3D POSE ESTIMATION AND NORMALIZATION FOR FACE RECOGNITION

Tony S. Jebara

Department of Electrical Engineering
McGill University

May 1996

A Thesis submitted to the Faculty of Graduate Studies and Research
in partial fulfilment of the requirements of the degree of
Bachelor of Engineering

Abstract

Automatic face recognition has been a difficult problem in the field of computer vision for many years. Robust face recognition requires the ability to recognize identity despite many variations in appearance the face can have in a scene. We propose preceding recognition with a hierarchical detection system capable of searching images for human faces efficiently and with invariance to position, deformation, illumination, scale, and 3D pose. The detection and localization of a face are used to perform 3D normalization to the face prior to 2D linear recognition.

Biologically motivated, low-level attentional mechanisms are applied at multiple scales to identify possible face regions. The facial contour is then estimated by computing symmetric enclosure and is used to guide the search for feature points within the face. Symmetric blob detection, limb extraction and signature analysis are used to locate the eyes, mouth and nose of each individual. A database of 3D range data of human heads allows us to align a 3D model to the coordinates of the detected feature points in the input image. The intensity image's textural representation of the face is mapped onto the 3D range data, thereby segmenting the face from the image. The 3D model is then rotated into a frontal view to synthesize a frontal "mug-shot" of the individual. Lighting and shading variations are corrected by histogram fitting. Once fully normalized, the image is projected into a low-dimensional subspace via Karhunen-Loeve Decomposition to compress the data and to verify detection. The resulting low-dimensional vector description is matched against a database using simple distance measures to determine the face's identity as one of the previously identified training examples. Due to the computational efficiency of the hierarchical detection scheme and the initial step of applying simple attentional mechanisms, tracking faces from a video source could be achieved.

Acknowledgements

Whether you think you can or think you can't – you are right. -Henry Ford

Thank you, Professor Martin Levine for making me think that I can, for sharing your extensive knowledge of the field, and for patiently reminding me each time I got ahead of myself to slow down and reflect.

Thank you, Thierry Baron, Marc Bolduc, Stephen Benoit, Don Bui, Franco Callari, Michael Daum, James Elder, Michael Kelly, Gal Sela, Gilbert Soucy, Kenong Wu and John Zelek for your advice, insight and technical expertise. Thanks to Ornella Cavaliere for invaluable advice and support. I also extend my thanks to Professor Peter Caines and Professor George Zames for their help.

Mom, Dad, and Carine, this is for you.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

Introduction

Automatic face detection and recognition has been a difficult problem in the field of computer vision for several years. Although humans perform the task in an effortless manner, the underlying computations within the human visual system are of tremendous complexity. The seemingly trivial task of finding and recognizing faces is the result of millions of years of evolution and we are far from fully understanding how the brain performs it. Furthermore, the ability to find faces visually in a scene and recognize them is critical for humans in their everyday activities. Consequently, the automation of this task would be useful for many applications including security, surveillance, gaze-based control, affective computing, speech recognition assistance, video compression and animation. However, to date, no complete solution has been proposed that allows the automatic recognition of faces in real (un-contrived) images. We wish to develop a vision system which would permit automatic machine-based face detection and recognition in uncontrolled environments. This thesis describes the theoretical foundations of such a system, its implementation and its evaluation.

This introduction begins with an outline of the main issues and constraints that need to be addressed in face recognition. Subsequently, a survey is presented which outlines the face recognition research that has been performed to-date and the strengths and weaknesses of a variety of machine-based systems. We then describe our proposed approach for the face recognition problem. Finally, an overview of the structure of this thesis is presented.

1. Fundamental Issues in Face Recognition

Robust face recognition requires the ability to recognize identity despite many variations in appearance that the face can have in a scene. The face is a 3D object which is illuminated from a variety of light sources and surrounded by arbitrary background data (including other faces). Therefore, the appearance a face has when projected onto a 2D image can

vary tremendously. If we wish to develop a system capable of performing non-contrived recognition, we need to find and recognize faces despite these variations. In fact, 3D pose, illumination and foreground-background segmentation have been pertinent issues in the field of computer vision as a whole.

Additionally, our detection and recognition scheme must also be capable of tolerating variations in the faces themselves. The human face is not a unique rigid object. There are billions of different faces and each of them can assume a variety of deformations. Inter-personal variations can be due to race, identity, or genetics while intra-personal variations can be due to deformations, expression, aging, facial hair, cosmetics and facial paraphernalia.

Furthermore, the output of the detection and recognition system has to be accurate. A recognition system has to associate an identity or name for each face it comes across by matching it to a large database of individuals. Simultaneously, the system must be robust to typical image-acquisition problems such as noise, video-camera distortion and image resolution.

Thus, we are dealing with a multi-dimensional detection and recognition problem. One final constraint is the need to maintain the usability of the system on contemporary computational devices ($\approx$100 MIPS). In other words, the processing involved should be efficient with respect to run-time and storage space.

2. Current Vision Systems for Face Recognition

Research in intensity image face recognition generally falls into two categories [**7**]: holistic (global) methods and feature-based methods. Feature-based methods rely on the identification of certain fiducial points on the face such as the eyes, the nose, the mouth, etc. The location of those points can be determined and used to compute geometrical relationships between the points as well to analyze the surrounding region locally. Thus, independent processing of the eyes, the nose, and other fiducial points is performed and then combined to produce recognition of the face. Since detection of feature points precedes the analysis, such a system is robust to position variations in the image. Holistic methods treat the image data simultaneously without attempting to localize individual points. The face is recognized as one entity without explicitly isolating different regions in the face. Holistic techniques utilize statistical analysis, neural networks and transformations. They also usually require large samples of training data. The advantage of holistic methods is that they utilize the face as a whole and do not destroy any information by exclusively processing

only certain fiducial points. Thus, they generally provide more accurate recognition results. However, such techniques are sensitive to variations in position, scale and so on which restricts their use to standard, frontal mug-shot images [**7**].

Early attempts at face recognition were mostly feature-based. These include Kanade's [**19**] work where a series of fiducial points are detected using relatively simple image processing techniques (edge maps, signatures, etc.) and their Euclidean distances are then used as a feature vector to perform recognition. More sophisticated feature extraction algorithm were proposed by Yuille, Cohen and Hallinan [**45**]. These use deformable templates that translate, rotate and deform in search of a best fit in the image. Often, these search techniques use a knowledge-based system or heuristics to restrict the search space with geometrical constraints (i.e. the mouth must be between the eyes) [**10**]. Unfortunately, such energy minimization methods are extremely computationally expensive and can get trapped in local minima. Furthermore, a certain tolerance must be given to the models since they can never perfectly fit the structures in the image. However, the use of a large tolerance value tends to destroy the precision required to recognize individuals on the basis of the model's final best-fit parameters. Nixon proposes the use of Hough transform techniques to detect structures more efficiently [**31**]. However, the problem remains that these detection-based algorithms need to be tolerant and robust and this often makes them insensitive to the minute variations needed for recognition. Recent research in geometrical, feature-based recognition [**9**] reported 95% recognition. However, the 30 features points used for each face were manually extracted from each image. Had some form of automatic localization been used, it would have generated poorer results due to lower precision. In fact, even the most precise deformable template matching algorithms such as Roeder's [**40**] and Colombo's [**8**] feature detectors generally have significant errors in detection. This is also true for other feature detection schemes such as Reisfeld's symmetry operator [**37**] and Graf's filtering and morphological operations [**15**]. Essentially, current systems for automatic detection of fiducial points are not accurate enough to obtain high recognition rates exclusively on the basis of simple geometrical statistics of the localization.

Holistic techniques have recently been popularized and generally involve the use of transforms to make the recognition robust to slight variations in the image. Rao [**36**] develops an iconic representation of faces by transforming them into a linear combination of natural basis functions. Manjunath [**28**] uses a wavelet transform to simultaneously extract feature points and to perform recognition on the basis of their Gabor wavelet jets. Such

techniques perform well since they do not exclusively compute geometric relationships between fiducial points. Rather, they compare the jets or some other transform vector response around each fiducial point. Alternate transform techniques have been based on statistical training. For example, Pentland [**44**] uses the Karhunen-Loeve decomposition to generate the optimal basis for spanning mug-shot images of human faces and then uses the subsequent transform to map the faces into a lower-dimensional representation for recognition. This technique has also been applied by Akamatsu [**1**] on the Fourier-transformed images instead of the original intensity images. Recent work by Pentland [**32**] involves modular eigenspaces where the optimal intensity decomposition is performed around feature points independently (eyes, nose and mouth). Pentland [**29**] has also investigated the application of Karhunen-Loeve decomposition to statistically recognize individuals on the basis of the spectra of their dynamic Lagrangian warping into a standard template. These transform techniques have yielded very high recognition rates and have quickly gained popularity. However, these non-feature based techniques do not fare well under pose changes and have difficulty with natural, un-contrived face images.

In most holistic face recognition algorithms, the face needs to be either segmented or surrounded by a simple background. Furthermore, the faces presented to the algorithms need to be roughly frontal and well-illuminated for recognition to remain accurate. This is due to the algorithms' dependence on fundamentally linear or quasi-linear analysis techniques (Fourier, wavelet, Karhunen-Loeve decompositions, etc. are linear transformations). Thus, performance degrades rapidly under 3D orientation changes, non-linear illumination variation and background clutter (i.e. large, non-linear effects).

3. The Proposed Approach

We propose a hybrid system that combines the robust detection of feature points with a holistic and precise linear transform analysis of the face data. The detection of feature points uses a robust model capable of detecting individual features despite a wide range of translations, scale changes, 3D-pose changes and background clutter. This allows us to locate faces in an arbitrary, un-contrived image. Since we wish to utilize linear transform techniques, however, a consistent, normalized frontal mug-shot view of the face is needed. Thus, we propose synthesizing the required mug-shot view from the one detected in the original image. This is performed by inverting the 3D projection of the original face in the image and re-mapping it into frontal view via a deformable 3D model. Then, we perform illumination correction and segmentation to obtain an ideal mug-shot view of the

individual in question. At this stage, we can safely apply holistic linear transform techniques (namely, the Karhunen-Loeve decomposition) and use the vector-representation of the face to recognize its identity.

Thus, our 3D normalization technique acts as a bridge between feature-detection and holistic face recognition. When guided with the results of the feature-detection, normalization removes the non-linear variations in the image and generates a face that is aligned and ready for recognition. Thus, the holistic face recognition stage is preceded by feature-based detection and normalization for increased robustness. In fact, the bridge is bidirectional since the feature-detection can also be complemented by the holistic face recognition stage. The Karhunen Loeve decomposition has demonstrated the ability provide a statistical measure of how face-like an image vector seems [44]. This measure can be fed back to the feature-detection algorithm to inform it if it has poorly localized the face to begin with. This will force the feature-detection to keep searching for the face and improve its detection results. Thus, holistic face recognition can also be used to assist the feature-detection stage and, ultimately, improve its precision.

Figure 1.1 depicts the interaction between the two stages: feature detection and holistic recognition. Although feature localization is robust, when used alone it is too insensitive for recognition. Although holistic face recognition is precise, its use of linear transformation is not robust to large non-linear face variations. Thus, combining the two approaches provides a superior overall system.

4. Structure of the Thesis

In Chapter 2, the thesis presents an overview of three important elements of vision: perceptual contrast, symmetry and scale. The significance of these concepts in human vision is described and their usefulness in image understanding is introduced. A palette of basic, biologically-motivated image processing tools that rely on these concepts is then defined for later use. These low level tools include edge detectors, symmetry operators and attentional mechanisms.

Chapter 3 describes a detection algorithm that utilizes biologically motivated low-level operations to find feature points on the face. A hierarchical, coarse-to-fine search is described for localizing the face. A method for the coarse detection of possible face regions in the image is outlined. We then propose a technique for the estimation of facial contour. Subsequently, we describe techniques for finding the eyes, the mouth and the nose in the face. We then describe the localization of the iris in the eye if it is visible in the image.

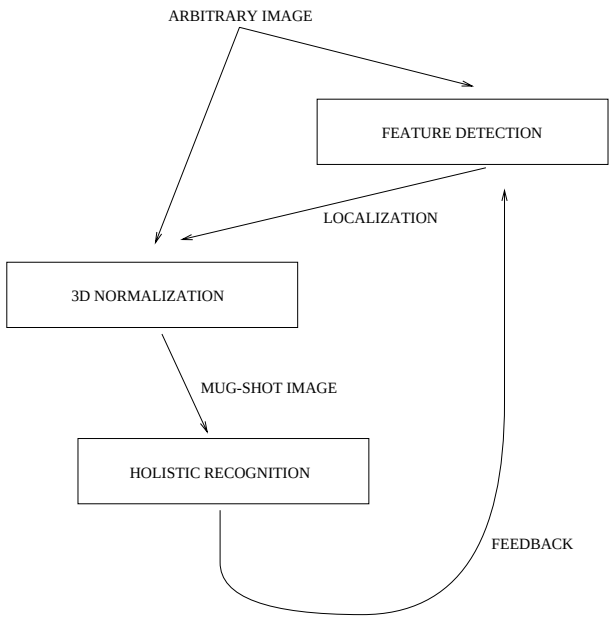


FIGURE 1.1. 3D Normalization as a Bridge Between Feature Detection and Face Recognition

Chapter 4 describes the normalization procedure used to generate high-quality mug-shot images. It also discusses the linear transformations used to recognize them and to optimize their localization. We define the 3D deformable model used for normalization and the pose estimation computations that generate frontal, mug-shot images. The illumination correction algorithm is also presented. We then describe the Karhunen-Loeve decomposition and its use to measure the "faceness" of an image. We then describe how we improve the localization of detected feature points by maximizing the "faceness" value. Finally, we describe the use of the Karhunen-Loeve decomposition to recognize the identity of the subject in the image.

The details of the implementation and the output of the algorithm are covered in Chapter 5. Performance analysis and sensitivity analysis is performed to test the algorithm.

Finally, Chapter 6 concludes the thesis with a summary of the work, its contributions and the direction of future research and improvements.

CHAPTER 2

Perceptual Contrast, Symmetry and Scale

For a computer based face recognition system to succeed, it must detect faces and the features that compose them despite variations each face has in the current scene [**30**]. Faces can be anywhere in an image, at a variety of sizes, at a variety of poses and at a variety of illuminations. Although humans quickly detect the presence and location of faces and facial features from a photograph, automatic machine detection of such objects involves a complex set of operations and tests. It is uncertain exactly how humans detect faces in an image, however we can attempt to imitate the perceptual mechanisms humans seem to employ. We begin by defining and discussing the significance of contrast, symmetry and scale in human vision. This will serve as a basis for the biologically motivated computational tools that we will be using. We then discuss our technique for the computational extraction of contrast information. The implementation of multi-scale analysis structure is then defined. Finally, two computational tools for obtaining information on symmetry are introduced: the symmetry transform and the selective symmetry detector.

1. Biological and Psychological Motivation

Some psychological research has proposed that certain parts of an image attract our attention more than others [**27**]. Through visuo-motor experiments, research has demonstrated that fixation time and attentional resources are generally allocated to portions in a given scene which are visually interesting. This "degree of perceptual significance" of regions in a given image allows a human observer to almost automatically discriminate between insignificant regions in a scene and interesting ones which warrant further investigation. The ability to rapidly evaluate the level of interest in parts of a scene could benefit a face recognition system in a similar way: by reducing its search space for possible human faces. Instead of exhaustively examining each region in an image for a face-like structure, the system would only focus computational resources upon perceptually significant objects.

It has been shown that three of the important factors in evaluating perceptual significance are contrast, symmetry and scale [**20**].

Neurophysiological experiments demonstrate that the retina performs filtering which identifies contrast spatially and temporally. For instance, center surround cells at the retinal processing stage are triggered by local spatial changes in intensity (contrast) [**39**]. In psychological tests, humans detect high contrast objects more readily than, say, objects with a similar colour to their background. A significant change in spatial intensity is referred to as an edge, boundary or a contour. Further research has shown that response to spatial contrast also varies with time [**24**]. A moving edge, for example, triggers a strong response at the retinal level [**24**]. Thus, contrast or intensity changes over space and time are important in vision and this has lead to the development of edge detectors and motion detectors.

Another property in estimating perceptual importance is symmetry [**37**]. The precise definition of symmetry in the context of attentional mechanisms is different from the intuitive concept of symmetry. Symmetry, here, represents the symmetric enclosure or the approximate encirclement of a region by contours. The appropriate arrangement of edges which face each other to surround a region attracts the human eye to that region. Furthermore, the concept of enclosure is different from the mathematical sense of perfect closure since humans will still perceive a sense of enclosure despite gaps in boundaries that surround the region [**20**].

Scale is also a feature which determines a region's relative importance in a scene [**4**]. It is progressively easier to detect a foreground object if it occupies a greater and greater area in our field of view. Generally, as an object is enlarged, it increases in relative importance.

2. Low Level Filtering for Perceptually Interesting Objects

A computational technique is needed which can combine the effects of contrast, symmetry and scale to find the set of the interesting regions in an image. An example of the possible output of such an algorithm would be a collection of points defining circular regions of a certain radius (or scale) which exhibit perceptual importance. A mask or filter is needed which can be quickly applied locally (topographically) over the whole image at multiple scales. The output of the mask would be a perceptual significance map which measures the level of contrast and symmetric enclosure of the image region overlapped by the filter. To detect large perceptually significant objects first, this mask would be applied first at large scales (i.e., with a relatively large mask) and then at progressively smaller

ones. Such a filter would provide us with an efficient attentional mechanism for quickly fixating further face-recognition computational resources only on interesting regions.

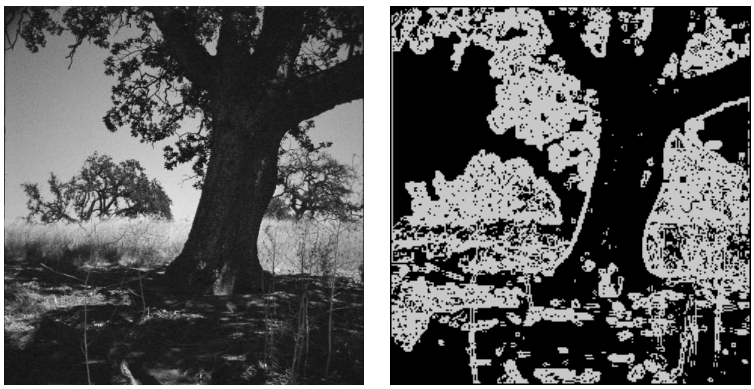
3. Edge Detection

The extraction of edges or contours from a two dimensional array of pixels (a gray-scale image) is a critical step in many image processing techniques. A variety of computations are available which determine the magnitude of contrast changes and their orientation. Extensive literature exists documenting the available operators and the post-processing methods to modify their output. A trade-off exists, though, between efficiency and quality of the edge detection. Fast and simple edge detection can be performed by filters such as the popular Sobel operator [**25**] which requires the mere convolution of a small kernel (3×3 pixels) over the image. Alternatively, more computationally intensive contour detection techniques are available such as the Deriche [**11**] or Canny [**6**] method. These detectors require that a set of parameters be varied to detect the desired scale and curvature of edges in the image. It is necessary to compare the simple Sobel detector and the complex Deriche-type detectors before selecting the edge detection scheme of preference.

3.1. The Sobel Operator The 3×3 Sobel operator acts locally on the image and only detects edges at small scales. If an object with a jagged boundary is present, as shown in Figure 2.1(a), the Sobel operator will find the edges at each spike and twist of the perimeter as in Figure 2.1(b). The operator is sensitive to high frequency noise in the image and will generate only local edge data instead of recovering the global structure of a boundary. Furthermore, smooth transitions in contrast that occur over too large a spatial scale to fit in the 3×3 window of the Sobel operator will not be detected.

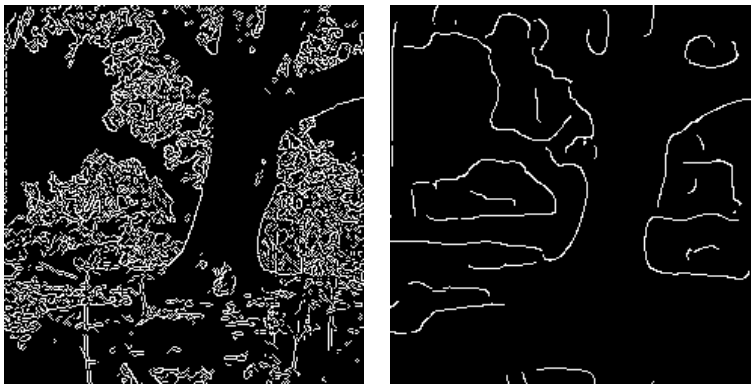
3.2. Deriche Edge Detection The Deriche output, on the other hand, can be adjusted with the α scale parameter to filter out high frequency noise and pixelization from the image by linking adjacent edges into long, smooth, continuous contours. This allows the edge map to reflect the dominant structures in the image. The effect of small and large α (the scale parameter) is shown in Figure 2.2(a) and Figure 2.2(b). Furthermore, the computation is not limited to a small window and can find edges which change gradually. Thus, the outline of the trees as separate whole objects is found instead of the outline of the leaves.

Despite its complexity, the Deriche technique, as with all other edge detectors, has its limits. While a human is capable of isolating objects and uses contextual knowledge of the scene to determine boundaries, the Deriche operator sometimes confuses the edges of



(a) (b)

FIGURE 2.1. Sobel edge detection. (a) The original intensity image. (b) The Sobel edge map.



(a) (b)

FIGURE 2.2. Deriche edge detection at multiple scales (a) Deriche edge map at a small scale. (b) Deriche edge map at a large scale.

nearby objects and links them into a single contour. The detection of erroneous phantom edges is also another problem as pointed out by Kelly and Levine [**21**].

The most significant disadvantage in using the Deriche operator and other complex edge detection schemes is in their computational cost. The Deriche extraction of edges from an image can take orders of magnitude more time when compared to the Sobel operator. If a real-time system is desired, image processing must be performed in fractions of a second. Deriche edge detection would simply be too time consuming on contemporary workstations.

(a) (b) (c)

FIGURE 2.3. Typical output from Sobel edge detection. (a) Input intensity image. (b) Sobel gradient magnitude map. (c) Sobel phase map.

3.3. Edge Data Enhancement Edge detection can be followed by further processing to enhance the output or to modify it. The techniques described below will be used on the output of the Sobel edge detection operation. The Sobel operation begins with an intensity image (Figure 2.3(a)) and produces a gradient magnitude map (Figure 2.3(b)) and an edge phase map (Figure 2.3(c)).

Thresholding: If a binary (i.e., black and white) version of the gradient map is desired, it can be obtained by thresholding [**14**]. All gradient values whose magnitudes are less than a threshold will be set to 0 and all gradient magnitudes greater than a threshold will be set to the maximum edge value (255). Thus, the maximum image contrast is compressed to 1 bit. One strategy for selecting the threshold in question is to first search for the strongest edge magnitude in the image region of interest. Once the peak edge magnitude is found, the threshold is computed as a percentage of this peak (e.g, 10%). A thresholded version of Figure 2.3(a) is Figure 2.4(a).

Non-Maximal Suppression: Non-maximal suppression nullifies the gradient value at a pixel if its gradient magnitude is non-maximal vis-a-vis neighbouring gradient magnitudes along the perpendicular to the pixel's gradient orientation. The result is a thinned edge map, as shown in Figure 2.4(b) where only the dominant edges are present.

Square Root of Edge Magnitude: It may be necessary to adjust the edge magnitudes returned by the Sobel operation to change the significance of the contrast in a computation. In other words, we may want to re-map the contrast levels to emphasize the weak edges in an image. In some subsequent processing steps, the

(a) (b)

FIGURE 2.4. Post-Processing Sobel edge detection. (a) Thresholded gradient magnitude map with threshold at 10% of peak edge magnitude. (b) Gradient map after non-maximal suppression.

magnitude of the edge at each point is replaced by the square-root of the magnitude to attenuate the effect of contrast.

4. Multi-Scale Analysis

We note at this point the significance of scale and the scalability of certain types of edge detection. Since structures and objects in an input image can have different sizes and resolutions, most spatial operators (for edge extraction or further processing) will require scalability.

As demonstrated previously, the scalability of the Deriche edge detection operation makes it more flexible than the highly local Sobel operator. The objects in an input image and the image itself can have different sizes and resolutions so a scalable operator is necessary to search for edges at different scales. Kelly and Levine [**21**] have approached edge detection by applying a Deriche-like operator over many scales. By changing the operator's size, several edge maps (one for each scale) are obtained and subsequently processed in parallel. Elder [**12**] proposes yet another technique wherein a single edge map is produced by computing the optimal scale of edge detection for each position of the input image and then performing scale-adaptive edge detection. In other words, the scale of the operator is varied appropriately at each position in the image.

However, scalable edge detection is too complex to be performed rapidly. Additionally, subsequent operators (i.e., for computing symmetric enclosure) might also be computationally inefficient when they are scaled. One "solution" to this issue is to scale the input image instead of enlarging the operators themselves. Thus, a pyramid is formed from a single

input image by reducing it by various scale factors. The resulting set of scaled images can then be processed by an operator of fixed size (such as the Sobel operator).

Given an image I(i,j) of dimensions $M \times N$, a scaled version, $I_s(i,j)$, of the image at a scale factor of s can be obtained with dimensions $[M/s] \times [N/s]$ by either subsampling the image or subaveraging it. The following describes the operations required for scaling by an integer factor s (although non-integer scaling is possible as well).

Subsampling: Subsampling involves selecting a sample from each neighbourhood of pixels of size $s \times s$. Each sample will be used as a pixel value in the scaled image. Subsampling assumes that the sample appropriately represents its neighbourhood and that the image is not dominated by high-frequency data. This process only requires $[M/s] \times [N/s]$ computations.

Subaveraging: Subaveraging is similar to subsampling except that the sample taken from each neighbourhood of pixels has an intensity that is the average intensity in the neighbourhood. Thus, the original image undergoes low-pass filtering before being sampled so that the scaled image approximates the original optimally even if it has high frequency data. To avoid possible errors when scaling high frequency components in the image, we choose to implement subaveraging instead of subsampling. This process requires $M \times N$ computations.

To form the pyramid, a set of values of s are selected covering the desired scale space range. The ratio of s between adjacent scales (i.e., how finely we sample scale-space) depends on the flexibility of the subsequent operators with respect to scale. Figure 2.5 illustrates the formation of an image pyramid via subaveraging and the corresponding sampling of scale space. Note, as well, that the Sobel operator that will be applied to the image pyramid is of fixed size. Figure 2.6 demonstrates the resulting multi-scale edge detection which is subsequently performed. Note the labelling at the side of the images which indicates large and small scales. The scale indicates the relative size of the operators, not the image. Thus, the smaller the intensity image, the larger the relative size of the operators acting upon it. The image at a scale value of $1\times$ is analyzed at a small scale since the operators are small relative to it and will detect tiny, local features in the image.

Thus, instead of resizing operators acting on the input image, the input is scaled and the operators are kept to a fixed size. This permits the use of simpler, faster, non-scalable operators such as the Sobel operator. Such a process is also more straightforward than painstakingly resizing a set of operators.

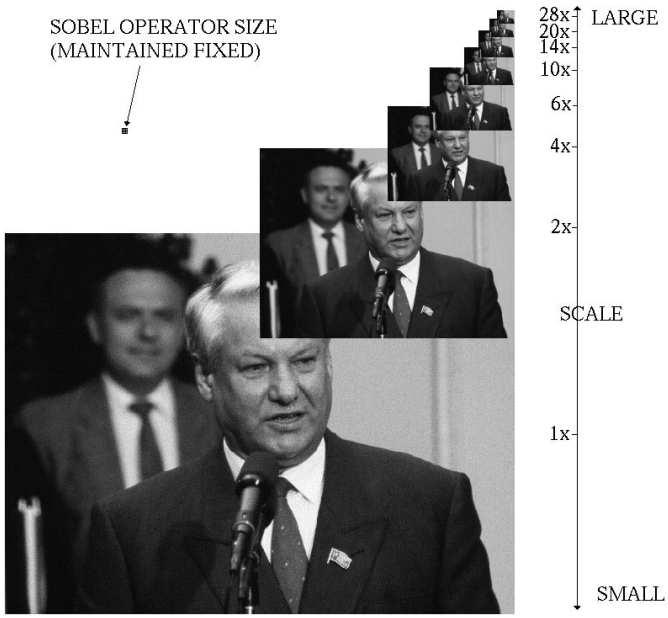


FIGURE 2.5. An input image pyramid for multi-scale edge extraction

Scaling input images is also computationally efficient. For example, convolution with a mask of size $m \times m$ with an image of size $M \times N$ requires $O(MNm^2)$ operations. However, if the image and the operator are scaled by s, the convolution would require $O(MNm^2s^{-4})_{\{for\ convolution\}} + O(MN)_{\{to\ scale\ image\}} + O(m^2)_{\{to\ scale\ mask\}}$. Large operators can thus be reduced to encourage computational efficiency as long as the input image is scaled appropriately.

Note, however, that the reduction in resolution brought about by scaling the image and the corresponding reduction in operator size gradually reduces the quality of the computation. Thus, it is necessary to select an operator size which acts on enough image pixels to perform the computation reliably.

5. Real-Time Symmetry Transform

To detect the centers of interesting regions from the multi-scale edge maps described previously, it is necessary to compute the degree of symmetric enclosure at each point in the image for each scale. Sela [**42**] proposes that symmetric enclosure can be computed from the axes of symmetry of the contours in an edge map. An interest map (or level

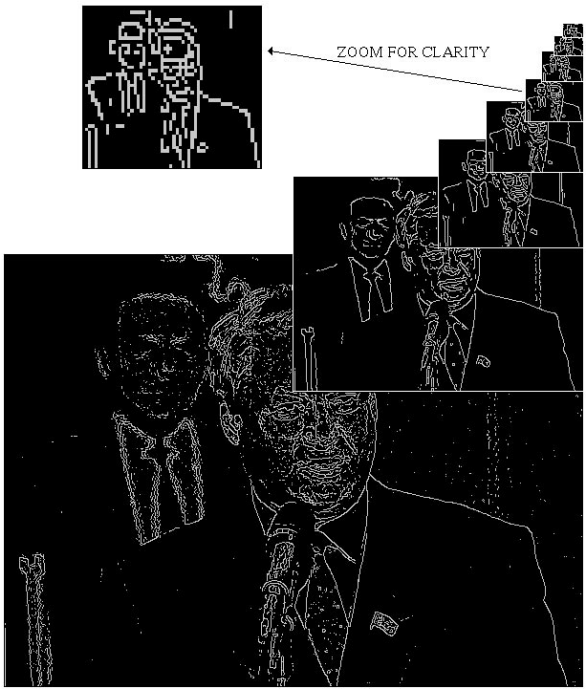


FIGURE 2.6. Multi-Scalar edge extraction with the Sobel operator

of perceptual enclosure) can be determined from the intersection of lines of symmetry. Other methods exist for determining the level of perceptual enclosure such as Reisfeld's Symmetry Transform [**37**]. However, Sela's technique [**42**] is particularly computationally efficient. The algorithm is described below and its discrete implementation is then outlined to demonstrate its efficiency.

5.1. Lines of Symmetry The computation of lines of symmetry begins with the concept of cocircular edges. A pair of edges is cocircular if a circle can be constructed with each edge as a tangent. Furthermore, if a line is drawn connecting the centers of the edges, the angle formed between this line and each edge's tangent is the same. We define the point of cocircularity as the center of the circle (x_c, y_c) and its radius r_c is the radius of cocircularity, as indicated in Figure 2.7. Note that $\theta_1 = \theta_2$ in the figure.

Furthermore, the center of cocircularity has an orientation associated with it. This orientation is determined by the phase of the cocircular edges contributing to this center of cocircularity. The phase value of an edge is the orientation of the normal of the edge. This normal points along the direction of the change of intensity from dark to bright. As

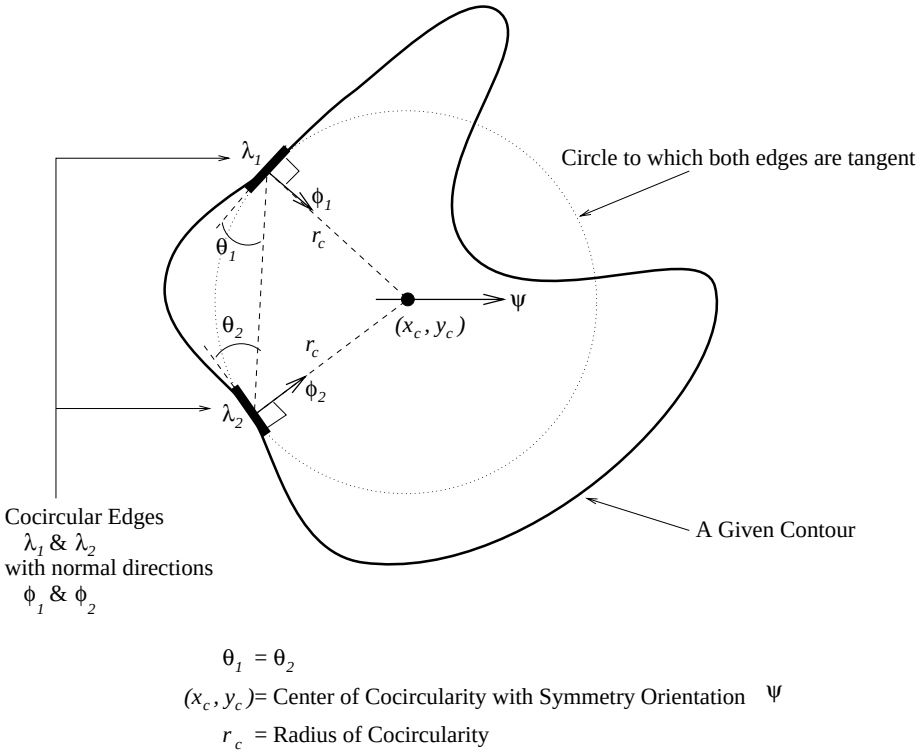


FIGURE 2.7. Cocircularity of edges

displayed in Figure 2.7, the edges have phases ϕ_1 and ϕ_2 which are computed from the Sobel edge detection. The symmetry orientation at the center of cocircularity, ψ, is the line that bisects the normals of the two edges. Equation 2.1 returns the value of ψ.

$$\psi(\lambda_i, \lambda_j) = \frac{\phi_i + \phi_j}{2} \tag{2.1}$$

The lines of symmetry are formed by linking all centres of circularity found in the image. In other words, circles are constructed from all pairs of cocircular edges and their centers (the centers of cocircularity) are used to trace out lines of symmetry. For each point $p = (x_c, y_c)$ in the image, we consider the surrounding neighbourhood of radius $r - \delta r < r < r + \delta r$. The range of r forms a ring shaped, circular annular region. Within that region, we consider all pairs of edges which have a center of cocircularity at p. We shall denote this set of cocircular edges as $\Gamma_r(p)$. We further constrain this set of cocircular edges such that only cocircular edges with a symmetry orientation of $\psi - \delta\psi < \psi < \psi + \delta\psi$ are considered. Thus, this subset of edges in the neighbourhood of p is denoted as $\Gamma_{r,\psi}(p)$. $\Gamma_{r,\psi}(p)$ will be used to refer to the set of all pairs of cocircular edges λ_i, λ_j with center of circularity at p with radius of cocircularity $r - \delta r < r < r + \delta r$ and with orientation of symmetry $\psi - \delta\psi < \psi < \psi + \delta\psi$.

Furthermore, the centers of cocircularity can be assigned different strengths depending upon the orientation of the edges and the intensity of the edges that contribute to forming them. For each point in the image p, at each scale or radius r and for each symmetry orientation ψ we find the set of cocircular pairs of edges $\Gamma_{r,\psi}(p)$. Sela defines the magnitude of symmetry in the (p, r, ψ) space as follows:

$$S_{r,\psi}(p) = \sum_{\lambda_i, \lambda_j \epsilon \Gamma_{r,\psi}(p)} \| \lambda_i \| \| \lambda_j \| (\sin \phi/2)^{w_1} \tag{2.2}$$

where $\| \lambda_i \|$ and $\| \lambda_j \|$ are the edge intensities of the two circular edges and ϕ is the angle separating their normals:

$$\phi = |\phi_i - \phi_j| \tag{2.3}$$

Cocircular edges with a larger value of $(\sin \phi/2)$, increasingly face and oppose each other and a stronger sense of symmetry is perceived at the point of cocircularity, p. The parameter w_1 is used to attenuate or boost the effect of $(\sin \phi/2)$. Selecting a large w_1 will diminish the contribution of non-facing edges so that only opposing cocircular edges will trigger $S_{r,\psi}(p)$. A value of $w_1 = 5$ is proposed by Sela [**42**].

Thus, the magnitude of the symmetry $S_{r,\psi}(p)$ at each point p, at each radius r and at each orientation ψ is obtained and represents the desired "lines of symmetry". It is possible to combine the lines of symmetry from multiple radii so that an overall, r-independent value of $S_\psi(p)$ is found as follows:

$$S_\psi(p) = \max_{r=0}^{r_{max}} S_{r,\psi}(p) \tag{2.4}$$

Note that these lines of symmetry are not really lines. Rather, we compute a symmetry magnitude at each combination of p, r and ψ so the result is a set of points with an orientation value. If true connected lines are required, these discrete points must be linked into curves using their orientation and scale value (see Chapter 3).

5.2. Intersection of Lines of Symmetry Sela proposes that the intersection of the lines of symmetry in $S_\psi(p)$ generates an interest point. This interest point has a magnitude depending on the configuration and strength of the lines of symmetry that generated it. We utilize the magnitude of the interest point as a measure of the level of symmetric enclosure at that point. Since perpendicular lines of symmetry generate the strongest sense of enclosure [**42**], the greater the level of orthogonality between two lines of symmetry, the stronger their contribution to the interest magnitude at point p. The contribution of each pair of lines of

symmetry intersecting point p is summed to generate an interest value $I(p)$ which is defined as

$$I(p) = \sum_{\psi_i, \psi_j} S_{\psi_i(p)} S_{\psi_j(p)} (\sin(\psi_i - \psi_j))^{w_2} \tag{2.5}$$

where (ψ_i, ψ_j) are the orientation values of a pair of symmetry lines intersecting point p whose symmetry magnitudes are $(S_{\psi_i(p)} S_{\psi_j(p)})$. The effect of orthogonality is included in the $\sin(\psi_i - \psi_j)$ term which is maximal when the lines of symmetry are perpendicular. The weight w_2 is used to tune the sensitivity of the computation to the orthogonality of the intersecting lines of symmetry. A value of $w_2 = 5$ is typically used so that orthogonal lines of symmetry dominate the response of the interest operator.

5.3. Real-Time Implementation A particular strength of the interest algorithm proposed above is its efficiency when implemented computationally. The relevant details and limitations of the real-time implementations will be discussed below. For a thorough analysis of the implementation, refer to Sela [**42**].

An important limitation to Sela's real-time implementation is its use of binary edge magnitudes. Thus, the gradient map must be thresholded. This process reduces the sensitivity of the computations since the values of λ (contrast) are limited to 1 bit of dynamic range.

In Equation 2.2, the symmetry lines are computed by considering the set of edges in $\Gamma_{r,\psi}(p)$. The latter can be visualized as a set of annular regions or rings centered around point p (see Figure 2.8). Only pairs of edges λ_i, λ_j from the edge map falling within the ring will be used to compute the symmetry lines at that point. Furthermore, edge magnitudes must be attenuated if their normals are misaligned with the normals of the annular sampling region. The normals of the annular sampling region are merely the normals of its contour as shown in Figure 2.9. The contribution of a cocircular edge is proportional to its magnitude projected onto the normals of the annular sampling region [**37**]. However, Sela deals exclusively with binary values for edge magnitude. Thus, he totally discards the contribution of edges whose phase angle is not within a range of the normal of the annular sampling region they fall into. His computation only considers edges whose phase is $\pm\omega$ degrees from the normal of the annular sampling region instead of gradually attenuating the contribution of misaligned edges. A value of ω=6 degrees is typical. Thus, only edges somewhat parallel or anti-parallel with the normal of the annular sampling region they

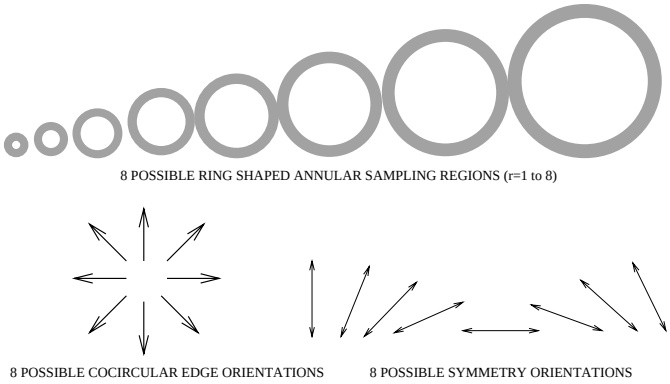


FIGURE 2.8. The set of circular sampling regions, the set of symmetry orientations and the set of cocircular edge orientations

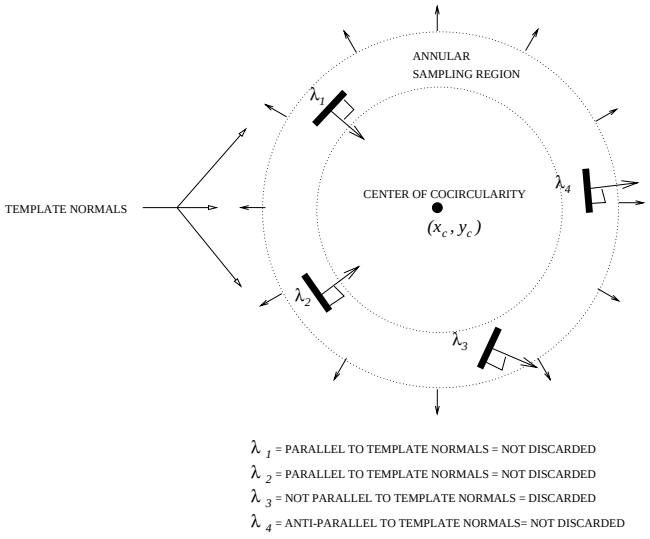


FIGURE 2.9. Discarding edges misaligned with the annulus normals

fall into are used in computing cocircularity. Figure 2.9 depicts the filtering performed to discard (instead of attenuate) the misaligned edges.

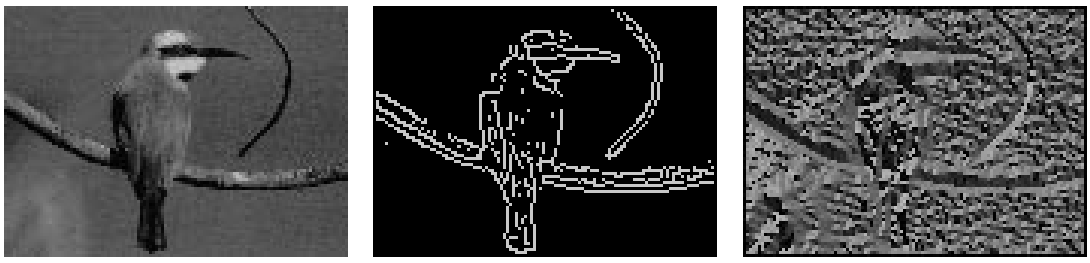
In Sela [**42**], the dynamic range of r is limited to 3 bits as is the dynamic range of ψ. In other words, only 8 different symmetry orientations at 8 different scales (sampling rings) are computed (see Figure 2.8). Furthermore, the phase description of the cocircular edges is also limited to one of 8 possible values. The reduction of dynamic range enables Sela to utilize pre-computed lookup tables to obtain the symmetry and interest maps instead of performing detailed calculations during the execution of the transform.

5.4. Dark and Bright Symmetry As mentioned previously, cocircular edges need to be parallel or anti-parallel with the normals of the boundary of the annular sampling

region to contribute to the symmetry. The resulting output will be referred to as "general symmetry". It is possible to restrict this definition further and perform the computations only on edges that are parallel to the normals of the sampling region; this yields "dark symmetry" and includes those edges which are oriented away from the center of cocircularity. Typically, dark symmetry will detect interest points and lines of symmetry in dark objects treating bright regions in the image as the background. "Bright symmetry", on the other hand, usually detects only bright objects by considering edges oriented towards the point of cocircularity.

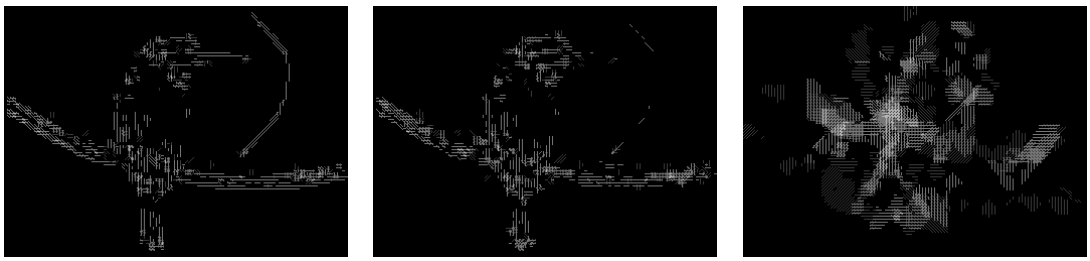
5.5. Application An algorithm very similar to Sela's proposed attentional mechanism is utilized to produce interest maps. These interest maps peak at the intersections of lines of symmetry which occur at the loci of symmetrically enclosed regions or "blobs". Certain modifications to Sela's implementation were made. These include the ability to change the range of the values of r of the annular sampling regions, as well as the ability to select between dark, bright and general symmetry. Furthermore, the symmetry lines are kept as an output of the algorithm since they will be utilized to identify "limbs". Limbs are symmetric structures with a single salient line of symmetry. Since this single axis is not intersected by other lines of symmetry, limbs do not generally trigger a strong response in the interest map. However, Kelly [**20**] claims that limbs can have significant perceptual significance despite this. Consequently, the intermediate data are very useful in the extraction of elongated limb-like structures, as proposed by Kelly. Limb extraction will be illustrated in the Chapter 3 as a technique for extracting the mouth from a face.

The following illustrates a typical application of the algorithm to compute the general symmetry transform of an image. Figure 2.10 displays the input to the algorithm. The lines of symmetry are computed over all 8 scales (i.e. at all 8 values of r) using Equation 2.2. Figure 2.11 shows the resulting line segments or points of symmetry which must be linked to form continuous lines of symmetry. The scales used for these symmetry maps are r=1, r=2 and r=8. Figure 2.12 shows the effect of Equation 2.4 which combines the 8 separate maps by selecting the maximum response from $r = 1$ to 8. The desired interest map I(p) is also shown in Figure 2.12. The points in I(p) undergo Gaussian smoothing and local maximum detection [**42**] to generate a set of discrete interest points at the centers of clusters of response found in I(p). The resulting interest points are finally displayed superimposed upon the input image.



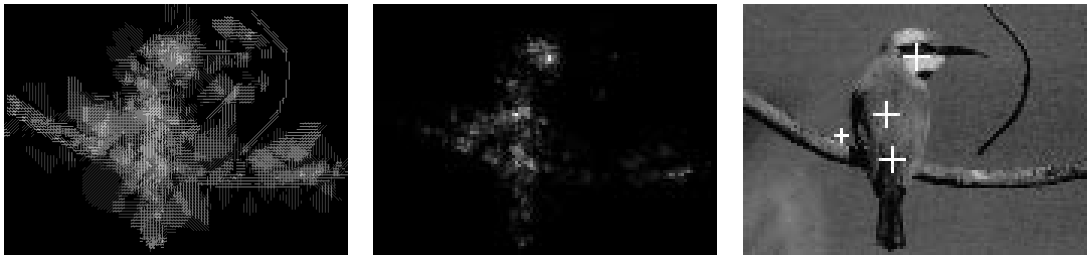
(a) (b) (c)

FIGURE 2.10. Input to the attentional mechanism. (a) Original intensity image. (b) Non maximally suppressed, thresholded, Sobel edge map. (c) Phase map.



(a) (b) (c)

FIGURE 2.11. Lines of general symmetry at multiple scales. (a) General symmetry lines at r=1. (b) General symmetry lines at r=2. (c) General symmetry lines at r=8.



(a) (b) (c)

FIGURE 2.12. Combining lines of symmetry (a) Maximum symmetry lines over all scales. (b) Resulting interest map, I(p). (c) Smoothed peaks of interest map superimposed as + signs on the original image.

6. Selective Symmetry Detection for Precise Blob Identification

The points generated by the interest map I(p) suffer from reduced accuracy due to the simplifications Sela proposes. Furthermore, these points return the center of regions of

interest and not the contours or structure of the regions. We have investigated the use of the symmetry line data to reconstruct the "blob" detected at p. The lines of symmetry at multiple scales and orientation do contain information about the structure of the region of symmetric enclosure. However, due to the many simplifications imposed upon the calculation, the data involved are compressed and the computation of this "inverse symmetry transformation" does not possess the accuracy needed to properly segment the "blob" for our purposes. A more accurate description of the "blob" would require a higher quality version of the symmetry transform.

Consequently, we propose the use of a high quality, but slower, symmetry transform as a post-processing stage to the previously described real-time interest operator. The significant points generated by the interest map I(p) serve as points of attention for this higher quality analysis. However, we do not attempt to actually invert the symmetry lines generated by this calculation to approximate the blob's contour. Instead, the annular sampling regions used to detect symmetry are deformed to function as templates for the specific shapes to be identified. This technique is similar to template matching. However, it utilizes the principles of symmetric enclosure to detect the desired blob and not merely the intersection of a template with edge data. This restricts the false alarms that might trigger simple template matching (as shown later in Figure 2.18). The following derivations outline the development of a symmetric enclosure measure for deformable annular sampling regions.

6.1. Semi-Elliptical Sampling Regions The sampling regions for the real-time symmetry transform are circular concentric rings of varying size and thickness (see Figure 2.8). The use of circular rings is ideal for detecting rotational invariance. However, we wish to develop an operator which will detects blobs of a certain form. This requires specific annular sampling regions to detect the contours of interest in the image. In other words, the sampling region should approximate the shapes we are searching for in the image. We proceed by defining a parameterized mathematical model of the templates to approximate facial contours. Figure 2.13 depicts the geometry of this so-called semi-elliptical model. It is a superset of the class of elliptical annuli which is itself a superset of the class of circular annuli. Such a model lends itself quite well to detecting human face and head contours which are not well-approximated by simple ellipses or circles. The model can be viewed as two half ellipses spliced together along a shared axis of length b. The model has the following parameters:

a: Length of one elliptical axis
b: Length of shared elliptical axis
c: Length of another elliptical axis

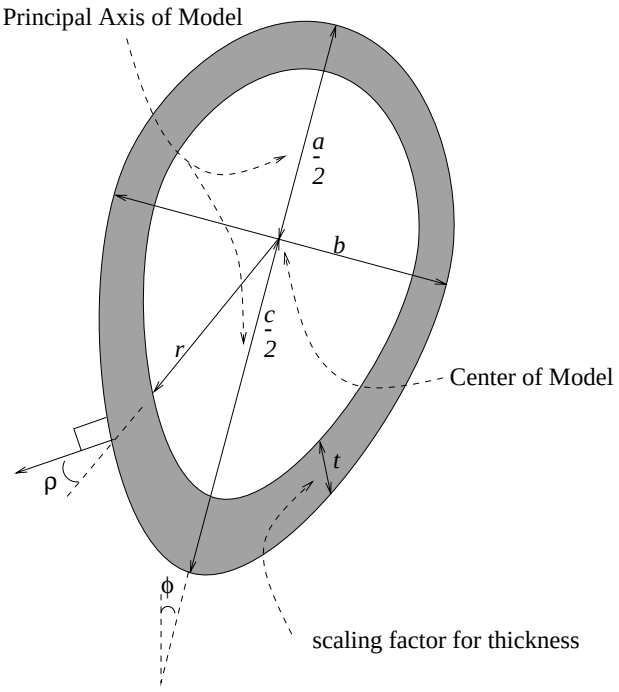


FIGURE 2.13. The semi-elliptical model

ϕ: Orientation of model with respect to vertical
t: Thickness as a scale factor of the model

If $a = c$ the model becomes an elliptical sampling region; if $a = b = c$ the template models a circular sampling region.

By varying the parameters of the model, we can generate the desired blob templates. The orientations of the normals at the boundary of the template are also pre-computed by a template generation process. Note that the orientation of the normals varies not only as a function of the angular position on the sampling region but also as a function of the radius from the center. The center of the model is a well defined point: it is the intersection of the shared elliptical axis and the meeting point of the other two axes (a and c).

The equations for an for an ellipse's boundary and its contour's normals are used to construct the templates [**43**]:

$$r = \frac{ab}{\sqrt{a^2\cos(\theta)^2 + b^2\sin(\theta)^2}} \tag{2.6}$$

$$\rho = \arctan\left(\frac{2r(b^2\sin(\theta)^2 + a^2\cos(\theta)^2)^{\frac{3}{2}}}{ab(a^2 - b^2)\sin(2\theta)}\right) - \frac{\pi}{2} \tag{2.7}$$

FIGURE 2.14. A sample template for face or head-like blob detection with template normals represented with intensity

a: length of major axis ($a_1 < a < a_2$)
b: length of minor axis ($b_1 < b < b_2$)
θ: angle of ellipse minor axis with respect to vertical
r: radius
ρ: angle between radius vector and boundary normal

Figure 2.14 depicts a typical template consisting of a semi-circular annulus on top of a semi-elliptical annulus. The orientation values of the normals of the template are represented by intensity. Note the similarity of this template to the shape of a human head or face.

6.2. Projecting Edges onto the Template To find a head or face like contour around a point p in an image, we begin by generating the appropriate template. The template is then centered at point p on the edge map. Any edges which fall into the sampling region of the template (the darkened region in Figure 2.13) are then considered as possible contributors to a template match. The magnitude of each edge determines the extent of its contribution. Furthermore, the orientation of the edge will also vary the contribution. If an edge is parallel to the boundary of the template, it is well aligned with the template's contour and it is a component of a boundary that is part of the overall shape we are seeking. However, if an edge is perpendicular to the boundary of the template, it is probably part of an external contour that crosses through the template. Such an edge's contribution should therefore be attenuated. In other words, edges which are not perpendicular to the normals at the template's boundary are misaligned and their magnitude should be weakened to reflect this. The normals of the template in Figure 2.13 are displayed.

Observe Figure 2.15. On the left is an annular sampling region which is triggered by 4 edge segments: $\lambda_1, \lambda_2, \lambda_3$, and λ_4. However, the edge labelled λ_3 is not aligned with the shape of the template. Its normal is not pointing radially inward/outward from the

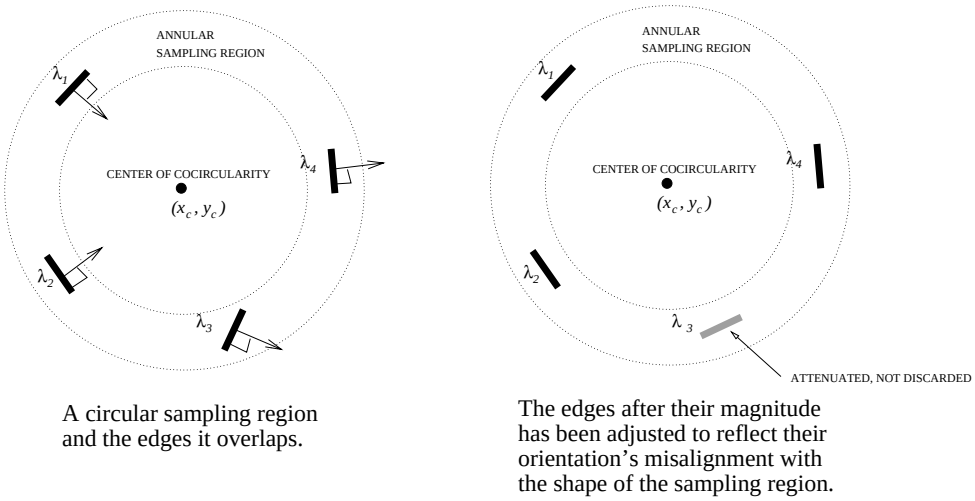


FIGURE 2.15. Projecting onto template normals to attenuate misaligned edges

center of cocircularity as is the case for the other edges. Thus, it is misaligned with the radial normals typical of this circularly shaped region. It's contribution is thus weakened by scaling down its magnitude so that it is equivalent to a weaker yet properly aligned edge. Turning our attention to the ring on the right of Figure 2.15 we can see λ_3 now in a lighter shade of gray, depicting the attenuation its magnitude has undergone. Its contribution to the template matching algorithm is weakened yet not totally discarded.

Equation 2.8 is used to scale the magnitude of each edge by a scaling factor $s_{magnitude}$ to vary its degree of contribution to the template's overall response.

$$s_{magnitude} = |\cos(\varrho - \varphi)|^{v_1} \tag{2.8}$$

ϱ: angle of template normal
φ: edge orientation
v_1: degree if attenuation of misaligned edges

The value of v_1 determines the degree of attenuation for misaligned edges. A higher v_1 completely attenuates misaligned edges thereby approximating Sela's method of discarding the edges. However, when the Sobel operator is used to compute edge maps, the phase values are not reliable enough to have such a severe attenuation. Thus, a value of v_1=2 was selected in our implementation.

6.3. Symmetric Enclosure After noting the edges intersecting the template and attenuating them appropriately, the edges are sorted into 32 angular bins depending on the region of the template they fall under. Figure 2.16 displays the division of the model by a

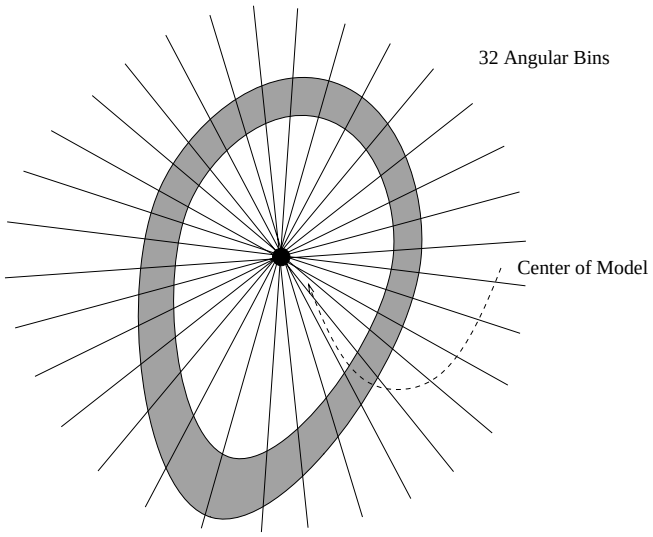


FIGURE 2.16. Splitting templates into angular bins

ray emanating from its center at 32 angular intervals to form these bins. Within each bin i, the strongest projected edge magnitude is stored as λ_i . In this way, we form an angular profile of the edge data as shown in Figure 2.17. Symmetric enclosure is then computed by summing the contribution to symmetry of each pair of bins as in Equation 2.9 which is derived from Equation 2.2 [**20**]:

$$SE = \sum_{i=1}^{32} \sum_{j=1}^{32} \lambda_i \lambda_j \sin(\frac{\vartheta_i - \vartheta_j}{2}) \tag{2.9}$$

ϑ_i: Angular position of i^{th} bin
λ_i: Peak projected edge magnitude in i^{th} bin

Due to the pixelization of the edge data, smaller templates will not overlap enough pixels in the edge map to trigger each of their 32 angular bins. Thus, small templates will yield consistently lower values of SE. Therefore each template's output is normalized and presented as a percentage of the possible peak output of the template. For each template, we compute the maximum possible value of SE. The final output is a value from 0 to 100% allowing the implementation of a template-independent threshold on SE. Thus, resolution variations in the image due to scaling and discretization will not affect the response of our template-based symmetry detection.

6.4. Application: Symmetric Enclosure versus Template Matching While the selective symmetry operation resembles template matching in the way it intersects

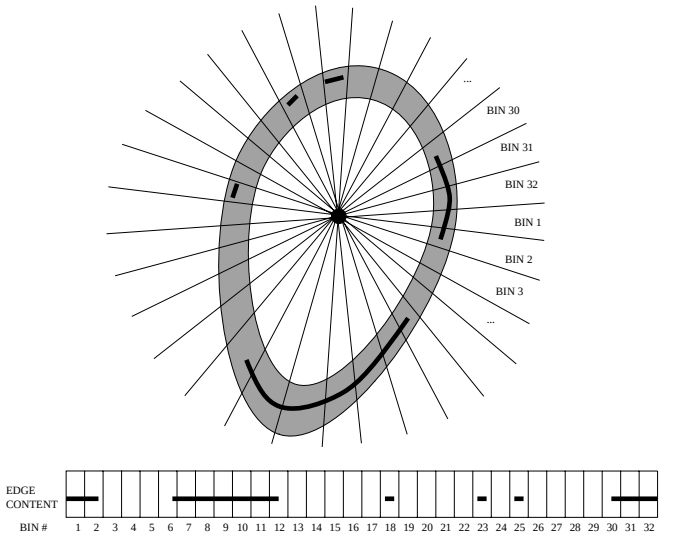


FIGURE 2.17. Computing the angular profile of a contour

the edge map with its semi-elliptical deformable model, it also utilizes the principles of symmetric enclosure. Concepts derived from Reisfeld [**37**], Sela [**42**] and Kelly [**20**] seem to enhance the template matching process. Symmetric enclosure is used to weight the response to the contours being detected by template intersection. The measurement of symmetric enclosure and the projection of edges along the template normals filter out a variety of inappropriate edge configurations which do not form adequate contours. These also amplify desired contours that would fail to trigger template matching. Figure 2.18 demonstrates the advantages of the selective symmetry operation.

Thus, the selective symmetry operation reliably detects the desired contours in a manner similar to template matching, with the added benefits of wide non-circular annular sampling regions, non-circular phase orientation weighting, and symmetric enclosure calculations. Furthermore, the selective symmetry operation is a higher resolution blob detector than Sela's symmetry transform since it keeps track of edge magnitude and gradually attenuates misaligned edges instead of discarding them. Furthermore the use of more bits to describe angular bins, edge orientation and edge magnitude provides a more reliable response.

However, the selective symmetry operation is not tuned for speed and does not use pre-calculated lookup tables. The symmetry calculations must be repeatedly evaluated. Thus, it is computationally slower than the symmetry transform.

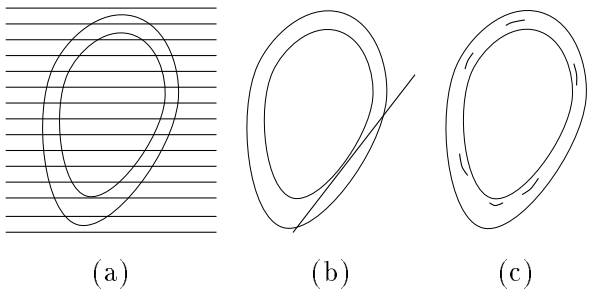


FIGURE 2.18. Symmetric enclosure versus traditional template matching. (a) Although these edges trigger many of the template's angular bins by intersection, they are severely misaligned with the template's normals and would yield very weak magnitudes when attenuated via Equation 2.8 under the selective symmetry operation. Traditional matching would erroneously register a strong response. (b) This contour would erroneously trigger a strong template match but not the selective symmetry operation due to a low measure of perceptual enclosure. (c) The contour here would erroneously fail to trigger template matching but will properly register under the selective symmetry operation due to a strong sense of enclosure around the model's center.

The selective symmetry detector is not intended to replace the symmetry transform but to be used in conjunction with it. By applying the selective symmetry operation in a neighbourhood around peaks in the interest map, we can refine the output and localize interest peaks more precisely. Furthermore, by varying the templates used by the operation, we can detect the specific shape that triggered the interest map around the interest point. Thus, the selective symmetry operation is applied as a post-processing step after applying the symmetry transform to improve the location of the peaks of the interest map and to estimate the contours that generated them.

CHAPTER 3

Face Detection and Localization

The detection of faces and facial features from an arbitrary uncontrived image is a critical precursor to recognition. A robust scheme is needed to detect the face as well as determine its precise placement to extract the relevant data from an input image. This is necessary to properly prepare the image's 2D intensity description of the face for input to a recognition system. This detection scheme must operate flexibly and reliably regardless of lighting conditions, background clutter in the image, multiple faces in the image, as well as variations in face position, scale, pose and expression. The geometrical information about each face in the image that we gather at this stage will be used to apply geometrical transformations that will map the data in the image into an invariant form. By isolating each face, transforming it into a standard frontal mug shot pose and correcting lighting effects, we limit the variance in its intensity image description to the true physical shape and texture of the face itself.

The set of input images in Figure 3.1 illustrates some of the variations in the intensity image that detection must be capable of overcoming to properly localize the face. These variations need appropriate compensation to isolate only the relevant data necessary for recognition. Furthermore, note that these variations can occur in any combination and are not mutually exclusive.

We propose a hierarchical detection method which can quickly and reliably converge to a localization of the face amidst a wide range of external visual stimuli and variation. It is necessary to precede expensive computations with simple and efficient ones in this hierarchy to maximize efficiency. The results of the initial, diffuse and large search space computations narrow the search space for the more localized, higher precision operations that will follow. In other words, the results of preliminary detections guide the use of subsequent operations in a feed-forward manner to restrict their application to only significant parts of the image. This reduces the probability of error since the subsequent detection steps will

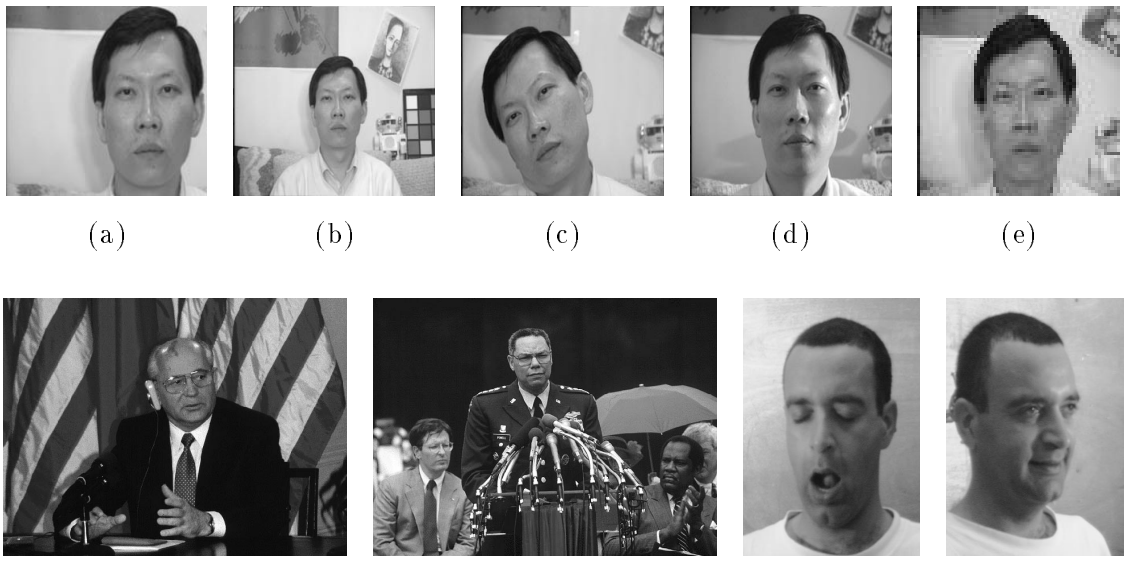


FIGURE 3.1. Variations in faces that require appropriate compensation. (a) Change in position. (b) Change in size or scale. (c) In-plane rotation of the face. (d) Shading and illumination effects. (e) Variation in image quality or resolution. (f) Clutter in the image background. (g) Multiple faces in the image. (h) Changes in facial expression. (i) Out-of-plane rotation.

not be distracted by irrelevant image data. Furthermore, more robust operations precede more sensitive ones in our hierarchy since the sensitive operations in the hierarchy need to have adequate initialization from previous stages to prevent failure.

Figure 3.2 displays the sequence of search steps for the face detection. We begin by searching for possible face or head-like blobs in the image. The detected blob candidates are examined to obtain an approximation of their contours. If these exhibit a face-like contour, their interior is scanned for the presence of eyes. Each of the possible pairs of eyes detected in the face are examined one at a time to see if they are in an appropriate position with respect to the facial contour. If they are, then we search for a mouth isolated by the facial contour and the position of the detected eyes. Once a mouth has been detected, the region to be searched for a nose is better isolated and we determine the nose position. Lastly, these facial coordinates are used to more accurately locate the iris within the eye region, if they are visible. The final result is a set of geometrical coordinates that specify the position, scale and pose of all possible faces in the image. The last few stages will be discussed in Chapter 4 which utilizes the facial coordinates to normalize the image and perform recognition. Note the many feedback loops which propagate data upwards in the

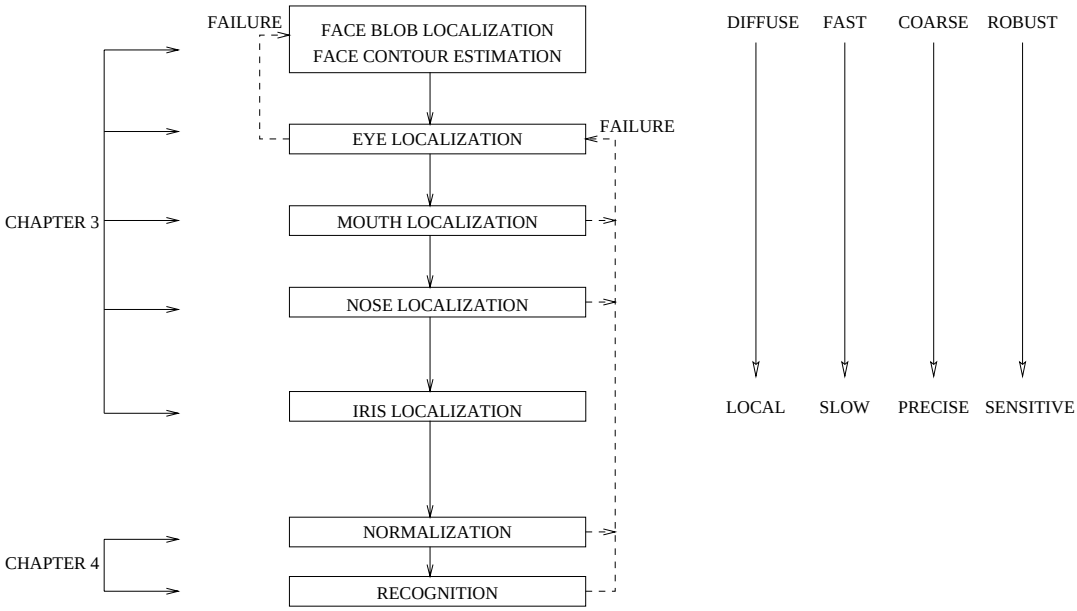


FIGURE 3.2. The hierarchical search sequence for faces and facial features

hierarchy. These are used by the later stages to report failure to the preliminary stages so that appropriate action can be taken. For instance, if we fail to find a mouth, the pair of possible eyes that was used to guide the search for the mouth was not a valid one and we should consider the use of another possible pair of eyes.

Note the qualitative comparison of the different stages on the right of Figure 3.2. This is a figurative description of the coarse-to-fine approach of the algorithm. The initial stages of the search are very fast and coarse since they use low resolution operators. Furthermore, these operators are used to search relatively large regions in the image. Additionally, the early stages are robust to noise and do not need to have constrained data to function. Later stages yield more precise localization information and use high resolution, slow operators. However, they are sensitive to distracting external data or noise and therefore need to be applied in a small, constrained window for a local analysis. In other words, they need to be guided by the previous, robust stages of the search. This figurative description of the stages is merely intended to reflect the spirit with which detection is to be approached. In short, it begins with a 'quick and dirty' estimate of where the face is and then slowly refines its localization around that neighbourhood by searching for more precise albeit elusive targets (such as the iris). This concept (coarseness to fineness) will become clearer as the individual stages of the algorithm and their interdependencies are explained later.

We implement this hierarchical search as a control structure which utilizes a palette of tools that includes the biologically and perceptually motived computations developed in Chapter 2. These are used to extract low-level geometrical descriptions of image data which will be processed to generate a robust and accurate localization of the face.

Note that the detection algorithm is based on a variety of heuristics that vaguely describe a model for the human face. The multitude of thresholds and geometric relationships that we introduce at each stage of the localization define our model of the human face cumulatively. Furthermore, the thresholds and constraints on this face model have been kept relatively lax to allow for a wide range of face imaging situations. Consequently, the numerical parameters that are utilized are not critical, nor are they optimal or unnecessarily sensitive. Rather, the parameters allow for large margins of safety and are forgiving, allowing face detection to proceed despite noise, variations, etc. Thus, a flexible, forgiving model gives the system greater robustness and fewer misdetections. In fact, a face is such a multi-dimensional, highly deformable object that an explicit, precisely parametrized model would be very difficult to derive and manipulate.

1. Face Localization

The human face is a highly correlated object due to the lack of variation in skin complexion. Even though facial features (i.e. mouths and eyes) differ in color from the skin tone, the hairless skin regions dominate facial surface area allowing it stand out against most backgrounds. Thus we expect a boundary around the face to be present. The foreshortening of the 3D face structure under most lighting conditions also accentuates the contour of the face. We propose that the face be considered as a blob or a region with a certain amount of contour enclosure. Furthermore, the scalp and the hair also usually triggers edge contours that extend the face blob. Consequently, we can expect both the face and head structures to behave as blobs or symmetric enclosures about a center point.

1.1. Face Blob Localization Having formally described a blob-detector, the symmetry transform, we elect to use it to initially detect all blobs in the image. Some detected blobs will correspond to non-facial structures however all faces and heads should trigger the interest map. Recall that the transform operates on edge maps so it should locate blobs despite the blob's intensity values or shading and illumination. Furthermore, the blob detector is rotation invariant.

We begin with an arbitrary image of a natural uncontrived scene containing people. We then generate an intensity image pyramid as in Figure 2.5 and a corresponding edge

Annular Region Number	Range of Radii
1	$0.75 < r < 2.25$ pixels
2	$1.75 < r < 3.25$ pixels
3	$2.75 < r < 4.25$ pixels
4	$3.75 < r < 5.25$ pixels
5	$4.75 < r < 7.25$ pixels
6	$6.75 < r < 9.25$ pixels

TABLE 3.1. The annular sampling regions to be used for face detection

map pyramid as shown in Figure 2.6. We use the edge map pyramid to apply the symmetry transform at various scales. This allows us to detect blobs of arbitrary size in the image. The blob detector uses only the 6 annular sampling regions described in Table 3.1. We can afford to limit the number of annular sampling regions to six at this stage since the subaveraging involved in the pyramid obviates the need for more scale invariance in the operator. We apply the general symmetry transform to each of the edge maps and mark the centers of the detected blobs on the intensity pyramid. The general transform (not the dark or bright symmetry transform) was utilized since heads and faces do not consistently appear either brighter or darker than the background of a scene. This multi-scale interest detection operation provides us with the blob detection pyramid displayed in Figure 3.3.

We thresholded the output of the interest map so that only attentional peaks exhibiting a certain minimal level of interest will appear in the output. The threshold on the interest map is very tolerant and allows many extremely weak blobs to register. Thus, the precise selection of an interest map threshold is not critical. Furthermore, we only consider the five (5) strongest peaks of interest or the five most significant blobs for each scale in the multi-scalar analysis. This is to prevent the system from spending too much time at each scale investigating blobs. We expect the face to be somewhat dominant in the image so that it will be one of the strongest five blobs in the image (at the scale it resides in). If we expect many faces or other blobs in the image at the same scale, this value can be increased beyond 5. This would be advantageous, for example, when analyzing group photos. Both a threshold on interest value and the limitation on the number of peaks are required since we do not wish to ever process more than 5 blobs per scale for efficiency and we require the blobs to exhibit a minimal level of significance to warrant any investigation whatsoever. Furthermore, we stop applying the interest operator for scales smaller than 4x. The interest

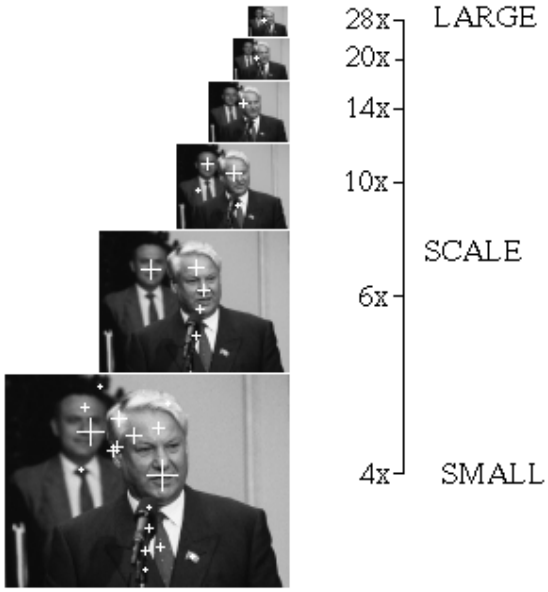


FIGURE 3.3. The multi-scalar interest map pyramid

operator is limited in size to r=9 pixels and consequently, the blobs detected at scales lower than 4x would be too small and would have insufficient resolution for subsequent facial feature localization and recognition. For example, the blobs detected at scale 3x would be less than 54×54 pixel objects and the representation of a face at such a resolution would prevent accurate facial feature detection.

The peaks in Figure 3.3 are shown before we threshold the interest response, threshold the number of blobs per scale (maximum of 5) and before we limit the scale of the search. Once these three limits are introduced, the number of peaks generated by the face blob detection stage will drop as shown on the right hand side of Figure 3.4. Only a total of 5 peaks are valid after this filtering (as seen by the 5 square grids that remain for processing by the next stages).

There is some redundancy as some blobs are detected more than once at adjacent scales. This is due to overlap in scale-space of our symmetry transform operator. However, this redundancy or multiple-hit phenomenon is not problematic since we will use later stages to select only one 'hit' or one face out of several redundant blob responses. Additionally, the detection of non-facial blobs is not problematic at this stage. Since each blob is to undergo further processing to determine if it is truly a face, we can allow false alarms during blob detection. Finally, lack of accuracy in our blob detector is also acceptable at this stage since we will refine the localization of faces in subsequent stages. What is most dangerous at this

early stage of the algorithm is a total miss of a face or head blob in the image. Fortunately, a clear miss of the face in our multi-scalar blob detection is extremely unlikely since heads and faces have consistently strong responses in the perceptual interest map.

1.2. Face Contour Estimation After having applied the symmetry transform over the whole image at multiple scales we have a set of loci for the perceptually significant blobs. Consequently, we have restricted the search space for faces and can afford to utilize the more computationally expensive selective symmetry detector. The selective symmetry operator is applied over a 5×5 neighbourhood around all loci generated by the multi-scalar interest maps. This is done to refine the coarse initial localization of interest peaks. The selective symmetry detector provides higher precision as demonstrated in Chapter 2. We also boost resolution at each scale by a factor of 2 to use large templates with the selective symmetry operator. This permits the detector to utilize more image data in the computation.

Also, note that the selective symmetry operator is applied over the gradient map (not the binary edge map) generated from these higher resolution images. However, the gradient map magnitudes are replaced by the square-root of the magnitude. This attenuates the effect of contrast in the calculation of symmetric enclosure. The facial contour is not necessarily of high contrast against the background and this is especially true of the chin area. The chin and the neck are composed of the same skin tone and thus the contrast generated from this contour between the two is only due to the foreshortening of the chin and the shading below it. Thus, a reduced sensitivity to contrast allows the selective symmetry detector to detect the strong sense of symmetric enclosure the chin contour brings to the facial structure despite its rather weak edge content.

Figure 3.4 depicts the use of the local maxima in the interest map to define the search space for the selective symmetry detector. The interest map peaks at scales 28, 20 ,14 and 10 are shown on the left side of the Figure. Note the 5×5 window of dots forming a neighbourhood around these peaks in the images on the right. These images on the right side are higher resolution versions of the ones on the left (double the resolution) and will be operated upon by the selective symmetry detector at each of the 25 white points within them. This process is performed first at large scales (at scale 28 in the example). This agrees with the notion that large scales are usually more perceptually significant. Subsequently, we use the selective symmetry operator to compute the structure of the blob more exactly.

Recall that the selective symmetry detector requires the creation of templates as explained in Chapter 2. We wish to detect facial contours at a variety of orientations to detect tilted heads as well as vertical ones. It is necessary to expect different aspect ratios as some

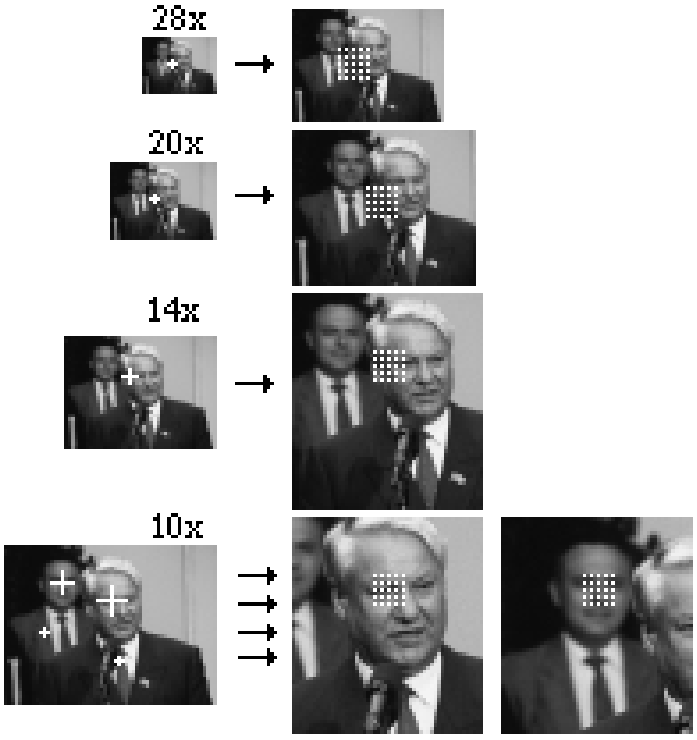


FIGURE 3.4. The search space for the selective symmetry detector

individuals have wide faces/heads while others have slim, elongated ones. Furthermore, recall the discrete sampling of scale-space that generated the multi-scalar interest maps. The intervals between scales require a certain scale flexibility of the operators. For instance, in Figure 3.4 we note that the image is scaled down by $28\times$ and $20\times$. Thus, an operator acting on these two images must span the intermediate scales to assure full coverage in scale space. Thus, the operator must be a thick annulus that overlaps itself when it is scaled by a ratio of $\frac{28}{20}$ to assure that there are no gaps in the sampling of scale-space. The symmetry transform had 6 rings of different radii. Similarly, the selective symmetry detector should have multiple template sizes as well so that the operator overlaps in scale-space. Thus we need to create templates with various orientations, aspect ratios and sizes.

We also need to guarantee a certain level of overlap between templates. For example, observe Figure 3.5 which displays 3 templates of a head with the following orientations: along the vertical, at +60 degrees from the vertical and at -60 degrees from the vertical. If a face is encountered at -30 degrees from the vertical, we will probably not detect it. What is needed is a certain amount of overlap between one template and the next so that intermediate face contours will be detected. Thus, we must finely sample the orientation, aspect

FIGURE 3.5. Insufficient operator overlap problems

ratio and size ranges in our template creation process to ensure overlap. We seek roughly 50% area overlap between neighbouring templates. Furthermore, when we proposed the search space for the selective symmetry operator as a 25-point neighbourhood we sampled the search space appropriately to ensure proper coverage as well. In other words, we do not have gaps in the spatial domain. The thick annular operator overlaps the search area well since the (x, y) points at which we apply the selective symmetry operation are densely arranged.

Figure 3.6 displays all the required templates once we have sampled the semi-elliptical model's scale-space, orientation space and aspect ratio space, appropriately. In total, we consider 5 possible orientations: -30, -15, 0, 15 and 30 degrees from the vertical, 3 scales and 2 aspect ratios for a total of 30 templates. The model described in Chapter 2 is utilized with $a = b$ and $c = AspectRatio \times b$. The ϕ parameter of the model is the orientation. Finally, the t parameter is set to $\frac{2}{3}$ so that the annular sampling regions overlap adequately.

Each of the templates is applied with its center aligned to each of the 25 points forming the search space of the selective symmetry detector. For each template at each of the 25 positions, we compute a value of SE, as shown in Equation 2.9. The template which generates the highest value of SE will be the estimate for the facial contour for the given peak in the interest map.

For each blob, we exhaustively attempt each template matching and the strongest template is our final estimate for the facial contour. It must generate a certain minimal value of SE for it to be a legitimate facial contour. We select a threshold on the value of SE at 25%. Recall that the value of SE is expressed as a percentage of the peak value that can trigger the template in question. If the best template at the given peak is weaker than 25%, it will be rejected, indicating that the interest map peak was generated by another structure which does not fit the shape of the face templates. Thus, certain points

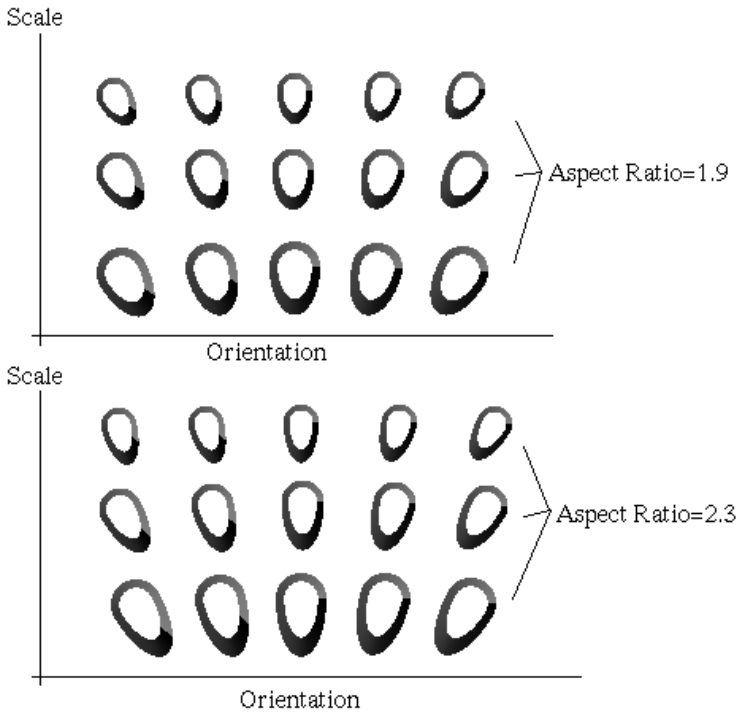


FIGURE 3.6. The face templates used by the selective symmetry detector

in the interest map will be rejected as non-faces at this stage if they fail to trigger the face templates adequately. The threshold value of 25% on the facial contour detection is a very tolerant one. All faces tested generated values of SE significantly above 25%. However, other non-facial yet symmetric structures will be discarded. The estimates for the facial contours resulting from the local peaks in the interest maps in Figure 3.4 are displayed as darkened annular regions superimposed upon the input intensity images as shown in Figure 3.7.

There is successful and precise detection of both face contours in cases (d) and (e) despite the variation in scale, focus, pose and illumination. Unfortunately, non-face structures also triggered the face contour detector as seen in cases (a), (b) and (c). The larger contours are triggered in part by the high contrast in the clothing of the individuals. Furthermore, the close proximity of the heads of the two individuals causes the selective symmetry detector to utilize contours from both faces in the image simultaneously. However, had a single face been the dominant object in the image, the contour detection would have triggered fewer false alarms. Once again, false alarms are permissible at this stage since further testing and elimination will subsequently fine-tune the output. It is critical, though, that there

(a) (b) (c) (d) (e)

FIGURE 3.7. The collection of detected possible facial contours

(a) (b)

FIGURE 3.8. Face contour detection in a single face image. (a) Original intensity image. (b) Only face contour detected.

are no misses at this stage since we only propagate the data that generated adequate facial contours to the subsequent testing stages in our hierarchy. Figure 3.8 depicts a situation where the face is dominant in an image and hence the only detected facial contour is the one corresponding to the actual face in the image.

Figures 3.4 and 3.7 do not show the search space (25 white points) or the facial contour estimate for the two weakest peaks in the interest map at the 10x scale. This is because these points failed to generate values greater than 25% for any of the face/head templates. This is understandable since they are triggered by the clothing of the individuals in the scene (not faces). Thus, this selective symmetry detector stage not only refines the localization of the face's center, it detects facial contour structure and also filters out non-face-like blobs from the output of the interest map. The final output is a collection of possible facial contours whose interior must now be processed to determine the existence of appropriate facial features.

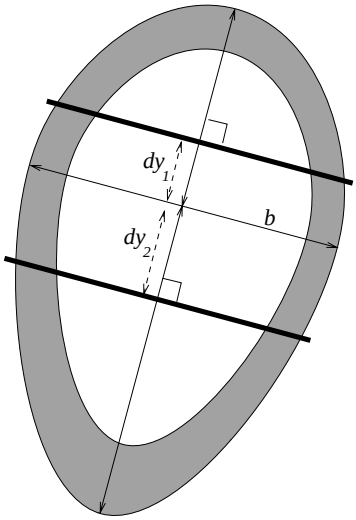


FIGURE 3.9. Generating eye search region from the facial contour

2. Eye Localization

Having found a set of possible facial contours in the image, we proceed with the detection of the eyes within the face. When we refer to the eyes in this section, we are referring not only to the iris but rather the collection of contours forming the pupil, iris, eyelids, eyelashes, eyebrows and the shading around the eye orbit. This general eye region is a larger and more dominant structure as a whole than its individual subcomponents. Therefore, it is more stable and easier to detect as a whole. Reisfeld utilizes large operators that span regions larger than the eyelashes, iris and pupils to improve reliability in the eye detection [**38**]. Although the process of including the surrounding region improves robustness, it reduces accuracy since the contours of the eyebrows and eye orbit shading may have a center that does not coincide with the pupil's center. Some high quality, deformable model methods for detecting the iris and eyelids have been proposed by Yuille and others [**45**]. However, they can be computationally expensive and are not as robust as the large operators acting on the whole eye region. For example, if an individual in the image is squinting or if the image quality is poor, the iris will not be clearly visible and such high precision methods which search exclusively for an iris or eyelids might fail.

We shall use the knowledge acquired about the facial contour structure from the previous stage to constrain the search for eyes. The spatial search space will be restricted by a wide band perpendicular to the principal axis of the facial contour. Figure 3.9 shows the semi-elliptical model is composed of two axes intersecting at the model's center. Beginning

(a) (b) (c) (d) (e)

FIGURE 3.10. Isolating the eye search regions

from the center of the model, we move up the principal axis a distance dy_1 and down the principal axis a distance dy_2, where we form two parallel lines that contain the band of interest. The lengths dy_1 and dy_2 can be selected by analyzing several faces from a database and noting the relative placement of eyes with respect to the detected facial contour. The setting $dy_1 = 0.15 \times b$ and $dy_2 = 0.45 \times b$ generates a band that more than adequately covers the eyes. Although this setting seems rather conservative, a wide margin of error is needed since the facial contour might surround the face or the whole head. Either the inner hair-line or the top of the head could be traced by the boundary of our facial contour, so a narrow eye-band might be unsafe. Figure 3.10 shows the eye bands or eye spatial search space as brightened strips superimposed upon the original intensity images.

2.1. Detecting Eye Regions The search space for the eye detection is now defined so we proceed to define the nature of the eye detection operation. In light of the highly symmetric, blob-like nature of the eye, we elect to use the symmetry transform to detect it as a peak in the interest map. Reisfeld proposes the use of a similar fixed size symmetry operator on the image [**38**]. Similarly, we employ our more efficient symmetry transform (which also has a fixed size with its pre-specified annular sampling regions). Observe Table 3.2 for the parameters of the annular regions for the symmetry transform at this stage. The usage of 8 different annular sampling regions with a wide range of radii is necessary due to the variety of sizes of the contours in the eye region. Large contours from the eye brows as well as small contours from the pupil are to be considered.

Note that the symmetry transform has a fixed set of annular sampling regions. If the face is very large in the image and the eye region has a large pixel area, the annuli (which are at most 14 pixels in radius) will not overlap the whole eye region. However, since we know the approximate dimensions of the facial contour from the preceding stage, we know

Annular Region Number	Range of Radii
1	$0.75 < r < 2.25$ pixels
2	$1.75 < r < 3.25$ pixels
3	$2.75 < r < 4.25$ pixels
4	$3.75 < r < 5.25$ pixels
5	$4.75 < r < 7.25$ pixels
6	$6.75 < r < 9.25$ pixels
7	$8.75 < r < 11.25$ pixels
8	$10.75 < r < 14.25$ pixels

TABLE 3.2. The annular sampling regions to be used for eye detection

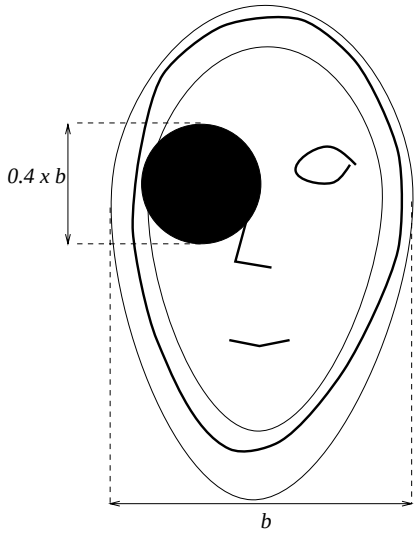
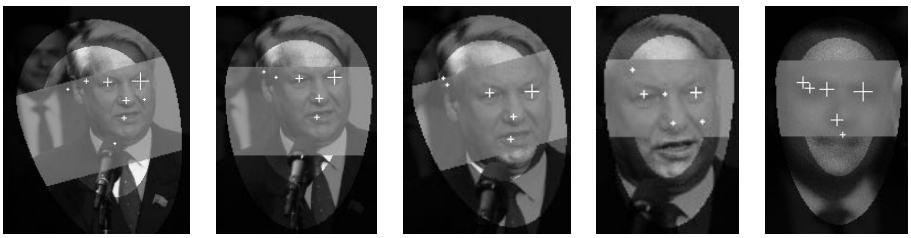


FIGURE 3.11. Eye operator size versus face size

roughly what size the eye region must be. Figure 3.11 shows a facial contour. The width of the facial contour is known (b) and thus, we expect the eye region to be at most $0.4 \times b$ in diameter. If $0.4 \times b$ is larger than 2×14 pixels, then the eye region is larger than the symmetry operator (who has a maximum annular region radius of 14 pixels). Thus, the input image is scaled by a factor s_{eyes} so that $\frac{0.4 \times b}{s_{eyes}} < 2 \times 14$. Unlike Reisfeld, we scale the image to accommodate the limited range of our fixed symmetry operator.

Furthermore, the eye region, eyebrow, and eyelashes are surrounded by skin, and the iris is surrounded by the bright white sclera. Thus, we expect the eye region objects to



(a) (b) (c) (d) (e)

FIGURE 3.12. Isolating the eye search regions

be darker with respect to their immediate background and can restrict the computation of symmetry to dark symmetry only.

Figure 3.12 displays the resulting peaks in the interest maps once the dark symmetry transform has been computed. Usually, the strongest peaks correspond to the eye regions. Thus we can limit the number of peaks to be processed further to the 5 strongest interest peaks[1].

The set of interest peaks (approximately 5) representing the possible eyes has been acquired. However, of these 5 peaks, which ones are the true eyes of the individual? It is possible to merely select the top two peaks in the eye band. Since eyes are such strong interest points, this is satisfactory in the majority of cases. However, it is sometimes possible that the top interest points are generated by other structures. For example, a loop of hair from the head could fall into the eye band and generate a strong interest peak (see Figure 3.13. Thus, we maintain the collection of possible eyes, accepting these false alarms for now. Further testing will isolate the true eyes from this set more reliably than the mere selection of the top two peaks.

We need to consider each pair of eyes in the set of detected peaks in the eye band. If 5 peaks are present, the total number of possible pairs is $\binom{5}{2} = 10$. However, we proceed by testing the strongest pairs first in a sequential manner until we find a pair that passes all tests. We can then stop testing any weaker pairs since we have converged to the two true eyes. Usually, the top two peaks will be the true eyes so we quickly converge without exhaustively testing all 10 pairs of possible eyes.

[1] It is possible to have more than 5 peaks if there is a tie for 5th place.

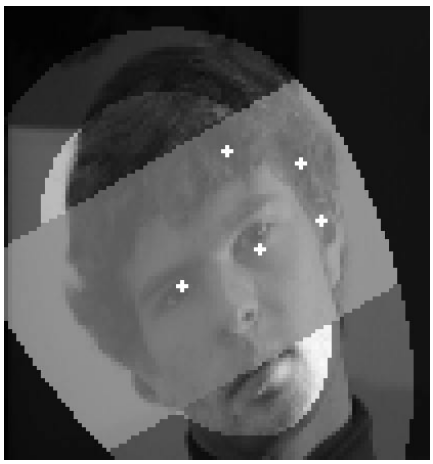


FIGURE 3.13. Strong symmetry responses from structures other than eyes

2.2. Geometrical Tests We test a pair of symmetry peaks to see if their position on the face is geometrically valid. If these peaks are not horizontally aligned or have insufficient intra-ocular distance, they could not be eyes and are to be discarded.

Horizontal Alignment of Eyes: The first test computes the line formed by the pair of symmetry peaks. This line should be roughly perpendicular to the axis of the face as detected by the face contour estimation. Symmetry peaks that form a line that is not perpendicular to within ± 30 degrees from the face's axis could not be eyes and are discarded.

Sufficient Intra-Ocular Distance: A pair of interest peaks within the eye band must have a certain intra-ocular distance separating them. If they are too close together, they cannot be eyes. Since the dimensions of the face contour are already known, we can estimate a minimum threshold distance between the eyes. However, the intra-ocular distance varies as the facial pose changes. For example, out of plane rotation induced when the subject is not looking straight into the camera will cause a reduction of the intra-ocular distance. Additionally, as the person rotates to the left or the right, the eyes do not seem centered within the facial contour in a 2D sense. Eyes travel to either side of the face as it is rotated severely. Thus, a threshold on the intra-ocular distance should be a function of the position of the center point between the two eyes relative to the face. In a near profile shot, as depicted in Figure 3.14 the center point between the two eyes is near the left side of the facial contour (note that 'left' and 'right' are defined with respect to the image viewer, not the photographed subject). The axis of the facial contour is marked with a vertical

FIGURE 3.14. Eye midpoint not centered in facial contour

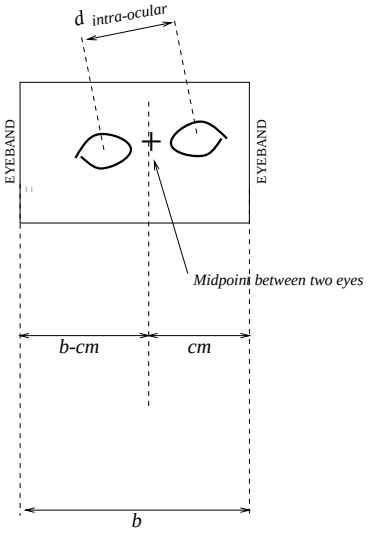
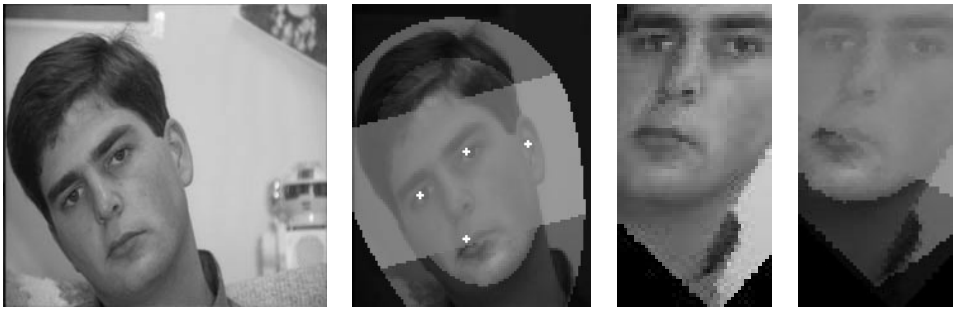


FIGURE 3.15. Minimum intra-ocular distance

line while the midpoint between the two eyes is marked with a cross. We propose to compute the threshold on the intra-ocular distance as follows (refer to Figure 3.15). We compute the midpoint between the two symmetry peaks under test which is shown in Figure 3.15. The eyes are shown in the Figure within the eye-band (of width b). The horizontal distances from the midpoint to the sides of the eye-band are cm and $b - cm$ where $cm < b - cm$. A variable threshold on $d_{intra-ocular}$ is then computed using Equation 3.1. The constant $k_{intra-ocular}$ is typically set to 0.2. This is a very conservative setting which can be tweaked if desired.

$$d_{intra-ocular} > k_{intra-ocular} \times cm \tag{3.1}$$



(a) (b) (c) (d)

FIGURE 3.16. Rotation normalization for horizontal eyes. (a) Original intensity image. (b) Result of the facial contour and eye detection. (c) Rotating the face into vertical view using the top two geometrically correct local interest peaks. (d) The image in (b) under the same rotation into vertical view.

2.3. Rotation Transformation for Mouth and Nose Detection Once a pair of interest peaks has a legal geometrical arrangement, we wish to test for the presence of a mouth and nose. If these structures are not present, then the detected pair of interest peaks was not a pair of eyes. Thus, we proceed by considering the next possible pair of eyes.

However, before we test for the presence of a mouth or nose, we rotate the face such that the two current possible eyes being evaluated lie on the horizontal. This simplifies the subsequent steps of detecting a mouth or a nose. We also rotate a mask representing the facial contour in a similar fashion to keep track of the interior of the facial contour. The rotated images of the face and the mask are displayed in Figure 3.16.

3. Mouth Localization

After having found a pair of possible eyes which satisfies the geometrical constraints imposed by the face, it is necessary to test for the presence of the mouth. This will be used not only to check the validity of the eyes but will more importantly localize the face further so that a more precise definition of its coordinates is obtained. We choose to locate the mouth after having located the eyes because it has a non-skin tone texture and stands out more clearly than the skin-covered nose. Furthermore, its position is more stable than the nose since it lies consistently between the two eyes despite rotations of the face. Thus, the next most reliable step in the hierarchy is mouth detection.

Unlike the eye region, the mouth does not have a blob-like, circular outline. When it is closed or slightly open, the mouth is a thin, elongated structure or a limb. Thus, it may not be detected by the symmetry transform's interest map. We propose the use of a limb

extraction stage as outlined by Kelly [**20**] to detect the mouth. The limb extraction process begins with the computation of the points of symmetry as displayed in Equation 2.2. This is an intermediate computation that was used to compute the interest map. Furthermore, only annuli of radius r=1 to r=6 are used, since the mouth's vertical thickness is slightly smaller than the vertical thickness of the eye region (from the top of the eyebrows to the bottom of the eye orbit). We only compute dark symmetry since the lips and the interior of the mouth are darker than the surrounding skin.

The result of the symmetry computation is an image at each value of r (r=1 to r=6) with each point containing 8 magnitudes for each symmetry orientation. Thus, a symmetry magnitude is computed for each point p for each r from r=1 to r=6 and for each ψ from ψ=1 to ψ=8 (8 symmetry orientations as in Figure 2.8). The resulting symmetry points are output as in Figure 2.11.

3.1. Horizontal Symmetry Projection Once the symmetry points have been computed for dark symmetry at the appropriate scale, we project the symmetry points along the horizontal. There is no need to consider non-horizontal symmetry points since the mouth is mainly a horizontally oriented limb. The symmetry maps derived by Equation 2.2 are projected using Equation 3.2. Recall that we compute only 8 values of $\psi \in \{0°, 22.5°, 45°, 67.5°, 90°, 112.5°, 135°, 157.5°\}$ (where 0° is aligned with the horizontal).

$$S_{r_{horizontal}}(p) = 2 \times S_{r,0°} + S_{r,22.5°} + S_{r,157.5°} \tag{3.2}$$

Thus, the 4 dimensional symmetry data $S_{r,\psi}(p)$ is reduced to 3 dimensions in $S_{r_{horizontal}}(p)$. The projected symmetry (or axial symmetry) maps for r=1 to r=6 are displayed in Figure 3.17. These maps are derived from the rotated intensity image in Figure 3.16 (c).

The 6 scales (r=1 to r=6) form our axial symmetry scale-space. The scale or r represents the vertical thickness of the horizontal symmetries detected in the image. A thin, closed mouth usually would generate a line of symmetry points at r=1. An open mouth, on the other hand, will generate a cloud of points at a larger r within its center. An open mouth's extremities taper off (since it is closed on both ends) regardless of its size. Thus, the mouth's extremities will appear as clouds at small r.

Note that the symmetry points in Figure 3.17 are also blurred horizontally with a 5×1 Gaussian window to improve connectivity and reduce small gaps. This allows us to

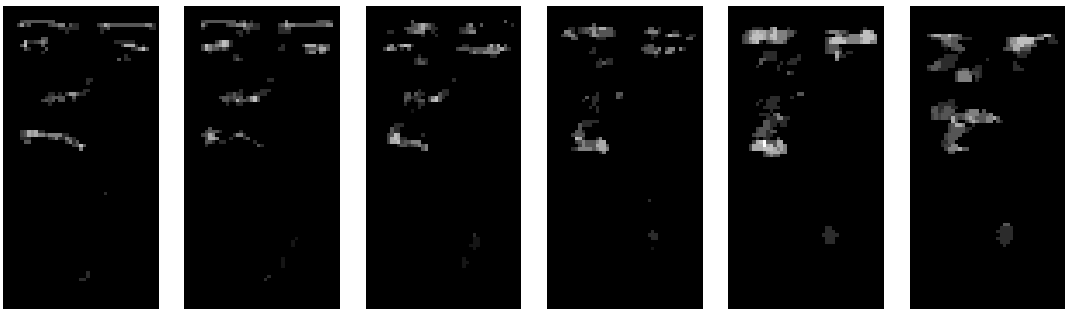


FIGURE 3.17. Horizontal Projection of Symmetry Points

use a linking procedure that connects the discrete points in the maps to form connected structures.

3.2. Limb Extraction Kelly proposes the linking of adjacent points in the axial symmetry maps as a means of recovering complete limb structures from symmetry data [**20**]. This process is simplified when only horizontally aligned symmetry lines are to be considered since limb extraction is reduced to a 3 dimensional linking process. Limb extraction involves grouping points that are adjacent in the (x, y, r, ψ) space into continuous lines. However, since we are only considering the horizontally projected symmetry lines, we only need to connect points in a (x, y, r) space. Adjacent points in the axial symmetry maps are linked using connected components analysis. Note that adjacent not only refers to spatial adjacency but also adjacency in scale-space (i.e, neighbouring values of r).

Since we are searching specifically for mouths, the connected component analysis can be further constrained. Not only is the symmetry data projected onto the horizontal, it is to be linked horizontally as well. We are searching specifically for connected horizontal limbs and not an arbitrary cloud of adjacent points from the images in Figure 3.17. The extraction will produce a set of 3D curves that flow through the symmetry data in Figure 3.17. Figure 3.18 displays the limb axes (shown with dashed lines) that should approximate the clouds (or manifolds) of symmetry points (enclosed with continuous lines) in the search space.

Furthermore, these 3D curves are to be as horizontal as possible and should not meander excessively. The mouth is a smooth limb so its axis should be a simple straight line or a slight curve. Furthermore, axial symmetry curves should not undulate excessively in scale-space, r (the third dimension). This prevents the vertical thickness of the mouth limb

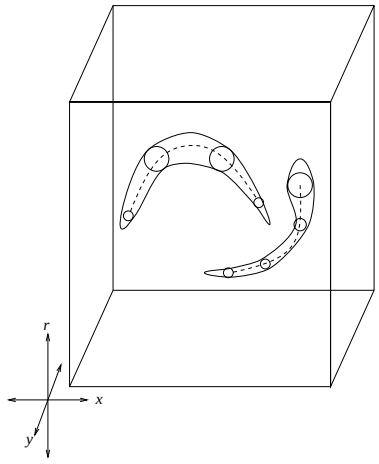
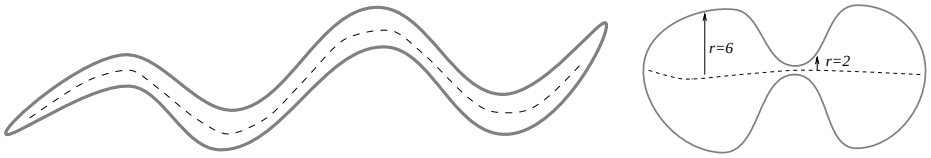


FIGURE 3.18. Limb extraction



(a) (b)

FIGURE 3.19. Excessive limb undulation. (a) Excessive spatial meandering. (b) Excessive meandering in scale space.

from varying wildly. Thus, mouth limbs which are maximally straight will be favored. Figure 3.19 displays limb extractions which do not satisfy these criteria and hence are probably not mouths. Such limbs should be attenuated during extraction so that they will not interfere with mouth detection. This attenuation agrees with the Gestalt psychology notion of "Pragnanz" which identifies a correspondence between simplicity in image structures and their perceptual significance [**3**].

Connected component analysis begins at a given point (p, r) in the 3D search space. This 3D search space covers values $r{=}1$ to $r{=}6$ and the (x, y) region depicted in Figure 3.21. This starting point ((p, r) or (x, y, r)) will be called the "seed". One of the requirements we impose is that the seed has a significant horizontal symmetry magnitude (25% of the maximum possible magnitude). From this starting point, we form a limb or 3D curve by propagating along two trajectories: to the left and to the right.

By left propagation, we seek a path towards the left of the current pixel (p, r) which flows through the strongest horizontal symmetry points in our 3D search space. Along the

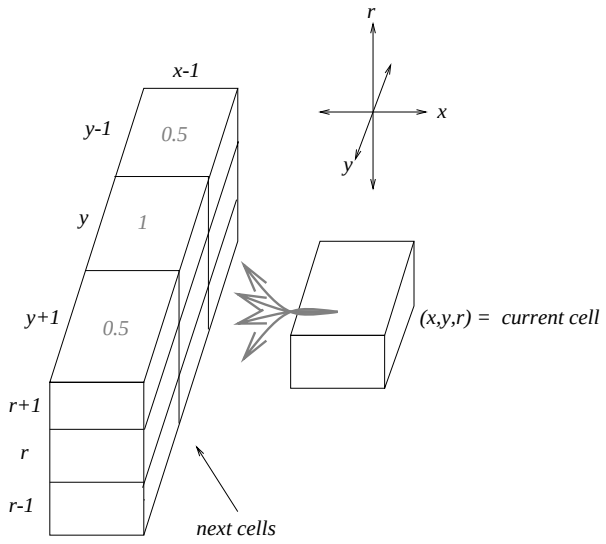


FIGURE 3.20. Left limb trajectory propagation kernel. The trajectory must flow along the arrows from the current cell to one of the 9 cells in the kernel to its left.

left propagation, we utilize a kernel that covers all the adjacent neighbours of the current point $(p, r) = (x, y, r)$ to the left of the current pixel. This kernel is depicted in Figure 3.20. This kernel contains 9 cells of which we must pick one. This new cell becomes the new "current cell" and we repeat the process, tracing out the next step in the path from the 9 possible choices in the kernel. This propagation is continued and the trajectory flows through the 3D space as it repeatedly selects from the 9 possible cells in the kernel.

The symmetry magnitudes are scaled by the weights depicted in the kernel in Figure 3.20. We thus measure a weighted symmetry value from the horizontal projection data for each cell in the kernel. The pixel location to the immediate left is favored most and its symmetry magnitude is consequently scaled by $1\times$. We simultaneously disfavor diagonally positioned pixels by scaling their magnitude by $0.5\times$. We begin by only considering the spatial neighbourhood at scale r. The peak weighted response from the three cells at r is determined. If it is greater than the threshold (25%) then we move (propagate) to the strongest cell's position and re-compute the kernel from there. If on the other hand, the scale at r has only weak symmetry, we repeat the analysis at $r + 1$ and $r - 1$ and propagate to the strongest of those 6 cells. Thus, we seek a path towards the left of the current pixel which flows through the strongest horizontal symmetry. However, we favor paths that are horizontal (limiting spatial meandering) and paths that are at the same scale (limiting meandering in scale). Equation 3.3 illustrates the weighted computation of the peak-response of the strongest cell in the kernel. The cell generating the peak-response, *peak* is the one

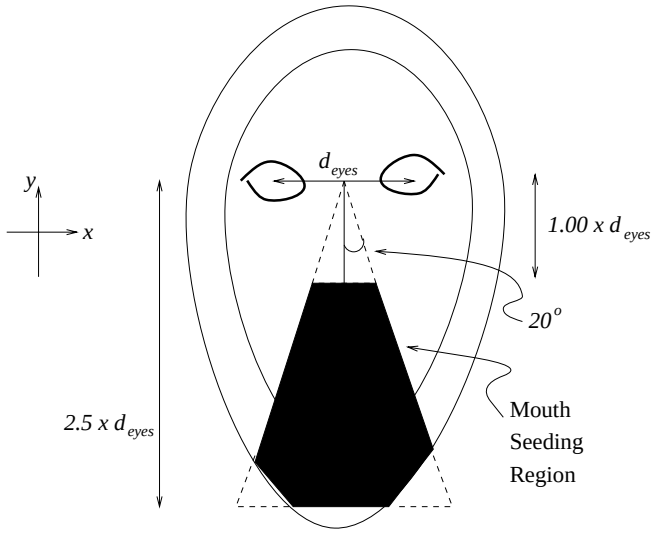


FIGURE 3.21. Search space for seeding mouth limb extraction

we move to next. It becomes the "current cell" and we repeat the kernel computation from there. The process stops when a dead end is reached and none of the 9 possible cells contain an output greater than the threshold (25%). In Equation 3.3, the horizontal symmetry values are labelled S where $S_{(x,y,r)} = S_{r_{horizontal}}(x, y)$.

$$(3.3) \quad peak = \left\{ \begin{array}{c} M_1 = max(\frac{1}{2}S_{(x-1,y-1,r)}, S_{(x-1,y,r)}, \frac{1}{2}S_{(x-1,y+1,r)}) \ : M_1 > 25\% \\ M_2 = max(\frac{1}{2}S_{(x-1,y-1,r\pm1)}, S_{(x-1,y,r\pm1)}, \frac{1}{2}S_{(x-1,y+1,r\pm1)}) \ : M_1 \leq 25\% \end{array} \right\}$$

A similar propagation is performed to the right of the seed (or starting cell) and the two trajectories are merged into one. This single trajectory in the spatial and scale domain represents the extracted limb as a whole.

Figure 3.21 shows (in black) the (x, y) region in the image where the mouth limb starting points will be selected. The dimensions of this triangular search region are defined in the figure. Note that the triangle does not extend beyond the face mask so that the mouth search is performed exclusively within the face contour. Any part of the mouth that intersects this triangular search space will trigger limb extraction. Thus, the mouth does not have to be perfectly centered upon the face's mid-line. This can happen if the face has an unusual pose or the eye detection is slightly inaccurate. The triangular search space is superimposed upon the rotated intensity image in Figure 3.22. Observe how the mouth is not perfectly aligned with the face's mid-line in Figure 3.22. However, it still falls within the generous triangular search region and is thus detectable.

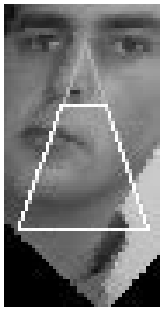


FIGURE 3.22. Search space for mouth superimposed on face

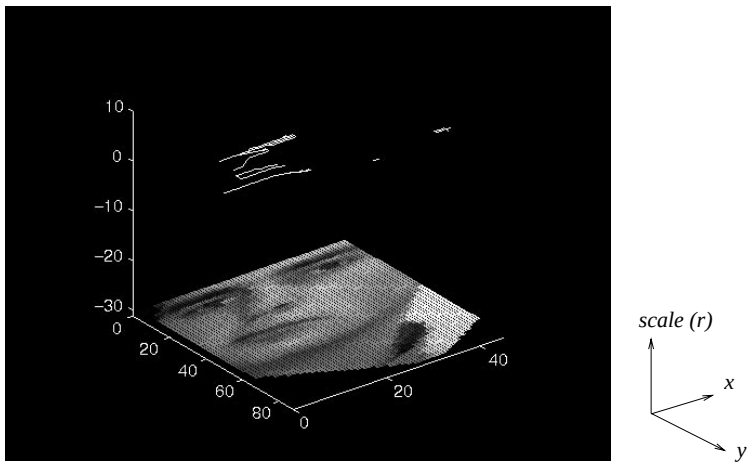


FIGURE 3.23. Limb axes extracted as 3D trajectories

Note that this triangular search space is rather redundant. Say a horizontal limb is detected at a seed position (x, y, r) on the image. A very similar limb will probably be detected when we start the limb extraction at $(x + 1, y, r)$. Thus, we do not need to start the limb extraction process at every point in the triangle. We merely start the limb extraction once for every value of y in the triangle. For each value of y we select the seeding point (x, y, r) by varying x and r to maximize $S_{r_{horizontal}}(x, y)$. Then, we begin the limb extraction and extract via propagation the trajectory of the limb towards the left and the right. This is repeated for each value of y in the triangle resulting in a collection of limb axes.

The resulting horizontal limb axes are stored as 3D curves (or trajectories) in the spatial-scale domain as shown in Figure 3.23. The height in this graph represents the scale or the dimension r of the trajectory. This value of r corresponds directly to the thickness

FIGURE 3.24. Intensity variance

of the limb being extracted. Now, we must test these limbs to determine which of them is the mouth. Some particularly short limbs are present in Figure 3.23 despite the Gaussian filtering. These limbs will be discarded later since they are too short.

3.3. Limb Length and Intensity Variance Having formed a set of extracted limbs, we next determine the most perceptually significant limb and return it as the mouth. Another test might be to find the limb which most closely resembles the geometry of a mouth. However, the geometry of the mouth is not fixed due to its extreme deformability. Thus a simple geometric constraint might reduce the expression invariance of the detection. We determine which of the limbs is most perceptually significant using two measures: limb length and intensity variance.

We propose the computation of limb length as opposed to limb area since a mouth can be extremely thin when closed. However, the mouth usually has a significant horizontal length. Recall, now, the kernel used to perform the limb extraction. The kernel favored horizontal displacement over diagonal displacement and scale change. Similarly, as we compute limb length, we weight the computation of limb length depending on its degree of undulation (in the spatial and scale domains). Thus, a set of limb lengths are computed and attenuated by the degree of undulation of the limb axis, as shown in the kernel in Figure 3.20. For each limb, $limb_i$, we compute the weighted length of the limb, d_i. We threshold this value so that limbs that are extremely short are discarded.

We also wish to determine the intensity variance of the isolated limb and compare it to the tone of the face. The lips, the interior of the mouth and the teeth are either brighter or darker than the surrounding skin. Thus, the intensity values enclosed by the mouth limb should have a significantly different intensity from the average intensity of the face. We compute the mean intensity of the skin, m_{face}, by averaging the intensity values below the eyes and within the facial contour. We then compute the variance in intensity at each

FIGURE 3.25. Intensity variance with mouth locus

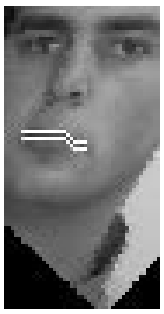


FIGURE 3.26. The segmented mouth limb

pixel as shown in Figure 3.24. The overall average intensity variance of the face within the facial contour is σ_{face}. The intensity variance of the region defined by each limb, σ_i, is then computed. If a limb has less variance than the average variance of the face then it does not exhibit any significant contrast or stand out strongly from the rest of the face. Such limbs cannot be mouths and are discarded.

The mouth is selected as the limb with the highest product $d_i \times \frac{\sigma_i}{\sigma_{face}}$. The center point of the strongest limb (the locus of the mouth) is displayed superimposed on the variance image in Figure 3.25. Since we know the trajectory of the limb axis and the thickness of the limb, we can directly compute the outline of the mouth, which is shown in Figure 3.26.

Instead of explicitly defining a geometric model of the mouth that is sensitive to multi-dimensional deformability, we have used a simple "definition" to localize the object of interest. Simply stated, we find the mouth as the longest horizontally symmetric limb, with a simple axis and significant intensity variance from the surrounding skin tone.

If we fail to find any limbs that are long enough or have a higher variance than the face as a whole, then no mouth has been detected and we return to the eye stage to investigate another pair of interest peaks.

A successful mouth point is found when the limb with the highest product $d_i \times \frac{\sigma_i}{\sigma_{face}}$ passes a threshold on d_i and a threshold on $\frac{\sigma_i}{\sigma_{face}}$. The locus of the mouth is then stored and we can proceed to the nose localization stage.

4. Nose Localization

If a successful mouth point has been found, we can add this data to our knowledge of the current face. It is then relatively easy to search for a nose at this stage due to the well defined search space that can be cut out. The nose is a very useful feature since it accurately gives us an estimate for the pose of the individual. This is due to the significant displacement the nose undergoes in a 2D sense as facial pose changes. The nose position relative to the eyes tells us quite precisely if the subject is looking to the left, to the right or is in frontal view. The mouth and the facial contour, on the other hand, are not as reliable for estimating pose. Furthermore, the nose is mostly rigid, so its locus cannot change with facial expression.

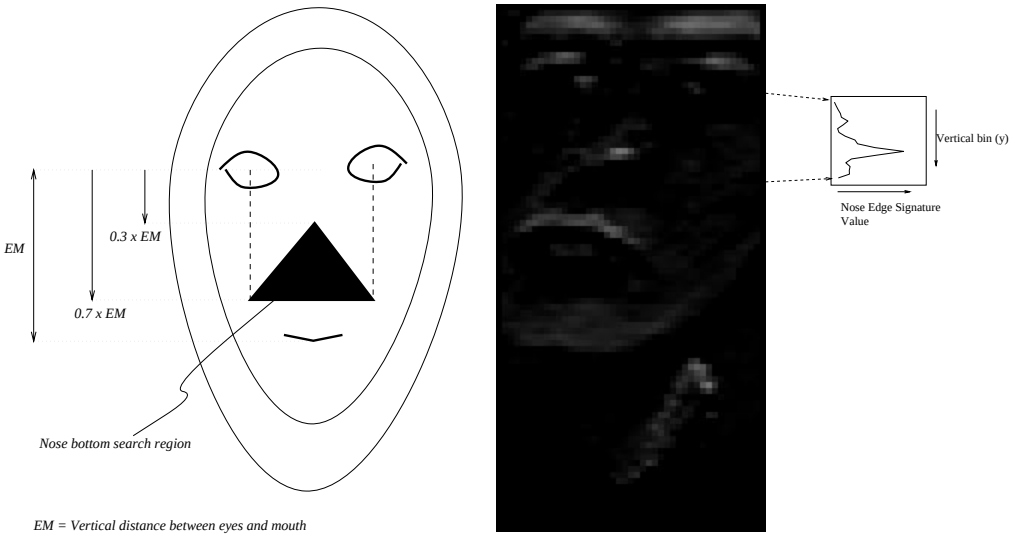
In most images, the nose is one of the brightest regions of the face. It protrudes from the face and is thus better illuminated than other regions. Simultaneously, the nostrils and its bottom surface are significantly darker than the rest of the nose. Even if the black nostrils are not present, a dark contour around the bottom of the nose is visible due to the shading under the nose and the steep foreshortening at the bottom of the nose tip. Thus, we can model the nose as a region of brightness with a dark boundary on the bottom.

We are interested in detecting this change of intensity from brightness to darkness as we travel from the eyes to the mouth. From the gradient and phase maps derived by Sobel edge detection, we can compute the projection of the gradient magnitude of each edge along the vertical. Thus, we only consider vertical contrast changes. Actually, more specifically, we consider contrast changes that occur from bright to dark as we move *downwards* along the vertical. Figure 3.27(a) contains the original gradient map and Figure 3.27(b) shows the effect of projecting the edges along the upward vertical. Equation 3.4 illustrates the projection of an edge i with magnitude λ_i and phase ϕ_i (where $\phi = 0$ corresponds to a vertical edge whose normal is along the horizontal). This generates the horizontally projected magnitude value, λ_i'.

$$\lambda_i' = \lambda_i \times |\sin(\phi_i)| \tag{3.4}$$

(a) (b)

FIGURE 3.27. Nose edge data. (a) Sobel gradient map. (b) Gradient map projected along vertical.



(a) (b)

FIGURE 3.28. Nose height. (a) Search space. (b) Gradient map projected along upwards vertical with corresponding nose signature. Note that this is not quite a conventional signature since a triangular summation region is used.

4.1. Vertical Signatures We now consider the use of vertical signatures of the projected gradient map to isolate the nose. This technique is reminiscent of Kanade's [**19**] signature analysis.

We define the spatial search space using the previously localized eyes and mouth. This will restrict the signature analysis so that no facial contours or external edges will affect edge signature analysis. Figure 3.28 shows the region where signature analysis will be performed. The edges contained by this triangle are summed into bins corresponding to their y value.

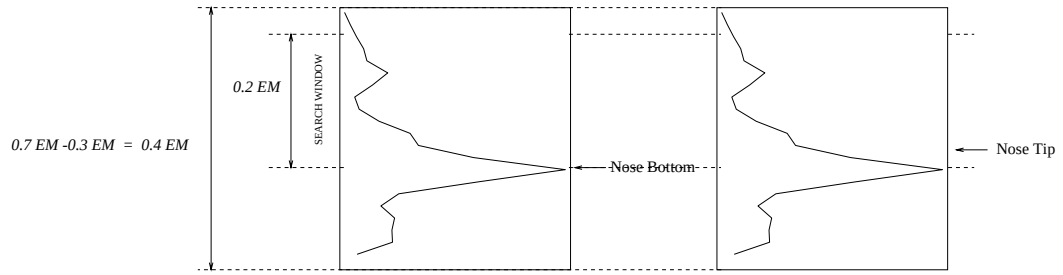


FIGURE 3.29. Finding the nose tip from the nose bottom using the edge signature.

These bins form the vertical signature of the nose. The bin with the strongest edge content is the one that corresponds to the vertical position of the nose's bottom. Figure 3.28 also shows the projected gradient map and the signature that was computed in the search space. The nose's bottom position corresponds to the peak value of the signature.

However, we are interested in the nose tip, not the nose bottom. The nose tip characterizes 3D pose more clearly since it strongly protrudes from the ellipsoidal structure of the head. Assume the nose bottom was detected at position $noseBottom_y$ at the peak signature value of $noseBottom_{value}$. We search a window of height of up to $0.2 \times EM$ above the nose bottom for a weaker signature value. The nose tip is defined as the closest point in the window with a signature value below $40\% \times noseBottom_{value}$. This simple adjustment is depicted in Figure 3.29. The positions of the nose bottom and the nose tip are shown as horizontal lines in Figure 3.30. Note the effect of this computation is quite minor and the nose-tip is only 2 pixels above the nose-bottom. Although the definition of the nose-tip and the use of the 40% threshold are somewhat arbitrary, we merely wish to move out of the region corresponding to the nose bottom (nostrils and shading) by a marginal amount so that the position detected has a 3D height. In other words, we wish to move upwards a small distance so that we localize a point somewhere on the nose, taking advantage of its 3D protrusion on the face (which specifies pose more exactly than non-protruding features on the face). Furthermore, the small upwards adjustment from nose-bottom to nose-tip does not have to be exact as long as we are somewhere on the nose and not on the junction between the nose-bottom and the face (which is not a 3D protrusion). Usually, the nose tip is brighter than the rest of the nose and the nose bottom is darker. However, the transition from nose tip to nose bottom or bright to dark is somewhat gradual. By moving upwards in search of a 40% signature value (rather than the maximum), we are searching for the beginning of this transition and moving closer to the true nose position in the process.

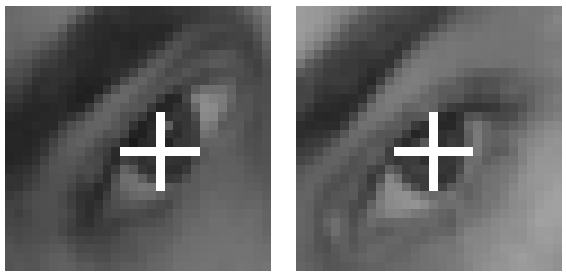
FIGURE 3.30. The nose-bottom-line and the nose-tip-line.

Thus, we have roughly determined the height of the nose tip with respect to the eyes. However, we are uncertain of the exact horizontal position of the nose. The required localization is difficult to perform using simple signature analysis. This is mainly due to the fact that noses have the same tone as the rest of the skin-covered face and hence have low perceptual significance. Thus, a more sensitive nose finding technique will be utilized to isolate the horizontal position of the nose. This technique requires the introduction of normalization and recognition algorithms which will be detailed in Chapter 4. For now, the nose-localization module merely outputs a height value at the nose-tip position so we do not have a single locus for the nose but, rather, a line of possible loci for the nose. This nose-line lies between the two eyes at a fixed perpendicular distance below them. Thus, the output of the nose detection defines a nose-line (as opposed to a nose locus) along which the nose is situated as depicted in Figure 3.32. The nose-line crosses the nose tip and is parallel to the line formed by the two eyes. Additionally, its length is equal to the intra-ocular distance. In other words, the nose-line starts below the left eye and ends below the right eye.

5. Iris Localization

The locations of the mouth and the two eye regions give a fairly stable measure of the size of a face. We can use these feature points to compute EM, the distance from the mid-point between the eyes to the mouth. From sample measurements, the value of EM was found to be reliable enough to predict the radius of the iris of a subject's face. The iris radius is typically expected to fall between 5% and 15% of the value of EM.

Consequently, the real-time symmetry transform is reset to use annular sampling regions that cover a radius between 5% and 15% of the value of EM. The transform is then utilized to compute two small interest maps around the previously located eye regions. The



(a) (b)

FIGURE 3.31. Iris Localization. (a) Left iris as an interest map peak ('+'). (b) Right iris as an interest map peak ('+').

peaks of these interest maps indicate possible positions of the left and right iris. Figure 3.31 shows the resulting iris loci.

The search space for the iris is centered around the old eye locus and is a square with length 25% $\times EM$ on each side. Figure 3.31 shows the actual search space windows with the best iris position represented with a + symbol. The strongest interest peak in that window is used as the iris position. The interest map is thresholded to discard all peaks which trigger symmetry values below 25% of the maximum possible output. This extra threshold allows us to avoid triggering the iris finder with other structures such as the eye brows. These and other objects generate a weak response and the iris finder might erroneously converge to their loci if the iris is not clearly visible (i.e. subject is squinting). Thus, the threshold allows us to report the absence of an iris in the search space if the peak response is too weak. Consequently, no valid peaks in the interest maps are found, and the iris localization function can merely default to the previously calculated position of the eye region. Therefore, if the individual in the image is squinting or the eyes are not clearly visible, we use the large, coarse eye-blob detection output instead of the iris finder as the position of the iris.

6. Improving and Filtering Localization

The localization procedure thus yields an output similar to the one found in Figure 3.32. Note that the horizontal (i.e., parallel with intra-ocular axis) position of the nose is uncertain. This is represented by a white line across the vertical position of the nose from the left eye to the right eye. The nose tip is a point on this line segment and its exact locus is found by the techniques described in Chapter 4. Furthermore, it is possible that the face detection computation will be triggered by a non-face which happens to have a blob-like

FIGURE 3.32. A typical output of the face detection stage.

structure with eye blobs and a mouth-limb. These 'false alarms' by the face detector will be rejected using techniques described in Chapter 4.

Thus, we have given a procedure for localizing the face and facial features in an image. We begin by finding the face-blob and then estimating a facial contour. From there, we can define the eye-band, a region where the eyes might be present. Eye-blob detection is then performed and the blobs are tested geometrically to see if an adequate pair can be found. If two blobs are "eye-like" geometrically, we search for a mouth between them and then, finally, a nose-line. At this stage, however, the localization is not complete and requires further development in Chapter 4. Consequently, we shall defer localization testing to Chapter 5. Several examples of the complete localization are given there. It is also important to note that the exact location of the facial features in Figure 3.32 (and other faces that are processed) is not critical. This is because this localization is to undergo further processing before face recognition is performed. This post-processing will desensitize the recognition to small localization errors.

CHAPTER 4

Face Normalization and Recognition

The position of a rigid object can be specified by 6 parameters: 3 rotations and 3 translations. The rigid motion of a face or any object is specified by these 6 parameters. Rigid motion of the face accounts for a great amount of variance in its appearance in a 2D image array. Furthermore, the lighting changes caused by light sources at arbitrary positions and intensities also account for a significant amount of variance. Simultaneously, the non-rigid deformations of the face (from muscular actuation and identity variation) cause more subtle variations in the 2D image. An individual's identity, however, is captured by these small variations alone and is not specified by the variance due to the large rigid body motion and illumination of the face. Thus, it is necessary to compensate or normalize a face for position and illumination so that the variance due to these is minimized. Consequently, the small variations in the image due to identity, muscle actuation and so on will become the dominant source of intensity variance in an image and can thus be analyzed for recognition purposes.

Recall the output of the face detection and localization stage. The eyes, the nose and the mouth were identified using direct image processing techniques. Assume for now that the nose's horizontal position was also determined and an exact locus for the nose tip is available. The detection of the loci of these feature points (eyes, nose and mouth) gives an estimate of the pose of an individual's face. Once the pose or the 3D position and orientation of the face is known, it is possible to invert the effect of translation and rotation and synthesize a standardized, frontal view of the individual. Furthermore, the position of the feature points allows us to roughly segment the contour of the face to discard distracting background information. Once segmented, a histogram of the face alone can be computed to compensate for lighting changes in the image.

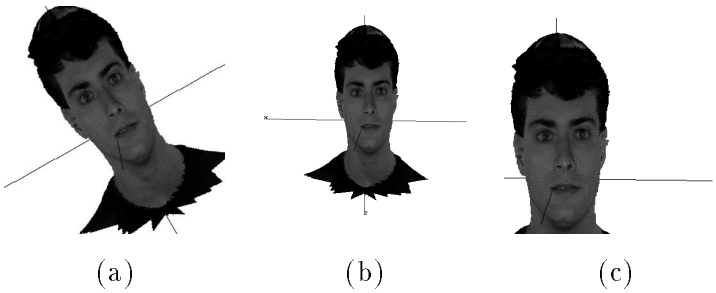


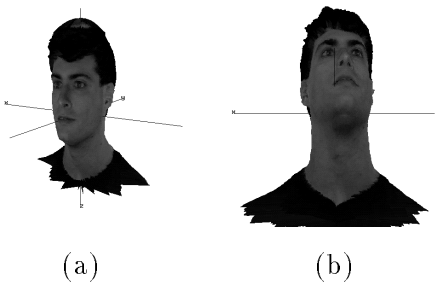
FIGURE 4.1. In-plane rotation, scaling and translation.

We follow the scheme described above to generate a normalized mug-shot image. It is then possible to analyze the variances left in the image using linear statistical techniques. We use these techniques to classify and characterize the variances that remain in the image since they are now constrained and limited. Statistical analysis of the Karhunen-Loeve Decomposition (KL) allows us to verify face detection and to improve feature localization by computing a "faceness" measure which quantifies how face-like an image region is. This "faceness" measure lets us try different loci for the nose and select the position which maximizes this statistical similarity to a fully frontal face. Finally, the KL-encoded faces are matched to each other using a nearest-neighbour classifier such as the Euclidean distance in the KL-space for recognition.

1. 3D Face Data for Normalization

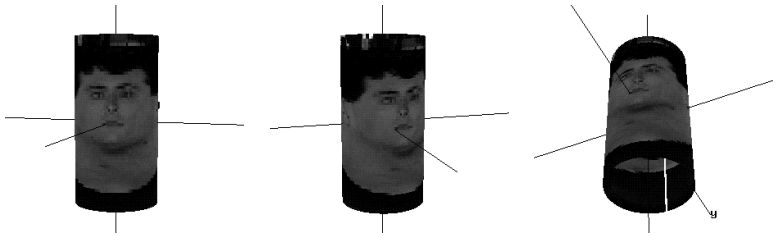
Several methods have been proposed to normalize a face's pose using a number of anchor points. Akamatsu et al. use an affine transformation which maps the triangle formed by three vertices (corresponding to the eyes and the mouth) into a standard view [**1**]. This normalization technique treats the face and the rest of the image as a thin sheet which can be scaled, rotated and sheared. This technique can account for translation, scaling and in-plane rotations of the face since these are only 2D effects that are not dependent on the 3D structure of the object. These in-plane rotations and translations are depicted in Figure 4.1. However, a rigid object has two more degrees of rotational freedom which change its 2D projection in a non-homogenous way, as shown in Figure 4.2. These can not be compensated for by a mere affine transformation since they induce non-homogenous warping and occlusion in the image and hence require a more sophisticated approach.

An alternate model for the face is an ellipsoid or other simple geometric structure such as a cylinder as in Figure 4.3 [**5**]. Unlike the "thin sheet" model which cannot account



(a) (b)

FIGURE 4.2. Out-of-plane or depth rotations.



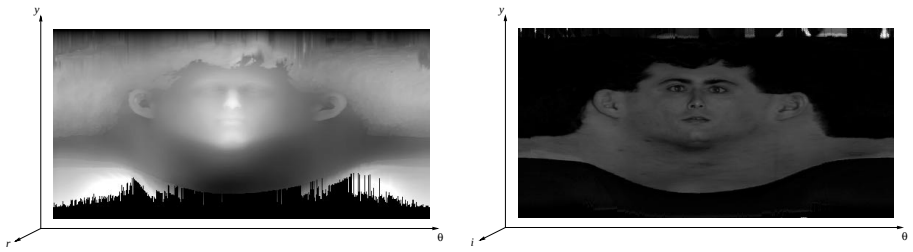
(a) (b) (c)

FIGURE 4.3. A cylinder as a geometric 3D model of a face. (a) The face, texture-mapped upon a cylinder in an up-right position. (b) Rotating the cylinder on its vertical axis (pitch). (c) Rotating the cylinder about its vertical axis (yaw). Note that the face does seem to be turning to its left and looking upwards in (b) and (c) but it is strangely warped by the geometry of the cylinder.

for yaw or pitch, the ellipsoid has the ability to roughly mimic the out-of-plane rotations the face can undergo. This is due to the curvature of the ellipsoid which exhibits non-homogenous warping in a 2D sense. Unfortunately, a simple ellipsoid cannot encompass all the nuances of the face and fully normalize its 2D projection. For example, the nose can cause occlusion by rotating in front of the cheek. In addition, the human head is not quite ellipsoidal and is difficult to approximate with standard 3D geometric models.

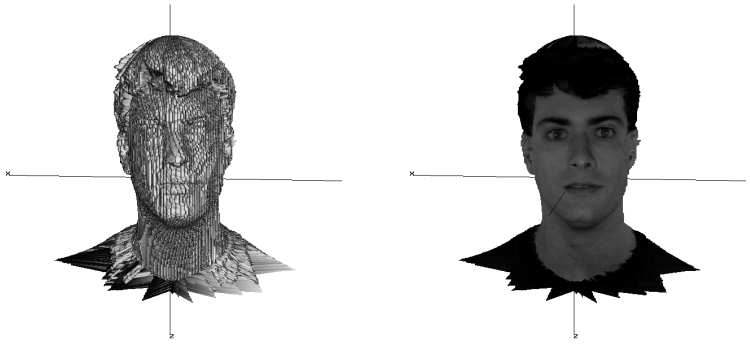
Clearly, the most accurate 3D model of a face would be the true 3D range data of the individual obtained from laser range-finder scanning. This cumbersome process is not only time-consuming and non-automated, it requires the use of sophisticated equipment which is not readily available[1]. Some sample data obtained from such devices is shown in Figure 4.4 as radial range and radial intensity images. The images are in a cylindrical coordinate system and the axes are appropriately labelled.

[1] Devices capable of performing such data acquisition include Cyberware's laser range scanners and Magnetic Resonance Imaging equipment.



(a) (b)

FIGURE 4.4. Radial range and intensity images. (a) Radial range data. (b) Radial intensity data.

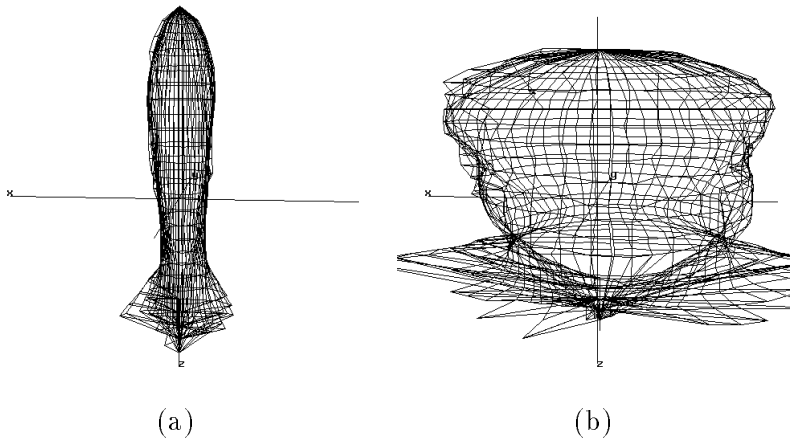


(a) (b)

FIGURE 4.5. Rendered 3D model from radial range and intensity data. (a) Shaded 3D model. (b) Texture-mapped 3D model.

From the radial range data, we compute a polygonal mesh by converting the cylindrical coordinates into Cartesian form. The Cartesian 3D data can then be rendered and displayed as shown in Figure 4.5(a). Subsequently, we can "colorize" the 3D model with the radial intensity data and obtain a texture-mapped 3D model of the individual as shown in Figure 4.5(b). This 3D model can then be used to synthesize any view of the individual by treating the head as a rigid object and rotating and translating it with 6 degrees of freedom (see Figure 4.1 and 4.2).

Unfortunately, we do not and cannot have a 3D model for each individual that we will photograph for our recognition system. Thus, we shall attempt to use another individual's 3D model to normalize the photograph under the assumption that the 3D structure of most faces is somewhat constant. Therefore, we can use one 3D model of a face and texture-map new photographed faces onto it. Unfortunately, some individuals will have thinner or



(a) (b)

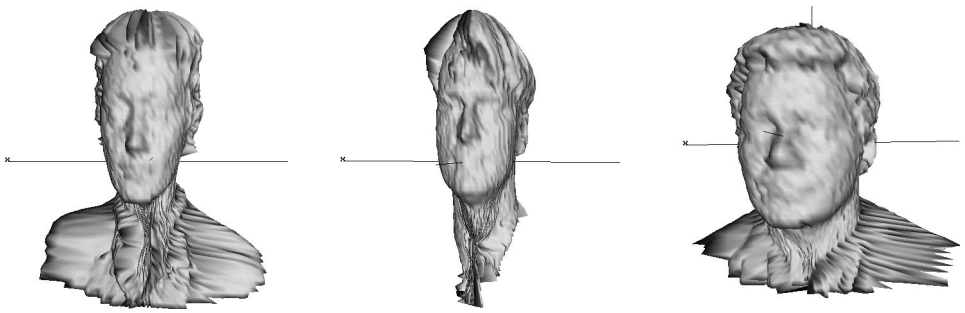
FIGURE 4.6. Deforming the 3D model along its vertical axis. (a) Stretched model. (b) Squashed model.

wider faces and the model will not fit them as well as it did with the original texture. We suggest deforming the model along its vertical axis to stretch or squash it to fit it to the new face, as shown in Figure 4.6. Ideally, we would like to deform the model arbitrarily with various small stretchings and warpings so that it can be locally adapted to each new individual. However, such a process is quite computationally expensive. Nevertheless, the single vertical stretch of the model and its six degrees of freedom gives us quite a good approximation of the faces we will encounter and is, by far, more accurate than the planar or ellipsoidal models used in previous experiments.

2. Generating the Average 3D Face

Although the sample face in Figure 4.4 is a typical human face, we choose to use an average 3D face from a database of previously sampled faces to obtain a smooth, mean 3D face. Figure 4.7 shows a few of the 3D range data models we used to obtain the average 3D face.

In averaging the 3D faces in a database, we wish to see the mean 3D face converge to a stable structure as we introduce more sample 3D faces. We also expect the mean 3D face to be "face-like" in the sense that the averaging process will not smooth out its features to the point where they are no longer distinguishable. In other words, the mean 3D face should still have a nose, a mouth, eyes and so on. If we do not see this convergence and the mean face is a mere blob or ellipsoid, then our hypothesis is incorrect: the 3D structure of a human face is not regular enough to approximate multiple individuals. Another possible source of divergence is inadequate normalization before the averaging process. If the 3D



(a) (b) (c)

FIGURE 4.7. Some of the 3D faces used to form the average 3D face.

faces in our database are not fully normalized before being averaged, then the mean face will not be face-like.

For each face in our 3D range data database, we manually select 4 points: the left eye, the right eye, the nose and the mouth and note their 3D coordinates. Each model in the database undergoes a 3D transformation with a vertical stretch to map its 4 anchor points to the same destination set of anchor points. Mathematically, the four 3D anchor points: $(\vec{n}_1, \vec{n}_2, \vec{n}_3, \vec{n}_4)$ for each model, are mapped to a destination set of 3D anchor points: $(\vec{m}_1, \vec{m}_2, \vec{m}_3, \vec{m}_4)$. This mapping is given in Equation 4.1 where matrix T is defined as follows:

$$T = \begin{Bmatrix} \cos\theta_y \cos\theta_z & -s_y \cos\theta_y \sin\theta_z & \sin\theta_y & t_x \\ \cos\theta_z \sin\theta_x \sin\theta_y + \cos\theta_x \sin\theta_z & s_y(\cos\theta_x \cos\theta_z - \sin\theta_x \sin\theta_y \sin\theta_z) & -\cos\theta_y \sin\theta_x & t_y \\ \sin\theta_x \sin\theta_z - \cos\theta_x \cos\theta_z \sin\theta_y & s_y(\cos\theta_z \sin\theta_x + \cos\theta_x \sin\theta_y \sin\theta_z) & \cos\theta_x \cos\theta_y & t_z \end{Bmatrix}$$

$$\begin{Bmatrix} x_f \\ y_f \\ z_f \end{Bmatrix} = T \begin{Bmatrix} x_i \\ y_i \\ z_i \\ 1 \end{Bmatrix} \tag{4.1}$$

Using ten 3D models, the best transformation matrix was found by optimizing the 7 parameters $(t_x, t_y, tz, \theta_x, \theta_y, \theta_z, s_y)$ to minimize the fitting error, E_{fit} as defined in Equation 4.2 below. There are 3 translation parameters (t_x, t_y, tz), 3 rotation parameters $(\theta_x, \theta_y, \theta_z)$ and one vertical stretch parameter (s_y):

$$E_{fit} = \sum_{i \epsilon \{1,2,3,4\}} \sqrt{(n_{i_x} - m_{i_x})^2 + (n_{i_y} - m_{i_y})^2 + (n_{i_z} - m_{i_z})^2} \tag{4.2}$$

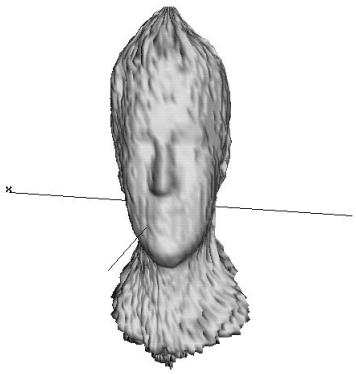


FIGURE 4.8. The average 3D face

The final average 3D face range model is shown in Figure 4.8. This is the only model that will be rotated, translated and deformed to approximate the structure of new faces and the other 10 database models are now discarded. As can be seen, the 3D mean face is a smooth, face-like structure with distinct features. The coordinates of the features (eyes, nose and mouth) are stored with the 3D model as $(\vec{m}_1, \vec{m}_2, \vec{m}_3, \vec{m}_4)$ for later use.

3. Inverse 3D Projection

It is essential to fit the 3D range data of the mean face so that it models a face found in a 2D image. Assume we have a 2D image with the coordinates of the eyes, the nose and the mouth perfectly pinpointed, as in Figure 4.9. Denote the 2D positions of the eyes, nose and mouth as $(\vec{i}_1, \vec{i}_2, \vec{i}_3, \vec{i}_4)$. The parameters of the 3D model $(t_x, t_y, tz, \theta_x, \theta_y, \theta_z, s_y)$ must be tuned to align its 3D anchor points $(\vec{m}_1, \vec{m}_2, \vec{m}_3, \vec{m}_4)$ so that their 2D projections coincide with the set of 2D anchor points $(\vec{i}_1, \vec{i}_2, \vec{i}_3, \vec{i}_4)$. Note that the alignment to the destination points $(\vec{i}_1, \vec{i}_2, \vec{i}_3, \vec{i}_4)$ involves minimizing 8 distances since each of these points is a 2D position. We observe that Equation 4.1, which only has 7 degrees of freedom, is over-specified. Thus, there is usually no exact solution to the fitting problem, only approximations.

3.1. P3P - Fitting the 3D Model to a 2D Image If the deformation variable, s_y is held constant, only 6 degrees of freedom remain in Equation 4.1 and an exact solution which involves 6 distances can be found to the problem of fitting to three 2D points $(\vec{i}_1, \vec{i}_2, \vec{i}_3)$. At this stage, we choose to use $(\vec{i}_1, \vec{i}_2, \vec{i}_3)$ which correspond to the eyes and the nose to perform the fitting since we consider their loci to be more accurate and more stable than the mouth's locus. The fitting then reduces to the well-known Perspective-3-Points (P3P)

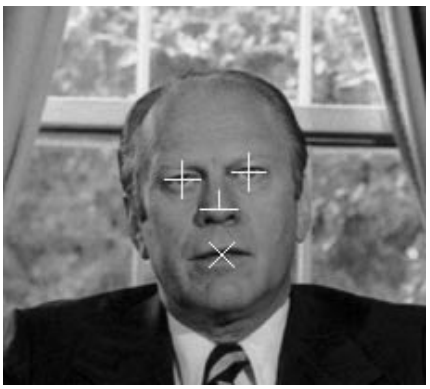


FIGURE 4.9. Image of U.S. President Ford with eyes, nose and mouth located

problem [**13**]. A direct solution is possible due to the 1-to-1 nature of the problem. There exist more complex solutions capable of finding optimal fits for over-constrained multi-point problems [**26**], [**18**] and [**16**]; however, we shall defer the fitting to the 4th point (the mouth) for now.

It is important at this stage to consider perspective. In 3D rendering, perspective generally enhances the realism of a scene. Buildings and geometric objects appear to taper off or shrink as they move away from the user in distance. If, however, the size of the object in depth is relatively small, the perspective effect is negligible. The effect of perspective on face images is subtle to the human eye and it is thus possible to render 3D face data without such perspective computations. We shall utilize this simplification and approach the solution of fitting to three points using the Weak-Perspective-3-Points (WP3P) technique [**2**]. A brief summary of the computation of the solution to WP3P is presented here, but for an in-depth analysis of the derivation the reader should consult [**2**].

Observe Figure 4.10 which depicts the desired, scaled orthographic projection of the model upon the image plane. The intra-point distances in the figure are (R_{01}, R_{02}, R_{12}) for the model and (d_{01}, d_{02}, d_{12}) for the image object. The overall scaling the model needs to undergo to fit the image is defined as s. The vertical heights from the image plane of the *aligned* model's two vertices are (H_1, H_2) before scaling or (h_1, h_2) after scaling. The parameters in Figure 4.10 are computed using Equations 4.3, 4.4 and 4.5:

$$s = \sqrt{\frac{b + \sqrt{b^2 - ax}}{a}} \tag{4.3}$$

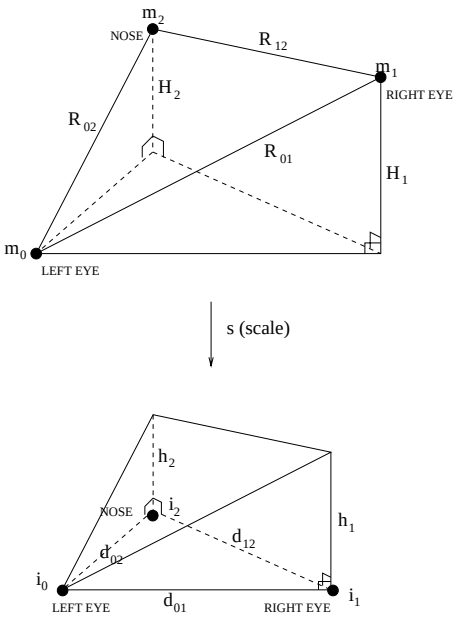


FIGURE 4.10. The scaled orthographic projection of the model upon the image plane

$$(h_1, h_2) = \pm(\sqrt{(sR_{01})^2 - d_{01}^2}, \sigma\sqrt{(sR_{01})^2 - d_{01}^2}) \tag{4.4}$$

$$(H_1, H_2) = \frac{1}{s}(h_1, h_2) \tag{4.5}$$

The intermediate variables are defined in Equations 4.6, 4.7, and 4.8 and 4.9:

$$a = (R_{01} + R_{02} + R_{12})(-R_{01} + R_{02} + R_{12})(R_{01} - R_{02} + R_{12})(R_{01} + R_{02} - R_{12}) \tag{4.6}$$

$$b = d_{01}^2(-R_{01}^2 + R_{02}^2 + R_{12}^2) + d_{02}^2(R_{01}^2 - R_{02}^2 + R_{12}^2) + d_{12}^2(R_{01}^2 + R_{02}^2 - R_{12}^2) \tag{4.7}$$

$$c = (d_{01} + d_{02} + d_{12})(-d_{01} + d_{02} + d_{12})(d_{01} - d_{02} + d_{12})(d_{01} + d_{02} - d_{12}) \tag{4.8}$$

$$\sigma = \left\{ \begin{array}{cc} 1 & if\ d_{01}^2 + d02^2 - d12^2 \leq s^2(R_{01}^2 + R02^2 - R12^2) \\ -1 & otherwise \end{array} \right\} \tag{4.9}$$

We then solve for the rotation matrix using the intermediate matrices A and B using Equation 4.10 and Equation 4.11:

$$A = \left\{ \begin{array}{ccc} (\vec{m}_1 - \vec{m}_0) & (\vec{m}_2 - \vec{m}_0) & ((\vec{m}_1 - \vec{m}_0) \times (\vec{m}_2 - \vec{m}_0)) \end{array} \right\} \tag{4.10}$$

$$B = \left\{ \begin{array}{ccc} x_{01} & x_{02} & y_{01} * H_2 - y_{02} * H_1 \\ y_{01} & y_{02} & -x_{01} * H_2 + x_{02} * H_1 \\ h_1 & h_2 & \frac{x_{01} * y_{02} - x_{02} * y_{01}}{s} \end{array} \right\} \tag{4.11}$$

where x_{01}, x_{02}, y_{01}, y_{02} are 2D coordinates relative to a coordinate system centered at the position of the left eye, $\vec{i}_0$:

$$x_{01} = i_{1_x} - i_{0_x} \tag{4.12}$$

$$x_{02} = i_{2_x} - i_{0_x} \tag{4.13}$$

$$y_{01} = i_{1_y} - i_{0_y} \tag{4.14}$$

$$y_{02} = i_{2_y} - i_{0_y} \tag{4.15}$$

The rotation matrix, R, can then be computed using Equation 4.16:

$$R = BA^{-1} \tag{4.16}$$

The translation vector, t, is computed simply by translating the centered coordinate system to the position of $\vec{i}_0$. The translation in the depth dimension is irrelevant and can be omitted since scaling is directly controlled by s (orthographic projection is not scaled by depth):

$$t = \{i_{0_x}\ i_{0_y}\ 0\}^T \tag{4.17}$$

Once the values of R and t have been determined, any 3D model point can be transformed and the points $(\vec{m}_0, \vec{m}_1, \vec{m}_2)$ will align with the image points $(\vec{i}_0, \vec{i}_1, \vec{i}_2)$. The transformation from model point to image point is thus:

$$\{i_x\ i_y\ i_z\}^T = R\{m_x\ m_y\ m_z\}^T + t \tag{4.18}$$

Of course, the i_z value is only a relative measurement of the depth of the model point. It is useful, however, for keeping track of the relative depth of the model point with respect to other model points.

Note that this is a direct solution of the WP3P problem save for one ambiguity: the $\pm$ value in the computation of (h_1, h_2) in Equation 4.4. This ambiguity allows two possible alignments of the model to the image points. The 3D face can either line up by facing towards or away from the viewer. Of course, we know that the face is projecting onto the image from behind the image plane and is facing the viewer (or "camera-man"). Thus we select either $+$ or $-$ in Equation 4.4 to assure that the 3D model is actually behind the image plane and is facing towards the camera. This ambiguity is resolved by computing the normal of the nose. In other words, a vector protruding from the nose on the model is introduced and undergoes the transformation in Equation 4.18. We begin by calculating Equation 4.4 with a '+'. We note the relative depth value of the vector i_z. If the vector is pointing away from the viewer (its tip is farther from the image plane than its base or, equivalently, has a larger i_z value) then the model is pointing away from the camera and we repeat the computation with a '-' in Equation 4.4.

The end result is a mapping from the 3D model to the image which lines up the eyes and the nose optimally.

3.2. Selecting the Optimal P3P for Deformation One more degree of freedom in our 7 parameter set, s_y has yet to be set. To determine the optimal value of this parameter, we shall now consider the position of the mouth on our model m_3 and on our image i_3. The other vertices (eyes and nose) have been exactly mapped to their destinations on the image via the WP3P computation. However, the same is not true for the mouth whose locus in the image has had no bearing so far on the calculation of our WP3P solution. In fact, we expect an error, E_{mouth} in the projected model mouth locus (mp_3) and the image's mouth locus (i_3). E_{mouth} is the distance between the model's mapped mouth and the image's mouth as shown in Equation 4.19:

$$E_{mouth} = \sqrt{(i_{3_x} - mp_{3_x})^2 + (i_{3_y} - mp_{3_x})^2} \tag{4.19}$$

To minimize E_{mouth}, we solve the WP3P for several iterations of s_y. We try the following values of s_y in attempting to find the best fit to the face: $s_y = 0.5 + 0.05i, i\epsilon[0, 14]$. The model and its four anchor points are scaled iteratively in the y dimension by each value of s_y, and the WP3P is solved. The WP3P solution which generates the smallest error,

(a) (b) (c)

FIGURE 4.11. Stretching the model in search of the best mouth fit.

E_{mouth}, is then found corresponding to the best s_y parameter. We have thus solved the problem of fitting to the mouth point by allowing our model to stretch along y to fit to "squashed" and elongated faces. Incidentally, since only a simple vertical stretching is involved in this minimization, it is evident that only the perpendicular distance of the mouth from the eyes will affect the deformation and not the horizontal displacement of the mouth. No matter where the mouth is horizontally, the model fitting will remain the same. Figure 4.11 depicts a few sample stretches to find the best model to fit the image. The texture on the face is a mesh representing the frontal face of the 3D model. The mesh undergoes multiple stretches and is aligned to the eyes and nose each time with the WP3P computation. The stretched mesh whose mouth point best aligns with the mouth point in the image is then used as the 3D structure which will be coated with the face in the intensity image.

Computing the above total 4-point fitting yields the rotation, deformation and translation to apply to the 3D model to fit it to the image. Our implementation is also highly computationally efficient, requiring less than 1 millisecond on an SGI Indy workstation.

3.3. Texture Mapping Once alignment of the 3D object's anchor points is complete, the eyes, nose and mouth of the 3D model are aligned with the eyes, nose and mouth of the face in the 2D image. The transformation, Equation 4.18, can then be computed. Each point in the 3D object undergoes this transformation to determine its locus relative to the 2D image. Essentially, the 3D object's range data points are rotated, translated and stretched to superimpose the 2D face's intensity points. It is then possible to compute the projection of the 3D range model onto the 2D plane and to associate each of its range data points with a corresponding 2D plane point. Each projected 3D range point can then be

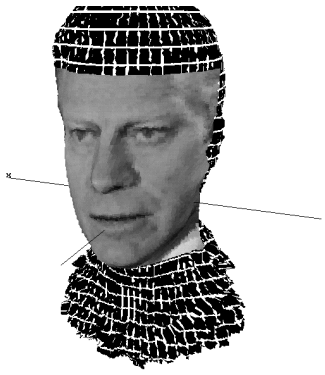


FIGURE 4.12. The 3D model coated with an intensity image's face.

"colorized" by the intensity value of the 2D plane point (or pixel) it intersects. The end result is a 3D model which is "coated" with the textural data of the face in the 2D image (see Figure 4.12).

3.4. Occlusion and Back Surfaces It is possible that certain points in the range data correspond to surfaces patches whose normals point away from the 2D image. For instance, the back of the head can be projected upon the 2D image yet it should not be coated with the intensity values it intersects since its surface points away from the image plane, not towards it. Furthermore, certain components of the range data may occlude others. For example, if a face is slightly rotated, the nose will overlap part of the cheek. Thus, it is necessary to avoid "colorizing" the range data points which are occluded since these can never project onto the image plane.

To prevent the erroneous "colorization" of 3D surface patches that are occluded or point away from the intensity image, it is necessary to compute the normals of each point in the 3D range data image and to perform exhaustive polygon occlusion checks or polygon rendering. However, real-time constraints prevent us from implementing such techniques in a fast face-recognition system. Instead, we shall using the symmetry of the human face to perform mirroring. This simple, efficient, though suboptimal technique is described below.

Furthermore, we choose to crop the 3D model so that only the front of the face will be utilized in texture mapping. The back of the head, the neck and the top of the head will not be useful for recognition, so there is no need to compute their projections onto the image or to worry about the validity of their colorization by tracking their surface normals.

3.5. Mirroring Along the Mid-Line In most scenarios, faces rotate along their vertical axis. For example, as people walk around in a room, they are unlikely to look

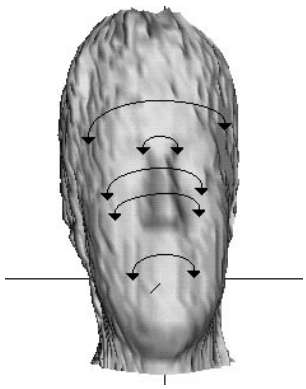


FIGURE 4.13. Mirroring intensity images from one side of the face to the other.

downwards at the floor or upwards at the ceiling. They walk erect and do not tilt their heads excessively. The faces do, however, rotate along the vertical axis as people look left or right and walk in different directions. Thus, the most likely source of self-occlusion will be due to the rotation around the vertical axis. This will cause parts of the cheek or a side of the face to be occluded by the nose and the other side of the face. Luckily, the human face is symmetric across its vertical axis and the occluded part of the face closely resembles the visible part under vertical-axis rotations. Consequently, if the face is rotated along its vertical axis and the left or right half are not visible, we can take advantage of this symmetry to 'guess' what the hidden side of the face looks like. In situations where one side is hidden from the camera, we implement mirroring by copying the pixel intensities from the closer side of the face (the one most visible to the observer) onto the hidden side of the face as shown in Figure 4.13. Thus, mirroring is only implemented if the individual turns to the left or right and one side of the face is occluding the other. The arrows in the figure show which pixel intensities are mapped to which destinations to generate the mirror-symmetry in the human face.

Mirroring is only used if the nose is not well centered between the eyes (indicating a strong vertical axis rotation). Figure 4.14 displays the exact range in which the mirroring process is triggered. If the nose falls within the central strip between the two eyes, we do not rely on mirroring and simply assume no significant self-occlusion has occurred.

4. Synthesizing 2D Images with a Texture Mapped 3D Model

Having fully coated the 3D model with the intensity image data, we can now rotate, translate and stretch the model arbitrarily to synthesize new views of the individual in question. By selecting a new set of parameters $(t_x, t_y, t_z, \theta_x, \theta_y, \theta_z, s_y)$ we can reposition our

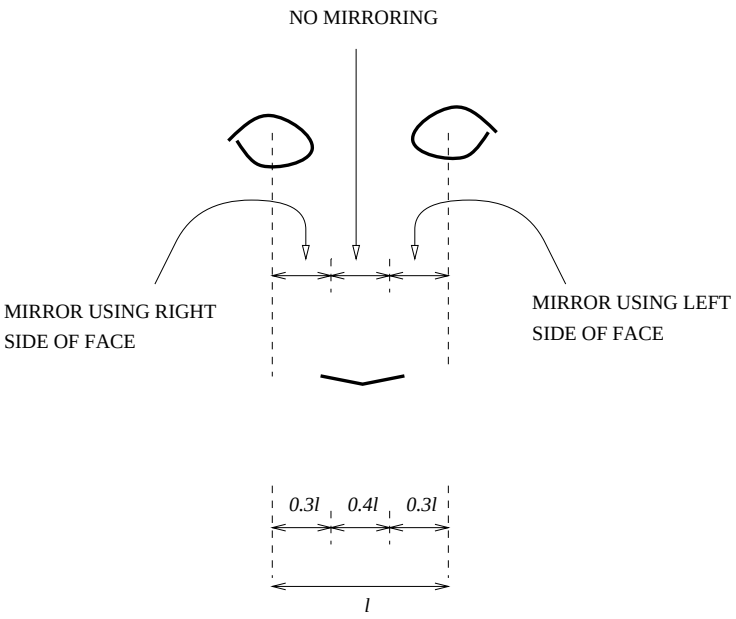


FIGURE 4.14. Range of nose positions where mirroring is necessary.

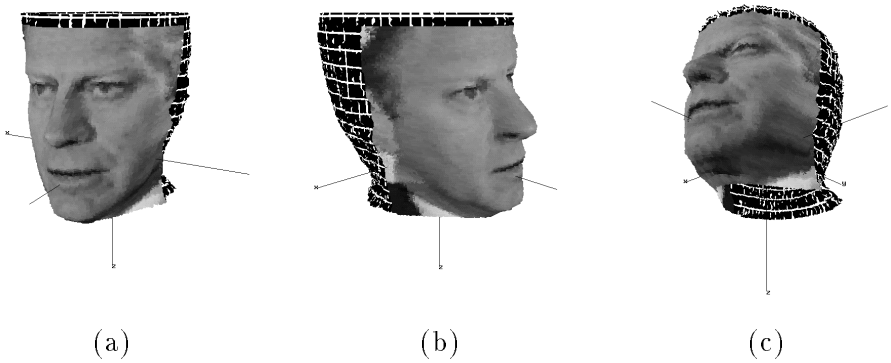


FIGURE 4.15. Some re-projections of the coated 3D model.

model and re-project a new 2D image of the original face. Sample re-projections of a 2D face obtained from a single view image are shown in Figure 4.15.

4.1. Synthesizing Segmented Mug-Shot Images for Recognition For recognition, we select a single pose or $(t_x, t_y, tz, \theta_x, \theta_y, \theta_z, s_y)$ value and use it exclusively to map the faces detected into a consistent view. The view we select is a frontal, mug-shot view which seems best for recognition purposes (although we could try synthesizing profile views, for instance and checking how recognition results vary with such data).

FIGURE 4.16. A synthesize mug-shot image of U.S. President Ford.

Since the faces will be projected into a standard, frontal view, using a pre-determined pose, a fixed cropping or segmentation of the 2D projection can be performed and reapplied to every new face undergoing the projection. Thus, we can automatically generate consistent frontal, segmented mug-shot views of any individual as long as we specify the four 2D anchor points corresponding to the eyes, the mouth and the nose. An example of a synthesized mug-shot face is shown in Figure 4.16.

5. Shading and Lighting Normalization

Now that pose variation has been considered, we turn our attention to the next most prominent source of variance in facial appearance that we wish to eliminate: lighting. Lighting can cause radical changes in the intensities of an image by varying from severe darkness to extreme brightness. Furthermore, lighting might be directional and cause one side of the face to be brighter than the other. Varying and nonuniform lighting need to be compensated for so that the large amount of variance they account for can be removed for recognition. Illumination corrections will be applied via histogram fitting.

5.1. Histogram Fitting Let $H(i)$ be the histogram function of an image, and let $G(i)$ be the desired histogram we wish to map to via a transfer function $f_{H\to G}(i)$. First, we compute a transfer function for both $H(i)$ and $G(i)$ that will map the histogram to a uniform distribution histogram, $U(i)$. The functions are $f_{H\to U}(i)$ and $f_{G\to U}(i)$ respectively. Equation 4.20 and Equation 4.21 depict the mapping to a uniform distribution which is also known as histogram equalization [**14**]:

$$f_{H\to U}(i) = \frac{\sum_{j=0}^{i} H(i)}{\sum_{j=0}^{n-1} H(i)} \tag{4.20}$$

$$f_{G \to U}(i) = \frac{\sum_{j=0}^{i} G(i)}{\sum_{j=0}^{n-1} G(i)} \tag{4.21}$$

where n is the number of discrete intensity levels, for 8-bit images, $n = 256$.

To find the mapping function, $f_{H \to G}(i)$, we invert the function $f_{G \to U}(i)$ to obtain $f_{U \to G}(i)$. Since the domain and the range of the functions of this form are identical, the inverse mapping is trivial and is found by cycling through all values of the function. However, due to the discrete nature of these functions, inverting can yield a function which is undefined for certain values. Thus, we use linear interpolation and assume smoothness to fill undefined points of the inverse function according to the values of well-defined points in the function. Thus, we generate a fully defined mapping $f_{U \to G}(i)$ which transforms a uniform histogram distribution to the distribution found in histogram $G(i)$. The mapping $f_{H \to G}(i)$ can then be defined as in Equation 4.22:

$$f_{H \to G}(i) = f_{U \to G}(f_{H \to U}(i)) \tag{4.22}$$

5.2. Selecting a Target Histogram It is possible to merely apply histogram equalization [**14**] to an image to correct lighting by imposing a uniform histogram distribution on the image. However, well-lit faces do not have a uniform histogram distribution and this process gives a surreal, unnatural illumination to the face. Recognition does not necessarily require "natural" illumination to be effective. It requires normalized illumination. However, a subtlety arises. The nature of the target histogram we wish to generate from our current histogram (be it Gaussian, uniform or some other distribution) will cause a "weighting" effect on the recognition. Regions which are extremely illuminated (very dark or bright) will contain more variance than normally lit regions and hence a "variance" type of analysis will tend to weight them more strongly during recognition. Since it is unknown which histogram configuration is optimal for recognition, we opt to normalize the histograms of the faces by giving give them an aesthetically appealing illumination.

Texts such as [**14**] encourage the normalization of a poorly illuminated image via histogram fitting to a similar, well illuminated image. For example, a poorly illuminated road map can be enhanced quite well by histogram fitting to a well-illuminated road map. Thus, a well illuminated average human face[2] is analyzed and its histogram is determined. This histogram will form our $G(i)$ destination histogram in the fitting process. We "histogram-fit"

[2]This "mean" face is an average of over 300 human faces under good illumination conditions.

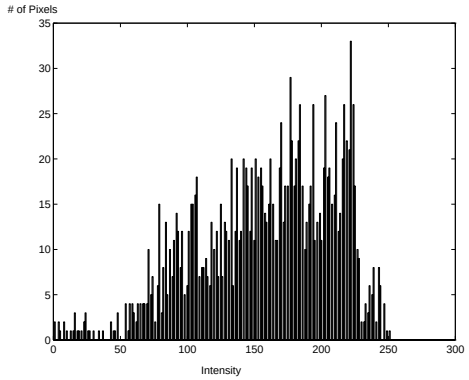


FIGURE 4.17. Histogram of the mean face.

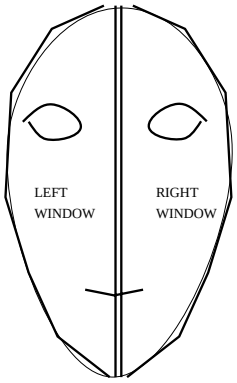


FIGURE 4.18. Windowing to split face histogram correction for each side of the face.

new faces to this average well-lit face histogram and noted its strong effectiveness visually. This technique was employed successfully by Phillips and Vardi [**33**]. However, they did not utilize a "mean" face but rather a random "sample" face to generate $G(i)$. The histogram of the mean face (our target histogram) is depicted in Figure 4.17.

5.3. Windowing Histogram Analysis To correct for non-uniform lighting or unequal illumination on the left and right sides of the face, we perform histogram fitting independently to both of these components of the image. Thus, a dark left side of the face will be brightened while a bright right side is simultaneously darkened. we perform a split or "windowed" histogram analysis on two components separately: the left side (l) and the right side (r) of the face. Two histograms, $H_l(i)$ and $H_r(i)$, generate the two mapping functions: $f_{H_l \to G}$ and $f_{H_r \to G}$ for the left window and the right window respectively. The windowing is shown in Figure 4.18.

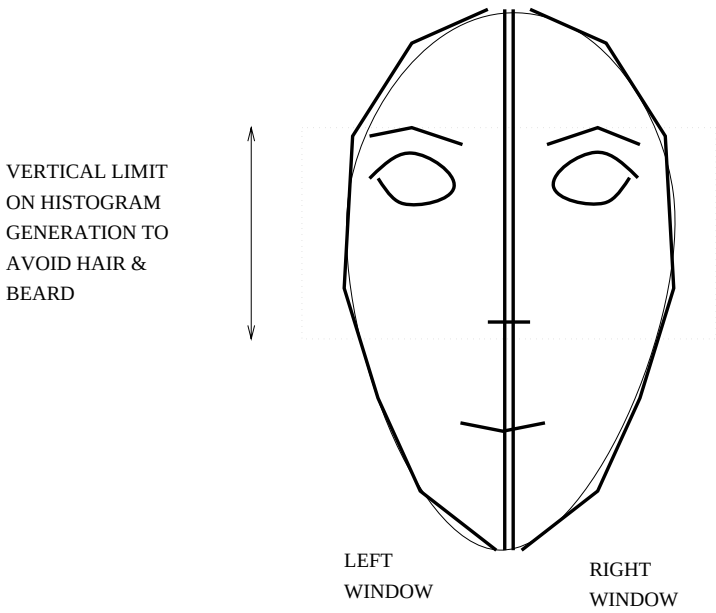


FIGURE 4.19. Limiting histogram generation to avoid hair and beards.

5.4. Beards and Hair Phillips and Vardi's technique [**33**] produced particularly poor results when the individual had a beard or other facial hair. The algorithm brightens the dark hair regions unnaturally. Consequently, the regions of the face that were likely to be obstructed with hair were omitted from the histogram generation process (both in the mean face and in new faces). Thus the inverse histogram mapping does not over-brighten the face to account for these extreme regions. The generation of $H_l(i)$,$H_r(i)$,$f_{H_l \to G}$ and $f_{H_r \to G}$ was limited vertically to cover only the eyebrows down to the nose as shown in Figure 4.19. However, we still apply the mappings in $f_{H_l \to G}$ and $f_{H_r \to G}$ to the whole face as in Figure 4.18. Thus, the illumination is corrected for quite nicely in the case of bearded individuals and other unusual cases.

5.5. Gradated Histogram Fitting Another artifact of the histogram fitting that was not addressed by Phillips and Vardi [**33**] was the sudden discontinuity in illumination as we switch from the left side of the face to the right side. This is induced by the switch from mapping function $f_{H_l \to G}$ to $f_{H_r \to G}$. We overcome this sudden transition by "averaging" the effects of $f_{H_l \to G}$ and $f_{H_r \to G}$ with a linear weighting that slowly favors one for the other as we move from the left side to the right side of the face. This "gradual" application of the mappings is implemented with the mapping function $f_{H_{total} \to G}$ shown in Equation 4.23, which uses a mixture of $f_{H_l \to G}$ and $f_{H_r \to G}$ and reduces discontinuities in histogram fitting:

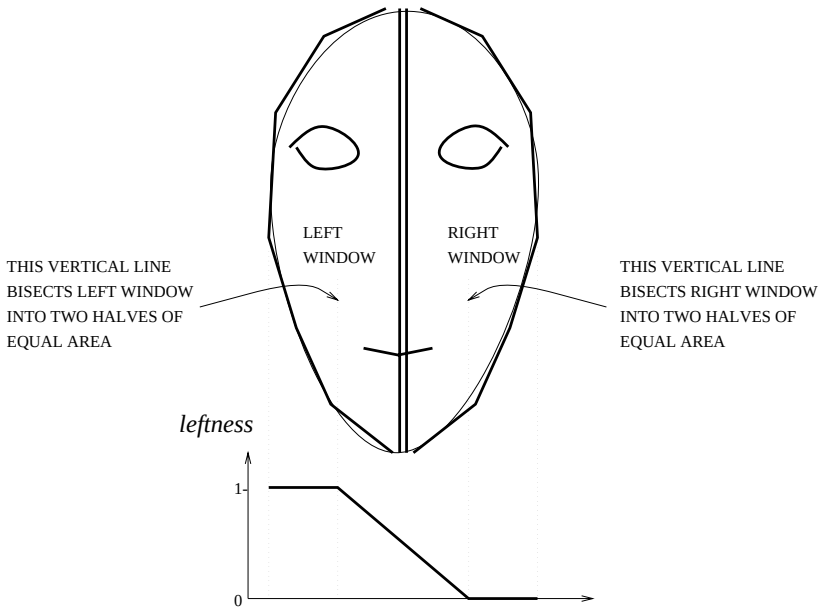


FIGURE 4.20. The mixture of histogram mappings from the left to the right side of the face.

$$f_{H_{total}\to G}(i) = leftness \times f_{H_l\to G}(i) + (1 - leftness) \times f_{H_r\to G}(i) \tag{4.23}$$

The parameter, $leftness$ is varied as we travel across the image from left to right as displayed in Figure 4.20.

6. Typical Normalization Results

In Figure 4.21, a gallery of fully normalized faces is presented with the initial source image. The results demonstrate the strength of the proposed normalization technique.

One curious oddity is the indifference of the algorithm to skin tone or albedo. Since the "mean" face is composed mostly of Caucasians, dark-skinned individuals lose their distinctive skin colour. Whether this necessarily has a negative impact on recognition algorithms is uncertain. It is evident, though, that the faces are still recognizable to a human observer after illumination normalization.

Overall, we note the regularity with which the faces appear in the normalized image gallery. The variance in appearance has been constrained to be a function of individual identity and expression alone, since lighting and pose have been filtered out of the image. This allows recognition and classification of identity to be performed on the basis of variance which we assume is dominated by the identity (not facial expression) of the individual in

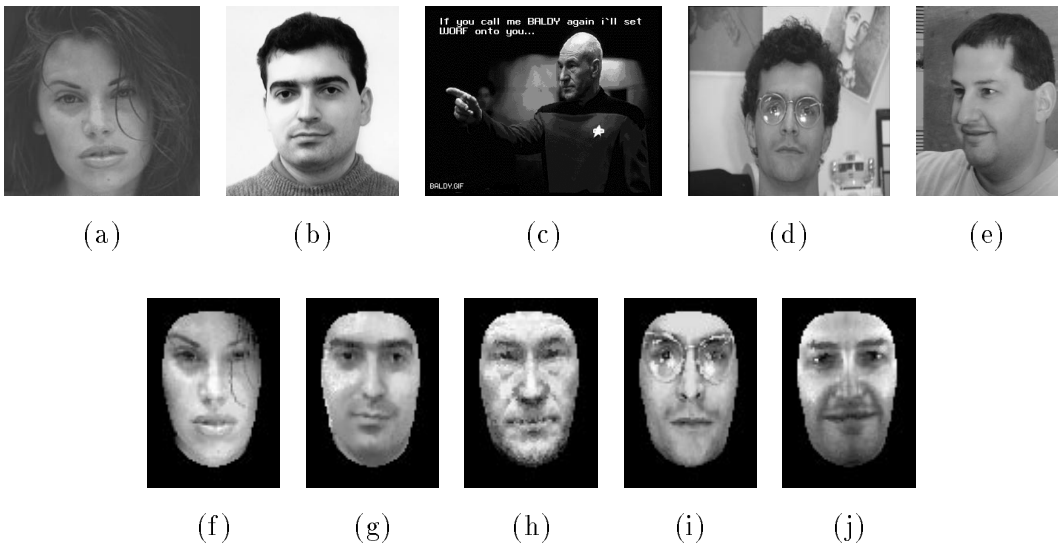


FIGURE 4.21. A gallery of face normalization results. (a) (b) (c) (d) (e) The original faces. (f) (g) (h) (i) (j) The corresponding synthesized mug-shots.

the image. This statement is true most of the time since people display a neutral expression in their daily activities (watching TV, walking, working, etc.)

On an SGI Indy workstation, the full normalization computation requires under 20 milliseconds for each 7000 pixel mug-shot shown above. The time required for the computation is almost proportional to the number of pixels in the output mug-shot being generated. For an 800 pixel mug-shot, the time required drops to below 2 milliseconds on the SGI Indy. Thus, the above normalization process is extremely efficient.

7. Karhunen-Loeve Decomposition for Statistical Recognition and Detection

At this stage, we have synthesized a normalized mug-shot for each individual in a scene. The large, nonlinear variance due to pose and illumination has been eliminated and it is now possible to classify individuals by simple linear techniques. We consider the use of Karhunen-Loeve Decomposition (KL), also known as Principal Component Analysis (PCA) on the intensity images [**17**]. This technique is traditionally used in statistical signal detection and estimation theory and has been adapted to compression and recognition.

Each intensity image is converted into a (raster) vector form. Note that this purely intensity-based coding of the image is not necessarily ideal for the application of KL-decomposition.[3] The normalized mug-shots in a database are represented by an ensemble

[3] We are currently investigating alternate encodings which depend simultaneously on intensity and (x,y) position.

FIGURE 4.22. The mean face.

of vectors, $\vec{v}_x$. Since the mug-shot images are 70×100 pixels, these vectors are 7000-tuples or 7000 element vectors. A total of 338 such mug-shot images are acquired and converted into vector form. The ensemble of vectors $\vec{v}_x$ is assumed to have a multi-variate Gaussian distribution since faces form a dense cluster in the large 7000-dimensional image space. We wish to produce an efficient decomposition which takes advantage of the redundancies or correlations in this data. Linear variance analysis makes it possible to generate a new basis which spans a lower dimensional subspace than the original 7000-dimensional space and simultaneously accounts optimally for the variance in the ensemble. PCA generates this small set of basis vectors forming this subspace whose linear combination gives the ideal approximation to the original vectors in the ensemble. Furthermore, the new basis exclusively spans intra-face and inter-face variations, permitting Euclidean distance measures in the sub-space to exclusively measure changes in identity and expression. Thus, we can use simple distance measurements in the subspace as a classifier for recognition.

We shall perform KL decomposition on an $n = 7000$ dimensional subspace with $N = 338$ training vectors. Some of these $\vec{v}_x$ vectors are shown in Figure 4.21. We begin by computing the mean, $\bar{v}$, of the 338 vectors using Equation 4.24 which generates Figure 4.22:

$$\bar{v} = \frac{1}{N} \sum_{x=0}^{N-1} \vec{v}_x \tag{4.24}$$

We then generate deviation vectors, $\vec{u}_x$, as in Equation 4.25 and arrange them in a dataset matrix, D, as in Equation 4.26. The dimensions of D are $n \times N$:

$$\vec{u}_x = \vec{v}_x - \bar{v} \tag{4.25}$$

$$D = \left\{ \begin{array}{cccccc} \vec{u}_0 & \vec{u}_1 & ... & \vec{u}_{N-2} & \vec{u}_{N-1} \end{array} \right\} \tag{4.26}$$

The covariance matrix C of our dataset is computed and its eigenvectors will form the orthonormal basis which optimally spans the subspace of the data (human faces). There exist two ways of computing and manipulating C which yield the same result yet with different efficiencies [**22**]. We begin by considering C generated using Equation 4.27.

$$C = D^T D \tag{4.27}$$

We can compute the n eigenvalues (λ_i) and eigenvectors ($\hat{e}_i$) of this $n \times n$ symmetric matrix. The eigenvectors and their corresponding eigenvalues are ranked such that $\lambda_i > \lambda_j$ for $i < j$. Mercer's theorem [**34**] can be used to obtain Equation 4.28:

$$D = \sum_{i=0}^{n-1} \lambda_i \hat{e}_i \hat{e}_i \tag{4.28}$$

Note that the magnitude of λ_i is equal to the variance in the dataset spanned by its corresponding eigenvector $\hat{e}_i$ (see Equation 4.29):

$$\lambda_i = \sigma_i^2 = \frac{1}{N} \sum_{x=0}^{N-1} \vec{u}_x \cdot \hat{e}_i \tag{4.29}$$

It then follows that any vector, $\vec{u}_x$, in the dataset, D, can be optimally approximated as in Equation 4.30. Thus, the n-dimensional face deviation vector can be re-defined as a linear combination of eigenvectors determined by M coefficients denoted by c_{x_i} (computed using Equation 4.31. M is the number of eigenvectors used to approximate the original vector. The larger the value of M, the more eigenvectors are used in the approximation and, consequently, the more accurate it becomes. M can be reduced allowing more efficient storage of each face. However, the quality of the approximation degrades gradually as fewer eigenvectors and coefficients are used in the linear combination [**22**]:

$$\vec{u}_x \approx \sum_{j=0}^{M-1} c_{x_j} \hat{e}_j \quad 0 \leq M \leq n \tag{4.30}$$

$$c_{x_j} = \vec{u}_x \cdot \hat{e}_j \tag{4.31}$$

Alternatively, we can obtain the same eigenvalues and eigenvectors from C', another symmetric covariance matrix, which is $N \times N$. C' is defined in Equation 4.32:

$$C' = DD^T \tag{4.32}$$

This equation yields N eigenvalues (λ'_i) and eigenvectors ($\hat{e}'_i$). These re also ranked such that $\lambda'_i > \lambda'_j$ for $i < j$. It is possible to then directly compute the first N eigenvalues and eigenvectors of C, as given by Equation 4.34:

$$\lambda_i = \lambda'_i \, \forall i \epsilon [0, N-1] \tag{4.33}$$

$$\hat{e}_i = \hat{e}'_i D^T \, \forall i \epsilon [0, N-1] \tag{4.34}$$

We shall now describe the covariance matrix we select as well as actual method used to extract the eigenvalues and eigenvectors.

7.1. Computing Eigenvalues and Eigenvectors The direct computation of eigenvalues and eigenvectors is a relatively straightforward operation for a symmetric matrix. We choose to solve the eigensystem using Jacobi transformations [**35**] (alternative techniques include Householder reductions followed by QR-QL and Hessenberg reductions followed by QR) [**35**]. Unfortunately, direct methods are usually computationally expensive and require $O(n^3)$ operations.Thus, solving the eigensystem of the form Equation 4.27 would require $O((n = 7000)^3)$ computations which would involve weeks of processing on an SGI Indy workstation. If we utilize the more efficient implementation found in Equation 4.32 we only need $O((N = 338)^3)$ computations to find the eigenvectors. This requires only several hours of processing on an SGI Indy workstation.

If we wish to have a dynamic face recognition system that continuously enlarges D in real-time as it views new individuals, it is possible to recompute the PCA using more efficient iterative gradient search methods, such as the one proposed by Roseborough and Murase [**41**]. However, we shall only compute the eigenvectors once for a dataset of $N = 338$ faces and then reuse these eigenvectors on new faces which were not part of the original dataset matrix D. This approach is based on the assumption that a large enough initial training sample of 300+ mug-shot faces would populate the "face-space" cluster within the larger "image-space" quite adequately (i.e., densely and extensively enough). Thus, new faces will simply fall within the predetermined face-space region and hence are well approximated by the span of the eigenvectors that were previously generated [**44**].

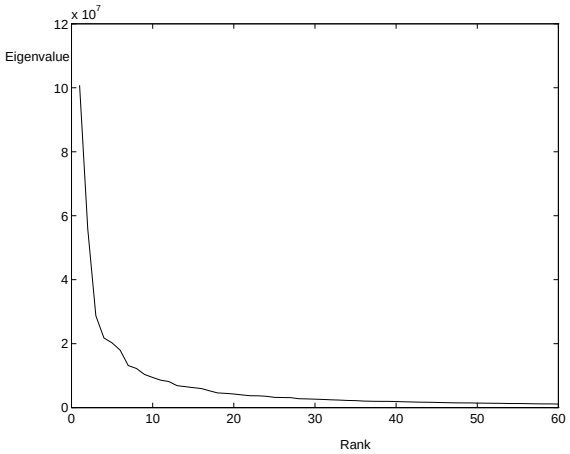


FIGURE 4.23. The ordered eigenvalues of the dataset.

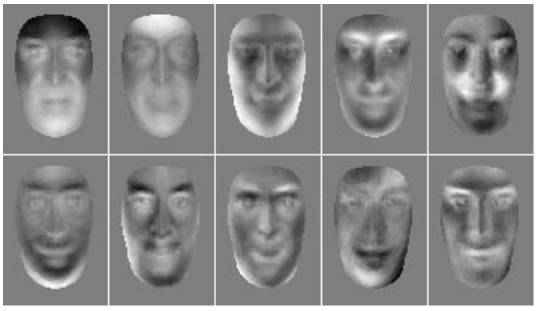
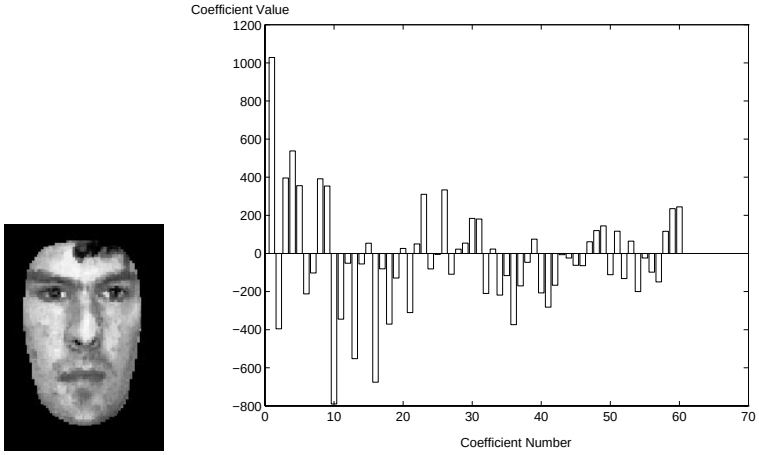


FIGURE 4.24. The ordered eigenvectors (or eigenfaces) of the dataset.

Figure 4.23 shows a plot of the magnitude of the eigenvalues versus their rank. The magnitude of an eigenvalue, λ_i, is equal to the variance in the data set that is spanned by its corresponding eigenvector, $\hat{e}_i$. Thus, it is obvious that higher-order eigenvectors account for less energy in the approximation of the data set since their eigenvalues have low magnitudes. We choose to truncate the set of eigenvectors to the first 60 vectors. This reduced set of eigenvectors accounts for enough face-variance to avoid excessively lossy compression. $M = 60$ will be used throughout the experiments [**22**]. The first 10 of our eigenvectors (also called eigenfaces since they are formed from face images) are shown in Figure 4.24.

7.2. Encoding Face Images with a Linear Combination Key We now have a way of converting each vector in the dataset into a set of 60 scalars which will form a 60-dimensional key coding of the image. The key is composed of the scalar coefficients that determine the linear combination of eigenvectors to approximate the original image vector.



(a) (b)

FIGURE 4.25. Converting a mug-shot into a key in KL-space.

The computation of the 60-coefficient key $(c_0, c_1, ..., c_{59})$ is performed using Equation 4.30. In Figure 4.25, we display the 60 scalar code representing one face from our training set. This encoding is performed for each face x $(x\epsilon[0, N))$ in the database giving us a total of N 60-element vectors of the form $(c_{x_0}, c_{x_1}, ..., c_{x_{59}})$.

With K-L decomposition, there is no correlation between the coefficients in the key (i.e., each dimension in the 60 dimensional space populated by face-points is fully uncorrelated)[**17**]. Consequently, the dataset appears as a multivariate random Gaussian distribution. The corresponding 60 dimensional probability density function is approximated in the L_2 sense by Equation 4.35 [**17**]:

$$p(c_0, c_1, c_2, ..., c_{M-1}) = \prod_{k=0}^{M-1} \left\{ \frac{1}{\sqrt{2\pi\lambda_k}} \exp(-\frac{c_k^2}{2\lambda_k}) \right\} \tag{4.35}$$

The envelope of this Gaussian distribution is a hyperellipsoid [**17**] whose axis along each dimension is proportional to the eigenvalue of the dimension. In other words, the hyperellipsoid is "thin" in the higher-order dimensions and relatively wide in the lower-order ones. Although it is impossible to visualize the distribution in 60 dimensions, an idea of this arrangement can be seen in Figure 4.26 which shows the distribution of the data set along the 3 first-order coefficients (associated with the 3 first-order eigenvectors).

7.3. Decoding a Key into an Image The encoded faces can be synthesized (or mapped back into image space) from their 60-coefficient key (c_j) by summing the mean face

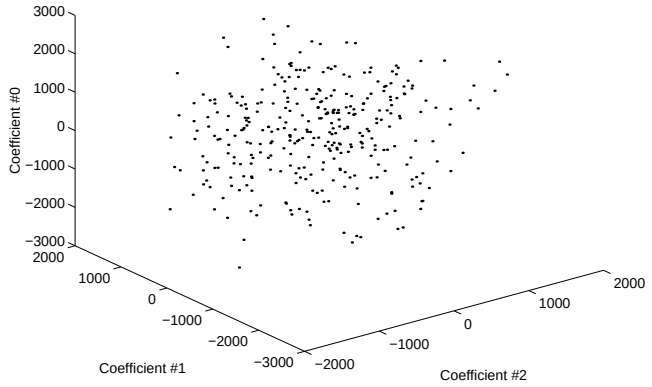


FIGURE 4.26. The distribution of the dataset in the first three coefficient dimensions.

(a) (b) (c) (d) (e)

FIGURE 4.27. Re-approximating the gallery's mug-shot images after KL-encoding

and the 60 weighted eigenvectors linearly. This forms an approximation n-tuple, $\vec{w}_x$, of each original face, $\vec{v}_x$, as given by Equation 4.36. The remapped approximations for Figure 4.21 are shown in Figure 4.27. Note the low degree of "lossiness" of the compression despite the large compression ratio of 7000 pixels to 60 coefficients (i.e., 117:1 compression).

$$\vec{w}_x = \bar{v} + \sum_{i=0}^{M-1} c_{x_i} \hat{e}_i \tag{4.36}$$

We can quantify the lossiness in the KL-based decomposition by measuring the variance between the original vector $\vec{v}_x$ and the optimal linear approximation vector $\vec{w}_x$. This residual variance, $residue_x$ accounts for the total variance in our vector that cannot be spanned by the eigenvectors. The total residual variance in the dataset that cannot be spanned by the eigenvectors is denoted by $\lambda_{res} = \sigma^2_{res}$ and computed using Equation 4.38:

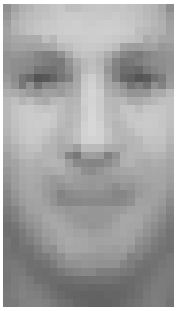


FIGURE 4.28. The mean face generated with smaller mug-shots.

$$residue_x = \|\vec{v}_x - \vec{w}_x\| \tag{4.37}$$

$$\lambda_{res} = \sigma_{res}^2 = \sum_{x=0}^{N-1} \|\vec{v}_x - \vec{w}_x\| \tag{4.38}$$

We have thus described a method for converting a mug-shot image into a 60-scalar code which describes it in the KL domain. Furthermore, we have shown how this transformation can be inverted so that the 60-element code can be used to regenerate an approximation to the original image. Thus, the KL transform and "inverse" KL transform are added to our palette of tools.

7.4. Varying n for Speed or Resolution Note that the original vectors and the eigenvectors are all n-tuples with $n = 7000$. Therefore, the computation of the 60-dimensional key $(c_0, c_1, ..., c_{59})$ given by Equation 4.30 requires convolution of the input image (an arbitrary $\vec{v}$ n-tuple) with 60 eigenvectors (the $\hat{e}_i$ n-tuples). Thus, 60 convolutions of 7000 pixel masks must be performed each time to convert an image into its key. To reduce this computation, we have also generated scaled versions of the input vectors and eigenvectors. These images are 22×39 pixels and require $n = 858$ element vectors. However, the computation of $(c_0, c_1, ..., c_{59})$ for the smaller vectors requires roughly 10% of the original processing time. The mean face for the smaller data set is shown in Figure 4.28. Thus, these versions of the KL decomposition can be useful when time is more critical than resolution or precision. However, the quality of such low resolution images makes them unreliable for face recognition purposes. These should only be used to perform coarse face detection. We shall now describe a technique for detecting a face using an image's 60-scalar KL code.

7.5. Using KL as a Faceness Detector Having processed an ensemble of training images with Karhunen-Loeve decomposition and having witnessed its face-compression abilities, we now turn our attention to its usefulness in signal detection. The KL decomposition has mapped each individual face $\vec{v}_x$ into a 60-dimensional key describing the linear combination of an orthonormal basis with a residual error, $residue_x$. We wish to have a scalar measure of how face-like a new image vector is by comparing it to the collection of faces we have already considered. Each face in our database maps to a point in KL-space and these points form a roughly Gaussian cluster. A new image will also map into a point in this space. By observing how close the point is to the cluster formed by D, we can measure how "face-like" it is. Thus, we can detect faces in a scene with this measure and reject non-face images.

Before we proceed, we shall add a dimension to the 60-dimensional space we have formed from our key. The value of *residue* indicates how well the KL decomposition approximates our image with its eigenvectors. Thus, a human face will be well approximated since the eigenvectors we formed from the database are optimal for such a task. Consequently, human faces should yield low *residue* values. A non-face will generate a high *residue* value since it is not in the span of the eigenfaces and can not be expressed as a linear combination of the face-like eigenvectors. As was the case for each value in the 60-dimensional key, the value of *residue* is expected to have a Gaussian distribution over the vectors in the dataset. The σ value of this distribution is σ_{res}.

Figure 4.29 depicts the distribution of the first two coefficients (c_0, c_1) of the key on the (x, y) plane and in the *residue* dimension on the z-axis (or vertical). Note the multivariate distribution is now 61-dimensional with the addition of the *residue* dimension. A new image vector that is presented to the KL-decomposition algorithm will map into a point in this 61-dimensional cloud. The closer it is to the 61-dimensional cloud of previously encountered points, the more face-like it appears. The probability of membership within the class of faces is defined via a probability density function (p.d.f or "pdf") similar to the one in Equation 4.35. We now discuss the pdf that will be used in our "faceness" equation.

The pdf we need must have a centroid at (0,0,0,...0) for all dimensions. Even though the mean *residue* value (which is always positive) in the database is not 0, we shall consider it to be 0. This is because the true centroid or mean of the data-set is the mean face (which we computed in Equation 4.24). The locus of the mean face in the 61-dimensional space is the 0-vector and so the centroid of the Gaussian distribution is the 0-vector (0,0,0...).

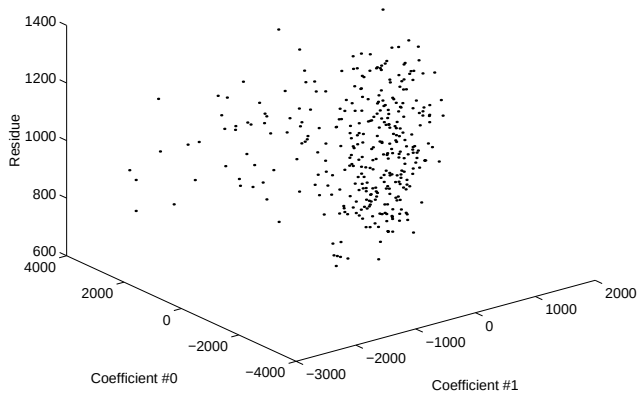


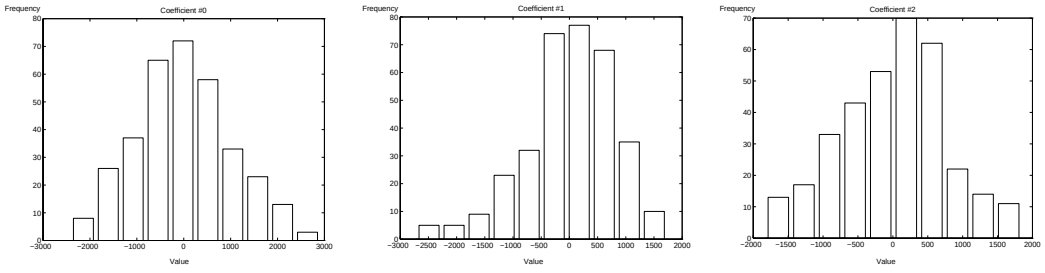
FIGURE 4.29. The distribution of first two coefficients and the residue (on the vertical axis) for the dataset.

Now, we analyse the 61-dimensional cloud of points we are trying to model. We wish to determine which Gaussian pdf will suit our needs. The value of this pdf will measure the "faceness" of an image by how close it is to this cloud of points determined by our original dataset, D.

We note, as expected [**17**], that the distribution of the points in the cloud is a multivariate Gaussian with a different σ value in each of its 61 dimensions. However, an important observation is that the distribution has its worst-case outliers at different extrema or distances along each dimension. In other words, the worst-case or L_∞ distance along each dimension is not constant. Even more importantly, it is not proportional to the σ value along the corresponding dimension.

Observe the data-distribution of c_1,c_2 and c_3 in Figure 4.30. The face points in the histograms seem to have a Gaussian distribution in each dimension. Note the presence of extreme outliers on either side of the plots. These are still valid faces despite their location to the far left and the far right of the bell-curve. If we approximate the distribution by a tightly-fitting Gaussian function, those outliers will be given an extremely low likelihood value. However, they are true faces and should therefore register a strong "faceness" probability. Thus, an equation similar to Equation 4.35 will not suit us as a face-detector since it will reject outliers.

Traditionally, statistically approaches to distribution modelling attempt to fit a Gaussian to the distribution in an L_2 sense [**17**]. However, we choose to consider an envelope that wraps around the whole cloud of face-points (enclosing all outliers as well). The shape of this envelope is hyper-ellipsoidal. The envelope is not defined by the variance in the data



(a) (b) (c)

FIGURE 4.30. The distribution of the dataset in the first 3 coefficients. (a) Coefficient 0. (b) Coefficient 1. (c) Coefficient 2.

set or by fitting to the points in the data set. Instead, it is shaped to contain all the points in the dataset and thus is defined by the boundary of the cloud or the most extreme points in the cloud. These are all *valid* faces and therefore a detector should not discard them, regardless of their distance to the cloud in an L_2 sense.

Therefore, the sigmas in the Gaussian pdf that we use for detection should not be related to the variance of the data in each dimension. Using the variance, as we have shown, will cause misdetection of the odd outliers. However, these outliers which lie quite far from the cluster are still valid faces. Therefore, we shall select the σ values for our multivariate Gaussian to be equal to the distance of the worst outlier in each dimension ($outlier_i$ as given by Equation 4.39 and $outlier_{residue}$ is given by Equation 4.40). The consequent pdf is computed using Equation 4.41:

$$outlier_i = \Big(\sum_{j=0}^{j<N-1} (c_{j_i|})^{\infty} \Big)^{-\infty} \tag{4.39}$$

$$outlier_{residue} = \Big(\sum_{j=0}^{j<N-1} (residue_j)^{\infty} \Big)^{-\infty} \tag{4.40}$$

(4.41)

$$faceness(c_0, ..., c_{M-1}, residue) = \prod_{k=0}^{M-1} \left\{ \exp\left(-\frac{c_k^2}{2(outlier_k)^2}\right) \right\} \exp\left(-\frac{residue^2}{2(outlier_{residue})^2}\right)$$

Alternatively, we can write the faceness value as a distance from the cloud. This distance is obtained by computing the logarithm of Equation 4.41. Thus, our distance from

facespace measure (DFFS) can be defined by Equation 4.42 (note that k is an arbitrary constant used to scale the output for display purposes):

$$DFFS(c_0, c_1, ..., c_{M-1}, residue) = k \times (\sum_{k=0}^{M-1} \left\{ \frac{c_k^2}{outlier_k^2} \right\} + \frac{residue^2}{outlier_{residue}^2}) \tag{4.42}$$

This DFFS measure is similar to Turk and Pentland's [44] approach to detection via a distance-to-facespace technique [44]. However, their technique merely utilizes the *residue* value in the computation and assumes all c_k are 0. Consequently, this form of distance measure assumes that faces form a hyperplane in image-space. We can see that this is not the case since the cluster of face-points we have generated appears to form a hyper-ellipsoidal cloud shape. Additionally, in Turk and Pentland's technique, an image which happens to be spanned nicely by eigenfaces will be classified as a face. Unfortunately, eigenfaces, (especially higher-order ones) can be linearly combined to form images which do not resemble faces at all. Hence, merely using the *residue* as a faceness measure is not reasonable.

Figure 4.31 shows some sample faces and non-faces with their corresponding "DFFS" value. The DFFS can be used for face-detection since it yields low values for faces and high values for non-faces. The DFFS value is not exactly zero for true faces since only the mean face is located precisely in the center of the cloud representing the distribution. All other faces have a distance from the center of the cloud and, consequently, have a non-zero DFFS.

7.6. Nose Localization Revisited At this point, we recall the problem of nose-localization that was discussed in Chapter 3. The solution for the exact horizontal position of the nose was deferred because it was too difficult to obtain using the direct image processing techniques introduced in Chapters 2 and 3. However, Chapter 4 has introduced a reliable "DFFS" measure based on Karhunen-Loeve statistical signal detection. We can use this measure to to assist us in locating the nose.

Recall that at the end of Chapter 3, we detected the eyes and the mouth but only had a line representing the nose. This situation is represented in Figure 4.32. This image is similar to the final result of the detection performed in Chapter 3. The horizontal position of the nose with respect to the eyes was uncertain and could be anywhere on the solid white line.

The nose localization problem is solved using an algorithm based on the development in Chapter 4. Along the horizontal line across the nose, a set of equally spaced points are

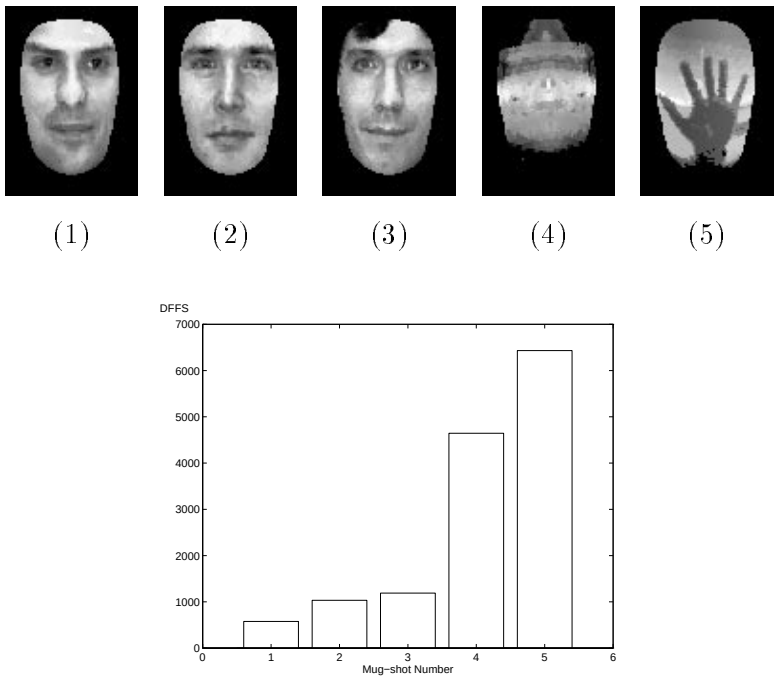
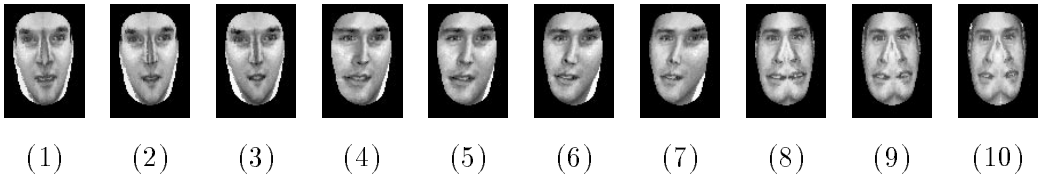


FIGURE 4.31. Mug-shots containing true faces and non-faces and a graph of their distance to face space (DFFS) values.

FIGURE 4.32. The nose localization problem.

picked. Each point is then used as the anchor point for the nose in the 3D normalization process. Then, we obtain a mug-shot of the face from the 4 detected anchor points (eyes, mouth and the nose being tested). This image is transformed into a key and a *residue* value via the KL-decomposition. Using Equation 4.42, the distance to face-space of the image is evaluated. This process is repeated for several trial anchor points along the line crossing the nose and generates several mug-shot images as in Figure 4.33. Also shown is the DFFS value for each mug-shot image. Note how misdetected noses generate a mug-shot image



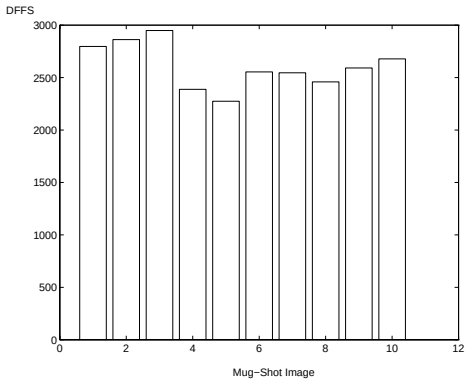


FIGURE 4.33. Distance to face space values for mug-shot images from different nose-point trials (from left to right across the nose-line)

FIGURE 4.34. The final localization.

that is very far from face-space and how the minimum "DFFS" value is registered when the nose anchor point is properly localized on the nose in the image.

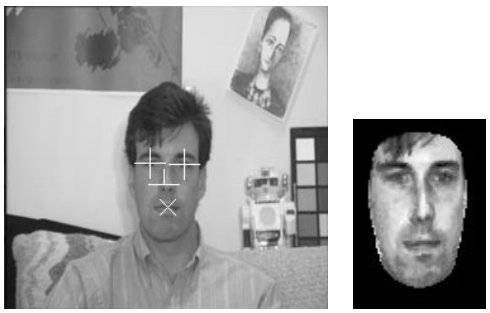
As we attempt different possible nose points, we are generating a trajectory in the 61-dimensional space which crosses through our cluster of database faces. The point at which the trajectory in the 61-dimensional space is "closest" to face-space or has a maximum "faceness" value corresponds to the a point on the nose-line. Using the sampled DFFS measures for each trial nose-line point, we can select the best nose point as the one that minimizes DFFS. Thus, the problem of finding the nose is overcome by testing each possible nose position on the line. The final, fully localized face is shown in Figure 4.34.

Thus, the face detection algorithm ends up with 4 anchor points corresponding to the eyes, the nose tip and the mouth as well as a "DFFS" measure. It also has a 60-element key to represent it as well as a *residue* value (generate from the KL transform).

7.7. Discarding Non-Faces before the Recognition Stage At the recognition stage, we have a "DFFS" value, the 60-element key, and the residue value of a fully normalized mug-shot image. First, if the "DFFS" value is above a threshold (specified by the user), the mug-shot image does not contain a true face. In other words, it is quite far from the cloud of faces in our training set and should probably be discarded from the recognition algorithm. Recall the face localization procedure presented in Chapter 3. It is possible that the face localization incorrectly converges to an image that is not a face. The detection process described in Chapter 3 only dealt with blobs and limbs. This simplistic description of image data allows us to detect faces in many poses, although this flexibility might also permit a non-face to be falsely detected by the algorithm. A soccer-ball, for example, might look like a face with two eyes, a mouth and a nose (from a simple blob and limb description of the image). Since the face-detection scheme described on Chapter 3 is based exclusively on a simple blob and limb description of the image, the recognition stage might obtain a mug-shot of a soccer ball or other non-face object. However, in the recognition stage, we utilize the statistically based "distance to face-space" measurement to ultimately reject non-face mug-shots from the localization procedure. Thus, the mug-shots generated by the face-localization process can be filtered with a threshold on their "DFFS" value before recognition is attempted. Equation 4.43 illustrates the use of a threshold on the "DFFS" value to prevent recognition attempts on non-faces. Typically, the value used for $DFFS_{threshold}$ is roughly 3000.

$$DFFS < DFFS_{threshold} \tag{4.43}$$

7.8. Face Recognition with Distance Measures Once we have a fully normalized mug-shot of a true, fully localized face, we can use it to probe for a match in the database of previously detected individuals. Each mug-shot in the database is stored as a 60-dimensional key (after KL). The most similar mug-shot in the database will be used to identify the probe mug-shot. We compute the Euclidean distance between the test image's 60-dimensional key and all the 60-dimensional keys in the database [**44**] using Equation 4.44. The key which is geometrically closest to our probe's key will yield the lowest distance (d_{min} as in Equation 4.45). This is the best match for the face, as given by Equation 4.46. The equation



(a) (b)

FIGURE 4.35. A probe mug-shot face generated from an automatically localized face in an arbitrary input image. (a) Arbitrary image. (b) Corresponding mug-shot for database query.

assumes that there are P entries in the database and that the xth face in the database is called $face_x$, where $0 \leq x < P$. The key of the xth face is c_{x_i} where $0 \leq i < 60$; in other words, i is the dimension of the coefficient in the key:

$$d(probe, face_x) = \sqrt{\sum_{i=0}^{59}(c_{x_i} - c_{probe_i})^2} \tag{4.44}$$

$$d_{min} = min_{x=0}^{x<P} d(probe, face_x) \tag{4.45}$$

$$match = z \ such\ that\ d(probe, face_z) = d_{min} \tag{4.46}$$

In this way, we obtain the best match in the database, face z, as the output of the recognition stage.

We illustrate the matching or recognition process for the test face in Figure 4.35. In Figure 4.36(a) and Figure 4.36(b) the test image and the closest five matches in the database are presented with their Euclidean distance $d(probe, face_z)$ from the test face. These are database matches ordered from nearest to farthest (left to right). Additionally, we present the original test image and the most similar original database images in Figure 4.36(c). The original image is shown with the features localized in the top left and the database images around it are ordered from nearest to farthest (left to right and top to bottom).

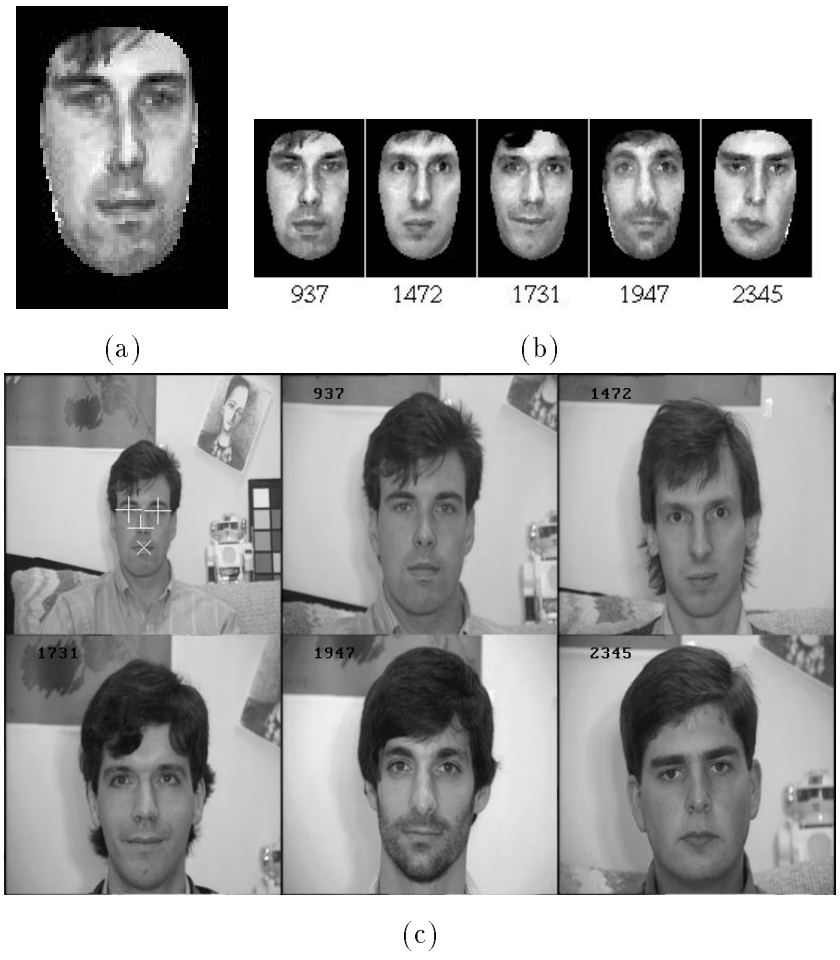


FIGURE 4.36. Database Matching. (a) The test mug-shot. (b) The closest five matches in database from left to right and their Euclidean distances from the probe mug-shot. (c) The original test image (top left) and the closest five matches in the database and their Euclidean distance from the test image.

8. Synopsis

We have thus presented a method for normalizing images for 3D pose changes and illumination. This technique is employed with the Karhunen Loeve decomposition to statistically determine how face-like an image is. This helps guide a nose detection search which minimizes the "DFFS" value. Finally, the fully localized face is normalized and analyzed using the KL to match it to an image in the database. This final stage completes the algorithm proposed in Chapter 3 and generates recognition results.

CHAPTER 5

Implementation and Testing

Having extensively discussed each module in our face detection, normalization and recognition system, we shall now describe an implementation of the complete system. We shall also briefly discuss the software design and the constraints on performance and efficiency. The system's user-friendly interface is then described. It facilitates the control of the algorithm and displays the localization and recognition results visually. To test the feature detection capabilities of the system, the localization results are verified for a few arbitrary input images. A large, standard database of faces is then used to test the recognition rates of the system. We follow the recognition and localization tests with a sensitivity analysis that provides us with some insight on the inter-dependence of these two components (recognition and localization) in the overall algorithm.

1. Implementation

The algorithm has a modular implementation and was written in C, C++ and assembly code on SGI Indy and Pentium-based systems. We shall now consider the system as a whole. The overall structure of the algorithm in terms of the separate modules is outlined. Then, we discuss the computational efficiency and performance of the algorithm. Subsequently, a description of its user-interface is presented.

1.1. System Overview A block diagram description of the overall algorithm is depicted in Figure 5.1. The diagram shows the flow through the individual modules used by the system. The algorithm's structure is complex and somewhat fragmented since it is designed to deal with multiple scales and multiple possible face detections in an image.

We begin at a large scale with the image reduced so that operators acting upon it detect the largest objects in the scene. The face blob localization module finds all blobs in the image at the current scale and then transmits the coordinates of the strongest blob to

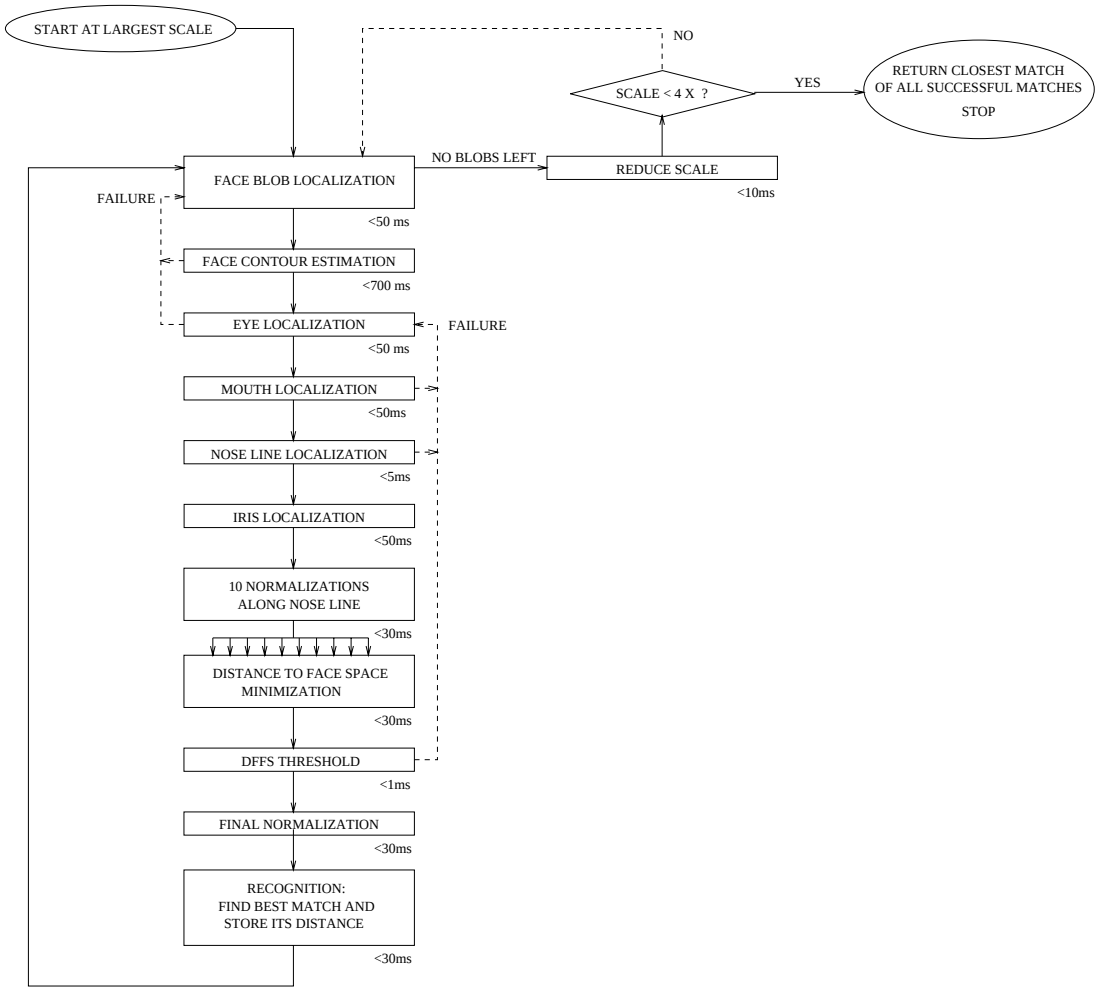


FIGURE 5.1. Block diagram description of the overall algorithm

the facial contour estimation module. If no face-like contour is present around the blob, the face-contour estimation module sends a failure signal to the blob detector which, in turn, provides it with another blob to process. If, however, a facial contour exists, the algorithm proceeds to the eye localization module.

The eye localization stage finds all eye-like blobs in the facial contour's eye band and sends the dominant pair to the mouth localization stage. We then find the nose line and the iris. To find the exact position of the nose, we sample the nose line ten times and generate 10 normalized mug-shots from 10 nose anchor points on the nose line. The DFFS is computed for each and the nose anchor point which yields the minimal DFFS is output. The nose is then fully localized and we compute a final normalization to obtain a high resolution mug-shot image (a probe). This probe image is recognized in the recognition

module which finds its closest match in the database. The match and its distance from the probe image are then stored and we loop back to the face blob localization stage to check out the remaining face blobs in the image.

If none of the 10 normalizations along the face-line generated an adequate DFFS, the DFFS threshold stage would generate a failure signal and inform the eye localization stage to transmit another pair of eye blobs. Similarly, the lack of a valid mouth or nose-line could also generate a failure from the corresponding module. This would also issue a request for another pair of eyes from the eye localization module.

If the eye localization module has transmitted all the possible eye blobs and none have successfully passed through all the subsequent stages, it generates a failure signal itself. This informs the face blob localization stage to transmit another blob to the face contour stage, forcing the search to process another face blob elsewhere in the image.

Once all blobs detected by the blob localization module have been investigated, it generates a signal to the 'Reduce Scale' module. This generates an image at a new scale at which face blob localization is re-executed. Thus, we have a new set of smaller face blobs to investigate. This process continues, allowing the algorithm to search each scale progressively (from large to small scales) for face blobs. Once the algorithm has reached the smallest allowable scale ($4\times$), all face blobs have been processed. The system then stops searching and generates its recognition output.

Throughout the search, the algorithm will have localized several face-like objects which were used as probe images to query its database of faces. Each probe image is matched to a database member and the distance from the probe image to the database member is stored. The probe image with the lowest distance to a database member is the one that most accurately resembles a member of our database. Thus, we return this face as the recognition result.

1.2. Performance and Code Efficiency In an effort to maintain computational efficiency and to allow the eventual adaptation of the algorithm to face tracking applications, intense optimization of the code has been performed. Although further development is in progress, the algorithm is currently fast and compact enough to run interactively on most generic platforms.

Note, first, the sequential hierarchical search which proceeds from large scales to small scales. This allows a rapid convergence if the face is dominant in the image. Furthermore, the algorithm does not always flow through the complete loop. It stops as soon as one of the

modules reports a failure and loops back to an earlier stage. For example, we do not search for a mouth if no eyes are found. In this case, no time is wasted in the mouth module.

Additionally, we utilized special programming techniques to reduce the run-time. For instance, wave propagation is used to generate the symmetry maps. This provides a computational efficiency that makes the symmetry operator a practical tool. The 3D normalization algorithm is also extremely efficient and uses look-up tables and minimal calculations for increased speed. The 10 normalizations and DFFS calculations required for nose-localization also utilize small mug-shot images and eigenfaces to increase efficiency.

Note the upper bound on the run-times for each module in Figure 5.1. The execution times are measure on an SGI Indy machine which has a rating equivalent to that of a 1996 home personal computer. The efficiency of the code allows a face to be found in an image in under 1 second if it is the dominant structure. However, we loop through all objects in the scene in an attempt to find all possible faces. Thus, the algorithm's loop is traversed multiple times even though a face could have been detected in an earlier iteration of the loop.

1.3. Graphical User Interface The graphical user interface links the modules in an intuitive, easy-to-use window environment. The interface is displayed in Figure 5.2 and its output window is shown in Figure 5.3.

Using the "Load" button, the user selects an arbitrary test image from disk and the system then loads and processes it. The test image is then shown in the top left of the video output window (see Figure 5.3) with the locations of the eyes, the nose and the mouth labelled. The identity of the face is then recognized and the best database matches are shown ordered from left to right and top to bottom. Additionally, the distance from each match to the input face is shown superimposed on the matches. The image with the minimum distance to the probe, (the one to the immediate right of the probe image) is the best match found in the system's database.

The user can also choose to run the algorithm on any one of the images shown in Figure 5.3 by selecting the number corresponding to it ("0", "1", ... "8" which identify the images from left to right and top to bottom in the figure). Also, the user can select "Randomize" to replace the images in Figure 5.3 with a random sample of the images in the database to quickly view some faces in the database.

By clicking on the "Recognize Video" button, the user can utilize the video camera and frame grabber to capture data directly from the current scene, as in Figure 5.4. Here, the user's own face has been localized and the closest match in the database is presented. In

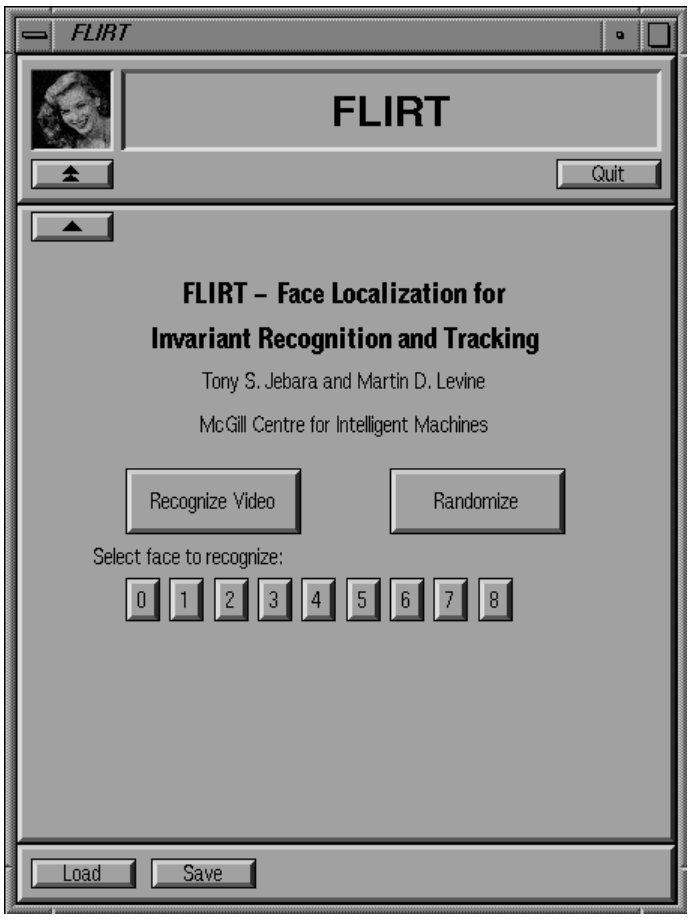


FIGURE 5.2. The user interface. The video output in Figure 5.3 is also part of the user-interface.

this case, the user is not in the database so the closest match is the member of the database that most closely resembles the user. Finally, the user can add a new face to the database by selecting the "Save" button. The user is then prompted to enter a name identifying the individual in the test image. Thus the database can be easily manipulated by the user who can train the system and customize it for the set of individuals to be recognized.

2. Testing

The algorithm is then put through three tests to evaluate its effectiveness. The localization test demonstrates the algorithms ability to detect faces and localize facial features in several, arbitrary test images. A recognition test is then performed on a standard database of faces to determine the recognition rates of the algorithm. In addition, we present a sensitivity test which relates recognition effectiveness to localization accuracy.

FIGURE 5.3. The video output for a sample image

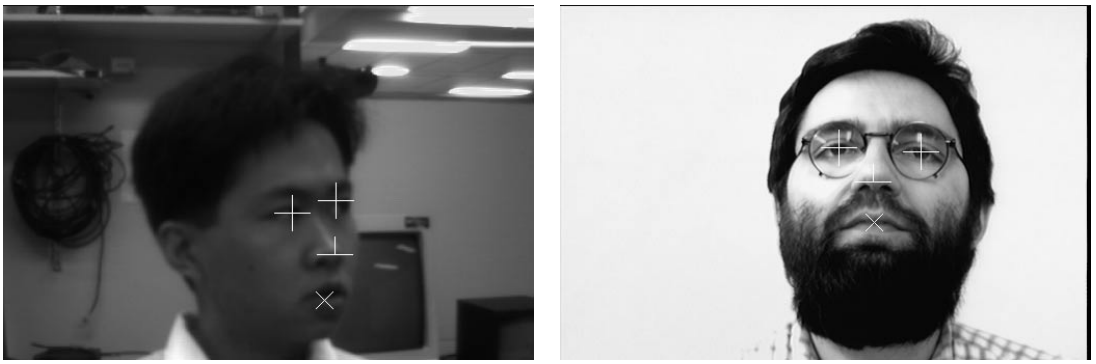
2.1. Localization Test The algorithm is executed on a few sample images that span the range of imaging situations the system is expected to deal with. The recognition module results of the algorithm are omitted for these calculations since we are only interested in the localization effectiveness. Figures 5.5, 5.6, 5.7, and 5.8 contain sample images with the detected feature points marked in white.

In Figure 5.5(a) the face is localized even though it is in a non-frontal pose and despite the thin veil that covers it. Figure 5.5(b) shows the localization of a face despite numerous other faces in the image. Figure 5.6(a) shows the localization of a blurry face with a large out-of-plane rotation and significant background data. Figure 5.6(b) shows the successful localization of a face with a beard and glasses which do not pose a problem for the algorithm. Additionally, the face is leaning backwards slightly. Figure 5.7(a) contains a localized face with a mustache and dishevelled hair. Figure 5.7(b) contains a face with thick glasses.

FIGURE 5.4. The video output for a sample camera snapshot

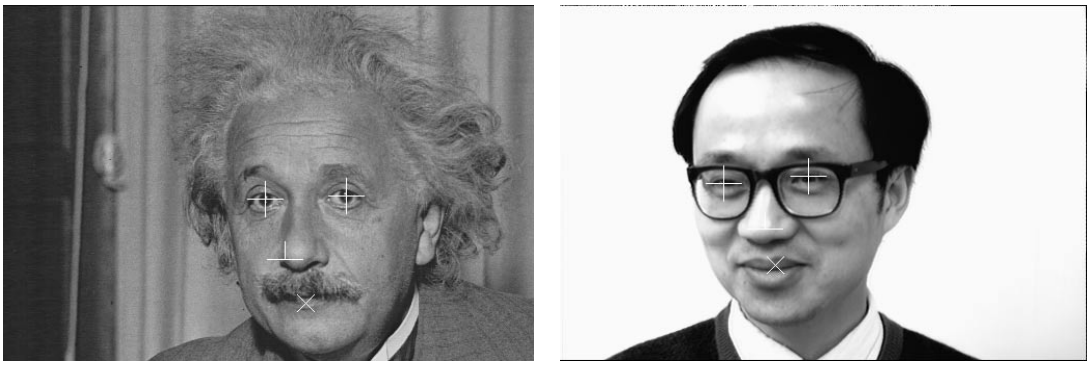
(a) (b)

FIGURE 5.5. Sample test images.



(a) (b)

FIGURE 5.6. Sample test images.



(a) (b)

FIGURE 5.7. Sample test images.

(a) (b) (c)

FIGURE 5.8. Sample test images.

FIGURE 5.9. The 30 individuals in the Achermann database

Figure 5.8(a) shows the localization of a face with a large in-plane rotation and unusual lighting. Figure 5.8(b) depicts a face which is situated in a complex background, under unusual lighting, with in-plane rotation and at a small scale. Figure 5.8(c) depicts the localization of a face with dark skin against a bright background. Note the algorithm's ability to localize the faces in these everyday images despite variations in pose, expressions, facial paraphernalia, lighting and background clutter.

2.2. Recognition Test The database chosen to test the algorithm is the Achermann database from the University of Bern in Switzerland. The database contains 30 individuals with 10 different views of each. Figure 5.9 contains a sample image of each of the 30 individuals in the database.

Unlike many other databases which contain only frontal views, the faces in this database span a variety of depth rotations. The individuals in the database are presented in 10

FIGURE 5.10. The 10 different views per individual in the database

different poses. Poses #1 and #2 display the individual in a frontal pose. Poses #3 and #4 depict the face looking to the right and poses #5 and #6 depict it looking to the left. Poses #7 and #8 depict the face looking downwards and poses #9 and #10 depict it looking upwards. The different views are shown in Figure 5.10.

The changes the face undergoes in Figure 5.10 include large out-of-plane rotations which cause large nonlinear changes in the 2D images. Thus, there is a need for an algorithm that can compensate for out-of-plane rotations, such as the proposed 3D normalization developed in Chapter 4.

For recognition, the algorithm is first trained with 1 sample image for each of the 30 faces. These training images are then stored as KL-encoded keys. Then each of the 300 images in the Bern database is presented to the algorithm as a probe image. The system outputs the best match to the probe from its collection of 30 training images (of the 30 individuals). A sample of the recognition output is shown in Figure 5.11. On the left is a probe image from the 300-element Achermann database and on the right is the closest match the system has in its 30 training images.

The number of correct matches and incorrect matches were determined and a recognition rate of 65.3% was obtained. In other words, of the 300 test images, the system

FIGURE 5.11. The recognition video output

recognized 196 correctly and 104 incorrectly. Had the system been purely guessing the identity of the subject, the recognition rate would be $\frac{1}{30} = 3.3\%$. This performance was achieved with only 1 training image per individual in the database. If the number of training images per individual is increased, superior performance can be expected [**23**]. These results are similar to the ones observed by Lawrence [**23**] whose recognition algorithm utilizes a self-organizing map followed by a convolution network. Lawrence achieves 70% recognition accuracy using only 1 training image per individual. However, his algorithm requires restricted pose variations. Furthermore, Lawrence tested his algorithm on the ORL (Olivetti Research Laboratory) database which has more constrained pose changes than the Achermann database.

In Figure 5.12 we plot the recognition rates of the algorithm for each of the different views (from pose #1 to pose #10) to analyze its effectiveness to pose variations. As expected, the algorithm fares best with frontal images (poses #1 and #2). The algorithm recognized the left and right views (poses #3, #4, #5 and #6) better than the up and down views (poses #7, #8, #9, #10). It seems that the recognition is more sensitive to up and down poses. In comparison, most conventional algorithms have trouble with poses #3 to #10 which are non-frontal.

In Figure 5.13 we plot the effectiveness of the algorithm for each of the 30 subjects displayed in Figure 5.9. Subjects #1, #8 and #9 were especially difficult to recognize while subjects #2, #15, #17 and #28 were always recognized. Subjects with particularly distinct features (such as a beard) were easier to recognize. This is expected since the algorithm distinguishes faces on the basis of intensity variances. Thus, large changes in the face induced by beards, etc. cause the most variance and are easiest to recognize.

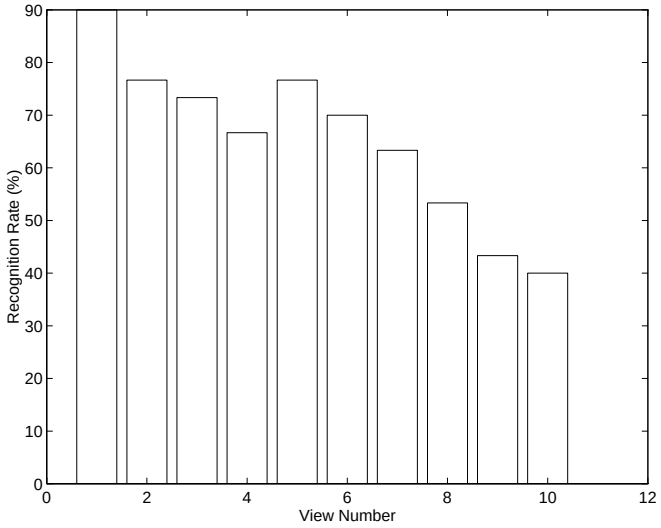


FIGURE 5.12. The recognition rates for different views

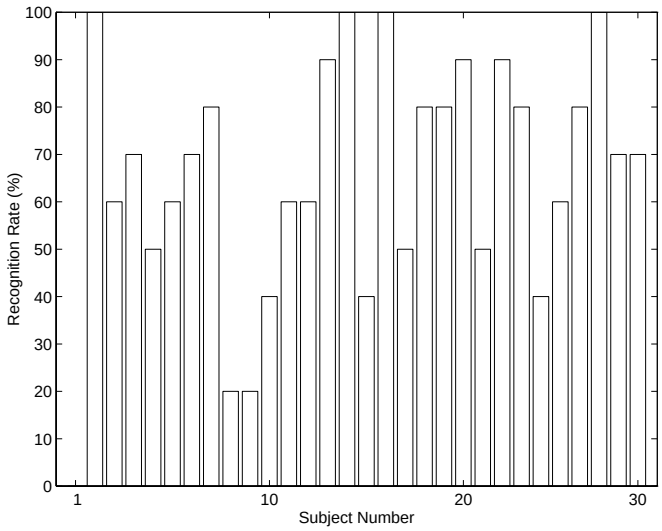


FIGURE 5.13. The recognition rates for different individuals

2.3. Sensitivity Analysis Evidently, the recognition stage depends on the accuracy of the localization stage. We wish to observe the effect of localization errors on the recognition process to quantify the sensitivity of the recognition algorithm. Recall the normalization procedure described in Chapter 4. We shall represent this procedure as a function, N, which acts on an image, I, to produce a standard mug-shot (*probe*) which can be used to search for a match in the database. The parameters of the normalization are the

(a) (b)

FIGURE 5.14. Original Normalization. (a) Input image with anchor points. (b) Corresponding synthesized mug-shot image.

(x, y) positions of the 4 feature points i_0, i_1, i_2 and i_3 (left eye, right eye, nose and mouth) and the original intensity image I. This computation is shown in Equation 5.1:

$$probe = N(i_{0_x}, i_{0_y}, i_{1_x}, i_{1_y}, i_{2_x}, i_{2_y}, i_{3_x}, i_{3_y}, I) \tag{5.1}$$

We then introduce the concept of recognition certainty. Certainty involves comparing the distance a face has from its correct match in the database to the distance it has from other members in the database. Evidently, our recognition output is more reliable (or has better certainty) if a face is much closer to its match and than it is to other members of the database. Consider face#0 from the database which is shown with its synthesized mug-shot image in Figure 5.14. The eyes, nose and mouth have been localized accurately at positions i'_0, i'_1, i'_2 and i'_3. These anchor points are used to generate a mug-shot version of face#0 using Equation 5.1.

This face is a member of our database ($face_0$) and so $d(probe, face_0) = d(face_0, face_0) = 0$. However, if we perturb the values i'_0, i'_1, i'_2 and i'_3 by a small amount, the resulting probe image using Equation 5.1 will be different and d_{min} will no longer be 0. The distance of the probe face ($probe$) to the correct match face#0 in the database is defined as $d(probe, face_0)$. The distance to the closest incorrect response is $d_{min|x\epsilon(0,P)}$, as defined in equation 5.2:

$$d_{min|x\epsilon(0,P)} = min_{x=1}^{x<P} d(probe, face_x) \tag{5.2}$$

In Equation 5.3 we compute a margin of safety which is positive when the probe resembles its correct match face#0 and negative when the probe matches another element of the database, instead.

$$c = d_{min|x\epsilon(0,P)} - d(probe, face_0) \tag{5.3}$$

Since the probe is actually the result of the function N in Equation 5.1, it has i_0, i_1, i_2 and i_3 as its parameters as well. By the same token, the value of c in Equation 5.3 also has i_0, i_1, i_2 and i_3 as parameters. Thus, it is more appropriate to write $c(i_{0_x}, i_{0_y}, i_{1_x}, i_{1_y}, i_{2_x}, i_{2_y}, i_{3_x}, i_{3_y})$. However, for compactness, we shall only refer to the certainty as c.

We shall now compute the sensitivity of the certainty (c) to variations in the localization (i_0, i_1, i_2 and i_3) for the image I corresponding to face#0. This is done by computing $c(i'_{0_x} + \Delta_{0_x}, i'_{0_y} + \Delta_{0_y}, i'_{1_x} + \Delta_{1_x}, i'_{1_y} + \Delta_{1_y}, i'_{2_x} + \Delta_{2_x}, i'_{2_y} + \Delta_{2_y}, i'_{3_x} + \Delta_{3_x}, i'_{3_y} + \Delta_{3_y})$.

We vary the Δ values which cause the localization of a feature point to move around its original position. The anchor point's displacement caused by a particularly large Δ is depicted in Figure 5.15. The dimensions of this image are 512×342 and the intra-ocular distance is approximately 60 pixels. In the experiments, the Δ values are varied over a range of [-15,15] pixels each. We then synthesize a new mug-shot image from the perturbed anchor points.

Figure 5.16 shows several synthesized images after perturbing Δ_{0_x} and Δ_{0_y}, with all other Δ values fixed at 0. Not surprisingly, the mug-shots that are synthesized appear slightly different depending on the position of i_0 (the locus of the left eye). Similarly, Figure 5.17 shows the approximations after the KL encoding of the mug-shots in Figure 5.16. The approximations, too, are affected, showing that the KL transformations is sensitive to errors in localization. Finally, in Figure 5.18 we show the value of c as we vary Δ_{0_x} and Δ_{0_y}, with all other $\Delta = 0$.

Figure 5.19 shows the value of c for variations in the (x, y) position of the right eye anchor point. Similarly, Figure 5.20 and Figure 5.21 show the same analysis for the nose point under two different views. Finally, Figure 5.22 shows the effect of perturbing the mouth point. This surface is quite different from the ones in the previous experiments. In fact, the value of c stays constant and positive indicating that the changes in the mouth

(a) (b)

FIGURE 5.15. Perturbed Normalization. (a) Input image with anchor points. (b) Corresponding synthesized mug-shot image.

position have brought no change in the synthesized mug-shot and that the recognition performance is unaffected by mouth localization errors. This is due to the insensitivity the 3D normalization procedure (defined in Chapter 4) has to the locus of the mouth point.

The above plots show that the localization does not have to be perfect for recognition to remain successful. In these graphs, as long as the value of c is positive, then the subject is identifiable. Even though these sensitivity measurements were made by varying only two parameters, the 8 dimensional sensitivity surface can be approximated by a weighted sum of the 4 individual surfaces described above. Thus, an 8 dimensional sub-space exists which defines the range of the 8 Δ perturbations that will be tolerated before recognition errors occur. In short, the anchor-point localizations may be perturbed by several pixels before recognition degrades excessively.

From the above plots, it is evident that the Δ_{2_y} (the nose locus) is the most sensitive anchor point since it causes the most drastic change in c. Consequently, an effort should be made to change the normalization algorithm to reduce the sensitivity of the recognition to this locus. On the other hand, there is a large insensitivity to the location of the mouth. This is due to the limited effect the mouth has in the normalization procedure. In fact, the mouth only determines the vertical stretch or deformation that needs to be applied to the 3D model. Thus, an effort should be made to use the location of the mouth more actively in the normalization algorithm discussed in Chapter 4. Recall that an error E_{mouth} was present in the normalization while the other 3 anchor points always aligned perfectly to the

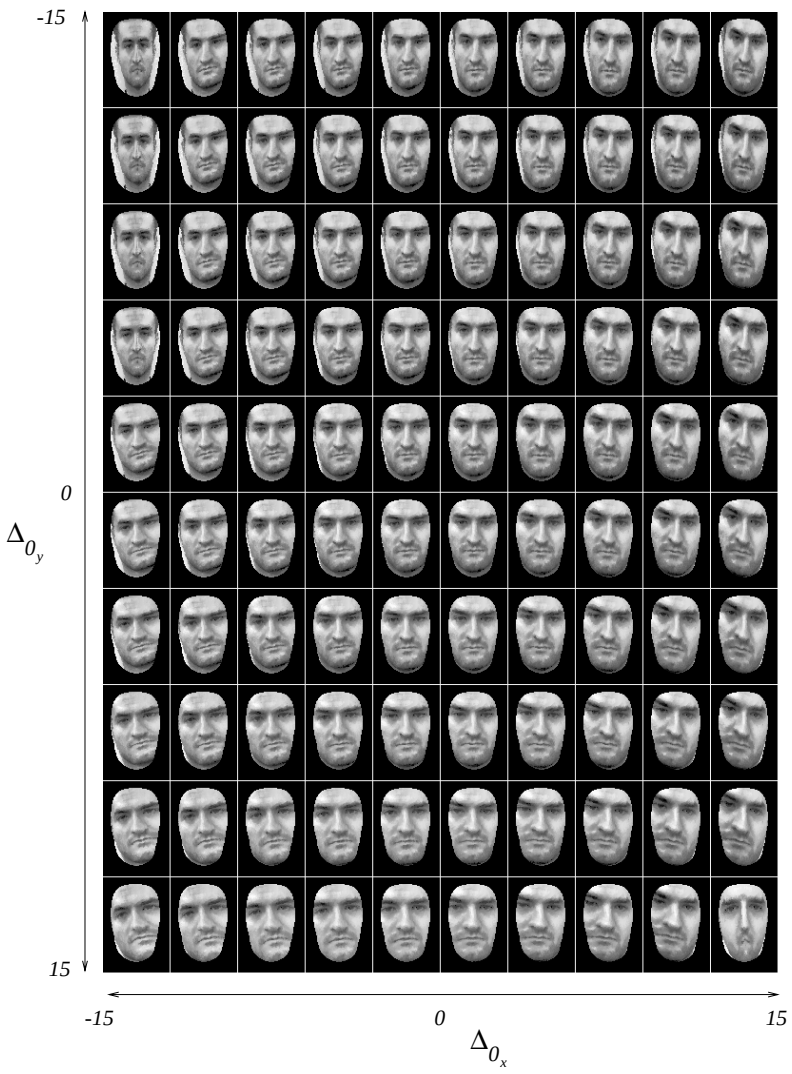


FIGURE 5.16. Synthesized mug-shots with left eye anchor point perturbed

eyes and nose in the image (using the WP3P). Thus the 3D normalization has an alignment error concentrated upon the mouth-point which is the only point on the 3D model which does not always line-up with its destination on the image. If this error could be distributed equally among all four features points, each point will be slightly misaligned and the total misalignment error would be less. Consequently, the 3D model's alignment to the face in the image would be more accurate, overall. Thus, we would attempt to minimize E_{total} as in Equation 5.4 instead of minimizing E_{mouth} with $E_{left-eye} = E_{right-eye} = E_{nose} = 0$. The end result would be an overall greater insensitivity and recognition robustness for the 8 localization parameters combined.

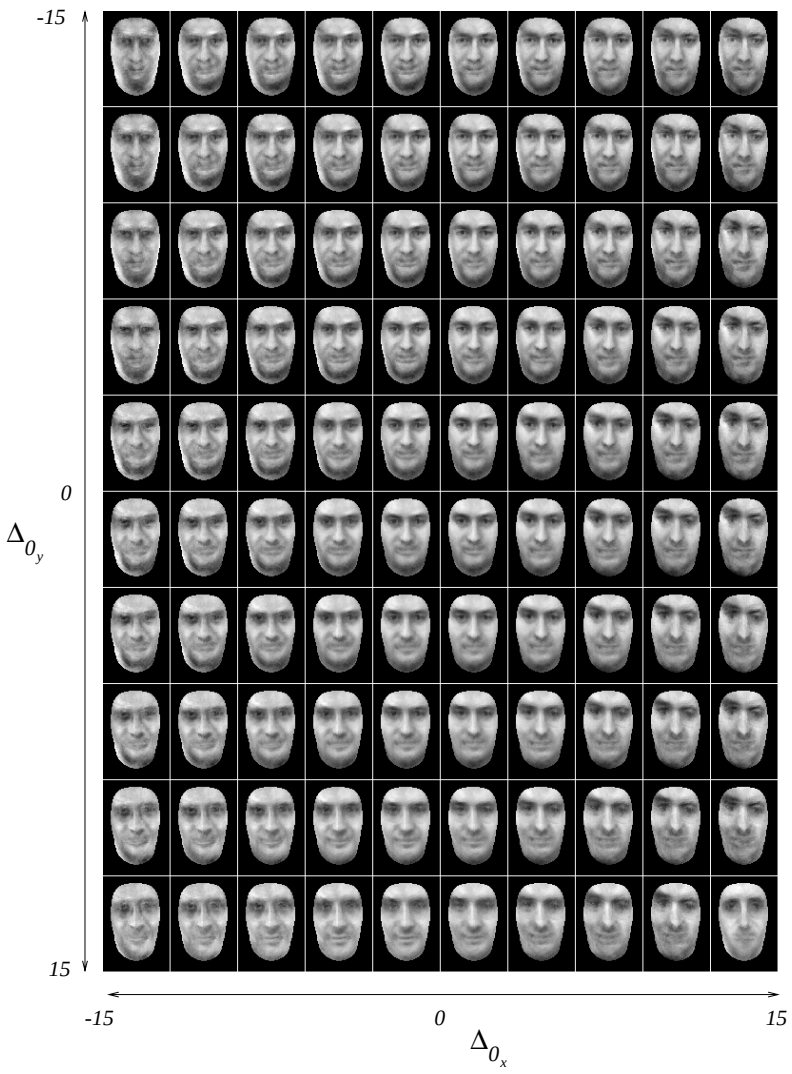


FIGURE 5.17. KL approximations to mug-shots with left eye anchor point perturbed

$$E_{total} = \sqrt{E^2_{mouth} + E^2_{left-eye} + E^2_{right-eye} + E^2_{nose}} \tag{5.4}$$

We have thus presented the system's structure as a whole. The localization and recognition tests evaluate its performance. For one training image, the system is competitive when compared to contemporary face recognition algorithms such as the one proposed by Lawrence[**23**]. Other current algorithms include [**32**] and [**29**] which report recognition rates of 98% and 92% respectively. However, these algorithms were tested on mostly frontal images (not the Achermann database or similar database). Finally, a sensitivity analysis depicts the dependence of the recognition module on the localization output. We see that

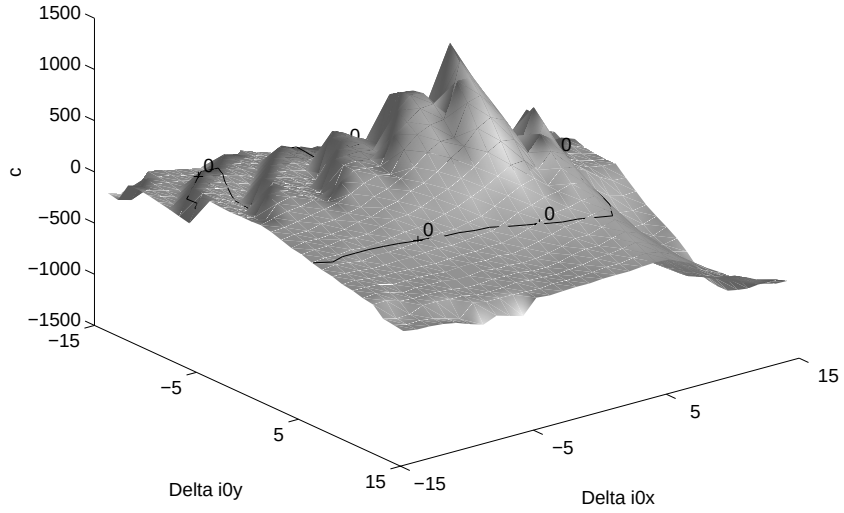


FIGURE 5.18. Variation in recognition certainty under left eye anchor point perturbation

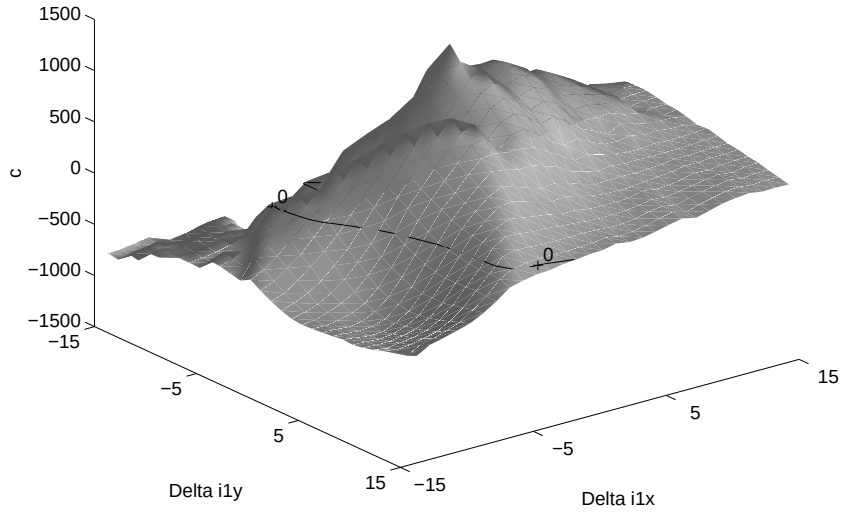


FIGURE 5.19. Variation in recognition certainty under right eye anchor point perturbation

the localization does not have to be exact for recognition to be successful. However, the sensitivity plots do show that recognition is not equally sensitive to perturbations in the localization of different anchor point. Thus, the normalization process needs to be adjusted to compensate for this discrepancy.

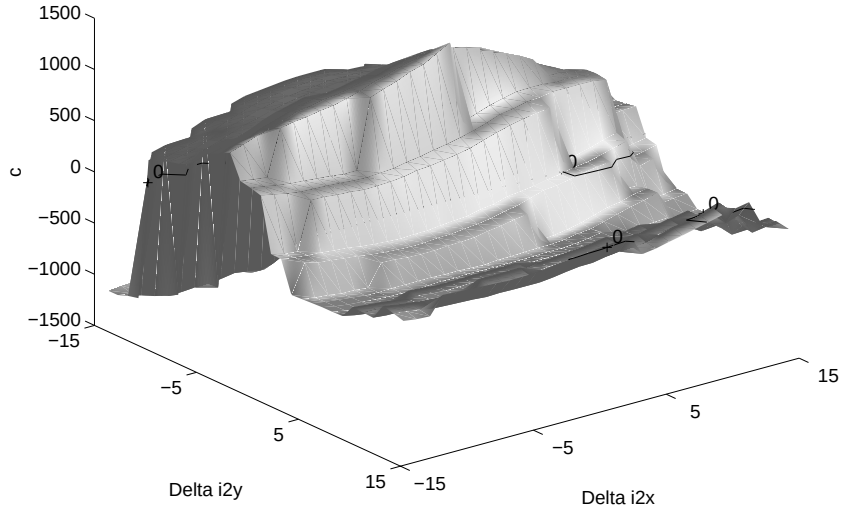


FIGURE 5.20. Variation in recognition certainty under nose anchor point perturbation (View 1)

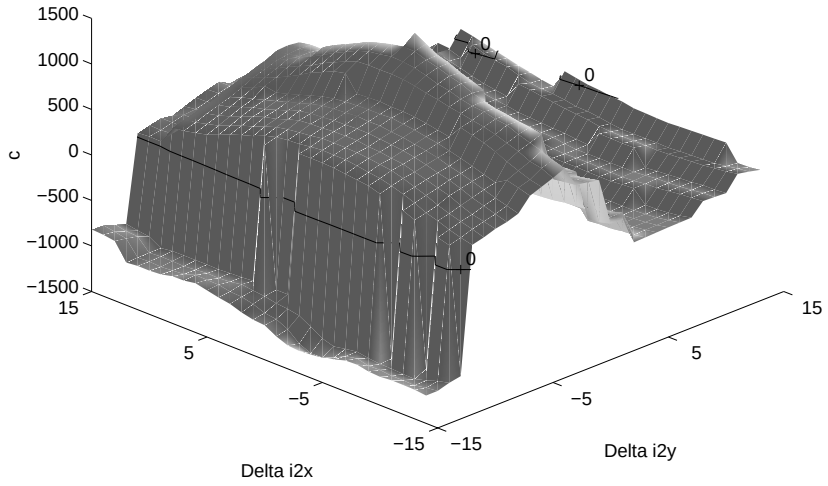


FIGURE 5.21. Variation in recognition certainty under nose anchor point perturbation (View 2)

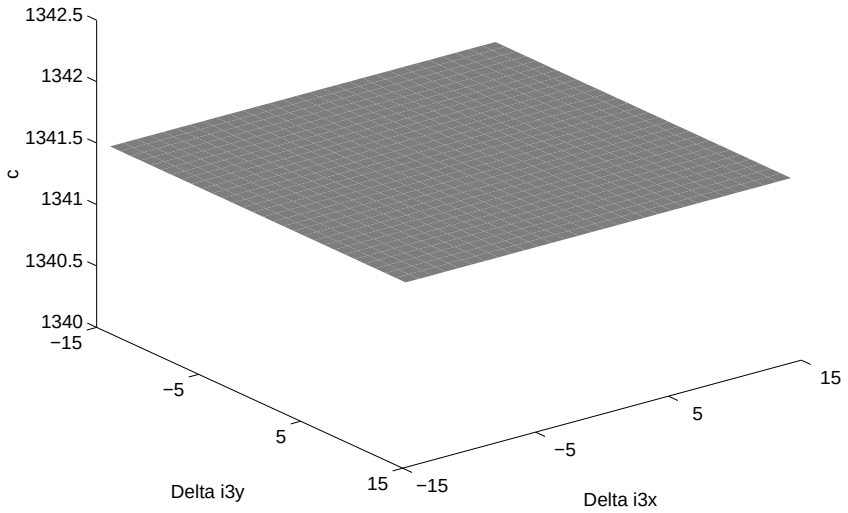


FIGURE 5.22. Variation in recognition certainty under mouth anchor point perturbation

CHAPTER 6

Conclusions

A face recognition system must be able to recognize a face in many different imaging situations. The appearance of a face in a 2D image is not only influenced by its identity but by other variables such as lighting, background clutter, and pose. This thesis has described a method for detecting the face despite these variations, as well as a 3D-based normalization algorithm to compensate for them. This step essentially removes the large nonlinear effects of pose, lighting and background clutter, eliminating much of the variance these variables generate. In essence, the normalization generates a mug-shot from an arbitrary, uncontrived image containing a face. This then permits the use of holistic linear recognition techniques (which require frontal mug-shots). Thus, the 3D normalization acts as a bridge connecting the detection algorithm (which can handle arbitrary input images) to the recognition engine.

The system was motivated by a number of criteria. The algorithm had to find faces efficiently without exhaustively searching the image. Thus, we proposed the use of low-level perceptual mechanisms based on contrast, symmetry and scale. This allows us to focus computational resources on perceptually significant objects in a scene. We discussed the development of attentional mechanisms and the symmetry transform which quickly pinpoint interesting objects in a scene. The symmetry transform was then adapted to selectively search for the symmetry found in facial contours.

We then described a framework for using low-level mechanisms to find faces and facial features in an arbitrary image. The proposed hierarchical search is efficient and robust and proceeds in a coarse-to-fine method as it searches for faces. We begin by finding face-like blobs in the image. Then we identify the facial contour that encloses the face and use it to restrict the search for eyes in the image. Blobs that resemble eyes are then detected and we

proceed to search for a mouth. Once a mouth is located, an approximation of the position of the nose is found using signature analysis.

Subsequently, we developed a 3D normalization technique which compensates for any facial pose by using the positions of the eyes, the nose and the mouth. This 3D normalization uses a deformable 3D model of the average human head. The model is used to synthesize a mug-shot from a localized face in an arbitrary input image. Thus, we effectively compensate for pose variations and background clutter. Subsequently, the lighting is normalized using a mixture of windowed histogram fitting transfer functions. The effectiveness of the normalization technique was then illustrated with several examples.

Next, the details of the Karhunen-Loeve based statistical detection and recognition algorithm were presented. The KL transform is used to compress the face data by over two orders of magnitude. In addition, we adapted the KL transform to function as a detection mechanism for improving the localization of faces and for discarding incorrect localizations. In addition, the KL provided a convenient way of matching a probe image to a database of faces using distance measurements.

The complete implementation was then described. The assembly of individual modules in software as well as the user interface were discussed. We then tested the system with arbitrary input images to determine its effectiveness at localizing faces. Then, the recognition rates were computed on a standard database. We also performed a sensitivity analysis to determine the effect of localization on recognition.

The localization and recognition results obtained demonstrate the algorithm's ability to handle a variety of pose changes, illumination conditions and background clutter. This is due to the 3D normalization stage which links the localization output to the recognition engine. Thus, we overcome the constraints of holistic linear recognition which requires mug-shot input images. The 3D normalization and illumination correction act as an intermediate step which links the robust feature detection stage to the precise linear recognition stage.

1. Contributions

We approached face detection using a hierarchical, multi-scalar, blob and limb search. The use of low-level attentional mechanisms and symmetry was found to be efficient for face localization and robust to pose variations. Traditional face localization techniques require contrived images and/or excessive computational resources.

We also introduced the selective symmetry transform to extend the concepts of symmetric enclosure to non-circular contours (such as the semi-elliptical ones found in human

faces). This permits the selective detection of particular blob shapes. Most other facial contour estimation techniques are not motivated by low-level perception theory and utilize template matching or deformable models without a measure of symmetric enclosure.

Traditionally, there have been two approaches to face recognition: feature-based techniques and holistic techniques. We propose a way of bridging the gap between the two techniques by synthesizing mug-shot images from arbitrary input images with a high-quality 3D normalization. The 3D normalization involves a deformable 3D model of the average human face which produces superior normalization for out-of-plane rotations than other techniques. Thus, we combine the robust face detection of a feature-based stage with the precise classification capabilities of a linear, holistic, recognition stage.

We proposed the use of mixed-histogram fitting to correct for illumination. The gradual mixture of different histogram correction functions allows a smooth illumination correction for both sides of a face. Typically, illumination correction was performed on the face as whole. We window the histogram analysis and use weighted mixtures of the histogram transfer functions to achieve superior illumination correction.

Finally, we proposed a sensitivity analysis relating the localization accuracy to the recognition certainty. This form of analysis relates the performance of two different components of face recognition: localization and classification.

2. Direction of Future Work

Current work is being done to improve the accuracy of the normalization, localization and recognition. Furthermore, other possible applications of the algorithm are being investigated.

We are investigating alternate ways of computing the WP3P to obtain a more accurate alignment of the 3D model. As was shown in the sensitivity analysis, the recognition is too sensitive to the nose localization. A 3D normalization which depends less upon this anchor point might improve recognition results.

Alternativeely, we can improve localization accuracy. Some proposed techniques include the use of eigenspace analysis of the mug-shot to determine normalization and localization errors directly. In other words, we are investigating a way to compute a DFFS vector or gradient as opposed to a simple DFFS scalar value. This would allow the recognition to detect and report the orientation of small localization errors to the normalization stage. Thus, a gradient descent could be used to optimize the localization. Furthermore, we also have the option of performing more DFFS measurements in the neighbourhood around the

initial localization from the face detection stage. The sample DFFS measurements could be used to approximate a vector or DFFS gradient which could then be used by a gradient descent optimization to quickly converge to a superior localization.

We are also considering preceding the recognition stage with some transforms to desensitize it to small errors in the localization stage. Preceding the KL transform with a Fourier, discrete cosine or wavelet transform will yield some small spatial invariance which might generate superior overall recognition rates. In addition, the size of the synthesized mug-shots could be increased so that more pixels are used to represent a face. Such higher resolution mug-shots could yield superior recognition rates. We are also investigating possible ways to group faces into categories (race, facial hair, sex, etc.) to generate specialized eigenspaces for each class. This would yield more accurate recognition since eigenvectors would be dedicated to the members of the group (i.e. eigenfaces for bearded men, eigenfaces for women, etc.) Finally, we are testing the algorithm's recognition rates for multiple training images per individual (instead of only one training image per individual).

The algorithm could be optimized to perform face tracking in real-time. This would allow the face to be used as a control device for the handicapped or for gaze-detection-based virtual reality. Furthermore, the algorithm could be used for low-bandwidth video conferencing applications since the face data is compressed to 60 scalar coefficients using the KL decomposition. Finally, we propose the use of the algorithm in interactive 3D movies since the 3D normalization allows the user to view many different poses of an individual from a single original photograph (or video stream). These are just a few of the many applications of face recognition, detection and normalization.

REFERENCES

[1] S. Akamatsu, T. Saski, H. Fukamachi, N. Masui, and Y. Suenaga. Robust face identification scheme. In *Proceedings of SPIE - The International Society for Optical Engineering*, pages 71–84, 1992.

[2] T.D. Alter. 3d pose from 3 corresponding points under weak-perspective projection. Technical Report 1378, MIT Artificial Intelligence Laboratory, 1992.

[3] F. Attneave. Some informational aspects of visual perception. *Psychological Review*, pages 183–193, 1954.

[4] J.M. Barr. Speeded phase dicrimination: evidence for global to local processing. In *Graphics Interface 1986*, pages 331–336, 1986.

[5] S. Basu, I. Essa, and A. Pentland. Motion regularization for model-based head tracking. Technical Report 362, MIT Media Laboratory - Perceptual Computing Section, 1996.

[6] J.F. Canny. Finding edges and lines in images, master's thesis. Technical report, MIT Artificial Intelligence Laboratory, 1983.

[7] R. et al. Chellappa. Human and machine recognition of faces: A survey. Technical Report CAR-TR-731, Univeristy of Maryland Computer Vision Laboratory, 1994.

[8] C. Colombo, A. Del Bimbo, and S. De Magistris. Human-computer interaction based on eye movement tracking. *Computer Architectures for Machine Perception*, pages 258–263, 1995.

[9] I.J. Cox, J. Ghosn, and P.N. Yianilos. Feature-based face recognition using mixture-distance. Technical Report TR-95-09, NEC Research Institute, 1995.

[10] I. Craw, D. Tock, and A. Bennett. Finding face features. In *Second European Conference on Computer Vision*, pages 92–96, 1992.

[11] R. Deriche. Using canny's criteria to derive a recursively implemented optimal edge detector. *International Journal on Computer Vision*, pages 167–187, 1987.

[12] J. Elder. Shadows, defocus and reliable estimation, phd. thesis. Technical report, McGill Research Centre for Intelligent Machines, 1995.

[13] M.A. Fischler and R.C. Bolles. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. In *Communications of the ACM*, number 6, pages 381–394, 1981.

[14] R.C. Gonzalez and R.E. Woods. *Digital Image Processing*. Addison-Wesley Publishing Company (New York), 1992.

[15] H.P. Graf, T. Chen, E. Petajan, and E. Cosatto. Locating faces and facial parts. In *International Workshop on Automatic Face- and Gesture-Recognition*, pages 41–46, 1995.

[16] R.M. Haralick, H. Joo, C.-N. Lee, X. Zhuang, V.G. Vaidya, and M.B. Kim. Pose estimation from corresponding point data. *IEEE Transactions on Systems, Man and Cybernetics*, (6):1426–1446, 1989.

[17] C.W. Helstrom. *Statistical Theory of Signal Detection*. Pergamon Press, 1968.

[18] R. Horaud. New methods for matching 3-d objects with single perspective views. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, (3):401–411, 1987.

[19] T. Kanade. *Picture Processing System by Computer Complex and Recognition of Human Faces, PhD. Thesis*. PhD thesis, Kyoto University, Japan, 1973.

[20] M.F. Kelly. *Annular symmetry operators: A multi-scalar framework for locating and describing imaged objects*. PhD thesis, McGill University, 1995.

[21] M.F. Kelly and M.D. Levine. The symmetric enclosure of points by planar curves. Technical Report CIM-1-93, McGill Research Centre for Intelligent Machines, 1993.

[22] M. Kirby and L. Sirovich. Application of the karhunen-loeve procedure for the characterization of human faces. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, (1):103–108, 1990.

[23] Steve Lawrence, C.L. Giles, A.C. Tsoi, and A.D. Back. Face recognition: A hybrid neural network approach. Technical Report UMIACS-TR-96-16, Institute for Advanced Computer Studies, University of Maryland, 1996.

[24] J.Y. Lettvin, H.R. Maturana, W.S. McCulloch, and W.H. Pitts. What the frog's eye tells the frog's brain. In *Proceedings of the IRE*, pages 1940–1951, 1959.

[25] M.D. Levine. *Vision in Man and Machine*. McGraw-Hill, 1985.

[26] S. Linnainmaa, D. Harwood, and L. Davis. Pose determination of a three-dimensional object using triangle pairs. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, (5):634–647, 1988.

[27] D. G. Lowe. *Perceptual Organization and Visual Recognition.* Kluwer Academic Publishers, 1985.

[28] B.S. Manjunath, R. Chellappa, and C.V.D. Malsburg. A feature based approach to face recognition. In *Proceedings, IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 373–378, 1992.

[29] B. Moghaddam, C. Nastar, and A. Pentland. Bayesian face recognition using deformable intensity surfaces. Technical Report 371, MIT Media Laboratory Perceptual Computing Section, 1996.

[30] C. Nastar and A. Pentland. Matching and recognition using deformable intensity surfaces. In *International Symposium on Computer Vision*, pages 223–228, 1995.

[31] M. Nixon. Eye spacing measurement for facial recognition. In *SPIE Proceedings*, pages 279–285, 1985.

[32] A. Pentland, B. Moghaddam, and T. Starner. View-based and modular eigenspaces for face recognition. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 84–91, 1994.

[33] P.J. Phillips and Y. Vardi. Data-driven methods in face recognition. In *International Workshop on Automatic Face- and Gesture-Recognition*, pages 65–70, 1995.

[34] H.V. Poor. *An Introduction to Signal Detection and Estimation.* Dowden and Culver, 1988.

[35] W.H. Press, S.A. Teukolsky, W.T. Vetterling, and B.P. Flannery. *Numerical Recipes in C.* Cambridge University Press, 1992.

[36] R. Rao and D. Ballard. Natural basis functions and topographics memory for face recognition. In *International Joint Conference on Artificial Intelligence*, pages 10–17, 1995.

[37] D. Reisfeld. Generalized symmetry transforms: attentional mechanisms and face recognition, phd. thesis. Technical report, Tel-Aviv University, 1994.

[38] D. Reisfeld and Y. Yeshurun. Robust detection of facial features by generalized symmetry. In *11th IAPR International Conference on Pattern Recognition*, pages 117–120, 1992.

[39] R.W. Rodieck and J. Stone. Analysis of receptive fields of cat retinal ganglion cells. *Journal of Neurophysiology*, 28:833–849, 1965.

[40] N. Roeder and X. Li. Experiments in analyzing the accuracy of facial feature detection. In *Vision Interface '95*, pages 8–16, 1995.

[41] J.B. Roseborough and H. Murase. Partial eigenvalue decomposition for large image sets using run-length encoding. *Pattern Recognition*, (3):421–430, 1995.

[42] G. Sela. Real time attention for robotic vision. Master's thesis, McGill University, 1995.

[43] G.B. Thomas. *Calculus and Analytic Geometry*. Addison-Wesley, 1960.

[44] M.A. Turk and A.P. Pentland. Face recognition using eigenfaces. In *IEEE Computer Society Conference Computer Vision and Pattern Recognition*, pages 586–591, 1991.

[45] A. Yuille, D. Cohen, and P. Hallinan. Feature extraction from faces using deformable templates. In *IEEE Computer Society Conference on Computer Vision and Templates*, pages 104–109, 1989.

Document Log:

Manuscript Version 0
Typeset by $\mathcal{A}_{\mathcal{M}}\mathcal{S}$-LaTeX — 2 May 1996

Tony S. Jebara

Centre for Intelligent Machines, McGill University, Montréal, Québec, Canada
E-mail address: jebarat@cim.mcgill.ca

Typeset by $\mathcal{A}_{\mathcal{M}}\mathcal{S}$-LaTeX