Face Pose Estimation using a Tree of Boosted Classifiers

Javier Cruz Mota
Project Assistant: **Julien Meynet**
Professor: **Jean-Philippe Thiran**

Signal Processing Institute,
École Polytechnique Fédérale de Lausanne (EPFL)

September 11, 2006

A mis padres, Juan e Inés

Acknowledgements

No podría empezar esta sección de agradecimientos sin dar el reconocimiento que se merece a mis padres, por haberme apoyado y motivado en todo momento a lo largo de estos 6 años. Parecía que nunca iba a llegar el final, pero con este documento finalizo esta etapa, muchas gracias por todo! Y gracias a vosotras también, Cristina y Silvia, mis dos hermanas, las que viven en la habitación de al lado y que tanto me dan la tabarra, pero a las que tanto quiero. Y por supuesto que tampoco puedo olvidarme del apoyo que he recibido de toda mi familia, de mis abuelos, mis tíos, mis primos (los que ya estaban y los que acaban de llegar)... a todos vosotros, muchas gracias!

Moltes gràcies a tu també Vero, si no fos per tu segurament mai m'hauria vingut al cap l'idea de venir a passar un temps a Suissa, i el que és més important, si no fos per tu no ho hauria disfrutat tant. A més, aquest temps també ens ha servit a nosaltres per conèixe'ns millor i descobrir que els dos junts ens ho passem molt bé!

No m'oblido de tu Ferran! Moltes gràcies per tot el que has fet per mi durant aquest últim any meu d'universitat, desde el facilitar-me la vinguda cap a Lausanne, fins al fabulós mes de Workshop a Dubrovnik, passant per aquells dies interminables preparant el Mock-Up de la sala CHIL o fent simulacions per l'article de l'ICIP. Moltes gràcies, he aprés moltíssim!

Je vous remercie aussi à vous, Jean-Philippe et Julien, pour votre constant soutien et conseil, sans lesquelles je n'aurais pas réussi de faire ce projet. Et merci aussi Jean-Philippe pour votre aide pour pouvoir assister au Workshop eNTERFACE, où j'ai appris beaucoup de choses a niveau académique et personnel.

Hi ha un altre grupet de gent a la que no puc oblidar tampoc a l'hora de donar gràcies, els meus companys CFISes y $\overline{\text{CFISes}}$ que m'han acompanyat durant tot aquest temps a la universitat. Vam comenar sent això, companys, però ara són veritables amics. Què hauria estat de nosaltres en la llunyania sense la llista SPAMica! I ja que he tret la paraula CFIS, també vull agrair des d'aquí a tota la gent del CFIS pel seu treball constant per facilitar-nos tant les coses.

I ja per finalitzar, també voldria donar les gràcies a tota la gent que m'he creuat durant l'Erasmus i que ha fet que a Lausanne em sentís com a casa meva. Gràcies Pablo, Àlex, Anna, Judith, Tibor, Louis, Julie, Kathrin, David, Ceci, David (bis), Borja, Ester, Jaume, Laure, Serge...

Merci beaucoup à tous!

Javier Cruz Mota

Abstract

Face detection in images or video sequences is a very challenging problem. It has a wide range of applications but at the same time it presents a great number of difficulties, since faces are non-rigid and very changeable objects that can adopt a lot of different poses and with a high inter and intra-person variation and a high sensitivity to lighting conditions.

Along this document, a new approach to the face detection and pose estimation problem is given. This approach is based on the method proposed by Viola and Jones in [1] but considering a wide range of face poses, varying the elevation and the out-of-plane rotation, and building specific classifiers for each one.

The proposed method can be easily adapted to consider other poses or to detect other objects. Especially, this approach is interesting when an object that can adopt several positions want to be detected, since the partition of the pose space allows to build classifiers specialised in only one or a few poses, which limits the large variance of the "global" class, the class containing all the poses.

In order to facilitate the reproduction of all the processes done in this document, we have used standard face datasets to train and test the system.

Contents

List of Figures

List of Tables

List of Algorithms

Part I

Introduction

Chapter 1

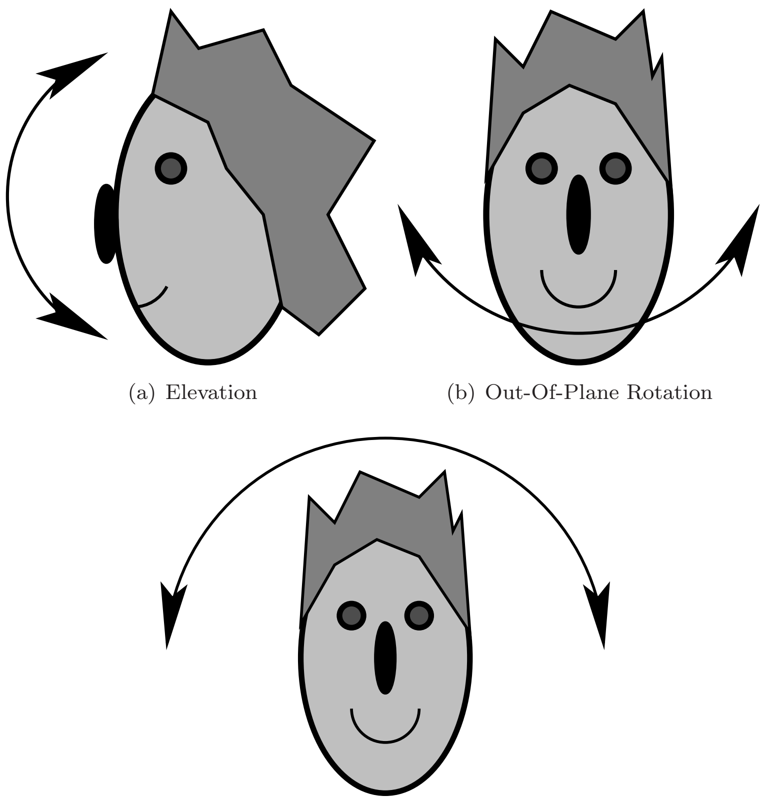
Introduction

Day after day, interaction among people and computers is more important. For this reason, great research efforts are being carried out to provide with human abilities these computers, in order to make this interaction more confortable for humans. One of the most important of these abilities is the sense of sight. This sense allows humans to distinguish objects and people. Thanks to the advances in the power of current computers as well as the advances in image processing and related matters, "Computer Vision" has become a reality. This has generated lots of applications, as well as has created new necessities, all of them with a common point, the exigency of a good face detector algorithm when interacting with humans.

As a face detection algorithm it is understood an algorithm that is able to detect one or more faces in an image or a video sequence. A face is a non-rigid and very changeable object that varies a lot from one person to another. In addition, faces are very sensitive to lighting conditions and, as a 3D object, they can adopt lots of poses, combination of the three basic movements: out-of-plane rotation, in-plane rotation and elevation (see figure (1.1)), as well as to be partially occluded by glasses, moustaches, hats, other body parts... All these reasons make face detection a very difficult and challenging problem. But apart of being a challenging problem it has innumerable applications going from *autonomous surveillance* to *intelligent human-computers interfaces*, passing through *biometric identification*, *video conferencing* or *image and video indexing*.

- In *Biometric Identification*, face detection is a crucial preprocessor block, when face recognition wants to be used, since without a good facial detection is not possible to do an acceptable recognition. An study on this can be seen in [2].

- Also in the design of *Intelligent Human-Computers Interfaces*, face detection is one of the main parts. In that kind of interfaces, it is important for the computer to be able to recognise people and to identify facial expressions or even to read lips. A previous face detection is crucial for all these functions, whose success depends mainly on the facial detector accuracy.



(a) Elevation

(b) Out-Of-Plane Rotation

(c) In-Plane Rotation

Figure 1.1: Basic poses that a face can adopt

- *Video Conferencing* has become a very useful and very used application thanks to the increasing capacity of networks. This application needs also a good face detection preprocessing to guarantee that the current speaker has always the focus, allowing this way also to optimise the video compression and the flow of data sent to the network.

- In the age of the "war" between search engines which we are living nowadays, lots of companies compete for having the most complete database with the greatest number of indexed objects. Between these objects there is not only text but also audio, images and video are becoming more and more popular. Is in these two last things, in the *Image and Video Indexing*, where also facial (and in general object) detection is very important since it allows to index automatically by *content*, and not by manually inserted labels as it is the most common way now. This indexing by content constitutes the core of what is known as CBIR (*Content-Based Image Retrieval*), an emerging application that could be classified inside the famous term *Web 2.0*.

- *Automatic* or *Semi-Automatic Surveillance* holds also an important place as a direct application of face detection. In this type of applications, face detection can carry out, without human interaction, the tracking of people recorded by a surveillance camera as well as the shooting of alarms when something unusual is detected.

As it has been showed in the last paragraphs, face detection is an important part of a wide range of emerging applications. In spite of this, "The Method" to detect faces in any pose and in any environment does not exist, although every time more robust methods with less false detections are being developed. This makes face detection a very interesting problem to be studied.

In the following sections, the process to build a tree of classifiers capable of detecting faces and estimating its pose in real-time will be described. As pose will be only considered the out-of-plane rotation and the elevation (see figures (1.1(b)) and (1.1(a))). The other possible movement, the in-plane rotation (see figure (1.1(c))), is not considered since it can be obtained by a simple rotation of the input image, although the scheme described here could be easily generalised to detect the three possible movements. In addition, this movement is not as interesting as the other two, since in most of the applications there are only upright faces. The images the system works with are always in greyscale.

The present document is organised as follows. In Chapter 2 an exhaustive vision over the actual State-of-The-Art in face detection is given. In Chapter 3 all the theoretical aspects of the system are described: the filters which responses will be used by the AdaBoost algorithm in Section 3.1, the AdaBoost algorithm in Section 3.2 and an introduction to decision trees in Section 3.3. Then, the practical implementation of the system is described in Chapter 4, giving an exhaustive description of the training set in Section 4.1, a description of the classifiers structure in Section 4.2, the concrete decision tree structure in Section 4.3 and the explanation of how the system searches for faces in Section 4.4. After that, some test and results are presented in Chapter 5, the conclusions in Chapter 6 and some future work in Chapter 7.

Chapter 2

State-of-The-Art

There exists a great number of techniques in the face detection field of study, some of them with their origin in the early 70's. Depending on the author, these techniques are divided in a different way. According to [3], the different approaches can be divided into 4 different types, *knowledge-based* methods, *feature invariant* methods, *template matching* methods and *appearance-based* methods.

- **Knowledge-based** methods try to encode the human knowledge about what constitutes a face, usually by means of relations between facial features.
- **Feature Invariant** methods find structural face features that remain invariant for changes in the pose, viewpoint or lighting conditions and use them to detect faces.
- **Template Matching** methods compute, to detect faces, the correlation between the input and the stored patterns that represent typical faces or typical facial features.
- **Appearance-based** methods are similar to *Template Matching* methods but instead of having some stored patterns, the patterns are learnt from a set of training images which, supposedly, capture the face's variability.

Nevertheless, that division does not have the boundaries between the different types well-defined, and therefore, in general, it is not easy to classify a method only in one class. For this reason, it results more convenient to use the division proposed in [4] or in [5], that makes use of only two types of methods, dividing all the possible techniques in *feature-based* approaches and *image-based* approaches.

- **Feature-based** approaches, where the first methods (chronologically) can be situated, make use of face appearance properties, such as skin colour or face geometry, to detect faces. Usually, the detection in these methods is accomplished by manipulating distances, angles or areas, inferred by the face appearance knowledge that the method is exploiting.
- **Image-based** approaches are more modern than *feature-based* ones. These methods take advantage of the development and advances in the pattern recognition field of study, and treat the detection of faces problem as a

general problem on pattern recognition. This approaches consider an image, or a subimage when they are searching for a face, as a whole object that has to be classified.

This division is simpler than the first one, and in addition, the boundary between the classes is better defined and a method can be classified, in general, only into one of them. Therefore, this division will be the used one in this document to classify the different exposed methods. However, the word “feature” can create confusion due to its use in both classes. In *Feature-based* methods, the word “feature” means a facial feature, as it can be an eye, a cheek or the mouth, whereas in *Image-based* methods, the word “feature”, when it is used, means the output of a filter applied to an image, and in general it does not have anything to do with an eye or a cheek.

In the following sections, the defined classes are further developed and references to the main works on each area are taken down. Some of the most relevant works are further explained, especially those *image-based* approaches involving machine learning techniques, since they are the base for the work developed in the following chapters.

2.1 Feature-based Approaches

The core techniques used in *feature-based* approaches go from the *low-level analysis* of pixel properties to the more modern *active shape models*, passing through *feature analysis* using information of face geometry. The common point in all of them is the objective of deducing facial features to be able to infer the presence of a face.

2.1.1 Low-level Analysis

Low-level techniques contain some of the first face detection works, as for example [6]. This work is further developed next due to its importance, since it was the first work in facial detection. This kind of approaches exploits, with greater or smaller success, a wide range of pixel-level properties:

- **Edge detections** are the most “primitive” features studied in computer vision [6, 7, 8, 9]. By means of filters, contours are tried to be found, and from them, facial features are inferred.
- **Grey level information** tries to exploit grey level changes around, or in, facial features [10, 11, 12], as for example the darkness of the eyebrows or the brightness of the nose tip.
- **Colour information** is also widely used, despite it is known [13, 14, 15] that human skin colour variation forms a small cluster in colour spaces, even considering different races. To try to minimise this inconvenient and to modelise as well as possible the skin colour, a great variety of different colour spaces has been proposed, like *normalised RGB* [13, 16, 17, 18], *HSI* [19], *YIQ* [20, 21], *YUV* [22, 23] or *CIE* [24].

- **Motion estimation** is very useful when a video sequence is available. In [25, 16, 26] moving silhouettes are searched by a thresholded interframe difference, and in [27], also by means of interframe difference, eyes presence is tried to be inferred from similar movements in adjacent regions. There are also more complex approaches than the interframe difference, as for example in [28], where a spatiotemporal Gaussian filter is introduced to detect moving boundaries.

- **Generalised measures** try to compute, at a low-level, generalised image properties as symmetries [29, 30, 31, 32] or convex and concaves shapes [33].

2.1.1.1 Edge Detection Approach by Sakai

The method introduced by T. Sakai et al. in [6] in 1972 has an important relevance since it was the first work in face detection. The aim of the proposed algorithm was to detect facial features in context-controlled grey-level photographs of full faces without glasses nor beard. If the different facial features were detected, the presence of a face could be inferred. The size of each picture was fixed to 140×208 pixels, and the algorithm was divided in two steps, a first preprocessing step to binarise the picture and a second facial detection step over the binary picture. This method could also be seen as a feature analysis method, since the detection is made detecting facial features. But, the facial features detection is accomplish by a kind of histogram generated (described next) with the binarized image, therefore the analysis done is better seen as a pixel-level analysis.

In the preprocessing step, the authors used a Laplacian operator (2.1) and a later thresholding to binarise the picture.

$$\begin{pmatrix} 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & -4 & -4 & -4 & 1 & 1 & 1 \\ 1 & 1 & 1 & -4 & -4 & -4 & 1 & 1 & 1 \\ 1 & 1 & 1 & -4 & -4 & -4 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \end{pmatrix} \tag{2.1}$$

Facial extraction step was subdivided in several subroutines (described next) following a tree structure as showed in figure (2.1). For the features computation, instead of the original image, it was used an integral projection in the sense of histogram calculation of the values under a mask called by the authors a *slit*. The computed features were detected as follows:

- **Top of head** is detected by descending a horizontal *slit*, of width equal to image width, from the top of the image until a “sufficient” value in the integral projection is obtained. This point, called H, is only used as a delimiter point for the later subroutines, for this reason, the accuracy is not very important.

Table 2.1: Test Results obtained by Sakai et al. in [6]

Category	**Num. of faces**	**Correct detections**	**Errors or fails**
“Standard” frontal faces (no glasses, no beard)	607	552	55
Frontal faces with glasses	77	0	77
“Standard” turned faces	79	63	16
Faces with beard	25	0	25

- **Sides of face at cheeks** are detected by scanning the image from a point below H by a horizontal *slit* of a determined width h. At the height of the cheeks, two “sandwiches” formed by the projections of the nose and the limit of the face at each side are detected, which allows to define the *sides of face at cheeks*. The obtained points are called L (for the left one) and R (for the right one).
- **Vertical positions of nose, mouth and chin** are encountered by placing a vertical *slit* of width $L * R/4$ at the centre of L and R. If found, the lower end of the nose, the upper lip and the chin are labelled as N, M and C respectively.
- **Chin contour** is determined by placing *slits* along lines drawn from M downward every 10 degrees. After this detection, a line smoothing process is performed.
- **Nose width** is calculated by tracing the nose by horizontal *slits* placed starting from N. Nose width is determined by points P and Q, located at the end of the nose at each side.
- **Eye positions** are placed by fusion and shrinking and a later connection detection over a *slit* situated over the nose. The eyes’ centres are labelled as S for the left and T for the right.
- Finally, the **face axis** is determined as the line throw $X = (S + T)/2$ and $Y = (P + Q)/2$.

The proposed approach to the face detection problem was tested over 788 images, not all of them with the assumptions of no glasses and no beard. The obtained results are summarised in table (2.1), where a correct detection means that all facial features have been correctly detected. As it can be seen, results were quite good, with about a 90% of correct detections in images with the assumptions of no glasses and no beard.

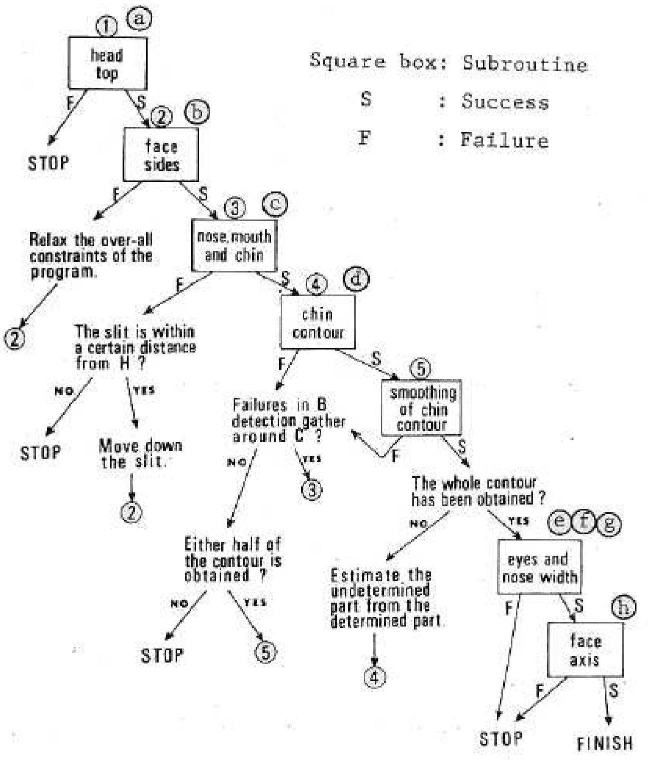

Square box: Subroutine
S : Success
F : Failure
head top
face sides
nose, mouth and chin
chin contour
smoothing of chin contour
eyes and nose width
face axis
STOP
Relax the over-all constraints of the program.
The slit is within a certain distance from H ?
Move down the slit.
Failures in B detection gather around C ?
Either half of the contour is obtained ?
The whole contour has been obtained ?
Estimate the undetermined part from the determined part.
FINISH

Figure 2.1: Facial extraction algorithm proposed by Sakai et al. in [6]

2.1.2 Feature Analysis

Feature Analysis methods were developed to eliminate the ambiguity existent when working only with local pixel properties, when for example background with a certain colour can be easily confused as part of a face. These methods make use of more global properties, such as face geometry, to make detections more robust. There are basically two classes of approaches inside this category, *Feature Searching* approaches and *Constellation Analysis* approaches.

- **Feature Searching** techniques search for an easily seen facial feature through which the presence of not so prominent features is inferred using anthropometric knowledge. If the inferred features are found, the confidence of the detection is expected to be high. As easily seen facial features can be used the eyes [25, 10, 27], a symmetry facial axis [34] or the head outline [6, 34, 35].

- **Constellation Analysis** techniques are less rigid than *Feature Searching* ones. Instead of searching for a prominent feature, *Constellation Analysis* methods group facial features in face-like constellations using statistical techniques to have a more robust structure. Several face constellations have been proposed in [36, 37, 38].

2.1.2.1 Constellation Analysis Approach by Lin

Lin et al. introduced in [38] a face detection algorithm based on *Constellation Analysis.* The system is capable to detect faces in grey scale images with different poses and complex backgrounds. The designed algorithm has two main parts, the first part searches for potential face regions and the second one verifies the existence of a face in each potential region. To achieve the face detection purpose, the authors defined a constellation composed by eyes and mouth for frontal faces, and an eye, the mouth and an ear for lateral faces.

To search for face candidates, the first part of the algorithm carries out 3 tasks. First of all, it binarizes the input image. Since the objects of interests (eyes, mouth and ears) are in general darker than the background, a threshold T is fixed and all pixels with a higher grey level are labelled as white (0) and those with an equal or lower value are labelled as black (1). After this binarization, all 4-connected blocks are searched and their centres are labelled. Finally, from these labelled centres, all isosceles triangles (potential frontal faces) and all right triangles (potential lateral faces) are searched (see figure (2.2)). These isosceles and right triangles are the potential detected faces.

The second part of the algorithm performs a face verification over the set of isosceles and right triangles. To make this, it firstly scales the input regions to a standard size of 60×60 pixels using a bicubic interpolation. After this scaling process, a weight value is calculated for each candidate scaled region by means of a previously calculated mask. This mask is calculated by binarizing the sum of 10 binary face samples. With the obtained mask, the final weight is calculated as indicated in algorithm (1). Finally, all candidate regions with a weighted value below a threshold, that is empirically set, are eliminated.

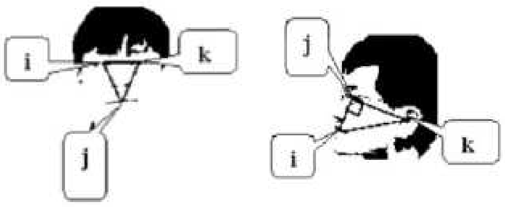


Figure 2.2: Frontal isosceles triangle and lateral right triangle, defined by facial features and used by Lin et al. in [38]

Algorithm 1: Weighting Algorithm described in [38] by Lin et al.

```
1  Weight = 0;
2  Candidate ⟵ Potential Facial Region;
3  Mask ⟵ Calculated Mask;
4  for $i = 1$ to number of pixels of potential facial region and mask do
5      if Candidate[$i$] == Mask[$i$] == black then
6          Weight = Weight + 6;
7      else if Candidate[$i$] == Mask[$i$] == white then
8          Weight = Weight + 2;
9      else if (Candidate[$i$] == black) AND (Mask[$i$] == white) then
10         Weight = Weight − 4;
11     else if (Candidate[$i$] == white) AND (Mask[$i$] == black) then
12         Weight = Weight − 2;
13     end
14 end
```

The authors tested their algorithm over 500 images containing 600 faces of 450 different people, some of them taken using a digital camera, some from scanned images and some from videotape sequences. Over this set of images, only 11 faces were not detected and the other 589 were correctly detected. With respect to execution time, the authors get only the values for two concrete images of similar size, which are 2.5 seconds and 28 seconds, claiming that the execution time depends on the size of the image and especially on the background complexity, due to the higher number of 4-connected detected components.

2.1.3 Active Shape Models

Active Shape Models were developed to perform nonrigid feature extraction such as lip tracking. An *Active Shape Model* interacts with local image features of a near feature to be deformed gradually until it takes its shape. There are basically 3 types of *Active Shape Models*:

- **Snakes** were the first developed *Active Shape Models* [39] and consist of a generic active contour that is progressively deformed to minimise an energy function. They are usually used to locate a head boundary [8, 40, 41, 42].
- **Deformable Templates**, introduced in [43], were an evolution of *Snakes* that improved its performance and took into account the a priori expected shape. This is achieved by defining shape classes for every facial feature. Some evolutions of this technique, which are basically appointed to reduce computation times, can be found in [8, 44, 45].
- **Point Distributed Models** [46] are parametrised descriptions of shapes based on statistics. The fitting process is made by discretizing the contour of the model into a set of points that are varied of position according to a previous training process. *Point Distributed Models* were first used in face detection in [47], and they are also used in facial expression interpretation [48] or in face recognition [49].

2.1.3.1 Snakes

A *Snake* is an active contour model introduced in [39] by Kass et al. As their authors defined it, a *Snake* is an energy-minimising spline guided by external constraints and influenced by lines, edges and subjective contours in an image. *Snakes* allow an accurate detection of contours, for this reason they are broadly used in image processing.

As it has been already commented, a *Snake* is an energy-minimising spline, i.e. is a spline that searches the position with less energy according to a pre-defined energy function. If the position of a *Snake* is expressed by means of its arc-length parameter it is obtained $v(s) = (x(s),\ y(s))$, and its energy function can be expressed as

$$\begin{aligned} E_{\text{snake}} &= \int_0^1 E_{\text{snake}}(v(s))ds \\ &= \int_0^1 (E_{\text{int}}(v(s)) + E_{\text{image}}(v(s)) + E_{\text{con}}(v(s)))ds \qquad (2.2) \end{aligned}$$

where E_{snake} and $E_{\text{snake}}(v(s))$ represent the total energy and the energy in $v(s)$, respectively, of the Snake, E_{int} the internal energy of the spline, E_{image} the energy of the image under the *Snake* and E_{con} the energy due to external constraints. The internal spline energy is usually expressed as

$$E_{\text{int}} = \frac{\alpha(s)\left\|\frac{\partial v(s)}{\partial s}\right\|^2 + \beta(s)\left\|\frac{\partial^2 v(s)}{\partial s^2}\right\|^2}{2} \tag{2.3}$$

where $\alpha(s)$ and $\beta(s)$ are parameters that control the snake continuity properties. The constraint energy has a great number of expressions, even sometimes it is considered 0, but usually it is expressed by means of two parameters called *spring* factor and *volcano* factor. But the important term in equation (2.2) is E_{image}, since this term fixes the image features that the *Snake* will capture. For instance, to capture lines, the authors propose to use $E_{\text{image}} = I(x, y)$, where $I(x, y)$ is the intensity value in (x, y), or $E_{\text{image}} = -\|\nabla I(x, y)\|^2$ to capture edges.

Although *Snakes* are accurate and robust, a good initialisation must be done in order to capture those features that are expected to be captured, and not minimise the energy function around another similar feature. This represents a problem usually solved by pre-detecting the features with more robust but less accurate algorithms, as in [8], where Huang et al. proposed a method that used *Snakes* to improve a face contour detection done with a *Rough Contour Estimation Routine*.

2.2 Image-based Approaches

The presented methods until here were based on explicit facial features modelling. The *feature-based* approaches, despite that some well results have been reported, have some problems with the unpredictability of face appearance and, in general, they are quite limited to quasi-frontal high-resolution faces. *Image-based* approaches can work in less limited situations, and by means of a common technique of scanning the image for every position and for several scales, multiple faces and semi-hidden faces can be found. In addition, to reformulate the face detection problem as a pattern recognition problem erases the setback of quasi-frontal faces, since now, the aim is to classify an input into a "face" or a "non-face" class, thing that the algorithms accomplish by means of a training process.

Image-based approaches to the face detection problem can be divided into three big families: *Linear Subspace* methods, *Machine Learning* methods and pure *Statistical* methods.

2.2.1 Linear Subspace Methods

Linear Subspace methods are based in the generation of a subspace, inside the image space, where any face can be represented. To represent this subspace, several multivariate statistical techniques like *Principal Component Analysis* (PCA) [50, 51, 52], *Linear Discriminant Analysis* (LDA) [53, 54, 55] or *Factor Analysis* (FA) [55] can be applied.

2.2.1.1 Eigenfaces Approach

L. Sirovich and M. Kirby introduced in [50] a new method to efficiently represent human faces based in SVD (Singular Value Decomposition) theory. The method proposed is explained in the following paragraphs in detail due to its importance, although avoiding some mathematical justifications.

Given a set of N gray scale face images $\underline{\varphi}_1, \ldots, \underline{\varphi}_N$ with M pixels each one ($M \gg N$ in general), where each image is stored as a vector by concatenation of rows, the average face can be computed as follows:

$$\underline{\overline{\varphi}} = \frac{1}{N} \sum_{i=1}^{N} \underline{\varphi}_i \tag{2.4}$$

With this value, the deviation of each face from the mean is defined as

$$\underline{\phi}_i = \underline{\varphi}_i - \underline{\overline{\varphi}} \tag{2.5}$$

where each $\underline{\phi}_i$ is called a caricature.

Considering the set of caricatures, the covariance matrix, that is symmetric and nonnegative, can be estimated as

$$\underline{\underline{C}} = \frac{1}{N} \sum_{i=1}^{N} \underline{\phi}_i \underline{\phi}_i^\top \tag{2.6}$$

or specifying the expression for a concrete spacial point (x, y) it is obtained

$$C(x, y) = \frac{1}{N} \sum_{i=1}^{N} \phi_i(x) \phi_i(y) \tag{2.7}$$

Taking a limit when $N \rightarrow \infty$ over equation (2.7), the conditions to apply the Mercer's Theorem [56] are accomplished and then, $C(x, y)$ can be expressed as follows in the sense of L^2 convergence:

$$C(x, y) = \sum_{i=1}^{\infty} \lambda_i u_i(x) u_i(y) \tag{2.8}$$

where $u_i(x)$ is the orthonormal eigenfunction set of corresponding eigenvalues λ_i.

With this theoretical account, it is immediate to define what the authors called an eigenpicture, known broadly today as an eigenface, which is the concatenation of values for each spatial point of each eigenfunction with $i \in [1, N]$, i.e.,

$$\underline{u}_i = (u_i(1), \ldots, u_i(M)),\ i = 1, \ldots, N \tag{2.9}$$

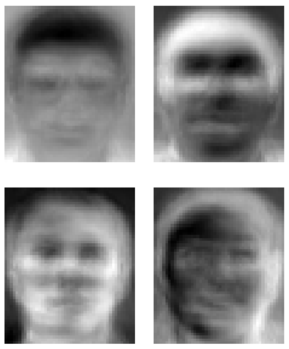


Figure 2.3: Example of 4 eigenfaces

This set of eigenfaces, when $N \to \infty$, constitute an orthonormal basis of the face space, and then each face image can be approximated by its decomposition in the N calculated eigenfaces as

$$\underline{\varphi}_i = \sum_{j=1}^{N} a_{ij} \underline{u}_j \tag{2.10}$$

In other words, what it is explained above means that each face can be expressed as an infinite sum of eigenfaces, and therefore approximated (in the L^2 sense) by a finite sum. This fact can be used by computing projections onto the eigenface's space to detect faces or facial features as introduced in [52] or to identify faces as introduced in [51]. An example of some eigenfaces, obtained by the procedure described above, can be seen in figure (2.3).

2.2.1.2 Linear Discriminant Analysis

Linear Discriminant Analysis (LDA) is a very used statistical technique. It consists of using linear discriminant functions to construct, by means of a training process, a piecewise linear function capable of classifying a given observation. What follows is a brief explanation of LDA to the 2-class case, although it can be "easily" generalised to the N-class case.

Given a set of training patterns $\underline{x}_1, \ldots, \underline{x}_n$, each one assigned to class ω_1 or ω_2, LDA searches for a vector $\underline{v}$ so that

$$\underline{v}^\top \underline{z}_i \begin{cases} > 0 \Rightarrow \underline{x}_i \in \omega_1 \\ < 0 \Rightarrow \underline{x}_i \in \omega_2 \end{cases} \tag{2.11}$$

where $\underline{z}_i = (1, \underline{x}_i^\top)^\top$ is called the augmented pattern vector. If $\underline{v}$ is capable of classifying all the samples well, then the data is said to be linearly separable. It is also possible to have, instead of $\underline{z}_i = (1, \underline{x}_i^\top)^\top$, $\underline{z}_i = (1, \underline{\phi}(\underline{x}_i)^\top)^\top$ and then work in a transformed space.

The objective of the training process to find $\underline{v}$ is to minimise misclassifications. There are a lot of criterions such as the perceptron criterion

$$J_P(v) = \sum_{\underline{y}_i \in \Upsilon} |\underline{v}^\top \underline{y}_i| \tag{2.12}$$

where Υ designates the set of misclassified samples and the optimum $\underline{v}$ is achieved by minimising $J_P(v)$. Another criterion, and probably one of the most famous, is the Fisher's criterion. It searches for the direction that better separates the two classes by means of maximising the ratio between the "between-class" and the "within-class" variances

$$J_F(w) = \frac{|\underline{\omega}^\top (\underline{m}_1 - \underline{m}_2)|^2}{\underline{\omega}^\top \underline{\underline{S}}_W \underline{\omega}} \tag{2.13}$$

where $\underline{\omega}$ is $\underline{v}$ without its first component, $\underline{m}_1$ and $\underline{m}_2$ the class means and $\underline{\underline{S}}_W$ the within-class covariance matrix.

In [55], Yang et al. used LDA with Fisher Linear Discriminant to classify an input as face or non-face. To make this, the authors generated 25 face classes and 25 non-face classes by means of Kohonen's Self-Organising Map [57]. Over these 50 classes, a Fisher Linear Discriminant was calculated. The training was accomplished with 16810 face images generated by randomly rotating and scaling 1681 original faces collected from Olivetti, UMIST, Harvard, Yale and FERET datasets. As non-faces, an initial set of 8422 images was used, increased by using boostraping. With this training, very low false detections and a rate of around a 90% of correct face detections were achieved, considering as correct detections those who include the eyes and the mouth.

2.2.2 Machine Learning Methods

Nowadays, the use of machine learning techniques to tackle pattern recognition problems has become very popular due to its good results. First approaches to face detection using machine learning [58, 59] were based on *Multi-Layer Perceptron*s, MLPs, and very promising results over very simple datasets were obtained. From these first approaches to the more modern as [60, 61, 62, 5], a lot of new techniques such as genetic algorithms or complex learning algorithms have been added to original machine learning approaches. These new techniques have allowed to confirm the promising results over complex datasets.

2.2.2.1 MultiLayer Perceptron Approach by Sung and Poggio

K. Sung and T. Poggio proposed in [63] the system considered as the first advanced image-based approach, in which a "face-like" class is defined to detect faces by means of searching exhaustively all possible locations and scales in the image. Although this approach is situated in "Machine Learning Methods", since it uses a *MultiLayer Perceptron*, it could be also seen as a "Linear Subspace Method" since such a face space is also introduced.

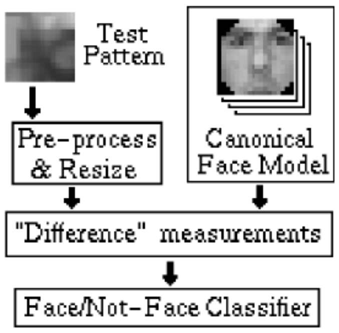


Figure 2.4: Face Pattern Detector proposed by Sung et al. in [63] in detail

The main part of this approach is the perceptron outlined in figure (2.4). This perceptron is composed of a “Canonical Face” model and three subsystems, one to compute the pre-processing and resizing tasks, another one to calculate the differences with the canonical model and a last one to decide whether the input is a face.

For the canonical face model generation, the authors had at their disposal 4150 normalised frontal and slightly rotated faces to modelise the face distribution, and 6189 normalised face-like images to the non-face distribution. With these images, they defined 6 face and 6 non-face pattern clusters, where each cluster is a multidimensional Gaussian distribution with a centroid location and a covariance matrix. These 6 clusters together built a nonisotropic Gaussian mixture model. To compute centroids and covariance matrices, a modified k-means clustering algorithm with a normalised Mahalanobis distance (2.14) were employed. The chosen number of pattern clusters, 6, was empirically selected by the authors so that the face detection rate versus the number of false detection were nearly constant for a slightly fewer or greater number of pattern clusters.

The “Pre-Process and Resize” block performs four actions. The first one is an image resizing to 19×19 pixels to after perform an image binary masking to remove background and to reduce the dimensionality of the window from 361 (19×19) to 283 (number of unmasked pixels). Immediately afterwards, an illumination gradient correction to substract the best-fit brightness plane is implemented, and finally, a histogram equalisation is applied.

The “Difference measurer” block computes, for each window at its input, a vector of 12 distances between the input window and the 12 centroids (6 of the face clusters and 6 of the non-face clusters). Each distance consists of two components. The first one is a normalised Mahalanobis distance between the given image and the corresponding centroid, calculated as

$$\mathcal{M}(\underline{x}, \underline{\mu}) = \frac{1}{2}\left(d \ln(2\pi) + \ln|\Sigma| + (\underline{x} - \underline{\mu})\Sigma^{-1}(\underline{x} - \underline{\mu})^{\top}\right) \tag{2.14}$$

where d is the space dimensionality, μ a cluster centroid, Σ the corresponding covariance matrix and $|\Sigma|$ its determinant. The second component is a normalised Euclidean distance between the input window and its projection in the lower-dimensional subspace spanned by the cluster's 75 (empirical value) largest eigenvectors.

Finally, the "Face/Non-Face Classifier" block, composed by a multilayer perceptron (MLP) classifier, decides wether a window corresponds to a face from the vector of 12 distances, composed each one of two components as commented before. The MLP network is composed of 12 pairs of input units, 24 hidden units and 1 output unit. For its training, the authors used a positive training set of 4150 face images and a negative training set dynamically collected, according to classification errors, from a total of 43166 nonface images.

K. Sung and T. Poggio tested their system over two datasets collected for the occasion. The first dataset, the "best case" dataset, consisted of 301 frontal and near-frontal face mugshots of 71 different people, all these images with high digitalising quality and high lighting variation. Over this group of images, a 96.3% of detected faces and 3 false detections were achieved. The other dataset, the "average case" dataset, consisted of 23 images with 149 faces, where a 79.9% of detected faces and 5 false positives were achieved.

2.2.2.2 Neural Network Approach by Rowley

This method, proposed in [64], was developed to detect upright, frontal views of faces in grayscale images. The system was designed divided in two levels, a neural network-based filter and an arbitrator.

The neural network-based filter is applied to regions of the main image of size 20×20 pixels and gives as output a single real number, ranging from 1 (presence of face) to -1 (absence of face). To be able to detect faces anywhere in the image and of any size, the filter is applied at any location and at different scaled images by a subsampling factor of 1.2. The different subsystems that compose this first level can be seen in figure (2.5). The preprocessing step is an adapted version of the one presented in [63] and its purpose is to equalise the intensity values across the 20×20 window in an oval region defined inside the given window. This oval region tries to fit the space occupied by a face in the window. The next step, the neural network, has 3 types of hidden units, as can be seen in figure (2.5). There are 4 hidden units that look at 10×10 pixels subregions, 16 that look at 5×5 pixels subregions and 6 that look at 20×5 pixels subregions, each one chosen to represent localised face features to make the detection. Although in figure (2.5) only one hidden unit is placed after each receptive field, these hidden units can be replicated to make the output more robust. The training process of the neural network is accomplished by standard error backpropagation algorithm over nearly 1050 real face images, with manually eyes and mouth alignment, and non-face images chosen from a set of $146,212,178$ total images, from where only false positive images are added to the training set.

On the other hand, the arbitrator tries to remove false detection of faces and to merge multiple detections of the same face at the output of the neural network-based filter. For the first purpose, since real faces are often detected at near

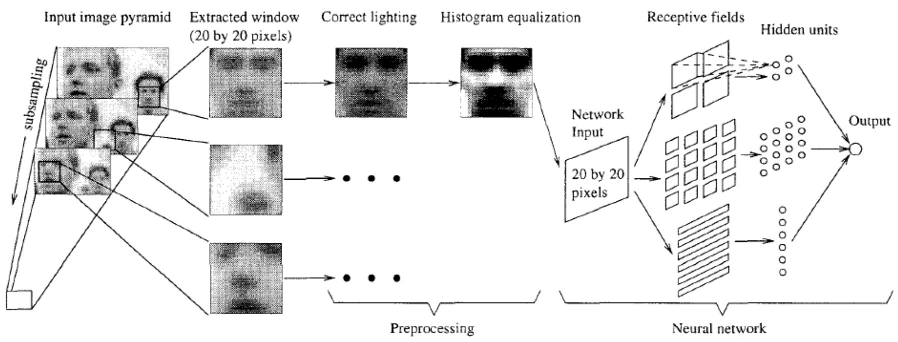


Figure 2.5: Filter Level of the algorithm proposed by Rowley et al. in [64]

positions and scales, a minimum threshold in the number of detections is fixed. For the second purpose, another heuristic that removes overlappings, preserving the detection with the highest confidence, is used. In addition, another technique can be used at this level by arbitrating the outputs of different neural networks trained with different random initial weights, making the system more robust.

The designed system does not have as output "Face" or "Non-Face" but a real number in the interval $[-1, 1]$. For this reason, the accuracy of the designed system depends on the fixed threshold. In figure (2.6), the ROC (*Receiver Operating Characteristic*) of the system over the three sets of images that the authors had at their disposal, with a total of 507 faces, can be seen.

A variation of this method was proposed by the same authors in [65] to detect frontal faces at any in-plane rotation value, by means of another neural network called a "router" that estimates the rotation value to correct the input pose of a given face.

2.2.2.3 Constrained Generative Model Approach by Féraud

A neural network model based on the *Constrained Generative Model* (CGM) to multiview (out-of-plane rotated) face detection was presented by Féraud et al. in [66]. This kind of neural networks uses counterexamples to increase the performance of the model and the training process tries to achieve the best model able to generate the input data. The authors justify its decision of using a generative model instead of a discriminant one due to the impossibility of collecting a representative set of non-faces.

The system is composed of 4 main blocks (see figure 2.7), a motion filter, a color filter, a MLP and the CGM. The first 3 blocks are intended to remove as much background as possible in as less time as possible. The main detector is the CGM, the most computational expensive block but the most accurate one. The aim of this block is to perform a non-linear PCA to model the distance of

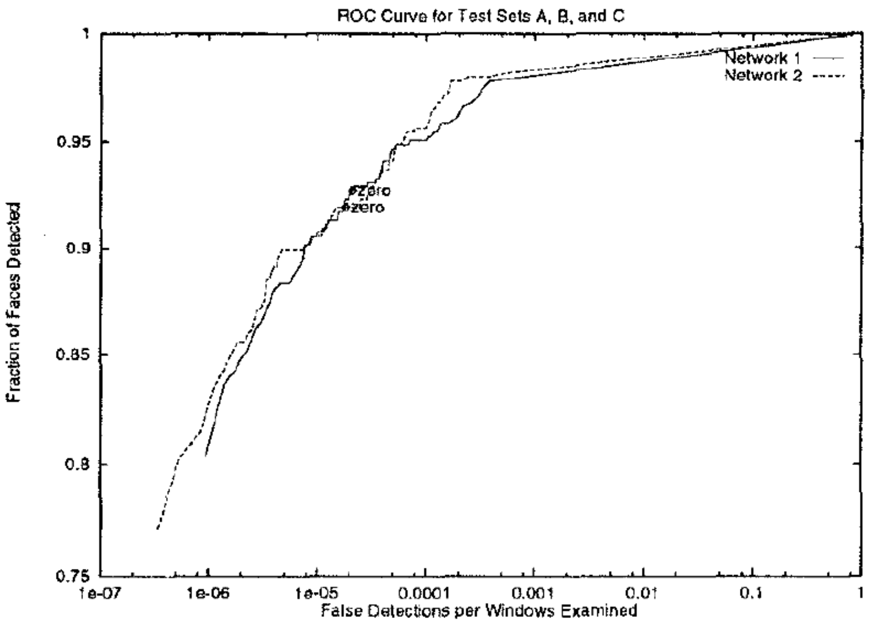


Figure 2.6: Detection rate against false positives of the system introduced by Rowley et al. in [64]

a given input $\underline{x}$ to the set of faces ν, defined as

$$D(\underline{x},\nu) = \|P(\underline{x}) - \underline{x}\| = \|\arg\min_{\underline{y}\in\nu}(d(\underline{x},\underline{y})) - \underline{x}\| \simeq \|P_{knn}(\underline{x}) - \underline{x}\| \tag{2.15}$$

where $P(\underline{x})$ is the projection of $\underline{x}$ over the face space, $d(\cdot,\cdot)$ is the euclidean distance and $P_{knn}(\underline{x})$ is a projection approximation. $P_{knn}(\underline{x})$ can be defined as

$$P_{knn}(\underline{x}) = \frac{1}{k}\sum_{i=1}^{k}\underline{v}_i \tag{2.16}$$

where $\underline{v}_1,\dots,\underline{v}_k$ are the k nearest neighbours of $\underline{x}$ in the training set of faces. The CGM computes the approximation of $D(\underline{x},\nu)$ described in 2.15 to evaluate the probability of being a face. This is achieved by minimizing

$$C_W = \sum_i (P_W(\underline{x}_i) - P_{knn}(\underline{x}_i))^2 \tag{2.17}$$

where W is the vectors of weights of the neural network and P_W designates the output of the neural network using W, i.e. the projection approximation.

The algorithms used to collect examples and counterexamples are explained in detail in [66]. To detect side view images and to decrease the number of false alarms, the authors used a conditional mixture of CGM's based on the "mixture of experts" introduced in [67]. Each CGM of the total system was trained using 2000 face images and around 2000 non-face images. A 87% of detected faces over CMU Test set with a false alarm rate of $1.15 \cdot 10^{-6}$ were achieved. Testing results over rotated faces can be seen in table (2.2).

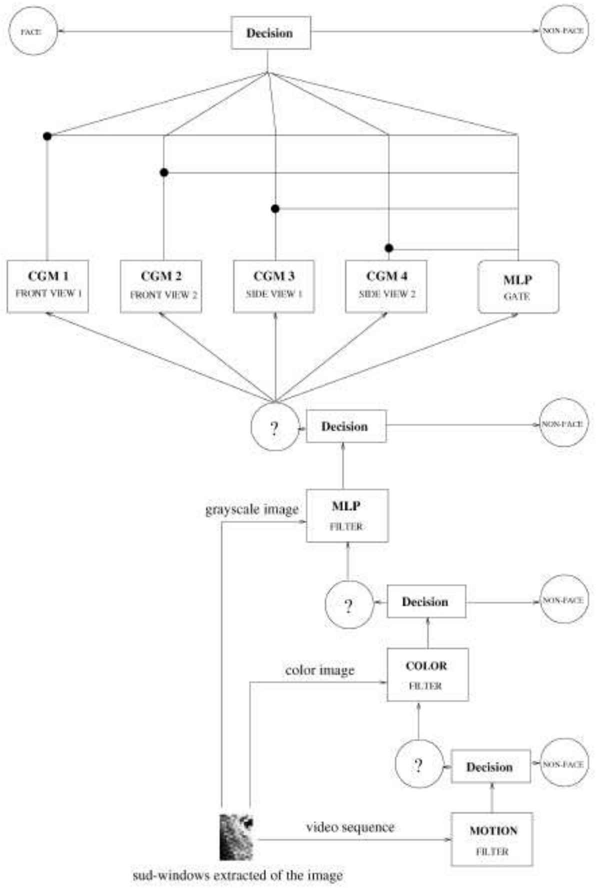


Figure 2.7: Face Detector proposed by Feraud et al. in [66]

Table 2.2: Test Results obtained by Féraud et al. in [66] over Sussex dataset

Out-of-Plane rotation (degrees)	Detection Rate
0	100%
10	100%
20	100%
30	100%
40	87.5%
50	62.5%
60	37.5%
70	25.0%

Table 2.3: Test Results obtained by Yang et al. in [68]

Method	**Test Set** 1		**Test Set** 2	
	Detect Rate	False Detects	Detect Rate	False Detects
SNoW with primitive features	94.2%	84	93.6%	3
SNoW with multi-scale features	94.8%	78	94.1%	3

2.2.2.4 SNoW Approach

Yang et al. introduced in [68] a frontal face detector system based on the SNoW (*Sparse Network of Winnows*) learning architecture. This architecture is a sparse network of linear functions over an incrementally learned feature space. This sort of neural networks is specifically intended to make the learning in the presence of a very large number of features.

A *Sparse Network of Winnows* is composed of linear units called *target nodes*. In the system presented by Yang et al. only 2 *target nodes* are used, one to represent face patterns and one to non-face patterns. This can be seen as a single SNoW unit with 2 subnetworks, each of them intended to classify one kind of patterns and reject the other one.

For training, the authors collected 1681 face images from Olivetti, UMIST, Harvard, Yale and FERET datasets. From each original face, 10 faces were generated by randomly rotating and scaling, obtaining 16810 face samples. For negative examples, an initial set of 8422 samples collected from 400 images was used. The training process is achieved by the *Winnow update rule* [69] using 2 kinds of boolean filters, one kind that encode position and intensity and the other one that encode position, intensity and mean and variance of a multi-scale pixel.

The system described above was tested over 2 sets of images of the MIT-CMU test set, using the common technique of scanning and scaling the image to search for faces in a given image. The "Test Set 1" consisted of 130 images with 507 frontal faces and the "Test Set 2" consisted of 23 images with 155 frontal faces. Some of the scored faces in both sets were hand drawn and cartoon faces. Over these sets, around a 94% of correct detections with a reduced number of false detection were achieved, as it can be seen in table (2.3).

2.2.2.5 Viola and Jones Approach

The method proposed by P. Viola and M. Jones in [1] was the first real-time frontal face detector system. This method introduces 3 main innovations, the use of the called "Integral Image", commonly used in computer graphics, the use of the AdaBoost algorithm, to build strong classifiers, and the use of a cascade

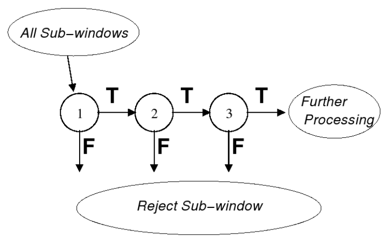


Figure 2.8: Schematic cascade of classifiers used by Viola and Jones in [1]. The second classifier is more accurate than the first one, the third more than the second and so on.

of classifiers to accelerate the erasure of background.

The system is composed of a cascade of classifiers (see image (2.8)) made up of strong classifiers built using the AdaBoost algorithm (see sections (4.2) and (3.2) for further details on cascades of classifiers and the AdaBoost algorithm respectively). As it has been said before, the authors chose Haar filters (see section (3.1.1)) to compute features due to its computation speed using the "Integral Image" representation (see section (3.1.1)). This representation allows to compute the response to any rectangular part of a Haar filter adding only 4 array elements, permiting the use of a high number of filters in each classifier without slowing down excessively the system.

The final classifier was formed by a cascade of 38 layers with a total of 6061 filters. The training was performed with a total of 9832 frontal face images (4916 and its vertical mirrored) and about 350 million of non-face images (but only a maximum of 10000 used at the same time), all of them of a size of 24×24 pixels. The final detector was able to process 384×288 pixel image in about 0.067 seconds in a Pentium III at 700MHz, using a scale factor of 1.25 and a step of 1.5 times the current scale. With this configuration, the authors obtained the ROC curve showed in figure (2.9) varying the final threshold from $-\infty$ to $+\infty$.

As an extension of this system to the multi-view face detection, the same authors proposed in [60] a system based in the one described above that was able to detect out-of-plane and in-plane rotated faces. Li et al. also introduced in [70] a system based in this one. In that work, the authors introduced a new learning algorithm called FloatBoost and new Haar-based filters to compute features.

2.2.2.6 IDIAP Multiview Face Detection System

The system introduced by Sauquet et al. in [5] is an in-plane and out-of-plane rotated faces detector (see section (1) for additional details on face rotations) and it is based on detector-pyramid (or cascade of classifiers) of Li et al. [70]. The system is able to detect faces with an out-of-plane rotation in the range of $[-90^o, +90^o]$ and an in-plane rotation in $[-67.5^o, +67.5^o]$, and it is composed of 2 classifiers, one for out-of-plane rotated faces and another one for in-plane rotated faces, both combined in a more general structure as can be seen in figure

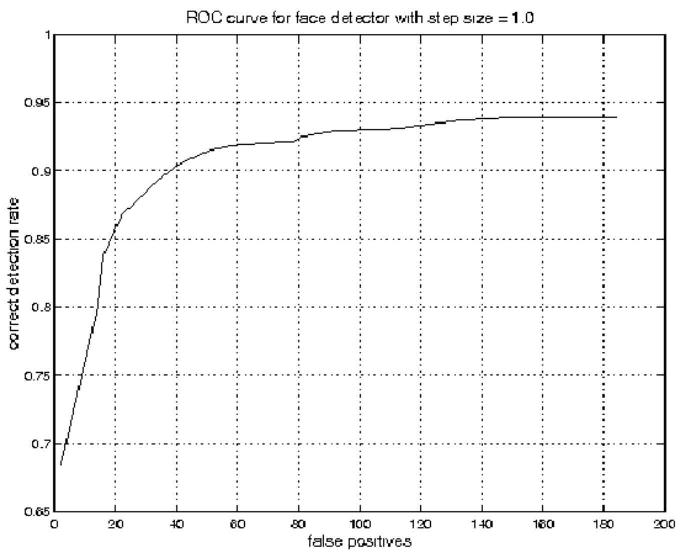


Figure 2.9: Obtained ROC curve by Viola et al. in [1]

(2.10).

The out-of-plane face detector is composed of 13 classifiers distributed on 3 levels and a fourth level of 9 *Multi-Layer Perceptrons*, and the in-plane detector is composed of 8 classifiers distributed on 2 levels and a third level of 3 *Multi-Layer Perceptrons*. *Multi-Layer Perceptrons* are used for postprocessing the potential face and decide definitely whether it should be classified as a face. After the detection process, a merging process is run to delete multiple detections. For further details on *Cascades of Classifiers* see section 4.2.

The training process, accomplished by the AdaBoost algorithm (see section 3.2) using *Local Structure Kernels* [71], was divided into two parts, the out-of-plane detector training and the in-plane detector training. The training of the out-of-plane detector was carried out with 8000 faces per pose, generated by mirroring and randomly scaling and translating 4700 faces collected from Feret, PIE and Prima Head Pose data sets. The in-plane detector was trained with 8744 faces per pose, generated from 5575 frontal faces from Feret, Stirling, Essex and Yale data sets. The non-face set was formed by about 800 million images.

Several tests and comparisons with the “state of the art” detectors are showed in [5]. A detection rate around a 92% of correct detections is achieved by the proposed multiview face detector, with the advantage that the detection is accomplished in real-time. With regard to the pose estimation, depending on the pose it is achieved between a 40% and a 100% of correct estimations.

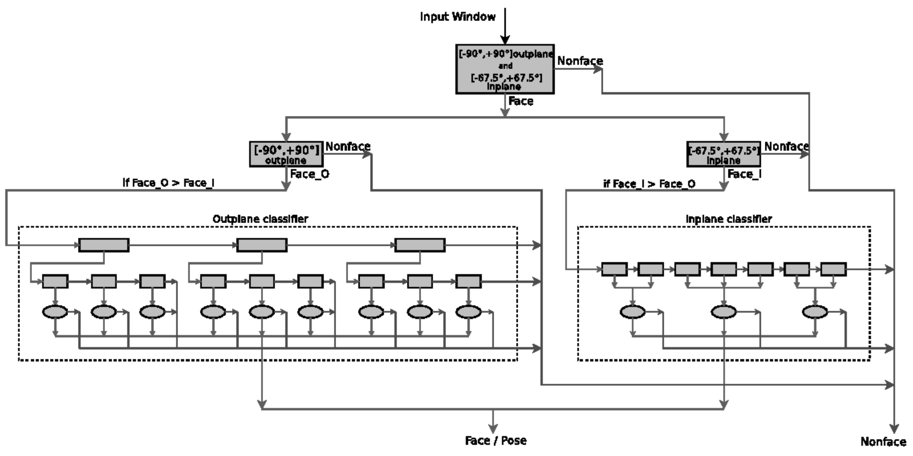


Figure 2.10: Pyramid of classifiers proposed by Sauquet et al. in [5]

2.2.2.7 Boosting Approach by Meynet

Meynet et al. introduced in [62] a frontal face detector based on the proposed one by Viola et al. in [1] using the AdaBoost algorithm. Two important improvements were introduced in this work: the use of *Anisotropic Gaussian Filters* to compute features, in complement of Haar-like filters, and a computation scheme using a mixture of classifiers to improve the detection accuracy. For a detailed description of *Anisotropic Gaussian Filters* see section 3.1.2.

The mixture of classifiers structure consists of a multi-classifier built from several classifiers in parallel, each one trained with different sets of images. This technique allows to decrease the influence of outliers in the training set, since the training sets are separated, and to decrease the complexity of the training process, since the sets of images has been splitted in several subsets. Furthermore, if some of the parallel classifiers fails in its decision, it does not mean that the final decision will be incorrect, since the final output is a combination of the outputs of each parallel classifier. This is a great advantage in front of a cascade structure because on these, if some cascade layer fails in its decision, deciding for example that a face is not a face, this error will not be recovered.

The final system consisted in a preprocessing cascade of Haar boosted classifiers to remove quickly easy non-face images and 5 parallel mixtures of about 200 classifiers. Each mixture was trained with 1900 faces and 4000 non-face's. This system was tested over the BANCA database, obtaining a very high detection rate of about a 96%, and over the CMU/MIT test set, which obtained ROC can be seen in figure (2.11).

2.2.3 Statistical Methods

Besides of *Linear Subspace* methods and *Machine Learning* methods there are also pure statistical methods based on information theory. These methods use

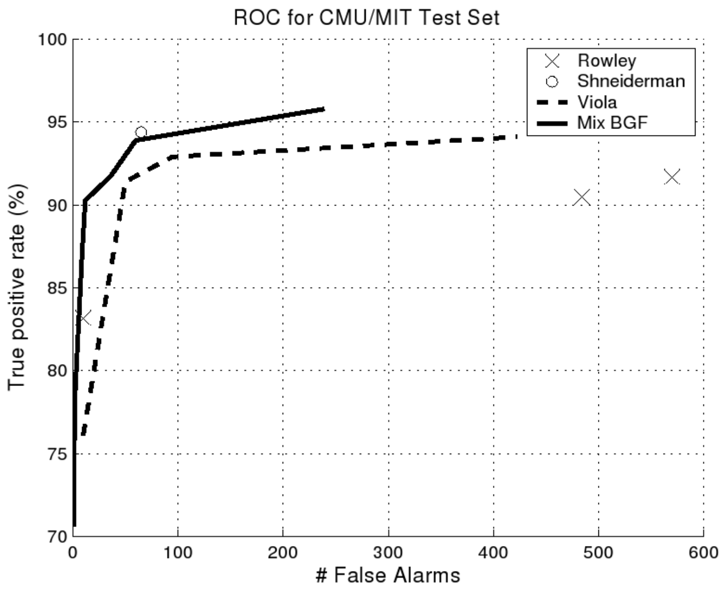


Figure 2.11: ROC obtained by Meynet et al. in [62] compared with other approaches over the CMU/MIT test set

tools such as *likelihood* [72, 73, 74, 75] or the *Bayes' decision rule* [76, 77].

2.2.3.1 Maximum Likelihood Approach by Colmenarez and Huang

Colmenarez and Huang introduced in [74] a face detector based on an earlier work on maximum likelihood by the same authors [72]. The system is based on Kullback relative information, a nonnegative measure of the difference between two density functions.

The aim of the algorithm designed by Colmenarez and Huang is to classify a given image part into a face class or into a non-face class. This is achieved by means of computing the likelihood ratio using probability models obtained during a learning process. To make this, the authors make use of Kullback relative information, also known as Kullback divergence

$$H_{P\|M} = \sum_{X^n} P_{X^n} \ln \frac{P_{X^n}}{M_{X^n}} \tag{2.18}$$

where X^n is a random process, P_{X^n} and M_{X^n} are two probability functions for X^n and $H_{P\|M}$ is the divergence of P with respect to M. If X^n is supposed to be a finite alphabet, stationary, kth order Markov process, the Kullback divergence can be obtained as

$$\begin{aligned} H_{P\|M}(X^n) = & \textstyle\sum_{i=1}^{k} H_{P\|M}(X_i \| X_1, \ldots, X_{i-1}) + \\ & + \textstyle\sum_{i=k+1}^{n} H_{P\|M}(X_i \| X_{i-k}, \ldots, X_{i-1}) \end{aligned} \tag{2.19}$$

Under these premises, given a random process X^n and given $S = \{s_i \in [1, n], i = 1, \ldots, n\}$ such that $s_i \neq s_j \Leftrightarrow i \neq j$, the kth order Markov process $X^n(S)$ from

X^n can be constructed by re-ordering its variables so that

$$P(X_{s_n}|X_{s_1}, \ldots, X_{s_{n-1}}) = P(X_{s_n}|X_{s_{n-k}}, \ldots, X_{s_{n-1}}) \tag{2.20}$$

then the objective is to find a S^* such that

$$H_{P\|M}(X^n(S^*)) \geq H_{P\|M}(X^n(S)) \quad \forall S \tag{2.21}$$

All this mathematical development can be applied to the face detection problem by considering an image as a realization of a random process, where each pixel is a value of this realization, and the aim is to be able to calculate, once calculated S^* from equation 2.21, the likelihood ratio of a given observation O^n as

$$L(O^n) = L(O_{s_1}) + \sum_{i=2}^{n} L(O_{s_i}\|O_{s_{i-1}}) \tag{2.22}$$

To construct the probability models, the system was trained with frontal-view images from FERET dataset and a collection of non-face images collected by the authors. The training process requantize the training images to 4 grey levels and calculates a probability model with the required parameters to compute equation (2.22). Samples are classified by selecting the position where the likelihood ratio is higher than a fixed threshold.

The system was evaluated over the CMU dataset (130 images containing 507 faces) with the typical scan and scale procedure to search faces in images. The authors considered as good detections those that the face was detected at the correct scale and the error in the position of the face was less than 10% of the total size of the face. Taking this into account, the system performs a detection rate between 87% and 98%, but with a high false alarm rate with respect to other techniques.

An evolution of this system using error bootstrapping was proposed by the same authors in [75] and a similar system, developed by Lew et al., can be found in [73].

2.2.3.2 Maximum a Posteriori Approach by Schneiderman and Kanade

Henry Schneiderman and Takeo Kanade introduced in [76] an algorithm for object recognition based on the *Maximum a Posteriori* probability (MAP), known also as *Bayes' Decision Rule*. They applied this algorithm to the frontal and profile face detection problem obtaining good results.

The *Bayes' Decision Rule* applied to decide whether an image X represents an object O, can be expressed as

$$P(O|X) \begin{cases} > P(\overline{O}|X) \Rightarrow \text{ Object} \\ < P(\overline{O}|X) \Rightarrow \text{ Not Object} \end{cases} \tag{2.23}$$

where $P(\overline{O}|X) = 1 - P(O|X)$ and $\overline{O}$ means "Not Object". Using this rule to make decisions, optimal performance is achieved in the sense of minimum rate of misclassification when the posterior probability function is accurate. This produces a problem since, in general, it is not feasible to completely represent $P(O|X)$. For this reason, the authors proposed 13 simplifications, like to consider a fixed image size or the decomposition into subregions, to be done to $P(O|X)$ to allow a feasible and sufficiently accurate representation. After this 13 simplifications, equation (2.23) leads to

$$\prod_{j=1}^{n_{\text{magn}}} \prod_{i=1}^{n_{\text{subs}}} \frac{P(q1^j|O)P(pos^j|q2^i, O)}{\frac{P(q1_1^j|\overline{O})}{n_{\text{subs}}}} \begin{cases} > \frac{P(\overline{O})}{P(O)} \Rightarrow \text{ Object} \\ < \frac{P(\overline{O})}{P(O)} \Rightarrow \text{ Not Object} \end{cases} \tag{2.24}$$

where n_{subs} is the number of subregions, n_{magn} the number of scales of resolution, $q1$ and $q2$ are quantisations of X and *pos* the positional distribution of objects.

The training set used by the authors was formed by $118,920$ face images (generated by random variation in orientation, size, intensity... from 991 original face images) and 1552 non-face images. With these images, the likelihood function was estimated to test the face detection system on the test sets used by Sung et al. in [63] and by Rowley et al in [64] and on 3 portions of the FERET dataset (0^o, 15^o and 22.5^o), obtaining a detection rate of about a 90% with a reduced number of false alarms.

Another approach proposed by the same authors, also using *Bayes' Decision Rule*, can be found in [77].

2.3 Feature-based vs Image-based

Each one of the two exposed families, *Feature-based* methods and *Image-based* methods, has its own pros and cons. *Feature-based* methods that use information provided by colour or motion are in general very fast, and can work in real-time without problem. On the contrary, *Image-based* methods, until the appearance of Viola and Jones approach, were usually not able to work in real-time. Furthermore, *Image-based* methods need a training process during which a huge number of images are required, and *Feature-based* approaches usually do not need such a process since the facial knowledge is implicitly expressed inside the algorithm.

However, *Image-based* methods are robust against changes in facial expression or in lighting conditions, and they can perform well with low-resolution grey-level images, whereas *Feature-based* methods need in general high-resolution images and they are sensitive to changes in facial expression or in lighting conditions. In addition, *Image-based* methods are robust to little changes in facial pose and, usually, *Image-based* face detectors can be generalised, with more or less effort, to be a multi-view face detector, whereas *Feature-based* methods are very sensible to facial pose and in general the face knowledge that the method is exploiting does not have to be possible to generalise to make the algorithm able to detect multiple facial poses.

Due to these pros and cons in each family, it is very common to have hybrid techniques that try to exploit the best points of each family of methods.

The aim of the present project is to build a multi-view real-time facial detector, able to work with low-resolution images as for example those provided by a webcam or a surveillance camera. For this reason, the presented algorithm has been chosen to be situated in the *Image-based* family, specifically in the *Machine Learning* subfamily, since it can be seen as an evolution of Viola and Jones approach [1].

Part II

Theoretical Background and Implementation

Chapter 3

Theoretical Aspects

As it has been commented in the introduction, the aim of this work is to build a multi-view face detector able to work in real-time. With this purpose, an intensive use of the AdaBoost algorithm will be done. The following section are organized as follows. In section 3.1, the filters that will be used to extract features from the greyscale input images are described. Next, in section 3.2, the AdaBoost algorithm is explained. Finally, in section 3.3 an introduction to decision trees is shown, since this kind of structures will be used in order to combine the different classifiers to estimate the face pose.

3.1 Filters

The face detector system, that will be developed in this project, needs to deal with images to classify them into 2 classes, the "Face" class and the "Non-Face" class. To accomplish this, the system could manage all the pixels in an image, but this would not be a robust approach, due to noise, light changes... An approach more robust consists on extracting simple and local geometrical features from the image using filters, since filters are more discriminant than simple pixels. Then, a decision to classify this image based on those features can be taken. Therefore, filters are needed.

A filter in this context can be seen as an image mask F that, applied to an image I, produces a response O as output, calculated as

$$O = \sum_{r=1}^{\#Rows} \sum_{c=1}^{\#Cols} I(r,c)F(r,c) \tag{3.1}$$

where $\#Rows$ and $\#Cols$ are the number of rows and columns, respectively, of both image and filter. The computed outputs, known as *Features*, will allow to discriminate between faces and non-faces, as well as between faces with different poses.

Two families of filters will be used in the present work, *Haar-based* filters and *Anisotropic Gaussian* filters.

3.1.1 Haar-based Filters

Haar-based filters are filters based on the Haar function

$$\psi(x) = \begin{cases} 1, & 0 \leq x < \frac{1}{2} \\ -1, & \frac{1}{2} \leq x < 1 \\ 0, & \text{otherwise} \end{cases} \tag{3.2}$$

where $x \in \mathbb{R}$. Generalising this function to a function in $\mathbb{N}^2$, where each coordinate corresponds to an image pixel, 5 different types of *Haar-based* filters are obtained as it can be seen in figure (3.1). Each filter consists of 2, 3 or 4 blocks of $W_{\text{pix}} \times H_{\text{pix}}$ pixels, with values of 1, -1 or -2 in its pixels and 0 outside, so that the filter has a null mean. By scaling these filters to different sizes and translating them to different places, $69,790$ filters are obtained for a window of 20×20 pixels, $21,000$ of types I and II, $13,230$ of types III and IV and $1,330$ of type V:

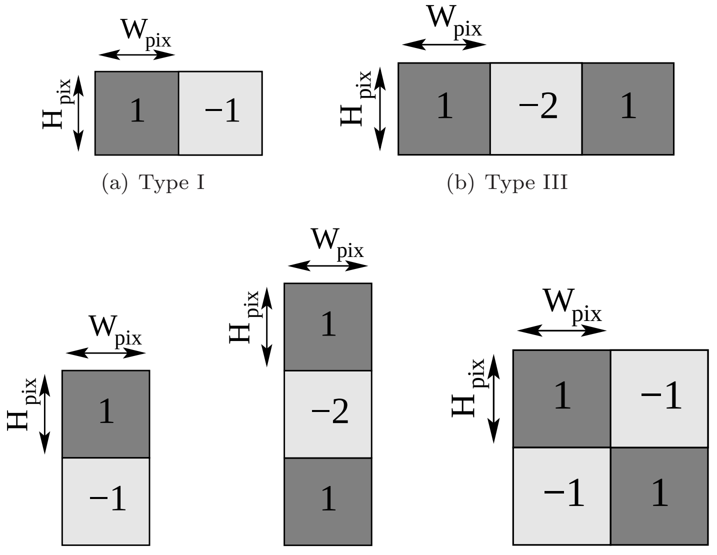
$$\text{Type I:} \quad \textstyle\sum_{W=2,4,6,\ldots,20} \left[\left(\sum_{i=1}^{20} i\right)(20-(W-1))\right] = 21.000 \tag{3.3}$$

$$\text{Type II:} \quad \textstyle\sum_{H=2,4,6,\ldots,20} \left[\left(\sum_{i=1}^{20} i\right)(20-(H-1))\right] = 21.000 \tag{3.4}$$

$$\text{Type III:} \quad \textstyle\sum_{W=3,6,9,\ldots,18} \left[\left(\sum_{i=1}^{20} i\right)(20-(W-1))\right] = 13.230 \tag{3.5}$$

$$\text{Type IV:} \quad \textstyle\sum_{H=3,6,9,\ldots,18} \left[\left(\sum_{i=1}^{20} i\right)(20-(H-1))\right] = 13.230 \tag{3.6}$$

$$\text{Type V:} \quad \textstyle\sum_{W=H=2,4,6,\ldots,20} ((20-(W-1))(20-(H-1))) = 1.330 \tag{3.7}$$



(a) Type I (b) Type III

(c) Type II (d) Type IV (e) Type V

Figure 3.1: Types of Haar filters

where W is the filter width and H is the filter height.

Computed features with *Haar-based* filters are not very selective, but they present a very important advantage. It is not necessary to compute expression (3.1) to obtain its output, since by means of the image representation known as *Integral Image*, the output to a block of any size can be obtained by only 4 additions. In the *Integral Image* representation, each pixel contains the sum of that pixel with pixels above and to the left of it, i.e., given an image $I(r, c)$, its *Integral Image* representation $II(r, c)$ is given by

$$II(r, c) = \sum_{\tilde{r} \leq r,\ \tilde{c} \leq c} I(\tilde{r}, \tilde{c}) \tag{3.8}$$

In addition, the *Integral Image* can be computed in only one pass over the original image using the following recurrences

$$S(r, c) = S(r, c-1) + I(r, c) \tag{3.9}$$
$$II(r, c) = II(r-1, c) + S(r, c) \tag{3.10}$$

where $S(r, c)$ is the cumulative row sum, $S(r, -1) = 0\ \forall r$ and $II(-1, c) = 0\ \forall c$.

By means of the *Integral Image*, the sum of all the pixels under a block, as mentioned above, can be computed in only 4 additions, as indicated in figure (3.2), where the sum of all the pixels under the block S can be computed as $A + D - B - C$, where A, B, C and D represent the value of the *Integral Image* in those pixels. This makes that the computation of the output to a *Haar-based*

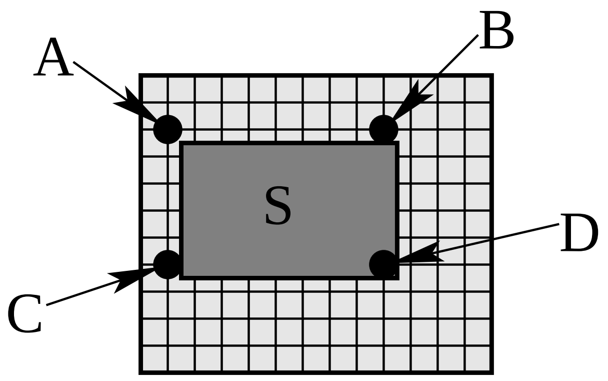


Figure 3.2: Important points in the computation of the sum of pixel values under the block S using the *Integral Image*

filter can be computed in 6 additions for types I and II, in 8 additions for types III and IV and in 9 additions for type V, since all the blocks are adjacent.

This kind of filters produces outputs that are not very selective, making them not suitable to be utilised to discriminate between poses. But they are very useful to discriminate between faces and non-faces, allowing to remove a lot of background very quickly.

3.1.2 Anisotropic Gaussian Filters

Anisotropic Gaussian filters, first introduced by Peotta et al. in [78], consist of the combination of a *Gaussian* function in one direction with its first derivative in the other one, what will be called the generative function, combined as well with 4 basic operations: *translation*, *rotation*, *bending* and *anisotropic scaling*. The resulting filters are directionally dependent, as the word "Anisotropic" indicates, and they are more selective than *Haar-based* ones, being able to catch face contours and shapes (see figure (3.3) for a test error rate comparison between using the AdaBoost algorithm with Haar-based filters and with Anisotropic Gaussian filters).

The generative function used to generate the *Anisotropic Gaussian* filters set is

$$g(x,y) = xe^{-(|x|+y^2)} \tag{3.11}$$

where x-direction, called singular-direction, has a Laplacian shape with odd-symmetry, and y-direction, called contour-direction, is a *Gaussian* function.

The 4 basic operations are defined as

- Translation
$$\mathcal{T}_{x_0,y_0} g(x,y) = g(x-x_0, y-y_0) \tag{3.12}$$
- Rotation
$$\mathcal{R}_\theta g(x,y) = g(x\cos\theta - y\sin\theta, x\sin\theta + y\cos\theta) \tag{3.13}$$

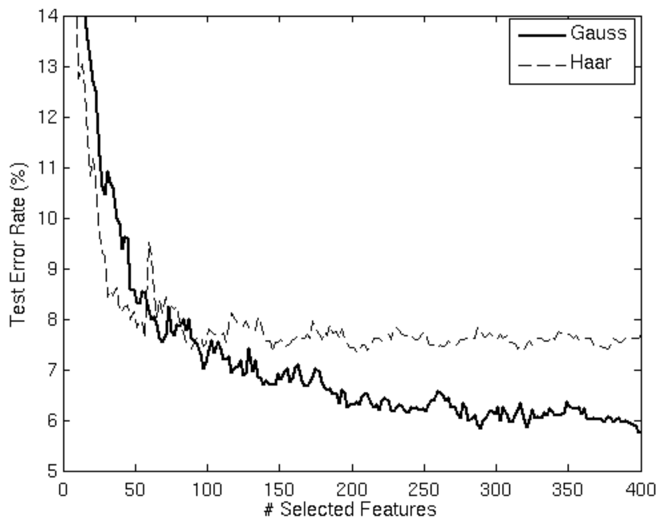


Figure 3.3: AdaBoost test error rate (see section (3.2.1)) comparison between Haar-based filters and Anisotropic Gaussian filters

- Bending

$$\mathcal{B}_\rho g(x,y) = \begin{cases} g\left(\rho - \sqrt{(x-\rho)^2+y^2}, \rho \arctan\left(\frac{y}{\rho-x}\right)\right) & \text{if } x < \rho \\ g\left(\rho - |y|, x - \rho + \rho\frac{\pi}{2}\right) & \text{if } x \geq \rho \end{cases} \tag{3.14}$$

- Anisotropic scaling

$$\mathcal{S}_{s_x,s_y} g(x,y) = g(\frac{x}{s_x}, \frac{y}{s_y}) \tag{3.15}$$

Combining the generative function with this 4 operations, a family $\mathcal{F}_g$ of filters is obtained

$$\mathcal{F}_g = \left\{\mathcal{T}_{x_0,y_0}\mathcal{R}_\theta\mathcal{B}_\rho\mathcal{S}_{s_x,s_y} g(x,y) |\ (x_0,y_0) \in \mathbb{R}^2,\ \theta \in [0,2\pi),\ \rho, s_x, s_y \in \mathbb{R}^+\right\} \tag{3.16}$$

In this case, the total number of filters for a window of 20×20 pixels is harder to calculate. In theory it is infinite, but since they must be put in an image with a finite number of pixels, then the set of possible filters become finite. Anyway, this number is too huge to be used, what forces to use a sampling of the parameter domain of $\mathcal{F}_g$. Some examples of *Anisotropic Gaussian* filters can be seen in figure (3.4).

Unlike the *Haar-based* filters, *Anisotropic Gaussian* filters can select very accurately contours and shapes, as it has been commented before, by varying their parameters. For this reason, computed features using these filters can be very selective, allowing to discriminate between faces with different poses. As disadvantage, these filters do not allow an efficient computation as *Haar-base* ones,

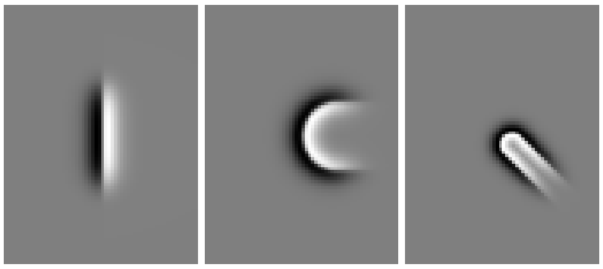


Figure 3.4: Three *Anisotropic Gaussian* filters with different rotating and bending parameters

and equation 3.1 must be utilised to compute their outputs. But another advantage is that equation 3.11 does not have to be computed each time, since the response to the unity image, $\mathbb{1}_{20\times 20}$ can be stored to be inserted in equation 3.1.

3.2 AdaBoost Algorithm

In a pattern recognition problem, as the case that is treated in this project, some learning technique must be used in order to classify data. Here, a *boosting* algorithm known as *AdaBoost* will be used to build classifiers that, by means of extracting features from images, will be capable of distinguishing between "Face" and "Non Faces" patterns.

Boosting is a deterministic and sequential technique that allows to produce an accurate prediction rule by combining inaccurate rules. The *AdaBoost* algorithm, first introduced by Yoav Freund and Robert E. Schapire [79] in 1995, is a *boosting* algorithm that gets adjusted adaptively to the errors of the inaccurate rules or weak hypothesis h_t. This adaptive adjustment is the principal difference and advantage of *AdaBoost* against other *boosting* algorithms. The authors gave in their article generalisations to the N-Class problem but they focused on the 2-Class problem. The same will be done here since it is the most interesting case for our purpose, to distinguish between "Face" and "Non-Face".

Before giving the explicit *AdaBoost* algorithm description, some definitions must be done:

- X is the domain space to which the samples x_i belong. In our case, as we work with 20×20 pixel images, $X = \mathbb{R}^{400}$.
- Y represents the label set, that will be assumed to be $\{-1, +1\}$, and $y_i \in Y$ is the label belonging to the sample x_i.
- D_t is the probability distribution over the training set $\{(x_i, y_i)|i = 1, \dots, N\}$ in the t-th iteration of the *AdaBoost* algorithm. D_t^i represents the weight of this distribution on (x_i, y_i).
- $\mathcal{H}$ is the set of filters used to compute features.

- h_t is the weak hypothesis or weak classifier chosen in iteration t, and H_T is the final hypothesis, also known as hard classifier, after T iterations of the *AdaBoost* algorithm, which is a linear combination of $h_1, \ldots, h_T$.

- ϵ_j is the training error due to the use of h_j to classify the samples of the training set given a fixed probability distribution.

- And finally, α_t is a parameter that measures the importance assigned to h_t.

With the above definitions, the *AdaBoost* algorithm, stated in algorithm (2), can be easily understood. The most important point of this algorithm, which is also one of the most important characteristics of *AdaBoost*, as it has been commented before, is its adaptive adjustment. This adaptive adjustment can be seen in steps 4 and 5 of algorithm (2), where it is easy to see that hard samples, those that are difficult to classify, increase their weight in D_t, while the easy ones decrease their weight.

Algorithm 2: AdaBoost Algorithm for the Two-Class problem $\{-1,+1\}$

```
  Data: $T,\ \{(x_1,y_1),\ldots,(x_N,y_N) \mid x_i \in X,\ y_i \in Y = \{-1,1\}\}$
1 Initialise probability distribution $D_1^i = \frac{1}{N},\ i = 1,\ldots,N$
2 for $t = 1$ to $T$ do
3     Calculate $h_t$ as
          $$h_t = \arg\min_{h_j \in \mathcal{H}} \epsilon_j = \arg\min_{h_j \in \mathcal{H}} \left( \sum_{i \mid y_i \neq h_j(x_i)} D_t^i \right)$$
4     Calculate $\alpha_t$ as
          $$\alpha_t = \frac{1}{2} \ln \left( \frac{1-\epsilon_t}{\epsilon_t} \right)$$
5     Update distribution
          $$D_{t+1}^i = \frac{e^{-\alpha_t h_t(x_i) y_i}}{\sum_{j=1}^N D_{t+1}^j} D_t^i,\ i = 1,\ldots,N$$
6 end
  Result:
          $$H_T(x) = \text{sign}\left( \sum_{t=1}^T \alpha_t h_t(x) \right)$$
```

The output of the *AdaBoost* algorithm is the hard classifier H_T. This hard classifier is the expression that computed for an input x allows to classify it in one of the two possible classes $\{-1,+1\}$ (or “Face” and “Non-Face” in our case).

3.2.1 Error analysis

Two different errors can be considered, *training error* and *generalisation error*. The *training error* is the error made classifying the training set, and the *generalisation error* is the error of the final hypothesis outside the training set.

The *training error* for a given iteration t, ϵ_t, is defined in step 3 of algorithm (2). Then, after the T iterations, the *training error* of the final hypothesis $H_T(x)$, ϵ, can be bounded by means of $\epsilon_1, \ldots, \epsilon_T$ as

$$\epsilon \leq \prod_{t=1}^{T} 2\sqrt{\epsilon_t(1-\epsilon_t)} \tag{3.17}$$

This result was demonstrated by Freund and Schapire in [79]. This bound is interesting because, contrary to other boosting algorithms, the bound depends on all the errors of each weak classifier that is used, and not only on the maximal error. This is an advantage in practical applications because the *training error* is expected to get lower while T grows if $\epsilon_t < 1/2\ \forall t$. This last condition means that *weak classifiers* must be better than random guessing.

Equation (3.17) guarantees that the error of the final hypothesis $H_T(x)$ is small over the training set, but what is wanted to be small is *generalisation error* ϵ_g, i.e. the error over the whole space X. The *generalisation error* can be expressed as

$$\epsilon_g = P_{(x,y)\sim\mathcal{P}}(H_T(x) \neq y) \tag{3.18}$$

This expression depends basically on the family of weak classifiers $\mathcal{H}$ and on the number of iterations T. The choice of $\mathcal{H}$ depends on the learning problem and our knowledge about it, but the choice of T can be done following some different methods, basically *structural risk minimisation* or *cross-validation*. The *structural risk minimisation* consists of finding an upper bound of the VC-dimension (see appendix A) of the concept class, which allows to compute an upper bound of the *generalisation error* of $H_T(x)$ for all T. With this method, obtained bounds are usually rough and consequently, the calculated T is usually larger than the needed one. For this reason, *cross-validation* is usually more useful. *Cross-validation* consist of keeping a fraction of the training set that instead of being used during the training, it is used after, as a “validation” set to estimate the *generalisation error* and halt the iterative process when an optimal is obtained.

In addition to the errors, there is also another aspect to take into account, the *overfitting* problem. The *overfitting* problem is a typical problem in statistics, when a model with too many parameters, with respect to the process complexity, is tried to be fitted to a process. In that case, can be thought that a perfect model, that fits exactly the process, has been found when actually what has been obtained is a false model that model an unreal process fitting exactly the training set. The *overfitting* problem is usually seen as a violation of Ockham’s razor principle,

> “Pluralitas non est ponenda sine neccesitate”

plurality shouldn't be posited without necessity.

This *overfitting* problem is also very important in machine learning, as it is the case of *boosting* algorithms, where techniques as *cross-validation* or *early-stopping* must be used to avoid *overfitting*. After this introduction to the *overfitting* problem and the structure of the *AdaBoost* algorithm, it could seem that some of the commented techniques should be used in order to stop the *AdaBoost* iterations before the appearance of the *overfitting*. But this is not really necessary since another of the interesting properties of *AdaBoost* is that in absence or presence of moderate noise in the training set, *AdaBoost* tends not to *overfit*.

3.2.2 Comparison with Support Vector Machines

The *boosting* techniques, and more precisely the *AdaBoost* algorithm, have a strong connection with SVMs (Support Vector Machines), but at the same time they present very different characteristics. In order to see the similarities, it is needed to write the expression used to maximise the minimum margin

$$\max_{\underline{\alpha}} \min_{i} \frac{y_i(\underline{\alpha} \cdot \underline{h}(x_i))}{||\underline{\alpha}|| ||\underline{h}(x_i)||} \tag{3.19}$$

where $\underline{\alpha} = (\alpha_1, \ldots, \alpha_N)$, $\underline{h}(x) = (h_1(x), \ldots, h_N(x))$.

Comparing *AdaBoost* with SVMs by means of equation (3.19) it can be seen that the differences between both methods are differences in the used norms. For boosting, the norms in the denominator are the 1-norm in $\underline{\alpha}$ and the ∞-norm in $\underline{h}$

$$||\underline{\alpha}||_1 = \sum_t |\alpha_t| \tag{3.20}$$

$$||\underline{h}(x)||_\infty = max_t |h_t(x)| \tag{3.21}$$

Note that, when $h_i(x) \in \{-1, +1\}$, then $||\underline{h}(x)||_\infty = 1$.

In SVMs, both norms are Euclidean (2-norm)

$$||\underline{\alpha}||_2 = \sqrt{\sum_t \alpha_t^2} \tag{3.22}$$

$$||\underline{h}(x)||_2 = \sqrt{\sum_t h_t(x)^2} \tag{3.23}$$

Then, looking only at expression (3.19), the differences between both methods seem to be very small, only differences in the computation of some norms. But, as commented in [80], there are some important differences:

- At the high dimensions that the *AdaBoost* algorithm and SVMs are usually used, the differences between the norms l_1, l_2 and l_∞ can be very significant.

- SVMs require *quadratic programming* while *AdaBoost* only requires *linear programming.*
- SVMs deal with the problem of operating in a very high dimensional space using the method of *kernels*, while *AdaBoost* employs *greedy search*, where the oracle role is played by the weak learner.

3.3 Decision Trees

A *decision tree* is basically an array of tests intended to make decisions about a topic. Each node performs a test over the data input and according to the result of this test, some decisions are adopted and maybe some consequences predicted. *Decision trees* have 4 types of structures inside:

- The ***Root*** node is the node situated on the top of the tree. All the data that comes into the tree goes through it.
- ***Non-Leaf*** nodes are nodes situated on the middle stages of the tree. They are connected to nodes before and after them.
- ***Leaf*** nodes are nodes situated on the bottom of the tree. They do not have more nodes after them.
- ***Branches*** connect nodes and bring data from one node to another one.

The purpose of each node, independently of its type (root, non-leaf or leaf), is to perform a test over the data that it receives and route the data accordingly to the results of that test. Then, when some data is input into a *decision tree*, it is classified by sorting it down the tree, beginning in the *root* node and ending in some leaf node.

For example, let's imagine that somebody wants to go to the beach but he is unable to decide whether he should go. A simple example of a *Decision Tree* intended to help that person to make a decision can be seen in figure 3.5.

As it will be seen in section 4.3, the *decision tree* used in this project to classify faces according to their pose has one peculiarity. When one sample is tested in any of the nodes of the tree, if the test returns "Face" then the sample is routed to all the nodes in the next stage that are connected to it, but if the test returns "Non-Face" the sample is routed to the "Non-Face" special node that does not perform any test, only discards samples as face candidates, taking them out of the tree.

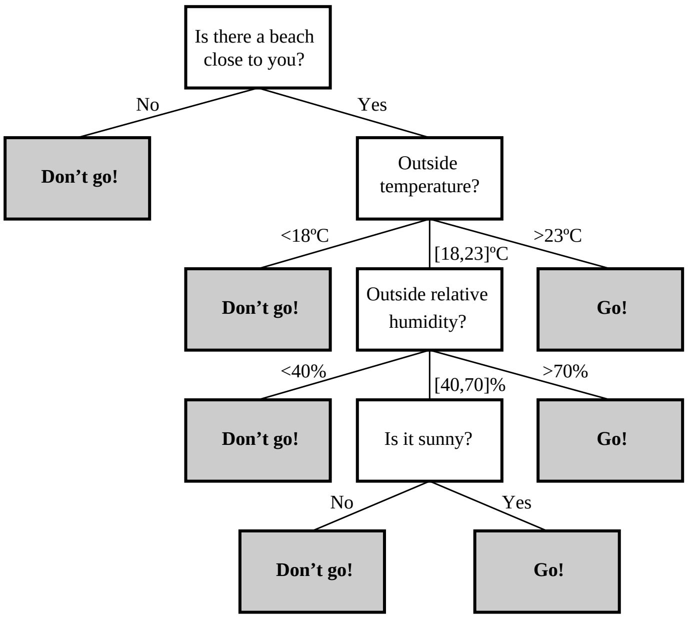


Figure 3.5: Example of a decision tree

Chapter 4

Implementation of the System

In the previous chapter, all the theoretical aspects that will be used to build the multi-view facial detector have been described exhaustively. In this chapter, the practical implementation of the system will be explained. In order to accomplish the objective of a multi-view facial detector, several classifiers must be trained and their outputs combined. For this reason this chapter is organized as follows. The chapter begins with section 4.1, where a description of the set of images that will be used to train the classifiers is given. Next, is section 4.2 the concrete structure of classifiers will be described, as well as the tree structure where the different classifiers will be put in order to combine their outputs in section 4.3. Finally, a description of how the system for experimental results works is given in section 4.4.

4.1 Data Sets

One of the most important things at the time of implementing the theory aspects exposed in the previous chapter is to have a good training set to build the face classifiers. This training set fixes a very important parameter of the final system, it limits its accuracy, since it is not possible to have more accuracy than the resolution of the training set. In addition, many examples are needed, since "Face" class has a large intra-class variance and "Non-Face" class is extremely large. In the next sections, the databases used to form the training set will be described, as well as the training set in itself.

4.1.1 INRIALPES Head Pose Database

The INRIALPES Head Pose database recorded by Nicolas Gourier, was recorded in order to test the algorithms developed in [81]. It consists of 15 sets of images from 15 people, each one containing 2 series of 93 images, with some of the series from people wearing glasses. Each one of the 93 images contained in a series has a different pose, considering as pose the out-of-plane rotation and the elevation:

- Out of plane rotation values in degrees: −90, −75, −60, −45, −30, −15, +0, +15, +30, +45, +60, +75 and +90.
- Elevation values in degrees: −90, −60, −30, −15, 0, +15, +30, +60 and +90.

For each elevation value, all the possible out-of-plane rotations are recorded, excepting for the values of elevation +90 and −90, where only the value of out of plane rotation equal to 0 degrees is recorded. All image files are stored in JPEG format following the next file name format:

`personne[PersonID][Series][Number][Elevation][OOPRotation].jpg`

- PersonID $\in \{01, 02, \ldots, 15\}$ indicates the number of the person recorded.
- Series $\in \{1, 2\}$ indicates the series number.
- Number $\in \{00, 01, \ldots, 92\}$ indicates the image number.
- Elevation $\in \{\pm 90, \pm 60, \pm 30, \pm 15, +0\}$ indicates the elevation value of the face recorded in that image.
- OOPRotation $\in \{\pm 90, \pm 75, \pm 60, \pm 45, \pm 30, \pm 15, +0\}$ indicates the out-of-plane rotation value corresponding to that image.

In figure (4.1) an image example from this database can be seen. This image corresponds to the file *personne09131-15-30.jpg*, i.e., is the image number 31 of the series 1 of person number 09 and the recorded pose corresponds to an elevation of −15 degrees and an out-of-plane rotation of −30 degrees.

The image acquisition process was done using the *FAME Platform* of the *PRIMA*[1] *Team* in INRIA Rhone-Alpes. The different poses are obtained by asking the

[1] http://www-prima.inrialpes.fr/

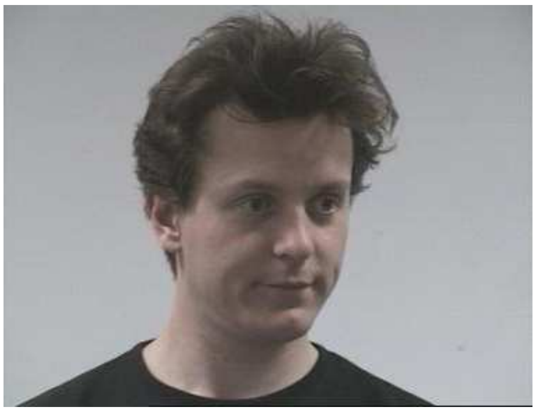
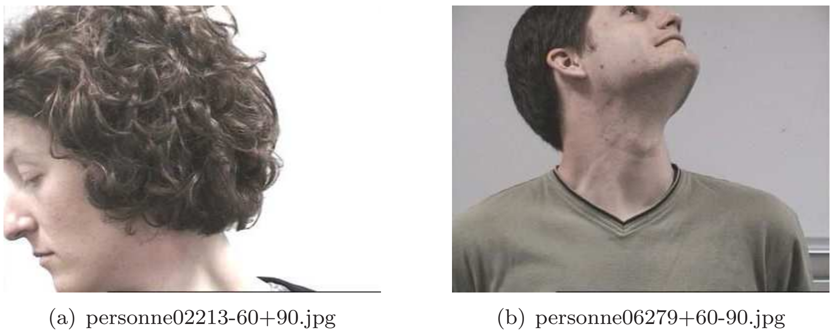


Figure 4.1: Example of an image from the INRIALPES head pose database. This image corresponds to the file personne09131-15-30.jpg



(a) personne02213-60+90.jpg

(b) personne06279+60-90.jpg

Figure 4.2: Incorrect face centring

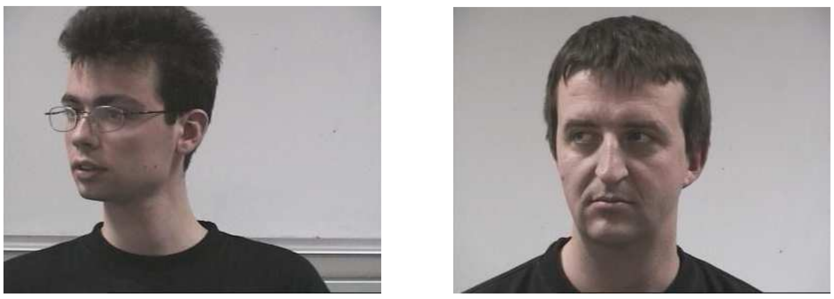
subject who is being recorded to look at different marks distributed over the room covering a half-sphere in front of the person.

4.1.1.1 Problems and Setbacks

At the time of using this database, some problems arise making not possible to use all the images. There are basically two problems, both caused by the "manual" positioning of the face to generate the different poses.

The first problem, which can be observed in figure (4.2), is that not in all the recordings a good centring of the face in the image has been done. This produces that some faces, when producing the different poses, go outside of the recording space due to the necessary movement that the subject who is being recorded must do with his or her face. This problem is easy to detect simply by looking at the images.

The second problem, which is not as easy to detect as the first one, is caused by



(a) personne05148+0+30.jpg (b) personne07148+0+30.jpg

Figure 4.3: Incorrect pose positioning. Both images are representing the same pose, but it can be seen, the pose of each subject is very different.

an incorrect pose adopted by the subject when looking at the marks. In figure (4.3(a)), a person looking correctly at the mark corresponding to a pose of 0 degrees of elevation and +30 degrees of out-of-plane rotation can be seen. In figure (4.3(b)), another person looking at the same mark can also be seen. As we can check, in the second figure the recorded subject is looking at the mark moving his eyes to the right, and not moving his head. The hypothetical noise that the use of these kind of images in a training set could generate is very important, since faces could be inserted in wrong sets.

In conclusion, an accurate revision of the images must be done before their use in any training or testing processes, mainly to avoid the second kind of wrong images described above, since considerable noise could be inserted in the used sets.

4.1.2 Pose, Illumination and Expression Database

The CMU Pose, Illumination and Expression (PIE) database, described in [82], is a facial database containing images from 68 people across 13 different poses, under 43 different illumination conditions and with 4 different expressions (neutral, smile, blink and talk). The images were recorded using 13 cameras in the CMU 3D Room with a special flash system to obtain the different illumination conditions.

The cameras are distributed so that 9 of them are located roughly at the same height than the recorded face, covering the out-of-plane rotation values from −90 degrees to +90 degrees with an approximate separation of 22.5 degrees. Two of the 4 remaining cameras simulate security cameras in the corner of a room, and the last two cameras are located slightly above and below the central camera. An example of the images obtained by each of them can be seen in figure (4.4).
Regarding the illumination, the flash system, that is composed of 21 flashes, allows to generate 21 different illumination conditions with the room lights switched on and 21 more with the lights switched off.

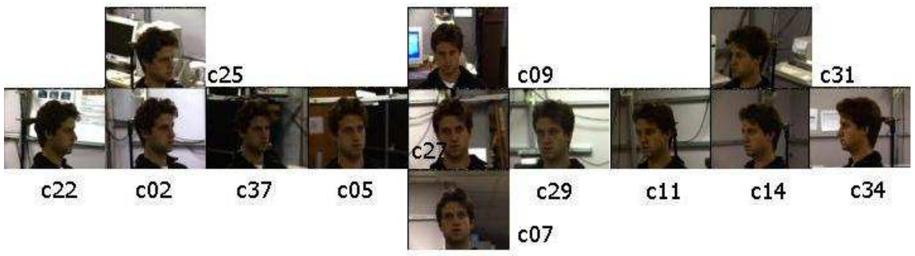
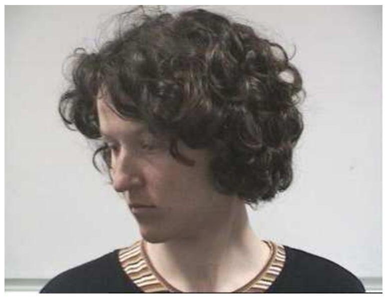


Figure 4.4: Cameras numbering in the PIE Database


(a) INRIALPES Head Pose Database image

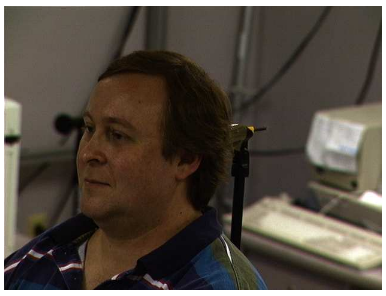

(b) CMU PIE Database image

Figure 4.5: Pose comparison between images from the INRIALPES Head Pose Database and the CMU PIE Database

4.1.2.1 Problems and Setbacks

Since there is no need of any movement by the person who is being recorded, because there are several cameras to do the recordings, there are no problems derived from this movement like in the INRIALPES database. This is an advantage but there is also a setback, the images out of he head height plane are not useful to simulate a pose of somebody looking at one point not just in front of him. This can be easily seen in figure (4.5), where the difference between somebody recorded from a camera not in his front and somebody looking at a point not in his front is very clear.

Then, for the purpose of a face detector and pose estimator, only the images from the head height plane in the CMU PIE database are useful.

4.1.3 Color FERET Database

The *Color FERET* database [83, 84] is a color version of the old *FERET* database or *Grey FERET* database. It was recorded by the NIST (National Institute of Standards and Technology) and it contains a total of 11.338 facial images at 13 different poses from 994 subjects, recorded in 15 session between 1993 and 1996.

Table 4.1: Color FERET Poses

Pose Name	Approximate Angle	Numer of Images	Number of Subjects
fa	0	1364	994
fb	0	1358	993
hl	+67.5	1267	917
hr	−67.5	1320	953
pl	+90	1312	960
pr	−90	1363	994
ql	+22.5	761	501
qr	−22.5	761	501
ra	+45	321	261
rb	+15	321	261
rc	−15	610	423
rd	−45	290	236
re	−75	290	236

The images have a size of 512×768 pixels, they are in PPM-format and all their names follow the next format

```
data/images/PersonID/PersonID_YYMMDD_PN_F.ppm.bz2
```

- PersonID $\in \{00000, 00001, \ldots, 99999\}$ indicates the number of the person recorded.
- YYMMDD indicates the data when the image was recorded.
- PN $\in \{\mathrm{fa, fb, hl, hr, pl, pr, ql, qr, ra, rb, rc, rd, re}\}$ indicates the pose (see table 4.1 for the meanings of each of these acronyms).
- F is an optional flag that can have the next values:

  a: subject is wearing glasses.

  b: subject changed his or her hairstyle for the image and is not wearing glasses.

  c: subject changed his or her hairstyle for the image and is wearing glasses.

In addition to all the facial images, this database has also background images as well as a wide range of "meta-data", like age, race, date of recording, sex, beard, glasses...

4.1.4 Non-Face Images

Another important point in order to do a good training of the classifiers is to select a good "Negative" set of data. It is not possible to use all the set of "Non-Face" images because, even in the case of greyscale images of 20×20 pixels, this set is too big (of the order of 10^{40} images). For this reason, a common practice is to collect natural images that do not contain human faces and chop them in

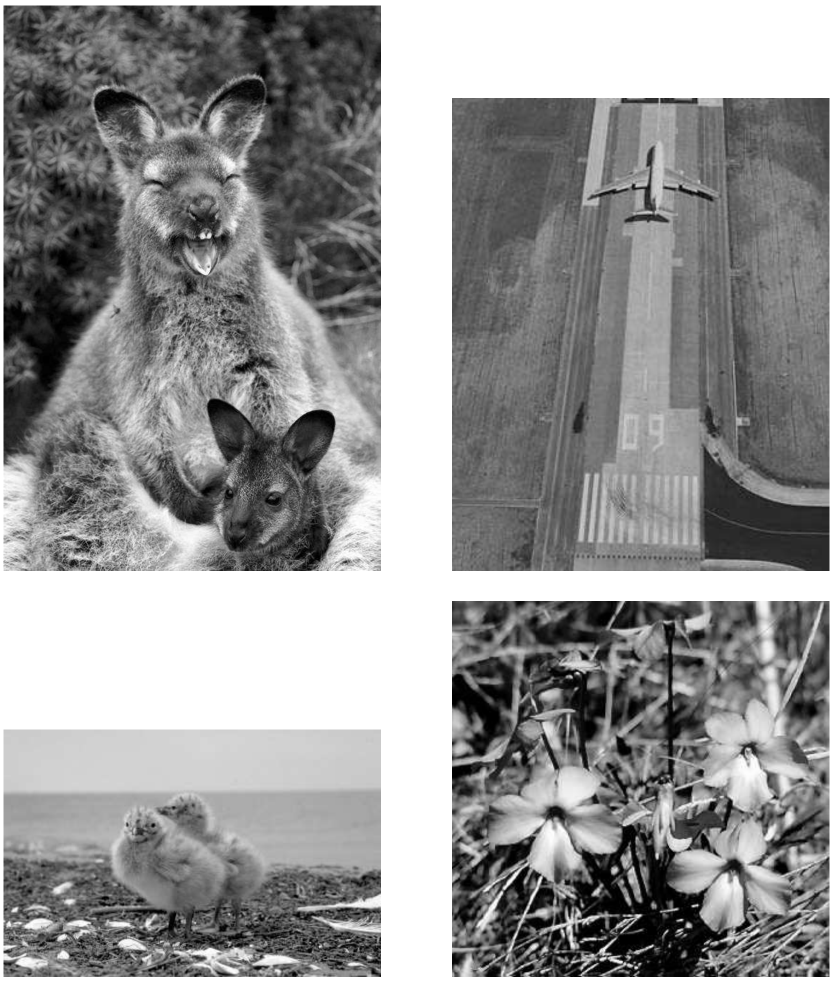


Figure 4.6: Some examples of images from the *Non-Face* set

pieces of the desired size (20×20 pixels in this case).

The *Non-Face Images* set used in this project consists of 196 images randomly chosen from the web with variable sizes that have been cut in 20×20 pixel pieces without overlapping, producing a *Non-Face* set of 97.947 images. Some examples of the images used as non-faces can be seen in figure (4.6).

4.1.5 Training Set

The final *Training Set* is a combination of images from the PIE database and the INRIALPES Head Pose database. From the PIE database, 47.954 images from the cameras c22, c02, c37, c05, c22, c29, c11, c14 and c34 have been used, and from the INRIALPES database 2.597 images from all the poses excepting for the values of elevation of $+90$, 0 and -90.

This total amount of 50.551 images has been manually cut, scaled to a size of 20×20 pixels and converted to grayscale images. Depending on the classifier that was being trained, the *Training Set* was divided in subsets according to the desired pose, as it will be seen in section 4.3.

With respect to the non-face images, all the blocks of 20×20 pixels without overlapping from the 196 images from the non-face set were considered, producing a total of 97.947 20×20 pixel images. All these pictures were converted to greyscale and, with them, 6 disjoint groups of 1.000, 5.000, 10.000, 15.000, 20.000 and 30.000 images were formed by random selection over the 97.947 image set.

4.2 Classifiers Structure

In section 3.1, two kinds of filters have been shown. On the one hand, *Haar-based* filters have very low computing requirements but are not selective enough to distinguish between faces with different poses. On the other hand, *Anisotropic Gaussian* filters are selective enough to distinguish between faces with different poses but the computation of its response is very "CPU consuming". For this reason, some strategy must be followed, as it will be described in section 4.3, in order to take as much profit as possible from each kind of filters without punishing substantially the final performance of the system.

In addition to the combination of the outputs that will be explained in section 4.3, another technique in order to speed up the rejection of *non-face* inputs will be used. This technique consists of building cascades of strong classifiers instead of a monolithic strong classifier.

The structure of a cascade of strong classifiers can be seen in figure (4.7). As it is shown there, a cascade of classifiers is composed of several classifiers connected by the positive output. That means that if one input is classified as "Face", then it has been classified by all the classifiers, in descending order, as "Face". But, if one input is classified by any of the stages as "Non-Face", then it is immediately discarded and it does not go to the lower stages of the cascade. Building this cascade with classifiers so that the lower in the structure they are, the higher number of features they have, allows to accelerate substantially the rejection of negative inputs (see Appendix D for an example). In the following, a classifier will mean a cascade of classifiers unless the contrary is specified.

Then, two kinds of filters will be used, producing with them two kinds of classifiers by means of the *AdaBoost* algorithm. With *Anisotropic Gaussian* filters, several classifiers will be built in order to distinguish between different poses. But with *Haar-based* filters, only one classifier will be built. This classifier produced with *Haar-based* filters will be called the *General Classifier* or simply *GC*. The classifiers produced using *Anisotropic Gaussian* filters will be called *Pose Classifiers* or *PC*s.

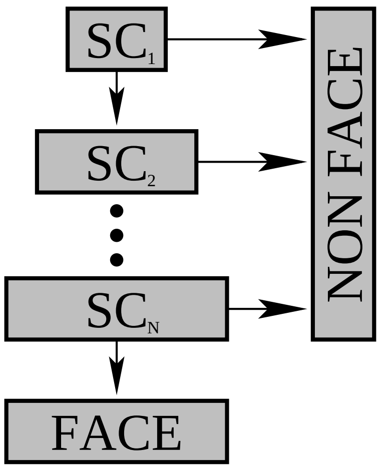


Figure 4.7: Cascade of N strong classifiers with an increasing number of features, i.e., with an increasing number of AdaBoost iterations

4.2.1 General Classifier

The aim of the *GC* is to remove the background and as much *non-face* image surface as possible, all this with a very small percentage of lost faces (less than 1%) and as fast as possible. For these reasons, it is not necessary a high selectivity and it must be fast, is why *Haar-based* filters are used to build this classifier.

All the possible *Haar-based* filters (see figure (3.1)) that can be inserted in a 20×20 pixels image are considered. This makes 69.790 filters, allowing to extract then 69.790 features from an image.

The employed *GC* is a cascade of 4 strong classifiers (see figure (4.7) with 10, 20, 40 and 80 features (iterations of the *AdaBoost* algorithm) respectively. The process to train the cascade of classifiers is detailed in algorithms 3 and 4. *(T)FS* and *(T)NFS* designate "(Training) Face Set" and "(Training) Non-Face Set" respectively. In the different tests that have been done with the system described here, the *GC* erased around a 97% of the samples passed to it.

4.2.2 Pose Classifiers

Regarding the *PC*s, their objective is to detect faces but only with the pose of the faces that the classifier has been trained with. Then, a high selectivity is required here, and for that reason, *Anisotropic Gaussian* filters are used to build these classifiers. Therefore, the output of *PC* classifiers is not "Face" or "Non-Face" but "Face with a specific pose" or "Not a face with a specific pose".

Pose Classifiers are 4 level cascades of classifiers with 5, 10, 20 and 40 features respectively. As pose, the out-of-plane rotation in the interval $[-\pi/3, \pi/3]$ *rad*

Algorithm 3: Training algorithm for the *GC*

```
1 $TFS_1 \leftarrow$ {INRIALPES Dataset}$\cup${Expression subset from PIE};
2 $TNFS_1 \leftarrow$ 5.000 images from the $NFS$;
3 for $i = 1$ to 4 do
4     Train stage number $i$ of the $GC$ using $TFS_i$ and $TNFS_i$;
5     Adjust final asymmetric threshold to have a high percentage ($\geq 99\%$)
      of true acceptance over $FS$ with a false acceptance rate $\leq 60\%$ over
      $NFS$ when possible;
6     $TNFS_{i+1} \leftarrow$ {Update of the $TNFS$ made with algorithm 4};
7 end
```

Algorithm 4: Update of the *TNFS*

```
  input  : TNFS$_i$, Number of false accepted faces
  output: TNFS$_{i+1}$
1 $MAXNewFaces =$
  min{Number of false accepted faces, 20% of $\#TNFS_i$};
2 $TNFS_{i+1} \leftarrow \{TNFS_i\} \cup \{MAXNewFaces$ from the false accepted faces$\}$;
```

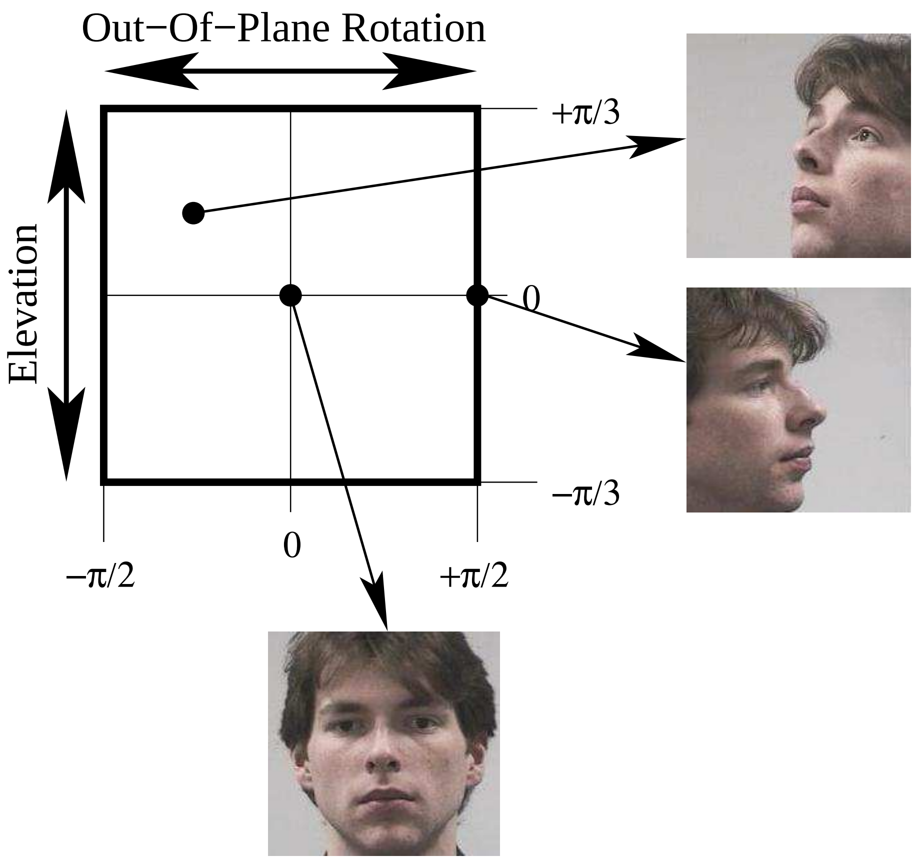


Figure 4.8: Examples of different poses

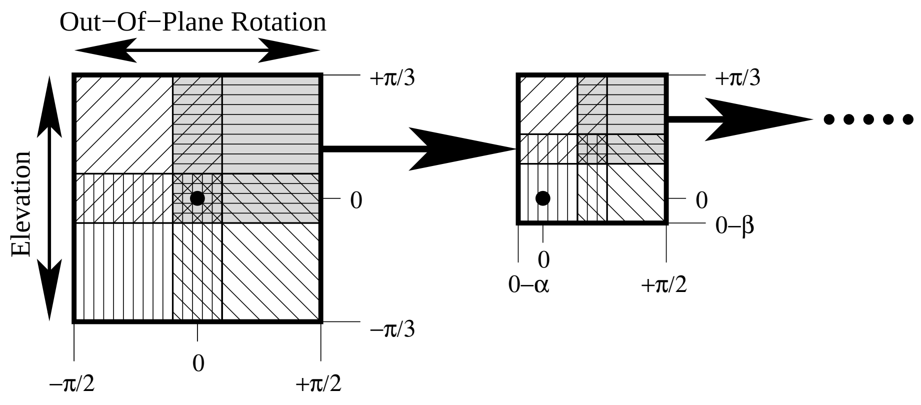


Figure 4.9: Segmentation scheme of the considered movements

Table 4.2: Parameters used to generate the set of *Anisotropic Gaussian* filters

Parameter	Minimum	Maximum	Step
Radius (pixels)	3	13	3
Rotation (radians)	0	2π	$\pi/9$
X Scale (pixels)	1	5	2
Y Scale (pixels)	1	5	2

and the elevation in the interval $[-\pi/2, \pi/2]$ *rad* are contemplated (see figure (4.8)). Both movements are illustrated in figures (1.1(a)) and (1.1(b)) respectively, and they are considered together, i.e., a classifier trained to detect faces with a given pose will detect faces with a given elevation and out-of-plane rotation, and not each one by separate. To train these classifiers, the plane generated by the variables of out-of-plane rotation and elevation is subdivided in pieces with a small overlapping margin, as indicated in figure (4.9), and this procedure is repeated until small regions are obtained. The overlapping margin is intended to avoid abrupt transitions between classifiers, and it will allow to do a better estimation of the pose by combining outputs from different classifiers.

The set of *Anisotropic Gaussian* filters used to extract features from images has been generated using the parameters specified in table 4.2, producing 103.968 filters for a size of 20×20 pixels.

The training process can be schematically seen in the algorithm (5). Note that "Construct Cascade of Classifiers for Pose Classifier" is simply the algorithm (3) adapting the training images sets and the number of stages and features. The final training process has not been as automatic as expressed in that algorithm due to problems with the number of images for some of the poses that forced to consider some poses together. For the exact partition of the pose plane see appendices B and C.

Algorithm 5: Training algorithm for Pose Classifiers

```
1 Divide total domain;
2 for $L = 1$ to 4 (number of domain subdivisions) do
3     for $D = 1$ to number of subdomain pieces do
4         Construct Cascade of Classifiers for Pose Classifier;
5     end
6     Redivide domain;
7 end
```

4.3 Decision Tree Structure

At this point, the process to build classifiers specialised in the detection of faces with a specific pose, PCs, and the process to build a very fast classifier set aside for removing the background and as much non-face image portions as possible, GC, have been described. Now, the tree structure combining all these classifiers will be detailed, but before that, some notation aspects must be specified:

- PC_i designates the "Pose Classifier" number i that receives the output from the GC

- $PC_{i_1,\ldots,i_n}$ designates the "Pose Classifier" number i_n that receives the output from $PC_{i_1,\ldots,i_{n-1}}$

The tree of classifiers is organised as follows. On the top node, the *root* node, the GC is placed. This classifier sends to the nodes on the first stage all the face candidates without any pose filtering, i.e., candidates to face in any pose. Then, in this first stage, 4 *Pose Classifiers*, PC_1, PC_2, PC_3 and PC_4, are placed. Each one of these classifiers has been trained with images from a quadrant of the pose plane, each one with a small overlapping margin with their neighbour regions. Therefore, these classifiers are selective in pose and they will only send to their "classifier sons" those face candidates to have a specific pose. In the second stage, the "sons" of PC_1 ($PC_{1,1}$, $PC_{1,2}$, $PC_{1,3}$ and $PC_{1,4}$), PC_2 ($PC_{2,1}$, $PC_{2,2}$, $PC_{2,3}$ and $PC_{2,4}$), PC_3 ($PC_{3,1}$, $PC_{3,2}$, $PC_{3,3}$ and $PC_{3,4}$) and PC_4 ($PC_{4,1}$, $PC_{4,2}$, $PC_{4,3}$ and $PC_{4,4}$) are placed, each one trained with a portion of the domain of its "father". This process is repeated (see figure (4.10) for a sketch of how these classifiers are connected) until a tree with 5 stages and a total of 60 classifiers is built. All the nodes have a connection with the "Non Face" special node, whose only task is to take out samples from the decision tree.

In appendix C, the exact structure of this tree of classifiers with all its connections is described. As it can be seen there, some classifiers can receive face candidates from more than one "father classifier". This is due to the overlapping regions in the pose domains.

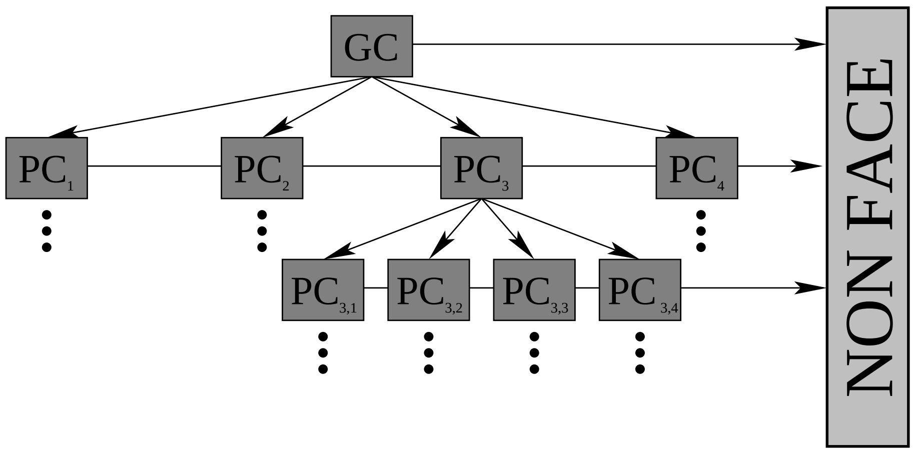


Figure 4.10: Structure of the final tree of classifiers

4.4 Detection Process

Up to now, the components that form the multi-view face detector have been described, going from the “bottom”, the AdaBoost algorithm, to the “top”, the final tree structure that contains all the classifiers, but nothing has been explained about how to detect faces in an input image using this final tree of classifiers, whose input is a greyscale image of 20×20 pixels.

Basically, when dealing with the face detection problem in an image without any previous knowledge of its content, exists two problems. The first one is that a hypothetical face can be in any point of the image and the second one is that this face can have any size. Furthermore, different faces in the same image can have very different sizes.

Then, the strategy to detect faces in an input image has two steps, as well as a previous greyscale conversion if the input is not a greyscale image. The first step is to build what is known as a *pyramid* of images (see figure (4.11)). That is a set of images composed of the original image and subscaled versions of it. In our case, all the subsampled images by a factor of 1.25 are calculated until it is achieved an image whose subsampled version would have a side smaller than 20 pixels. Then, over the whole set of images from the *pyramid* of images, the second step is to apply the tree of classifiers to each portion of 20×20 pixels. This two steps assure that a face in any position and of any size (if the scale factor is small enough) can be detected.

4.4.1 Peculiarities of the System

In addition to the general operation of the system, it presents some peculiarities, basically in the computation of the results of each classifier:

- All the filters have zero mean and are applied to normalised greyscale

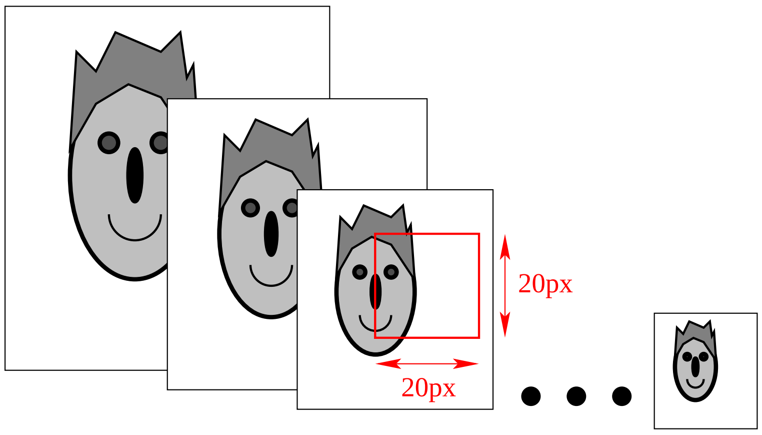


Figure 4.11: Example of a pyramid of images. The image on the left is the original one, and the image on the right is the smallest one, whose subsampled version would have a side smaller than 20 pixels. The scanning 20×20 pixels window can be seen in red in the third image.

images (images with pixel values in the range $[0, 1]$). This implies that the output of a filter applied to an image, what is called a feature, is a real value, and not $+1$ or -1.

- Due to the previous point, the used *AdaBoost* algorithm is an adapted version of the algorithm described in algorithm (2). Basically, the difference with it, is only that the "sign" function is replaced by a threshold:

$$\text{If } \sum_{t=1}^{T} \alpha_t h_t(x) \geq \sum_{t=1}^{T} \frac{1}{2}\alpha_t \text{ then "Face", else "Non-Face"}$$

  This $\frac{1}{2}$ is the mean value of $h_i(x)$.

- Finally, the last peculiarity is that in the cascade structure of the classifiers, another threshold is used between stages, so that the expression written in the previous point is finally:

$$\text{If StageThreshold} \cdot \sum_{t=1}^{T} \alpha_t h_t(x) \geq \sum_{t=1}^{T} \frac{1}{2}\alpha_t \text{ then "Face", else "Non-Face"}$$

  This "StageThreshold" is the "Final Asymmetric Threshold" calculated in step 5 of algorithm (3).

Part III

Tests and Results

Chapter 5

Tests and Results

In the previous chapters, we have described a system intended to detect faces in any of the considered poses, as well as to estimate the pose of the face that it has detected. Next, some tests of that system will be carried out in order to see its performance.

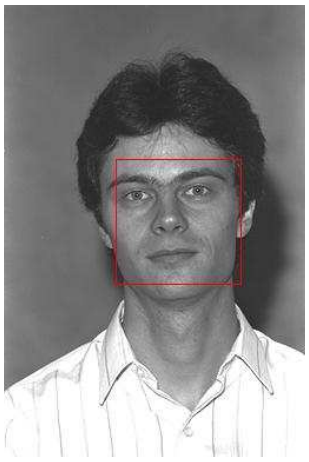
One very important concept must be defined before any test can be carried out, "what is considered as a *correct detection*". It will be considered as a *correct detection* a detection that contains the eye(s) and the mouth, and whose sides are shorter than twice the width and the height of the detected head (see figure (5.1) for some examples). All the detections that do not accomplish these conditions are not considered as *correct detections*. A more unbiased evaluation protocol could be considered by extending the work of Popovici et al. in [85], but this is not in the scope of this project.

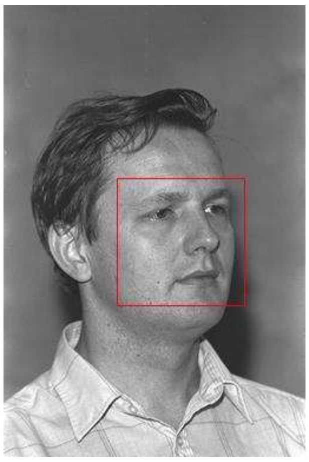
But the system developed here is also able to estimate the pose of the face that has been detected, then "what is a correct pose estimation" must be also defined. The problem with that definition is that we do not have calibrated images to test if the estimation is correct or not. For that reason, it has been decided to accept as good pose estimations those that the estimated pose is approximately that that a person would say. To make this task easier, two circumferences are drawn alongside detection. The top one has an arrow indicating the out-of-plane rotation value, and the bottom one has an arrow indicating the elevation value (see figure (5.2)).

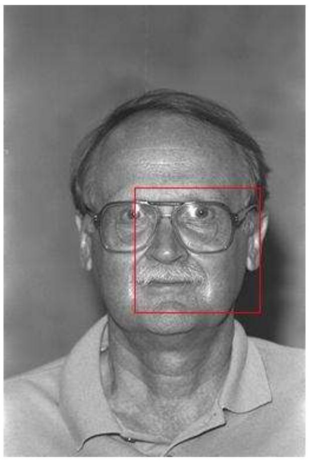
Then, following these criterions, the images showed in figure (5.3) are examples of correct face detections and correct pose estimation.

At this point, the parameters that will be used to measure the system performance can be defined. These parameters are the *True Acceptance Rate* (TAR) and the number of *False Detections*. The *True Acceptance Rate* shows the rate of correct detections and is calculated as the rate between the number of correct detected faces and the real number of faces, i.e.,

$$\text{TAR} = \frac{\#\text{Correct Detections}}{\#\text{Faces}} \tag{5.1}$$


(a) Correct detection


(b) Correct detection


(c) False detection

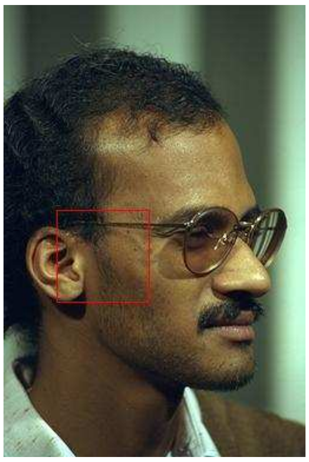

(d) False detection

Figure 5.1: In images (5.1(a)) and (5.1(a)), two face detections that would be considered as correct are shown. In images (5.1(a)) and (5.1(a)), the detections would be considered as incorrect, and therefore counted as *false detections*

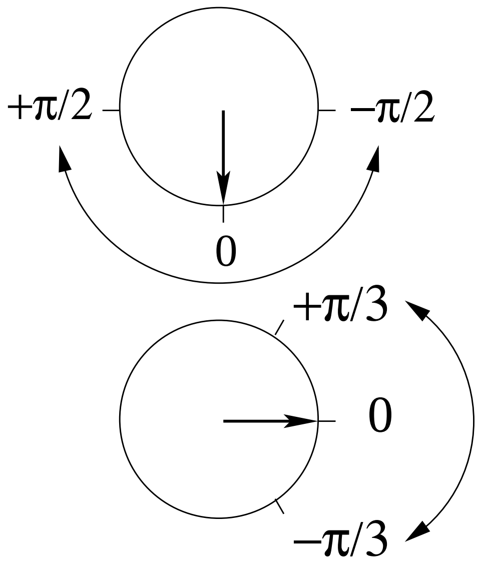


Figure 5.2: Out-of-plane rotation and elevation representation

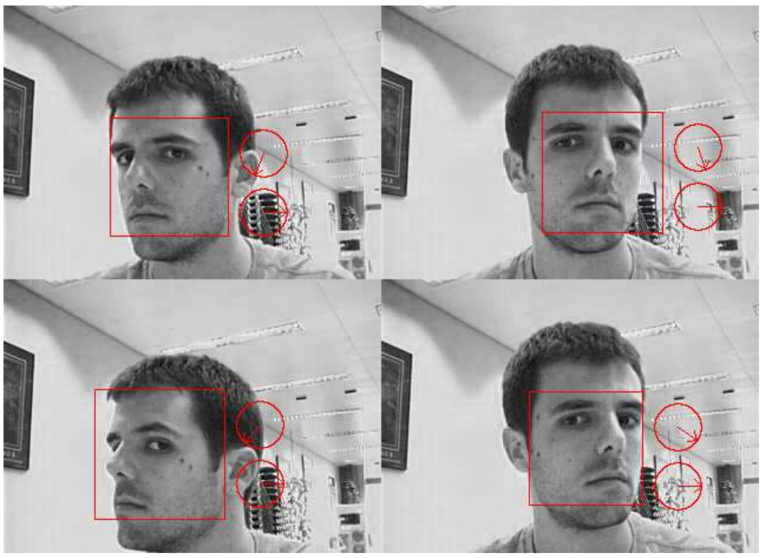


Figure 5.3: Correct detections and pose estimations

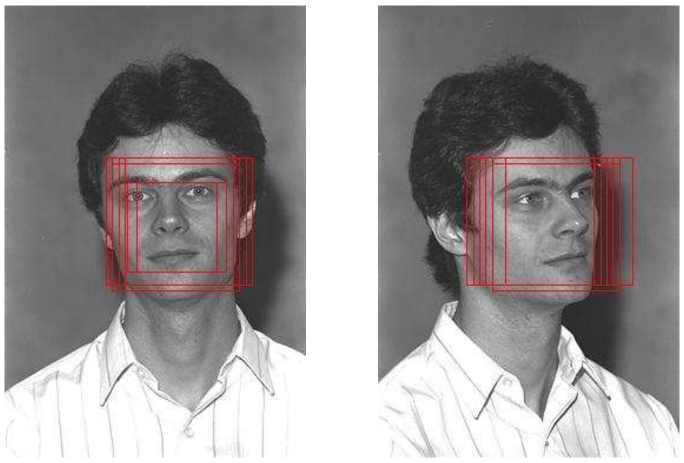


Figure 5.4: Examples of multiple detections of the same face

This *True Acceptance Rate* induces another equivalent parameter, called the *False Rejection Rate* (FRR), that shows the rate between the number of not detected faces and the total number of faces. It is calculated as

$$\text{FRR} = \frac{\#\text{Faces} - \#\text{Correct Detections}}{\#\text{Faces}} = 1 - \text{TAR} \tag{5.2}$$

The second parameter, the number of *False Detections*, is the number of incorrectly detected faces as well as non-faces detected as faces. This parameter gives an idea of the robustness of the system and it depends on the system by itself but also on the arbitration used to remove multiple detections. The *False Positive Rate* (FPR), the equivalent of the FRR for false detections, is not used here since it would be extremely low and it would not give useful information. This parameter would be very low due to the high number of windows that are scanned in an image for the different scaling factors and positions, making the denominator of this rate very high.

Then, an important point when the system is being evaluated is the arbitration done over the total set of detections. This step is mandatory due to the "scanning" principle of the system that origins multiple detections in close positions and scales (see figure (5.4)). In the following tests, the results from 6 very simple different arbitrations will be reported:

- **ArbA:** Mean detection from each "single pass" cluster of detections.
- **ArbB:** Best detection, according to the final score of each detection, from each "single pass" cluster of detections.
- **ArbC:** Weighted, according to the final score of each detection, mean detection from each "single pass" cluster of detections.
- **ArbD:** Mean detection from each "multiple pass" cluster of detections.

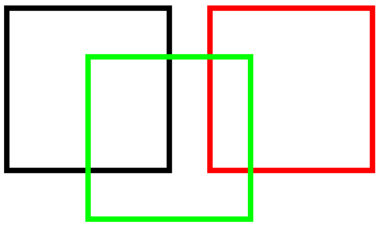


Figure 5.5: In “single pass” cluster computation, considering as the initial detection the black one, the black detection and the green detection would form a cluster of detections, while the red detection would form another independent detection. In “multiple pass” cluster computation, the three detections would form one only cluster of detections.

- **ArbE:** Best detection, according to the final score of each detection, from each “multiple pass” cluster of detections.
- **ArbF:** Weighted, according to the final score of each detection, mean detection from each “multiple pass” cluster of detections.

When talking about “single pass” clusters, we mean that one detection is taken as the initial one and all the other detections that overlap this one are considered as belonging to the cluster. While in the “multiple pass” clusters, each detection added to the cluster can be considered to compute its overlapping with additional detections (see figure (5.5) to see an example).
In order to have also an idea of where could be the problem in a hypothetical bad detection, the *True Acceptance Rate* without any arbitration and without caring on the number of false detections will be given.

5.1 Test Over Color FERET Database

The whole Color FERET Database is too big to be tested with the face detector and pose estimator system, since the results must be checked “by hand” to see if the detection and the pose estimation has been correctly done. For this reason a small subset from the small images (256×384 pixels) of FERET has been chosen. All the images from the 20 first subjects (PersonID $\in \{00001, 00002, \ldots, 00020\}$), making a test set composed of 194 images with 194 faces (one face per image) in different poses but always with elevation 0 degrees.

The results showed in table (5.1) are obtained scanning these images in all possible locations with windows of sizes between 77×77 pixels and the maximum that fits in the image with a scale factor of 1.4. This makes a total of 7.514 windows to be scanned by the system, of which around a 97% are erased by the *GC*, leaving for the lower stages only around 200 windows to be tested. Note that in the pose TAR calculation, the denominator is not the total number of faces but the total number of correctly detected faces, since in an incorrectly detected face or in a non-face detected as face, the pose estimation has no sense.

Table 5.1: Results from the test on the Color FERET Subset

Arbitration	TAR	False Detections	Pose TAR
None	85.1%	–	–
ArbA	76.3%	310	75.0%
ArbB	70.1%	322	83.8%
ArbC	74.7%	313	76.6%
ArbD	51.5%	251	77.0%
ArbE	61.9%	231	82.5%
ArbF	55.2%	244	72.9%

Another interesting point to observe here is the number of windows discarded in each stage of the tree, to confirm whether the proposed progressive segmentation of the pose plane prevents the propagation of positive windows from the *GC* to all the classifiers, speeding up the system. And thus it is, since over this test set, the first stage of the tree, classifiers 1, 2, 3 and 4 (see appendix C), erase the 49% of the samples that they process, the second stage, classifiers 5 to 20, the 86%, and the third stage, classifiers 27 to 30 and 36 to 39, erase the 79%.

The mean time of processing in a Pentium Centrino at 2.0GHz for each image is about 0.2 seconds, but it depends on the outgoing images from the *GC*, that makes range the processing time between 0.09 seconds and 0.61 seconds.

5.2 "Live" Test

In order to test the system in a more real situation, it has been implemented in a program able to read images directly from a webcam, allowing to simulate a face detection in a video-conferencing-like environment. This program has allowed to find out that the system is very dependent on the illumination, and that the pose estimation for values of elevation different to 0 is not accurate. This last problem has a very easy and probably explanation, datasets, since the only poses with abundant data and then the only classifiers trained with an acceptable number of training samples are those for elevation 0.

The system can perform without problems in real-time in a Pentium Centrino at 2.0GHz, processing the 320×240 pixel images supplied by the webcam at around 8 frames per second with the smallest face dimension limited to 76 pixels and a scaling factor of 1.25.

In the CD that comes with this document, some videos, as a subjective data from this test, are available to see how well the system works in good illumination conditions and for values of 0 degrees of elevation, and how the system fails when there is a bad illumination or the face pose has an elevation substantially different to 0.

Part IV

Conclusions and Future Work

Chapter 6

Conclusions

Along this document, a new approach to the face detection and pose estimation problem has been proposed. This new approach makes two main contributions. On the one hand, it shows the feasibility of building classifiers trained to detect faces in specific regions of the pose space. On the other hand, it demonstrates that the proposed method of building a tree of classifiers where the more we descend on the structure, the more concrete pose the classifiers detect, is a good way of combining these pose-specific classifiers.

In addition, the idea of placing a Haar-based boosted classifier on the top of the tree structure, in order to erase quickly as much background as possible, has also been confirmed as a good idea. Moreover, this classifier allows the system to work in real-time at an acceptable frame rate without using "time-dimension information", i.e., without enclosing the scanning region, processing the whole image on each frame.

The obtained results are very promissing, since the system achieves a *True Acceptance Rate* before the arbitration like in other state-of-the-art techniques, but with the advantages that this system is not limited to frontal faces and that the system can estimate the pose of each detected face very accurately, providing hypothetical third party applications with very valuable information.

But the proposed system presents also some problems, like the high number of false detections. This high number of false detection implies a more difficult arbitration process due to the "noise" that these false detections add to the correct ones. Moreover, the correct operation with poses with an elevation value different to 0 degrees could not be confirmed. The bad function in those poses is probably due to the bad classifiers that were placed in the tree for them, since we only had a very limited amount of images to train them, but this aspect should be confirmed.

In order to try to solve the problems that have been observed, as well as to study in depth some aspects of the designed system, a wide variety of future work can be done taking this document as basis. In the next chapter, some proposals of interesting future work are suggested.

Chapter 7

Future Work

The work presented along this document opens a wide range of possible future work. In order to make my proposed future work more understandable to the reader and not only enumerate a series of improvements, these proposed tasks will be divided into three groups, *"Pre" Future Work*, *"During" Future Work* and *"Post" Future Work*. In the *"Pre" Future Work* group, there are proposed tasks or improvements related with things that were done during the preparation of this project. In the *"During" Future Work*, they are related with things that were done during the development of this project, and finally, in the *"Post" Future Work* there are tasks or improvements that could be done with this project already developed.

7.1 "Pre" Future Work

- ***Data Set***: One of the biggest problems along the course of this project has been to collect good and enough data in order to make a good training of each classifier. This has not been easy, and the amount of data finally collected has not been enough to build a uniform sampling of the pose plane (see section 4.3 and appendix B). The principal problems in the data collection process have been that the databases that made a good pose recording, as the INRIALPES Head Pose database (see section 4.1.1), did not have a big amount of data, and the databases that had a big amount of data, as the CMU PIE database (see section 4.1.2), did not make a good pose recording for our purposes (see section 4.1.2.1). For this reason, the recording of a good database with a good sampling of the pose plane and with an abundant quantity of images would be very interesting in order to improve this project as well as to help researchers who work in this field all over the world.

7.2 "During" Future Work

- ***Arbitration***: In this document, only very simple arbitrations has been done to erase multiple detections of the same face, using only the information provided by the last classifier on the tree that the sample has gone through. One of the first improvements that the system could have is a more complex and accurate arbitration. In particular, it would be interesting to study how to exploit the information of the route followed inside the tree as well as the successive obtained scores by the sample.

- ***Training***: In the trainings of classifiers carried out in this project, only non-face images were in the non-face training datasets. An interesting thing to try is if including faces from far poses in the non-face training set reduces the number of false detections or bad pose estimations in images.

- ***Real-Time***: In the face detector developed in this project, the information contributed by the time dimension, when working in real-time with data from a video sequence or captured from a camera, is not used. One improvement that would accelerate the execution speed of the face detector, taking into account this information, would be a Kalman or particle filter to follow the position, the scale and the pose of detected faces, since at a high frame rate it is not possible to have big changes in these parameters. Using these techniques, it would be only necessary to scan the whole image every certain time period to detect new faces or to correct possible errors.

- ***Eigenfaces***: *Haar-based* filters and *Anisotropic Gaussian* filters are used in this face detector. It would be a good work to study the possibility of using another kind of filters, in special, *eigenfaces* are matrices very correlated with faces and they could be used as projection matrix into the "faces space" to distinguish there, by means of a classifier built with the *AdaBoost* algorithm, between "Faces" and "Non Faces". For this reason, it could be interesting to study the results that would be obtained by using *eigenfaces* as filters to extract features from images.

- ***Evaluation of Anisotropic Gaussian filter parameters***: The parameters given in table (4.2) were chosen so that the generated filter set were big enough to produce acceptable results. An exhaustive study of the implications of each parameter would be very interesting to allow to generate a set of filters as selective as possible in face movements, allowing to hypothetically improve the face detection and/or pose estimation accuracy.

- ***Scale study***: The process to scan an image for a face used in this work has been an exhaustive search over a pyramid of images (see section 4.4). This requires a lot of computation time, above all for the biggest images from the pyramid, since a lot of windows of 20×20 pixels have to be tested. For this reason, a different strategy intended to delimit the number of scales that have to be scanned would be a great improvement in the system speed. With this objective, an study of some kind of image segmentation or texture granularity in images containing faces would be very interesting, since it may end in a preprocessing step able to estimate the scale candidates to be scanned for faces.

7.3 "Post" Future Work

- ***Face recognition***: The outputs of the face detector developed here would be very good inputs for a face recognition system due to the accurate face detection and, especially, to the pose information that the system provides. This pose information could be used by a hypothetical face recognition system in two ways. On the one hand, a pose correction of the detected face could be computed in order to always feed the face recognition with frontal images. On the other hand, instead of correcting the pose of the input face, this pose information could be used to choose between different recognition subsystems, for example if PCA is used to recognise faces, this pose information could choose between different projection matrices, each one composed of eigenfaces calculated with faces with the given pose.

- ***Evaluation procedure***: At the time of testing the performance of a face detection system, no standard procedure exists, forcing to each author to define what she or he considers as a correct detection. This has a very direct implication, usually it is not possible to compare performances from different approaches. Due to this, it would be extremely interesting to try to define a standard and widely accepted process to test face detector systems, and hypothetically build a testing environment to test algorithms.

Part V

Appendices

Appendix A

VC Dimension

VC Dimension is one of the most important concepts introduced by Vladimir Vapnik and Alexey Chervonenkis in their computer learning theory known as *Vapnik Chervonenkis Theory* or *VC Theory* (see "The Nature of Statistical Learning Theory" [86] for further details). This theory tries to explain statistically the learning process, covering 4 fields as explained in [86]:

- Theory of consistency of learning processes.
- Non-Asymptotic theory of the rate of convergence of learning processes.
- Theory of controlling the generalization ability of learning processes.
- Theory of constructing learning machines.

Before defining the *VC Dimension*, a previous concept must be defined, the *shattering*. Given a classication model $f(\underline{\alpha})$, where $\underline{\alpha}$ is a parameter vector, and given a set of data points $(\underline{x}_1, \ldots, \underline{x}_n)$, it is said that f *shatters* the set of data points if for all assignments of labels to those data points, there exists a parameter vector $\underline{\alpha}$ such that the model $f(\underline{\alpha})$ makes no errors classifying those points.

Vapnik and Chervonenkis defined the *VC Dimension* as follows. Considering the $2-$Class problem, given a set of functions $\{f(\underline{\alpha})\}$, where $\underline{\alpha}$ represents a generic set of parameters, then $f(\underline{x}, \underline{\alpha}) \in \{-1, +1\}\ \forall \underline{x}, \underline{\alpha}$. The *VC Dimension* for the set of functions $\{f(\underline{\alpha})\}$ is the maximum number of training points that can be shattered by $\{f(\underline{\alpha})\}$.

Appendix B

List of Classifiers

In the table B.1, the exact list of the 55 classifiers that were trained to build the decision tree is detailed. The table indicates the classifier *ID* as well as the cameras from the PIE database (see section 4.1.2) and the elevation and out-of-plane rotation values in degrees from the INRIALPES Head Pose (abbreviated as IHP in the table) database (see section 4.1.1) that were used for its training.

Table B.1: List of classifiers

ID	PIE Cameras	IHP Elevation	IHP OOP Rotation
0	OMPLIR	OMPLIR	OMPLIR
1	$27, 29, 11, 14, 34$	$+15, +30, +60$	$0, +15, +30, +45, +60, +75, +90$
2	$27, 05, 37, 02, 22$	$+15, +30, +60$	$0, -15, -30, -45, -60, -75, -90$
3	$27, 05, 37, 02, 22$	$-15, -30, -60$	$0, -15, -30, -45, -60, -75, -90$
4	$27, 29, 11, 14, 34$	$-15, -30, -60$	$0, +15, +30, +45, +60, +75, +90$
5	-	$+15, +30, +60$	$+45, +60, +75, +90$
6	-	$+15, +30, +60$	$0, +15, +30, +45$
7	$27, 29, 11$	$+15$	$0, +15, +30, +45$
8	$11, 14, 34$	$+15$	$+45, +60, +75, +90$
9	-	$+15, +30, +60$	$0, -15, -30, -45$
10	-	$+15, +30, +60$	$-45, -60, -75, -90$
11	$37, 02, 22$	$+15$	$-45, -60, -75, -90$
12	$27, 05, 37$	$+15$	$0, -15, -30, -45$
13	$27, 05, 37$	-15	$0, -15, -30, -45$
14	$37, 02, 22$	-15	$-45, -60, -75, -90$
15	-	$-15, -30, -60$	$-45, -60, -75, -90$
16	-	$-15, -30, -60$	$0, -15, -30, -45$
17	$11, 14, 34$	-15	$+45, +60, +75, +90$
18	$27, 29, 11$	-15	$0, +15, +30, +45$
19	-	$-15, -30, -60$	$0, +15, +30, +45$
20	-	$-15, -30, -60$	$+45, +60, +75, +90$
21	-	$+60$	$+45, +60, +75, +90$
22	-	$+15, +30$	$+45, +60$
23	-	$+15, +30$	$+75, +90$

ID	PIE Cameras	IHP Elevation	IHP OOP Rotation
24	-	$+60$	$0, +15, +30$
25	-	$+15, +30$	0
26	-	$+15, +30$	$+15, +30$
27	$29, 11$	-	-
28	$27, 29$	-	-
29	$14, 34$	-	-
30	$11, 14$	-	-
31	-	$+60$	$0, -15, -30$
32	-	$+15, +30$	$-15, -30$
33	-	$+60$	$-45, -60, -75, -90$
34	-	$+15, +30$	$-75, -90$
35	-	$+15, +30$	$-45, -60$
36	$37, 02$	-	-
37	$02, 22$	-	-
38	$27, 05$	-	-
39	$05, 37$	-	-
40	-	$-15, -30$	$-45, -60$
41	-	$-15, -30$	$-75, -90$
42	-	-60	$-45, -60, -75, -90$
43	-	$-15, -30$	0
44	-	$-15, -30$	$-15, -30$
45	-	-60	$0, -15, -30$
46	-	$-15, -30$	$+15, +30$
47	-	-60	$0, +15, +30$
48	-	$-15, -30$	$+75, +90$
49	-	$-15, -30$	$+45, +60$
50	-	-60	$+45, +60, +75, +90$
51	34	-	-
52	14	-	-
53	11	-	-
54	29	-	-
55	27	-	-
56	05	-	-
57	37	-	-
58	02	-	-
59	22	-	-

Appendix C

Exact Final Tree Structure

In table (C.1), the connections between the different nodes in the tree are specified. With this information and the information in table (B.1) it is possible to write the file *savetreemodel.conf* (contained in the CD) that passed as a parameter to the program SaveTreeModel allows to save the tree structure in a file to be used with *FPDetector*. A plot of this tree can be seen in figure (C.1).

Table C.1: List of connections in the tree of classifiers

ID	Connected Classifiers IDs
0	$1, 2, 3, 4$
1	$5, 6, 7, 8$
2	$9, 10, 11, 12$
3	$13, 14, 15, 16$
4	$17, 18, 19, 20$
5	$21, 22, 23$
6	$21, 22, 24, 25, 26$
7	$22, 25, 26, 27, 28$
8	$22, 23, 29, 30$
9	$25, 31, 32, 33, 35$
10	$33, 34, 35$
11	$34, 35, 36, 37$
12	$25, 32, 35, 38, 39$
13	$38, 39, 40, 43, 44$
14	$36, 37, 40, 41$
15	$40, 41, 42$
16	$40, 42, 43, 44, 45$
17	$29, 30, 48, 49$
18	$27, 28, 43, 46, 49$
19	$43, 46, 47, 49, 50$
20	$48, 49, 50$
27	$53, 54$
28	$54, 55$
29	$51, 52$
30	$52, 53$

ID	Connected Classifiers IDs
36	$57, 58$
37	$58, 59$
38	$55, 56$
39	$56, 57$

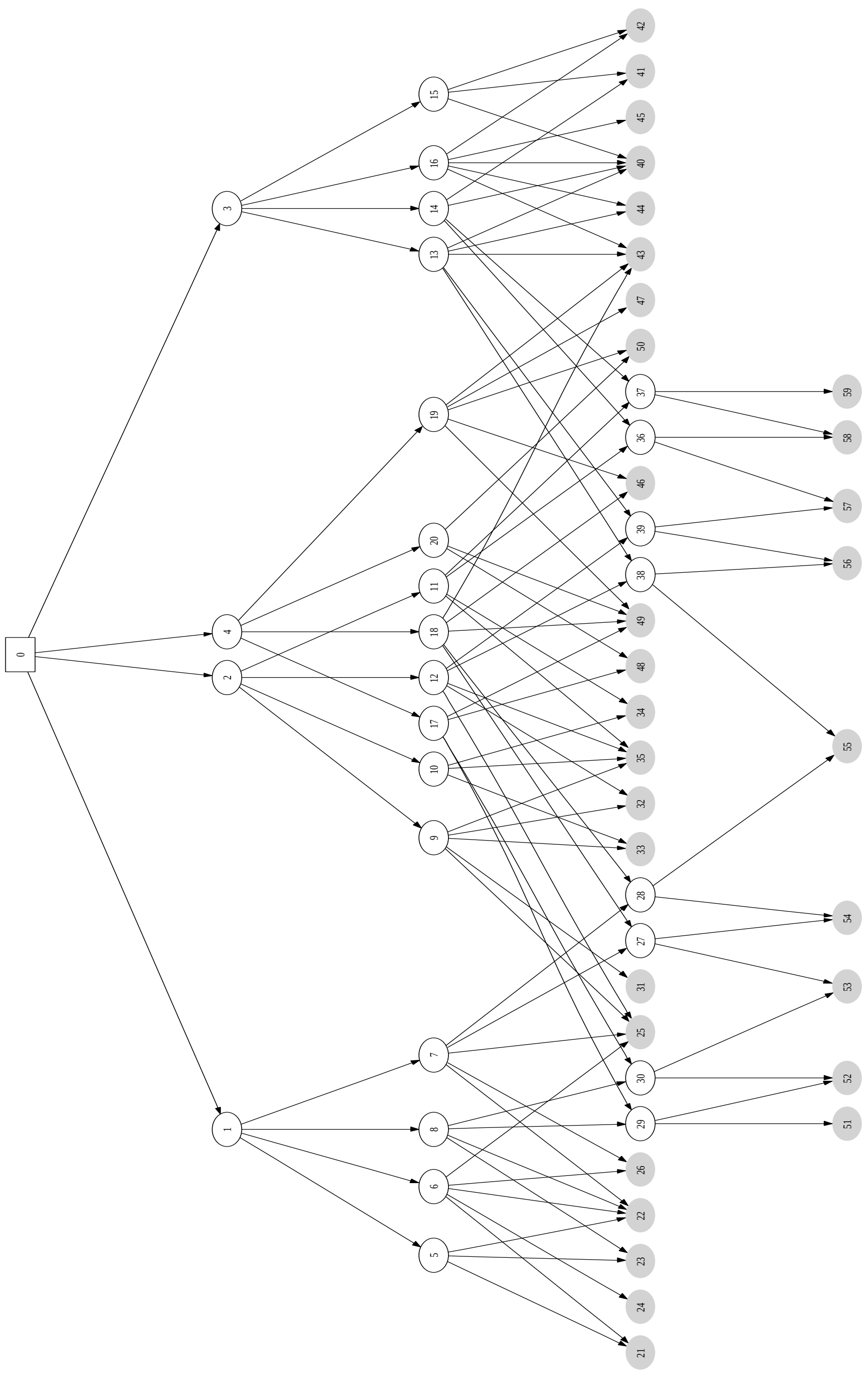


Figure C.1: Tree of classifiers generated using the connections specified in table (C.1). The *General Classifier* is drawn with a square and the *Leaf* nodes are painted in grey.

Appendix D

Cascade of Classifiers

A cascade of classifiers allows to accelerate the rejection of negative inputs, with respect to “monolitic” classifiers, because those samples rejected by a stage do not go to the lower stages. In a “monolitic” classifier, the decision of rejection or acceptance is carried out after computing all the features, while in the cascade only the features corresponding to the filters in the stage that is being computed are calculated to take a decision.

To show this, some outputs from the training of classifier number 55 (see Appendix B) are shown. In the first output there are some of the results of calculating the first stage threshold. As it can be seen, for the “optimum” threshold (less than 1% of faces rejected), 82.046 non-face images from the total of 97.947 are discarded. Then, more than an 80% of the images do not go to the lower stages and have been discarded using only 5 features.

```
Testing model...
Model OK!
Face nb:7237 NFace nb:97947
Results on iteration 0:
 FalseNegativeRate = 44.9909
 FalsePositiveRate = 0.0653415
 TotalTestError = 3.15646
 Face errors: 3256 Non face errors: 64
 Threshold = 1.95

...

Results on iteration 20:
 FalseNegativeRate = 0.704712
 FalsePositiveRate = 16.2371
 TotalTestError = 15.1671
 Face errors: 51 Non face errors: 15901
 Threshold = 0.950001
Optimum obtained (TAR>=99%) for Threshold=0.950001
```

This second output shows the results for the same classifier but with two stages. With these two stages (5+10 filters), 12.744 of the samples that go to the second stage are discarded, making a total of 94.790 samples from the 97.947 discarded. Then, to the third stage will only go 3.157 non-face images.

```
Testing model...
Model OK!
Face nb:7237 NFace nb:97947
Results on iteration 0:
 FalseNegativeRate = 93.1288
 FalsePositiveRate = 0
 TotalTestError = 6.40824
 Face errors: 6740 Non face errors: 0
 Threshold = 1.95

...

Results on iteration 19:
 FalseNegativeRate = 0.994888
 FalsePositiveRate = 3.22306
 TotalTestError = 3.06994
 Face errors: 72 Non face errors: 3157
 Threshold = 1
Optimum obtained (TAR>=99%) for Threshold=1
```

In this third output, the same is shown when the classifier already has 3 stages. The third stage discards 2.155 of the 3.157 non-face images that it receives, giving to the fourth stage only 1.002 non-face images from the initial set of 97.947 non-face images.

```
Testing model...
Model OK!
Face nb:7237 NFace nb:97947
Results on iteration 0:
 FalseNegativeRate = 99.9958
 FalsePositiveRate = 0
 TotalTestError = 6.8808
 Face errors: 7237 Non face errors: 0
 Threshold = 1.95

...

Results on iteration 21:
 FalseNegativeRate = 0.994888
 FalsePositiveRate = 1.02301
 TotalTestError = 1.02106
 Face errors: 72 Non face errors: 1002
 Threshold = 0.900001
Optimum obtained (TAR>=99%) for Threshold=0.900001
```

Finally, in this fourth output, the threshold for the fourth and last stage is calculated. Knowing that the stages have 5, 10, 20, and 40 filters in the first, second, third and fourth stage respectively it has computed a total of $5 \cdot 97.947+$

$10 \cdot 15.901 + 20 \cdot 3.157 + 40 \cdot 1.002 = 751.965$ features from the non-face samples, while a “monolitic” classifier with only 40 filters has to compute $40 \cdot 97.947 = 3.917.880$ features over the same non-faces set.

```
Testing model...
Model OK!
Face nb:7237 NFace nb:97947
Results on iteration 0:
 FalseNegativeRate = 99.9958
 FalsePositiveRate = 0
 TotalTestError = 6.8808
 Face errors: 7237 Non face errors: 0
 Threshold = 1.95

...

Results on iteration 20:
 FalseNegativeRate = 0.994888
 FalsePositiveRate = 0.353252
 TotalTestError = 0.3974
 Face errors: 72 Non face errors: 346
 Threshold = 0.950001
Optimum obtained (TAR>=99%) for Threshold=0.950001
```

Bibliography

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

[2] Y. Rodriguez, F. Cardinaux, S. Bengio, and J. Mariéthoz. Estimating the quality of face localization for face verification. In *International Conference on Image Processing*, pages 581–584, 2004.

[3] M.-H. Yang, D. J. Kriegman, and N. Ahuja. Detecting faces in images: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(1):34–58, 2002.

[4] E. Hjelmas and B. K. Low. Face detection: a survey. *Computer Vision and Image Understanding*, 83(3):236–274, September 2001.

[5] T. Sauquet, S. Marcel, and Y. Rodriguez. Multiview face detection. Technical Report IDIAP-RR 05-49, IDIAP Research Institute, September 2005.

[6] T. Sakai, M. Nagao, and T. Kanade. Computer analysis and classification of photographs of human faces. In *Proceedings First USA-JAPAN Computer Conference*, pages 55–62, 1972.

[7] D. Marr and E. Hildreth. Theory of edge detection. Technical Report AIM-518, MIT Artificial Intelligence Laboratory, April 1979.

[8] C.-L. Huang and C.-W. Chen. Human facial feature extraction for face interpretation and recognition. *Pattern Recognition*, 25(12):1435–1444, 1992.

[9] J. Choi, S. Kim, and P. Rhee. Facial components segmentation for extracting facial features. In *Audio- and Video-Based Biometric Person Authentication*, March 1999.

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

[11] C. Kotropoulos and I. Pitas. Rule-based face detection in frontal views. In *IEEE International Conference on Acoustics, Speech and Signal Processing*, volume IV, pages 2537–2540, April 1997.

[12] X. Lv, J. Zhou, and C. Zhang. A novel algorithm for rotated human face detection. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 760–765, June 2000.

[13] M. Hunke and A. Waibel. Face locating and tracking for human-computer interaction. In *28th Annual Asimolar Conference on Signals, Systems and Computers*, volume 2, pages 1277–1281, August 1994.

[14] S. J. McKenna, S. Gong, and J. J. Collins. Face tracking and pose representation. In *British Machine Vision Conference*, pages 755–764, 1996.

[15] J. Yang and A. Waibel. A real-time face tracker. In *Workshop on Applications of Computer Vision*, pages 142–147, 1996.

[16] H. P. Graf, E. Cosatto, D. Gibbon, M. Kocheisen, and E. Petajan. Multimodal system for locating heads and faces. In *2nd International Conference on Automatic Face and Gesture Recognition*, pages 88–93, 1996.

[17] L. H. Koh, S. Ranganath, M. W. Lee, and Y. V. Venkatesh. An integrated face detection and recognition system. In *International Conference on Image Analysis and Processing*, page 532, September 1999.

[18] S. Kawato and J. Ohya. Real-time detection of nodding and head-shaking by directly detecting and tracking the 'between-eyes'. In *International Conference on Automatic Face and Gesture Recognition*, pages 40–45, 2000.

[19] C. H. Lee, J. S. Kim, and K. H. Park. Automatic human face location in a complex background using motion and color information. *Pattern Recognition*, 29(11):1877–1889, 1996.

[20] Y. Dai and Y. Nakano. Face-texture model based on SGLD and its application in face detection in a color scene. *Pattern Recognition*, 29(6):1007–1017, 1996.

[21] G. Wei and I. K. Sethi. Face detection for image annotation. *Pattern Recognition Letters*, 20(11-13):1313–1321, 1999.

[22] M. Abdel-Mottaleb and A. Elgammal. Face detection in complex environments from color images. In *International Conference on Image Processing*, pages 622–626, October 1999.

[23] F. Marqués and V. Vilaplana. A morphological approach for segmentation and tracking of human face. In *International Conference on Pattern Recognition*, pages 5064–5067, 2000.

[24] C. Chen and S. P. Chiang. Detection of human faces in color images. *IEE Proceedings on Vision Image and Signal Processing*, 144(6):384–388, December 1997.

[25] P. J. L. van Beek, M. J. T. Reinders, B. Sankur, and J. C. A. van der Lubbe. Semantic segmentation of videophone image sequences. In *Proceedings SPIE Visual Communications and Image Processing*, volume 1818, pages 1182–1193, November 1992.

[26] L. Yin and A. Basu. Integrating active face tracking with model based coding. *Pattern Recognition Letters*, 20(6):651–657, 1999.

[27] J. L. Crowley and F. Bérard. Multi-modal tracking of faces for video communications. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 640–645, 1997.

[28] S. J. Mckenna, S. Gong, and H. Liddell. Real-time tracking for an integrated face recognition system. In *Second European Workshop on Parallel Modelling of Neural Operators*, November 1995.

[29] D. Reisfeld and Y. Yeshurun. Robust detection of facial features by generalized symmetry. In *International Conference on Pattern Recognition*, pages I:117–120, 1992.

[30] C.-C. Lin and W.-C. Lin. Extracting facial features by an inhibitory mechanism based on gradient distributions. *Pattern Recognition*, 29(12):2079–2101, 1996.

[31] D. Reisfeld and Y. Yeshurun. Preprocessing of face images: Detection of features and pose normalization. *Computer Vision and Image Understanding*, 71(3):413–430, September 1998.

[32] S. Katahara and M. Aoki. Face parts extraction windows based on bilateral symmetry of gradient direction. In *8th International Conference on Computer Analysis of Images and Patterns*, volume 1689 of *Lecture Notes in Computer Science*, pages 489–497, September 1999.

[33] A. Tankus, Y. Yeshurun, and N. Intrator. Face detection by direct convexity estimation. *Pattern Recognition Letters*, 18(9):913–922, September 1997.

[34] I. Craw, H. Ellis, and J. R. Lishman. Automatic extraction of facial features. *Pattern Recognition Letters*, 5(2):183–187, February 1987.

[35] L. C. De Silva, K. Aizawa, and M. Hatori. Detection and tracking of facial features by using a facial feature model and deformable circular template. *IEICE Transactions on Information and Systems*, E78-D(9):1195–1207, September 1995.

[36] Y. Sumi and Y. Ohta. Detection of face orientation and facial components using distributed appearance modeling. In *International Workshop on Automatic Face and Gesture Recognition*, pages 254–259, June 1995.

[37] K. C. Yow and R. Cipolla. Feature-based human face detection. *Image and Vision Computing*, 15(9):713–735, 1997.

[38] C. Lin and K.-C. Fan. Triangle-based approach to the detection of human face. *Pattern Recognition*, 34(6):1271–1284, 2001.

[39] M. Kass. Snakes: Active contour models. *International Journal of Computer Vision*, 1(4):321–331, 1980. also in ICCV1, 1987.

[40] H. Wu, T. Yokoyama, D. Pramadihanto, and M. Yachida. Face and facial feature extraction from color image. In *2nd International Conference on Automatic Face and Gesture Recognition*, pages 345–350, 1996.

[41] T. Yokoyama, Y. Yagi, and M. Yachida. Facial contour extraction model. In *3rd International Conference on Automatic Face and Gesture Recognition*, pages 254–259, 1998.

[42] A. Nikolaidis and I. Pitas. Facial feature extraction and pose determination. *Pattern Recognition*, 33(11):1783–1791, November 2000.

[43] A. L. Yuille, P. W. Hallinan, and D. S. Cohen. Feature extraction from faces using deformable templates. *International Journal of Computer Vision*, 8(2):99–111, 1992.

[44] G. Chow and X. B. Li. Towards a system for automatic facial feature detection. *Pattern Recognition*, 26(12):1739–1755, December 1993.

[45] Y. H. Kwon and N. da Vitoria Lobo. Face detection using templates. In *International Conference on Pattern Recognition*, pages A:764–767, 1994.

[46] T. F. Cootes and C. J. Taylor. Active shape models: Smart snakes. In *British Machine Vision Conference*, pages 267–275, 1992.

[47] A. Lanitis, C. J. Taylor, and T. F. Cootes. Automatic tracking, coding and reconstruction of human faces using flexible appearance models. *Electronics Letters*, 30(19):1587–1588, September 1994.

[48] A. Lanitis, C. J. Taylor, and T. F. Cootes. Automatic interpretation and coding of face images using flexible models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(7):743–756, 1997.

[49] G. J. Edwards, C. J. Taylor, and T. F. Cootes. Learning to identify and track faces in image sequences. In *3rd International Conference on Automatic Face and Gesture Recognition*, pages 260–267, 1998.

[50] L. Sirovich and M. Kirby. Low dimensional procedure for the characterization of human faces. *Journal of the Optical Society of America*, 4(3):519–524, 1987.

[51] M. Turk and A. Pentland. Eigenfaces for recognition. *Journal of Cognitive Neuro Science*, 3(1):71–86, 1991.

[52] B. Moghaddam and A. Pentland. Face recognition using view-based and modular eigenspaces. In *Automatic Systems for the Identification and Inspection of Humans*, volume 2277, pages 12–21, 1994.

[53] A. W. Senior. Face and feature finding for a face recognition system. In *International Conference on Audio- and Video-Based Biometric Person Authentication*, pages 154–159, March 1999.

[54] S. Li and Q. Gu. Combining feature optimization into neural network based face detection. In *International Conference on Pattern Recognition*, pages 2814–2817, 2000.

[55] M. H. Yang, N. Ahuja, and D. J. Kriegman. Face detection using mixtures of linear subspaces. In *International Conference on Automatic Face and Gesture Recognition*, pages 70–76, March 2000.

[56] J. Mercer. Functions of positive and negative type, and their connection with the theory of integral equations. *Philosophical Transactions of the Royal Society of London. Series A, Containing Papers of a Mathematical or Physical Character*, 209:415–446, 1909.

[57] T. Kohonen. The self-organizing map. *Proceedings of the IEEE*, 78(9):1464–1480, September 1990.

[58] G. Burel and D. Carel. Detection and localization of faces on digital images. *Pattern Recognition Letters*, 15(10):963–967, 1994.

[59] M. B. E. Propp. An artificial neural network architecture for human face detection. Master's thesis, University of Nebraska–Lincoln, 1995.

[60] M. J. Jones and P. Viola. Fast multi-view face detection. Technical Report TR2003-96, Mitsubishi Electric Research Laboratories, August 2003.

[61] C. Garcia and M. Delakis. Convolutional face finder: A neural architecture for fast and robust face detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(11):1408–1423, 2004.

[62] J. Meynet, V. Popovici, and J.-P. Thiran. Face detection with mixtures of boosted discriminant features. Technical Report TR-ITS-2005.35, Ecole Polytechnique Fédérale de Lausanne (EPFL), Signal Processing Institute, November 2005.

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

[64] H. A. Rowley, S. Baluja, and T. Kanade. Neural network-based face detection. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 203–208, 1996.

[65] H. A. Rowley, S. Baluja, and T. Kanade. Rotation invariant neural network-based face detection. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 38–44, 1998.

[66] R. Féraud, O. Bernier, J.-E. Viallet, and M. Collobert. A fast and accurate face detector based on neural networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(1):42–53, 2001.

[67] R. Jacobs, M. Jordan, S. J. Nowlan, and G. E. Hinton. Adaptive mixtures of local experts. *Neural Computation*, 3:79–87, 1988.

[68] M.-H. Yang, D.n Roth, and N. Ahuja. A SNoW-based face detector. In Sara A. Solla, Todd K. Leen, and Klaus-Robert Müller, editors, *Neural Information Processing Systems Conference*, pages 862–868, 1999.

[69] N. Littlestone. Learning quickly when irrelevant attributes abound: A new linear-threshold algorithm. *Machine Learning*, 2(4):285–318, April 1988.

[70] S. Z. Li, Z. Zhang, H.-Y. Shum, and H. Zhang. Floatboost learning for classification. In Suzanna Becker, Sebastian Thrun, and Klaus Obermayer, editors, *Neural Information Processing Systems Conference*, pages 993–1000, 2002.

[71] B. Fröba and A. Ernst. Face detection with the modified census transform. In *International Conference on Automatic Face and Gesture Recognition*, pages 91–96, 2004.

[72] A. Colmenarez and T. S. Huang. Maximum likelihood face detection. In *2nd International Conference on Automatic Face and Gesture Recognition*, pages 307–311, 1996.

[73] M. S. Lew and N. Huijsmanns. Information theory and face detection. In *International Conference on Pattern Recognition*, volume 3, pages 601–605, 1996.

[74] A. Colmenarez and T. S. Huang. Face detection with information-based maximum discrimination. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 782–787, 1997.

[75] T. S. Huang and A. J. Colmenarez. Pattern detection with information-based maximum discrimination and error bootstrapping. In *International Conference on Pattern Recognition*, volume 1, pages 222–224, 1998.

[76] H. Schneiderman and T. Kanade. Probabilistic modeling of local appearance and spatial relationships for object recognition. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 45–51, 1998.

[77] H. Schneiderman and T. Kanade. A statistical method for 3D object detection applied to faces and cars. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 746–751, June 2000.

[78] L. Peotta, L. Granai, and P. Vandergheynst. Very low bit rate image coding using redundant dictionaries. In *Proceedings of the SPIE, Wavelets: Applications in Signal and Image Processing X*, volume 5207, pages 228–239, November 2003.

[79] Y. Freund and R. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, 55(1):119–139, 1997. Special Issue for EuroCOLT '95.

[80] Y. Freund and R. Schapire. A short introduction to boosting. *Journal of Japanese Society for Artificial Intelligence*, 14(5):771–780, 1999.

[81] N. Gourier, D. Hall, and J. L. Crowley. Estimating face orientation from robust detection of salient facial features. In *Proceedings of Pointing 2004*. International Workshop on Visual Observation of Deictic Gestures, 2004.

[82] T. Sim, S. Baker, and M. Bsat. The CMU pose, illumination, and expression (PIE) database. In *International Conference on Automatic Face and Gesture Recognition*, pages 53–58, 2002.

[83] P. J. Phillips, H. Wechsler, J. Huang, and P. J. Rauss. The FERET database and evaluation procedure for face-recognition algorithms. *Image and Vision Computing*, 16(5):295–306, April 1998.

[84] P. J. Phillips, H. J. Moon, S. A. Rizvi, and P. J. Rauss. The FERET evaluation methodology for face-recognition algorithms. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(10):1090–1104, October 2000.

[85] V. Popovici, J.-P. Thiran, Y. Rodriguez, and S. Marcel. On performance evaluation of face detection and localization algorithms. In *Proceedings of the 17th International Conference on Pattern Recognition*, volume 1, pages 313–317, 2004.

[86] N. V. Vapnik. *The Nature of Statistical Learning Theory.* Springer-Verlag, New York, 2000.