

Automatic Face Feature Localization for Face Recognition

Christopher I. Fallin

Fall 2008 - Spring 2009

Contents

1	Introduction	1
1.1	Face Recognition	1
1.2	Classifier Types	2
1.3	Holistic: PCA	3
1.4	Feature-based Matching: EBGM	4
1.5	Gabor Jets without EBGM	5
2	Gabor Jets and EBGM	7
2.1	Gabor Jets	7
2.2	EBGM	10
2.2.1	Bunch Graph and System Initialization	10
2.2.2	The Face Graph	11
2.2.3	Distance Metrics	11
2.2.4	Localization	13
2.2.5	CSU EBGM	14
3	Information Content in Gabor Jets	16
3.1	Variance-based Goodness Measure	17
3.2	Face Space	18
3.3	Face Space Results	18
4	Fiducial Point Selection	20
4.1	What is a Fiducial Point?	21
4.2	Regions of Information Content	21
4.3	Edge Structure by Triangulation	23
5	Prelocalization	26
5.1	Prelocalization: Optimizing Placement	26

5.2	Jet Pseudo-Bunch Construction	27
5.2.1	K-means clustering	27
5.2.2	Displacement Estimation from Phase	30
5.2.3	Prelocalization Results and Conclusions	31
6	FERET Matching Results	34
6.1	Sample Face Graphs	34
6.2	Experimental Setup	34
6.3	Recognition Rate Results: Full FERET and Statistical Per- mutations	36
6.4	Further Work	38
7	FRGC: Large-Dataset Optimization and Results	41
7.1	FRGC Challenges	42
7.2	FRGC Results	43
7.3	FRGC Future Work	44
A	Technical Details	47
A.1	The mark5 System	47
A.1.1	Datasets	48
A.1.2	Information Content: Condor Run	48
A.1.3	Fiducial Points and Prelocalization	49
A.2	CSU EBGM	49
A.3	FRGC Additions	50
A.4	A Last Note	51

List of Figures

1.1	Fixed-Position Gabor Jet Results	6
2.1	Several examples of Gabor wavelets	9
2.2	Gabor jet extraction sample	9
2.3	A face graph overlaid on corresponding face image	12
3.1	Variance Metric	19
4.1	Information content map over face space	22
4.2	Probability density over face space	22
4.3	Delaunay triangulation of an early face graph	25
5.1	Displacement-estimation plots: green lines trace displacements	32
6.1	Face graphs before and after prelocalization	35
6.2	ROC curves on FERET dataset	37
7.1	<code>ptpicker</code> application	43
7.2	FRGC fiducial point picker results	43
7.3	ROC curves for FRGC dataset performance	45
7.4	Average faces compared	46
7.5	Face graphs compared	46

Abstract

Classifiers employed in face recognition typically take as input a rectangular matrix of pixel values. Feature-based classifiers will examine some subset of these pixels – fiducial points – in order to extract useful information and make a decision. In the Elastic Bunch Graph Matching method, fiducial points must be placed manually on an initial set of images so that the system learns the features. In this thesis, a method for automatically choosing these points is presented, eliminating a tedious process and generalizing the system to different sorts of images while making almost no impact on matching performance. Recognition rates are examined and presented against performance of various other systems on the FERET dataset. The thesis closes with a technical overview of the prototype system built for this purpose.

Chapter 1

Introduction

Biometrics in general is the field of study relating to measures of human identity. The etymology of the word – *bio-*, life, and *metric*, measure, suggest a quantitative study of some measurable aspects of humans, usually some aspect of appearance. Work has been done to establish human identity through various measures, among them fingerprint, iris, and face images, and also more exotic measures such as gait (walking pattern) recognition. Two fundamental problems are studied: identification and verification. Identification observes a face without any other information given and answers the question “Who is this?” Verification is simpler: given a face and a claimed identity, either confirm or deny that identity. This thesis focuses on the identification task.

1.1 Face Recognition

Face recognition is a broad subfield of biometrics, drawing equally from the fields of computer vision and machine learning and attracting researchers with diverse backgrounds in signal processing, pattern recognition and mathematics, for example. Face recognition aims to solve the two problems described above – identification and verification – by the use of images (of whatever sort) of the human face. Many techniques have been used to solve this problem over the past two decades. The problem at its heart is relatively straightforward to state, however: given some digitized image¹ of a human

¹Visible-light and infrared images have been used, as well as video, both as a source of still frames and as a motion sequence.

subject's face, draw upon knowledge extracted from a training set to classify the given face. The task reduces to the extraction of the distinguishing features of a given subject's identity, the mapping of the feature-space of the training set into some structure that can be easily and efficiently used by the classifier, and finally the evaluation of an unknown *probe image* against the learned knowledge. The feature extraction and matching must be specific enough to retain accuracy even in huge training sets, yet coarse enough to generalize to probe images that have never before been seen. In essence, the model must neither overfit nor underfit the data.

Such a model, built from training data in anticipation of unknown probe data, is called a *classifier* (as studied in machine learning) and several types exist in face recognition. We begin by examining several of these types.

1.2 Classifier Types

A classifier is a function $h : X \mapsto Y$ where X is the input space (typically, a vector of pixel values for a digitized face) and Y is the set of classes, in this case the identities of the subjects.

Two general categories of classifiers exist in face recognition: holistic (appearance-based) and feature-based systems [9]. Holistic classifiers treat face images as a matrix of pixels only, without assigning more specific meaning to particular regions; typically these systems use statistical methods to extract the meaningful portions of the images. For such an approach to perform well, there is typically some knowledge of the image content, if only in preprocessing stages, for example to ensure that faces are aligned before matching them. However, such a system ultimately works with information learned directly from a matrix of pixels.

Feature-based classifiers are the other major type of system. Such a system has some knowledge of specific face features (eyes, nose, or some other distinguishing location) and builds a feature vector based on these particular features. This can provide pose-invariant matching and can also lead to more compact feature vectors. However, accurate localization of predetermined features will usually require a well-tuned feature detection system and careful selection of the pertinent features themselves.

An example of each type of system is given below.

1.3 Holistic: PCA

Principal component analysis (PCA) is an application of statistical techniques to feature vectors in order to extract basis vectors that best represent the original subjects. Typically only a small number of the first N basis vectors are taken, reducing the dimensionality of the data significantly; the first basis vectors represent the most significant variations in the dataset, allowing this reduction of dimensionality with minimal impact on accuracy. This has the advantage of compact representation, but more significantly, matching comparisons are more meaningful when the important variations are isolated.

To find these basis vectors, the images are normalized by subtracting the average image and the covariance matrix C of these image vectors is computed. Specifically, if each image (following DC normalization) is represented by a column vector Φ_n , and $A = [\Phi_1 \Phi_2 \dots \Phi_n]$, then $C = AA^T$. The first N eigenvectors of this covariance matrix, when sorted by largest eigenvalue first, are taken to be the basis vectors. This maximizes the variance of the face-space vectors; the proof is beyond the scope of this overview. In order to compute the eigenvectors efficiently, usually the eigenvectors of $A^T A$ are taken (this matrix scales by the number of training images rather than the size of the images, and thus is much smaller); then for a given eigenvector v_i of this smaller matrix, Av_i is the corresponding eigenvector in the original dimensionality.

The original application of PCA to face recognition was by Turk and Pentland in [7], in which the basis vectors are labeled “eigenfaces.” The system projects unknown faces onto the basis vectors to obtain feature vectors for novel faces, and then uses Euclidean distance between feature vectors to determine face similarity. Given a training set of faces, the classification of a probe image requires finding the feature vector in the training set that is nearest to the probe vector.

Interestingly, because face space is a subspace of the original image space, the system lends itself to an easy test as to whether a face is present in an image. It is enough to find the distance between an arbitrary vector in image space and its projection into the face subspace; if the error is less than a certain threshold, then the image can be considered to be a face. Turk and Pentland in [7] leverage this ability to build a real-time face tracker using eigenfaces.

The PCA-based eigenface method runs quickly and works reasonably well

when the training and probe images come from a controlled environment. As noted in [7], system accuracy is somewhat tolerant of lighting variation but declines with orientation variation and degrades rapidly with scale variation. Feature-based systems attempt to work around this limitation by first localizing the interesting features in some way.

1.4 Feature-based Matching: EBGM

A popular feature-based matching system, Elastic Bunch Graph Matching (EBGM), models a face by a graph with nodes corresponding to features [8, 2]. The graph is *elastic* – that is, the nodes are shifted to match feature locations in the localization process. In order to localize the feature nodes effectively, the model face graph has a *bunch* (collection) of canonical features at each node, and the localization process uses the best match to perform the optimization.

The feature vector extracted from the localized graph consists of the *Gabor jets* computed at each node location. A Gabor jet is a vector of complex numbers, each resulting from the convolution of the image around a point with a Gabor wavelet mask. A Gabor wavelet is a complex-valued sinusoidal function, with the oscillations oriented along a particular axis. The envelope of the wavelet decays exponentially, as a Gaussian. It can be viewed as a set of masks, the even (cosine) and odd (sine) parts corresponding to the real and imaginary components, and the convolution result thus describes the magnitude and the phase of the frequency content at a particular location in the image in a particular direction. In practice, the magnitude is fairly insensitive to slight variations in jet location, whereas the phase can be approximated linearly at small displacements. Taking phases of multiple frequency components, one can estimate displacement accurately, and this technique is at the core of bunch graph node localization.

Matching on these feature vectors can be performed in a straightforward nearest-neighbor manner, using either a phase-insensitive measure (for example, to select the best jet out of a bunch, before point localization) or a phase-sensitive measure (for final matching). Much like other standard face recognition techniques, the existence of a well-defined distance metric between faces allows for computation of rank-one matching scores as well as standard ROC curves and the like.

EBGM will be covered in much more detail in the following chapter, as

it serves as the basis of the work presented in this thesis.

1.5 Gabor Jets without EBGM

Even without the framework of EBGM to localize a particular set of meaningful Gabor jets in a face image, the wavelet techniques show some promise as a simple method of extracting information. Prior to working with EBGM, the author set out to find Gabor jet locations that would yield a feature vector providing some level of recognition performance. This has been explored in one particular manner before – specifically, by use of a genetic algorithm. Gökberk et al. [3] place a regular 7×7 lattice over face images and extract the Gabor jets at each point, comparing the matching accuracy of the feature vectors composed of these jets to a manually-placed fiducial point graph. Several optimization algorithms were used to choose a subset of the points, with the GA obtaining the best results. Surprisingly, better performance was obtained from the regular grid (91.07% rank-one recognition) than from the manually-placed points (87.97% rank-one recognition). Running a genetic algorithm over the set of jets precomputed for each pixel to pick the optimal points for matching yielded a rank-one recognition rate of 96.50%.

The author ran a simple experiment similar to the grid-based matching, placing a 5×5 grid over FERET images shifted and skewed to normalize eye locations (taken from metadata included in Colorado State’s Face Identification Evaluation System) and taking the Gabor jets at each lattice point. Sorting these points by individual matching performance and building feature vectors from the first N sorted lattice points, a maximum recognition performance of 78.52% was obtained at $N = 12$ (results in Figure 1.1). This level of performance is encouraging for such a simple experiment.

These results lend credence to an alternate approach that uses statically-located fiducial points without a separate localization phase. The recognition performance achieved in [3], in particular, is remarkable given the lack of a localization phase for each face image. Clearly, still, the localization performed by EBGM should prove advantageous, especially for outlier faces that might trip up an inflexible static-location system. The rate achieved by [3] should serve, therefore, as a baseline for our EBGM fiducial point optimizations in this thesis, in order to show this to be true.

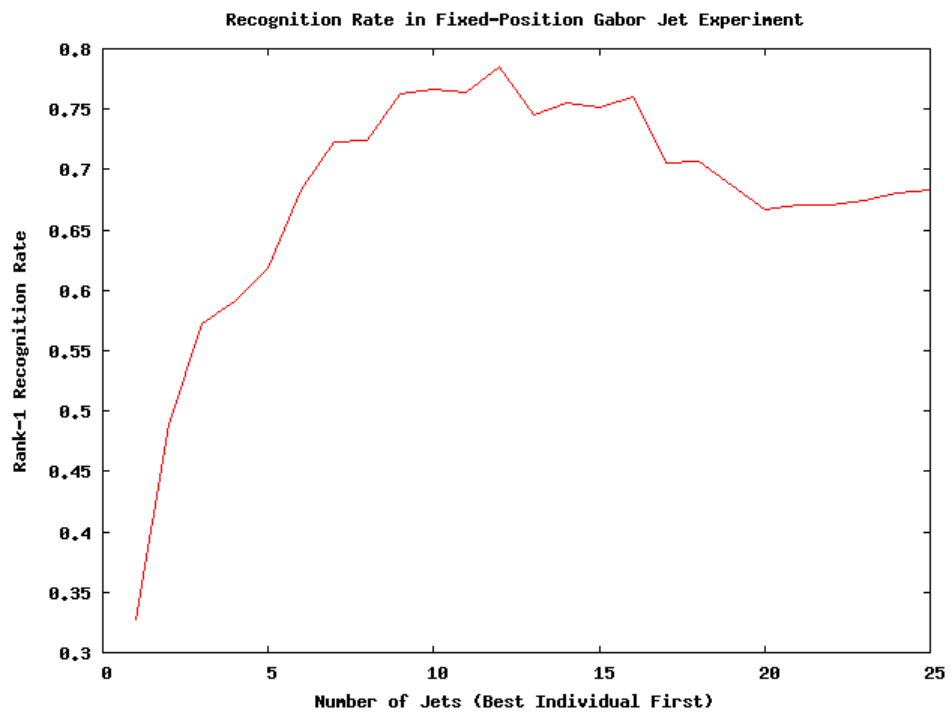


Figure 1.1: Fixed-Position Gabor Jet Results

Chapter 2

Gabor Jets and EBGM

Elastic Bunch Graph Matching (EBGM) [8], a face recognition system that localizes and extracts features using Gabor wavelets, forms the basis for the work presented in this thesis. In order to understand the specific improvements made to the system, let us first describe the system in detail, building a basic understanding of Gabor jet extraction and the structure of the EBGM process. A more detailed overview of EBGM can be found in [2], which is a masters thesis describing the construction of the CSU EBGM implementation used in this work.

2.1 Gabor Jets

EBGM builds on the hypothesis that a combination of frequency-domain and spatial-domain information will yield a more stable representation of face features than either domain alone. A wavelet is a basis function that captures both frequency and location. The Gabor wavelet kernels used in this work are complex exponential functions: a complex sinusoid oriented in a particular direction in two dimensions, modulated by a Gaussian envelope. The wavelet mask is convolved with the image at the point of interest, and the result of this convolution is a complex number indicating the magnitude and phase of image content at that wavelet's frequency at that particular location. For convenience in implementation, the complex wavelet is often considered to be a pair of masks – the real and imaginary, or even (cosine) and odd (sine) masks, yielding a filter response of two real scalars in the feature vector.

We convolve the image with a set of masks (typically 40 complex-valued wavelets) at particular orientations and wavelengths in order to extract sufficient information to characterize the image around the center point of convolution. The set of responses resulting from this process is called the ‘‘Gabor jet’’ at that point. A Gabor jet is thus a vector of 80 scalars, or 40 complex numbers, associated with a single pixel in the source image, representing the neighborhood around that point.

Examples of such Gabor wavelet masks are given in Figure 2.1. We follow the implementations in [8, 2] as far as choice of wavelengths and orientations: we take 8 orientations, at steps of $\pi/8$, and 5 wavelengths, starting at 4 pixels and increasing geometrically with factor $\sqrt{2}$. The parameter σ of the Gaussian envelope is typically set equal to the wavelength λ .

The wavelet mask as implemented here can be formulated as:

$$\begin{aligned} x' &= x \cos(\theta) + y \sin(\theta) \\ y' &= -x \sin(\theta) + y \cos(\theta) \\ W(x, y, \theta, \sigma, \lambda) &= e^{\frac{x'^2 + y'^2}{2\sigma^2}} \cdot e^{j \frac{2\pi}{\lambda} x'} \end{aligned}$$

Essentially, we first transform the xy -plane so that the wavelet sits centered at the origin with zero rotation, and then develop the wavelet as a complex exponential modulated by a Gaussian decaying as a function of r .

The complex element $g_{t,l}$ of Gabor jet g at image point $I(x, y)$ (extracted with a convolution window of size L) is then given by:

$$\begin{aligned} \theta &= \frac{\pi}{8}t \\ \lambda &= 2^{2+\frac{1}{2}l} \\ \sigma &= \lambda \\ g_{t,l} &= \sum_{i=-L}^{+L} \sum_{j=-L}^{+L} W(i, j, \theta, \sigma, \lambda) I(x + i, y + j) \end{aligned}$$

The full Gabor jet g consists of all $g_{t,l}$ for $t \in \{0, 1, \dots, 7\}$ and $l \in \{0, 1, 2, 3, 4\}$. For convenience, we number these elements consecutively as g_k for $1 \leq k \leq 40$.

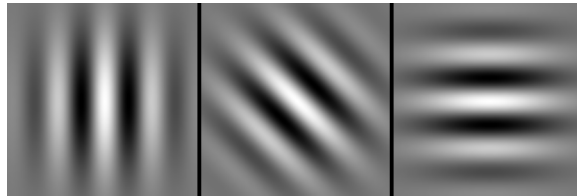


Figure 2.1: Several examples of Gabor wavelets

Properly speaking, the wavelet responses result from convolution extending to infinity; however, for large L , the values are very close due to the Gaussian envelope that decays to 0 away from the origin. In practice, we use $L = 2\sigma$.

An example of Gabor jet extraction can be seen in Figure 2.2; in this plot, grayscale intensity corresponds to the magnitude of g_{33} (an arbitrary-chosen element of the 40-vector) at each pixel. The original image (taken from FERET) and the wavelet mask, with $\theta = 0$ and $\lambda = 16$, are both shown beside the plot.

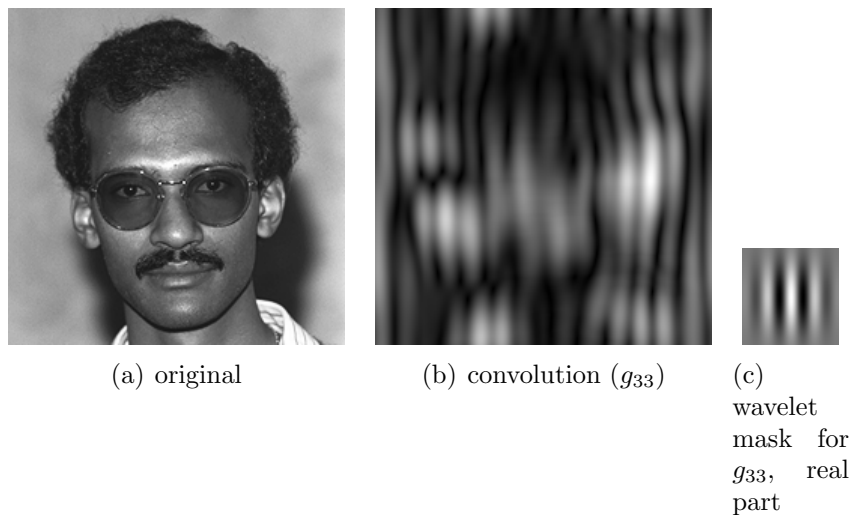


Figure 2.2: Gabor jet extraction sample

Gabor jet extraction is the fundamental image operation on which EBGM, and thus the work presented here, is built. With this understanding, we now introduce EBGM itself.

2.2 EBGGM

Elastic Bunch Graph Matching, as briefly introduced in the previous chapter, is a technique of extracting features from a novel face image by fitting a bunch graph over the image and localizing its nodes onto the image’s features. With the nodes accurately placed, the Gabor jet extracted at each point provides a concise and accurate description of that feature. The resulting feature vectors can then be matched against a known gallery with a distance metric, and a nearest-neighbor rule selects the best match, tests it against a rejection threshold, and completes the face recognition identification task.

The technique can be decomposed into several phases. First, the bunch graph is built from a set of sample images with specific features labeled. The jets at the labeled points are taken as examples and combined into bunches at each graph node. This is performed once for a given type of input image (say, a full frontal view of a human face) and the bunch graph is valid for any novel image that falls into that category. Then, for each novel image given to the system, the graph is fitted initially over the image by using some known anchor (say, eye location) and making use of the average lengths of graph edges to estimate feature locations to a first degree. Localization takes this rough estimate and performs an optimization on each feature node. The localization procedure will find the jet in the bunch that best matches the jet at the estimated point in the novel image. Then, using displacement estimation, it will infer a more accurate location based on phase differences, and possibly repeat this step and/or an exhaustive search over the neighborhood of the point to find the best match to the bunch jet according to a phase-insensitive similarity measure. Given the novel jets that are most representative of the canonical feature represented by each node in the graph, we have a feature vector for the novel image and we can add this to a gallery or evaluate a distance metric against another image in the gallery. We will cover each of these steps in turn.

2.2.1 Bunch Graph and System Initialization

The bunch graph is initialized by taking a set of images with pre-labeled features and extracting jets from each of the features, building a single master graph – the “bunch graph” – from these jets. In theory, it would be possible to optimize the set of jets retained at each node to best represent the variations in that feature and to best capture the uniqueness of that feature relative to

its surroundings. In practice, it is sufficient to retain the jets from all pre-labeled images, so long as that set is relatively small. The implementation of EBGM used in this work was shipped with a set of fiducial point placements on 70 randomly-chosen FERET images. In order to maintain experimental control, we maintain this training-set size and specify that bunch graphs will have 70 jets per bunch.

Fiducial point placement in these pre-constructed face graphs is nominally a manual process, and the core of this thesis’s original work is the replacement of this manual process with an automatic algorithm based on the information content at different points. Much more will be said on this subject in subsequent chapters.

2.2.2 The Face Graph

Before progressing too much further, we formally define a face graph. A face graph is an undirected graph G with a Gabor jet attached to each node (or, in the case of a bunch graph, a bunch of jets) and a length label on each edge. Each node is given a point location in the 2D space of the face image, corresponding to the location of the feature represented by that node. The length labels give standard Euclidian distance. In the case of a bunch graph, a length label indicates the average distance (in normalized pixels) between its endpoint nodes over all pre-labeled face images that went into the bunch graph. A face graph is not necessarily planar, and in the usual case, nodes that are nearer to each other are more likely to be connected by an edge. The graphs tend to be constructed so that the edges form a robust structure, able to flex in the localization phase as faces contort away from the canonical average face yet able to make a reasonable first guess in locating features before the localization phase.

A sample face graph is presented in Figure 2.3, overlaid on its corresponding face image. This face graph comes from the set of automatically-generated graphs produced by the fiducial point algorithm discussed later in this thesis.

2.2.3 Distance Metrics

It is necessary to define a function on a pair of face graphs that evaluates the distance – or, equivalently, the similarity – between the two graphs. We define these first as functions between two jets, and then graph similarity can

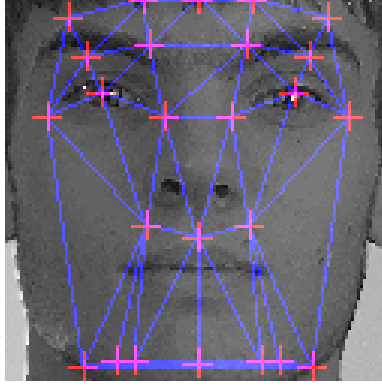


Figure 2.3: A face graph overlaid on corresponding face image

be composed of the more fundamental function applied on pairs of equivalent jets between two face graphs. As well, the localization procedure requires a similarity function on an individual jet in order to find the best match for a bunch jet in a novel image.

We define two similarity metrics on jets g and h , where the complex elements g_i, h_i are in polar form with magnitude $mag(g_i), mag(h_i)$ and angle $arg(g_i), arg(h_i)$. Both of these metrics are the standard EBGM definitions, as formulated in [2].

The first similarity metric is phase-insensitive and provides a less accurate value that is valid over a wider range:

$$S_a(g, h) = \frac{\sum_{i=1}^N mag(g_i)mag(h_i)}{\sqrt{\sum_{i=1}^N mag(g_i)^2 \sum_{i=1}^N mag(h_i)^2}}$$

This metric is useful in the first part of the localization phase, prior to displacement estimation and adjustment of node locations, during which the jet locations may not be very accurate but a rough measure is needed in order to select the best bunch jets.

The second similarity metric accounts for phase differences and provides a more accurate value that is valid over a smaller range:

$$S_\phi(g, h) = \frac{\sum_{i=1}^N mag(g_i)mag(h_i)\cos(arg(g_i) - arg(h_i))}{\sqrt{\sum_{i=1}^N mag(g_i)^2 \sum_{i=1}^N mag(h_i)^2}}$$

With both of these metrics in place, we can now explore the localization phase in detail.

2.2.4 Localization

Localization takes a graph that has been tentatively fitted over a novel image and makes use of jet bunches at each node to find the best-matching bunch jet and then shift the node in the novel image to best match this jet. This is the critical step for accurate performance; properly-performed localization on a good input image can sometimes achieve sub-pixel accuracy.

EBGM localizes features one at a time, and in two steps: selecting the best bunch jet, and then shifting the novel jet location. This circular dependency may at first seem tenuous: but note that we operate off the assumption that the initial placement of the graph is fairly good. This depends heavily on good normalization of the input images and placement of the anchor points (e.g., eye coordinates), and also on a bunch graph with average edge lengths that are fairly representative of both the novel images and the training set. In practice, this assumption tends to be valid, as demonstrated ultimately by the good matching performance achieved by EBGM.

Selection of the best bunch jet for a given novel jet is straightforward: best-match suffices, using the phase-insensitive similarity metric S_a defined above. With a sufficiently diverse bunch, and a well-placed graph, there should be at least one good match.

Estimation of displacement from true feature location is slightly more complex. Using the phase information from each Gabor wavelet convolution, and with knowledge of the wavelength and primary axis of each wavelet, we can estimate a linear displacement from the angular displacement in each jet element. This estimation is limited in maximal extent by the wavelength of the wavelet mask, since the phase shift will be periodic with the wavelength as we displace from true center. By incorporating each element of the jet, we have an accuracy up to the longest wavelength.

Concretely, such an estimation can be derived from the phase-aware similarity metric defined in the previous section. We wish to maximize this function over the displacement of the novel jet: that is, to maximize $S_\phi(g, h, \vec{d})$ where the third function argument is a vector that displaces h . If we assume that the magnitude of each complex element h_i varies little over a small displacement, then we can approximate the effect of the displacement by altering the phase angles only. By approximating the cosine term in the numerator with its second-order Taylor series expansion, one can set the derivative equal to zero and solve for \vec{d} .¹ This yields the following expression for performing

¹The details of this derivation can be found in the EBGM references; they are omitted

displacement estimation [2]:

$$d_x = \frac{1}{\Gamma_{xx}\Gamma_{yy} - \Gamma_{xy}\Gamma_{yx}}(\Gamma_{yy}\Phi_x - \Gamma_{yx}\Phi_y)$$

$$d_y = \frac{1}{\Gamma_{xx}\Gamma_{yy} - \Gamma_{xy}\Gamma_{yx}}(\Gamma_{xy}\Phi_x + \Gamma_{xx}\Phi_y)$$

where:

$$\Gamma_{ab} = \sum_{j=1}^N \text{mag}(g_j)\text{mag}(h_j)k_{ja}k_{jb}$$

$$\Phi_a = \sum_{j=1}^N \text{mag}(g_j)\text{mag}(h_j)k_{ja}(\text{arg}(g_j) - \text{arg}(h_j))$$

$$k_{jx} = \frac{2\pi}{\lambda_j}\cos(\theta_j)$$

$$k_{jy} = \frac{2\pi}{\lambda_j}\sin(\theta_j)$$

and λ_j, θ_j are the wavelength and orientation for the Gabor wavelet mask corresponding to jet element g_j .

Localization makes use of displacement estimation as defined here to step toward the true feature locations in a novel image. In practice, multiple techniques exist, ranging from a single displacement-estimation step to a more complete exhaustive search over the neighborhood of the initial placement to find the optimal match. The EBGM literature contains several comparisons of these methods, an analysis which is outside the scope of this work.

2.2.5 CSU EBGM

In the interests of reducing duplicated work, the research leading up to this thesis was performed using the Colorado State University Face Identification Evaluation System [1]. This is the system extended in [2] to support EBGM,

here for brevity and clarity.

and it has been developed explicitly to support repeatable results on a common framework using the FERET data. Given the FERET image set and the set of 70 images with labeled fiducial points, we run a script provided with the distribution to produce a distance matrix on all pairs of gallery images. We then run a simple script over that matrix to find the best match for each image and compute the rank-one matching score. The FIES system is thus a black box that takes our fiducial points and gives a quantified evaluation of matching performance. The distribution provides a baseline in its manual fiducial point placements. The framework is thus nicely laid out to provide a backend for the fiducial point experiments that form the remainder of this work. Details of FIES usage are provided in the *Technical Details* appendix.

Chapter 3

Information Content in Gabor Jets

Prior to the study of fiducial points per se, we must establish the ability of a particular point in a set of face images to yield good information about the identity of a face in the context of the dataset provided. We make the reasonable assumption that the ensemble of individually strong fiducial points will provide a strong face graph in the whole. We begin, then, by defining some measure of information content – “goodness” – at a given point.

First, we must be able to speak about “a point” in all images in the set simultaneously; as a convention, we will normalize face images by affine transformations so that the eye coordinates, which are known *a priori* from some trusted source, map to the same normalized coordinates in all images. Normalization also removes the DC component and scales the contrast to a nominal value. Then, a *point* is the set of Gabor jets at the given location in all normalized face images.

In order to determine the viability of a particular point to be a good candidate for fiducial point selection, we must model its ability to separate distinct subjects while recognizing the similarity of a novel image to prior images of the same subject. Note carefully that this criterion does not necessarily require that points correspond to face features as a human observer might define them – nor does it necessarily produce such points in all cases. Concretely, as will be seen when results are presented, the map of jet goodness over the normalized face space gives a recognizable face shape but has surprising peaks and valleys. The objectivity of the information-content ap-

proach is an important and intentional characteristic as we search to replace a human-labor process with an automatic algorithm.

3.1 Variance-based Goodness Measure

Fundamentally, to find the goodness of a trial point, we want to quantify the separation of clusters normalized over the variance within a single cluster. Our goodness measure is, intuitively, the average distance in jet-space of distinct subjects in terms of the standard deviation of a single subject’s jet at the trial point.

Given the set of jets $J[i]$ extracted from each face image at the trial point, and given subject numbers s_i where two jets have the same subject numbers if they come from face images of the same human subject, we quantify the goodness of the trial point as follows:

$$\begin{aligned}
 F(J[i], s_i) &= \frac{\frac{1}{|E|} \sum_{(i,j) \in E} |J[i] - C[j]|^2}{\frac{1}{|I|} \sum_{(i,j) \in I} |J[i] - C[j]|^2} \\
 C[i] &= \frac{1}{|S_i|} \sum_{j \in S_i} J[j] \\
 E &= \{(i, j) | s_i \neq s_j\} \\
 I &= \{(i, j) | s_i = s_j\} \\
 S_k &= \{i | s_i = k\}
 \end{aligned}$$

We first compute the centers $C[i]$ of the jets for each subject, and then compute two sums-of-squares, the numerator running over extra-subject pairs and the denominator running over intra-subject pairs.

This measure is similar in idea to a signal-to-noise ratio: it quantifies how much separation exists between a positive match and a negative match. We are now ready to evaluate this measure over all points in face space and then pick our fiducial points accordingly.

3.2 Face Space

We define face space as a two-dimensional image space into which our face images are mapped through a particular normalization process. Specifically, face space is a 128×128 -pixel discrete space with continuous grayscale intensities at each pixel. In line with traditional 2D computer graphics coordinate systems, we point the Y-axis downward so that the top row corresponds to $y = 0$ and the bottom row to $y = 127$. Each face image is mapped through an affine linear transform so that all eyes coincide at predefined coordinates: (32, 32) and (96, 32) for the left and right eye, respectively. The CSU FIES system provides manually-determined eye coordinates for each FERET image; as such, our system need not provide some additional means to locate the eyes, but rather simply use this list. (In practical application, one of any number of reliable eye-finding algorithms could be employed.)

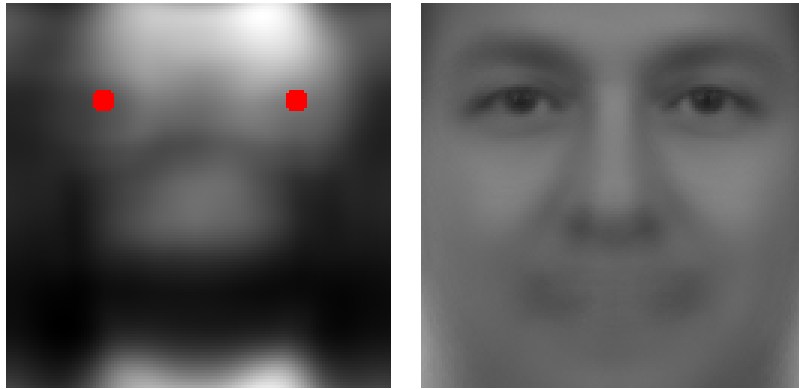
Of particular note is that the CSU system utilizes an eye-coordinate normalization that does not correspond to ours. CSU places the eyes at (52, 64) and (76, 64), zooming out by a factor of $\frac{8}{3}$ and translating the face downward to expose more of the forehead and hair. Our coordinate system was chosen to allow for more detail in the critical regions of the face; because we renormalize the coordinates prior to transfer to the CSU system, this difference is of limited significance except that it perhaps gives us slightly more accurate placements.

Once the face images are mapped to face space, they undergo a spatial-domain normalization before jet convolutions can take place. The average face is computed and subtracted from all faces in order to avoid the possibility of a strong DC component washing out the meaningful face variations. Contrast is then normalized on a per-image basis to compensate for lighting variation during image acquisition.

3.3 Face Space Results

Given a uniform face space, we can evaluate the information content function F at each point in the 2D space. Results for the FERET dataset are displayed in Figure 3.1. Only subjects with at least five face images were included in the dataset so that the intra-cluster variance would be meaningful. The average face is shown next to the variance map for comparison.

With this measure of information content of a particular point in the



(a) Inter/Intra-Variance Measure on Face Data (b) Average Face on Same Data

Figure 3.1: Variance Metric

image set, we are ready to proceed to the automated choice of the fiducial points themselves.

Chapter 4

Fiducial Point Selection

We now tackle the primary problem that we aim to solve in this work: given a map of information content across face space, how can we choose fiducial points automatically? We aim to replace the manually-labeled fiducial points on each of the bunch graph initialization images with the fiducial points we choose – and so essentially, we are choosing the points that will define the feature vectors used by EBGM recognition.

The key observation is that the information content function varies slowly and forms regions of high value. This confirms the expectation that the face will have certain features that show more uniqueness between subjects and other characteristics that are fairly homogenous across all subjects, and gives credence to a feature-based approach, over against a pessimistic alternative in which no single point or feature by itself provides significant discrimination ability. Indeed, by taking the most powerful individual features and fusing them into a face graph, we can obtain competitive matching results.

We can treat the information content map as a density map. We treat the point-placement problem as an iterative random point placement, using the map as a probability distribution. Each point placement alters the probability map with a Gaussian decay around the placed point, so that points will not cluster tightly around peaks but spread more evenly. This, with a few other heuristics – particularly, a symmetry rule taking advantage of frontal face symmetry – forms the essential core of the work comprising this thesis.

4.1 What is a Fiducial Point?

A fiducial point, precisely speaking, is a point in face space that we have chosen as a node in the face graph. The fiducial points in face space are mapped through inverse eye-alignment transforms to coordinates in the original images in the bunch-graph initialization set. These points are then given to the CSU EBGM system in order to build the bunch graph. Later, we add a postprocessing phase called “prelocalization” that performs EBGM localization on the fiducial points within each image using pseudo-bunches of jets, in order that the final bunches be drawn from more accurately-aligned labels in the image set. This will be covered in detail later.

4.2 Regions of Information Content

Examining the information content map shown in Figure 4.1, we see clearly that there are several roughly-defined regions surrounded by black void. We would like to capture these regions by spreading fiducial points throughout them, with a slight preference for placing points nearer to the edge of the region. We can quantify this with a density function:

$$\begin{aligned} p(x, y) &= F(x, y)G(x, y) \\ G(x, y) &= G_x(x, y)^2 + G_y(x, y)^2 \end{aligned}$$

where $F(x, y)$ is the information content map, and $G_x(x, y)$ is the result of convolving a 3×3 Sobel kernel oriented along the x -axis over $F(x, y)$, and similarly for $G_y(x, y)$. After calculating $p(x, y)$ over face space, we normalize it so that it sums to 1, forming a probability density function. The probability map given by this is shown in Figure 4.2.

We still have not incorporated some measure to account for previous points’ placement. Intuitively, a previous point placed near a candidate point lessens the appeal of that candidate point: as a Gabor jet captures information about its neighborhood, close clusters of fiducial points yield little advantage. We thus modify the probability density as follows:

$$p(x, y) = F(x, y)G(x, y) \prod_i (1 - e^{-\frac{r_i^2}{2\sigma^2}})$$

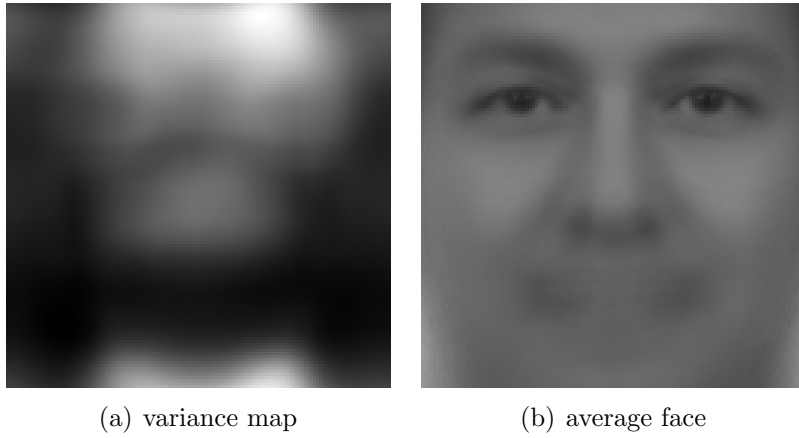


Figure 4.1: Information content map over face space

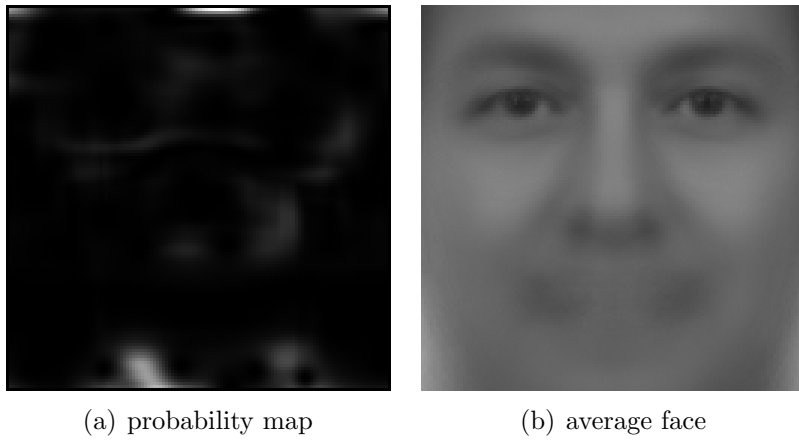


Figure 4.2: Probability density over face space

where the terms r_i give distances to previously-placed points in face space, and the parameter σ is set to an appropriate decay factor (in practice, $10\sqrt{2}$ worked well).

There is another constraint that we specify. Frontal face images tend to be symmetrical about the vertical centerline. Taking this as a hard constraint yields a dual advantage: first, point placement happens twice as quickly, since placing one point determines another; and second, we guarantee a symmetric face graph, which will yield better matching performance. In actuality, the constraint is slightly more complicated: if a placed point is more than 10 pixels from the centerline, we place the next point at its corresponding mirror point. Otherwise, we snap the point exactly to the centerline, in order to avoid tight clustering. The disadvantage to this approach is that we deal only with frontal face images; there is less allowance for pose variation. However, the constraint is easily removed in order to deal with non-symmetric image classes.

Finally, the first two points placed are always the fixed eye coordinates – because we are given accurate coordinates from an external source, it is prudent to make use of this information in fiducial point placement, as it is likely to be at least slightly better than an automatically-placed point that we determine.

The main fiducial point placement loop is given in Algorithm 1.

The implementation of $random(p)$ may be trivial: in practice, we simply pick a random number in $[0, 1)$ and scan the matrix in row-major order, decrementing the value by each cell’s probability until it becomes negative. A more efficient ($O(\log n)$ rather than $O(n)$) implementation would be possible if we precomputed the cumulative distribution function of p once and performed a binary search for each point placement; however, it is important to note that the probability density function changes by the additional spatial Gaussian decays after each point placement.

4.3 Edge Structure by Triangulation

We have fiducial point locations, but a bunch graph must also specify the edge structure that allows for approximate first placement in the EBGGM localization phase. Rather than attempt to develop a rule for edge placement based on some aspect of information content, or closeness or connectedness of fiducial points, we invoke a predefined algorithm to find edges. Specifically,

Algorithm 1 Fiducial point placement algorithm

Require: N {number of points to place}

Require: $p[x][y]$ {probability density map}

Require: $(x_1, x_1), (x_2, y_2)$ {eye locations}

Ensure: list of (x, y)

$list \leftarrow []$

$append(list, (x_1, y_1))$

$append(list, (x_2, y_2))$

for $i = 1$ to $N - 2$ **do**

$x, y \leftarrow random(p)$

if $|y - 64| \leq 10$ **then**

$y \leftarrow 64$

else

force next random point to $(x, 128 - y)$

end if

$append(list, (x, y))$

for $X = 1$ to 128 **do**

for $Y = 1$ to 128 **do**

$dist \leftarrow (X - x)^2 + (Y - y)^2$

$p[x][y] \leftarrow p[x][y] * (1 - e^{-\frac{dist}{2\sigma}})$

end for

end for

end for

we make use of Delaunay triangulation [4], and we use an implementation of this algorithm by Geoff Leach of RMIT, Australia¹.

The Delaunay triangulation of a set of points is the dual graph of the Voronoi diagram. The Voronoi diagram, in turn, is a partitioning of the plane into one region for each input point such that for all points in a given region, the closest point in the input set is the input point contained that region. The Delaunay triangulation is formed by connecting two input points with an edge iff the regions for those two points share a border.

Intuitively, the Delaunay triangulation connects points that are nearby in a planar graph that does a good job of providing robust connectivity. In particular, this triangulation minimizes angles between edges, allowing a graph more flexibility as it is localized to a particular novel image. A sample of Delaunay triangulation on an early version of the fiducial point output is given in Figure 4.3.

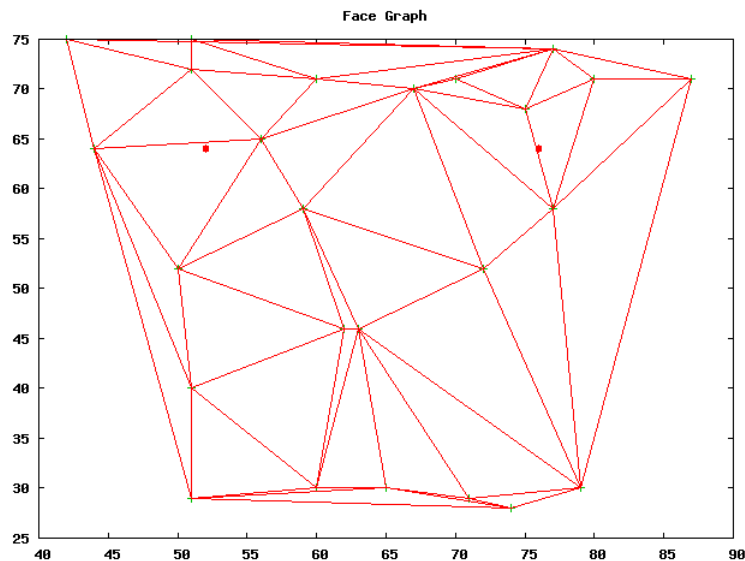


Figure 4.3: Delaunay triangulation of an early face graph

With a set of fiducial points and edges connecting them, we are ready to specialize the generic face graph for each image in the bunch graph initialization list.

¹Available at <http://goanna.cs.rmit.edu.au/~gl/research/comp-geom/delaunay/delaunay.html>.

Chapter 5

Prelocalization

The fiducial point placement process described in the previous chapter places a set of points at identical locations in all normalized face images. While this should be sufficient for matching – as evidenced by the results in [3], in which an immobile grid of points was placed over all images – we attempt to do better by tuning the point locations in each image. Using wavelet phase information, similar to the localization phase of EBGM, we perform “prelocalization” on the fiducial points for each image.

5.1 Prelocalization: Optimizing Placement

The intent of fiducial point labels on the initial image list is to provide accurate training to the system so that the bunch graph can make fine distinctions and successfully localize to single-pixel precision. Using a single generic face graph constructed by the process above, mapped back to the original coordinates in each image, will not provide sufficient training in fine details for this to happen. Thus we establish another phase of processing, after fiducial point selection, to localize the fiducial point labels to each image.

EBGM localization makes use of phase information in the wavelet filter responses to tune the jet locations precisely. We take advantage of the fact that we have two similarity metrics for a given pair of jets: one that incorporates phase, and so falls off sharply as a jet becomes misaligned, and another that incorporates only the magnitude of the responses for each wavelet filter, thus performing well with approximate location. Localization requires an accurate jet bunch: it works by selecting the jet in the bunch that most

closely matches the approximate location, using the phase-insensitive metric, and then attempting to match this canonical feature description as closely as possible by adjusting the jet location in the novel image, driven by the phase-sensitive similarity function.

5.2 Jet Pseudo-Bunch Construction

Prelocalization makes use of the displacement estimation formula to shift fiducial point labels. In order to do this, we need a jet bunch from which to choose the best-matching canonical feature jet. We form this pseudo-bunch from the jets extracted at the initial fiducial point placement, using K-means clustering to find an optimal set of cluster centers that will form the bunch. Prelocalization then proceeds as EBGM localization would, picking the best match out of the bunch by the phase-insensitive similarity measure and then using the phase-dependent displacement estimation formula to shift the point.

The pseudo-bunch clustering is necessary for a readily-apparent reason: if we did not reduce the set of all jets to a smaller set of cluster centers, then the best match for each jet would be itself, and there would be no displacement. We must simulate having a canonical knowledge of the particular feature, for lack of real outside knowledge from manual training, by making a reasonable guess. We run K-means clustering on the jets with K running from 2 to $\frac{N}{10}$. For each clustering, we take the inter-cluster variance over intra-cluster variance, similar to prior-discussed metrics, and then we pick the clustering with the best inter-cluster separation. In practice, we obtain K from 3 to 6. The centers of these clusters then form the pseudo-bunch. The algorithm is detailed in Algorithm 2.

5.2.1 K-means clustering

The K-means algorithm was first described by J. MacQueen in [5], and is a remarkably simple solution to the clustering problem. Given an arbitrary initial clustering into K classes, the algorithm alternates between recomputing the class centers and then recomputing the classifications until the classifications converge (i.e., do not change between iterations). We give the formal algorithm in Algorithm 3.

The running time of this algorithm, for a given fixed K , is $O(N)$, by the

Algorithm 2 Pseudo-bunch selection algorithm

Require: $points[N]$, $N > 0$

Ensure: $centers[K]$, K small

$bestGoodness \leftarrow 0$

$bestClasses \leftarrow null$

for $K = 2$ to $n/10$ **do**

$classes, centers \leftarrow Kmeans(points, K, 100)$

$interVariance \leftarrow 0$

$intraVariance \leftarrow 0$

for $i = 1$ to N **do**

$minDist \leftarrow \infty$

for $j = 1$ to K **do**

if $j = classes[i]$ **then**

continue

end if

$dist = euclidianDistSquared(points[i], centers[classes[j]])$

if $dist < minDist$ **then**

$minDist \leftarrow dist$

end if

end for

$interVariance \leftarrow interVariance + minDist$

$intraVariance \leftarrow intraVariance +$

$euclidianDistSquared(points[i], centers[classes[i]])$

end for

$goodness \leftarrow interVariance/intraVariance$

if $goodness > bestGoodness$ **then**

$bestGoodness \leftarrow goodness$

$bestClasses \leftarrow classes$

end if

end for

Algorithm 3 K-means clustering algorithm

Require: $points[N]$, $N > 0$ {set of input points}

Require: $K > 0$ {clustering into K subsets}

Require: $maxIters > 0$ {an upper limit on the number of iterations}

Ensure: $points[N]$ {final cluster assignments}

$centers \leftarrow$ list[K] of float

$count \leftarrow$ list[K] of integer

$classes \leftarrow$ list[N] of integer

$newClasses \leftarrow$ list[N] of integer

for $i = 1$ to N **do**

$classes[i] \leftarrow$ random($1...K$)

end for

for $i = 1$ to $maxIter$ **do**

for $j < K$ **do**

$centers[j] = 0$

$count[j] = 0$

end for

for $j < N$ **do** {compute cluster centers}

$centers[classes[j]] \leftarrow centers[classes[j]] + points[j]$

$counts[classes[j]] \leftarrow counts[classes[j]] + 1$

end for

for $j < K$ **do** {normalize averages}

$centers[j] \leftarrow centers[j]/count[j]$

end for

for $j < N$ **do** {re-assign clustering}

$newClasses[j] \leftarrow$ closest($points[j], centers$)

end for

if $classes = newClasses$ **then** {complete if clusters have converged}

 break

end if

$classes \leftarrow newClasses$

end for

simple observation that the number of iterations is bounded by a constant and the work per iteration scales linearly in N for both array recomputations. Linear time is an essential benefit to K-means for any sizable dataset.

5.2.2 Displacement Estimation from Phase

Given pseudo-bunches extracted from a generic point placement on a set of face images, we can make use of the EBGm displacement-estimation formulae to prelocalize the point placement on each image. We rely on the assumption that the initial placement is accurate enough, and the clustering approach is sufficient, to produce accurate pseudo-bunches that represent the feature in question. If the pseudo-bunch is not accurate (in the worst case, simply a collection of unrelated Gabor jets from random points on faces) then displacement estimation will be, at best, meaningless, and at worst, harmful. Regardless, we attempt to perform displacement estimation here.

Recall the displacement-estimation formulae, reproduced here from an earlier chapter:

$$d_x = \frac{1}{\Gamma_{xx}\Gamma_{yy} - \Gamma_{xy}\Gamma_{yx}}(\Gamma_{yy}\Phi_x - \Gamma_{yx}\Phi_y)$$

$$d_y = \frac{1}{\Gamma_{xx}\Gamma_{yy} - \Gamma_{xy}\Gamma_{yx}}(\Gamma_{xy}\Phi_x + \Gamma_{xx}\Phi_y)$$

where:

$$\Gamma_{ab} = \sum_{j=1}^N \text{mag}(g_j)\text{mag}(h_j)k_{ja}k_{jb}$$

$$\Phi_a = \sum_{j=1}^N \text{mag}(g_j)\text{mag}(h_j)k_{ja}(\text{arg}(g_j) - \text{arg}(h_j))$$

$$k_{jx} = \frac{2\pi}{\lambda_j}\cos(\theta_j)$$

$$k_{jy} = \frac{2\pi}{\lambda_j}\sin(\theta_j)$$

and λ_j, θ_j are the wavelength and orientation for the Gabor wavelet mask corresponding to jet element g_j .

The generic formulae make use of Gabor jets with filter responses g_i and h_i , and in practice, an EBGM implementation will often build a sophisticated framework of iterative optimization algorithms around this core; for our purposes, because we have no guarantee of ultimately accurate pseudo-bunches, we take only a single iteration of the displacement estimation. We take one jet to be the best match (by the phase-insensitive measure) in the pseudo-bunch, and the other jet to be the generic point placement on the novel image. We then displace the generic point by the predicted amount. We have no a-priori knowledge of feature locations (except for the eyes, which we do not displace), and so this is the best that we can do.

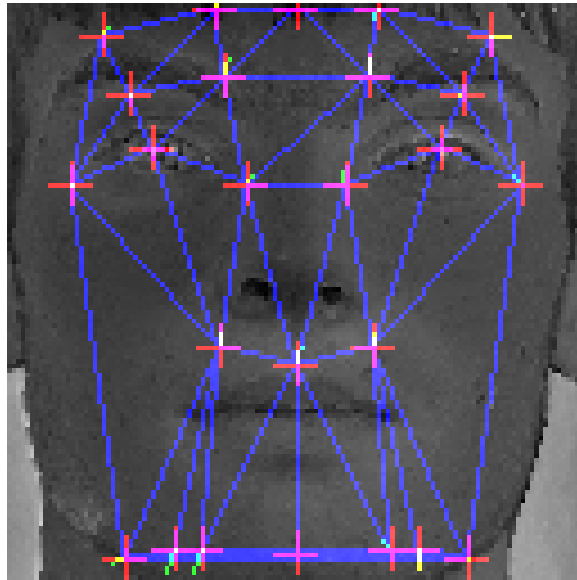
5.2.3 Prelocalization Results and Conclusions

We will present EBGM matching algorithm results using prelocalized fiducial points in the following chapter. However, it is instructive to examine prelocalization by itself to determine whether the accuracy is acceptable subjectively.

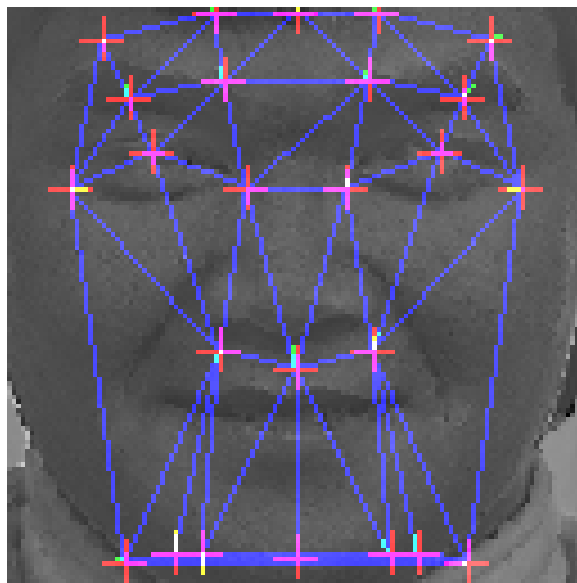
Objectively, we displace fiducial points by a mean of 2.628 pixels over the 70-image training subset of FERET. The ground-truth manual placements provided with the CSU EBGM implementation have an average displacement of 2.021 pixels from the average in each point.

Two prelocalization displacement plots are presented in Figure 5.1. We note that the displacements are tiny – in fact, very difficult to see at first. Most changes happened at the chin-line points and the outer eyebrow points, from empirical inspection of results; in particular, in the second image displayed, the displacements brought the lower points into line and pulled the eyebrow points closer to the irises, in conformity with most of the other images in the training set. Subjectively, then, the results appear reasonable.

Nevertheless, as will be shown in the following chapter, it was found that prelocalized face graphs in the training set led to worse EBGM matching performance than the graphs that did not undergo the procedure. Our hypothesis, faced with this data, is that the prelocalization has no a-priori ground truth knowledge of face features, and so given this, it is impossible to choose more precise jet locations than those that were chosen based on the statistical fiducial point goodness measure. Most likely, the pseudo-bunches were collections of disparate features rather than a single cohesive set; work might be done to perform more intelligent clustering, cognizant of the goal to unify a set of sample features into the one canonical feature that the pseudo-bunch defines. But without significant study, we do not have a way



(a) FERET image 00049fa001d_931230



(b) FERET image 00096fa010_940128

Figure 5.1: Displacement-estimation plots: green lines trace displacements

to improve this performance.

Chapter 6

FERET Matching Results

Performance of the system with the automatically selected fiducial point labels is comparable to performance with the original manual placements. It was determined that the bunch graph resulting from our system’s output without prelocalization performs better than with prelocalization; this is likely due to an inability of displacement estimation to perform well without a better bunch jet, indicating a shortcoming in the clustering approach used to determine the pseudo-bunch. Nevertheless, we consider this work to have provided an affirmative answer to the initial quest: to replace the manual fiducial point placements with an automatic process.

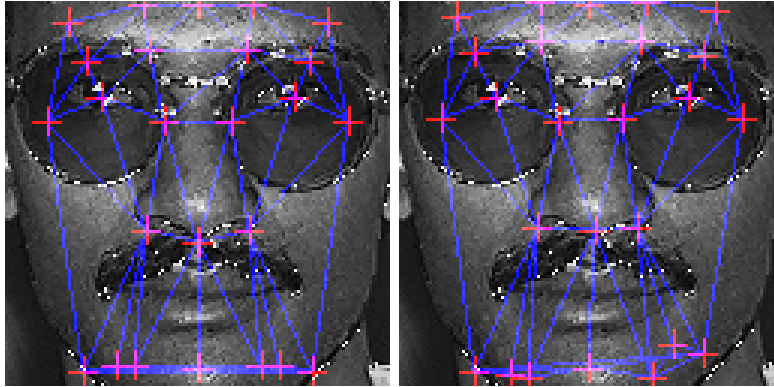
We note, additionally, that nothing in this system depends on the fact that we process a face. We require only two anchor points, such as the eyes, on the input images. We thus conjecture that the automatic system would make an extension to another face pose, or even another class of object entirely, trivial relative to the effort that a face-specific system might require. We leave this for further work.

6.1 Sample Face Graphs

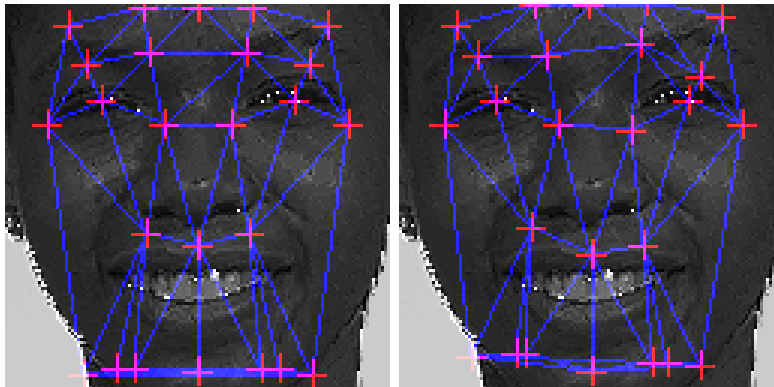
We present three face graphs, before and after prelocalization, in Figure 6.1.

6.2 Experimental Setup

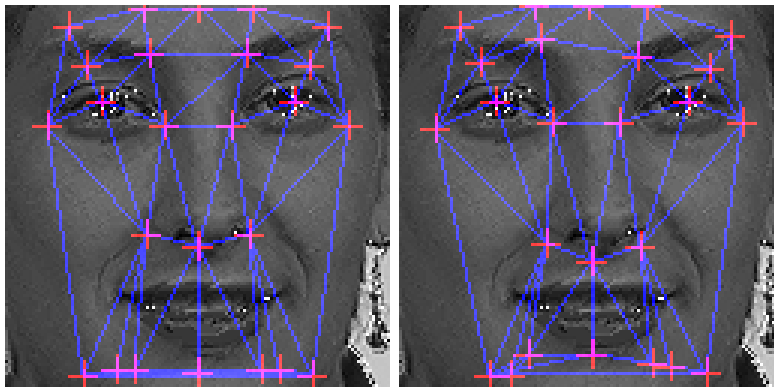
Fundamentally, we experiment on a single variable: the fiducial point labels given to the EBG system. The dataset, the system itself and its parameters



(a) FERET image 00003fa010_930831



(b) FERET image 00654fa010_941031



(c) FERET image 00655fa010_941031

Figure 6.1: Face graphs before and after prelocalization

are unchanged between runs. An EBGGM run produces a distance matrix from which we can extract a rank-one matching score, indicating the fraction of face images for which the best match (other than itself) is another image of the same subject.

Additionally, we place some constraints on the fiducial point face graphs that we pass to the EBGGM system. Because the original manual placements give a 25-node graph, we maintain this node count. We provide the graphs for exactly the same set of randomly-chosen FERET images for which the original placements were produced. Thus the only difference between runs is in the face graph itself for each image.

We compare three EBGGM runs: with original fiducial points, with automatic points without prelocalization, and with automatic points with prelocalization.

6.3 Recognition Rate Results: Full FERET and Statistical Permutations

We achieve the following rank-one recognition rates taking the entire FERET database as the gallery and probe set:

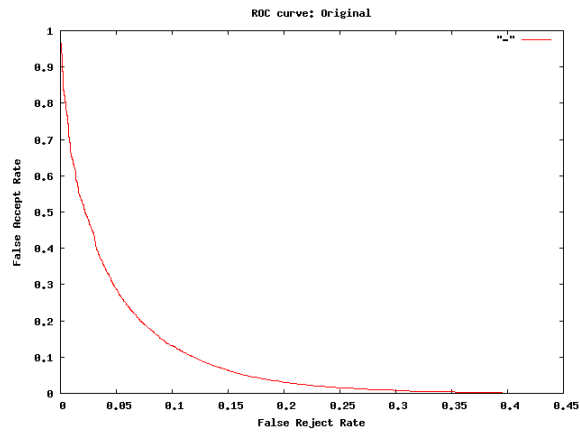
Original	82.16%
Without Prelocalization	81.83%
With Prelocalization	78.47%

ROC curves for each of the three cases are presented in Figure 6.2. The equal error rates for each case are: original, 11.1%; without prelocalization, 11.4%; with prelocalization, 11.9%.

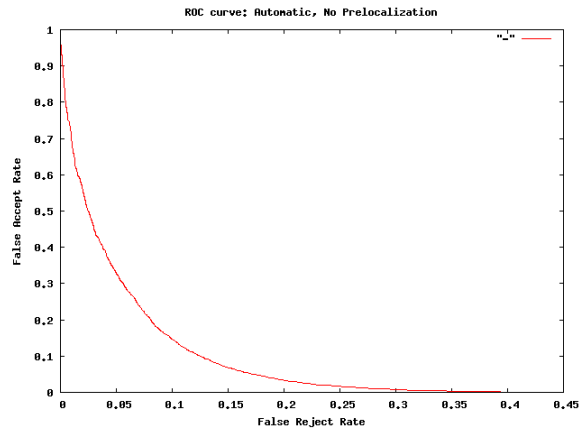
Of interest is that prelocalization results in degraded performance. As conjectured above, this is likely due to poor pseudo-bunches used by the prelocalization displacement estimation process.

Nevertheless, the automatically chosen fiducial points yield a degradation of less than one percent relative to the original manually-chosen points, nearly equivalent performance.

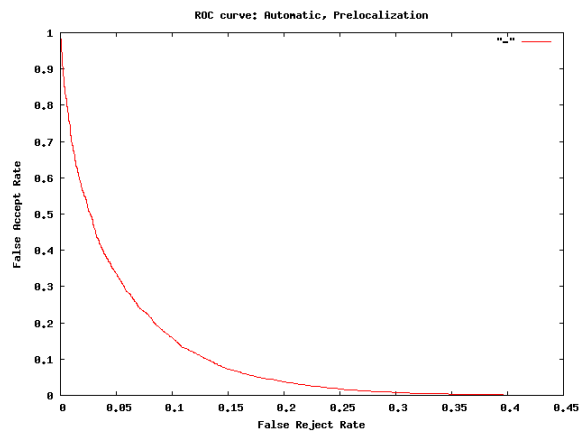
We note, incidentally, that the prelocalization rank-one matching rate is very slightly lower than that achieved by the fixed-location Gabor jet experiment presented in Chapter 1, namely, 78.52%. The fact that prelocalization does worse than fixed jets indicates that the prelocalized bunches fail com-



(a) Original



(b) Without Prelocalization



(c) With Prelocalization

Figure 6.2: ROC curves on FERET dataset

pletely at localization in EBGM recognition – that is, that there is no benefit afforded by finding a specific feature in a novel image, and instead we are degraded to taking jets at random locations.

We also made use of an experiment provided with the CSU EBGM system, namely, the permute experiment, described in detail in [2]. This experiment runs a certain number of iterations of a rank-one matching calculation over randomly-chosen gallery and probe subsets of FERET, and models the rank-one rate as a random variable with normal distribution. Given two or more such models, for different matching score matrices (i.e., produced from variations in the training fiducial points), one can find the probability that one system is objectively better than the other. The Permute experiment was run with three sets of results: original fiducial points, fiducial points without prelocalization, and fiducial points with prelocalization. The comparison probabilities are as follows:

Algorithm A	Algorithm B	$P(A > B)$
original	no preloc	0.7834
original	preloc	0.8276
no preloc	preloc	0.5204

We see here that the comparison is less favorable than the rank-one results over the full FERET dataset. Nevertheless, this data simply gives the probability that the original system, with manual fiducial points, is better than the system with automatic fiducial points; the full rank-one data above show that it is not much better. Interestingly, the comparison of prelocalization to the no-prelocalization case is very close to 50%, or neutral.

6.4 Further Work

Most generally, there is ample opportunity to further explore prelocalization. While the initial fiducial point placement algorithm has been shown to provide a very good foundation for automatic fiducial point selection, prelocalization has the opportunity to fine-tune these selections on an individual basis in order to provide better performance than the original hand-placed points. While it is possible that the 82.16% rank-one matching rate achieved by the original system is an upper limit for the CSU EBGM implementation in its present state, it is more likely that our automatic system should be

able to derive insight from the statistical measures we have developed that the human point-placers might have overlooked. Certainly, the first step to doing this is to examine in detail the prelocalization displacements on each FERET image and draw some general conclusions; for lack of time on this project, no comprehensive study was done at this detailed a level.

The clearest path to better prelocalization is to a better pseudo-bunch and then run displacement estimation based on that. More specifically, it might be possible to develop some heuristics that work well, or even to perform a limited exhaustive search in the neighborhood of the fiducial point on each image. The goal in prelocalization is to shift the fiducial point per image such that the extracted jets are similar, or at least can be lumped into several clusters that are internally similar, an indication that they all describe the same face feature. We find here a general optimization problem that could potentially yield additional research opportunity.

The graph edge structure was not given much thought in this implementation; Delaunay triangulation was found to be generally sufficient. If one were hoping to optimize performance better, it might be worthwhile to study the performance of CSU EBGGM localization in more detail and determine how good an initial fit the graph edges yield, and then see if a general principle emerges as to which edges are helpful and which are not. Again, much more thought needs to go into this path before a clear improvement emerges.

On a more practical note, it might be generally beneficial to integrate more closely with the CSU codebase. Though the interface is cleanly defined in our case, as we provide only the set of fiducial points in a well-defined file format (one file per FERET image), the process of transferring the points and running the CSU code is cumbersome. As well, it would pay to Condorize the CSU system – while the multi-hour information content function computation is split into 128 Condor jobs that take about five minutes apiece, a new run of CSU EBGGM takes about 1.5 hours on a dual-core Athlon 64 Linux system.

Finally – and perhaps most optimistically – the ability of the system to determine fiducial points automatically, without human intervention, opens the door to the use of arbitrary image classes, so long as we have at least two anchor points analogous to the eye coordinates. If the EBGGM system is capable of adaptation – and it should be, given that it makes no assumptions about face structure other than those implied by the bunch graph itself – then there is nothing stopping the combined systems from handling profile views of faces, photographs of buildings, biological identification tasks (such

as leaf species), or the like. This seems at first a bit far-fetched, but it is important to realize that what we have done is to remove the human-labor driven specialization of the system and replaced it with a relatively easy machine-driven specialization. It appears that the largest roadblock to exploratory research in this direction would be the availability of large image databases of arbitrary classes of objects in the way that face images are available to biometrics researchers.

Chapter 7

FRGC: Large-Dataset Optimization and Results

The initial draft of this thesis described work done with the FERET dataset, leading ultimately to the successful result presented in the previous chapter. With this success in mind, it was suggested to bring the tool set to a larger and more recent dataset, the Face Recognition Grand Challenge [6] dataset. This larger dataset consists of approximately 50,000 recordings, a combination of 2D and 3D images. For this work, we took the subset of subjects with at least five 2D images, and then for those with more than five, we randomly picked ten of that subject's recording. This yields 5404 images of 568 unique subjects; the compressed JPEG images comprise 7GB of data. In comparison, taking the same cutoffs (minimum five images per subject, maximum ten) on FERET, we have 619 images from 94 unique subjects. Such a large dataset as FRGC pushed the limits of the toolset constructed for the earlier work; as such, much of the challenge of the FRGC work proved to be in re-engineering the experiment flow to fit within computation time and space bounds.

The FRGC experiments presented in this chapter confirm the system's performance with FERET; in particular, the no-prelocalization automatic-points case yielded a rank-one score just below the rank-one score achieved by manual fiducial points, as seen with FERET, and prelocalization performed worse but only slightly.

7.1 FRGC Challenges

FRGC proved daunting initially due to the large dataset size. The matching score matrix computed by the CSU EBGM code scales as $O(n^2)$. Thus it was expected that the computation time would become unreasonable if not parallelized. However, in practice it seems that the linear-time portion of the EBGM matching algorithm (i.e., face graph localization and Gabor jet extraction) dominated the runtime, and so the computation time for the CSU EBGM portion remained reasonable and no parallelization was necessary.

More challenging was the adaptation of the fiducial point picker and prelocalization phases. The dataset no longer fits in main memory in its pre-normalized form; a special pre-normalization tool and a memory dump format was developed that is able to compress the dataset into a 700MB monolithic file. Each Condor job then reads this dump at startup. On the Notre Dame network, such a job can load data from AFS in about 20 minutes. In the process of optimizing the fiducial point picker, several other internal inefficiencies were identified and reworked; a set of 128 Condor jobs that would have been an overnight task was able to complete in about 2 - 3 hours on average.

In addition, FRGC fiducial point metadata was limited to eye coordinates; unlike FERET, we had no ground truth to serve as a baseline in experimentation. Fortunately, only the eye coordinates are required to normalize the images. In order to create a set of hand-picked face graphs analogous to those provided by the CSU EBGM system for FERET, a custom point-picker application was created using the OpenCV/Python framework. The resulting graphs, 70 in total with 25 fiducial points each, are available as part of the distribution of our system. The face features used in the FERET graphs were retained, as was the graph structure; the utility provides an sample face image and graph, highlighting the current point, and waits for the operator to click on the corresponding point in the novel image. A screenshot is shown in Figure 7.1.

Finally, the CSU EBGM codebase had to be adapted for use with FRGC, as the system had been built around FERET. Fortunately, this proved trivial: after some investigation, we determined that we need only provide an image list in the `.srt` format used by the toolset, and modify the run scripts to use the new image list and manual point placements. The remainder of the system made no FERET-specific assumptions. Details of the adaptation can be found in the Technical Appendix.

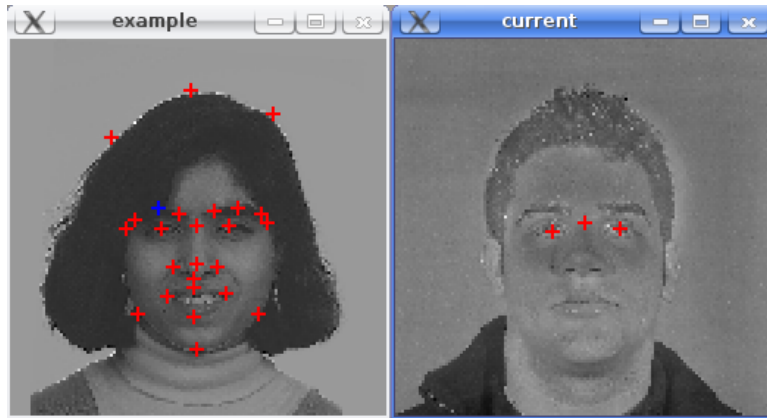


Figure 7.1: ptpicker application

7.2 FRGC Results

The FRGC probability map and resulting face graph are shown in Figure 7.5.

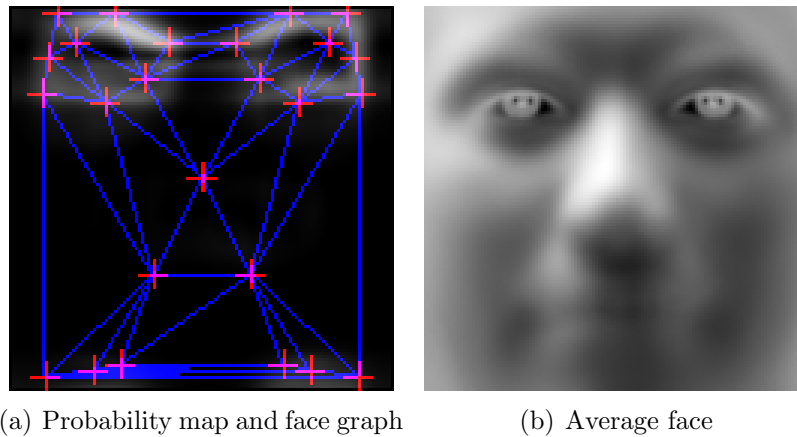


Figure 7.2: FRGC fiducial point picker results

We achieve the following rank-one rates with the FRGC image set:

Manual Points	76.85%
Automatic Points, No Prelocalization	75.80%
Automatic Points, Prelocalization	74.61%

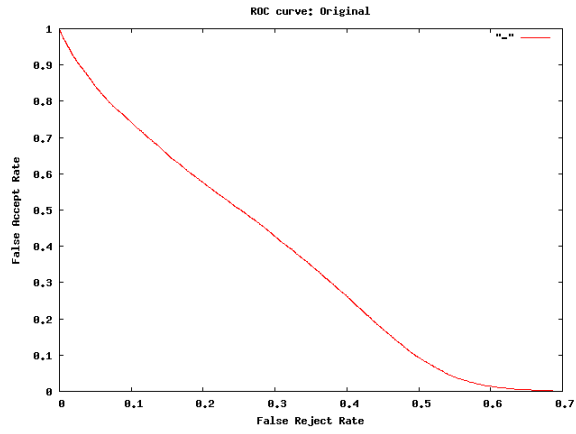
ROC curves are presented in Figure 7.3. The equal error rates are 34.8%

for the manual points, 31.4% for the automatic fiducial points without pre-localization and 31.4% for the automatic fiducial points with prelocalization.

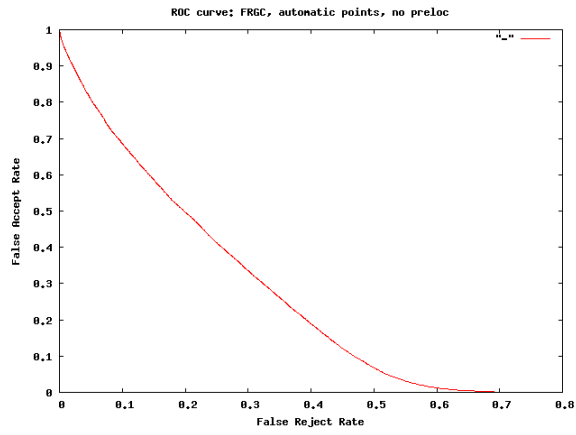
The average face computed from the FRGC dataset is shown in Figure 7.4, with the FERET average face for comparison. Likewise, the point-placement probability map and the resulting face graph are shown in Figure 7.5.

7.3 FRGC Future Work

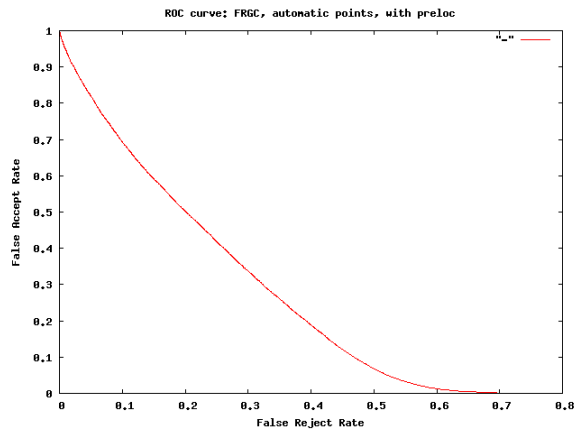
It might be instructive to take subsets of the FRGC dataset in order to attempt to obtain more uniform face features across the training set and thus improve performance. A demographic study on EBGMM might also provide interesting results on the universality of the example jets in the bunch graph; localizing a bunch graph trained on one demographic against gallery and probe sets drawn from another demographic would form an interesting experiment. The FRGC dataset is provided with extensive metadata on the subjects in various demographic dimensions.



(a) Manual Points



(b) Automatic Fiducial Points, No Prelocalization



(c) Automatic Fiducial Points, With Prelocalization

Figure 7.3: ROC curves for FRGC dataset performance

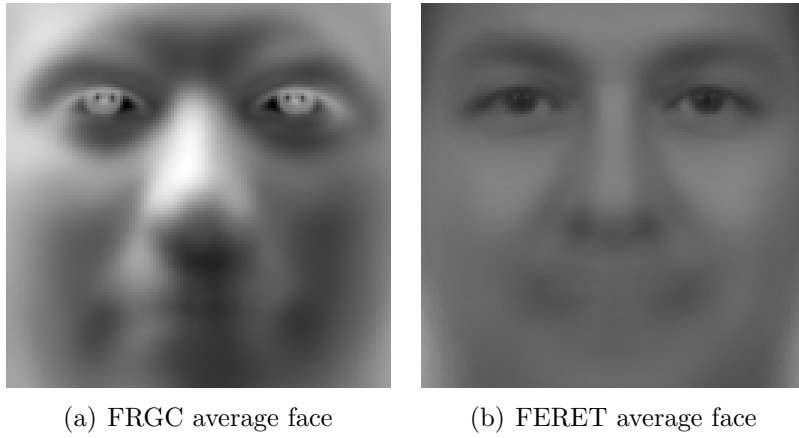


Figure 7.4: Average faces compared

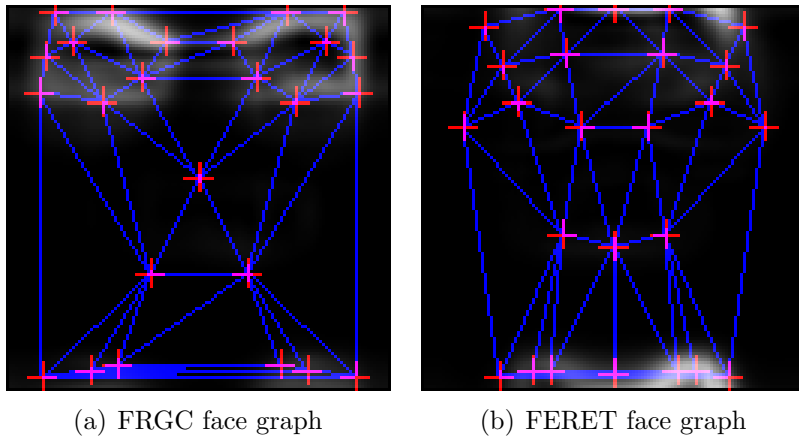


Figure 7.5: Face graphs compared

Appendix A

Technical Details

We aim to provide a guide to our system and to use of the CSU system specific to this research, and to do this in sufficient detail to allow third-party reproduction of our results. This appendix will additionally serve as a sort of technical note, sketching the overall architecture of the system and specifying particular implementation details.

A.1 The mark5 System

Our system is named “mark5,” for no other reason than that it is the fifth codebase produced as the author iterated through various formulations of and approaches to the problem.

The source is publicly available at <http://c1f.net/research/mark5/>, consisting of about 4300 lines of ANSI C. A Mercurial repository exists at <http://c1f.net/hg/mark5/>. The only external dependencies are the `expat` XML-parsing library, available by default on most modern Linux distributions, and `libjpeg`, also widely available. The system is built with the `jam` build system, a replacement for `make`, which is available as the `ftjam` variant at <http://freetype.sourceforge.net/jam/index.html>.

The source tree is organized in a straightforward manner: the `src/` directory contains system source, the `dct/` directory is a redistribution of Geoff Leach’s Delaunay triangulation tool, modified slightly to run on 64-bit systems, and `exp/` contains scripts, source data and results for experiments labeled by date and serial number (for example, 20081119-01 contains the

final rank-one matching scores and scripts to reproduce them, as well as final versions of the previous experiments that led directly to this final result). Each experiment directory additionally contains a file called `changeset` that indicates the Mercurial changeset/revision of `src/` used to run that experiment. Several of the older experiments make use of features in the system that have been modified or removed since the experiment was written.

In order to build the system, simply run “make” in `dct/` and “jam” in `src/`.

A.1.1 Datasets

The experiments packaged with the system depend on the FERET dataset, which is not freely redistributable and so must be obtained separately. Most biometrics research groups have access to this dataset. The original distribution was split by disc; `mark5` expects all `.pgm` files in one large directory. For historical reasons, this dataset is called the “eyecoords” dataset, the name originating from the fact that CSU FIES provides a list of eye coordinates for most of the FERET images.

In addition, `mark5` makes use of a very simple XML file format, the `<imageset>` schema, in order to list sets of images and eye coordinates for each one. The file `eyecoords.xml` is included in `exp/20081119-01`.

The experiment scripts in the system distribution expect `eyecoords.xml` as well as the image directory `eyecoords/` to be in `../datasets/` relative to the `mark5` root. If this assumption is broken, many scripts must be modified appropriately.

A.1.2 Information Content: Condor Run

Computation of the information content function is performed via a set of 128 Condor jobs, with all supporting scripts and files in `exp/20081119-01/condor/`. The script `submit.py` generates a Condor job-submission specification. It will be necessary to place the `eyecoords` dataset in a shared location (AFS or otherwise) accessible to all members of the Condor pool. The binary `run_split` (again named for historical reasons) performs the work in this stage and should be placed alongside the submit file. When all jobs complete, `run_split.out.*` are concatenated with `exp/20081005-01/concat.py` to form `run.log`, which is a textual form of the information content map. The

experiment 20081005-01 contains several auxiliary scripts to visualize this result.

A.1.3 Fiducial Points and Prelocalization

After the IC function is computed, we choose fiducial points based on the density map. This work is performed by two binaries, `run_fiducial` and `run_prelocalize`. The details have been wrapped in two corresponding shell scripts in `exp/20081119-01`: `fiducial.sh` and `prelocalize.sh`. The former requires the bundled Delaunay triangulation tool in `dct/` to be compiled. The latter script will produce four directories: `sfi`, `sfi.nodisp`, `sfi.csu`, and `sfi.csu.nodisp`. Each contains SFI files, one per FERET image in the `ebgm70.xml` list (corresponding to the set of images for which manual fiducial point labels were provided). An SFI file is a textual dump of a face graph, consisting of a list of nodes and then a list of edges between them. The first two directories also contain visualizations of these graphs overlaid on the corresponding images, produced by the `visualize` tool in `src/`. The `nodisp` variants have omitted prelocalization's displacement estimation step. The latter two – the `csu` variants – have had their coordinates renormalized to CSU eye-coordinate standards, for compatibility with the CSU FIES system. Once we have `sfi.csu` and `sfi.csu.nodisp`, we are ready to perform EBGM recognition runs to determine rank-one scores.

A.2 CSU EBGM

The CSU FIES system is quite complex, and capable of running quite a few experiments; we use it in a quite limited fashion.

First, we set up the system and preprocess the image set. Place the FERET images in `data/FERET/source/pgm/` and then run `scripts/EBGMPreprocessing.sh`.

The EBGM scripts expect the manually-labeled face graphs as SFI files in `ebgmdata/ManualLandmarkLocations/`. This directory, as shipped, contains the hand-placed points; copy it to a safe location before modifying the files.

In order to perform a run on, say, no-prelocalization fiducial points, copy `sfi.csu.nodisp/*.sfi` from the `mark5` tree into `ManualLandmarkLocations/`, and then run `scripts/EBGM_CSU.sh`. This will run for approximately 1.5 hours on a modern dual-core 64-bit machine, and will eventually produce a

complete distance matrix in `distances/feret/EBGM_CSU_PredictiveIter/`. The matrix is written as a sequence of SFI files, but different in format than the face graph dumps: each of these files is named after one FERET image, and each line contains the name of another FERET image followed by a matching score.

The script `rank1.py` in `exp/20081119-01/` has been written to process this distance matrix and calculate a rank-one matching score. Running the script within the matrix output directory will produce a real number between 0 and 1 indicating the fraction of images whose best match (excluding self) was with another image of the same subject. This score is our metric for the success of our fiducial point selections.

A.3 FRGC Additions

The addition of FRGC required two main efforts: a fiducial point picker to generate ground truth, and a modification of the fiducial point flow internals to handle a larger dataset. Nevertheless, the user experience is largely the same at the script-driven level.

The fiducial point picker, called simply `ptpicker`, is located in the `mark5` root. It takes a directory of pre-normalized `.pgm` images and uses an example image and point set to guide the user through fiducial point selections; when the session ends, it writes SFI files to the image directory in the format expected by the CSU EBGM code, using the graph structure (the edge list) from the sample graph. On startup, it will load any partial point selections that had been saved previously.

Some FRGC metadata has been created and is provided in the `frgc-compact` subdirectory within `mark5/datasets/`. The `frgc.xml` and `manual.xml` files are XML image lists (in the `eyecoords` format) for the entire subset of FRGC used, and the manual fiducial points (training) subset, respectively. The FRGC subset consists of images from all subjects with at least five images per subject; when more than ten images exist for an individual subject, a subset of ten is chosen randomly. A Python script, `frgc-copy.py`, is provided to copy the FRGC subset into a single directory.

Additional FRGC-related files are in `exp/20090301-01`; the `csu/` directory tree mirrors the structure of the CSU FIES EBGM system. SRT image list files, manual fiducial points and modified scripts to run the system on FRGC are provided.

The fiducial point tools have new frontends for FRGC use. The `flow_*` tools (`flow_normalize`, `flow_dump` and `flow_split`) are part of an optimized flow built around memory dumps of image sets. The first tool, `flow_normalize`, reads an ordinary `eyecoords`-format XML image list and produces a single large memory dump file containing the pre-normalized images, in order to facilitate quicker operation when many instances of Condor jobs start up from the same data set. The second, `flow_dump`, dumps such a file into a directory of PGM images – for debugging, or for use with the fiducial point picker, for example. Finally, `flow_split` is intended to be launched as a Condor job; it replaces `run_split` and is functionally identical except for its data input format.

The experiment 20090214-01 contains scripts to run three steps: `1-normalize.sh`, to produce the memory dump file; `2-condor/`, a directory with a submit file for Condor and a postprocessing Python script; and `3-fiducial.sh`, to run the fiducial point picker over the Condor-computed goodness map.

Use of the CSU EBGM system should be identical after the resulting fiducial point files are copied into place as before. The scripts `FRGC_*.sh` added to the scripts directory in the CSU EBGM tree will compute the scores matrix as before.

Finally, the utility 20090329-01/`roc.c` was written to load distance matrices and sweep out ROC curves, producing GNUplot-format output. This was used to produce ROC curves both for the FRGC data and for the FERET data.

A.4 A Last Note

The system has been constructed to be as general as possible, comprising several small tools tied together by experiment scripts. Additionally, the source base itself consists of several pieces pulled from previous iterations of the codebase or from other projects entirely; it is our hope that the system will prove useful not only as a single-minded implementation of the ideas presented in this thesis but also as a general resource of working techniques and reusable tidbits for future researchers in this field. The codebase is released under the MIT open source license for this reason.

Acknowledgments

This work would not have been possible without the guidance and supervision of my research advisor, Dr. Patrick J. Flynn, and I owe my progress first and foremost to his input.

I am grateful to the research group at Colorado State University that assembled the Face Evaluation Identification System (FEIS), the system used in my experiments. Theirs were the most important shoulders upon which I stood. I acknowledge as well the Delaunay triangulation implementation by Geoff Leach at RMIT, Australia, which I used to choose automatically the edges for every face graph produced.

This thesis owes its existence to the many hours stolen from friends (and even family, when I should have been enjoying summer break) in favor of heaps of reading, marathons of coding and endless writing. It has given me a more solid foundation as a researcher, but not without expense. And so I acknowledge the understanding and support afforded countless times by the many people in my life.

Finally, shout-outs to Emacs and Mr. Coffee for being my closest, most loyal friends throughout this endeavor.

*And then, his thesis completed
His data and code he deleted.
But along came advisor
And said 'twould be wiser
If results could again be repeated!*

Bibliography

- [1] J.R. Beveridge, D. Bolme, B.A. Draper, and M. Teixeira. The csu face identification evaluation system: Its purpose, features, and structure. *Machine Vision and Application*, 16(2):128–138, Feb 2005.
- [2] David Bolme. Elastic bunch graph matching. Master’s thesis, Colorado State University, Summer 2003.
- [3] B. Gokberk, M.O. Irfanoglu, L. Akarun, and E. Alpaydm. Optimal gabor kernel location selection for face recognition. *Image Processing, 2003. ICIIP 2003. Proceedings. 2003 International Conference on*, 1:I-677–80 vol.1, Sept. 2003.
- [4] L. Guibas and J. Stolfi. Primitives for the manipulation of general subdivisions and the computation of voronoi diagrams. *ACM Transactions on Graphics*, 4(2):74–123, 1985.
- [5] J. B. MacQueen. Some methods for classification and analysis of multivariate observations. In L. M. Le Cam and J. Neyman, editors, *Proc. of the fifth Berkeley Symposium on Mathematical Statistics and Probability*, volume 1, pages 281–297. University of California Press, 1967.
- [6] P.J. Phillips, P.J. Flynn, T. Scruggs, K.W. Bowyer, J. Chang, K. Hoffman, J. Marques, J. Min, and W. Worek. Overview of the face recognition grand challenge. *Computer Vision and Pattern Recognition, 2005. Proceedings CVPR ’05., IEEE Computer Society Conference on*, 2005.
- [7] M. A. Turk and A. P. Pentland. Face recognition using eigenfaces. pages 586–591, 1991.

- [8] Laurenz Wiskott, Jean-Marc Fellous, Norbert Krüger, and Christoph von der Malsburg. Face recognition by elastic bunch graph matching. pages 355–398, 1999.
- [9] W. Zhao, R. Chellappa, P. J. Phillips, and A. Rosenfeld. Face recognition: A literature survey. *ACM Comput. Surv.*, 35(4):399–458, 2003.