Face Localization Based on Affine Invariant Local Feature Face Models and Evidence Accumulation

Marc Tarrés Deulofeu

June 3, 2003

Acknowledgements

Ves per on, semblava que no arribaria mai el dia, però ja he acabat la carrera. En castellà diuen que "es de bien nacido ser agradecido", o sigui, que "qui és agraït, és un ben parit". Per tant, m'agradaria donar les gràcies a tothom qui ha fet possible la consecució d'aquest títol.

Primerament, gràcies pare i mare. Gràcies per moltes coses: per animar-me a fer telecos, per aguantar-me la mala baba que no us mereixeu... Però sobretot, perquè si no fós per vosaltres, jo no seria aquí. Aquests agraïments són evidentment extensius a la resta de la família: germaneta, avis, àvies, oncles, ties i cosins.

Je voudrais remercier aussi Jean-Philippe, pour m'avoir donné l'opportunité de venir à l'EPFL et vivre une expérience si enrichissante.

And of course, thank you Vlad, for assisting me during all these months and never losing your temper. It's been a pleasure working with you.

Per res del món no voldria oblidar-me de la colla: merci, Toni i Xevi, pel suport tècnic; merci, Pitu i Iris, pel suport moral; merci, Sergi, per atendre les consultes *tècniques* de ma mare mentre feia els seus primers *pinitos* informàtics. I merci a la resta de la colla, perquè d'una manera o altra, i tal com diu en Pitu, tots formeu part de *la meva gent*.

I si bé tothom diu que fer un Erasmus serveix per fer currículum i ampliar els teus horitzons formatius i professionals, la veritat és que al cap dels anys, de l'únic que te'n recordes és de la gent amb qui has compartit l'experiència. Són molts els que he conegut durant aquests nou mesos, i hi ha hagut temps de fer moltes coses: un *Macumbazo*, una manifestació contra la guerra, una volta al llac Léman en bicicleta, patir les conseqüències d'una cimera del G-8, i a estones lliures, un projecte de final de carrera. En aquest apartat, cal esmentar els membres de La Comuna, la comunitat Falesiana, tots els borinots, carallots i d'altres projectistes, *assistants* i estudiants de l'EPFL i de l'UNIL. Una menció especial a tots aquells amb qui he compartit laboratori, estrès, nervis, paranoies, alegries i *fondues*: Alessandro, Álvaro, Cristian, David, Emilio, Gianluca, Irene, Issa, Iván, Julien, Mei, Naara, Vanessa i Yin.

Finalment, *last but not least*, un petit record per tots aquells amb qui he compartit alguna altra part de la meva vida. Ja siguin companys i professors del col.legi i de l'institut, col·laboradors de Ràdio Bisbal, col·legues del Penyafort, companys de feina (becaris, fixos o ETT's) o bohemis propietaris del Pou Dolç, tots heu col.laborat a fer de mi el que sóc. Si heu fet una bona feina o no, això ja no ho entraré a valorar...

Abstract

In this project, a method for detecting and localizing human faces based on affine invariant local feature face models and evidence accumulation has been developed. It can be classified midway between template-based and pure feature-based methods, so it benefits from the advantages of both approaches.

First, local parts of the face are detected and classified using a template-based technique, thus obtaining robustness against illumination changes. Then, the attention is focused on the spatial configuration of these local features. To be able to deal with variation in pose and facial expression, the geometrical arrangement of the local features is studied and classified in an affine invariant space, where all differences due to affine transformations (rotation, scaling, translation, squeezing...) are removed.

Finally, the system is also able to cope with partial occlusions of the face by not requiring the detection of the full set of facial features in order to form a hypothesis and check it. Using evidence accumulation, it is possible to detect a face with some missing features, if the detected features provide enough evidence. Also, different face models have been built, taking into consideration all possible adverse cases: no mouth features, no nose features, or no eye features detected.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Context and Motivation

The problem of automatic face recognition is a complex and composite task. It involves detection and localization of faces in a cluttered background, normalization, and recognition or verification. Depending on the nature of the application, e.g. sizes of training and testing datasets, resolution of the images, variability in pose, lighting and background, noise, occlusion, and finally speed requirements, some of the subtasks could be very challenging.

However, while the problem of face recognition has received a lot of attention in the last decade, and there are good methods for face recognition relying on normalized faces, the problem of correct localization of faces is still open, especially in the cases where a greater variability in data is allowed. As the practice has proved, the face detection and localization step is determinant for subsequent processing. Even though human beings seem capable of being able to perform this task effortlessly, it is not easy in computer terms. It is this stage that has been addressed, in the context of high resolution grayscale images with variability in pose, background and lightning conditions and with possible occlusions due to different hairstyles, glasses, facial hair, etc.

1.2 Methods Overview

Face detection methods can be mainly divided into two categories: *template-based* (or *image-based*) and *feature-based* detection. In the first category, a subset of the image is classified to determine whether there is a human face at that location or not. The face detection problem is treated as a pattern recognition one, ranging from simple template matching to more sophisticated approaches such as *wavelet decomposition*. In the second class, faces are detected by grouping facial features according to their geometric configuration in a face model.

1.3 Proposed Approach

The presented method can be placed midway between the template-based methods and pure feature-based techniques. While it is based on features, there are also two levels of templates. One is at feature level — each feature is treated as a whole object — and the other is at face level, where is the face template — a flexible template of features. It is expected that combining both techniques, the method will benefit from the advantages specific to its ancestors: robustness against illumination changes (like in template-based methods) and against pose variation (like in feature-based approaches). Moreover, relying on features as a first stage of the algorithm will eliminate the need of a multiscale sliding window search that is specific to template-based methods. In this way, it is expected to have a direct way of estimating the scale and position of the face. The features detected in this stage are valuable information for face recognition.

1.4 Organization

The rest of this report is organized as follows: In Chapter 2, both approaches to face detection (template-based and feature-based methods) are presented in detail and their advantages and drawbacks are discussed. Chapter 3 contains the basis of the approach, while Chapter 4 explains its implementation and results. Finally, a critical discussion about the method, conclusions, and future work can be found in Chapter 5.

Chapter 2

Existing Face Detection Methods

As mentioned before, existing face detection approaches reported in literature can be broadly divided into template-based and feature-based methods. Next sections present in detail each one of these approaches.

2.1 Template-based Detection

These methods treat the face as a whole feature (holistic approach). They all consist of a strategy for exploring the image and a classifier which decides if a window of the image contains a face or not. The most commonly employed strategy is to explore the image with a sliding window, at different scales and, eventually, at different orientations. However, template-based systems vary in their implementation depending mainly on the need for computational efficiency. The size of the scanning window, the subsampling rate and the step size are typical parameters that can be tuned to obtain the desired performance.

Early methods used templates predefined by experts. Decision was made based on correlation values. This simple template-matching systems were easy to implement, but they could not effectively deal with variation in scale, pose and shape.

More recent and complex methods make use of learning algorithms to extract the relevant characteristics from example images. These learned characteristics usually are probability distributions or discriminant functions that are further used for classification. In a general object detection scheme, let $\Omega = \{\omega_1, \omega_2, ..., \omega_M\}$ be the set of object classes. These methods estimate the *likelihood* $P(\mathbf{x} \mid \omega_c)$, $c = 1, ..., M$, for $\mathbf{x}$ to belong to the object class ω_c, using a training set of N_T patterns $\{\mathbf{x}^t\}_{t=1}^{N_T}$, each pattern belonging to one of the classes in Ω. In face detection $\Omega = \{f, nf\}$, that is the object classes are

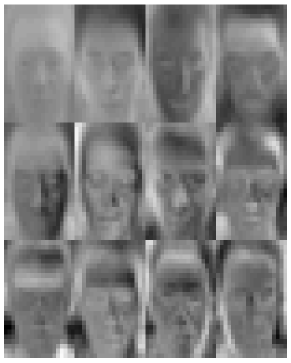


Figure 2.1: Some examples of eigenfaces.

restricted to: face and non-face. To classify a new pattern $\mathbf{z}$, the maximum likelihood principle is often used:

$$\omega^* = \arg\max_{\omega_c} P\left(\mathbf{z}\left|\omega_c\right.\right) \tag{2.1}$$

2.1.1 PCA Applied to Face Detection: Eigenfaces

As images can be regarded as high-dimensional vectors, and human faces lie in a subspace of the overall image space, dimension reduction techniques have been used for efficiency's sake. One of the most successfully used is *Principal Component Analysis* (PCA), first applied to the problem of face representation by Sirovich and Kirby [50], and later on by Turk and Pentland [57] and Moghaddam and Pentland [30] to detect and recognize human faces. The main idea behind this approach is to estimate the *likelihood* functions in a low-dimensionality space, but keeping most of the *information* (i.e. the in-class variability of human faces).

Given a set of m-by-n images $\{I^t\}_{t=1}^{N_T}$, a training set of vectors $\{\mathbf{x}^t\}$, where $\mathbf{x} \in R^{N=mn}$, can be formed by lexicographic ordering of the pixel elements of each image I^t. The basis functions in a Karhunen-Loève Transform (KLT) [27] are obtained by solving the eigenvalue problem

$$\Lambda = \Phi^T \Sigma \Phi \tag{2.2}$$

where Σ is the covariance matrix of the data, Φ is the eigenvector matrix of Σ and Λ is the corresponding diagonal matrix of eigenvalues. In PCA, a partial KLT is performed to identify the largest-eigenvalue eigenvectors and obtain a principal component feature vector $y = \Phi_M^T \tilde{\mathbf{x}}$, where $\tilde{\mathbf{x}} = \mathbf{x} - \overline{\mathbf{x}}$ is the

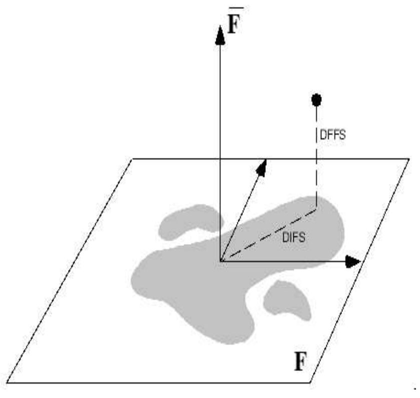


Figure 2.2: The principal subspace F and its orthogonal complement $\overline{F}$.

mean-normalized image vector $\left(\overline{\mathbf{x}} = \frac{1}{N_T}\sum_{t=1}^{N_T}\mathbf{x}^t\right)$ and Φ_M is a sub-matrix of Φ containing the principal eigenvectors. These eigenvectors are the so-called *eigenfaces*, the look of which can be seen in Fig. 2.1.

PCA can be seen as a linear transformation $y = T(\mathbf{x}) : R^N \rightarrow R^M$ which extracts a lower-dimensional subspace of the KL basis corresponding to the maximal eigenvalues. This corresponds to an orthogonal decomposition of the vector space R^N into two mutually exclusive and complementary subspaces: the principal subspace (or feature space) $F = \{\Phi_i\}_{i=1}^{M}$, containing the principal components, and its orthogonal complement $\overline{F} = \{\Phi_i\}_{i=M+1}^{N}$, as illustrated in Fig. 2.2.

In a partial KL expansion, the residual reconstruction error is defined as

$$\epsilon^2(\mathbf{x}) = \sum_{i=M+1}^{N} y_i^2 = \|\tilde{\mathbf{x}}\| - \sum_{i=1}^{M} y_i^2 \tag{2.3}$$

and can be easily computed from the first M principal components and the L_2-norm of the mean-normalized image $\tilde{\mathbf{x}}$. Consequently the L_2 norm of every element $\mathbf{x} \in R^N$ can be decomposed in terms of its projections in these two subspaces. The component in the orthogonal subspace $\overline{F}$ is referred to as the "distance-from-feature-space" (DFFS), which is a simple Euclidean distance and is equivalent to the residual error $\epsilon^2(\mathbf{x})$ in Eq. 2.3. The component of $\mathbf{x}$ which lies *in* the feature space F is called the "distance-in-feature-space" (DIFS), which is not a distance-based norm, but can be interpreted in terms of the probability distribution of y in F.

Assuming that $P_F(\mathbf{x}\,|\omega_c)$ can be approximated as a high-dimensional Gaussian density, and that the mean $\overline{\mathbf{x}}$ and covariance Σ of the distribution from the training set $\{\mathbf{x}^t\}$ have been estimated, the likelihood estimate of an input pattern $\mathbf{x}$ can be expressed as the product of two independent Gaussian densities

$$\widehat{P}\left(\mathbf{x}\,|\omega_c\right) = P_F\left(\mathbf{x}\,|\omega_c\right)\widehat{P}_{\overline{F}}\left(\mathbf{x}\,|\omega_c\right) \tag{2.4}$$

where $P_F\left(\mathbf{x}\,|\omega_c\right)$ is the true marginal density in F-space and $\widehat{P}_{\overline{F}}\left(\mathbf{x}\,|\omega_c\right)$ is the estimated marginal density in the orthogonal complement $\overline{F}$-space. In case $P_F\left(\mathbf{x}\,|\omega_c\right)$ cannot be adequately modelled using a single Gaussian, a Mixture-of-Gaussians model can be used.

To detect faces, the maximum-likelihood principle (Eq. 2.1) is used. The density estimation $\widehat{P}\left(\mathbf{x}\,|\omega_c\right)$ is computed for each image vector $\mathbf{x}$ at location (i,j) and the class ω_c is determined. See [30] for details.

This method has been reported to achieve a good performance in large test sets. One of its advantages is that it can be applied to detect other objects such as eyes, nose and mouth, as it has been done by Pentland *et al.* [29, 34]. However, a disadvantage of this system (and indeed, of all template-based systems) is that each window has to be projected into a subspace before classification. This involves a matrix multiplication for every image window and the time spent is considerable.

2.1.2 Other Dimension-reduction Techniques

Sung and Poggio [54, 56] proposed an approach very close to the idea behind *eigenfaces*. Their system consists on two components, distribution-based models for face/non-face patterns and a multilayer perceptron classifier. First, a modified version of the classical k-means clustering algorithms is applied to the training set of face patterns and non-face patterns to divide each set into 6 clusters. The non-face patterns are generated in a "bootstrap" fashion, avoiding the problem of explicitly collecting a representative sample of non-face patterns (moreover, using the "bootstrap" method only those non-face patterns close to face clusters are collected and modelled). The result of this process is 6 face centroids and 6 non-face centroids together with their covariance matrices. Two distance metrics are computed between an input image pattern and the prototype clusters. The first distance component is the normalized *Mahalanobis* distance between the test pattern and the cluster centroid, measured within a lower-dimensional subspace spanned by the cluster's 75 largest eigenvectors. The second distance component is the Euclidean distance between the test pattern and its projection onto the 75–dimensional subspace. The last step is to use a multilayer perceptron (MLP) network to classify face window patterns from non-face patterns using the twelve pairs of distances to each face and non-face cluster. The system seems to perform quite well on a well-known test set such as the CMU. However, the number of face and non-face clusters are chosen quite arbitrary. It is not clear how the system would perform for a different number of clusters, and there is no general rule for selecting these parameters.

Yang *et al.* [61] proposed two methods using a mixture of linear subspaces. The first one uses common Factor Analysis (FA), which is a statistical method for modelling the covariance structure of high dimensional data using a small number of latent variables. FA assumes that the variance of a single variable can be decomposed into common variance and unique variance. Unlike PCA, which considers the total variance of all variables, FA analyzes only the common variance of the observed variables, thus avoiding "unwanted" variations, such as independent noise in the data. A mixture of factor analyzers in this first method. Given a set of training images, the Expectation-Maximization algorithm [8] is used to estimate the parameters in the mixture model. This model is then applied to sub-windows in the input image and outputs the probability of a face being present at the current location.

The second method uses Fisher Linear Discriminant (FLD) [9] to project samples from the high dimensional image space to a lower dimensional feature space. The training face and non-face samples are decomposed into several subclasses using Kohonen's Self Organizing Map (SOM) [21]. A projection matrix is then determined by Fisher Linear Discriminant which maximizes the ratio between the between-class variance and within-class variance. The whole training set is projected onto this subspace and Gaussian distributions are used to model each class conditional density. Parameters of the model are estimated with maximum likelihood. New patterns **z** are also classified by using the maximum likelihood principle as in Eq. 2.1. Good results have been reported for both methods. Nevertheless, important parameters such as the number of clusters for the face and non-face class in the second method are much dependent on the size of the training set. As in [54], generic rules for selecting those parameters are not known.

Schneiderman and Kanade [49] describe a face detector based on a combination of PCA and Bayes' decision rule (Eq. 2.5, in the form of a likelihood ratio).

$$\frac{P\left(\mathbf{x}\left|\omega = face\right.\right)}{P\left(\mathbf{x}\left|\omega = non - face\right.\right)} > \frac{P\left(\omega = non - face\right)}{P\left(\omega = face\right)} \tag{2.5}$$

The joint probability $P\left(\mathbf{x}\left|\omega\right.\right)$ is estimated based on local appearance and position of face patterns (subregions of the face) at multiple resolutions using a naive Bayes classifier (i.e., no statistical dependency between the subregions). This is due to two main reasons. First, it provides better estimation of the conditional density functions of these subregions. Second, a naive Bayes classifier provides a functional form of the posterior probability to capture the joint statistics of local appearance and position on the object. At each scale, a face image is decomposed into four rectangular subregions. These subregions are then projected to a lower dimensional space using PCA and quantized into a finite set of patterns, and the statistics of each projected subregion are estimated from the projected samples to encode local appearance. Under this formulation, the method decides that a face is present when the likelihood ratio is larger than the ratio of prior probabilities (Eq. 2.5). This method shows a good performance and is able to detect some rotated and profile faces.

2.1.3 Other Template-based Methods

Many other template-based techniques have been proposed for classifying image patches as *face* or *non-face.* For example, *Neural Networks* are quite a popular technique. They have been implemented by, among others, Rowley *et al.* [42, 43]. Their system incorporates face knowledge in a retinally connected neural network. The input fixed-size windows are preprocessed through lighting correction (a best fit linear function is subtracted) and histogram equalization. After that, the windows are sub-scanned by a hidden layer of the network. These different observations are combined to produce a final output. The typical problem of multiple detections about window-scanning techniques is tackled here through two heuristics: 1) if the number of detections in a small neighborhood surrounding the current location is above a certain threshold, a face is considered to be present at this location, and 2) when a region is classified as a face according to the previous rule, then overlapping detections are rejected as false positives. This is a robust system, but is restricted to detecting frontal faces only. In order to improve this, the method has been extended to detect rotated faces. This has been achieved by using a router network which processes each input window to determine the possible face orientation and then rotates the window to canonical orientation. The rotated window is then fed to the neural networks as described above. The detection rate for frontal faces is degraded, but a good detection result is reported for detecting faces of different orientations. Nevertheless, the pose variation problem is not considered.

A new learning architecture called *SNoW* (Sparse Network of Winnows) is applied to face detection in Roth *et al.* [41]. It is a technique specifically tailored for learning in domains in which the potential number of features taking part in decisions is very large. SNoW for face detection is a neural network consisting of two linear threshold units (LTU), representing the classes of faces and non-faces. These LTUs are separate from each other and are sparsely connected over their operating Boolean feature space. They are updated following the efficient Winnow updating rule [26], which promotes and demotes weights in cases of misclassification. As in Sung and Poggio's method, training is performed with the bootstrap learning algorithm, and the images are preprocessed with the same procedure used in to previous approach. This method gives a great frontal face detection result.

Support Vector Machines (SVMs) have also been used in face detection. SVMs can be considered as a new paradigm to train polynomial function, neural networks, or radial basis function (RBF) classifiers. Unlike most classifier-training methods, which are based on minimizing the training error, i.e., the *empirical risk*, SVMs aim to minimize an upper bound on the expected generalization error, following the *structural risk minimization* principle. An SVM classifier is a linear one, where the separating hyperplane is chosen to minimize the expected classification error of the unseen test patterns. This optimal hyperplane is defined by a weighted combination of a small subset of the training vectors, called *support vectors.* The classifier is trained using a subset of the original data set, which is iteratively updated. An optimal condition and a strategy for improvement sis specified so that after each iteration the system can decide if it has reached the optimal solution and define a way to improve the cost function if it has not. Osuna *et al.* [32] developed an efficient method to train an SVM for large scale problems, and applied it to face detection. In their method, an SVM with a 2nd–degree polynomial as a kernel function is trained with a decomposition algorithm which guarantees global optimality. Similar to the previously mentioned methods, Osuna *et al.* also use the bootstrap method for generating training samples and preprocess all images with histogram equalization. The detection accuracy is comparable to that of other best methods.

Hidden Markov Model (HMM) is another face detection method reported in literature. The underlying assumption of the HMM is that patterns can be characterized as a parametric random process and that the parameters of this process can be estimated in a precise, well-defined manner. In developing an HMM for a pattern recognition problem, a number of hidden states need to be decided first to form a model. Then, one can train HMM to learn the transitional probability between states from the examples where each example is represented as a sequence of observations. The goal of training an HMM is to maximize the probability of observing the training data by adjusting the parameters in an HMM model with the standard Viterbi segmentation method and Baum-Welch algorithms [39]. After the HMM has been trained, the output probability of an observation determines the class to which it belongs. Regarding the face detection problem, a face pattern can be divided into several regions such as the forehead, eyes, nose, mouth, and chin. A face pattern can then be recognized by a process in which these regions are observed in an appropriate order (e.g., from top to bottom and left to right). Instead of relying on accurate alignment as in the other template-based methods, this approach aims to associate facial regions with the states of a continuous density Hidden Markov Model. HMM-based methods usually treat a face pattern as a sequence of observation vectors where each vector is a strip of pixels. The boundaries between strips of pixels are represented by probabilistic transitions between states, and the image data within a region is modelled by a multivariate Gaussian distribution. An observation sequence consists of all intensity values from each block. The output states correspond to the classes to which the observations belong. As it has already been said, once the HMM has been trained, the output probability of an observation determines the class to which it belongs (face or non-face). Many authors have used this approach when facing the face detection problem, e.g., Samaria and Young [46], Nefian and Hayes [31], and Rajagopalan *et al.* [40]. Experimental results show that this approach has a higher detection rate than *Neural Networks*, although it also has more false alarms.

Schneiderman and Kanade later extended their method (commented in Section 2.1.2, p. 14) with wavelet representations to detect profile faces and cars [48]. A wavelet transform can capture information regarding visual attributes in space, frequency, and orientation and thus should be well suited for describing the characteristics of the human face. When tested with frontal faces this system is outperformed by the Bayesian-PCA approach. However, it shows its best when tested with profile faces. In that case, it performs far better than the other system.

Many other approaches have been proposed, some of them based on *Information Theory*. These techniques profit in the contextual constraint. In the case of a face pattern, this contextual constraint is specified by a small neighborhood of pixels. Context-dependent entities such as image pixels are modelled using *Markov Random Field* (MRF) distributions, thus characterizing mutual influences among them. Alternatively, the face and non-face distributions can be estimated using histograms. Using Kullback relative information [7], the Markov process that maximizes the information-based discrimination between the two classes can be found and applied to detection [6, 25].

Finally, some systems make use of *Inductive Learning* algorithms. For example, Huang *et al.* applied Quinlan's C4.5 algorithm [38] to learn a decision tree from positive and negative examples of face patterns [15]. From these examples, C4.5 builds a classifier (the decision tree), whose leaves indicate class identity and whose nodes specify tests to perform on a single attribute. This systems is reported to perform well in a set of frontal face images.

2.2 Feature-based Detection

As it has been seen when presenting template-based methods, they are very sensitive to both rotation and scaling. To detect faces of different scales, the input image or the template must be resized to an appropriate value so that classification or matching can be carried out. The problem of in-plane and out-of-plane (or pose variation) rotation, however, is more serious. In order to detect rotated faces, another training set of faces of different views and rotations may be required, or a completely new set of rules must be devised. Feature-based methods offer a sensible solution to both rotation and scaling problems.

These methods can be divided into two groups. The first one is a bottom-up approach in which facial features are detected and grouped according to their geometric relationships. The other approach is top-down, where facial regions are first located, and further analyzed in more detail to confirm the hypotheses.

2.2.1 Bottom-up Methods

The first approaches proposed in this category were based on simple edge detection. In these approaches, edges need to be labeled and matched to a face model in order to verify correct detections. Govindaraju *et al.* [10] accomplished this by labeling edges as the left side, hairline, or right side of a front view face and matching them against the face model using a cost function, which uses the *golden ratio* ($\frac{height}{width} = \frac{1+\sqrt{5}}{2}$). A group of features with a cost less than a predefined threshold formed an hypothesized face candidate. Hypotheses are tested in a second stage, using the eyes and symmetry about the vertical axis, to verify the detection. This method is quite limited since it assumes faces are upright, unoccluded, and frontal. Moreover, because of the variability of shape between different people, it is hard to devise a reliable cost function.

More recent methods propose to detect full salient facial features, such as eyes or noses, instead of simple edges. The feature detection can be performed, for example, using any of the methods presented in Section 2.1. The detection of prominent features then allows for the existence of other less salient features to be hypothesized using anthropometric measurements of face geometry. In [19], Jeng *et al.* proposed a system based on anthropometric measures. They initially try to locate the eyes in a binarized preprocessed image. For each possible eye pair the algorithm goes on to search for a nose, a mouth, and eyebrows. Each feature has its associated evaluation function. These functions outcomes are weighted by their facial importance with manually selected coefficients, as shown in Eq. 2.6. The hypothesis obtaining the highest score is determined to be the most likely face candidate. This system is reported to obtain a good detection ratio on a test-set with a cluttered background and subjects positioned in various directions.

$$E = 0.5E_{eye} + 0.2E_{mouth} + 0.1E_{Reyebrow} + 0.1E_{Leyebrow} + 0.1E_{nose} \tag{2.6}$$

Leung *et al.* developed a probabilistic method to locate a face in a cluttered scene based on local feature detectors and random graph matching [23]. First, features such as eyes, nose and nostrils are identified by convolving the input image with a set of Gaussian derivative filters at different scales and orientations. A vector of filter responses at a particular spatial location is then matched against a template vector response. A feature is detected at a location if the degree of matching is above a threshold. The top two feature candidates with the strongest response are selected to search for the other facial features. Having selected these features, the expected locations of the other features are estimated using a statistical model of mutual distances. Constellations are formed only from candidates that lie inside the appropriate locations, and the most face-like constellation is determined. Finding the best constellation is formulated as a random graph matching problem in which the nodes of the graph correspond to features on a face, and the arcs represent the distances between different features. Ranking of constellations is based on a *Maximum Likelihood* (ML) scheme, i.e. the probability density function that a constellation corresponds to a face versus the probability it was generated by an alternative mechanism (i.e., non-face). A drawback of this method is that since the distances between facial features are used in the matching, it can accommodate only faces with little rotation in depth.

Instead of using mutual distances to describe the relationships between facial features in constellations, an alternative method for modelling faces was also proposed by Burl, Leung *et al.* [4, 24]. The representation and ranking of the constellations is accomplished using the statistical theory of shape, developed by Kendall [20] and Dryden & Mardia [28]. The shape statistics is a joint probability density function over N feature points, represented by (x_i, y_i), for the i–th feature, under the assumption that the original feature points are positioned in the plane according to a general $2N$–dimensional Gaussian distribution. Also, instead of detecting the features using Gaussian derivative filters, features are found using the *orientation template correlation* (OTC). OTC is a technique which operates on the *orientation map* of an image, a representation introduced by G.H. Granlund in [11]. The main advantage of OTC is its independence from illumination, unlike other correlation methods. Finally, they apply the same ML method to determine the location of a face. These methods are reported to be able to deal with partially occluded faces, although no details are given on how this is accomplished.

In [64, 66], Yow and Cipolla presented a feature based method that uses a large amount of evidence from the visual image and their contextual evidence. The first stage applies elongated Gaussian derivative filters to detect interest points. The second step examines the edges around these interest points and groups them into regions, according to proximity and similar orientation between them. Those points that have roughly parallel edges on both sizes are selected. They are then classified into different feature classes (eyebrow, eye, nose and mouth) based on measurements such as edge length and edge strength. An image region becomes a valid facial feature candidate if the Mahalanobis distance between the candidate and the corresponding feature model is below a threshold. The features are now grouped based on the face model. Each facial feature and grouping is then evaluated using a Bayesian network. The overall detection rate of this method is quite high and it can detect faces at different orientations and poses. However, the reported false detection rate is also quite high and the implementation is only effective for faces larger than 60×60 pixels. Subsequently, this approach has been enhanced with *active contour models* [65, 5].

There are many other proposed methods belonging to the bottom-up feature-detection scheme. Just briefly mention Han *et al.* and their morphology-based technique to extract what they call eye-analogue segments (defined as edges on the contours of eyes) for face detection. Recently, Amit *et al.* presented a method based on focusing and intensive classification. Focusing is based on spatial arrangements of edge fragments extracted using intensity difference. Each region of interest is then classified as face or background using a learned CART tree.

Finally, even though it has been said that color information is used mainly in top-down approaches, there are some bottom-up methods that make use of it. For example, Jebara *et al.* [18] segment skin color regions in the image and the feature detection is carried out by using a symmetry transformation. The detected features are tracked in a sequence and a 3D structure of the face can be constructed. Sun *et al.* [53] also use local symmetry information. A local symmetry map is obtained and fused with the color map which present the skin color likeness of each pixel. Facial features correspond with the local maxima of the map resulted. Theses features are then grouped according to a face geometry model. Both methods are based on the assumption that facial regions can be segmented from the background reliably using color information only. This assumption limits their applicability.

2.2.2 Top-down Methods

As said before, in this class of methods, face regions are segmented from the background using color information. Face candidates are formed from these regions and further verified. These methods are very fast, faces can be detected in real time. However, no method seems to be able to resolve the problem of background objects having skin color, especially when they are merged with the face region. Also, these methods are very sensitive to varying lighting conditions.

Sobottka and Pitas proposed a method for face localization and facial feature extraction using shape and color [51]. After segmenting the skin-like regions, a growing algorithm is applied at a coarse resolution in order to determine connected components. For each component, the best fit ellipse is computed using geometric moments. Those ellipses that are good approximations of the connected components are selected and considered as face candidates. Subsequently, these candidates are verified by searching for facial features inside of the connected components. Features, such as eyes and mouths, are extracted based on the observation that they are darker than the rest of the face.

Saber and Tekalp [44] propose a similar method. They segment skin color regions using Gibbs Random Field filtering. Next, an elliptical face template is used to determine the similarity of the skin color regions based on Hausdorff distance [16]. Feature detection is then carried out inside the ellipses. The eyes centers are localized using several cost functions which are designed to take advantage of the inherent symmetries associated with face and eye locations. The tip of the nose and the mouth are located based on the location of the eyes. One clear drawback of this system is that it is only effective for a single frontal view face and when both eyes are visible.

In contrast to pixel-based methods, a detection method based on structure, color and geometry was proposed by Yang and Ahuja in [60]. First, multiscale segmentation is performed to extract homogeneous regions in an image. At each scale, a Gaussian skin color model is used to extract regions of skin tone. These regions are merged until the shape of the total region is approximately elliptic. The goodness of the approximation is based on the number of pixels of the region inside its elliptic shape. Candidate regions with dark areas or holes inside are considered human faces based on the observation that facial features either do not have skin color or are darker than their surrounding areas. Experimental results show that this method is able to detect faces at different orientations with facial features such as beard and glasses.

Instead of merging, Wei and Sethi [59] use an iterative partitioning of the human skin region to detect faces. First, skin color classification at each pixel location is performed, thus obtaining a binary image. Again, regions that can be approximated well by an ellipse are considered face candidates. The iterative process goes on further partitioning the rejected regions, and tries to approximate them by ellipses. The partitioning process stops when all subregions are too small and no face candidate is found. Faces are detected by verifying features in the face candidate regions. After applying a histogram-based thresholding, facial features should correspond to the dark parts. Based on relative positions of these dark parts and their shapes within a face region, a face region is classified as face or not.

There are some other methods that do not rely on color information. Yang and Huang use a hierarchical knowledge-based method to detect faces in grayscale images [63]. Their system consists of three levels of rules. The rules at a higher level are general descriptions of what a face looks like while the rules at lower levels rely on details of facial features. A multiresolution hierarchy of images is created by averaging and subsampling. The lowest resolution (Level 1) image is searched looking for uniform regions. The face candidates are further processed at finer resolutions. At Level 2, local histogram equalization is performed, followed by edge detection. Surviving candidate regions are then examined at Level 3 with another set of rules that respond to facial features such as the eyes and mouth. One attractive feature of this method is the low required computation, achieved by using a *coarse-to-fine* or *focus-of-attention* strategy. Although it does not result in a high detection rate, the ideas of using a multiresolution hierarchy and rules to guide searches have been used in later face detection works.

Kotropoulos and Pitas [22] presented a method which is an extension of the algorithm presented above. First, facial features are located with a projection method. Let $I(x, y)$ be the intensity value of an $m \times n$ image at position (x, y), the horizontal and vertical projections of the image are defined as $HI(x) = \sum_{y=1}^{n} I(x, y)$ and $VI(y) = \sum_{x=1}^{m} I(x, y)$. The horizontal profile of an input image is obtained first, and then the two local minima, determined by detecting abrupt changes in *HI*, are said to correspond to the left and right side of the head. Similarly, the vertical profile is obtained and the local minima are determined by the locations of mouth lips, nose tip, and eyes. These detected features constitute a facial candidate. Subsequently, eyebrow/eyes, nostrils/nose, and the mouth detection rules are used to validate these candidates. An acceptable detection rate is achieved, but the system has some drawbacks: it becomes difficult to locate a face in a complex background, and it cannot readily detect multiple faces.

2.3 Hybrid Methods

Even though most of the existing face detection methods belong to one of the two groups of approaches presented above, there are some of them that could be included in both categories. These are methods that make use of local features which are, in fact, image templates. These *feature-templates* are then used for face detection, either via their geometrical arrangement or by matching them on the image. As examples of these *hybrid methods*, the works of Heisele *et al.* [13], and Viola and Jones [58] can be cited.

In the first one, they independently detect local parts of the face, arguing that for small rotations, the changes in the components are relatively small compared to the changes in the whole face pattern. Changes in the 2–D locations of the components due to pose changes are accounted for by a learned, flexible face model. Then, a geometrical configuration classifier performs the final face detection by combining the results of the component classifiers.

Viola and Jones, on the other hand, do not use facial features such as eyes or nostrils, but rectangle features. More exactly, the features consist on the difference between the sum of the pixels within two or more rectangular regions. Evaluating these features could be done at any scale and location in a few operations. They accomplish this by using an image representation called *integral image*. Briefly, the integral image at location x, y contains the sum of the pixels above and to the left of x, y, inclusive:

$$ii\left(x,y\right) = \sum_{x' \le x, y' \le y} i\left(x', y'\right) \tag{2.7}$$

where $ii\left(x,y\right)$ is the integral image and $i\left(x,y\right)$ is the original image. Using a recurrent method, the integral image can be computed with a single pass over the input image. Working with this representation, any rectangular sum can be computed with four values of the integral image (i.e., the values of the integral image at the corners of the rectangle of interest). Thus, the difference between two rectangular sums can be computed with eight references to the integral image. Finally, the face detection is carried out by applying a cascade of weak classifiers, each one of them evaluating a rectangle feature. These features are learned using the AdaBoost learning algorithm, aiming to reject the highest possible number of false alarms in the earlier stages of the cascade. In this way, a nearly real-time detector is achieved.

2.4 Discussion

Having presented most of the existing face detection approaches, the ground has been set to discuss and compare them all. Before this, it should be stated that the way the methods have been divided (template-based *vs.* feature-based) is not the only possible one. Furthermore, for example, feature-based approaches could be divided into *low-level analysis* (edges, gray-levels, color...) and *feature analysis* (eyes, nose...), instead of into *bottom-up* and *top-down*.

Many problems arise when it comes to compare different face detection methods. First, the reported results commented above differ on their training sets and tuning parameters (in fact, most of the methods are tested on data sets with a very small number of images). Moreover, a general consensus about what a "correct detection" is, is far from being reached among the research community – and the fact that many authors do not give any detail on how have they obtained their reported results, is not helping in this sense.

The second factor is the training time and execution time, which are often ignored. Third, the number of scanning windows in template-based methods vary because they are designed to operate in different environments (i.e., to detect faces within a determined range). For example, it seems clear that if a method scans more windows than another one, it will obtain a higher number of false alarms. So, it is important to report the system's ROC curve (the correct detections/false positives ratio). Finally, the evaluation criteria should take into consideration the final purpose of the detector. It all depends on the misclassification costs. In some applications, having a false alarm is not so crucial as missing a face (though it requires a higher computational efficiency). For example, in a validation/control access system, it is unlikely that a false alarm would be identified as an individual of the database, so the criterion should be set to minimize the false negative detections.

In order to compare methods fairly, a few benchmark data sets have been compiled, e.g. the MIT database (collected by Sung and Poggio in [56]), the CMU dataset (containing the MIT database, and compiled by Rowley *et al.* [42]), the FERET database [36], the face database from AT&T Cambridge Laboratories (formerly known as the Olivetti database) [45], the M2VTS Database [37], etc. Some surveys have compared the most representative methods of each category on these "standard" test sets, and they have taken into account the previous considerations in evaluating them. Their conclusions can be summarized as follows:

1. Template-based approaches are the most robust techniques, specially for detecting grayscale frontal faces. They also can detect slightly ro-

tated faces. Nevertheless, it is necessary to have a face detector that could detect well faces which are rotated in depth. However, it is very hard to extend these systems to detect faces with out-of-plane rotation. Another large training set of faces with different poses may be required and even if such a training set is available, it is not guaranteed that existing classification methods will still give a good performance.

2. The feature-based approach promises an elegant solution to the problem of rotation in depth. However, it is very dependent on the feature detection phase. What is more, the feature detection result is not well localized. With a large variation in facial expression, it is not surprising that the facial feature detection in a general scene is not reliable. This make the relative distances between facial features unreliable for the feature grouping step, and the feature-based methods themselves not as robust as template-based methods.

3. Most template-based algorithms are based on multiresolution window scanning to detect faces at all scales, making them computationally expensive.

4. On the other hand, feature-based methods are applicable for real-time systems where color and motion are available. In these situations, the most widely used technique is skin color segmentation based on one of the methods mentioned in Section 2.2.2.

5. Nonetheless, color information has been used with limited success. This is largely because of the fact that the background may also have human skin color. However, the face skin color and the background color are still different. A better color segmentation method may successfully segment the face from background. The difficulty is that the size of the face has to be large enough for the segmentation to be reliable. Also, lighting condition and some features like glasses or facial hair may lead to undesirable segmentation result.

6. All things considered, seems clear that the future of face detection lies in combining both classes of approaches (template-based and feature-based):

    (a) For the first type of methods, multiresolution window scanning could be avoided by combining the template-based approach with a feature-based method as a preprocessor with the purpose of guiding the search based on visual clues such as skin color.

(b) As for feature-based methods, they could be combined with a final template-based check step (shall it be a simple cross-correlation with a pattern, or a more complex process). Another enhancement could be detecting the different facial features using a template-based approach. Both these upgrades have been implemented in this project.

Chapter 3

Proposed Method

3.1 Introduction

As it has been already mentioned, the proposed method cannot be classified as pure template-based nor feature-based, as it benefits from the advantages of both approaches. Nonetheless, the attention has been focused on the feature-based approach, using a *Local Feature Face Model* (LFFM) [55]. LFFMs model the face as a set of local parts arranged in a deformable spatial configuration, and not as a whole. The presence and the location of a face is determined according to the information provided by this spatial configuration. But the geometric distribution of the features on the image plane (or the *image space*) is not a reliable representation, as the useful information is hidden by differences due to translation, rotation and scaling (TRS). Besides, human faces are a class with a high inherent variability (for instance, due to emotional expressions), and they can be rotated in depth, thus distorting even a TRS–invariant configuration.

To cope with all these difficulties, the face models have been learned and tested in an affine-invariant space (the so-called *face-space*, or *affine-shape-space*). In such a space, no Euclidean shape descriptions are used (*i.e.* those requiring absolute distances, angles and areas) and, instead, descriptions involving *relative* measurements are employed (*i.e.* those which depend only upon the configuration's intrinsic geometric relations).

3.2 Affine Shape

3.2.1 Affine Transformation

An affine transformation is an important class of linear 2–D geometric transformations which maps variables into new ones (*e.g.* in image processing, pixel intensity values located at position (x_1, y_1) in an input image are mapped to a new position (x_2, y_2) in an output image). The mapping is carried out by applying a linear combination of *translation*, *rotation*, *scaling*, and/or *shearing* (*i.e.* non-uniform scaling in some directions) operations.

The general affine transformation is commonly written in homogeneous coordinates as shown below:

$$\begin{bmatrix} x_2 \\ y_2 \end{bmatrix} = \mathbf{A} \times \begin{bmatrix} x_1 \\ y_1 \end{bmatrix} + \mathbf{B} \tag{3.1}$$

where $\mathbf{A}$ and $\mathbf{B}$ are a constant 2×2 matrix and 2×1 vector, respectively. In total, 6 parameters to define the transformation.

Figure 3.1 shows some examples of affine transformations. Consider the binary artificial image in Fig. 3.1(a). A pure *translation* transformation can be carried out (Fig. 3.1(b)) by defining only the $\mathbf{B}$ vector:

$$\mathbf{A} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \mathbf{B} = \begin{bmatrix} b_1 \\ b_2 \end{bmatrix} \tag{3.2}$$

where b_1 and b_2 are the displacement (in pixels) applied to the original image.

Pure *rotation* uses the $\mathbf{A}$ matrix and is defined as:

$$\mathbf{A} = \begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix}, \mathbf{B} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \tag{3.3}$$

with θ accounting for the rotation. Fig. 3.1(c) contains a 90-degree rotated version of the image.

Similarly, pure *scaling* is:

$$\mathbf{A} = \begin{bmatrix} a_{11} & 0 \\ 0 & a_{22} \end{bmatrix}, \mathbf{B} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \tag{3.4}$$

where a_{11} and a_{22} are independent scaling factors. Two particular cases are shown in Figures 3.1(d) and 3.1(e), respectively. In the first one, a_{11} and a_{22} are held equal, and an uniform scaling is achieved. In the second one, the image has been reflected along its vertical axis by setting $a_{11} = -1$ and $a_{22} = 1$ (note that a translation has been posteriorly added in order to keep the figure "inside the image").

Finally, in Fig. 3.1(f), a combination of all of these partial transformations has been applied to the image.

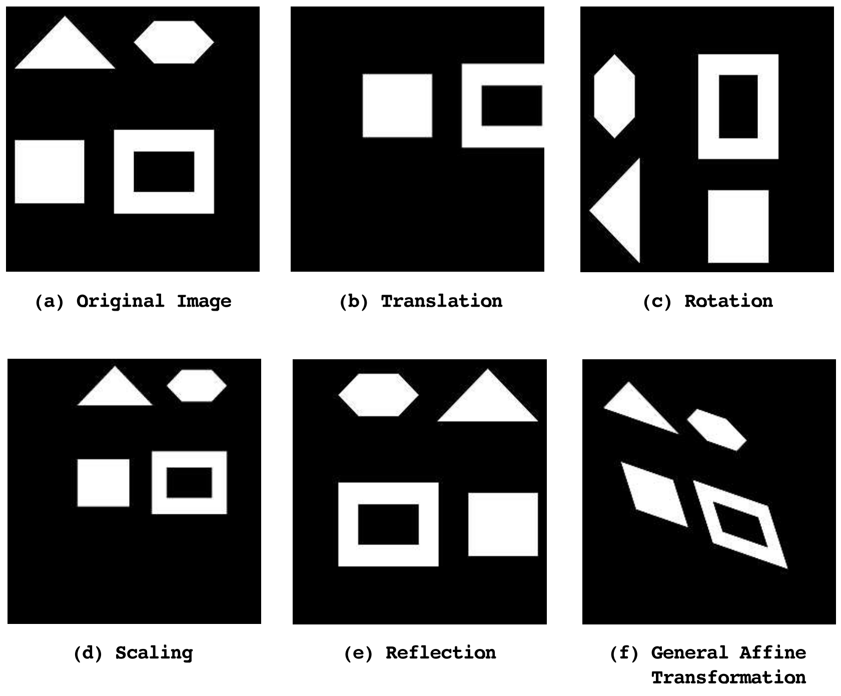


Figure 3.1: Examples of affine transformations.

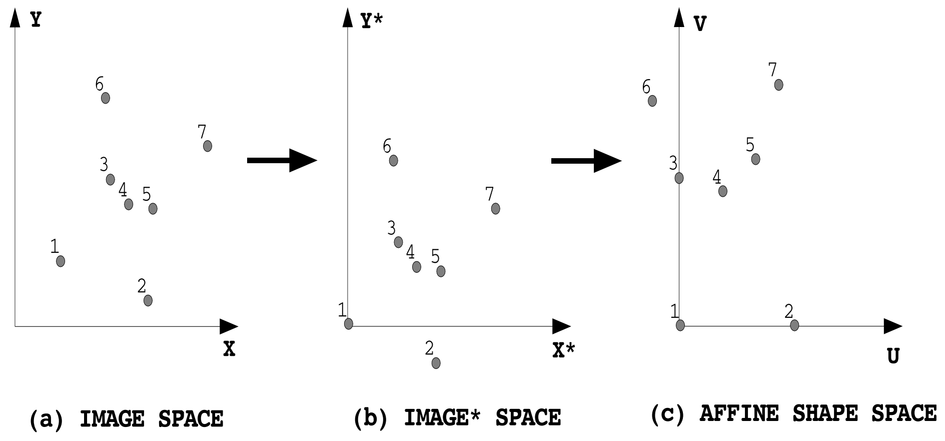


Figure 3.2: Affine transformation from image space to affine shape space.

3.2.2 Affine-invariant Model

As reasoned before, the models are built in an affine-invariant space, by the means of an affine transformation which maps the information-bearing data (the locations of facial features) from the image space to the shape space.

Consider a face in an image as formed by N labeled feature points with image space coordinates $[x_i, y_i]^T$ for $i = 1, ..., N$. As it has also been stated, an affine transformation is defined, in a 2–D space, by 6 parameters. Consequently, 3 points are needed for determining the 6 parameters and for defining the transformation. Let these 3 points be: (x_1, y_1), (x_2, y_2), and (x_3, y_3). Theoretically, any triplet of non-collinear points would be acceptable (see Sec. 4.6 for more information). Then, the transformation parameters are chosen in such a way they map (x_1, y_1) to the origin, and (x_2, y_2), (x_3, y_3) to $(1, 0)$ and $(0, 1)$, respectively.

This process is schemed for $N = 7$ in Fig. 3.2, and its steps are next explained in more detail:

- Fig. 3.2(a) shows the original 7 labelled features and their arrangement in image space.

- Next (Fig. 3.2(b)), a new coordinate system (the so-called *image* space*) is built, in which (x_1, y_1) gets mapped to the origin. The coordinates of the other features are given by:

$$\begin{bmatrix} x_i^* \\ y_i^* \end{bmatrix} = \begin{bmatrix} x_i - x_1 \\ y_i - y_1 \end{bmatrix} \tag{3.5}$$

This is equivalent to say that $\mathbf{B} = -\left[x_1, y_1\right]^T$.

- Finally, to achieve an affine invariant representation, the feature points are referenced to the basis defined by (x_2^*, y_2^*) and (x_3^*, y_3^*). This is done by defining the projection matrix $\mathbf{A}$ as follows:

$$\mathbf{A} = \begin{bmatrix} x_2^* & x_3^* \\ y_2^* & y_3^* \end{bmatrix}^{-1} \tag{3.6}$$

  In this way, (x_2^*, y_2^*) and (x_3^*, y_3^*) get mapped to $(1, 0)$ and $(0, 1)$, respectively (Fig. 3.2(c)), and the remaining feature points $\left[x_i^*, y_i^*\right]^T$ are represented by their *affine coordinates* $\left[u_i, v_i\right]^T$:

$$\begin{bmatrix} u_i \\ v_i \end{bmatrix} = \mathbf{A} \times \begin{bmatrix} x_i^* \\ y_i^* \end{bmatrix} \tag{3.7}$$

  with $i = 4...N$ (in this case, $i = 4, ..., 7$).

Grouping the *affine coordinates*, a general N-feature object is represented by a $(2N - 6)$–dimensional *affine shape vector*:

$$\mathbf{U} = \left[u_4, ..., u_N, v_4, ..., v_N\right]^T \tag{3.8}$$

3.2.3 Classification

One approach for estimating the likelihood that a set of points describe the shape of the object that is sought, is to try to estimate the *pdf* that characterizes the class of objects. Then, for classifying an *affine shape vector* (Eq. 3.8) as being a face or not, its probability density function could be estimated as in [24]. Assuming that the image space variables (x_i, y_i), for $i = 1, ..., N$, follow a general *2N*–dimensional Gaussian distribution:

$$\mathbf{X} = \left[x_1, ..., x_N, y_1, ..., y_N\right]^T \sim N_{2N}(\nu, \Omega), \tag{3.9}$$

then the affine shape variables in the affine shape vector (Eq. 3.8) are described by the following distribution:

$$f_{\mathbf{U}}(\mathbf{U}) = \frac{(N-3)! \cdot e^{-\frac{g}{2}}}{\pi^{(N-3)}} \sqrt{\frac{|\mathbf{C}|}{|\Sigma|}} \sum_{\substack{k_1, k_2 \\ k_3, k_4}} \prod_{i=1}^{4} \lambda_i^{k_i} L_{k_i} \left\{ -\frac{\rho_i^2}{2} \right\} \tag{3.10}$$

which is based on the Dryden-Mardia shape density [28], and it makes use of the Laguerre polynomial (L_{k_i}) to approximate even powers of gaussian random variables that appear in the expression (this works as long as the chosen reference features cannot become collinear). For full detail on how to obtain Equation 3.10, see [24].

This result can be generalized to the case when the image space variables should be modelled with a Mixture of Gaussians (as it is the case with face shapes). In that case, as a Mixture of Gaussians is only a linear combination of *pdf*s, the joint probability density function of the shape vector **U** is simply a mixture of shape densities corresponding to the modes of the Gaussian mixture density. The mixing parameters are estimated using the Expectation Maximization algorithm.

The same process is carried out for the non-face class. Its joint *pdf* is learned from a group of examples, formed by both full sets of random points, and combinations of real feature locations and random points.

However, this methodology has a big problem: it does not allow to cope with missing features, as the whole set of $2N-6$ variables is needed to evaluate the likelihood of a candidate face. To overcome this, instead of obtaining the likelihood of the full set of affine variables, the individual *pdf* of each one of the $N-3$ affine variables is estimated. In the affine shape space, features get more or less grouped in clusters (Fig. 3.3). The compactness of these clusters depends in great measure on the 3 features chosen to characterize the affine transformation (more information in Sec. 4.6).

Each cluster is modelled with a bivariate normal distribution:

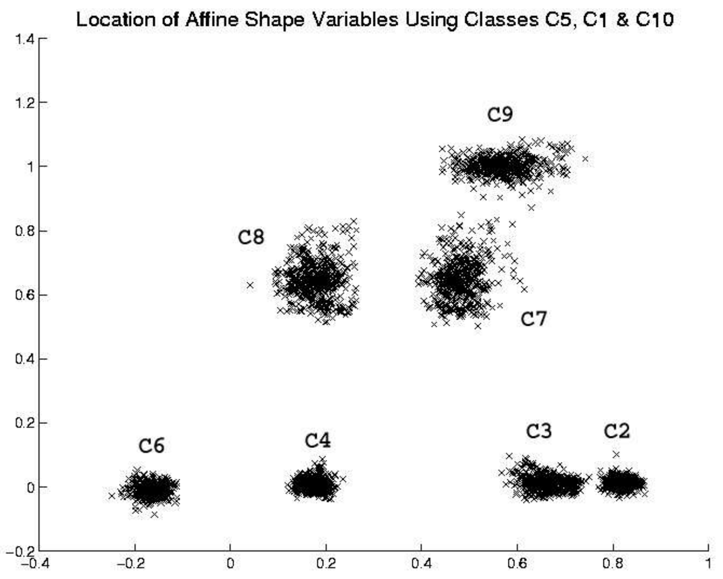
$$f(\mathbf{x}) = \frac{1}{2\pi\sqrt{|\Sigma_i|}} e^{-\frac{1}{2}[\mathbf{x}-\mu_i]^T \Sigma_i^{-1} [\mathbf{x}-\mu_i]} \tag{3.11}$$

where μ_i and Σ_i are the mean vector and covariance matrix of the i–th cluster, respectively. This probability densities define confidence regions surrounding each corresponding cluster in where to look for supporting features (for details, see Sec. 4.4.2). In this way, bad-labelled features are expected to be removed, and hypotheses are formed. This can be done even if no instances of a determined feature class are found, as the hypotheses formation procedure is based on evidence accumulation.

3.3 Evidence Accumulation

As it has been shown, to construct a 2–D shape-space, 3 points are needed. The algorithm consists basically on forming candidate *triplets* of features [1],

[1]For computational efficiency's sake, a conditional pruning is applied to reduce the total number of possible candidate triplets. See Sec. 4.4.1 (p. 41).



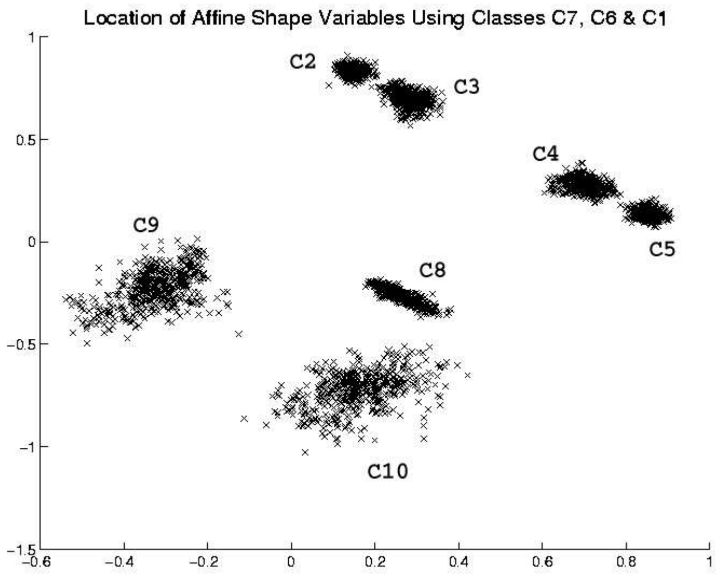


Figure 3.3: Two examples of face models in affine-shape-space. The selected feature classes (C5, C1, and C10 in the first example, and C7, C6, and C1 in the second one) have been mapped to $(0, 0)$, $(1, 0)$, and $(0, 1)$, respectively.

and validating and reinforcing the hypotheses by collecting evidence from the other features.

Evidence accumulation is an efficient way to form hypotheses, first proposed by Hough for finding parametric curves in images, and further generalized and adapted for other purposes. It can be regarded as a kind of model-based recognition approach where an internal object model is matched to the image by some voting method. Local parts of the object are detected from the image, and the detected candidate parts are combined to create hypotheses. After some plausible hypotheses have been created, they are verified one by one.

Detection/recognition approaches can be classified into *sequential hypothesis* (or template matching) and *evidence accumulation* depending on how they handle variations caused by global transformations[52]. A sequential hypothesis test directly matches the internal object model with an input image at "blind-guessed" poses. As these guesses are based on no evidence, most of the hypotheses are not useful at all and high computational resources are required. On the contrary, evidence accumulation does not generate an hypothesis by a "blind-guess", but it aggregates partial matching evidences to form a global pose hypothesis. Using this method, only the supported hypotheses are verified, and a higher computational efficiency is achieved.

In [33], a face detection method based on evidence accumulation is presented. The goal of the evidence accumulation step is to determine a face pose transformation T that maps points in the face model onto image points (Eq. 3.12). The possible face pose is limited to a TRS–transform (Eq. 3.13).

$$\left[x_1^i, y_1^i, 1\right]^T = T \times [x_1^o, y_1^o, 1]^T \tag{3.12}$$

$$T = \begin{bmatrix} s\cos\theta & -s\sin\theta & t_x \\ s\sin\theta & s\cos\theta & t_y \\ 0 & 0 & 1 \end{bmatrix} \tag{3.13}$$

Given a couple of point correspondences between the model and the image, the transform T can be uniquely determined, since there are 4 parameters and 4 constraint equations:

$$\left.\begin{array}{l} x_1^i = x_1^o s\cos\theta - y_1^o s\sin\theta + t_x \\ y_1^i = x_1^o s\sin\theta + y_1^o s\cos\theta + t_y \\ x_2^i = x_2^o s\cos\theta - y_2^o s\sin\theta + t_x \\ y_2^i = x_2^o s\sin\theta + y_2^o s\cos\theta + t_y \end{array}\right\} \Longrightarrow \widehat{T} \tag{3.14}$$

The accumulation space is defined by a discrete 4–dimensional array indexed by (t_x, t_y, θ, s), and the estimated transformation $\widehat{T}$ is accumulated in it.

After some iterations, the most voted transformation is chosen. Finally, the hypothesis is verified and the estimated transform is refined with a local-matching procedure.

The implemented method includes an evidence accumulation operation. It serves as an alternative to the joint probability density estimation seen in the previous section. Performing evidence accumulation, it is possible to cope with missing features. If a certain feature has not been detected, but nontheless the other ones provide strong evidence of belonging to a face, a hypothesis can still be formed and further validated.

But contrarily to the other method, what is going to be accumulated here are the elements of the transformation matrix, not their TRS interpretation. The purpose of the evidence accumulation step is to choose those hypotheses with highly concentrated parameters, no matter their meaning. Though the estimated transformation matrix is slightly different, an analogous accumulation space is used (see Section 4.4.2 for details).

Chapter 4

Implementation and Results

4.1 Considerations

Before summarizing the proposed method and giving details about each step of the detection process, some working issues and assumptions must be considered.

- This algorithm has been designed to detect faces in grayscale high-resolution still images. Color information has not been used, even if the algorithm could be enhanced to use this information also.

- It has been assumed the presence of at most one face in each image. Also, a large rotation of the face (be it in-plane or out-of-plane) is not dealt with (e.g., it has been assumed that there will not be upside-down faces). The most common application of a face localizer is as a first step of a face recognition system, so the assumptions made are coherent with this.

- The face models have been learned, and the tests have been performed on the BANCA database[2]. The BANCA database is a multi-modal (audio and video) and multi-language database designed for research in the field of automatic person authentication. Generally there is one person present in the image and the ground-truth for the facial features is given, making it ideal for benchmarking face detection/localization algorithms. It has been compiled in three different environments: controlled, degraded, and adverse. For each of the four languages (French, English, Spanish, and Italian), there are 52 subjects (26 males and 26 females) each performing 12 recording sessions, with 4 session per scenario. In total, there are 6240 images per language. Half of the images in the English database (3120) have been used as training set, and the

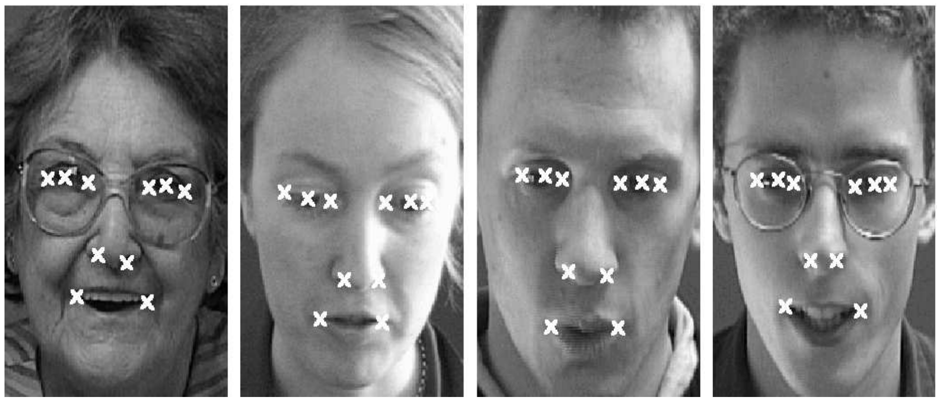


Figure 4.1: Training images with the ten considered features marked on them.

rest as testing set. The subjects are different in each subset (i.e., faces appearing in the test set do not belong to any subject appearing in the training set).

- Ten feature classes have been considered in this work. These are: the left corner of the left eye (C1), the central point of the left eye (C2), the right corner of the left eye (C3), the left corner of the right eye (C4), the central point of the right eye (C5), the right corner of the right eye (C6), the left and right nostrils (C7 and C8), and the left and right corners of the mouth (C9 and C10). These features, which can be seen in Fig. 4.1, are a representative set of salient facial features. The BANCA database includes ground-truth files (one file per image), containing the manually-labeled location of each one of these 10 features. These ground-truth files have been used for training and testing the system.

- The system has been implemented in C++ language, using the RAVL [1] and TINA [2] libraries. Data from the train sets have been collected and models have been learned using MATLAB and Perl.

4.2 Outline of the Algorithm

A summary of the proposed method can be seen in Alg 1. First, local facial features are detected and labeled. This is carried out by an interest point detector followed by a template-based scheme, which classifies the detected

[1] Documentation and source codes can be found at http://ravl.sourceforge.net/
[2] Software and documentation available at http://www.niac.man.ac.uk/Tina/

interest points into one (or more than one) of the defined feature classes. Second, for speeding up the process, an anthropometrical-based geometrical pruning is performed in order to reduce the total number of possible candidate triplets. Next, for each surviving candidate triplet, an affine-invariant space is constructed and the other features are projected onto it. Only the points falling inside their corresponding search regions are kept. Pairs of such points are formed randomly and some transformation parameters are estimated. These parameters are accumulated and only hypotheses obtaining enough evidence are passed onto the final validation step.

Algorithm 1 Face Detection Method

```
Detect local features and label them;
for All possible feature triplets do
    Apply constraints;
end
for All triplets having overcome all constraints do
    Feature Selection;
    Evidence Accumulation;
    if There is enough evidence of being a face then
        Cut image patch;
        Validate face candidate;
        if Validation Score sufficiently high then
            Early stopping;
        end
    end
end
Select the highest Validation Score;
```

Following sections explain in deeper detail each one of the forementioned steps.

4.3 Feature Detection and Classification

4.3.1 Detection of Interest Points

As high-resolution images are used, the template-based methods characteristical sliding-window technique is avoided (since that would significantly slow down the process). Instead, an interest point detector is used. Among all existing detectors, the Harris Corner Detector [12] has been implemented, regarding its good performace [47].

The Harris Corner Detector is based on a matrix related to the autocorrelation function. This matrix $\mathbf{M}$ averages derivatives of the signal in a window W around a point (x, y):

$$\mathbf{M}(x,y) = \begin{bmatrix} \sum_{p_k \in W} \left(I_x(p_k)\right)^2 & \sum_{p_k \in W} I_x(p_k) I_y(p_k) \\ \sum_{p_k \in W} I_x(p_k) I_y(p_k) & \sum_{p_k \in W} \left(I_y(p_k)\right)^2 \end{bmatrix} \tag{4.1}$$

where $I(p_k)$ is the image function and p_k are the points in the window W around (x, y). This matrix captures the structure of the neighborhood. If its rank is two, (i.e. both of its eigenvalues are large), a corner is detected.

Two parameters can be tuned: the threshold and the window width. Though obtaining a higher number of false detections, the threshold is fixed quite low in order not to miss the true features. For the same reason, the window width is set at its minimum: 3 pixels. If a wider window is used, the conditions to overcame the threshold become more restrictive, and so there are fewer detected corners. Fig. 4.2 shows the output of a Harris Corner Detector.

4.3.2 Feature Classification

The list of points obtained in the first step correspond to candidate locations for a facial feature. Most of these detected interest points are not facial features at all. To reject false alarms, and to label candidates as belonging to one or more of the defined feature classes, a template-based local feature detection method is performed. The idea is that the appearance of local features is much less affected by moderated out-of-plane rotations, than the appearance of the whole face pattern is. Thus, while it has been shown that holistic template-based methods are not reliable when faced with this sort of problem (i.e., affine transformations), they can perform in an acceptable manner when applied to local features.

The implemented method is based on Support Vector Machine[3] combined

[3]SVMs have been roughly presented in Section 2.1.3 (p. 16). A more detailed description can be found in [3].

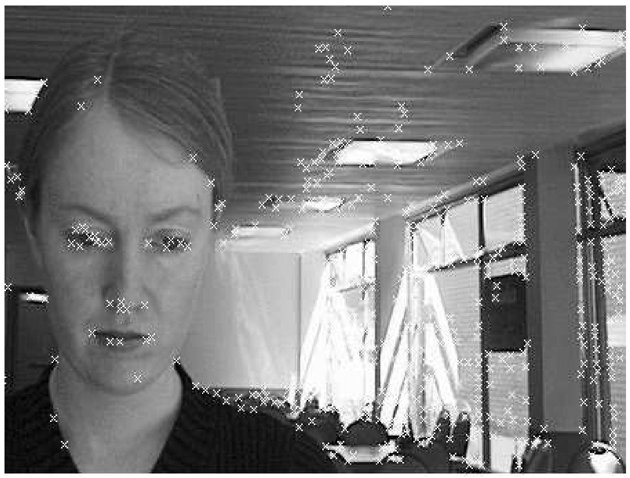


Figure 4.2: Interest points detected by the Harris Corner Detector in a cluttered scene.

with *Independent Component Analysis* (ICA), in a similar way than in [1]. ICA is a statistical and computational technique for revealing hidden factors that underlie sets of random variables, measurements, or signals. It defines a generative model for the observed multivariate data. In the model, the data variables are assumed to be linear or nonlinear mixtures of some unknown latent variables, and the mixing system is also unknown. The latent variables are assumed non-gaussian and mutually independent, and they are called the *independent components* of the observed data. These independent components, also called *sources* or *factors*, can be found by ICA. It is a much more powerful technique than PCA or FA, capable of finding the underlying factors or sources when these classic methods fail completely. The main reason being that ICA gets more than second order statistics (covariance), and much of the information that perceptually distinguishes faces is contained in the higher order statistics of the images. For more information on ICA, see [17].

Ten SVM classifiers (one for each chosen feature class) have been trained. First, fixed-size patches (16×16, 32×32... pixels) are cut around each feature. For each patch, a PCA projection is performed to reduce the dimensionality, passing from high-dimensionality (256, 1024...) to a more handleable number of components (with 50 components, more than 95% of the total variance is kept). From the PCA transformed space, the ICA is applied and the SVM classifier is trained in the ICA space using a radial basis function kernel. Once the SVM classifiers are trained, the feature classification step of the face detection algorithm is performed in an analogous way: patches are cut around each detected interest point, then they are fed to the PCA-ICA projection procedure, and finally each SVM classifier determines wether the patch belongs to its associated facial feature class or not.

This feature classification step must be seen as a "black-box" producing 10 lists of candidate features out of an input list of detected interest points. This means that any other well-performing template-based classification method could have been used (i.e. Neural Networks, HMMs, etc.). In fact, during the testing phase (see Sec. 4.7 for details) the full local feature detector and classifier has been substituted by a much faster emulator. It consists of a subroutine which generates an output analogous to that of the original system, both the amount of it and its distribution. Actually, what it does is to generate a certain number of *inliers* and *outliers* for each real facial feature. This generated false alarms are labelled like their progenitor. Operating in this way, the testing time is improved significantly while not reducing the performance of the system.

4.4 Feature Grouping Analysis

4.4.1 Reducing the Search Space

As seen in Section 3.2, three points are needed to construct the affine-invariant shape space in which to project the other features. If this had to be carried out for every possible combination of 3 candidate features, an extremely high computational power would be required. Having lists containing a mean of M features each, the total number of subsets formed by instances of 3 determined classes is $O(M^3)$. In order to reduce the search space, an anthropometrical-based pruning is performed.

The geometrical pruning is based on a set of constraints concerning the current *triplet* of 3 candidate features: the relative positions between them, the angles they form, and the ratios of distances between them. The constraints are applied sequentially to the candidate triplets. If a constraint is not overcome, the triplet is automatically rejected, and the process starts again with the next candidate triplet. After applying this chain of constraints, in most of the cases the search domain is reduced above the 99% (i.e., less than 1% of total possible triplets survive).

The constraints have been learned from collected data of the training set. When explaining them in detail, the 3 chosen candidate features are referred to as A, B, and C.

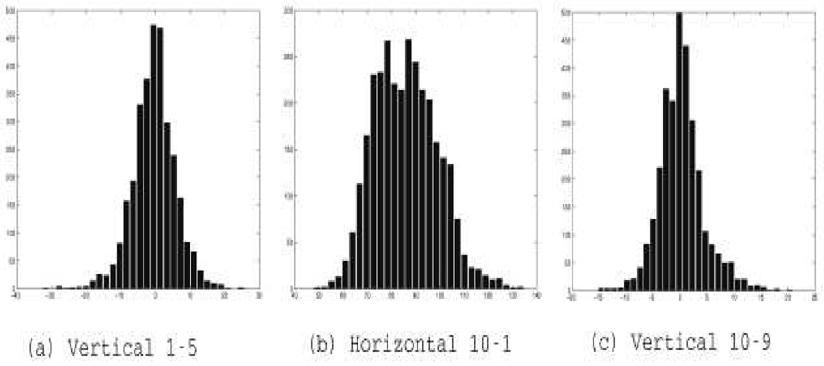


Figure 4.3: Histograms of signed distances between features.

Relative Positions

Constraint#1 accounts for the geometric position of feature B relative to A, while Constraint#2 accounts for the position of feature C relative to both A and B. That is to say, the vertical and horizontal positions of features are compared. The decision is made according to the model learned from the training set. For example, if a facial feature appears repeatedly in the training set below another one, this can be assumed as a general rule (e.g., as it has been assumed that no upside-down faces are going to appear in the database, it can be stated that mouth features always appear below eye features).

The models are learned as follows: for each image, signed distances (horizontal and vertical) between features are computed (all vs. all). These distances are modelled to follow a normal distribution, which is coherent with experimental data (Fig. 4.3). The mean values and standard deviations of these distances are computed. All possible situations are summarized in Table 4.1, where μ_h and σ_h are the mean value and the standard deviation of the horizontal distance ($column_i - column_j$), while μ_v and σ_v are the mean value and the standard deviation of the vertical distance ($row_i - row_j$).

Distance Ratios

As Fig. 4.4 shows, 3 distances (d_1, d_2, and d_3) have been defined for any given triplet of features (*F1*, *F2*, and *F3*). Raw distances between features are not a good decision tool, as different scales are taken into consideration. So,

	Cases	Decision made
Horizontal	$\mu_h - 3\sigma_h > 0$	*Feature* **i** *is right of Feature* **j**
	$\mu_h + 3\sigma_h < 0$	*Feature* **i** *is left of Feature* **j**
	else	*Ambiguous (no decision made)*
Vertical	$\mu_v - 3\sigma_v > 0$	*Feature* **i** *is below Feature* **j**
	$\mu_v + 3\sigma_v < 0$	*Feature* **i** *is above Feature* **j**
	else	*Ambiguous (no decision made)*

Table 4.1: Constraint#1 and Constraint#2 applied to features i and j.

instead of on distances, the constraints have been based on distance ratios:

$$Constraint\,\#3: \quad D_{3,1} = \tfrac{d_3}{d_1}$$

$$Constraint\,\#4: \quad D_{3,2} = \tfrac{d_3}{d_2}$$

$$Constraint\,\#5: \quad D_{2,1} = \tfrac{d_2}{d_1}$$

As the previous constraints, the distance ratios have also been learned from the training set, and they have been modelled as univariate gaussian distributions. The decision rules are easy: if any distance ratio is outside a *confidence interval*, the current triplet is rejected. Otherwise, it is kept for further analysis. The confidence interval is defined as: $\left[\mu_{D_{i,j}} - 3\sigma_{D_{i,j}}, \mu_{D_{i,j}} + 3\sigma_{D_{i,j}}\right]$, where $\mu_{D_{i,j}}$ and $\sigma_{D_{i,j}}$ are the mean value and the standard deviation of the corresponding distance ratio, respectively.

Angles

For a triplet of features, the angles they form are useful information about their arrangement. To take profit of this, 3 angles have been defined for any triplet of features: α, β, and γ (Fig. 4.4). Again, the constraints have been learned from the training set, and they have been modelled with normal distributions. The angles have been obtained out of the distances computed for the previous constraints, and applying the *Law of Cosines*:

$$Constraint\,\#6: \quad \alpha = \arccos \tfrac{d_1^2 + d_3^2 - d_2^2}{2 d_1 d_3}$$

$$Constraint\,\#7: \quad \beta = \arccos \tfrac{d_1^2 + d_2^2 - d_3^2}{2 d_1 d_2}$$

$$Constraint\,\#8: \quad \gamma = \arccos \tfrac{d_2^2 + d_3^2 - d_1^2}{2 d_2 d_3}$$

The decision rules for these constraints are analogue to those exposed above for the distance ratios.

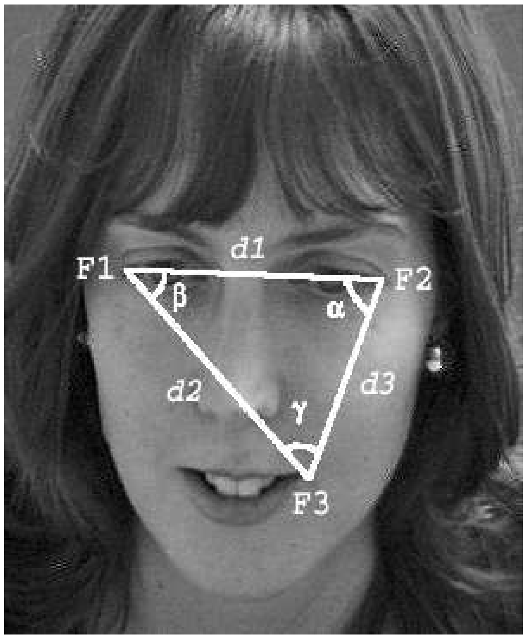


Figure 4.4: Defined distances and angles for *F1*, *F2*, and *F3* (belonging to *C1*, *C6*, and *C10*, respectively).

4.4.2 Forming Hypotheses

As stated above, after applying the set of constraints the search space has been drastically reduced. This is a more suitable environment where to form hypotheses. To become an hypothesis, a candidate triplet must be strengthen by supporting features, and by the evidence they provide.

As explained in Sec. 3.1, features need to be transformed and projected into an affine-invariant space (the *affine-shape-space*), as they do not provide reliable information while in the *image-space*. On the other hand, when projected to the shape-space, features get clustered in easily separable "feature-clouds" (remember Fig. 3.3, p. 33).

Face Model

From the training set, the affine shape variables are computed and collected, and the face models are built. Given a selection of 3 feature classes [4], the remaining 7 classes are modelled with a bivariate normal distribution each one. Their mean vectors and covariance matrices are estimated from the data,

[4]For details on how to choose these 3 classes, see Sec. 4.6 (p. 51).

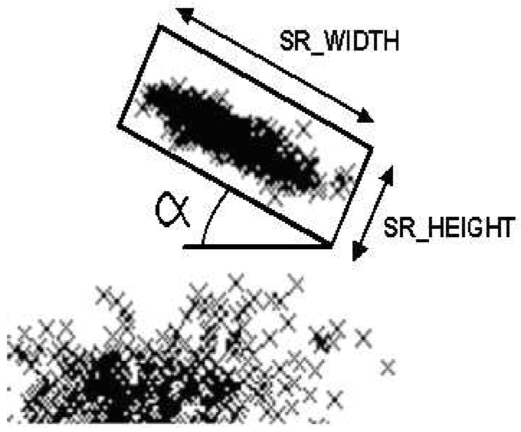


Figure 4.5: Search region for a certain feature cluster.

and they are kept in order to evaluate candidate features. The covariance matrices' eigenvalues and eigenvectors are also kept, since they are needed to determine the search regions in which to look for supporting features. To make things easy, these search regions have been designed rectangular, but fitting the clusters as well as possible.

Fig. 4.5 shows a feature cluster and its corresponding search region, whose parameters (size and orientation) are to be estimated. If the cluster was horizontal or vertical ($\alpha = 0$ and $\alpha = \pm\frac{\pi}{2}$, respectively), the size of its search region would be given the partial standard deviations. As in general it is not the case, the best way to obtain the size (and also the orientation) is via a PCA analysis of the data. The eigenvector with the highest eigenvalue determines the angle, while SR_WIDTH and SR_HEIGHT are obtained from the highest and lowest eigenvalues (λ_1 and λ_2, respectively). The actual expressions are:

$$\begin{aligned} SR\_WIDTH &= 6\sqrt{\lambda_1} \\ SR\_HEIGHT &= 6\sqrt{\lambda_2} \end{aligned} \tag{4.2}$$

The eigenvalues account for the variance of the data. And it is a known property of the normal distribution that 99% of the variance is kept in the interval: $mean\_point \pm 3 \times standard\_deviation$. So, using the values in Eq. 4.2, it is ensured that almost any good feature is going to be missed.

Feature Selection

Alg. 2 summarizes the procedure for accepting or rejecting a feature. The

Algorithm 2 Feature Selection

```
for All candidate triplets do
    Construct transformation matrix;
    for All remaining 7 feature classes do
        for All detected features of current class do
            Project feature to affine shape space, center it, de-rotate it;
            if Feature is inside search region then
                Add feature to a selected-features list.
            end
        end
    end
end
```

transformation matrix **A** is constructed as in Eq. 3.6, and it is used to project a feature f_i into affine shape space:

$$f_i^* = \mathbf{A}\left(f_i - p_1\right) \tag{4.3}$$

where p_1 is the first feature in the current triplet (the one which gets mapped to $(0,0)$). Once in affine shape space, the feature is centered (i.e., the mean value of its corresponding class cluster is substracted to it) and de-rotated (the angle of its class' cluster is also substracted to it). Now, the feature has been aligned with the search region, so it is very easy to determine if it falls inside or outside. If the current feature lies inside the search region, it is evaluated with the learned probability density function (Eq. 3.11). The result is normalized by the maximum value (i.e., the evaluation of the mean vector μ_i), in order to be able to define a relative threshold th_1 (left as an input parameter of the system). If the evaluation outcome surpasses th_1, the feature coordinates in image-space are kept in a list of selected features, which are used in the next step, the search for supporting evidence.

Searching Supporting Evidence

After having carried out the feature selection, if a triplet has not obtained any supporting feature whatsoever, it is discarded and no further considered. If it has, an evidence accumulation process is engaged, as briefly outlined in Sec. 3.3.

For a given pair of selected features, there exists a transformation matrix **T** to map their image-space coordinates out of the center of their cluster in affine-shape-space:

$$\begin{pmatrix} f_{1_x} & f_{2_x} \\ f_{1_y} & f_{2_y} \end{pmatrix} = \mathbf{T} \times \begin{pmatrix} \mu_{1_x} & \mu_{2_x} \\ \mu_{1_y} & \mu_{2_y} \end{pmatrix} + \begin{pmatrix} p_{1_x} \\ p_{1_y} \end{pmatrix} \tag{4.4}$$

where p_1 is current triplet's Point#1, f_1 and f_2 are the image-space coordinates of Feature*#1* and Feature*#2*, respectively, and μ_1 and μ_2 are the mean values (or centers of the clusters) of their respective classes. From this expression, the transformation matrix **T** can be easily obtained:

$$\mathbf{T} = \begin{pmatrix} \mu_{1_x} & \mu_{2_x} \\ \mu_{1_y} & \mu_{2_y} \end{pmatrix} \begin{pmatrix} f_{1_x} - p_{1_x} & f_{2_x} - p_{1_x} \\ f_{1_y} - p_{1_y} & f_{2_y} - p_{1_y} \end{pmatrix}^{-1} = \begin{pmatrix} T_{11} & T_{12} \\ T_{21} & T_{22} \end{pmatrix} \tag{4.5}$$

The elements of matrix **T** are the evidence that is accumulated. For each chosen pair of features, matrix **T** is computed as in Eq. 4.5 and their elements T_{ij} are stored in a *4*–dimensional array of bins (i.e., a *4*–D histogram). Only triplets with supporting features that obtain highly clustered transformation parameters T_{ij} (and thus providing strong hints of belonging to the same face), are going to be kept as hypotheses.

Even though the initial search space has been reduced, and only the well-placed features have made it through the process, the algorithm is further sped up by not considering all possible pairs of features. Instead, an iterative process is performed: a sort of *Bernoulli Trial* is repeated in order to choose pairs of classes and an example feature of each. In fact, it is not a *Bernoulli Trial*, since the trials are performed on a discrete uniform probability distribution (each one of the *feature-providing* classes has the same probability to be chosen). This process is repeated a fixed number of iterations (i.e., a fraction of the total number of combinations), and is summarized in Alg. 3. The criteria followed for deciding whether there is or there is not enough evidence is described in Fig. 4.6. For each candidate triplet, after having accumulated the evidence, the total number of votes, as well as the indexes and number of votes of the two most-voted bins are kept and used to make the decision. If there is a single bin (or two neighboring bins) with a high percentage of the votes (i.e., a number of votes above a threshold th_2, set by the user), the

Algorithm 3 Evidence Accumulation

```
if 2 or more different classes provide supporting features then
    Compute the total number of possible feature pairs ($K$);

    for A subset of K iterations do
        Randomly choose a pair of feature samples;
        Obtain transformation parameters $T_{ij}$ and accumulate them;
    end
    if There is enough evidence then
        Keep current candidate triplet for final validation;
    end

end
```

current candidate triplet it is considered to be a good hypothesis and worth validating it. If not, the triplet is discarded and the process continues with the next one.

Concerning the evidence accumulation process, there are two related aspects that must be taken into consideration: the histogram range and the number of bins it contains. The bounds for every parameter have been obtained from the training data. Each parameter has been modelled with a normal distribution, and the margins have been chosen to contain almost all the variability (again, the mean value $\pm$ three times the standard deviation).

Having fixed the histogram's range, the number of bins (B) must be carefully chosen. If B is too small, the resolution will be too coarse and the histogram will not accurately model the votes distribution (a sparse distribution of votes could be taken as an acceptable one). But if B is too large, then the accumulated evidence will get too scattered (too much resolution), and an unrealistically large amount of data would be required to populate adequately the histogram.

In the proposed method, B is left as a variable parameter. Learning and training have been carried out for both $B = 5$ and $B = 10$ values. No significant variation in performance has been observed.

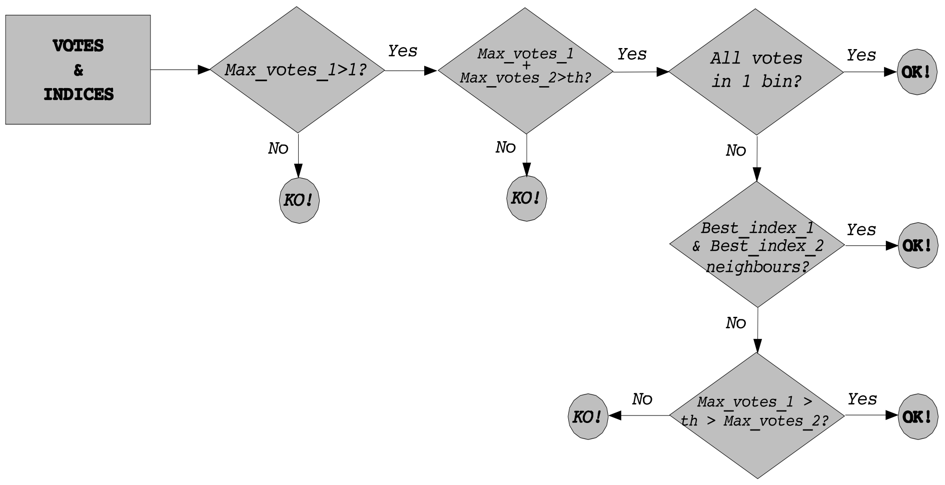

VOTES & INDICES
Max_votes_1>1?
Yes
No
KO!
Max_votes_1 + Max_votes_2>th?
Yes
No
KO!
All votes in 1 bin?
Yes
OK!
No
Best_index_1 & Best_index_2 neighbours?
Yes
OK!
No
Max_votes_1 > th > Max_votes_2?
Yes
OK!
No
KO!

Figure 4.6: Evidence accumulation decision rules. After the voting process, this is how it is decided if a candidate feature has obtained enough evidence or not.

4.5 Validation

Finally, if a candidate triplet has reached this point, it means that it has provided reasonable evidence of belonging to an actual face, and that is worth taking into consideration. So, the final step of the algorithm is a template-based validation procedure.

As in the local feature detection stage, any of the methods briefly described in Sec. 2.1 could serve. In this case, a crosscorrelation-based template matching proceeding is used. The presumed face patch is cut from the image and it is crosscorrelated with the mean face. The mean face has been obtained as the average of all faces in the training set. To get a mean face as representative as possible, non-upright faces have been previously derotated before averaging them. Also, since a non-uniform lateral illumination is present throughout the database, a simple post-processing is applied to the computed mean face: it is made symmetrical by averaging it with its specular image along the vertical axis:

$$MF^{*}\left(i,j\right)=\frac{MF\left(i,j\right)+MF\left(i,\mathbf{w}-j\right)}{2} \tag{4.6}$$

where $MF\left(i,j\right)$ is the actual mean face, $\mathbf{w}$ is the width of the mean face patch, and the indices i and j account for *rows* and *columns*, respectively.

To cut the image patch, the location of the centers of the eyes, as well as the location of the corners of the mouth, are needed. As the transformation

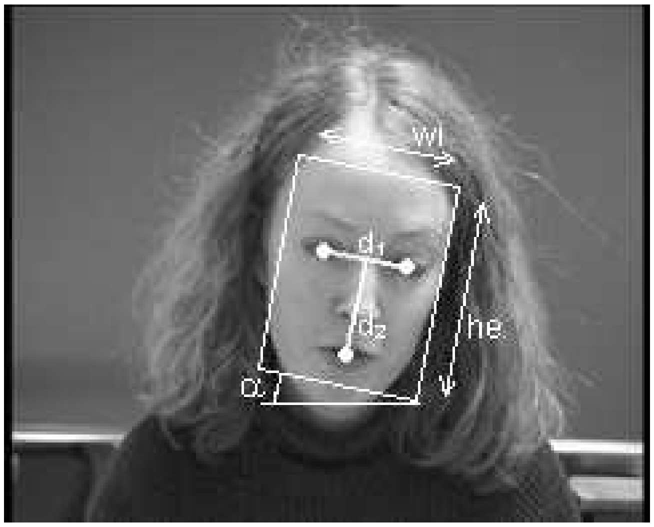


Figure 4.7: Cutting a face patch (either when computing the mean face or evaluating a face candidate).

to map points from affine-shape-space to image-space has already been obtained, it is just a matter of anti-transforming the mean value of each needed class ($C2$, $C5$, $C9$, and $C10$). The shape and orientation of the rectangle surrounding the face can be estimated from the relative positions of these anti-transformed points (Fig. 4.7). The rectangle is oriented according to the angle α (formed by the line connecting both eye centers and the bottom of the image), and the rectangle dimensions are obtained as follows:

$$\begin{aligned} he &= 2 \times d_2 \\ wi &= a\_ratio \times he \end{aligned} \tag{4.7}$$

where d_2 is the distance between the midpoint of the mouth corners and the midpoint of the eyes centers, and a_ratio is the aspect ratio of the mean face patch (19×25 pixels in this implementation).

The validation function returns a normalized value (ranging from -1 to $+1$) which is stored along with the current triplet's coordinates. After having studied all the candidate triplets, the one obtaining the highest score (above a minimum threshold th_3, set by the user) is chosen as the estimated location of the face. To speed things up, an "early-stopping" mechanism has been implemented. When validating a candidate, if there is enough proof of being a face at the current location (i.e., the validation score is above a certain threshold), the algorithm stops and a face is decided to be present there. Since this is a somewhat risky decision, the selected "early-stopping" threshold th_4 must be quite high (around 0.8 or 0.9). Like the other thresholds, this one is also set by the user.

4.6 Selection of Feature Classes for Face Models

As explained in previous sections of this document, to build an affine shape face model three feature classes are needed. This raises an immediate question: *What happens if no instances of one of these needed classes are found?*

The obvious answer is: since no affine-invariant space where to project supporting features can be built, and so the detection process cannot continue, no face will be detected at all. The main reason for not detecting any instance of a feature class is occlusion, due principally to the presence of glasses and hair (let it be the hair style, or facial hair like beards or mustaches). The easiest and most straightforward solution is to learn and use more than one single model, trying to cover all possible eventualities (i.e., occlusion of the eye, nose or mouth features). That leads to the another question: *Which are the most suitable classes for these models?*

Having a set of 10 different facial feature classes, the total number G of groups that can be formed with 3 of these features is: $G = C_{10}^3 = 120$. Before answering which are the *most suitable* classes, it must be pointed out that there are some of these 120 combinations that are *not suitable* at all. As an inverse matrix has to be computed in order to obtain the *affine coordinates* (Eq. 3.7), it must be ensured that this matrix is accurately invertible. This is achieved if the matrix is not singular, and that can be checked with the *conditioning number*. This number is defined as the ratio of the largest to the smallest eigenvalue, and the matrix is considered to be invertible if this ratio is not larger than the inverse of the machine accuracy or relative rounding off error.

Geometrically, this can be fulfilled by avoiding triplets of collinear or nearly-collinear features. In practice, what has been done is to avoid groups formed entirely by eye features (which are almost collinear), which leaves the total number of possible groups in: $G' = C_{10}^3 - C_6^3 = 120 - 20 = 100$.

Even though none of these 100 groups would lead to a degenerate solution for the affine transformation, there are some of them that would lead to a highly error-sensitive transform. This kind of groupings are those whose 3 features are close to each other. The practical consequence is that the *feature-clouds* in the affine-shape-space become more dispersed or even overlapped between them. This is an undesirable situation, unless there is no other choice (i.e., the subject's eyes are occluded by sunglasses, and only nose and

Combination	**Clustering Factor**
2-6-7	0.005127
3-6-7	**0.005528**
1-6-7	0.005647
2-5-7	0.005689
2-6-8	**0.005924**
1-5-7	**0.005940**
1-5-8	**0.006072**
1-6-8	0.006146
2-5-8	0.006191
3-6-8	0.006379

Table 4.2: Selected models (without mouth features).

Combination	**Clustering Factor**
1-6-9	**0.001427**
1-5-9	0.001523
2-6-9	0.001609
2-5-9	**0.001903**
1-6-10	0.001948
2-6-10	**0.001979**
1-4-9	0.002202
3-6-9	0.002295
1-5-10	**0.002349**
2-5-10	0.002606

Table 4.3: Selected models (without nose features).

mouth features have been detected). But as this is not going to be the most usual case, it is preferable to form models with selections of feature classes that conduct to a highly clustered data. In this way, the associated search regions will be smaller, and that conveys stronger constraints to potential supporting features.

As 3 different classes are needed to form a model, and it has been showed that a model cannot be formed by eye features only, that forces the system to find instances of at least two of the feature class groups: eye features, nostrils, and mouth corners. If this is not fulfilled, the detection process cannot be carried out at all.

Once again, the training part of the database has been used to determine which are the most reliable triplets of feature classes for each of these adverse

Combination	**Clustering Factor**
8-9-10	**0.116465**
7-9-10	**0.127947**
7-8-9	**0.332697**
7-8-10	**0.677998**

Table 4.4: Selected models (without eye features).

situations:

1. Eye features not detected (because of hairdo, glasses, hat, etc.).
2. Nostrils not detected (because of pose, presence of a mustache, etc.).
3. Mouth corners not detected (because of presence of beard, mustache, clothes, etc.).

All 100 possible models have been formed, and their clustering has been compared. For each model, the eigenvalues of all the feature clouds have been computed, and their average value is kept (hereafter, the *Clustering Factor*).

Tables 4.2, 4.3, and 4.4, show the best selection of classes for building the models (in clustering factor terms). The three cases have been taken into account: no mouth features detected (Tab. 4.2), no nose features available (Tab. 4.3), and no eye features present (Tab. 4.4). The selected combinations (four for each case) have been marked in bold letter. Notice that the selection criteria has not been just to pick the four lowest-clustering factor combinations. Instead, the chosen selections, while having a clustering factor as low as possible, intend to provide a wider variety of classes. For example, if the 4 top-scoring combinations in Table 4.2 had been chosen, class#8 would not have been present in any model. So, the chosen combinations of classes are (in clustering factor descending order):

No nose classes: 1-6-9, 2-5-9, 2-6-10, 1-5-10,

No mouth classes: 3-6-7, 2-6-8, 1-5-7, 1-5-8,

No eye classes: 8-9-10, 7-9-10, 7-8-9, 7-8-10.

During the execution of the method, after the features have been detected and classified, the models to use are chosen. First, the best model according to the found features is selected, and the face detection process is engaged. If the detection fails, no matter the reasons, another model is selected, and the process is started again, until a face is detected or there are no models left.

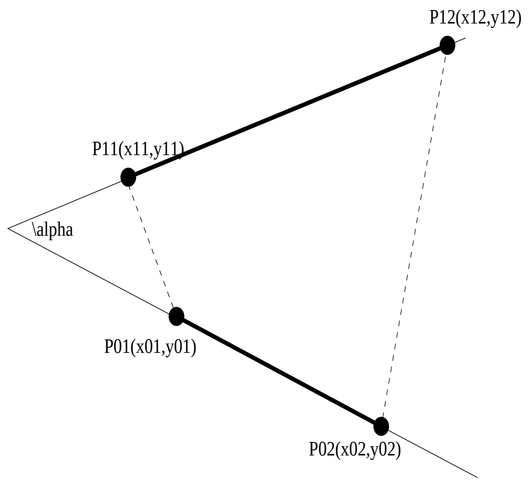


Figure 4.8: Model for the evaluation method.

4.7 Results and Evaluation

Having already described the theoretical background and all the details concerning the implementation of the algorithm, it only remains to present the obtained results, and to evaluate the overall performance of the system.

One of the considerations in the discussion in Section 2.4, is that the reported results and systems performances are not generally reliable, as their testing conditions differ significantly, and no details are given about how to evaluate the detections. To put some remedy to this situation, a detailed, objective and accurate evaluation method has been used, and is presented in next section.

4.7.1 Evaluation Method

The implemented evaluation protocol is based on the positions of the true eyes' centers and the detected ones. As showed before, even if these features (the centers of the eyes) could not be detected, they can be computed applying the estimated affine transformation to the mean vector of the eyes' centers clusters. The real positions, on the other hand, are obtained directly from the ground-truth files in the database.

Figure 4.8 shows a sketch of the scenario. Let P_{01} and P_{02} be the true positions of the eyes, and let P_{11} and P_{12} be the detected positions. The score given to a detection is based on some parameters concerning the lines connecting both pair of eyes' centers. A good parameter to model the relationship between both lines is the cosine of the angle they form ($\alpha = \widehat{\overrightarrow{P_{01}P_{02}}, \overrightarrow{P_{11}P_{02}}}$). Other parameters used are some distance ratios between the defined points:

$$d_a = \frac{\left|\overrightarrow{P_{11}P_{12}}\right|}{\left|\overrightarrow{P_{01}P_{02}}\right|}$$

$$d_b = \frac{\left|\overrightarrow{P_{01}P_{11}}\right|}{\left|\overrightarrow{P_{01}P_{02}}\right|}$$

$$d_c = \frac{\left|\overrightarrow{P_{02}P_{12}}\right|}{\left|\overrightarrow{P_{01}P_{02}}\right|}$$

These four parameters are well suited for the job, because they do not depend on the scaling nor the rotation of the lines (i.e., they are TRS–invariant). Each one of them is computed and scored individually, according to this scoring function:

$$\Phi\left(x;\gamma,\delta,\mu\right) = \begin{cases} \exp\left(-\gamma^2\left(\left(x-\mu\right)+\delta^2\right)^2\right) & x \leq \mu-\delta^2 \\ 1 & \mu-\delta^2 < x < \mu+\delta^2 \\ \exp\left(-\gamma^2\left(\left(x-\mu\right)-\delta^2\right)^2\right) & \mu+\delta^2 \leq x \end{cases} \quad (4.8)$$

This is a smooth function of x, parametrized by γ, δ, and μ, which are real values. γ defines the slope of the tails of the function, δ defines the width of the *plateau*, and μ sets the center of the function. Fig. 4.9 shows a plot of the scoring function with $\gamma = 0.5$, $\delta = 1.5$, and $\mu = 2$.

4.7.2 Obtained Results

The face detection algorithm has been tested on the testset part of the BANCA database. It consists of 3120 images from 26 people (none of them appearing in the training set), in all three environment conditions (controlled, degraded, and adverse). For each image, the detected locations of the centers of the eyes have been collected, along with the real positions from the corresponding ground-truth file.

For each detection, the four parameters ($\cos\alpha$, d_a, d_b, and d_c) are computed, and they are scored according to their associated function Φ. Table 4.5 summarizes the values of γ, δ, and μ used for each parameter, which are next commented:

- The parameter μ accounts for the center of the function. For $\cos\alpha$ and d_a it is set to 1, as this is the expected value for a "perfect" detection:

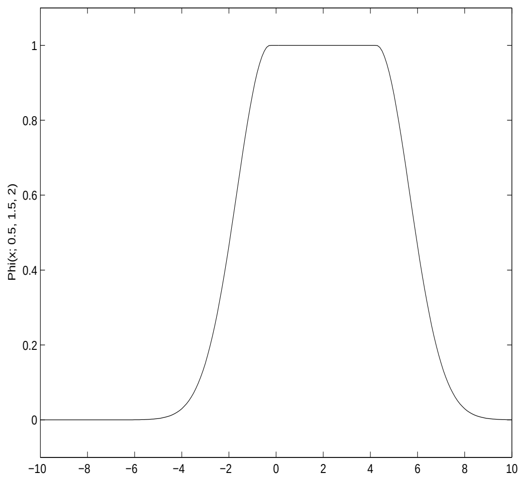


Figure 4.9: Φ Scoring Function.

if the detected and the true points were the same, the distance ratio would be 1, and the angle they form would be 0 (and thus, its cosine would be equal to 1). On the other hand, μ is set to 0 for the other two parameters (d_b and d_c), as the distances $\overrightarrow{P_{01}P_{11}}$ and $\overrightarrow{P_{02}P_{12}}$ would be 0, should the detected and the true locations coincide.

- The parameter δ determine the width of the *plateau* (Φ is 1 from $\mu - \delta^2$ to $\mu + \delta^2$). The chosen values of δ set the *tolerance intervals* for each parameter (i.e. those values for which is considered to be a good detection). For the distances, a value of $\delta = 0.3162$ ($\delta^2 = 0.1$) has been set, as it gives a 10% tolerance. The chosen value for $\cos \alpha$ (0.1232) ensures a tolerance of ± 10 degrees.
- Finally, the parameter γ controls the slope of the two branches of the function. It has been tuned to give a 0 score (or almost zero, as the branches have an exponential form) to unacceptable values of the detection.

As explained, the final score is obtained by multiplying the four parameter scores. Figure 4.10 shows the obtained results. A threshold should be set for classifying detections as “good” or “poor”. Since the final scores come from the product of four partial ones, a high global score can only be achieved if all parameters obtain high scores. For instance, supposing all detection parameters ($\cos \alpha$, d_a, d_b, and d_c) obtain a partial score of 0.9, their combined value becomes only $\simeq 0.66$. So, the final detection threshold has been set

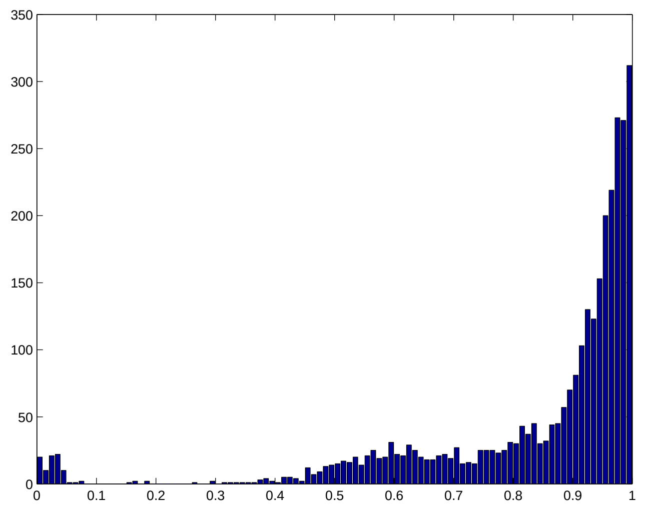


Figure 4.10: Algorithm results.

	γ	δ	μ
$\cos\alpha$	2	0.1232	1
d_a	2	0.3162	1
d_b	2	0.3162	0
d_c	2	0.3162	0

Table 4.5: Φ function parameters.

Thres	% of scores above *Thres*
0.1	97.21
0.2	97.05
0.3	96.96
0.4	96.47
0.5	94.17
0.6	87.82
0.7	80.93
0.8	73.65
0.9	59.78

Table 4.6: Distribution of the scores.

in **0.7**: detections obtaining a higher score are labelled as "good detections", while detections with a final score below this threshold are defined as "poor detections". In Fig. 4.10, there are more than 80% of "good detections", as their scores are above the 0.7 threshold. The full score distribution can be observed in Table 4.6. It can be seen that only about 5% of the scores are below 0.5, and that about 60% of the detections are scored above 0.9, *i.e.* "almost-perfect-detections".

In Fig. 4.11, some examples of good detections can be observed. The first and second images have obtained a score of 1, while the last one has received a score of 0.7886. Following the previously exposed criterion, this would be considered a roughly good detection. However, subjectively it does certainly seem a good detection.

On the other hand, Fig. 4.12 shows examples of poor detections. They have obtained scores between 0.4 and 0. Even though the real facial features have clearly been missed, the subjective impression of the detection is not so bad. At least, in all cases both eyes and mouth lie within the framed part of the image.

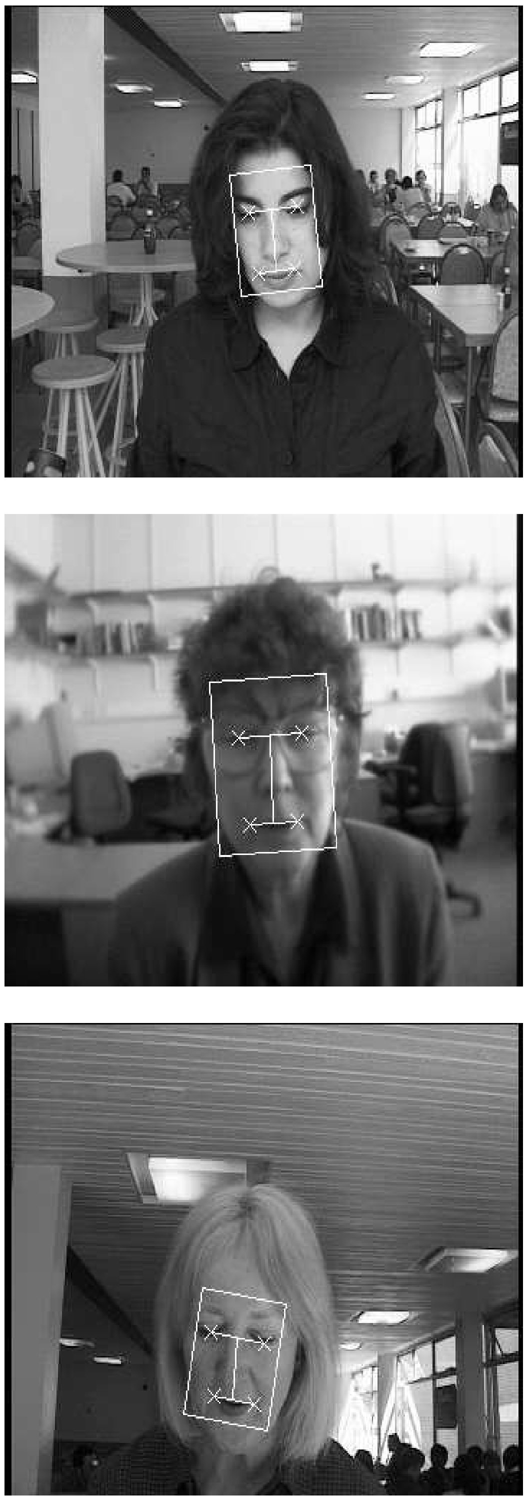


Figure 4.11: Examples of good detections

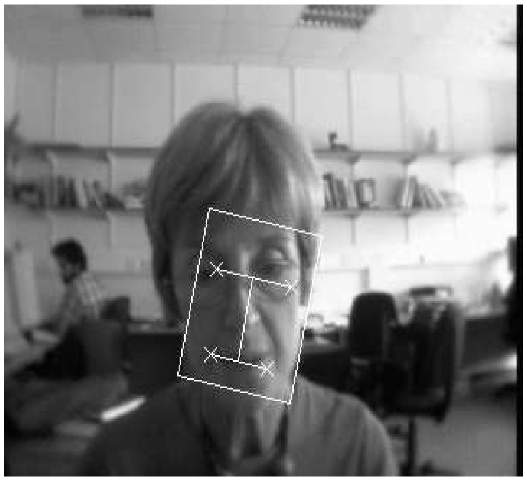
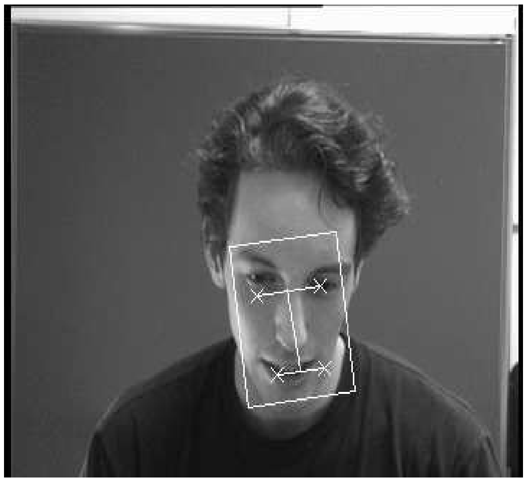
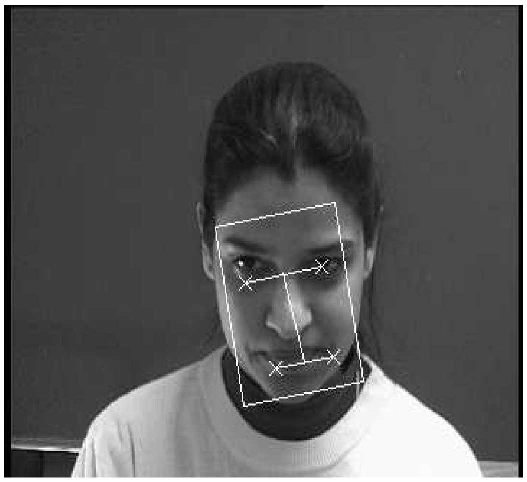




Figure 4.12: Examples of poor detections.

Chapter 5

Conclusions and Future Work

5.1 Discussion

A method for detecting and localizing human faces based on affine invariant local feature face models and evidence accumulation has been presented. It has been designed to work in the context of high resolution grayscale images with variability in pose, background and lightning conditions and with possible occlusions due to different hairstyles, glasses, facial hair, etc. As the method can be classified midway between the template-based and pure feature-based methods, it benefits from the advantages of both approaches.

By detecting and labelling local parts of the face using a template-based technique (in this case, an SVM), robustness against illumination changes is achieved. Variation in pose and facial expression is dealt with by projecting and classifying the these local parts in an affine invariant space, where all differences due to affine transformations (rotation, scaling, translation, squeezing...) are removed. Finally, the problem of occlusions is overcome by not requiring the full set of facial features to be detected in order to form a hypothesis and check it. Using evidence accumulation, it is possible to detect a face with some missing features, provided that the others give enough evidence. Also, different face models have been used, covering all possible eventualities (i.e., no mouth features, no nose features, or no eye features detected).

The system performs quite well: it obtains a 80% of good detections. However, even in detections obtaining a high score, the precision is not as good as it should be. This is mainly due to the local feature detection scheme. It is reported to produce around 30 false alarms for each good classification. This leads to having many missclassified features, and many other non-features at all which are labelled as if they were.

Furthermore, some of the false alarms produced by the local classifier are *inliers*, i.e., false alarms appearing very close to a real feature. This kind of false alarms are very difficult to reject. Since they lie near a good location, the small difference can be assumed by the inner variability of the model, and hypotheses can be formed with them. The final validation step should be able to correct this problem. But, since it consists of a simple cross-correlation (while still a valid method for rejecting improbable background false positive detections), it sometimes selects a worse score-achieving hypothesis before another better one.

5.2 Future work

As it has been explained in the previous section, most of the problems come from the template-based steps of the algorithm (i.e., the local feature classifier, and the final validation process).

- Concerning the local classifier, it has been said that it produces too much false alarms. Obviously, a better local feature classifier system with a better ROC curve is expected to improve the overall performance of the system, as it would significantly reduce the amount of possible candidates and hypothesis right from the start.

- About the validation procedure, its main problem is its simplicity. A cross-correlation with a mean face, while being able to distinguish a face pattern from a background one, it is not well-suited to choose among a set of candidate patches differing only by a small shifting and/or rotation. Though at cost of losing some computational efficiency, the system performance would be enhanced if using a well-trained, template-based method (such as those presented in Sec. 2.1) as the final checking procedure.

Although the proposed method has shown some robustness, there is still work to be done. A really robust face detection system should be effective under full variation in:

- lighting conditions,
- orientation, pose, and partial occlusion,
- facial expression, and
- presence of glasses, facial hair, and a variety of hair styles.

Bibliography

[1] G. Antonini, V. Popovici, and J.-P. Thiran, "Independent Component Analysis and Support Vector Machine for Face Feature Extraction," Signal Processing Institute, Swiss Federal Institute of Technology Lausanne. Lausanne, 2003.

[2] S. Bengio, F. Bimbot, J. Mariéthoz, V. Popovici, F. Porée, E. Bailly-Baillière, G. Matas, and B. Ruiz, "Experimental Protocol on the BANCA Database," IDIAP-RR 05, IDIAP, 2002.

[3] C.J.C. Burges. "A Tutorial on Support Vector Machines for Pattern Recognition. *Data Mining and Knowledge Discovery*, 2(2), pp.121-167, 1998.

[4] M.C. Burl, T.K. Leung, and P. Perona, "Face Localization via Shape Statistics," *Proc. First Int'l Workshop Automatic Face and Gesture Recognition*, pp. 154-159, 1995.

[5] R. Cipolla and A. Blake, "The Dynamic Analysis of Apparent Contours," *Proc. Third IEEE Int'l Conf. Computer Vision*, pp 616-623, 1990.

[6] A.J. Colmenarez and T.S. Huang, "Face Detection with Information-based Maximum Discrimination," *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, pp. 782-787, 1997.

[7] T. Cover and J. Thomas, *Elements of Information Theory*, Wiley Interscience, 1991.

[8] A.P. Dempster, N.M. Laird, and D.B. Rubin, "Maximum Likelihood from Incomplete Data via the EM Algorithm," *J. Roy. Statist. Soc.* **39**, 1977, pp. 1-39.

[9] R.A. Fisher, "The Use of Multiple Measurements in Taxonomic Problems," *Ann. Eugenics* **7**, 1936, pp. 179-188.

[10] V. Govindaraju, “Locating Human Faces in Photographs,” *Int. J. Comput. Vision* **19**, 1996.

[11] G.H. Granlund, “In Search of a General Picture Processing Operator,” *Computer Graphics and Image Processing*, 8, pp. 155-173, 1978.

[12] C. Harris and M. Stephens, “A Combined Corner and Edge Detector,” *Alvey Vision Conference*, pp. 147-151, 1988.

[13] B. Heisele, P. Ho, and T. Poggio, “Face Recognition with Support Vector Machines: Global versus Component-based Approach,” Massachusetts Institute of Technology, Center for Biological and Computational Learning.

[14] E. Hjelmas and B.K. Low, “Face Detection: A Survey,” Computer Vision and Image Understanding, 83, pp. 236-274, 2001.

[15] J. Huang, S. Gutta, and H. Wechsler, “Detection of Human Faces Using Decision Trees,” *Proc. Second Int'l Conf. Automatic Face and Gesture Recognition*, pp. 248-252, 1996.

[16] D. Hutenlocher, G. Klanderman, and W. Rucklidge, “Comparing Images Using the Hausdorff Distance,” *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 15, pp. 850-863, 1993.

[17] A. Hyvaerinen and E. Oja, “Independent Component Analysis: Algorithms and Applications,” *Neural Networks*, 13(4-5), pp. 411-430, 2000.

[18] T. Jebara and A. Pentland, “Parameterized Structure From Motion for 3D Adaptive Feedback Tracking of Faces,” *Proc. of CVPR'97*, 1997.

[19] S.H. Jeng, H.Y.M. Liao, C.C. Han, M.Y. Chern, and Y.T. Liu, “Facial Feature Detection Using Geometrical Face Model: An Efficient Approach,” *Pattern Recog.* **31**, 1998.

[20] D.G. Kendall, “Shape Manifolds, Procrustean Metrics, and Complex Projective Shapes,” *Bull. London Math. Soc.*, vol. 16, pp. 81-121, 1984.

[21] T. Kohonen, *Self-Organizing Maps*, Springer-Verlog, Berlin, 1995.

[22] C. Kotropoulos and I. Pitas, “Rule-based Face Detection in Frontal Views,” in *Proc. Int. Conf. on Acoustic, Speech and Signal Processing*, 1997.

[23] T.K. Leung, M.C. Burl, and P.Perona, "Finding Faces in Cluttered Scenes Using Random Labeled Graph Matching," *Proc. of The Fifth International Conference on Computer Vision*, 1995.

[24] T.K. Leung, M.C. Burl, and P. Perona, "Probabilistic Affine Invariants for Recognition," *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, pp. 678-684, 1998.

[25] M.S. Lew, "Information Theoretic View-based and Modular Face Detection," *Proc. Second Int'l Conf. Automatic Face and Gesture Recognition*, pp. 198-203, 1996.

[26] N. Littlestone, "Learning Quickly when Irrelevant Attributes Abound: A New Linear-threshold Algorithm," *Machine Learning*, vol. 2, pp. 285-318, 1988.

[27] M.M. Loève, *Probability Theory*, Van Nostrand, Princeton, 1955.

[28] K.V. Mardia and I.L. Dryden, "Shape Distributions for Landmark Data," *Advanced Applied Probability*, vol. 21, pp. 742-755, 1989.

[29] B. Moghaddam and A.Pentland, "Face Recognition Using View-based and Modular Eigenspaces," *Automatic Systems for the Identification of Humans, SPIE, 1994,* Vol. 2277.

[30] B. Moghaddam and A. Pentland, "Probabilistic Visual Learning for Object Representation," *IEEE Trans. Pattern Anal. Mach. Intell.* **19**(1), 1997.

[31] A.V. Nefian and M.H. Hayes III, "Face Detection and Recognition Using Hidden Markov Models," *Proc. IEEE Int'l Conf. Image Processing*, vol. 1, pp. 141-145, 1998.

[32] E. Osuna, R. Freund, and F. Girosi, "Training Support Vector Machines: An Application to Face Detection," *IEEE Proc. of Int. Conf. on Computer Vision and Pattern Recognition, 6, 1997.*

[33] H.J. Park and H.S. Yang, "Invariant Object Detection Based on Evidence Accumulation and Gabor Features," *Pattern Recognition Letters 22*, pp. 869-882.

[34] A. Pentland, B. Moghaddam, and T. Strarner, "View-based and Modular Eigenspaces for Face Recognition," *IEEE Proc. of Int. Conf. on Computer Vision and Pattern Recognition, 1994.*

[35] T.V. Pham and M. Worring, “Face Detection Methods: A Critical Evaluation,” *ISIS Technical Report 11*, Informatics Institute, University of Amsterdam, 2000.

[36] P.J. Phillips, H. Moon, S.A. Rizvi, and P.J. Rauss, “The FERET Evaluation Methodology for Face-recognition Algorithms,” *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 22, no. 10, pp. 1090-1134, Oct. 2000.

[37] S. Pigeon and L. Vandendrope, “The M2VTS Multimodal Face Database,” *Proc. First Int'l Conf. Audio- and Video-based Biometric Person Authentication*, 1997.

[38] J.R. Quinlan, *C4. 5: Programs for Machine Learning.* Kluwer Academic, 1993.

[39] L.R. Rabiner and B.-H. Jung, *Fundamentals of Speech Recognition.* Prentice Hall, 1993.

[40] A. Rajagopalan, K. Kumar, J. Karlekar, R. Manivasakan, M. Patil, U. Desai, P. Poonacha, and S. Chaudhuri, “Finding Faces in Photographs,” *Proc. Sixth IEEE Int'l Conf. Computer Vision*, pp. 640-645, 1998.

[41] D. Roth, M.-H. Yang, and N. Ahuja, “A SNoW-based Face Detector,” *Advances in Neural Information Processing Systems 12 (NIPS 12)*, MIT Press, Cambridge, MA, 2000.

[42] H.A. Rowley, S. Baluja, and T. Kanade, “Neural Network-based Face Detection,” *IEEE Trans. Pattern Anal. Mach. Intell.* **20**, January 1998, 23-38.

[43] H.A. Rowley, S. Baluja, and T. Kanade, “Rotation Invariant Neural Network-based Face Detection,” *IEEE Intl. Conf. on Computer Vision and Pattern Recognition, 1998*, pp. 38-44.

[44] E. Saber and A.M. Tekalp, “Frontal-view Face Detection and Facial Feature Extraction Using Color, Shape and Symmetry Based Cost Functions,” *Pattern Recognition Letters*, vol. 17, no. 8, pp. 669-680, 1998.

[45] F. Samaria, “Face Recognition Using Hidden Markov Models,” PhD thesis, Univ. of Cambridge, 1994.

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

[47] C. Schmid, R. Mohr, and C. Bauckhage, "Evaluation of Interest Point Detectors," *International Journal of Computer Vision*, vol. 37, 2, pp. 151-172, 2000.

[48] H. Schneiderman and T. Kanade, "A Statistical Model for 3D Object Detection Applied to Faces and Cars," *IEEE Conference on Computer Vision and Pattern Recognition,* 2000.

[49] H. Schneiderman and T. Kanade, "Probabilistic Modelling of Local Appearance and Spatial Relationships for Object Recognition," *IEEE Conference on Computer Vision and Pattern Recognition, 6, 1998.*

[50] L. Sirovich and M. Kirby, "Low-dimensional Procedure for the Characterization of Human Faces," *J.Opt.Soc.Amer*: **4**, 1987, 519-524.

[51] K. Sobottka and I. Pitas, "Face Localization and Feature Extraction Based on Shape and Color Information," *Proc. IEEE Int'l Conf. Image Processing*, pp. 483-486, 1996.

[52] G. Stockman, "Object Recognition and Localization via Pose Clustering," CVGIP 40, 361-387, 1987.

[53] Q.B. Sun, W.M. Huang, and J.K. Wu, "Face Detection Based on Color and Local Symmetry Information," *Proc. of the Third International Conference on Automatic Face and Gesture Recognition*, pp. 130-135, Nara, Japan, 1998.

[54] K.K. Sung, "Learning and Example Selection for Object and Pattern Detection," PhD thesis, Massachusetts Inst. of Technology AI Lab, 1994.

[55] K.K. Sung and T. Poggio, "Learning Human Face Detection in Cluttered Scenes," V. Hlavac and R. Sara, editors, *Computer Analysis of Images and Patterns*, pp. 432-439. Springer, Berlin, 1995.

[56] K.K. Sung and T. Poggio, "Example-based Learning for View-based Human Face Detection," *IEEE PAMI*, 20(1):39-51, 1998.

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

[58] P. Viola and M. Jones, "Robust Real-time Object Detection," *Second International Workshop on Statistical and Computational Theories of Vision - modelling, Learning, Computing, and Sampling.* Vancouver, Canada, July 2001.

[59] G. Wei and I.K. Sethi, "Face Detection for Image Annotation," *Pattern Recognition Letters*, 20(11):1313-1321, 1999.

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

[61] M.-H. Yang, N. Ahuja, and D.J. Kriegman, "Face Detection Using Mixtures of Linear Subspaces," in *Proceedings Fourth IEEE International Conference on Automatic Face and Gesture Recognition, 2000.*

[62] M.-H. Yang, D.J. Kriegman, and N. Ahuja, "Detecting Faces in Images: A Survey," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, Vol. 24, NO. 1, pp. 34-58, January 2002.

[63] G. Yang and T.S. Huang, "Human Face Detection in a Complex Background," *Pattern Recog.* **27**, 1994, 53-63.

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

[65] K.C. Yow and R. Cipolla, "Enhancing Human Face Detection Using Motion and Active Contours," *Proc. Third Asian Conf. Computer Vision*, pp. 515-522, 1998.

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