a stage post-optimization procedure and the rotation-class estimator for purposes of increasing the overall performance of the final detection system.

7 REFERENCES

[1] Yang, M.-H., Kriegman, D. J., Ahuja, N., "Detecting Faces in Images: A Survey", *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, vol. 24, no. 1, pp. 34-58, 2002

[2] Yang, G., Huang, T. S, "Human face detection in complex background", *Pattern Recognition*, vol. 27, no. 1, pp. 53-63, 1994

[3] Kotropoulos, C., Pitas I., "Rule-based face detection in frontal views", *ICASSP '97*, vol. 4, pp. 2537-2540, 1997

[4] Chupeau, B., Tollu, V., Stauder, J., Grasland, I., "Human face detection for automatic classification and annotation of personal photo collections", *Visual Communications and Image Processing*, vol. 5150 (3), pp. 1848-1856, 2003

[5] Dai, Y., Nakano, Y., "Face-Texture Model Based on SGLD and Its Application in Face Detection in a Color Scene", *Pattern recognition*, vol. 29, no. 6, pp. 1007-1017, 1996

[6] Yow, K. C., Cipolla, R., "Feature-Based Human Face Detection", *Image and Vision Computing*, vol. 15, no. 9, pp. 713-735, 1997

[7] Zhang, Q., Izquierdo, E., "A Multi-feature Optimization Approach to Object-Based Image Classification", *CIVR (Image and Video Retrieval, 5th International Conference)*, pp. 310-319, 2006

[8] Sakai, T., Nagao, M., Fujibayashi, S., "Line Extraction and Pattern Detection in a Photograph", *Pattern Recognition*, vol. 1, pp. 233-248, 1969

[9] Cootes, T. F., Taylor, C. J., "Locating Faces Using Statistical Feature Detectors", *Proc. Second Int'l Conf. Automatic Face and Gesture Recognition*, pp. 204-209, 1996

[10] Osuna, E., Freund, R., Girosi, F., "Training Support Vector Machines: An Application to Face Detection", *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, pp. 130-136, 1997

[11] Rowley, H., Baluja, S., Kanade, T., "Neural Network-Based Face Detection", *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 20, no. 1, pp. 23-38, 1998

[12] Turk, M., Pentland, A., "Eigenfaces for Recognition", *J. Cognitive Neuroscience*, vol. 3, no. 1, pp. 71-86, 1991

[13] Sung, K.-K., Poggio, T., "Example-Based Learning for View-Based Human Face Detection", *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 20, no. 1, pp. 39-51, 1998

[14] Schneiderman, H., Kanade, T., "Object Detection Using the Statistics of Parts", *International Journal of Computer Vision*, vol. 56, no. 3, pp. 151-177, 2004

[15] Jones, M., Viola, P., "Fast Multi-view Face Detection", [MERL Tech. rep. TR2003-96], Mitsubishi Electric Research Laboratories, Cambridge, MA, 2003

[16] Kanade, T., "Picture Processing by Computer Complex and Recognition of Human Faces", [PhD thesis], Kyoto Univ., 1973

[17] Wikipedia, the free encyclopedia, "YCbCr", 2008. Document available from URL http://en.wikipedia.org/wiki/YCbCr (December 2008).

[18] Serge Beucher, "The Watershed Transformation", 2000. Document available from URL http://cmm.ensmp.fr/~beucher/wtshed.html (December 2008).

[19] Sai Ho Kwok, Constantinides, A.G., Wan-Chi Siu, "An efficient recursive shortest spanning tree algorithm using linking properties", *Circuits and Systems for Video Technology, IEEE Transactions on*, vol. 14, issue 6, pp. 852-863, June 2004

[20] Brand J., Mason J., "A comparative assessment of three approaches to pixel-level human skin-detection", *Proc. of the International Conference on Pattern Recognition*, vol. 1, pp. 1056–1059, 2000

[21] Yang, M.-H., Ahuja, N., "Detecting Human Faces in Color Images", *Proc. IEEE Int'l Conf. Image Processing*, vol. 1, pp. 127-130, 1998

[22] Augusteijn, M. F., Skufca, T. L., "Identification of Human Faces through Texture-Based Feature Recognition and Neural Network Technology", *Proc. IEEE Conf. Neural Networks*, vol. 1, pp. 392-398, 1993

[23] Haralick, R. M., Shanmugam, K., Dinstein, I., "Texture Features for Image Classification", *IEEE Trans. Systems, Man, and Cybernetics*, vol. 3, no. 6, pp. 610-621, 1973

[24] Orr, G., "Kohonen's Self-Organizing Map (SOM)", 1999. Document available from URL http://www.willamette.edu/~gorr/classes/cs449/Unsupervised/SOM.html (December 2008).

[25] Wikipedia, the free encyclopedia, "YIQ", 2008. Document available from URL http://en.wikipedia.org/wiki/YIQ (December 2008).

[26] Yow, K. C., Cipolla, R., "A Probabilistic Framework for Perceptual Grouping of Features for Human Face Detection", *Proc. Second Int'l Conf. Automatic Face and Gesture Recognition*, pp. 16-21, 1996

[27] Wikipedia, the free encyclopedia, "Bayesian network", 2008. Document available from URL http://en.wikipedia.org/wiki/Bayesian_network (December 2008).

[28] Leung, T. K., Burl, M. C., Perona, P., "Finding Faces in Cluttered Scenes Using Random Labeled Graph Matching", *Proc. Fifth IEEE Int'l Conf. Computer Vision*, pp. 637-644, 1995

[29] Sirohey, S. A., "Human Face Segmentation and Identification", [Technical Report CS-TR-3176], Univ. of Maryland, 1993

[30] Amit, Y., Geman, D., Jedynak, B., Wechsler, H., et al., "Efficient Focusing and Face Detection", *Face Recognition: From Theory to Applications*, vol. 163, pp. 124-156, 1998

[31] Graf, H. P., Chen, T., Petajan, E., Cosatto, E., "Locating Faces and Facial Parts", *Proc. First Int'l Workshop Automatic Face and Gesture Recognition*, pp. 41-46, 1995

[32] Permuter, H. H., Francos, J., Jarmyn, I. H., "Gaussian mixture models of texture and colour for image database retrieval", *IEEE International Conference on Acoustics, Speech, and Signal Processing*, pp. -, 2003

[33] Won, C. S., Park, D. K., Park, S.-J., "Efficient use of MPEG-7 edge histogram descriptor", *ETRI Journal*, vol. 24, no. 1, pp. 23-30, 2002

[34] Chapelle, O., Haffner, P., Vapnik, V. N., "Support vector machines for histogram-based image classification", *IEEE Trans. on Neural Networks*, vol. 10, no. 5, pp. 1055-1064, 1999

[35] Wu, H., Chen, Q., Yachida, M., "Face Detection from Color Images Using a Fuzzy Pattern Matching Method", *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 21, no. 6, pp. 557-563, 1999

[36] Sirovich, L., Kirby, M., "Low-dimensional procedure for the characterization of human faces", *Journal of Optical Society of America*, vol. 4, no. 3, pp. 519-524, 1987

[37] Kirby, M., Sirovich, L., "Application of the Karhunen-Loe`ve Procedure for the Characterization of Human Faces", *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 12, no. 1, pp. 103-108, 1990

[38] Wikipedia, the free encyclopedia, "Cross-correlation", 2008. Document available from URL http://en.wikipedia.org/wiki/Cross-correlation (December 2008).

[39] Marius, D., Pennathur, S., Rose, K., "Face Detection Using Color Thresholding, and Eigenimage Template Matching", 2003. Document available from URL http://www.stanford.edu/class/ee368/Project_03/Project/reports/ee368group15.pdf (December 2008).

[40] Miao, J., et al., "A Hierarchical Multiscale and Multiangle System for Human Face Detection in a Complex Background Using Gravity-Center Template", *Pattern Recognition*, vol. 32, no. 7, pp. 1237-1248, 1999

[41] Craw, I., Ellis, H., Lishman, J., "Automatic Extraction of Face Features", *Pattern Recognition Letters*, vol. 5, pp. 183-187, 1987

[42] Craw, I., Tock, D., Bennett, A., "Finding Face Features", *Proc. Second European Conf. Computer Vision*, pp. 92-96, 1992

[43] Samal, A., Iyengar, P. A., "Human Face Detection Using Silhouettes", *Int'l J. Pattern*

Recognition and Artificial Intelligence, vol. 9, no. 6, pp. 845-867, 1995

[44] Yuille, A., Hallinan, P., Cohen, D., "Feature Extraction from Faces Using Deformable Templates", *Int'l J. Computer Vision*, vol. 8, no. 2, pp. 99-111, 1992

[45] Wikipedia, the free encyclopedia, "Active shape model", 2008. Document available from URL http://en.wikipedia.org/wiki/Active_shape_model (December 2008).

[46] Shuicheng, Y., et al., "Texture-Constrained Active Shape Models", 2002. Document available from URL http://people.csail.mit.edu/celiu/pdfs/GMBV_TCASM.pdf (December 2008).

[47] Kass, M., Witkin, A., Terzopoulos, D., "Snakes: Active Contour Models", *Proc. First IEEE Int'l Conf. Computer Vision*, pp. 259-269, 1987

[48] Lam, K., Yan, H., "Fast Algorithm for Locating Head Boundaries", *J. Electronic Imaging*, vol. 3, no. 4, pp. 351-359, 1994

[49] Shawe-Taylor, J., Cristianini, N., "Support Vector Machines and other kernel-based learning methods", Cambridge University Press, 2000

[50] Sewell, M., "Support Vector Machines (SVMs)", 2008. Document available from URL http://www.svms.org/ (December 2008).

[51] Vapnik, V. N., "The Nature of Statistical Learning Theory", Springer-Verlag New York, Inc., 1995

[52] Li, Y., Gong, S., Sherrah, J., Liddell, H., "Support vector machine based multi-view face detection and recognition", *Image and Vision Computing*, vol. 22, no. 5, pp. 413-427, 2004

[53] Haykin, S., "Neural networks. A comprehensive foundation", Prentice Hall, 1999

[54] Wikipedia, the free encyclopedia, "Artificial neural network", 2008. Document available from URL http://en.wikipedia.org/wiki/Artificial_neural_network (December 2008).

[55] Rowley, H., Baluja, S., Kanade, T., "Rotation Invariant Neural Network-Based Face Detection", *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, pp. 38-44, 1998

[56] Agui, T., et al., "Extraction of FaceRecognition from Monochromatic Photographs Using Neural Networks", *Proc. Second Int'l Conf. Automation, Robotics, and Computer Vision*, vol. 1, pp. 18.8.1-18.8.5, 1992

[57] Juell, P., Marsh, R., "A Hierarchical Neural Network for Human Face Detection", *Pattern Recognition*, vol. 29, no. 5, pp. 781-787, 1996

[58] Lin, S.-H., Kung, S.-Y., Lin, L.-J. , "Face Recognition/Detection by Probabilistic Decision-Based Neural Network", *IEEE Trans. Neural Networks*, vol. 8, no. 1, pp. 114-132, 1997

[59] Wikipedia, the free encyclopedia, "Eigenface", 2008. Document available from URL http://en.wikipedia.org/wiki/Eigenface (December 2008).

[60] Wikipedia, the free encyclopedia, "Eigenvalue, eigenvector and eigenspace", 2008. Document available from URL http://en.wikipedia.org/wiki/Eigenvector (December 2008).

[61] Wikipedia, the free encyclopedia, "Principal components analysis", 2008. Document available from URL http://en.wikipedia.org/wiki/Principal_components_analysis (December 2008).

[62] Nikolaev, N. Y., "Multilayer Perceptrons", 2008. Document available from URL http://homepages.gold.ac.uk/nikolaev/311multi.htm (December 2008).

[63] Schneiderman, H., Kanade, T., "A Statistical Method for 3D Object Detection Applied to Faces and Cars", *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, vol. 1, pp. 746-751, 2000

[64] Morrell, D., "Statistical Pattern Recognition, Lecture Note 1: Bayesian Decision Theory", 1996. Document available from URL http://cmp.felk.cvut.cz/cmp/courses/recognition/resources/PR-uni-arizona/lecture1.pdf (December 2008).

[65] Viola, P., Jones, M., "Rapid object detection using a boosted cascade of simple features", *Computer Vision and Pattern Recognition*, vol. 1, pp. I-511 - I-518, 2001

[66] Shakhnarovich, G., Viola, P. A., Moghaddam, B., "A unified learning framework for real time face detection and classification", *International Conference on Automatic Face and Gesture Recognition*, pp. 14-21, 2002

[67] Wikipedia, the free encyclopedia, "AdaBoost", 2008. Document available from URL http://en.wikipedia.org/wiki/Adaboost (December 2008).

[68] Papageorgiou, C., Oren, M., Poggio, T., "A general framework for object detection", *International Conference on Computer Vision*, pp. 555-562, 1998

[69] Wikipedia, the free encyclopedia, "Haar-like features", 2008. Document available from URL http://en.wikipedia.org/wiki/Haar-like_features (December 2008).

[70] Jones, M., Viola, P., "Face Recognition Using Boosted Local Features", [MERL Tech. rep. TR2003-25], Mitsubishi Electric Research Laboratories, Cambridge, MA, 2003

[71] Zhang, Z. Q., Li, M. J., Li, S. Z., Zhang, H. J., "Multi-View Face Detection with FloatBoost", *Sixth IEEE Workshop on Applications of Computer Vision*, pp. 184, 2002

[72] Li, S. Z., Zhang, Z. Q., Shum, H., Zhang, H. J., "FloatBoost Learning for Classification", 2002. Document available from URL http://books.nips.cc/papers/files/nips15/AA65.pdf (December 2008).

[73] Yang, M.-H., Roth, D., Ahuja, N., "A SNoW-Based Face Detector", *Neural Information*

Processing Systems, pp. 862-868, 1999

[74] Samaria, F.S., "Face Recognition Using Hidden Markov Models", [PhD thesis], University of Cambridge, 1994

[75] Samaria, F., Young, S., "HMM Based Architecture for Face Identification", *Image and Vision Computing*, vol. 12, pp. 537-583, 1994

[76] Lew, M. S., "Information Theoretic View-Based and Modular Face Detection", *Second International Conference on Automatic Face and Gesture Recognition*, pp. 198-203, 1996

[77] Huang, J., Gutta, S., Wechsler, H., "Detection of Human Faces Using Decision Trees", *Second International Conference on Automatic Face and Gesture Recognition*, pp. 248-252, 1996

[78] Duta, N., Jain, A. K., "Learning the Human Face Concept from Black and White Pictures", *Proc. International Conference on Pattern Recognition*, pp. 1365-1367, 1998

[79] Rowley, H. A., "Henry A. Rowley", 1998. Document available from URL http://www.cs.cmu.edu/~har/ (December 2008).

[80] Meyneta, J., Popovici, V., Thiran, J.-P., "Face detection with boosted Gaussian features", *Pattern Recognition*, vol. 40, issue 8, pp. 2283-2291, 2007

[81] Wikipedia, the free encyclopedia, "Receiver operating characteristic", 2008. Document available from URL http://en.wikipedia.org/wiki/ROC_curve (December 2008).

[82] Masnadi Shirazi, H., Vasconcelos, N., "High Detection-rate Cascades for Real-Time Object Detection", *ICCV07*, pp. 1-6, 2007

[83] Lienhart, R., Maydt, J., "An Extended Set of Haar-Like Features for Rapid Object Detection", *IEEE ICIP 2002*, vol. 1, pp. 900-903, 2002

8 ANNEXES

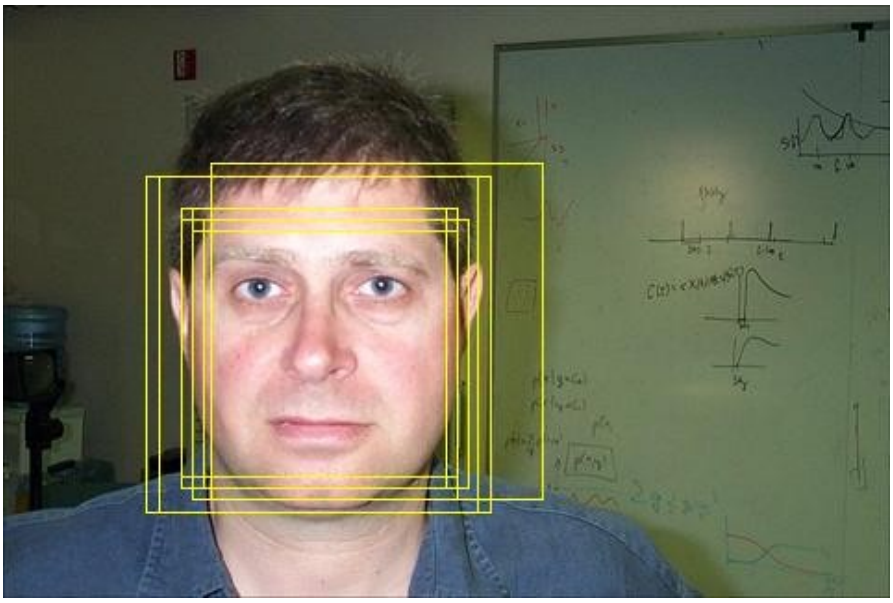

Figure 47a. During scanning the input image, multiple detections of a single face may occur.

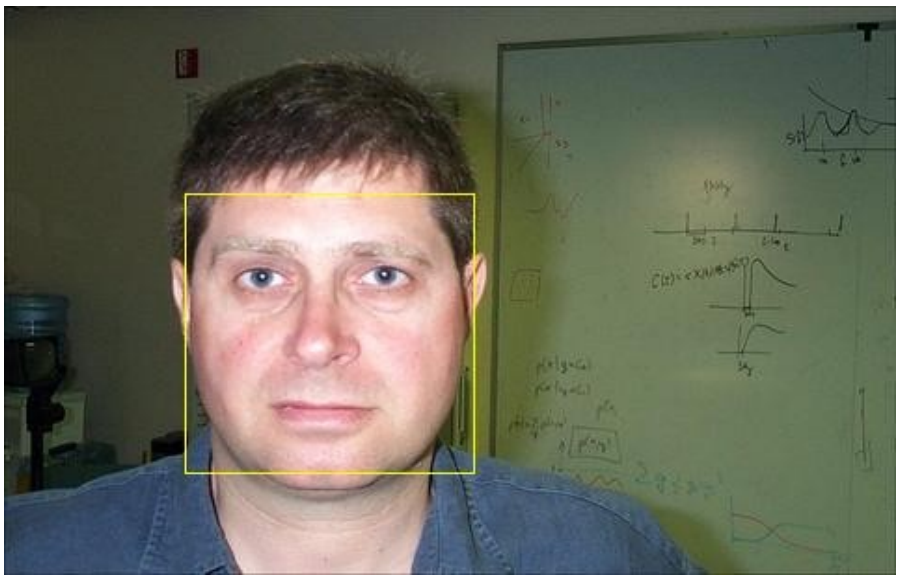

Figure 47b. Multiple detections are merged into a single average detection.

Figure 48. In this image 28 of 29 faces were successfully detected with 7 false positives by applying the 360-degree try-all-profiles detector.

Figure 49. The final detector is able to deal with various in-plane rotations. Due to a simple background, no false alarms had occurred during detection.

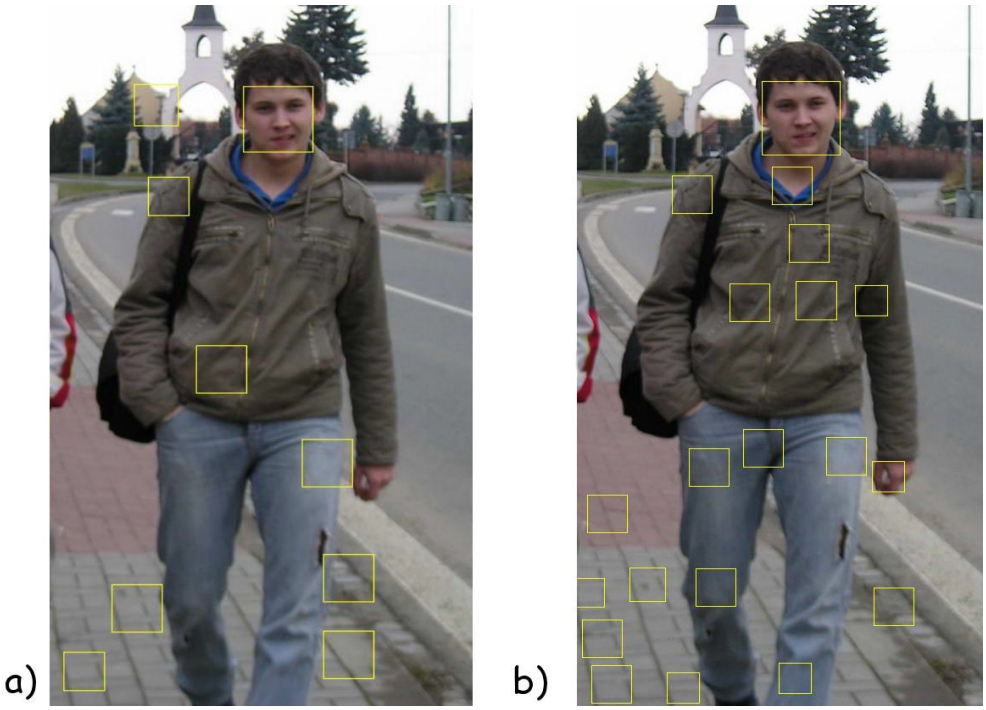

Figure 50. Influence of image resolution on a false alarm rate. In a 1024x768 pixel image (a) 1 face was detected with 8 false positives only, while in the same image (b) with resolution of 1280x960 pixels the face was detected with 19 false positives.

Figure 51. In this test image 9 of 12 faces were correctly detected with 11 false positives.

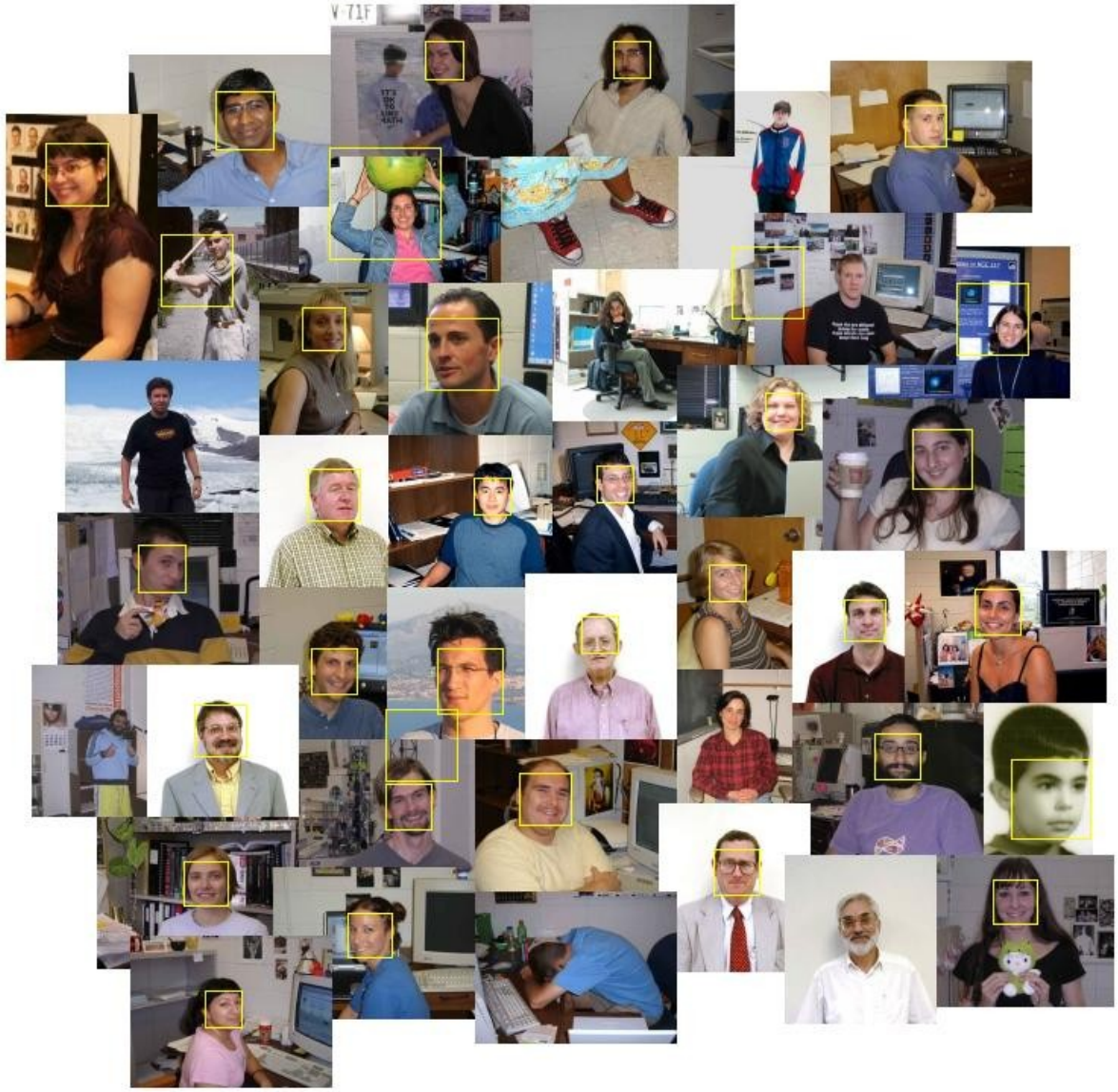

Figure 52. Example detections for the 360-degree try-all-rotations detector. Within this test image about 29 of 36 faces were successfully detected with 4 false alarms (or inaccurate detections) only.