THESIS

ELASTIC BUNCH GRAPH MATCHING

Submitted by

David S. Bolme

Computer Science

In partial fulfillment of the requirements

for the Degree of Master of Science

Colorado State University

Fort Collins, Colorado

Summer 2003

COLORADO STATE UNIVERSITY

May 22, 2003

WE HEREBY RECOMMEND THAT THE THESIS PREPARED UNDER OUR SUPERVISION BY DAVID S. BOLME ENTITLED ELASTIC BUNCH GRAPH MATCHING BE ACCEPTED AS FULFILLING IN PART REQUIREMENTS FOR THE DEGREE OF MASTER OF SCIENCE.

Committee on Graduate Work

Committee Member

Committee Member

Adviser

Department Head

ABSTRACT OF THESIS

ELASTIC BUNCH GRAPH MATCHING

Elastic Bunch Graph Matching is a face recognition algorithm that is distributed with CSU's Evaluation of Face Recognition Algorithms System. The algorithm is modeled after the Bochum/USC face recognition algorithm used in the FERET evaluation. The algorithm recognizes novel faces by first localizing a set of landmark features and then measuring similarity between these features. Both localization and comparison uses Gabor jets extracted at landmark positions. In localization, jets are extracted from novel images and matched to jets extracted from a set of training/model jets. Similarity between novel images is expressed as function of similarity between localized Gabor jets corresponding to facial landmarks. A study of how accurately a landmark is localized using different displacement estimation methods is presented. The overall performance of the algorithm subject to changes in the number of training/model images, choice of specific wavelet encoding, displacement estimation technique and Gabor jet similarity measure is explored in a series of independent tests. Several findings were particularly striking, including results suggesting that landmark localization is less reliable than might be expected. However, it is also striking that this did not appear to greatly degrade recognition performance.

David S. Bolme
Computer Science
Colorado State University
Fort Collins, Colorado 80523
Summer 2003

TABLE OF CONTENTS

Chapter 1

Introduction

1.1 Overview

A goal of the Colorado State University (CSU) Evaluation of Face Recognition Algorithms project is to establish a set of baseline algorithms to use for algorithm comparison. One of these algorithms is based on the Bochum/USC (University of Southern California) Elastic Bunch Graph Matching (EBGM) algorithm[15]. This thesis describes the CSU implementation of the EBGM algorithm and examines how the algorithm functions.

CSU chose the EBGM algorithm as one of the baseline algorithms for two reasons. First, the algorithm is fundamentally different from others implemented at CSU because it recognizes faces by comparing their parts, instead of performing holistic image matching. Second, the Bochum/USC algorithm performed very well in the FERET study[13]. Although the resulting CSU EBGM implementation does not replicate Bochum/USC performance on the FERET database, the results are very good in relation to other algorithms evaluated in the original FERET test.

The CSU EBGM algorithm is available from the CSU HumanID web page [3] as part of our Face Identification Evaluation System[4]. The system is part of an open source project that includes four baseline algorithms and a set of tools and scripts that can be used to evaluate the performance of face recognition algorithms.

1.2 The EBGM Algorithm

This section will provide a high level introduction to the CSU EBGM algorithm by illustrating how the algorithm works and defining the key terms used through out the thesis. The algorithm needs examples of what the landmark jets look like to locate the landmarks in a novel image. The features are represented by

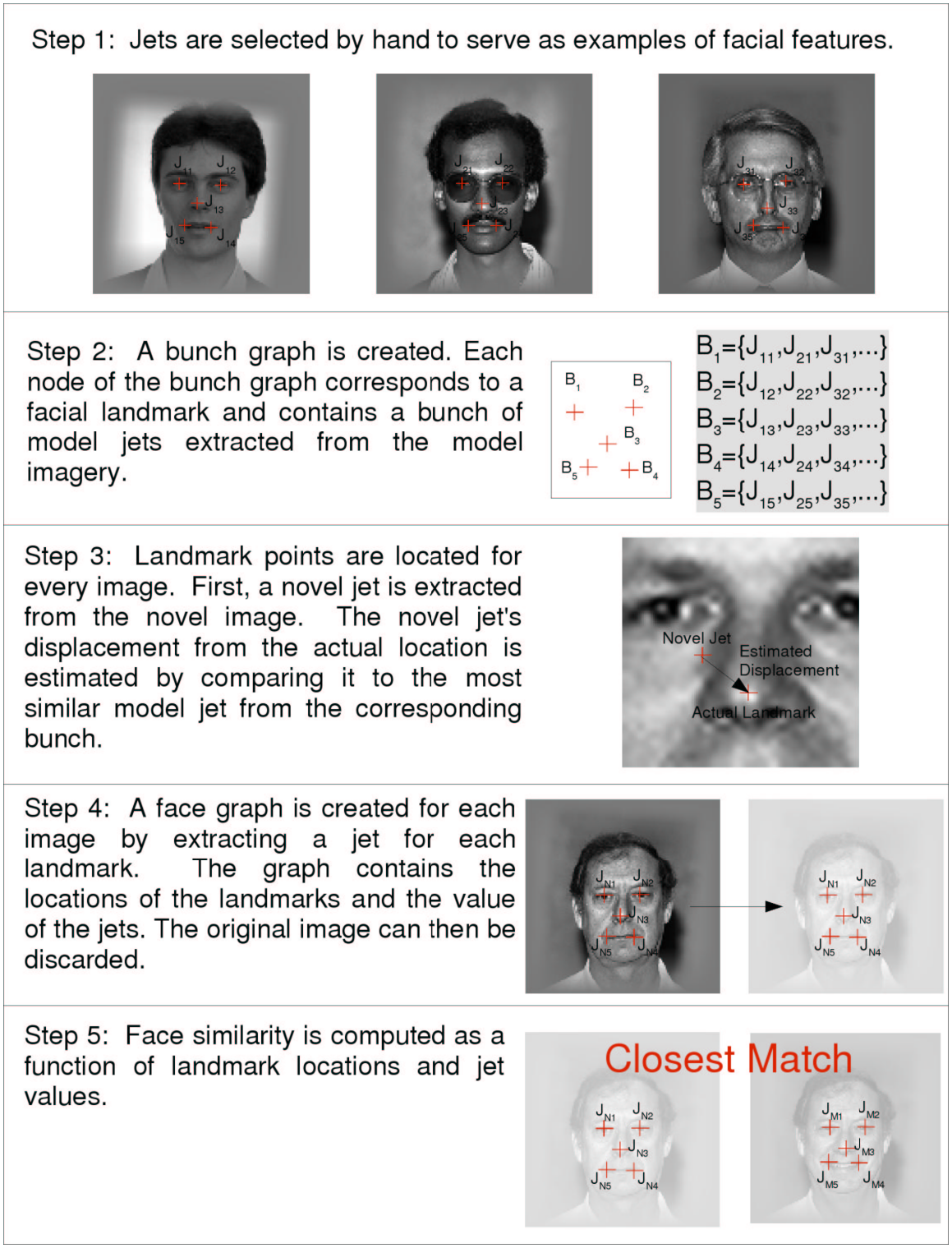


Figure 1.1: The basic steps in the EBGM algorithm.

Gabor jets, in this case referred to as **model jets**. The jets are extracted from images with manually selected landmark locations.

The model jets are then collected in a data structure called a **bunch graph**. The bunch graph has a node for every landmark on the face. Every node is a collection model jets for the corresponding landmark. The bunch graph serves as a database of landmark descriptions that can be used to locate landmarks in novel imagery.

The EBGM algorithm computes the similarity of two images. To accomplish this task, the algorithm first finds **landmark locations** on the images that correspond to facial features such as the eyes, nose, and mouth. It then uses **Gabor wavelet** convolutions at these points to describe the features of the landmark. All of the wavelet convolution values at a single point are referred to as a **Gabor jet** and are used to represent a landmark. A **face graph** is used to represent each image. The face graph nodes are placed at the landmark locations, and each node contains a Gabor jet extracted from that location. The **similarity** of two images is a function of the corresponding face graphs. See Figure 1.1 for a brief step by step walk through of the EBGM algorithm.

Locating a landmark has two steps. First, the location of the landmark is estimated based on the known locations of other landmarks in the image, and second, that estimate is refined by extracting a Gabor jet from that image and comparing that jet to one of the models. The algorithm assumes that the eye coordinates are already known[1]. Estimating the location of the other landmarks is easy based on the known eye coordinates. In theory all of the landmark locations could be estimated from the eye coordinates. In practice, each new landmark location is estimated based on the set of previously localized points. For example, the eye coordinates are used to estimate the landmark location corresponding to the bridge of the nose. Because the bridge of the nose is relatively close to the eyes the estimate should be very accurate. That landmark location is then refined by comparing a Gabor jet extracted from the estimated point to a model jet from the bunch graph. Now the location of three points is known, the eyes and the bridge of the nose, and all three can be used to estimate the location of the left eyebrow. This process is iterated until all landmark locations are found.

The landmark location is refined by extracting a **novel jet** from the estimated location of the landmark in the novel image. The most similar jet is selected from the bunch graph and this jet then serves as a model. Both the novel jet and model jet contain frequency information about the local image region around their extraction point. Using phase information stored in the two jets, it is possible to calculate a displacement of the novel jet from the true location of the landmark. This is accomplished by finding a displacement that would make

[1]The CSU system primarily uses the FERET database which has manually selected eye coordinates.

the phase information of the novel jet very similar to the phase information of the model jet. Chapters 2 and 3 focus on the background and mathematics of this process.

Once the landmarks have been located, a structure called a face graph is created where each node corresponds to a landmark. The landmarks are characterized by two things, a location in the image and a Gabor jet extracted from that location. After the face graph is created, the image is discarded, and the face graph becomes the internal representation of that image. The face graph occupies less memory than the image, and computing the similarity of face graphs is much faster than computing the similarity of images. For this reason, an entire database of faces can be kept in memory, and new images can be identified rapidly.

Because it is a possible point of confusion for those first learning this algorithm, the algorithm uses just one bunch graph but computes face graphs for every image being recognized. As just described, the bunch graph is used to refine the precise location estimate for each landmark in each new face image to be recognized. The bunch graph contains Gabor jets from a set of representative images, the model images. A face graph must be created for each image to be recognized. It is the face graphs that are compared during recognition.

After the algorithm has created a face graph for two images, their similarity can be computed. There are two different ways to measure the similarity of a face graph. The first method compares of the geometry of the graphs (landmark locations). The second method compares the similarity of the Gabor jets (landmark jets).

1.3 EBGM and the FERET Test

As mention before, CSU selected the Bochum/USC algorithm because it performed well on the FERET test. In the FERET test, the algorithm was modified[9] in order to perform better on the FERET database. Unfortunately there was an absence of detailed documentation on this particular version of the algorithm. As a result I have based the CSU algorithm on [15], which is a very detailed text on how an earlier version of the algorithm was implemented, and on many communications with Kazunori Okada from USC in which he clarified many aspects of the algorithm.

There are a few areas in which the CSU implementation differs from the Bochum/USC algorithm used in the FERET tests. Some of these differences have been tested to determine what effect such changes have on the CSU implementation. Each difference was tested in a manor similar to the experiments presented in Chapter 6.

Automatic Face Finding: The original algorithm used automatic face finding. Each original image was scanned for a face. Once the size and pose were determined, the face was geometrically registered

such that each face occupied the center of a smaller image. After the landmark localization process, the Gabor wavelets were rescaled based on the distance between all landmark locations. Thus, the effective scale of the face was a function of all landmark locations. The CSU system uses manually selected eye coordinates to determine the location of an image, thus bypassing the automatic location process. Normalization takes place before the EBGM algorithm is run. The images are rotated, translated, and scaled, such that the eye coordinates in the images are aligned. In this system, the eye coordinates determine the effective size of the normalized face, instead of using all landmark locations. Unfortunately face finding would be a significant addition to the CSU algorithm and its effect cannot be tested at this time.

Normalization: A detailed description of the Bochum/USC image normalization process is not available. It is known that the Bochum/USC algorithm geometrically normalized the face based on the output of the automatic face finding procedure. The face occupied the center of an image the center of an image that was 128 pixels on an edge. The Bochum/USC system also assumed the normalized images wrap around in a toroidal fashion as a by product of the FFT performed on the image. To reduce the edge effects in image wrapping, edge to edge border was smoothed. A detailed description of the CSU Normalization process can be found in Section 5.1.

Histogram Equalization: Another difference in the algorithms is the way that histogram equalization is applied. The Bochum/USC algorithm applied histogram equalization immediately after landmark localization. The equalization was based on a rectangle of pixel values completely contained in the face; however, the equalization effected the entire image. The CSU algorithm applied histogram equalization based on the entire image which effected the entire image. Thus, the CSU implementation bases the histogram equalization on many pixels that are of the background. To test this effect, a different histogram equalization was applied during the normalization process. This equalization was applied before localization, and it was based on a rectangular region contained in the face. All attempts to replicate the Bochum/USC equalization resulted in worse performance.

Wavelet Convolution: The Bochum/USC system performed wavelet convolution by computing the FFT of the image and then convolving with a Gaussian in frequency space. The CSU system precomputes wavelet masks and convolves the mask with the original image. The effect of this difference has not been tested.

DC Free Wavelets: The Gabor wavelets used in the Bochum/USC algorithm were carefully selected to be DC free. The CSU version uses the exact same parameters as the Bochum/USC system; however, the

basic wavelet equation does not included the DC free term. Thus, if a solid gray image is convolved with an even CSU wavelet, it will yield a small convolution value where the original system would produce zero. Adding the DC Free term to the CSU algorithm had very little effect on algorithm performance.

Landmark Set: There is no precise definition of which landmarks and interpolated points were used in the original system. Documentation suggest approximately 80. The CSU system uses 25 landmarks and 55 interpolated points, although the exact points not the same as those chosen by Bochum/USC. The effect of this difference has not been tested.

Model Set: The Bochum/USC algorithm used 70 model images; however, information on the particular model images used is not available. The CSU system uses 70 randomly chosen images from the FERET database. The effect of this difference has not been tested.

Various Poses: The CSU algorithm is not designed to handle faces at different poses. The Bochum/USC algorithm could detect and recognize images taken at a different pose. This involved detecting the face, locating the landmarks, and then transforming the Gabor jets in such a way that they would approximate jets that were taken from a frontal image. Because the CSU system only works on frontal images, this effect was not tested.

Similarity Measure: The Bochum/USC algorithm and CSU algorithms use different jet similarity measures to compute the similarity of faces. The Bochum/USC algorithm used a measure, referred to in this thesis as FGMagnitude, which ignored the phase component of the Gabor jets. The CSU algorithm uses another similarity measure developed by USC that uses magnitude, phase, and displacement compensation (FGPredictiveStep). Bochum/USC selected the FGMagnitude measure because it performed slightly better on the fb probe set. Results from tests on the CSU algorithm have shown that displacement compensated measures are much more robust. A detailed analysis of similarity measures can be found in Section 6.5.

The results of these tests indicate that none of the above differences will have a significant effect on the performance of the algorithm, and therefore, none of these modifications were adopted for the CSU implementation of the EBGM algorithm.

Other configuration experiments in Chapter 6 indicate that some modifications to the algorithm can increase performance. These tests led to two different configurations for the CSU algorithm: CSU EBGM Standard and CSU EBGM Optimal. Both these algorithms are based on a different set of wavelets and use alternative

landmark localization methods and face graph similarity measures. Both of these configurations perform better than a previous configuration based on the Bochum/USC algorithm. Details on the configurations are found in Section 6.6.

Table 1.1 compares the performance of the CSU EBGM algorithms to some of the other algorithms that performed well on the FERET tests and their respective CSU implementations. Although the CSU EBGM implementation performs well, it is not able to achieve Bochum/USC's outstanding performance on the FERET test. The CSU algorithm does not stand out in the same way as the USC algorithm; however, its results are still very competitive with the rest of the algorithms evaluated in the FERET test.

CSU also chose three other algorithms that performed well on the FERET test to serve as a baseline for the Evaluation of Face Recognition algorithm. These algorithms attempt to match whole images based on their pixel values. The other algorithms, Principle Components Analysis (PCA and ef_hist_dev_md)[16, 6], Linear Discriminant Analysis (LDA and umd_mar_97)[16, 17, 18], and Bayesian Interpersonal Classifier (BIC and mit_sep_96)[7], are trained to find a linear subspace that provides good matches during nearest neighbor classification. PCA and LDA both perform smart data reduction on the images before doing nearest neighbor classification. LDA also attempts to produce a basis that is good for discrimination among the clusters formed by multiple training images of the same subject. The BIC attempts to classify the difference of two images as being of the same subject or different subjects. Instead of analyzing localized image features to match images, all of these techniques compute statistical features of the entire image that are thought to increase matching ability.

PCA, also known as eigenfaces, is the most studied of the CSU algorithms. Throughout the CSU project, many different aspects of PCA have been studied and refined. The result is an algorithm that performs significantly better than the version used in the original FERET test. The results show that the CSU EBGM and CSU PCA have similar performance.

The CSU implementations of LDA and BIC do not perform as well as the original versions developed for the FERET tests. We believe that this is most likely due to differences in training methods. Regardless, the CSU EBGM performance is comparable to the original algorithms chosen to be included as the CSU baseline.

1.4 Thesis Organization

This first chapter has given a basic introduction to the thesis and has defined the key terms.

Chapter 2 introduces Gabor wavelets and their relationship to a frequency space decomposition of an image.

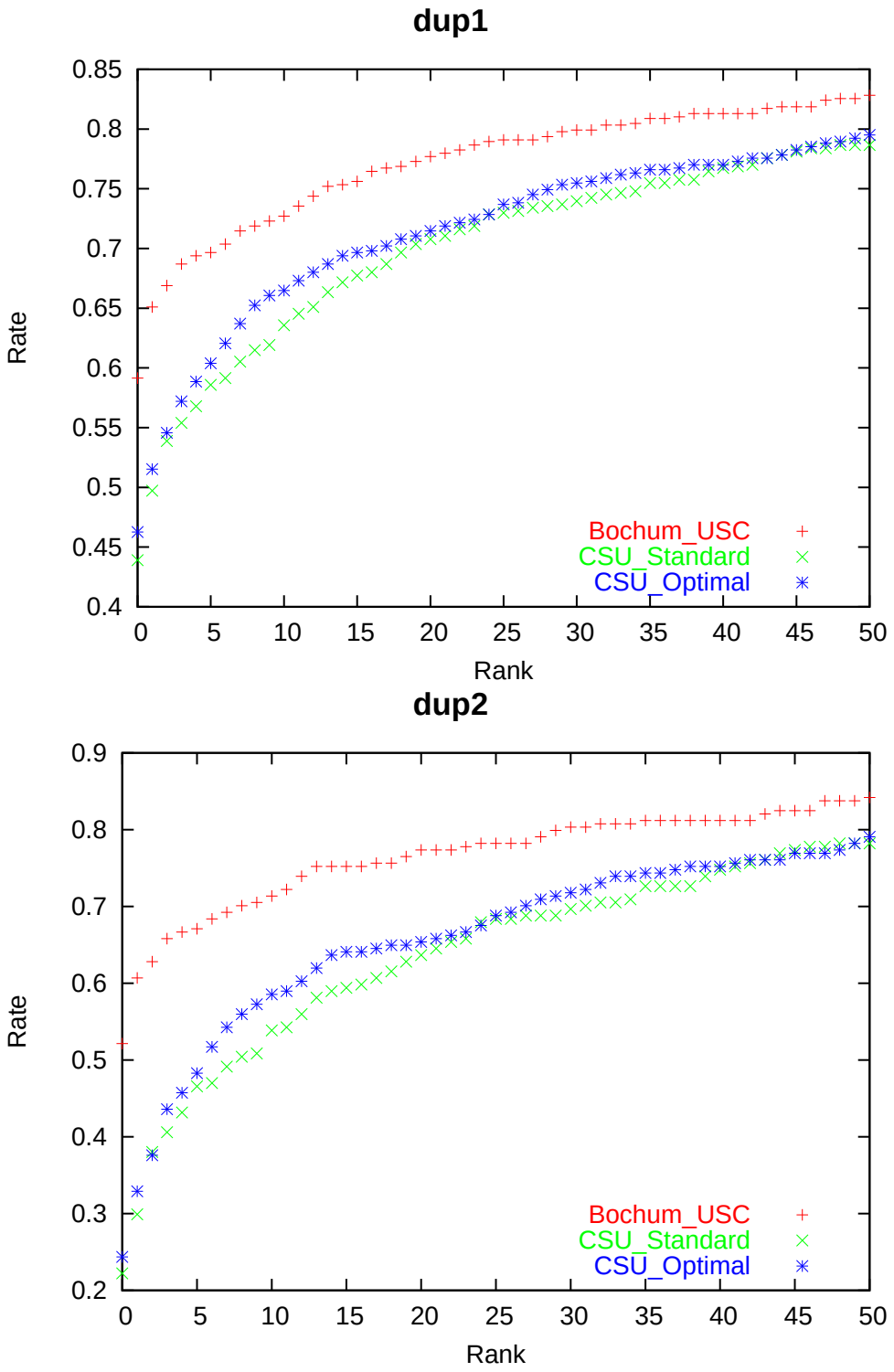


Figure 1.2: Comparison of the CSU EBGM algorithm to the Bochum/USC algorithm in the FERET test dup1 and dup2.

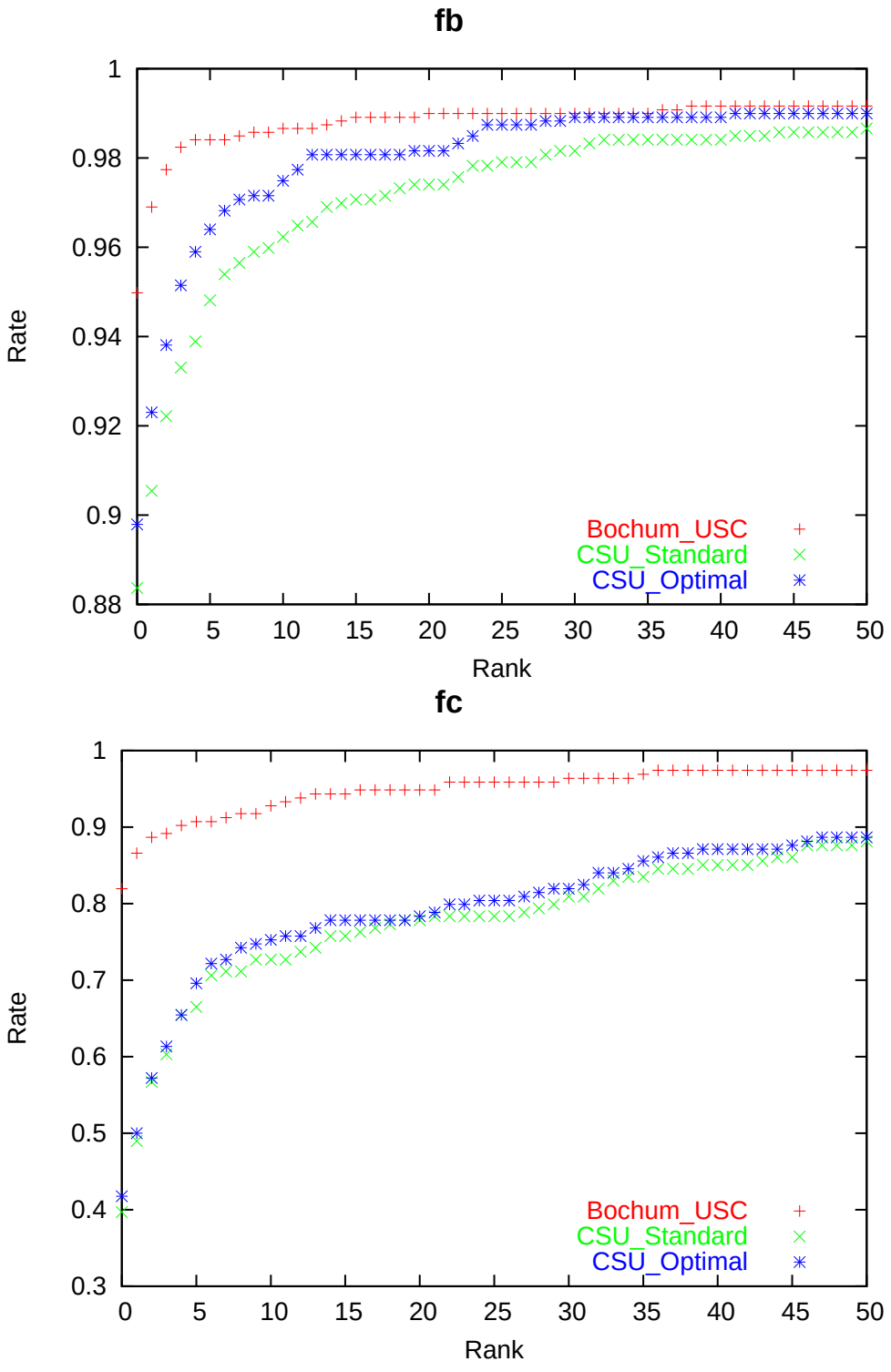


Figure 1.3: Comparison of the CSU EBGM algorithm to the Bochum/USC algorithm in the FERET test fb and fc.

Table 1.1: Results on the FERET Probe sets. Original FERET test results were obtained from the FERET Database web site [11]. CSU results are from release 5.0.

Algorithm	fb	fc	dup1	dup2
CSU EBGM Standard	87.2%	38.6%	42.9%	22.6%
CSU EBGM Optimal	89.8%	41.8%	46.3%	24.4%
usc_mar_97	95.0%	82.0%	59.1%	52.1%
CSU PCA	85.3%	65.5%	44.3%	21.8%
ef_hist_dev_md	74.1%	23.2%	42.2%	16.7%
CSU LDA	70.1%	3.6%	30.5%	13.2%
umd_mar_97	96.2%	58.8%	47.2%	20.9%
CSU BIC	81.9%	37.1%	52.4%	31.6%
mit_sep_96	94.8%	32.0%	57.6%	34.2%

One key part of this chapter is the discussion of wavelet phase and its relationship to displacement. Understanding these basic concepts is fundamental to understanding the landmark localization process.

Chapter 3 discusses another fundamental concept in the EBGM algorithm: Landmarks. The chapter explains in detail what landmarks are and how they are located. It also discusses the use of Gabor jets as an internal representation of the landmark's appearance, how to compute the similarity of Gabor jets, and how the phase information in Gabor jets is used to localize the landmark locations in an image. In addition, this chapter includes a detailed study on different localization methods and how well they perform.

Chapter 4 describes the structure and use of face graphs. It also defines exactly how the similarity of two face graphs is computed. The chapter introduces the two types of similarity function: landmark jet similarity, and geometry similarity.

Chapter 5 covers the same information as the algorithm overview presented in Section 1.2; however, it goes into much more detail by putting the concepts discussed in earlier chapters into the context of the full algorithm. It also discusses other implementation details and software structure that is necessary to understand the system.

Chapter 6 tests many different algorithm configuration options to determine what design decisions make a difference in the algorithms performance. Various experiments will test the choice of Gabor wavelets, model imagery, localization methods, and similarity measure. The result is an optimized configuration for the algorithm that will be compared to the other baseline algorithms developed at CSU.

Chapter 7 summarizes the results of a variety of tests on the CSU EBGM algorithm and future work on the algorithm will be discussed.

Chapter 2

Gabor Wavelets

2.1 Functions in Frequency Space

Gabor wavelets are fundamental to the EBGM algorithm. To fully understand the algorithm, it is necessary to develop a background in wavelet and Fourier analysis [5]. Wavelet and Fourier analysis are both used to analyze frequency space properties of an image. The difference between the two is that wavelets operate on a localized image patch, while the Fourier transform operates over the entire image. Much of the wavelet theory presented in this paper finds its roots in Fourier analysis.

A Fourier transform decomposes a signal such that it can be represented as a combination of sinusoids. It is very useful in signal processing because there are many things that a frequency space analysis can reveal about a signal that are not obvious from the original data.

A one dimensional Fourier transform of the function x is computed using the following equation:

$$\mathcal{F}(x(t))(\omega) = \int_{-\infty}^{\infty} x(t)e^{-i\omega t}dt$$

The transform produces a new function that is a function of frequency instead of time. It is much easier to see what is going on if we break the transform into its real and imaginary parts.

$$\mathcal{F}(x(t))(\omega) = \int_{-\infty}^{\infty} x(t)\cos(\omega t)dt - i\int_{-\infty}^{\infty} x(t)\sin(\omega t)dt$$

This equation shows that the Fourier transform has decomposed the function into sine and cosine functions. Evaluating $\mathcal{F}(x(t))(\omega)$ at a particular frequency will yield a complex number that corresponds to the amplitude of the cosine (real) and sine (imaginary) parts of the original function at that frequency. If we sum all of the cosine and sine waves over all frequencies, it is possible to reconstruct the original signal. This is called an inverse Fourier transform and is computed using the equation:

$$x(t) = \frac{1}{2\pi}\int_{-\infty}^{\infty} \mathcal{F}(x(t))(\omega)e^{i\omega t}d\omega$$

When the Fourier transform is applied to a real function its output is complex.

At each frequency, there is both a real ($a_{\omega,r}$) and an imaginary ($a_{\omega,i}$) component to the original function:

$$x_\omega(t) = a_{\omega,r}\cos(t) + a_{\omega,i}\sin(t)$$

When summed together the two form a new sinusoid that contains all of the information at that particular frequency. It is possible to represent this new sinusoid as a single cosine function by giving that cosine function a specific amplitude and phase:

$$x_\omega(t) = a_{\omega,r}\cos(t) + a_{\omega,i}\sin(t) = a\cos(t+\phi) \tag{2.1}$$

This new equation has transformed the Fourier information from a Cartesian representation to a Polar representation. The new representation specifies a total magnitude and the phase angle of the sinusoid.

Now that we have conducted a limited review of Fourier theory, we can discuss Gabor wavelets. Wavelets are in many ways like Fourier Analysis; however, the wavelets have a limited scope. Gabor wavelets are basically a sinusoid multiplied by a Gaussian. Thus, when a function is convolved with the Gabor wavelet, the frequency information near the center of the Gaussian is captured, and frequency information far away from the center of the Gaussian has a negligible effect.

One equation that specifies a one dimensional Gabor wavelet is:

$$W(t,t_0,\omega) = e^{-\sigma(t-t_0)^2}e^{-i\omega(t-t_0)}$$

Convolution for the wavelet transform is defined as:

$$C(x(t))(t_o,\omega) = \int_{-\infty}^{\infty} x(t)W(t,t_0,\omega)dt$$

By expanding the wavelet function, the convolution looks a lot like the Fourier transform:

$$C(x(t))(t_0,\omega) = \int_{-\infty}^{\infty} x(t)e^{-\sigma(t-t_0)^2}e^{i\omega(t-t_0)}$$

$$C(x(t))(t_0,\omega) = \int_{-\infty}^{\infty} x(t)e^{-\sigma(t-t_0)^2}\cos(\omega(t-t_0)) + i\int_{-\infty}^{\infty} x(t)e^{-\sigma(t-t_0)^2}\sin(\omega(t-t_0))$$

The integral here produces a complex coefficient $C(x(t))(t_0,\omega)$, that describes the local frequency information of the function x at a specific frequency ω and time t_0. Like the Fourier transform this new wavelet coefficient has real and imaginary parts which correspond to cosine and sine function.

$$C(x(t))(t_0,\omega) = a_{real} + i\,a_{imag}$$

The only difference is that this time those sinusoids have been multiplied by a Gaussian. Revisiting Equation 2.1, this complex coefficient can also be represented in polar coordinates having a total magnitude of a and a phase angle of ϕ.

$$a_{real} = a\cos\phi \tag{2.2}$$

$$a_{imag} = a\sin\phi \tag{2.3}$$

Inversely, a and ϕ are found using the equations:

$$a = \sqrt{a_{real}^2 + a_{imag}^2} \tag{2.4}$$

$$\phi = \begin{cases} \arctan(a_{imag}/a_{real}) & \text{if } a_{real} > 0 \\ \pi + \arctan(a_{imag}/a_{real}) & \text{if } a_{real} < 0 \\ \pi/2 & \text{if } a_{real} = 0 \text{ and } a_{imag} \geq 0 \\ -\pi/2 & \text{if } a_{real} = 0 \text{ and } a_{imag} < 0 \end{cases} \tag{2.5}$$

Finally, it is important to understand the relationship of time displacements and the phase angle of the coefficient. Much of the EBGM algorithm relies on this polar coordinate transformation of the coefficients to estimate displacements of image points. If wavelet coefficients are computed at two points that are separated by a small displacement, the difference in the phase angle of the coefficients should be roughly proportional to the displacement between the two convolution points. This concept is discussed in more detail in Section 2.4; however, an illustrated example is shown here.

Figure 2.1 was produced by a computational experiment in Matlab(TM). The top graph is a function that is composed from 10 pseudo random sinusoids. The function was created by adding a sine wave and cosine wave with random amplitudes between -1 and 1 for each frequency.

Second from top are real (green) and imaginary (blue) Gabor wavelet functions. The frequency of the Gabor wavelets is the same as one of the sinusoids included in the original function.

The third graph shows the value of the wavelet convolutions at each point along the function. This shows the wavelets are responding to the sinusoid in the original function. When phase of the wavelet and the function are in alignment, the convolution values are high. When the phase of the function and the wavelet are out of sync by π the convolution values are negative.

The fourth graph shows the magnitude of the wavelet responses. This shows that the energy of that particular frequency component of the function is constant near the center of the function. The drop in the magnitude on the left and right sides of the graph is expected because the function is not defined past the edges.

The fifth graph shows the phase of the wavelet coefficients. It is easy to see that the phase of the coefficients complete one complete cycle during a period of the sinusoid. The code to generate this figure can be found in Appendix A.

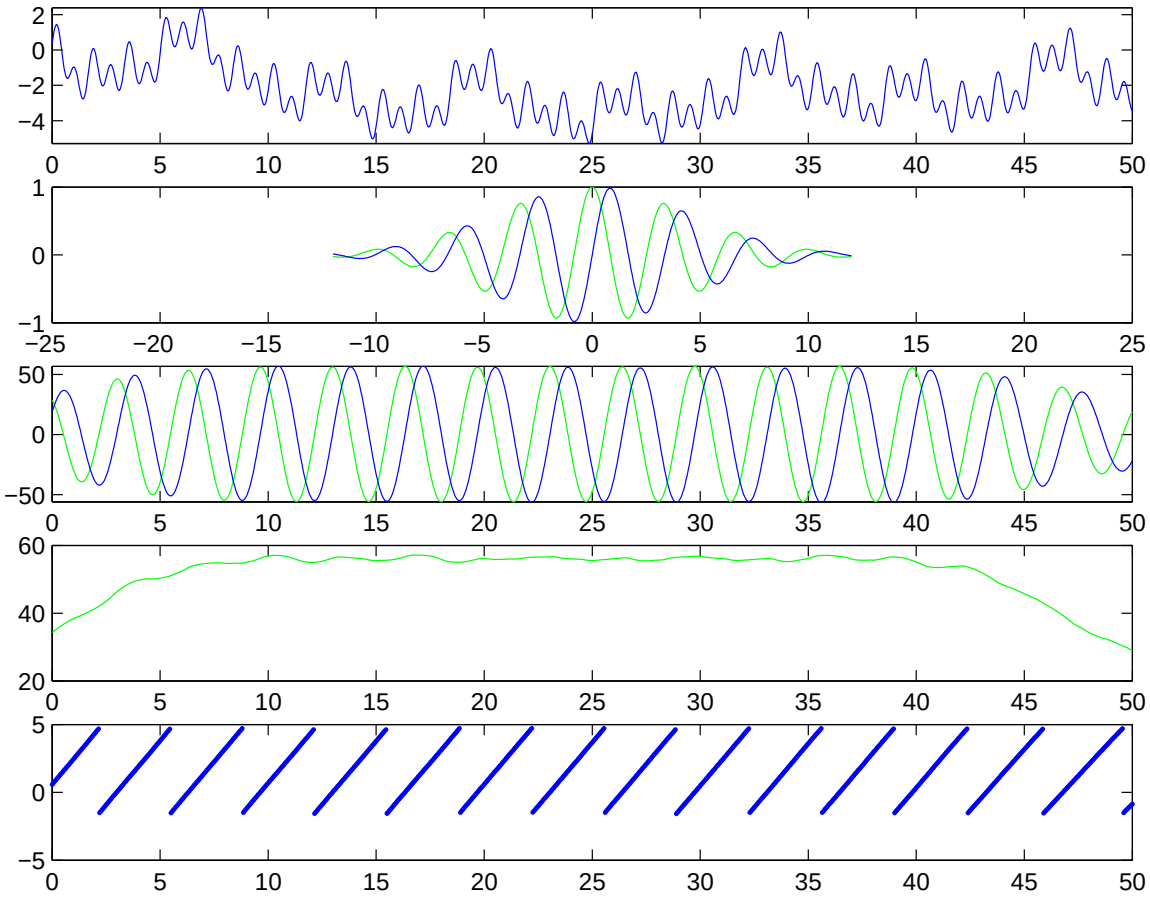


Figure 2.1: Wavelet convolution example.

2.2 1D to 2D

The previous section introduced the notion of one dimensional Gabor wavelets. The EBGM algorithm uses a two dimensional form of Gabor wavelet for image processing. The wavelet consists of a planer sinusoid multiplied by a two dimensional Gaussian. The sine wave is activated by frequency information in the image. The Gaussian insures that the convolution is dominated by the region of the image close to the center of the wavelet. This section introduces the different parameters used to describe the Gabor wavelets.

Gabor wavelets can take a variety of different forms. The wavelets used in this algorithm have parameters that control orientation, frequency, phase, size, and aspect ratio. To accurately describe the frequency information of a feature in an image, it is necessary to convolve the location with many instantiations of the wavelet. These instantiations typically sample at different frequencies and different orientations. This gives rise to the idea of the "Jet" which contains many coefficients from convolving with a variety of wavelets.

Figure 2.2 shows the result of convolving an image of a phase with a real and imaginary wavelet. Two dimensional Gabor wavelets respond to image features that are of the same orientation and frequency of the wavelet. This figure shows the original image and the two dimensional Gabor wavelet masks used for convolution. The bottom two images show the magnitude and phase convolution values at each point. The magnitude image indicates that the wavelets seem to respond to the nose and left ear and that the magnitude values change very slowly with displacement. The phase values are also roughly proportional to the horizontal displacement, much like the one dimensional example.

The EBGM algorithm performs the wavelet convolution using precomputed wavelet masks. Each mask is a two dimensional array that is used as a lookup table for wavelet values during convolution. The masks are centered over the correct location in the image, and each corresponding value is multiplied and added to the sum. To compute both the real and imaginary part of the wavelet, it is necessary to convolve the image with two masks that are out of phase by $\pi/2$, corresponding to the use of sine and cosine in the wavelet transform.

To fully understand what each of these parameters means, we will look at the full wavelet equation and discuss each of the parameters in turn. The CSU algorithm is designed to load wavelet parameters from a file. Thus, the system can easily specify the orientation, wavelength, phase, size of the Gaussian, and the aspect ratio of the Gaussian. An additional parameter controls the size of the mask for each Gabor wavelet kernel. The wavelets specification is based on equations used in [10]. This representation was chosen for its straightforward formulation and flexibility:

$$\mathcal{W}(x,y,\theta,\lambda,\varphi,\sigma,\gamma) \quad = \quad e^{-\frac{x'^2+\gamma^2y'^2}{2\sigma^2}} cos(2\pi\frac{x\prime}{\lambda}+\varphi) \tag{2.6}$$

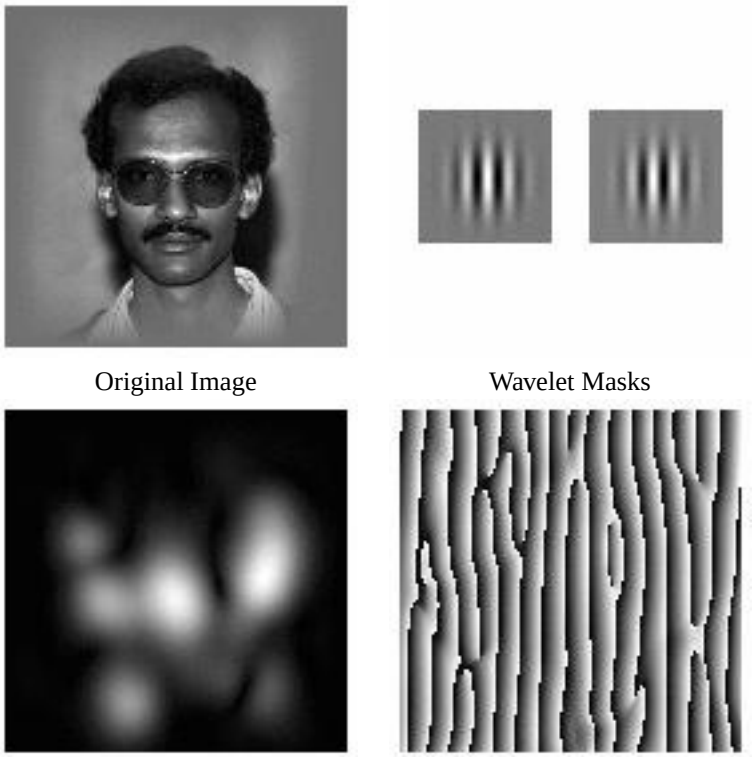


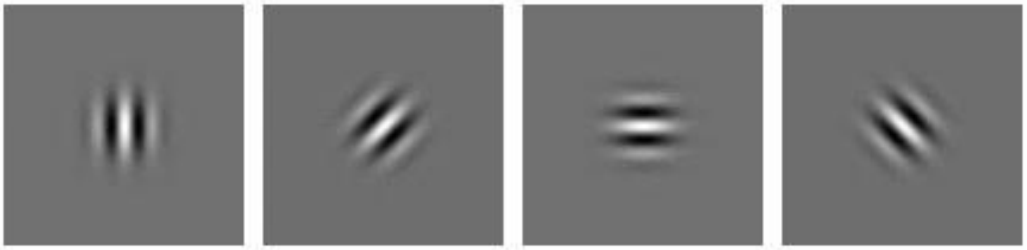
Figure 2.2: Two Dimensional Wavelet Convolution Example

$$x\prime = x\cos\theta + y\sin\theta \tag{2.7}$$

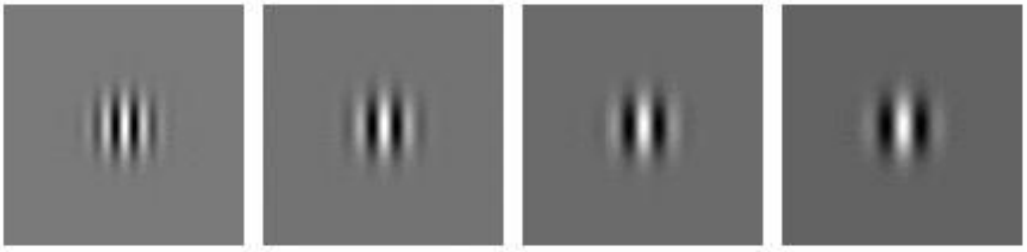
$$y\prime = -x\sin\theta + y\cos\theta \tag{2.8}$$

There are five parameters that control the wavelet:

- θ specifies the orientation of the wavelet. This parameter rotates the wavelet about its center. The orientation of the wavelets dictates the angle of the edges or bars for which the wavelet will respond. In most cases theta is a set of values from 0 to π. Values from π to 2π are redundant due to the symmetry of the wavelet. For example, consider an cosine based wavelet such that $\varphi = 0$. This wavelet is symmetric about the origin. In this case if the set of theta values contains values that are offset by π, i.e. $\theta \in 0, \pi$, the corresponding wavelets would be identical. On the other hand, consider an sine based wavelet with $\varphi = \pi/2$. Now the convolution values will have the same magnitude but opposite sign. The images below shows wavelets with θ of 0, $\pi/4$, $\pi/2$, and $3\pi/4$.

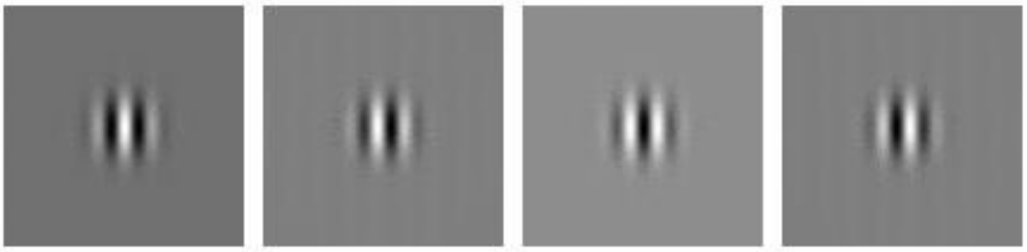


- λ specifies the wavelength of the cosine wave, or inversely the frequency of the wavelet. Wavelets with a large wavelength will respond to gradual changes in intensity in the image. Wavelets with short wavelengths will respond to sharp edges and bars. The four images below show the effect as λ is slowly increased from 8 to 16 pixels.

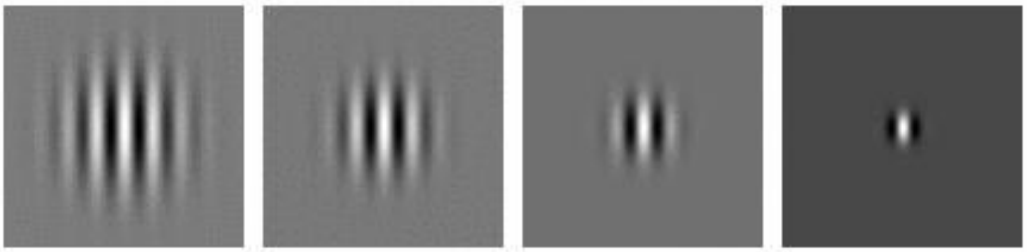


- φ specifies the phase of the sinusoid. Typically Gabor wavelets are based on a sine or cosine wave. In the case of this algorithm, cosine wavelets are thought to be the real part of the wavelet and the sine wavelets are thought to be the imaginary part of the wavelet. Therefore, a convolution with both phases

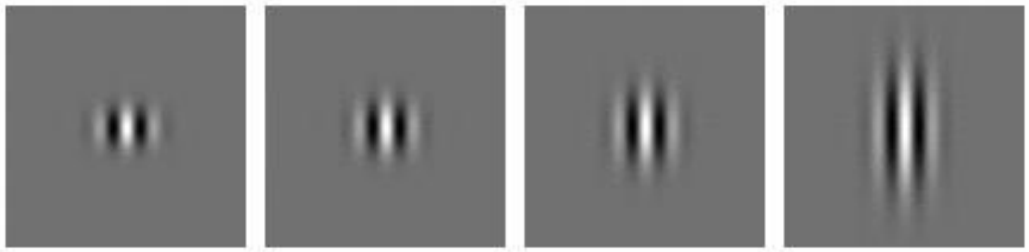
produces a complex coefficient. The mathematical foundation of the algorithm requires a complex coefficient based on two wavelets that have a phase offset of $\pi/2$, i.e. $\varphi \in \{ 0, \pi/2 \}$, although it is possible to use any pairing as long as the real and imaginary wavelet pair has a phase difference of $\pi/2$. For this reason φ almost always is 0 or $\pi/2$; however, the following images illustrates the values of 0, $\pi/2$, π, and $3\pi/2$. Convolutions with the last two images would be redundant for similar reasons as were described for θ.



- σ specifies the radius of the Gaussian. The size of the Gaussian is sometimes referred to as the wavelet's basis of support. The Gaussian size determines the amount of the image that effects convolution. In theory the entire image should effect the convolution; however, as the convolution moves further from the center of the Gaussian, the remaining computation becomes negligible. This parameter is usually proportional to the wavelength, such that wavelets of different size and frequency are scaled versions of each other, i.e. $\sigma = c\lambda$. The mask size is also closely related to the size of Gaussian. There is no strict relationship; however, mask sizes are chosen to capture the significant regions of the Gaussian. This figure shows σ values of 16, 12, 8, and 4.



- γ specifies the aspect ratio of the Gaussian. This parameter was included such that the wavelets could also approximate some biological models[10]. Most wavelets tested with the algorithm use an aspect ratio of 1.0. The figure below shows four different aspect ratios ranging from 0.5 to 1.5.



2.3 Wavelet Specification

To be consistent with the original Bochum/USC algorithm, the CSU algorithm adopted the following parameters from [15]. When translated into the new representation, the parameters are defined as:

- θ specifies the orientation of the wavelet. This particular set uses eight different orientations over the interval 0 to π. Orientations from π to 2π would be redundant due to the even/odd symmetry of the wavelets (see φ), i.e. $\theta \in \{\ 0, \pi/8, 2\pi/8, 3\pi/8, 4\pi/8, 5\pi/8, 6\pi/8, 7\pi/8\ \}$
- λ specifies the wavelength of the sine wave. This set starts at 4 pixels and continues to longer wavelengths at half octave intervals, i.e. $\lambda \in \{\ 4, 4\sqrt{2}, 8, 8\sqrt{2}, 16\ \}$
- φ specifies the phase of the sine wave. Typically Gabor wavelets are either even or odd. The even form of the sine wave corresponds to a cosine function; the odd form corresponds to a sine function. In the case of this algorithm, even wavelets are thought to be the real part of the wavelet and the odd wavelets are thought to be the imaginary part of the wavelet. Therefore, a convolution with both phases produces a complex coefficient, i.e. $\varphi \in \{\ 0, \pi/2\ \}$
- σ specifies the radius of the Gaussian. This parameter is usually proportional to the wavelength, such that wavelets of different size and frequency are scaled versions of each other, i.e. $\sigma = \lambda$
- γ specifies the aspect ratio of the Gaussian. This parameter was included such that the wavelets could also approximate some biological models. The wavelets used here have circular Gaussian, i.e. $\gamma = 1$

This yields 8 orientations, 5 frequencies, and 2 phases for a total of 80 different wavelets. A coefficient is computed by convolving a location in the image with the wavelet kernel in Equation 2.6. Figure 2.3 shows all 80 wavelet filters.

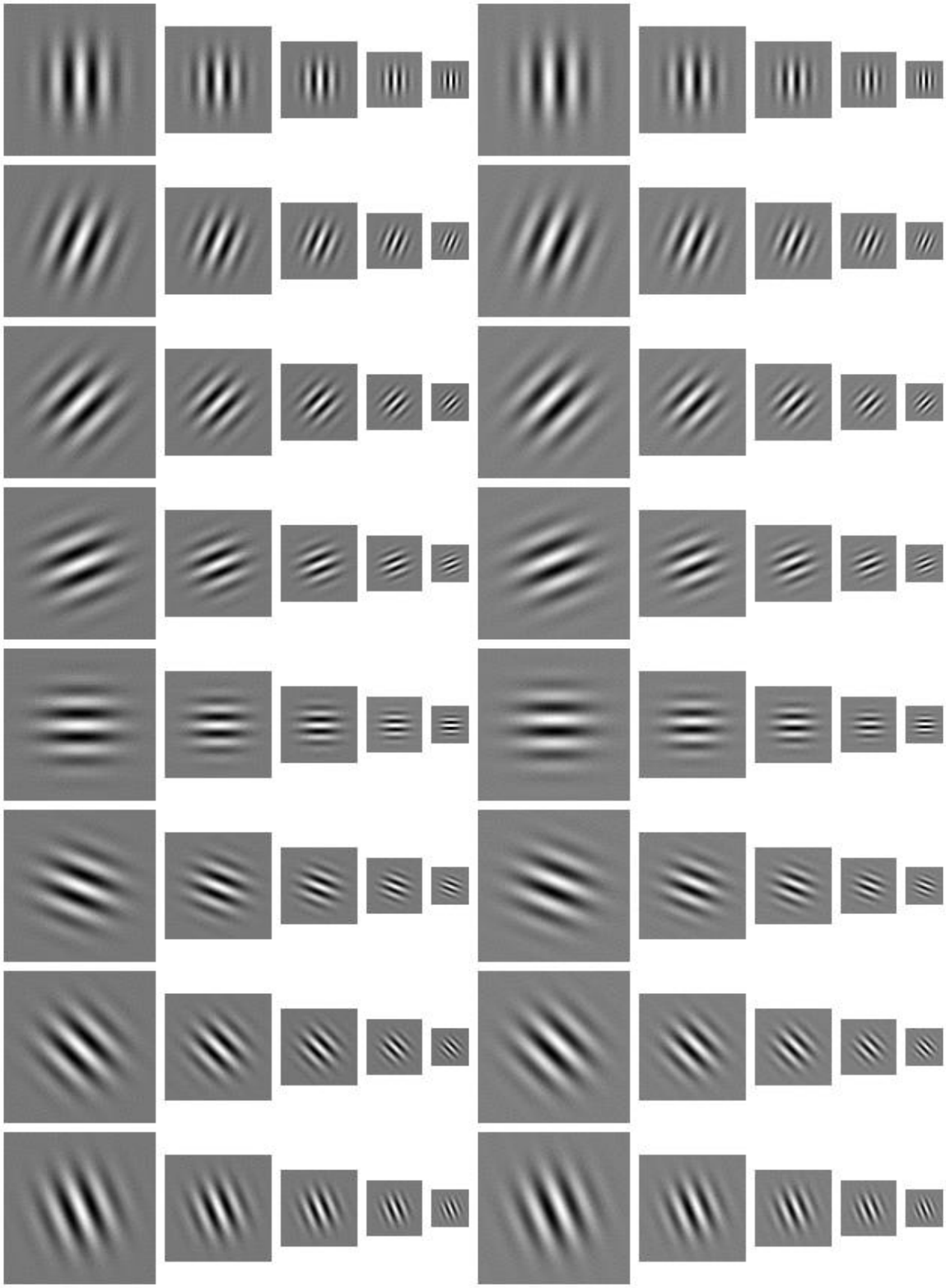


Figure 2.3: This figure shows 80 Gabor wavelet masks based on [15]. The left set of wavelets have a phase angle of 0 (even), and the right have a phase angle of $\pi/2$ (odd). The figure also shows the wavelengths (left to right) and orientations(top to bottom).

2.4 Estimating Wavelet Convolutions with Small Displacements

There are many times when the EBGM algorithm needs to find the value of a wavelet coefficient for a point in an image that is a small distance from a point where a wavelet coefficient has already been computed. The obvious way to determine the coefficient is to compute a new coefficient at that point by performing a wavelet convolution. An alternative to performing a convolution is to estimate the value of the coefficient based on the value that was previously computed.

Figure 2.4 illustrates this concept. Option 1 shows the result of shifting the entire wavelet. This would be the proper way to perform the wavelet computation at the new location. Option 2 shows the results of estimating the convolution at the new location. In this case the part of the image upon which the coefficients are based has not changed; however, the phase component has be shifted to approximate the coefficient at the new location.

It is possible to quickly estimate the value of a complex wavelet coefficient from a location a small distance from a known coefficient by shifting the phase of that known coefficient. This property is extremely useful in the EBGM algorithm because the wavelet convolutions are the dominating factor in the computational effort of the algorithm.

Chapter 3 explains how this estimation method can be applied to an entire Gabor jet of convolution values. This approximation method is very useful during landmark localization and similarity computation.

Imagine two one-dimensional Gabor wavelets displaced slightly on a one dimensional function. It is reasonable to assume that the magnitude of the convolution will change very slowly because most of the information remains relatively unchanged. The phase of the information under the Gaussian will change in a way that is proportional to the displacement.

This is obvious in Figure 2.1. The magnitude of the response remains constant over the middle of the function. Even near the edge of the function, the magnitude falls off slowly. The phase however is clearly a linear function of displacement. Figure 2.2 shows the same experiment conducted in two dimensions on a human face. As you can see the following assumptions still hold true:

1. The magnitude of the coefficient changes slowly with displacement.

2. The phase of the coefficient changes linearly under small displacements in the direction of the sinusoid.

We can now extend this theory to two-dimensional images and conduct a similar experiment. Imagine com-

Option 1: Shift the entire mask.

Option 2: Shift the phase fo the convolution value/

Figure 2.4: Estimating coefficients with small displacements.

puting complex coefficients for two locations in an image separated by a small displacement:

$$\vec{d} = \begin{pmatrix} dx \\ dy \end{pmatrix} \tag{2.9}$$

As with one-dimension, the magnitude a of the coefficients will be very similar, because the total magnitude of the frequency response under the Gaussian is almost the same. However, the phase component ϕ will change in a way that is proportional to the distance of displacement along the direction of the sinusoid.

To describe this property mathematically, first define a vector $\vec{k}$ such that it points in the direction of the sine wave, and has the magnitude equal to frequency of the sine wave. Thus, $\vec{k}$ has a value of radians per pixel.

$$\vec{k} = \begin{pmatrix} \frac{2\pi cos\theta}{\lambda} \\ \frac{2\pi sin\theta}{\lambda} \end{pmatrix} \tag{2.10}$$

The displacement along the direction of the sinusoid is easily computed as a dot product of these two vectors:

$$\vec{k} \cdot \vec{d}$$

The estimate for the value of the coefficients $a\prime$ and $\phi\prime$ for a small displacement is found with the equations:

$$a\prime \approx a \tag{2.11}$$

$$\phi\prime \approx \phi + \vec{k} \cdot \vec{d} \tag{2.12}$$

Now we can conduct a two dimensional experiment similar to the one presented in Figure 2.5. This experiment is conducted on actual face imagery. First, the wavelet coefficient is determined for the tip of the nose. Then for each pixel in the image, the wavelet coefficient is computed using a full convolution. The new coefficient is used to estimate the coefficient at the tip of the nose by shifting the phase of the wavelet by the known displacement. Finally, the similarity of the estimate and the actual coefficient extracted at the nose are compared.

The similarity values are presented as images. One of the images in Figure 2.5 shows the similarity of the magnitudes, computed as:

$$S_{magnitude} = exp(-0.1 * (a' - a)^2)$$

This similarity measure has a maximum of 1 when the estimate and actual value are the same. The similarity of phase is computed with the function:

$$S_{phase} = cos(\phi' - \phi)$$

This similarity measure is computed between 1 and -1. The measure shows the alignment of the phase angles.

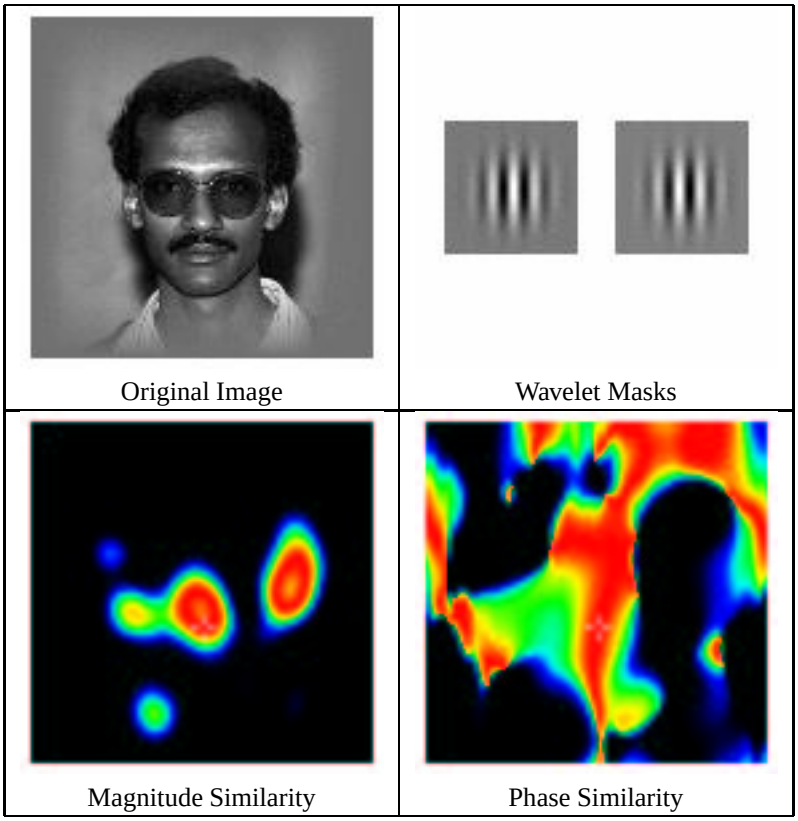


Figure 2.5: Wavelet Displacement Results

Figure 2.5 shows the results of the experiment. Each point in the original image is convolved with the even and odd Gabor masks . Then for each point in the image, the wavelet coefficient is estimated for the tip of the nose. The similarity of this convolution value is compared to the actual coefficient calculated at the tip of the nose. The similarity images are shown in false colored images with red indicating high similarity and with black indicating very low similarity.

The magnitude image shows that there is a region of high similarity around the marked point. This means that the magnitudes are very similar around that point. Further from the tip of the nose, the magnitude changes and the similarity decreases. This is because the nose tip has very little influence on the the image patches outside the local region.

The phase image also shows that there is a region of high similarity surrounding the nose tip. Without the displacement compensation, there would be periodic banding similar to the phase image in Figure 2.2.

Chapter 3

Landmarks

3.1 Landmark Locations and Features

The CSU EBGM algorithm is easily divided into three steps. The first step is to determine the landmark locations. Next, a face graph is created for every image that needs to be matched. Finally, the similarity of face graphs is computed and final classification is based on this similarity. This chapter describes in detail what landmarks and Gabor Jets are, and how they are used to locate the landmarks and measure similarity.

Landmarks are parts of the face that are easily located and have similar structure across all faces. Some obvious examples of landmarks are the eyes, nose, and mouth. Each of these landmarks is well defined, is common to all faces, and has distinct representations in image space. Typically landmarks are defined such that their location has a very small error tolerance. Instead of defining a landmark as the "nose", the landmark is defined as the "nose tip". The nose tip can be located to within a few pixels. The nose in its entirety is a large structure, and it is difficult to select a single point to represent its location.

There are other parts of the face that would serve as poor landmarks. Some examples of these would be the forehead or cheeks. Although these are common structures of the face, it is very difficult to determine an exact location for these structures. Even if you could define such a point, such as a cheekbone, it is very difficult to locate it to a high degree of accuracy because there may be nothing in an image that cues the algorithm to such a point.

There are times when it may be necessary to define a point that is difficult to locate, such as a point on the edge of the head. In the algorithm it is important to find such points to determine the border of the face. However, finding an exact point is difficult. If you consider a point of the side of the face near the ear, it is easy to define the horizontal coordinate of that point. Determining the vertical coordinate is much more difficult because the point is valid anywhere along the side of the face.

The EBGM algorithm uses local frequency information on facial structures to perform classification. Facial structures are typically referred to as landmarks. The concept of a landmark splits into two distinct parts: landmark locations and landmark jets.

Landmark Locations are the pixel coordinates of landmarks. Landmark locations define the geometry of the face. An example of this is the nose tip, which has a well defined location.

Landmark Features are defined by the frequency information of the local regions that surround the landmark locations. A landmark jet is not a particular point but instead contains information about the pixel values surrounding the landmark location. A landmark jet refers to information on what the landmark looks like.

Gabor jets are used to represent the landmark jet information in the EBGM algorithm. A Gabor jet is produced by convolving the landmark location with a collection of Gabor masks. Therefore, the Gabor jet will contain a good description of the local frequency information around the landmark. The structure of Gabor wavelets allow this information to be heavily weighted in the area immediately surrounding the landmark, while still covering enough of the image to get a good description of the landmark.

Gabor jets act as feature vectors that describe the landmark from which the jet was taken. Landmark similarity is based only on the Gabor jets taken from the two landmarks. The jets are based on a variety of Gabor masks and contain an accurate description of the landmark. Therefore, the similarity of jets is a good indicator of the similarity of the landmarks.

3.2 Estimating Landmark Locations

The landmark localization process is divided into two steps: initial estimation and refinement. This section discusses the first step of locating a landmark in a novel image. When locating a landmark, the algorithm first makes an educated guess of the landmark's position. The algorithm then refines that guess based on the localized image region surrounding that point. The refinement process is described in Section 3.5.

During landmark localization the algorithm first needs an initial estimate of the location. The imagery used for this thesis was geometrically normalized using the eye coordinates. Because of this the eyes will always be at in the same point of the image with very little error. It follows that the rest of the landmark locations in the image are approximately known.

To more accurately estimate the location of the points, the algorithm bases its estimate on the locations of previously localized landmark locations. The algorithm starts by locating the eye coordinates and then

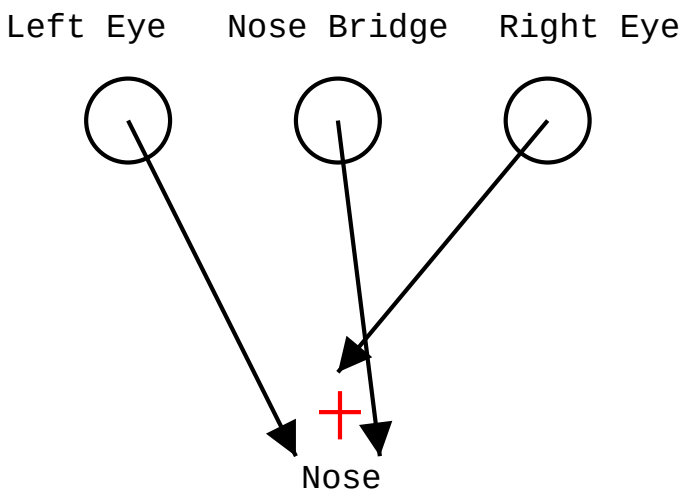


Figure 3.1: This figure shows how the algorithm produces an initial estimate of a landmark's location, such as the nose. In this example the Left Eye, Nose Bridge, and Right Eye have already been placed. The vectors show the average displacement of the nose from the previously placed nodes. The initial estimate of the location of the nose is a weighted average of these three points.

proceeds radially outward from the center of the face, locating one landmark at a time. For each landmark, the previously located points are used to estimate the location of the next landmark.

Each node in the bunch graph contains the average location of its landmark. The difference between the landmarks can be described as a two dimensional vector:

$$\overrightarrow{v_{mn}} = \overrightarrow{p_n} - \overrightarrow{p_m}$$

where $\overrightarrow{v_{mn}}$ is the difference in positions of nodes m and n. The model imagery can give us a good estimate of $\overrightarrow{v_{mn}}$. If we know the location of one of the nodes in a novel image, we can easily estimate the location of the other using the equation:

$$\overrightarrow{p_n} \approx \overrightarrow{p_m} + \overrightarrow{v_{mn}}$$

When the locations of multiple nodes are known, this estimate can be refined such that it is the centroid of all such estimates of $\overrightarrow{p_n}$. This gives rise to the equation:

$$\overrightarrow{p_n} \approx \frac{1}{M} \sum_{i=1}^{M} \overrightarrow{p_i} + \overrightarrow{v_{in}}$$

where nodes from 1 to M have already been placed. See Figure 3.1 for a graphical illustration.

Finally, nodes that are further away from the location being estimated should have less effect on the estimate than the nodes that are very close. For example, you would expect a node for the mouth would produce a good estimate for the location of the chin, while the estimate from the top of the head to the chin would probably be

very unreliable. This leads to the final equation for estimating the location of a landmark. The final equation is a weighted average, such that the nodes that are closer to the landmark carry a heavier weight.

$$\begin{aligned} \vec{p_n} &\approx \frac{\sum_{i=1}^{M} w_{in}(\vec{p_i} + \vec{v_{in}})}{\sum_{i=1}^{M} w_{in}} \\ w_{in} &= e^{-|v_{in}|} \end{aligned}$$

3.3 Gabor Jets

The CSU EBGM algorithm uses Gabor jets to represent landmark jets. The Gabor jets describe the local frequency information around the landmark locations. This section will define Gabor jets and show how they are created.

Gabor jets are a collection of complex Gabor coefficients from the same location in an image. The coefficients are generated using Gabor wavelets of a variety of different sizes, orientations, and frequencies.

For the standard configuration, Gabor jets are based on 40 complex wavelets where each wavelet has a real and imaginary component. These jets use eight orientations, five frequencies, and two phases (real and imaginary). A jet from location (x,y) in an image is produced by convolving that point with every mask in Figure 2.3. The resulting values are converted into polar coordinates using Equations 2.4 and 2.5, and are stored in an array such that a_j and ϕ_j correspond to the j^{th} complex wavelet pair.

As a result, Gabor jets contain a "description" of the frequency information localized around that single point in the image. Each wavelet coefficient captures information about one combination of phase, orientation, and frequency. In practice, this results in a description across multiple frequencies and orientations.

3.4 Jet Similarity

Measuring the similarity for two jets is fundamental and is required for both landmark localization and face graph similarity measurement. Three similarity measures are considered here: S_ϕ, S_a, and S_D. The first measure is very similar to correlation:

$$S_\phi(J, J') = \frac{\sum_{j=1}^{N} a_j a'_j \cos(\phi_j - \phi'_j)}{\sqrt{\sum_{j=1}^{N} a_j^2 \sum_{j=1}^{N} a_j'^2}} \tag{3.1}$$

where N is the number of wavelet coefficients in the jet. This function is called the "Phase" similarity measure. This measure effectively computes a similarity between -1.0 and 1.0. Although not exactly correlation this measure yields relative quantities similar to correlation of all 40 original complex convolution values.

The measure is based on the similarity of the magnitudes of the frequency response; however, these values are weighted by the similarity of the phase angles. Thus, high scores are achieved only when both the magnitudes and the phase angles are similar.

The second of these measures is referred to as the "Magnitude" similarity measure. This measure will only compute the similarity of the energy of the frequencies. The phase information is not used in this measure. The result is a similarity measure, S_a, based on the covariance of the magnitudes.

$$S_a(J,J') = \frac{\sum_{j=1}^{N} a_j a'_j}{\sqrt{\sum_{j=1}^{N} a_j^2 \sum_{j=1}^{N} a_j'^2}} \tag{3.2}$$

While this method is tolerant of small displacements, it is completely unaffected by differences in phases. Thus, it measures the energy of the frequency responses but is unaffected if the frequencies are out of phase. Because the similarity is completely unaffected by phase, the measure can be easily confused and may respond to an incorrect spatial feature.

Each of the first two measures has its own advantages. S_ϕ will correctly respond to the phase information in the images. Because the phase information changes rapidly with the displacement, the measure will have a low similarity if two jets are compared that come from a similar landmark jet but are displaced by a small amount. In this case, the S_a similarity measure will produce high similarity under these conditions. It will, however, produce some false positives because it ignores the phase information.

This final equation attempts to correct for small displacements in the phase similarity measure. The equation will estimate the similarity as if J' was extracted from a displacement $\vec{d}$ from its current location. The equation uses the displacement correction method from Section 2.4 to approximate the value of the jet under a small displacement. This new solution retains the phase information and can compensate for small displacements. The phase similarity plus displacement correction yields the following equation :

$$S_D(J,J',\vec{d}) = \frac{\sum_{j=0}^{N} a_j a'_j \cos(\phi_j - (\phi'_j + \vec{d}\cdot\vec{k_j}))}{\sqrt{\sum_{j=0}^{N} a_j^2 \sum_{j=0}^{N} a_j'^2}} \tag{3.3}$$

This similarity measure is based on both the magnitude and phase components of the coefficients and can compensate for the differences in phases. Unfortunately the vector $\vec{d}$ is undefined. The next section discusses four methods to determine a value for $\vec{d}$.

3.5 Displacement Estimation

To properly refine the estimated landmark locations, the algorithm relies on the model jets that are stored in the bunch graph. The obvious way to find a landmark location would be to extract a novel Gabor jet from

every point in the image and compare these jets with the models. The pixel whose jet best matches one of the model jets indicates the correct location of the landmark.

Much of the computation of the algorithm is in Gabor jet convolutions. Computing a Gabor jet for every pixel in an image takes hours[1]. Because landmark locations in an image are always contained in a small region, it is only necessary to search a small part of the image. The algorithm can also save time by using the S_D function, which allows the algorithm to estimate the similarity under small displacements. The displacement estimated similarity measure can greatly reduce the number of Gabor wavelet convolutions that need to be computed by extracting one jet to represent the region near the landmark. Therefore, the landmark locations can be found by extracting one jet in the region close to the landmark and then searching that local area for a maximum using the S_D similarity measure.

The obvious way to find the maximum of the function S_D is to use a direct solution method. The goal of this method is to maximize the function S_D with respect to a displacement in two dimensions, i.e. $\vec{d} = [u, v]$. We would first take the partial derivative with respect to u and with respect to v, set these to zero, and then find values of u and v that together satisfy both equations. Computing the partial derivatives of the function is simple. The resulting function is a sum of 40 distinct sinusoids of various amplitude and frequency. Unfortunately, there is no obvious and succinct analytic method to find the zeros of the these two functions. For these reasons, alternative methods for finding the optima of this function have been investigated.

The first and simplest of these methods uses a grid sampling search strategy to determine the maximum of the S_D function. The algorithm searches an evenly spaced grid around zero displacement. At each point of the grid the S_D function is evaluated. The displacement that produces the maximum similarity is the optima of the function. The primary advantage of this method is that it searches the entire space to determine the maximum of the function.

The second method was adopted from the Bochum/USC algorithm [15]. This method uses knowledge of the search space to predict the location of the maximum. Instead of directly solving for the optima of the original similarity function, S_D, the method directly solves for the optima of an approximation of S_D.

This uses a two term Taylor expansion to approximate the cosine terms in the similarity function[2]. By replacing the cosine terms in the equations with the Taylor expansion, the algorithm approximates the similarity

[1] Individual jet extraction is relatively quick (on the order of 30 jets per second on modern processors). Each image has 16,304 pixels. If a Gabor jet is extracted from each pixel in the image the algorithm would have to compute 1,310,720 convolutions.

[2] Also known as the small angle approximation.

function immediately surrounding zero displacement. It logically follows that if the similarity function has a maximum nearby, its approximation will also have a similar maximum. The approximation takes the form:

$$\begin{aligned} \cos(\theta) &\approx 1 - \frac{1}{2}\theta^2 \\ S_D(J, J', \vec{d}) &\approx \frac{\sum_{j=1}^{N} a_j a'_j [1 - 0.5(\phi_j - \phi'_j - \vec{d}\vec{k}_j)^2]}{\sqrt{\sum_{j=1}^{N} a_j^2 \sum_{j=1}^{N} a_j'^2}} \end{aligned}$$

It is possible to set $S_D(J, J', \vec{d})$ to zero and solve for $\vec{d}$ directly using this approximation. The displacement vector is calculated with the following equation[3]:

$$\begin{aligned} \begin{pmatrix} d_x \\ d_y \end{pmatrix} &= \frac{1}{\Gamma_{xx}\Gamma_{yy} - \Gamma_{xy}\Gamma_{yx}} \times \begin{pmatrix} \Gamma_{yy} & -\Gamma_{yx} \\ \Gamma_{xy} & \Gamma_{xx} \end{pmatrix} \begin{pmatrix} \Phi_x \\ \Phi_y \end{pmatrix} \\ \Phi_x &= \sum_j a_j a'_j k_{jx} (\phi_j - \phi'_j) \\ \Gamma_{xy} &= \sum_j a_j a'_j k_{jx} k_{jy} \end{aligned}$$

This method is very fast and works well when the true optimum is near. Because this method approximates the similarity function surrounding zero displacement, it produces poor estimates as the distance from the novel jets to the true location increases.

The final method performs a local search for the equation maximum. This method starts by estimating the similarity at zero displacement. It then estimates the similarity at each of its North, East, South, and West neighbors. The neighboring location with the highest similarity is chosen as the new center of the search. This process is iterated until none of the neighbors offers an improvement over the current location.

Each of these optimization methods can be used to create displacement estimation (DE) algorithms. There are five algorithms that will be studied here: DEGridSample, DEPredictiveStep, DEPredictiveIter, DEFixedLocalSearch, and DENarrowingLocalSearch. The success of each algorithm depends on the algorithm's speed and accuracy. A comparison of algorithm speed can be found in Appendix B.

Each DE algorithm searches a neighborhood around its current position. These neighborhoods are classified as follows:

Grid This is the neighborhood defined by the grid sampling method. Every location on the grid is searched.

Predictive The Bochum/USC based algorithm predicts the location of the maximum using a direct solution to an approximation of the similarity function.

[3]The value of the phase term should be constrained to $-\pi < \phi_j - \phi'_j < \pi$.

ALGORITHM	SEARCH STRATEGY	NEIGHBORHOOD
DEGridSample	Exhaustive	Grid
DEPredictiveStep	Single Step	Predictive
DEPredictiveIter	Iterative	Predictive
DEFixedLocalSearch	Iterative	Fixed
DENarrowingLocalSearch	Iterative	Narrowing

Table 3.1: Displacement Estimation Algorithm Strategies

Fixed This is one option for the local search. The neighborhood is selected using a fixed step size to neighbors.

Narrowing This local search starts with a large step size and slowly narrows the step size as the search progresses.

Each algorithm also uses different search strategies to traverse its neighborhood.

Exhaustive The grid search method searches every point on the grid surrounding zero displacement. This would be an exhaustive search.

Single Step The predictive step method takes only one step to the new location. The new location is assumed to be the optimum of the similarity equations. No further search is performed.

Iterative The local search methods and the predictive iteration methods use iteration to incrementally find better solutions. The local search methods iterate until there is no step that will improve the similarity function. The predictive iteration method iterates until the predicted optimum stops moving.

3.5.1 Displacement Estimation Grid Sample (DEGridSample)

DEGridSample is probably the simplest algorithm for estimating displacement. This algorithm uses grid sampling to determine the optimum of the similarity function. Pseudo-code for this algorithm is shown below.

```
J = ExtractJet(ModelImage,ModelPoint)
J' = ExtractJet(NovelImage,EstimatePoint)
for d_x = −8; d_x ≤ 8; d_x+=0.5
   for d_y = −8; d_y ≤ 8; d_y+=0.5
       RefinedPoint = Maximum(S_D(J,J',d⃗))
```

```
    end
  end
```

The algorithm searches a 16 by 16 pixel grid around the novel jet for the best estimated similarity. The displacement that produces the best similarity is the estimated displacement of the novel jet. In this way the algorithm effectively estimates the distance from the novel jet to the true location of the landmark. This version of DE algorithm is very good at finding the displacement of a novel jet; however, it is very slow.

3.5.2 Displacement Estimation Predictive Step (DEPredictiveStep)

This algorithm follows the method presented in Bochum/USC algorithm[15]. This method has been used successfully in the algorithm and is a good starting point for comparison with other methods of solving for the vector $\vec{d}$. A disadvantage of this method is that it does not account for the periodic nature of the similarity function and it is only capable of accurately estimating very small displacements. The method has the advantage that it is very fast because it solves for the displacement directly. All of the other methods tried at CSU use iteration or multiple evaluations of the similarity function to determine a maximum. For this reason this method is at least 3 times faster than the others.

3.5.3 Displacement Estimation Predictive Iteration (DEPredictiveIter)

This algorithm solves for the displacement in the exact same way as DEPredictiveStep; however, the algorithm iterates to further refine the estimate. The algorithm starts with a displacement of (0,0). The maximum of the Taylor expansion of S_D is computed and that estimate becomes the new estimate of $\vec{d}$. The Taylor expansion is recomputed around this new point and is used to refine the estimate. This process is repeated until there is very little change or the algorithm has iterated 10 times.

3.5.4 Displacement Estimation Fixed Local Search (DEFixedLocalSearch)

DEFixedLocalSearch uses a local search method to find an optimum. This method performs a “smart” search of the same space searched in DEGridSample. It appears to be almost as robust as DEGridSample while offering a significant performance advantage.

The DEFixedLocalSearch method starts its search with no displacement. For each iteration the method computes S_D for the north, east, south, and west neighbors at 0.5 pixel displacements. The best neighbor is selected and the process is repeated from that location. The method quits when none of the neighbors are

Figure 3.2: Displacement Experiment Novel Image.

better than the current location. The method is also limited to 50 steps to narrow the search area and increase the speed of of the search.

3.5.5 Displacement Estimation Narrowing Local Search (DENarrowingLocalSearch)

This method implements a local search very similar to DEFixedLocalSearch; however, it starts with a step size of 1.0 pixel and reduces the step size by half when no better solution can be found. The method stops at a step size of 0.125.

3.6 Displacement Estimation Analysis

The experiment used to analyze the jet based displacement estimation is very similar to the experiment used to analyze wavelet phase estimation in Section 2.4. It starts by extracting a model jet from the tip of the nose of the image in Figure 3.2. For each pixel in the image, a novel jet is extracted and both jets are used to estimate the displacement of the novel jet from the tip of the nose.

It is expected that the displacement estimators will function well when the novel jet was extracted very close to the model. As the distance to the model increases, information overlap between the model jet and the novel jet will decrease. For this reason, displacement estimation should be more difficult as the distance increases.

This first experiment tests the displacement estimation methods under ideal conditions. When refining a landmark location in an image, a model jet is needed that serves as a template for the landmark. In the EBGM algorithm, the model jet often comes from a different image. For this experiment conditions are ideal because the model jet for the nose is the best possible representation for this feature because it came from the

same image. The next section will test the displacement estimation methods under similar conditions to the complete EBGM algorithm.

Figure 3.3 shows the estimated displacement for every jet around the nose tip for the four displacement estimation algorithms. The location (0,0) represents the point from which the model jet was extracted. Each vector corresponds to the estimated displacement based on novel jets extracted from a grid around the landmark location. All of the displacement estimation methods perform rather well. There is a clear point of attraction corresponding to the location of the model jet. The vector field for DEPredictiveStep shows that all the vectors around the true landmark locations point in the right direction; however, they are not all of the correct length. The other methods show that the displacement estimators are doing a much better job at estimating the locations of the landmarks. For the region surrounding the landmarks, all of the vectors have the correct direction and the correct length.

These results are shown in a different format in Figure 3.4. In this plot the x axis is the Euclidean distance between the model jet and the novel jet, i.e. the distance from the origin to the start of the vectors in the previous figure. The y axis, corresponding to error, is the distance between the estimated location of the tip of the nose and the manually selected location of the tip of the nose, i.e. the distance from the tip of the vectors in the previous figure to the origin. The scatter plots show that the estimation method has very little error when the novel jets are close to the true location. As the jets are extracted from points further away, the error increases. The red line is the average of the points.

The point clusters presented in these plots correspond directly to the results presented in Figure 3.3. The DEPredictiveStep plot shows that there if very little error in the estimation method when the novel jet is extracted within two pixels of the model location. As the novel jet moves further from the center the error increases because the displacement estimation method estimates the correct direction but not the correct distance.

The DEPredictiveIter graph shows a strong cluster with zero error corresponding to the left, bottom, and right of its corresponding vector plot in Figure 3.3. The second cluster corresponds to the top of the vector plot where the displacement estimation method did not find the correct displacement.

Finally, the local search plots show that there is an attractor at the correct location, as well as a location nine pixels from the model point that does not show up in Figure 3.3. In fact, attractors tend to occur in a grid like pattern at a spacing of 8 pixels for all displacement estimation methods. It is most likely an effect of the periodic nature of the similarity equation S_D.

The graphs show that most of the algorithms work well up to six pixels. At six pixels the algorithms become

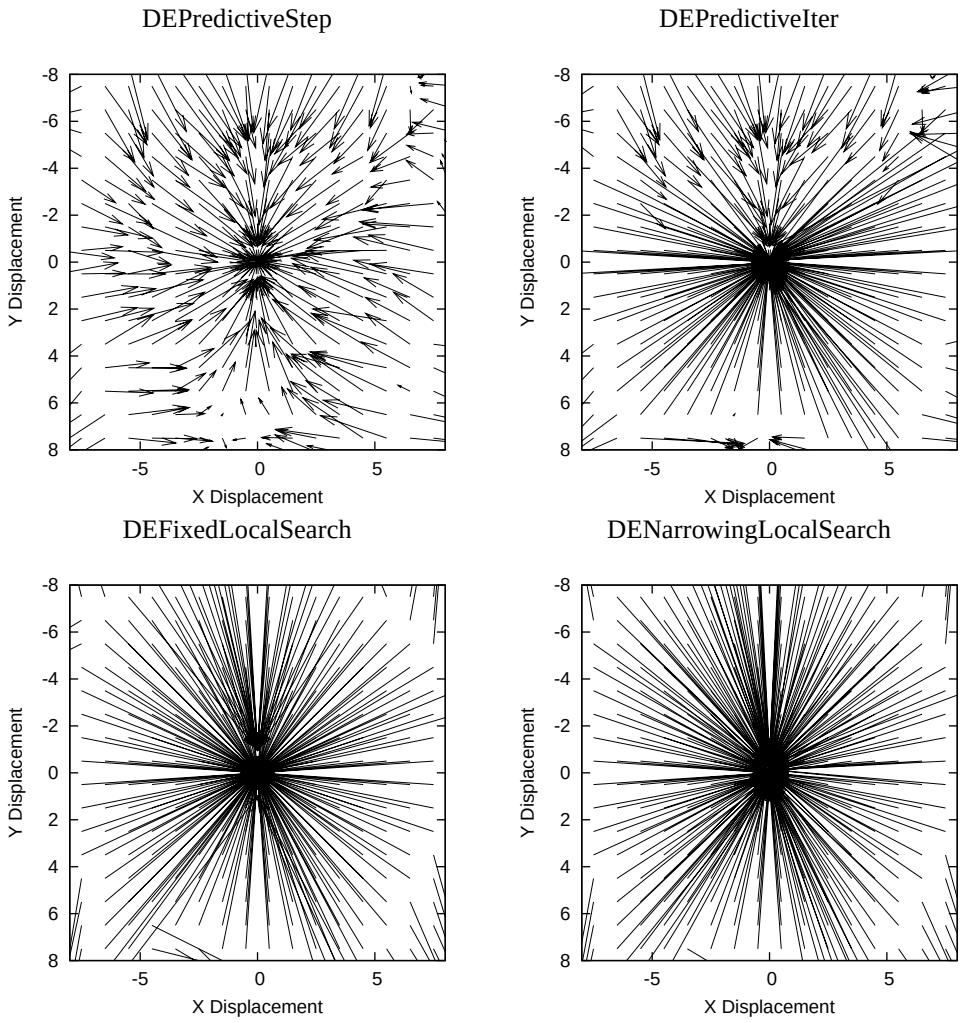


Figure 3.3: Vector Fields for Displacement Estimation Methods.

attracted to other local optima in the similarity function. By nine pixels the distribution the displacement estimation methods fail.

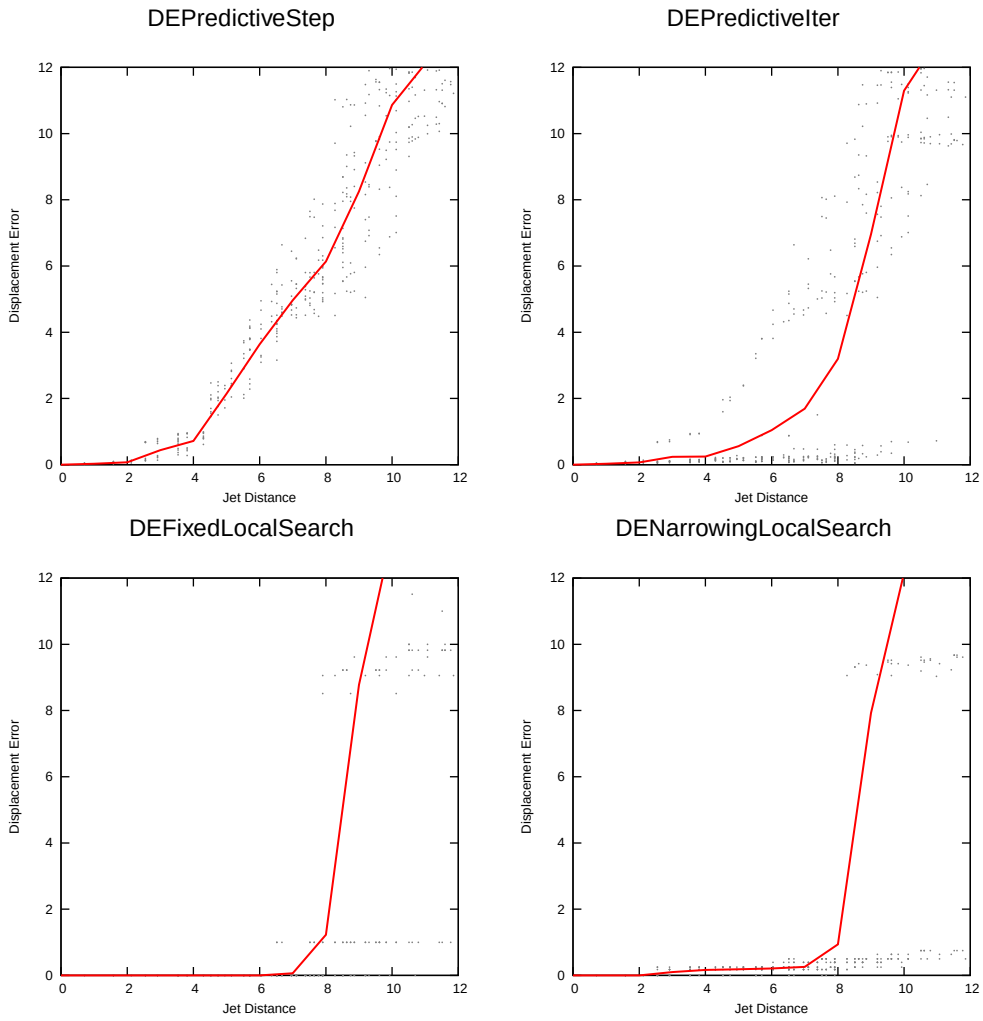


Figure 3.4: Displacement Estimation Analysis - Nose Tip

Figure 3.5 shows the cumulative results for 17 landmark locations. These points are all inside the face and do not include points around the edge of the head. The distributions are very similar to the ones in the previous figure. This shows that the methods perform in a similar way for all landmark locations.

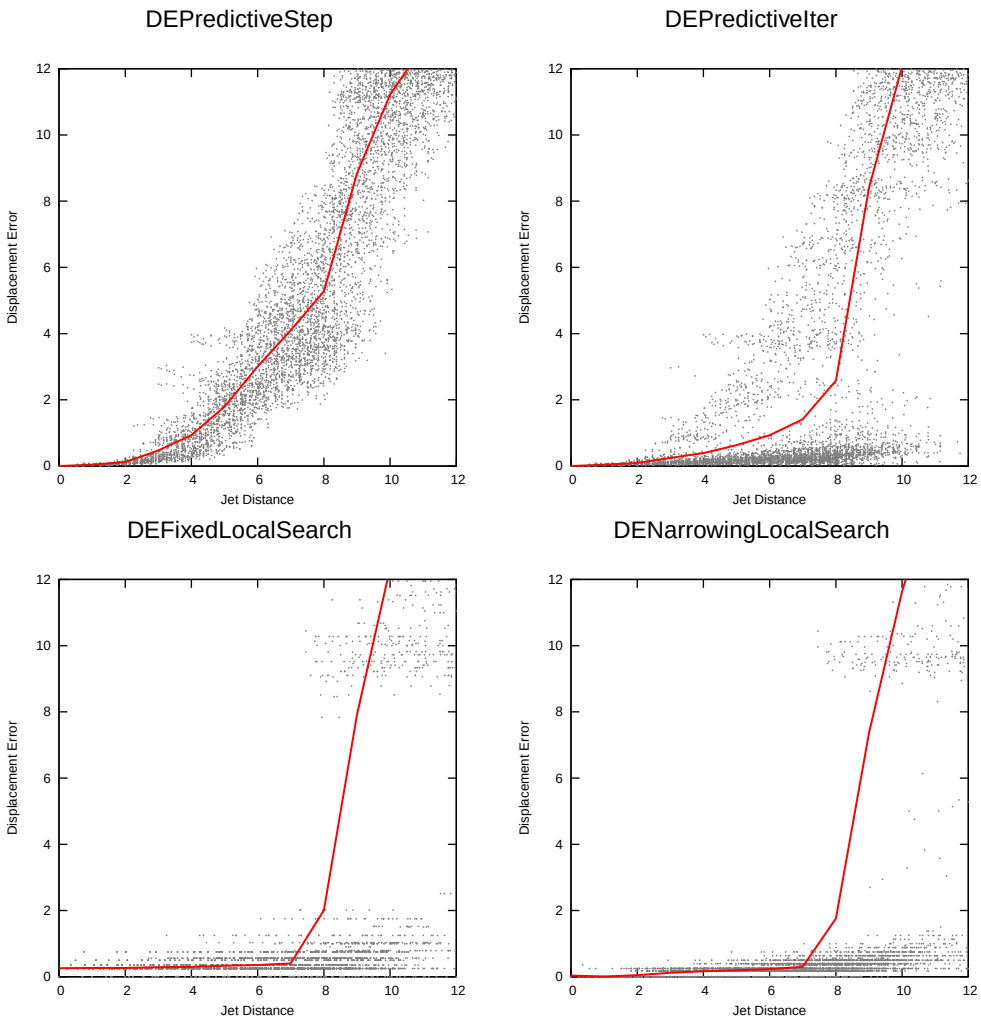


Figure 3.5: Displacement Estimation Analysis - 17 Internal landmark locations.

3.7 Bunch Graph Estimation Analysis

The CSU EBGM algorithm does not use one model jet for displacement estimation. Instead it uses many model jets that are stored in a bunch graph. This section will perform the same experiment as the previous section, except the model jet will be selected from a bunch of 70 jets in the same way as the complete algorithm.

When locating landmarks on a novel image, it is necessary to have a good model jet for each landmark. Each bunch has a variety of jets that can be used for this purpose. When a new jet is extracted from an image, it is compared to each model jet in the bunch. The model jet with the highest similarity is selected as the model for the localization process.

There are many models stored in a bunch graph. In theory, this gives the localization process a large variety of jets to choose from. For example, if the algorithm is trying to locate a point corresponding to the mouth of someone with a beard, it should use a jet that comes from a model image of someone with a bearded mouth. By selecting the jet with the highest similarity, the algorithm should select a jet that will produce a good displacement estimate.

To test the ability of the bunch graph localization process, an experiment similar to the one in the previous section was conducted. Instead of extracting a model jet from the tip of the nose, the model jet is selected from a bunch.

For each pixel near the landmark a novel jet is extracted. The similarity is computed with every model jet in the bunch using the S_D similarity function. The jet with the highest similarity to the novel jet is selected to serve as the model. The experiment proceeds as before, to test the accuracy of the displacement estimation method.

Figure 3.6 show the displacement fields generated for the nose tip. These results are typical for nose tip landmarks. All of the methods estimate that the location of the landmark is about three pixels above the manually selected point. In fact, offsets from the manually selected locations seem to be norm instead of rare occurrences.

Figure 3.7 is also consistent with this data. The error figures clearly show that all of the methods find a point about 3 pixels away. I believe that there are two error sources that cause this type of effect: small image differences and error in manually selected landmark locations.

The first reason is that the jets now come from different images. In the new experiment, the model jet is extracted from a completely different image. Both the novel and model are similar because they are both from human faces. There are, however, differences in the lighting, facial dimensions, expressions, etc. that will cause the jets to be slightly different. These subtle differences will, of course, cause errors in the estimated locations of the landmarks.

A second source of error is in the manually selected landmark locations. Both images use different manually selected landmark locations. The manual selection process is subject to human error. For example, even if the same image is presented to the human a second time it is doubtful that he would select the exact same pixel. If there is a “true” landmark location on the face, the manually selected point will naturally drift around that point. This drift will cause some error in the landmark localization results.

For example, assuming that both the model image and novel image have noses that are very similar, the displacement estimation method should accurately measure the displacement of novel jets extracted from

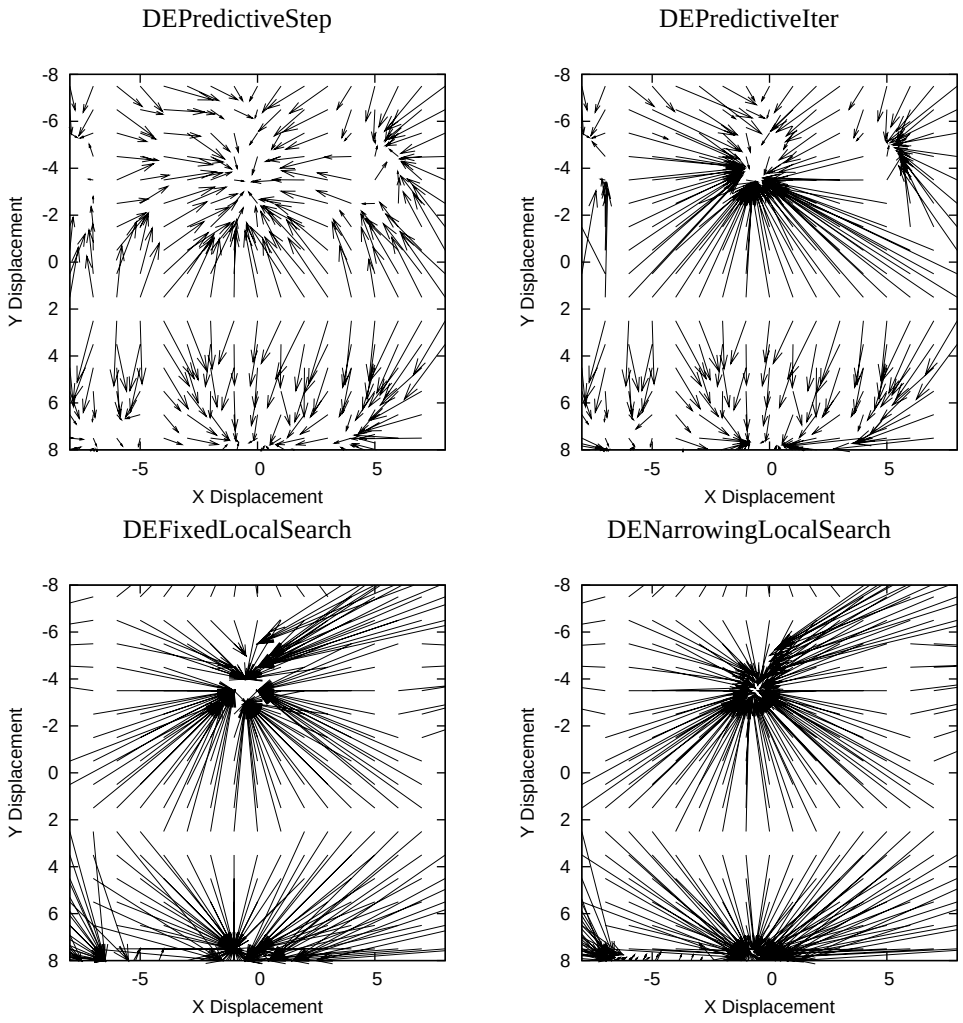


Figure 3.6: Vector Fields for the displacement estimation methods.

that region. Also assuming that manually selected points are always placed on the exact same point on the nose, the “true” location, the experiment results for the landmark should be very similar to the experiments conducted in the previous section.

A second possibility is that the manually selected point can drift around the true location. If the model jet was extracted from a location one pixel above the true location, it is expected that the estimated displacement of the novel jet would be one pixel above the true location in the novel image. The results shown here also depend on the manually selected landmark locations in the novel image that are used to calculate error. If the point selected for the novel image was selected one pixel low, the error reported for the experiment would then grow to 2.0.

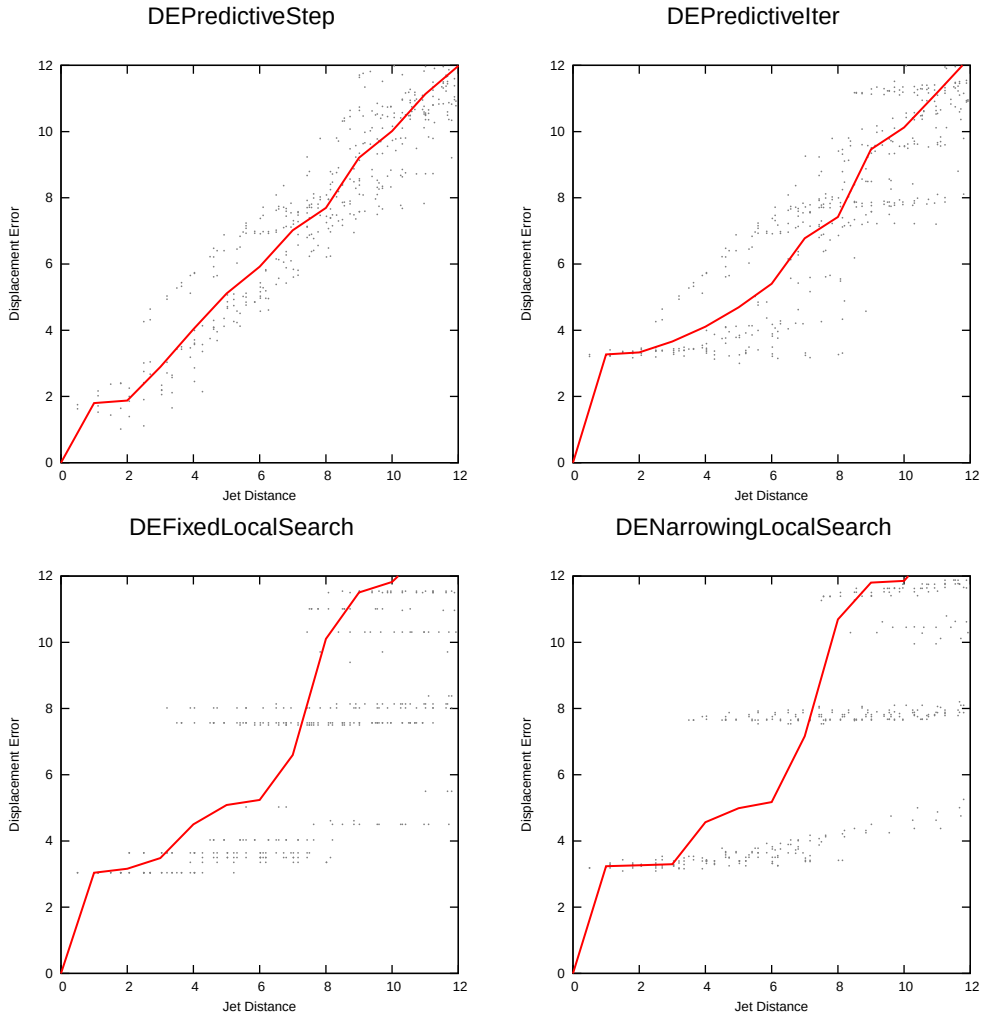


Figure 3.7: Displacement Estimation Analysis

Figure 3.8 shows the combined error for the 17 internal landmark locations. These plots show the super

position of 17 plots like Figure 3.7. It is difficult to see what is going on. The most obvious conclusion is that there is quite a bit of difference between the manually selected landmark locations and the points of attraction found by the bunch graph. Looking at the DEPredictiveIter and DENarrowingLocalSearch, it is possible to discern bands corresponding to attractors, as seen in the previous figure. DEFixedLocalSearch should also have such bands but it is difficult to see them because the discrete steps that show up are very distracting.

The means are also very interesting. All of the mean lines start out with a slope near zero. This low slope indicates that there is a strong attractor point. This would indicate that even though the DE methods are not attracted to the manually selected points, they are clearly attracted other points nearby. The mean line suggests that the local search methods have a larger basin of attraction around the optimal points.

It is difficult to tell from Figures 3.7 and 3.8 if DEPredictiveStep is working at all. Fortunately, Figure 3.6 shows that the estimator is attracted to a point; however, it is not estimating the correct distance to that point. Clearly these results look much worse than the ideal results shown in the previous section.

The results shown here indicate that the bunch graph estimation process is not as reliable as one might hope. It is impossible to make any judgments as to which estimation method work best in the final algorithm. The local search methods appear to be strongly attracted to the optima; however, the predictive methods appear have a lower average error when very close to the manually selected point.

Section 6.4 will examine each of these methods in the context of the complete algorithm to determine what effect each has on algorithm performance.

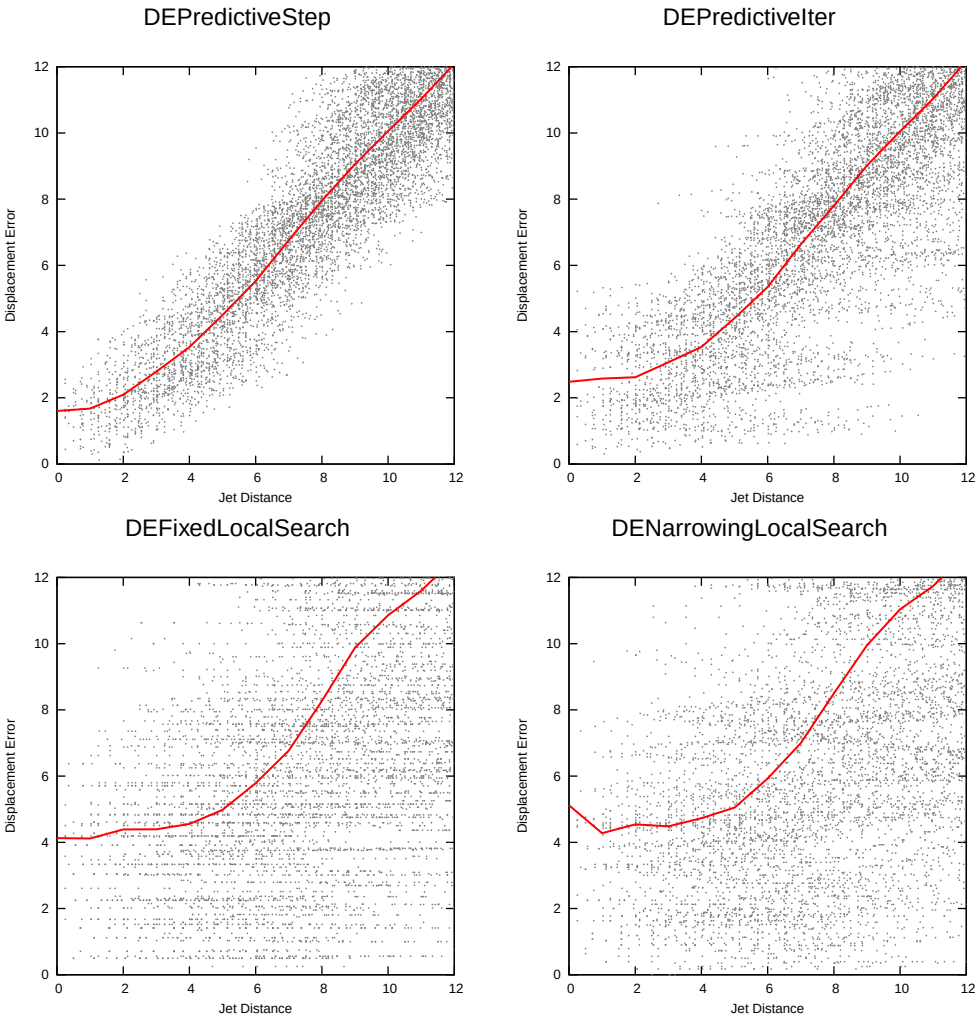


Figure 3.8: Displacement Estimation Analysis - 17 Internal landmark locations.

Chapter 4

Face Graphs

4.1 Face Graphs

A face graph is the structure used to represent a face. Every test image in the system has a corresponding face graph. The graph has one node for each landmark in the image. Every node in the graph contains the location of the landmark and a Gabor jet extracted at that point. The algorithm computes the similarity of two faces using the data stored in the face graphs. Face graphs are compared in order to compute the similarity between images. Similarity can be computed as a function of the landmark jets, landmark locations, or both. The landmark jets and landmark location similarity are discussed in this chapter.

A face graph also contains jets that were extracted at locations interpolated between landmarks. For example, there are no landmark locations on the cheeks or forehead. These regions include much of the surface area of the face. The Bochum/USC algorithm sampled these locations, as well as other interior face locations, by extracting jets from interpolated locations between landmarks. To approximate this aspect of the Bochum/USC algorithm, the CSU system extracts jets from the midpoint of every edge in the graph.

A face graph for an image consists of 25 landmark locations, 25 landmark jets, and 55 interpolated jets. This structure is useful for two reasons. First, the face graph contains information about the images that is useful for recognition. Second, the face graph provides a form of compression that allows recognition to be performed rapidly.

4.2 Landmark Jet Similarity

The landmark jet similarity of the entire face graph is defined as the average of the similarity of the Gabor jets. There are many options for computing the similarity of the jets, including magnitude only, phase, or

displacement compensated Gabor jet similarity. These similarity measures are all computed as:

$$L_{jet}(G, G') = \frac{1}{80} \sum_{i=0}^{80} S_x(J_i, J_i')$$

where S_x is a specific method for computing the similarity of Gabor jets, and J_i and J_i' are jets from the ith landmarks, from the graphs G and G'. Seven choices for S_x are described below.

FGMagnitude This measure is the average similarity of all of the face graph jets based on the S_a Gabor jet similarity function.

FGPhase This measure is the average similarity of all of the face graph jets based on the S_ϕ Gabor jet similarity function.

FGGridSample This measure is the average similarity of all of the face graph jets based on the S_D Gabor jet similarity function. DEGridSample is used to estimate the displacement of two jets.

FGPredictiveStep This measure is the average similarity of all of the face graph jets based on the S_D Gabor jet similarity function. DEPredictiveStep is used to estimate the displacement of two jets.

FGPredictiveIter This measure is the average similarity of all of the face graph jets based on the S_D Gabor jet similarity function. DEPredictiveIter is used to estimate the displacement of two jets.

FGFixedLocalSearch This measure is the average similarity of all of the face graph jets based on the S_D Gabor jet similarity function. DEFixedLocalSearch is used to estimate the displacement of two jets.

FGNarrowingLocalSearch This measure is the average similarity of all of the face graph jets based on the S_D Gabor jet similarity function. DENarrowingLocalSearch is used to estimate the displacement of two jets.

The Bochum/USC algorithm as presented in [15] selected the FGMagnitude similarity measure over FGPredictiveStep. This measure was chosen because it performed better on a small portion of the FERET fb probe set. Some early experiments with the CSU implementation also confirmed those results using the entire fb probe set. Tests using the other FERET probe sets revealed that the FGPredictiveStep out performed the magnitude measure on a greater variety of problems.

4.3 Euclidean Geometry Similarity (GeoL2)

Another method for determining similarity is based on the positions of the landmarks. If the landmark locations are placed accurately on the landmarks in the image, then the positions of these locations can be

used to identify the person in the image. The presumption is that images of the same subject will have very little difference in the landmark location, and images from different subjects should have large differences in their respective landmark locations.

One of the simplest ways to compute the similarity between two sets of landmark locations is to compute the sum of the Euclidean distances between the locations. Since the images have already been aligned using the eye coordinates of the subject, the landmark locations should therefore be aligned relatively well, and Euclidean distance can be used to measure similarity.

$$L_{GeoL2}(G_1, G_2) = \sqrt{\sum_{i=0}^{25} (x_{i1} - x_{i2})^2 + (y_{i1} - y_{i2})^2}$$

where (x_{i1}, y_{i1}) is the location of the ith landmark in the first face graph, and (x_{i2}, y_{i2}) is the location of the landmark in the second face graph.

4.4 Least Squares Geometry Similarity (GeoLeastSquares)

During face normalization eye coordinates are used to align the faces. If the eye coordinates are off by a small amount, all of the locations in the face will be off by small amounts. If the landmark locations are from the same image normalized with slightly different eye coordinates, it can be assumed that there will be a two dimensional transformation to the image such that the locations will line up exactly. That transformation will bring the two images into alignment.

In practice, the algorithm will need to do this with different images from the same subject. Because the images are taken from slightly different angles and the two images have different expressions, it is doubtful that the two sets of locations will line up exactly. In theory locations from the same subject should line up better than locations from different subjects.

To compute a similarity for two sets of locations based on this best fit alignment, it is necessary to first compute the two dimensional transformation that will produce the best fit. The first step of this process is to analyze the problem and limit the search space of the transform.

During registration, each image is subject to uniform scale, rotation, and translation. This is the type of transformation that needs to be determined. These transformations are used to properly align the images in such a way that the relative geometry of the image is preserved. It is also necessary to avoid transformations that would distort this geometry because it could lead to similarity scores that are too high for images of different subjects. Some examples of these transformations are non-uniform scale, shear, or perspective transformations.

The solution should restrict the transformation terms to scale (s), translation (dx and dy), and rotation (θ). This transformation produces a system of equations of the following form:

$$\begin{aligned} x'_{i1} &= x_{i1} s\cos(\theta) - y_{i1} s\sin(\theta) + dx \\ y'_{i1} &= x_{i1} s\sin(\theta) + y_{i1} s\cos(\theta) + dy \end{aligned} \tag{4.1}$$

While these equations will produce the transformation needed, they are not linear in terms of s, dx, dy, and θ. Therefore this system cannot be solved using a linear least squares method. To convert this into a linear problem, a simple substitution is made:

$$\begin{aligned} a &= s\cos(\theta) \\ b &= s\sin(\theta) \end{aligned}$$

With these substitutions, the transformation equations become linear:

$$\begin{aligned} x'_{i1} &= a x_{i1} - b y_{i1} + dx \\ y'_{i1} &= b x_{i1} + a y_{i1} + dy \end{aligned} \tag{4.2}$$

The transformation can be determined as the least squares solution to the following system:

$$\begin{bmatrix} x_{11} & -y_{11} & 1 & 0 \\ y_{11} & x_{11} & 0 & 1 \\ x_{21} & -y_{21} & 1 & 0 \\ y_{21} & x_{21} & 0 & 1 \\ \vdots & \vdots & \vdots & \vdots \\ x_{N1} & -y_{N1} & 1 & 0 \\ y_{N1} & x_{N1} & 0 & 1 \end{bmatrix} \begin{bmatrix} a \\ b \\ dx \\ dy \end{bmatrix} = \begin{bmatrix} x_{12} \\ y_{12} \\ x_{22} \\ y_{22} \\ \vdots \\ x_{N2} \\ y_{N2} \end{bmatrix}$$

After solving for a, b, dx, and dy, the original locations can be transformed into the best fit coordinate system using Equation 4.2. It is not necessary to solve the system for s and θ, but if these values are needed, they can be easily computed as:

$$s = \sqrt{a^2 + b^2}$$

$$theta = \arctan(\frac{b}{a})$$

After the locations are transformed, the similarity is computed using the L2 distance measure:

$$L_{GeoLS}(G_1, G_2) = \sqrt{\sum_{i=0}^{25} (x'_{i1} - x_{i2})^2 + (y'_{i1} - y_{i2})^2}$$

where the primed coordinates are the transformed landmark locations from the first face graph.

4.5 Corrected Geometry Distance Measures

The least squares method presented in Section 4.4 is based only on the geometry of the face graph represented by the landmark locations. Each face graph contains a jet for each landmark location. It is possible to use these jets to make adjustments to the landmark locations immediately before computing the geometry similarity.

Sections 3.6 and 3.7 analyze the displacement estimation technique under two different conditions. The first was that both jets came from the same image, and the results showed that the displacement estimation methods work very well under those conditions. The second tested the use of the bunch to estimate locations. In this case the results were much worse.

The technique uses the Gabor Jets in the face graphs to produce small corrections to the landmark locations. Assuming the two face graphs being compared are from the same subject, the jets in the two face graphs should be very similar. The jets may be slightly out of phase because the bunch graph localization used to extract the jets is not perfect. The displacement estimation based on these two jets may also be more accurate than using a jet from a different subject, selected from a bunch.

For each landmark location in the graph, a correction is applied by estimating a displacement between the corresponding jets for each face graph. This correction is applied for each image pair, immediately before computing the GeoLeastSquares similarity. If the face graph is of the same subject, the correction will in most cases increase the geometry similarity. If the face graphs are not from the same subject, the correction should have very little effect on the geometry similarity.

Four geometry similarity measures are based on this correction method. Each method is based on a deferent displacement estimation method from Section 3.5:

GeoLeastSquaresPS uses the DEPredictiveStep method.

GeoLeastSquaresPI uses the DEPredictiveIter method.

GeoLeastSquaresFLS uses the DEFixedLocalSearch method.

GeoLeastSquaresNLS uses the DENarrowingLocalSearch method.

4.6 Similarity Measure Performance

This chapter has introduced face graph similarity measures for the EBGM algorithm. Experiments presented in Chapter 6 have found that the similarity measures based on the Gabor Jets are far superior to measures

based on the geometry of the face. The experiments have also found that measures that utilize the displacement estimation have an advantage over the uncorrected measures.

In Chapter 6, the similarity measures will be directly compared. Each class of similarity measure (jet based and geometry based) will be tested independently because the difference in performance of these classes is substantial.

Chapter 5

Algorithm Implementation

In the CSU Face Recognition System [4], an "algorithm" is responsible for taking normalized imagery and computing a distance matrix that specifies the similarity between every image in the database. Landmark localization is where the CSU EBGM algorithm officially begins, and the algorithm is concluded with similarity measurement. This chapter, however, will cover every part of the system from normalization to nearest neighbor classification. These two topics are handled by other programs in the CSU system but are still important parts of the complete "image identification" system.

Figure 5.1 shows the data flow for the CSU EBGM algorithm. The complete system has five data processing steps, and there are six intermediate data types. The EBGM algorithm is divided into three executables: landmark localization, face graph creation, and similarity measurement.

The output of the algorithm is a distance matrix which specifies the similarity between each image in the database. The CSU system assumes algorithms are structured as nearest neighbor classifiers where more similar images have smaller distances between them. The final parts of the complete system are analysis tools. The tools assess algorithm performance based on the distance matrix computed by the algorithm.

The complete identification processes is defined as follows:

Normalization: Prior to execution of each algorithm, the raw face imagery is processed to reduce the amount of image variation due to lighting and image registration. Normalization is performed by the program csuPreprocessNormalize. Special settings are used by the EBGM algorithm to improve the accuracy of the landmark localization process. See Section 5.1.

Landmark Localization: The purpose of the landmark localization process is to automatically locate landmarks in novel images. Landmarks are found using the program csuEBGMGraphFit. The output of this program is a file for each novel image that contains the automatically placed landmark locations

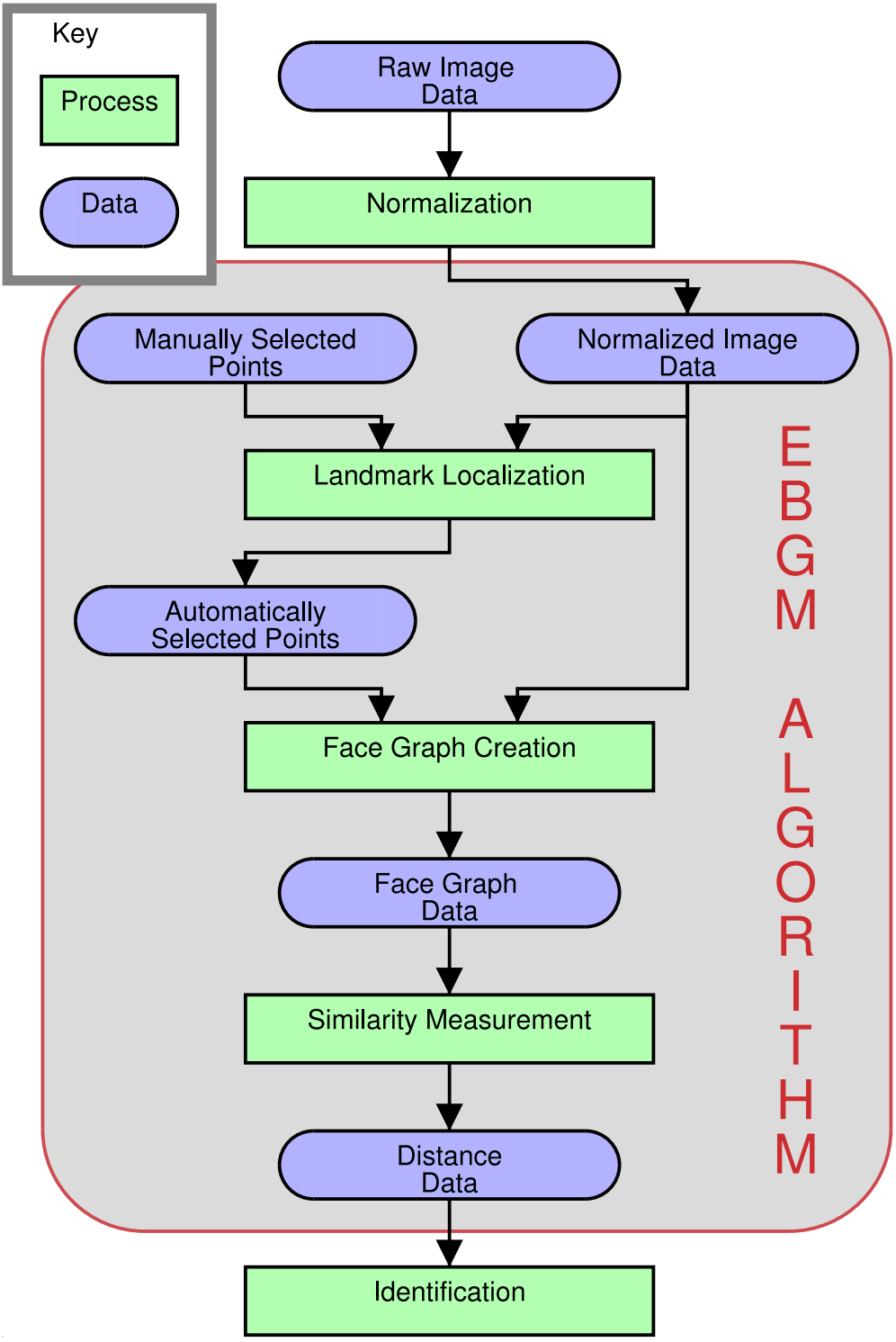


Figure 5.1: Data flow for the EBGM algorithm

for that image. This program has two high level parts: bunch graph creation and landmark localization. See Sections 5.2 and 5.3.

Face Graph Creation: The program csuEBGMFaceGraph uses the automatically placed landmark locations from the previous process and normalized imagery to create face graphs for every image in the database. See Section 5.4.

Similarity Measurement: The final step of the EBGM algorithm is to produce a distance matrix for the database. All of the analysis techniques for the CSU Face Recognition Evaluation System utilize the distance matrix to determine system performance. See Section 5.5.

Identification: In the CSU system identification is based on nearest neighbor classification. The system is not intended to serve as a real time face recognition system. It is intended to analyze the performance of face recognition systems based on well designed experiments. The tools used to analyze the algorithms are based on experiments done in the FERET evaluation [14] and work done at CSU. See Section 5.6.

5.1 Normalization

Although normalization is common to all CSU algorithms, the EBGM algorithm uses a customized normalization algorithm designed to improve localization performance. The standard CSU normalization process produces a 130X150 face image with a mask. As will be discussed in more detail shortly, it was found that the image mask disrupted the landmark localization process. To compensate, a new normalization process is used to produce images similar to those used in the Bochum/USC algorithm. Figure 5.2 shows the difference between standard normalization for the other CSU algorithms and the customized normalized images used buy the EBGM algorithm.

An early problem was that our system had difficulty locating landmarks on the face. We believe this problem was caused by the effects of a mask that was applied during the standard normalization. One of the first implementations of this algorithm used standard normalized images that are used for the other algorithms under development at CSU. During the normalization process, a mask is applied to the imagery to occlude the sides of the face. The top and bottom of the image was also used to occlude the subject's hairline and neck. We believe the sharp edges of this mask produced a large event in frequency space that confused the wavelet localization process.

To reduce this problem we modified the normalization process to eliminate the mask and reduce the edge effects of the image. Figure 5.3 presents an overview of the new normalization process adapted for the EBGM algorithm.

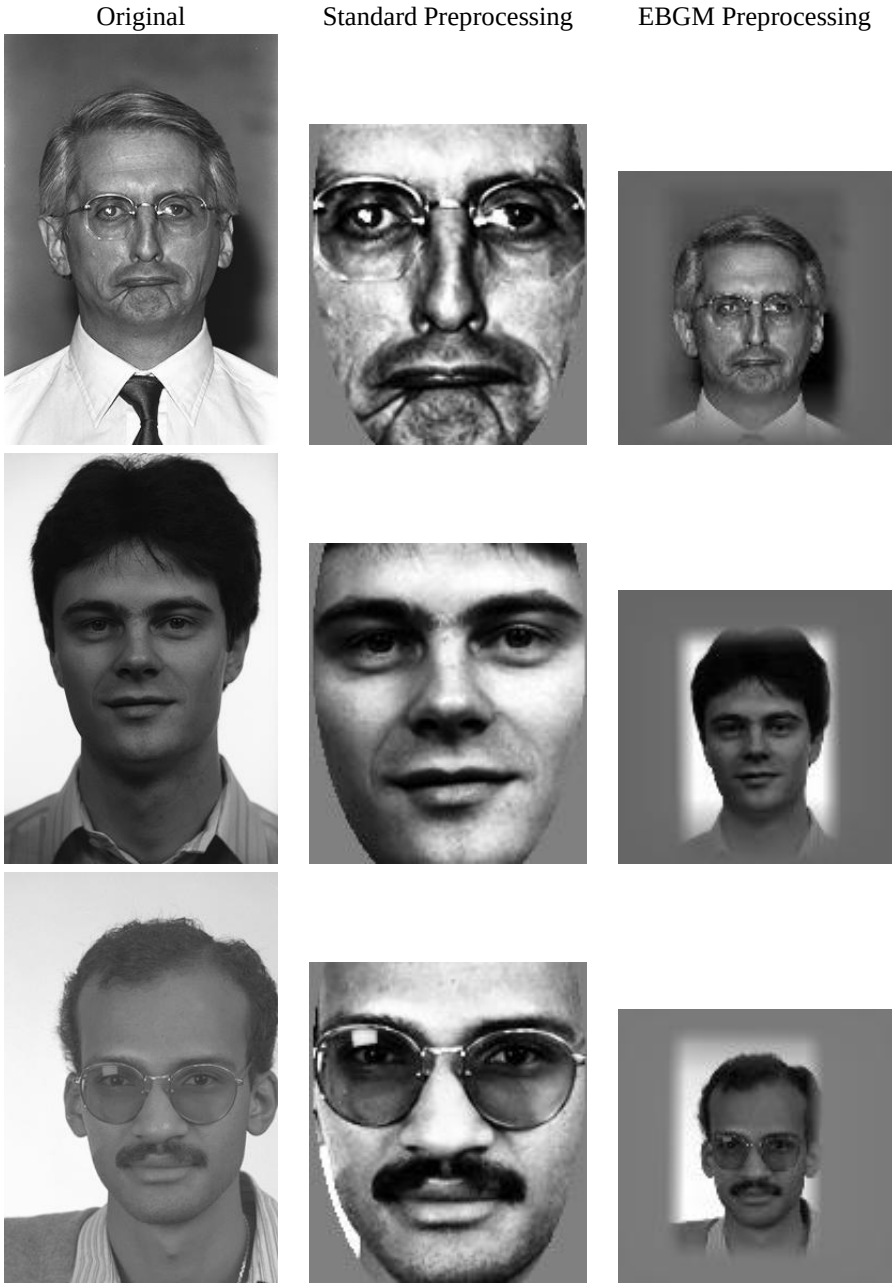


Figure 5.2: Comparison of the the three types of normalized images used at CSU. Original images (left) are scaled to half size. The normalized images are shown at the same scale.

1. Mean center the pixel values in the source image.
2. Linearly smooth the edges of the source image across a 30 pixel border.
3. Geometrically normalize the image such that the eye coordinates line up and the image is rescaled to 128x128 pixels.
4. Normalize the contrast of the new image with histogram equalization.
5. Mean center the pixel values of the new image and scale values to one standard deviation.
6. Linearly smooth the edges of the image across a 30 pixel border.

Figure 5.3: EBGM Face Normalization Process

Pixel locations outside the image are interpreted as having a value of zero, so the algorithm zero centers the pixel values of the source image to reduce the effect of the image border. This is done by subtracting the mean pixel value from every pixel in the image.

Second, the edge of the image is smoothed to reduce its effect in frequency space. Pixel values are weighted by $\frac{d}{30.0}$ where d is the distance from the edge. Pixels greater than 30 pixels from the edge are not modified. This smoothes the image edge from actual pixel values in the center of the image to zero at the edge of the image. These first two steps are necessary because in most FERET images the edges of the original full sized image are visible in the final normalized image. See Figure 5.2.

The image is then geometrically normalized to align the eye coordinates at pixels (52,64) and (76,64) and the image is rescaled to a size of 128 pixels on an edge. The algorithm then performs contrast and brightness normalizations. The edge smoothing process is applied again to this image to reduce any new edge effects introduced in the geometric and contrast normalizations.

5.2 Bunch Graph Creation (Landmark Localization)

A bunch graph is created by collecting model Gabor jets for every facial landmark and storing them in a graph-like structure. The CSU algorithm uses manually selected landmark locations to produce the bunch graph. The manually selected landmark locations are collected by a tool, with a graphical user interface, that allows a user to select points on an image that correspond to landmarks.

The output of the graphical tool is a set of files that contain the landmark locations for certain images. Each file corresponds to an image that was processed manually and contains a set of pixel locations that correspond

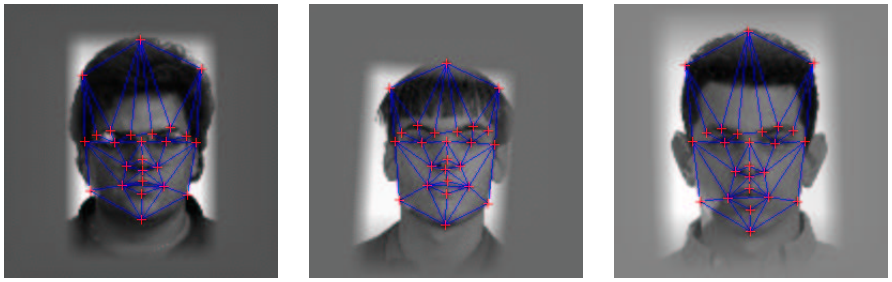


Figure 5.4: Sample model graphs.

to the landmark locations.[1]

To create the bunch graph, the algorithm loads each image that is used as a model. It then extracts a model jet from each manually selected landmark location and adds the jet to the appropriate bunch. Each node in the bunch graph contains a group of 70 Gabor jets corresponding to one landmark. The jet bunches can then provide jets to serve as model landmarks when locating landmarks in novel images.

One powerful aspect of the bunch graph is that each bunch contains many different examples of its particular landmark jet. For example, the algorithm can provide model jets of eyes with glasses, eyes that are closed, and eyes under different lighting. Furthermore, the algorithm can select jets from different subjects for each landmark on the face. For a particular face the algorithm can mix and match eyes, noses, and mouths such that each landmark has the best possible example and, therefore, the best possible localization.

Although the selection process seems straightforward, details tend to complicate the training process. The two most prominent details are selection of model images and selection of landmark locations. Most experiments presented in this thesis use 70 model images that were selected randomly from the CSU training partition of the FERET database. For each, 25 landmark locations were selected by hand around the eyes, nose, mouth, and the edge of the head.

The landmark locations and a list of model images used in this paper are included with version 5.0 of the CSU Face Identification Evaluation System, which is available through the CSU web site [1].

[1]These files can be found in the directory “ebgmdata/ManualLandmarkLocations/” from release 5.0 of the CSU Face Identification Evaluation System. [1]

5.3 Landmark Localization

The details of the landmark localization process were already discussed in Chapter 3. This section will present a high level discussion of how the process is implemented. Landmark localization has two parts. The first is to estimate the new location based on the locations of known landmarks in the image. The second is to refine that guess based on the phase information from a Gabor jet extracted from the estimated location.

The initial estimation method is described in Section 3.2. When a new image enters the system, it is first normalized using the eye coordinates. The eye coordinates are always transformed to line up in the same points. These points are a very good estimate of the locations of the eyes, and for this reason, the algorithm always locates the eye landmarks first.

To locate the eye landmarks the algorithm extracts a jet from each of the eye coordinate locations. It then uses the bunch graph and one of the landmark localization methods, from Section 3.5, to refine that estimate. It would be possible to uses the eye coordinates as ground truth; however, the eye coordinates are not as carefully located as the eye coordinates that were manually selected for building the bunch graph.

The algorithm locates points near the eye coordinates first, because the estimates of the eyes are very reliable. The algorithm then works radially outward until it reaches the edge of the head. For example, once the eye landmarks are located, the algorithm uses those points to estimate the location of the bridge of the nose and then refines that guess using the bunch graph and Gabor jet phase information. The next point placed is the left eye brow peak, and the location estimate is based on the previous three located points. This process continues for each landmark where every new estimate is based on all of the previously located points. The landmarks are localized in the following order.

1. LEye	6. LEyeBrowInside	11. CNoseBottom	16. LMouthCorner	21. LFaceEdge
2. REye	7. REyeBrowInside	12. LNoseBottom	17. RMouthCorner	22. RFaceEdge
3. CNoseBridge	8. LEyeBrowOutside	13. RNoseBottom	18. CTopHead	23. CChin
4. LEyeBrowPeak	9. REyeBrowOutside	14. CMouthTop	19. LTopHead	24. LJaw
5. REyeBrowPeak	10. CNoseTip	15. CMouthBottom	20. RTopHead	25. RJaw

(L = Left, C = Center, R = Right)

Once the algorithm has estimated the location of a landmark, it needs to refine that guess using a displacement estimation method from Section 3.5. First a jet is extracted from the estimated location. The jet is compared to every corresponding jet in the bunch graph using the chosen phase based similarity. The model jet with the highest similarity is selected and is used to estimate the displacement. Finally, the displacement is used to adjust the landmark location.

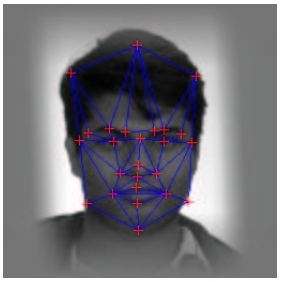
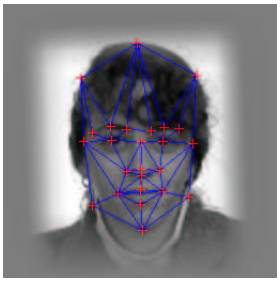
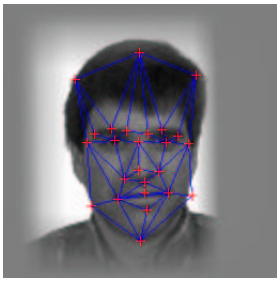
a.  b.  c. 

Figure 5.5: Automatically selected landmark locations.

Figure 5.5 shows a few samples of the output of the landmark localization process. Visual inspection of such graphs clearly shows that the localization process finds correct landmarks; however, individual graph nodes are sometimes slightly different from what I would consider correct placement.

Because I manually selected the points for the model images I feel that I should comment on the placement of these points. I notice the following problems with the images in the figure:

a. RJaw is too far right. LEyeBrowOutside is too far left. REyeBrowPeak needs to be Centered. CTopHead is too low. CNoseTip too high.

b. LJaw and RJaw misplaced. LEye too far left.

c. RJaw too far right. RNoseBottom and RMouthCorner too far up and right. CTopHead is too low.

With the exception of the jaw line for image "b." all of these mistakes seem to be minor. Many of them were difficult to spot.

The experiments conduced in Section 3.7 would indicate that every landmark location in these images would be off by approximately three pixels. Visual inspection of such images indicate that those errors are acceptable.

In the CSU implementation the bunch graph creation and landmark localization process are one executable. The program reads in model images and model graphs that are used to create the bunch graph. For each model image, jets are extracted from the manually selected landmark locations and are then added to the bunch graph. After the bunch graph is created, each novel image is loaded, and the landmark locations are found. For each image, a file is saved to disk that only contains the locations of the landmarks. Because the images are processed one at a time, the code is very memory efficient. The process can also be easily divided for parallel processing by running multiple executables on different sets of novel images. There is no need for thread or MPI support.

5.4 Face Graph Creation

The next section of the algorithm is face graph extraction. This part of the code is extremely simple. The algorithm extracts jets from the landmark locations placed in the previous program and stores those jets and their location in a face graph structure. The algorithm also has to extract jets from locations between landmarks. For each graph edge, the midpoint is determined, and a jet is extracted from that point and included with the face graph.

Even though the process is simple, it is included as a stand alone executable because computing the wavelet convolutions is computationally expensive. The executable reads in each image and the corresponding landmark locations, and extracts a jet for each landmark and interpolated location. The executable saves a file containing face graph information for each image. Like the previous executable, it is memory efficient and easily parallelized.

5.5 Distance Measurement

Chapter 4 discusses similarity measurement in detail. This part of the algorithm is responsible for computing a similarity matrix that contains the similarity values for every pair of face graphs in the database.

The CSU system assumes the use of a distance based nearest neighbor classifier for recognition. The algorithm, therefore, needs to output distance instead of similarity. In the distance based system, smaller values are more similar. Most of this thesis presents similarity measures such that larger values are more similar. To translate similarities into distances, the similarity values are negated when stored as a distance matrix.

Most of the experiments conducted at CSU are based on NIST's FERET database [13]. These experiments are based on a subset of 3368 frontal images. The distance matrix for this subset of the database would be a 3368X3368 element matrix, where element i, j corresponds to the similarity between the face graph of image i and image j. By computing every possible similarity value, the system can run the "algorithm" once to compute distances and then run as many virtual experiments as necessary based on that configuration of the algorithm.

An example of this concept is constructing the FERET cumulative rank recognition curves. Once the distance matrix is computed, the analysis tools can run four experiments for each of the fb, fc, dup1, and dup2 probe sets. The algorithm does not need to be run for each combination of probe and gallery.

One problem with computing the distance matrix is that all 3368 faces have to be compared to every other face graph in the database. The obvious way to compute this is to load all of the face graphs into memory

and iterate through the matrix computing one row at a time. Computing all distances for a single image at one time causes serious memory problems. Each face graph contains 80 jets with 80 convolution values each, and 25 landmark locations with 2 coordinates each. Using double precision floating point numbers, the entire database would take up a minimum of 165 Mb. The current implementation also keeps track of the original real and imaginary wavelet convolution values in addition to the Polar representation. Each jet therefore contains 160 values instead of just 80 and memory requirements are now 330 Mb.

If the machine has 256 Mb of ram, the computer would have to access swap for every computed row. To use swap and cache more efficiently, the algorithm recursively divides the matrix into blocks. At each level the matrix is split into four blocks: top left, top right, bottom left, and bottom right. Each block is computed in its entirety before proceeding to the next. The recursive limit on the block size is 8X8[2]. For each 8X8 block, the algorithm needs to access 16 face graphs. Because the matrix is split in a recursive manor, it can use cache, main memory, and swap effectively. In practice this method is much faster than computing similarity row by row on machines with small memories. The algorithm can also be more efficient than a real time system that might have to process one image at a time.

This executable produces a distance file for each image in the database. Each distance file contains the distance from that image to every other image in the database. Unfortunately this method uses a lot of disk space; however, the files are easy to read and it is possible to only load small parts of the matrix that are needed for each analysis.

5.6 Identification

CSU has provided a number of analysis tools for use with algorithm generated distance matrices. All of the tools operate under the assumption that face recognition algorithms are nearest neighbor classifiers. The difference between face recognition algorithms is how they compute the distance of the faces. Each algorithm in the CSU system computes the distance between every pair of faces in the database and outputs a distance matrix. The analysis tools can then read the distance matrix and compute statistics for a particular algorithm.

5.6.1 FERET Test

This thesis relies on three analysis tools distributed with the CSU system. The first of these tools generates standard cumulative rank match curves, similar to the ones used in the FERET test [12]. This test evaluates

[2]This parameter can be altered in the source.

the algorithms performance under a number of different conditions including expression, lighting, and time differences.

The FERET test is based on five sets of images from the FERET database.

fa This is the gallery set that is used for all experiments. The gallery contains images of 1196 subjects, one image each.

fb contains 1195 images that were taken in the same session but with a different facial expression.

fc contains 194 images from the same session as fa but with different lighting conditions.

dup1 contains 722 images that were taken later in time.

dup2 contains 234 images that is a hard subset of dup1.

5.6.2 Permutation Experiment

The permutation experiment [2] evaluates the recognition performance of an algorithm on a set of images. The images are a subset of 160 subjects from the FERET database that have four images per subject[3]. The permutation code reads in the distance files generated for the subset. For each subject, one image is randomly chosen for the probe and another image is randomly chosen for the gallery. The code then is able to compute a recognition rate based on the section of probe and gallery images.

Because the permutation code uses a full distance matrix, there is no need to run the algorithm for each choice of probe and gallery. The code can, therefore, run thousands of experiments based on different selections of probe and galleries in a few seconds. The sample distribution of recognition rates can then be analyzed to produce interesting statistics, such as:

- Mean
- Median
- Standard Deviation
- 95% Confidence Interval

[3]The list of images used for this experiment can be found in the file ”imagelists/list640.srt” from release 5.0 of the CSU Face Identification Evaluation System. [1]

Theses statistics can help to better understand the performance of an algorithm that is not evident from a single recognition rate.

5.6.3 Algorithm Comparison

This analysis is based on the permutation experiment [2]; however, it designed to directly compare two different algorithms using a paired test. This analysis randomly permutes the probe and gallery in the same way as the permutation experiment. For both algorithms (A and B) a recognition rate is computed. Instead of directly analyzing the recognition rates, the code analyzes the difference in those recognition rates: $D = R_A - R_B$. The analysis produces a statistical data on D including the mean, mode, and 95% confidence interval. In addition to these statistics, the probability that algorithm A will perform better than algorithm B is used to determine the statistical significance of any improvements between the two algorithms.

Chapter 6

Algorithm Configuration

6.1 Base Configuration

The base configuration for the CSU EBGM algorithm is taken from the Bochum/USC implementation presented in [15]. The configuration attempts to copy most aspects of the Bochum/USC algorithm including details on wavelets, model set selection, location refinement, and similarity measurement. This chapter tests these four different parts of the EBGM algorithm to determine the effects of alternative configurations.

Figure 6.1 shows the three main steps in the EBGM algorithm: Landmark Localization, Face Graph Extraction, and Similarity Measurement. The blue boxes also show which parts of the algorithm are tested in the study. Within the blue boxes are the names of the study and the different configuration options studied.

Wavelets: The set of wavelets used in the algorithm effects everything from landmark localization to similarity measurement. This study will test the effect of three different wavelets sets. The the algorithm depends on a set of Gabor wavelets for jet extraction. The base configuration has adopted the set from [15]. There are, however, many different sets that may be used in the algorithm. Section 6.2 examines two alternative wavelets.

Model Set Selection: The set of model jets effects the landmark localization process. The base configuration uses a set of 70 images[1] with manually selected landmark locations to create the bunch graph. There are two parts to this configuration study. The first tests the size of the model set by using 1, 2, 4, 8, 16, 32, 64, and 128 model images. The second part of the study tests whether or not it matters if the

[1]The exact list of model images used for EBGM training is found in the file "imagelists/ebgm70.srt" from release 5.0 of the CSU Face Identification Evaluation System.

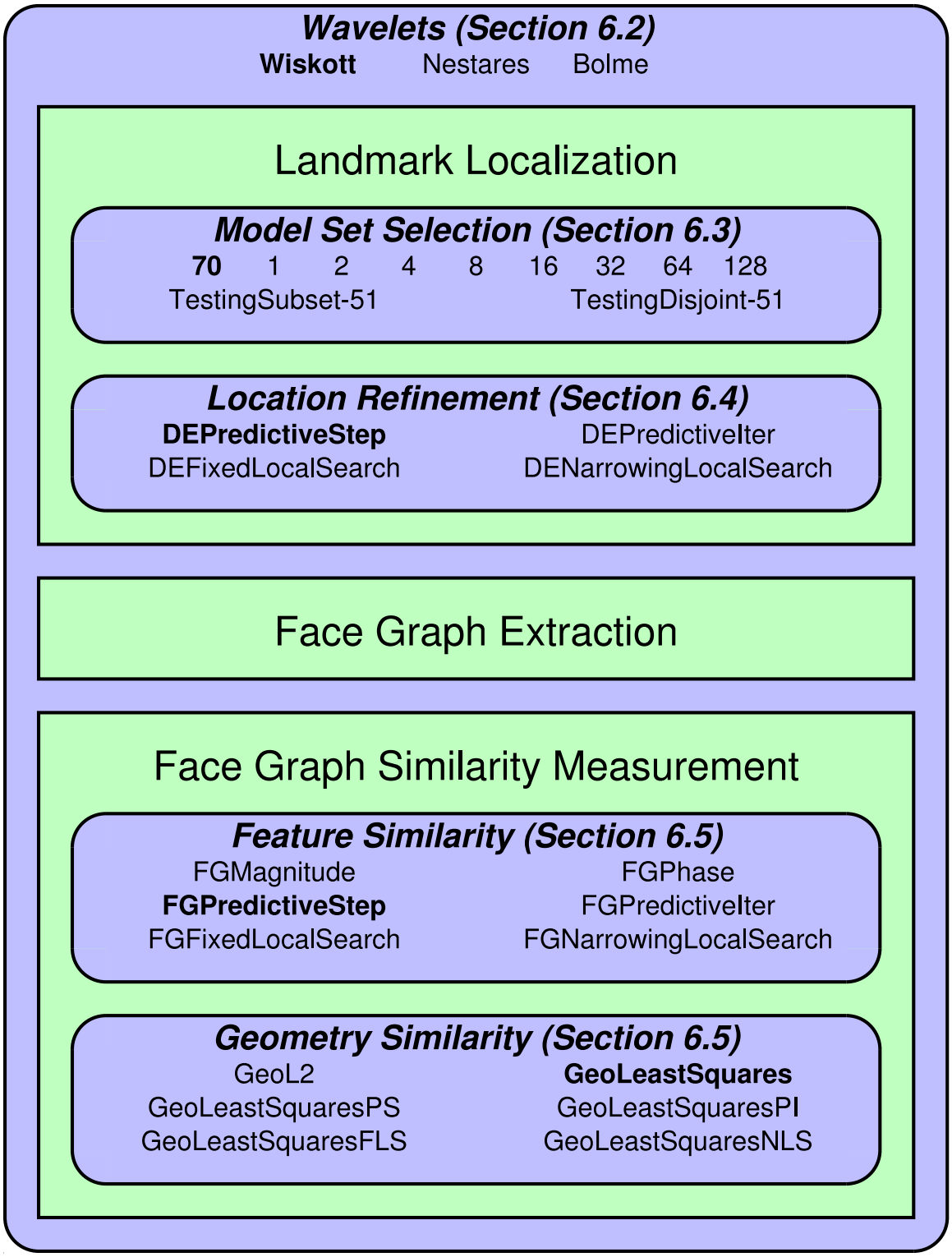


Figure 6.1: Configuration Experiment Overview. The default configuration option is in bold face.

model images are also in the testing set. This effect is tested by running the algorithm twice, first using model images that are a subset of the testing set, and then using model images that are disjoint from the testing set. Details on this study are found in Section 6.3.

Location Refinement: There are four suitable methods for refining landmark locations, and each of these is tested. After the bunch graph has been loaded, the algorithm adjusts the landmark locations. The landmarks are localized using techniques described in Chapter 3. The algorithm first estimates the location of the landmarks and then refines that estimate using the displacement estimation algorithms defined in Section 3.5. The base configuration of the algorithm uses the DEPredictiveStep method for refining the initial location estimates. Section 6.4 tests the other displacement estimation techniques to determine their effect on the algorithms identification performance.

Feature and Geometry Similarity: The final step in the algorithm is to measure the distance between face graphs. The base configuration of the CSU algorithm uses the FGPredictiveStep. The Bochum/USC version uses FGMagnitude. In addition, there are other feature based similarity measures that can be used for similarity computation. The effect of six feature based distance measures is discussed in Section 6.5. Geometry similarity is not used in the standard configuration or the Bochum/USC algorithm; however, it is the subject of a lot of modern research. For this reason, Section 6.5 also tests six variations of geometry similarity.

Optimal Configuration: After studying all of these different aspects of the algorithm, the best of each study will be used to produce an optimal configuration for the EBGM algorithm. The resulting configuration will be compared to the base line algorithm.

The different algorithm configurations are compared using the permutation test using 160 subjects with 4 replicates each[2], described in Section 5.6. The permutation analysis produces an average rank one recognition rate as well as statistical data on the distribution of the rank one recognition rate. The permutation tests will also be used to perform a pairwise comparison of the different configuration options, and will determine the significance of any difference in recognition rate. The test is described in more detail in Section 5.6.

Most experiments focus on two face graph similarity measures to evaluate algorithm performance. The first measure is FGPredictiveStep because it is a good jet based measure. Tests on this measure show how configuration modifications effect the standard algorithm.

[2]These images are listed in the file “list640.srt” included with the CSU Face Recognition Evaluation System.

The second measure computes the similarity of the face graphs using only the landmark locations. The similarity of jets is never considered. The measure, therefore, effectively tests the output of the landmark localization process and will determine how much information is included in geometry alone. Using this measure it is possible to make inferences about the quality of landmark localization.

6.2 Gabor Wavelets Configuration

This experiment tests the effect of substituting different Gabor wavelets into the algorithm. The base algorithm uses a set of 40 Gabor filters for each jet. This experiment tests two other sets of wavelets used for the Gabor jets. The base algorithm uses a set of 40 complex (80 masks) wavelets based upon [15]. These wavelets will be referred to as "Wiskott" wavelets after the author of that paper. There are two other sets tested: Nestares[8] and Bolme.

The Nestares wavelet set uses 4 orientations and 4 frequencies and was designed to be very fast with good image reconstruction properties. The resulting jets have 16 complex wavelet pairs. One of the main differences between the Wiskott and Nestares wavelets is that the Nestares wavelets have much smaller Gaussians relative to their wavelengths. The Nestares wavelets also use frequencies based at full octaves instead of frequencies at half octaves. The Nestares wavelets are much faster because the masks are smaller, and there is less than half as many convolutions to perform per jet.

The Bolme wavelets are a modification to the Wiskott set intended to improve localization performance. There are two parameters that are different from the Wiskott set. The first is that the size of the Gaussian has been reduced. The Wiskott wavelets have very large Gaussian that show 4 to 5 cycles of the sinusoid. Reducing the size of the Gaussian helps the local features to stand out during image localization. The second modification is in the phase of the wavelets. The Wiskott wavelets use phases of 0 and $\pi/2$ for each complex wavelet pair which creates an even and odd wavelet for each complex pair. It is possible that this pairing might favor displacements in one direction. One example is that in the localization experiment the Wiskott wavelets tend to estimate points that are above and to the left of the manually selected values. The Bolme complex wavelet pairs use phase angles of $-\pi/4$ and $\pi/4$ instead of 0 and $\pi/2$. This creates two wavelet masks that are identical except they are reflected across the center of the wavelet. This will insure that the wavelets are equally likely to make phase calculation errors in both directions.

Figure 6.2 shows some of the automatically placed graphs nodes for the wavelet configuration experiment. Visual inspection of many images of this type indicate that Nestares wavelets are not as accurate as the other two sets at locating the landmark jets. There are subtle differences between the Bolme wavelets and the

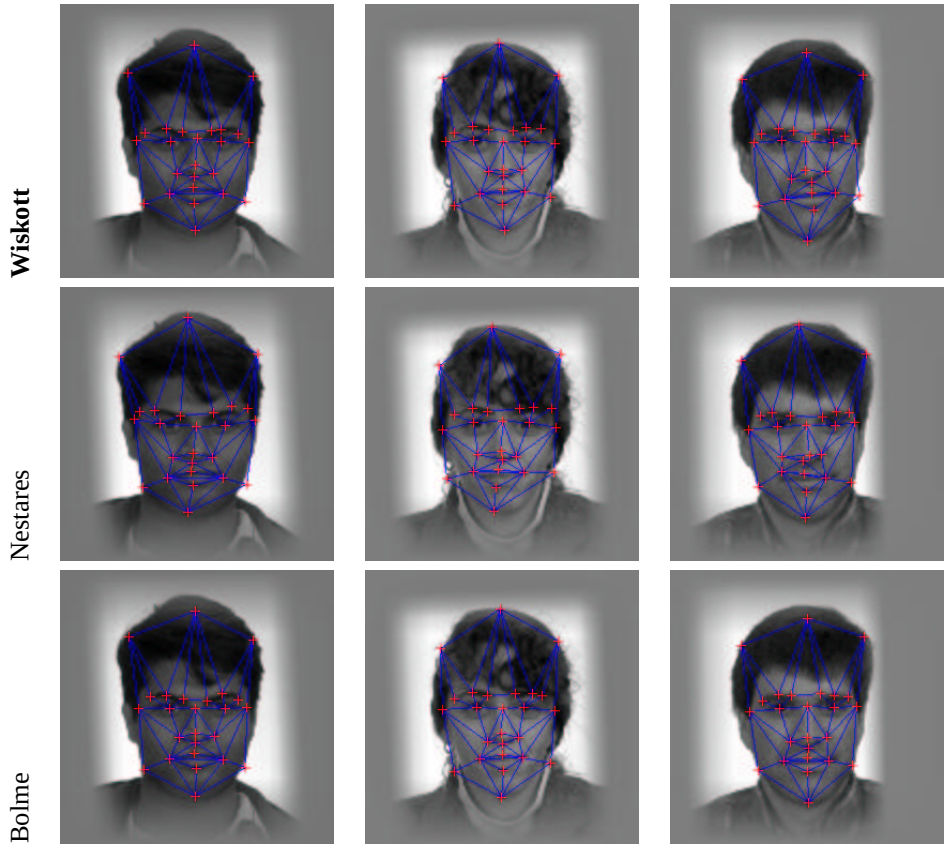


Figure 6.2: Wavelet configuration automatically placed graphs nodes.

Wiskott wavelets. Careful examination revealed that the Bolme wavelets seem to track the landmark locations better.

Table 6.1 shows the results of the permutation test based on landmark jet similarity. The Nestares wavelets perform rather poorly compared to the other two. The Bolme are marginally better than the Wiskott wavelets by a few percentage points.

Table 6.2 shows very different results. The Bolme wavelets still perform the best; however, the Nestares wavelets are not far behind. This would seem to indicate that even though the Nestares wavelets do not track the correct landmark. These results highlight an important aspect of geometry based similarity measures. They do not indicate directly how accurate the landmark localization process is, but how consistent the localization process is within same subject pairs. For example, if the localization process estimates the nose in an image is three pixels below the "true" location, the geometry measure will do fine as long as the nose location is three pixels low for all images of that same subject.

Table 6.3 shows that both the Bolme and Wiskott wavelets are significantly better than the Nestares wavelet.

Table 6.1: The table shows the Mean, Mode, and 95% range for the number of images correctly recognized at rank 1. The corresponding recognition rates are shown where rate is just the count divided by 160: the number of images in this experiment. The base configuration is shown in italics.

WAVELET CONFIGURATION: FGPREDICTIVESTEP

Tested Wavelet	Mean	Mode	Range	Mean %	Mode %	Range %
Wiskott	*99.7*	*100*	*91:108*	*62.3%*	*62.5%*	*56.9%:67.5%*
Nestares	91.2	91	83:99	57.0%	56.9%	51.9%:61.9%
Bolme	104.0	104	96-112	65.0%	65.0%	60.0%:70.0%

Table 6.2: The table shows the Mean, Mode, and 95% range for the number of images correctly recognized at rank 1. The corresponding recognition rates are shown where rate is just the count divided by 160: the number of images in this experiment. The base configuration is shown in italics.

WAVELET CONFIGURATION: GEOLEASTSQUARES

Tested Wavelet	Mean	Mode	Range	Mean %	Mode %	Range %
Wiskott	*36.8*	*37*	*29:45*	*23.0%*	*23.1%*	*18.1%:28.1%*
Nestares	40.6	41	33:49	25.4%	25.6%	20.6%:30.6%
Bolme	41.8	41	33-55	26.1%	25.6%	20.6%:31.9%

The Bolme wavelets also outperform the Wiskott, but those results are not statistically significant. Table 6.4 also shows the Bolme wavelet as best; however, no wavelet has a clear advantage in the geometry only test.

The results of this experiment indicate that the Nestares wavelets are not suitable for this algorithm. To be fair, Nestares wavelets were never intended for use in this type of algorithm. Both the Bolme and Wiskott wavelets performed well; however, the Bolme wavelets showed a marginal improvement in all three analysis performed in this experiment.

Table 6.3: These results show the difference in recognition rates for each pair of wavelet sets using the feature based distance measure FGPredictiveStep. Statistically significant results are shown in bold.

Algorithm 1	Algorithm 2	Mean	Range	P(Alg1 ¿ Alg2)
Bolme	**Nestares**	**12.8**	**3:23**	**0.9970**
Bolme	*Wiskott*	4.2	-3:12	0.8748
Wiskott	**Nestares**	**8.6**	**-1:18**	**0.9678**

Table 6.4: These results show the difference in recognition rates for each pair of wavelet sets using the geometry based distance measure GeoLeastSquares. Statistically significant results are shown in bold.

Algorithm 1	**Algorithm 2**	**Mean**	**Range**	**P(Alg1 ¿ Alg2)**
Bolme	Nestares	1.1	-10:12	0.5456
Bolme	*Wiskott*	5.0	-6:16	0.8163
Nestares	*Wiskott*	3.9	-7:15	0.7437

6.3 Model Set Selection

These experiments analyze the selection of model images that are used to create the bunch graph. There were two factors tested in this model set configuration study. The first was the size of the model set. This first experiment used the basic algorithm using model sets of different sizes. The second tested the effect of using model images that are also in the testing set.

The CSU Face Recognition Evaluation System has a set of 150 images[3] with manually selected landmark locations. Seventy of the 150 images were selected for use as models in the base configuration of the EBGM algorithm. The number, 70, was chosen to match the original configuration in [15].

The experiment tested model sets of 1, 2, 4, 8, 16, 32, 64, and 128 model images. In each run of the algorithm, the model images were randomly chosen from the set of 150 images with manually selected landmark locations. For each run there is a possibility that a particularly bad or particularly good set of model images is chosen. For this reason each size was run 10 times to determine an average recognition rate.

Figure 6.3 shows the results of this experiment. The graph shows the log of the model set size verses the recognition count. The "+" shows the mean recognition rate for each test. The line runs through the average of the 10 points for each size and indicates that a larger size will produce better results; however, these results may be somewhat misleading. The maximum recognition count in each column is almost exactly the same. There is even an experiment with one model image that has a recognition rate equivalent to the average of the larger sets. This would seem to indicate that choosing the right image for the model set is very important. The results here seem to indicate that a set size 70 is a reasonable choice.

Also of note is that as the set size gets larger, the variation decreases. This is probably do to the fact that the amount of overlap between model sets increase as the set size gets larger. There are only 150 images available to serve as models; thus, every pair of experiments for 128 model images is guaranteed to have at

[3] ebgm150.srt

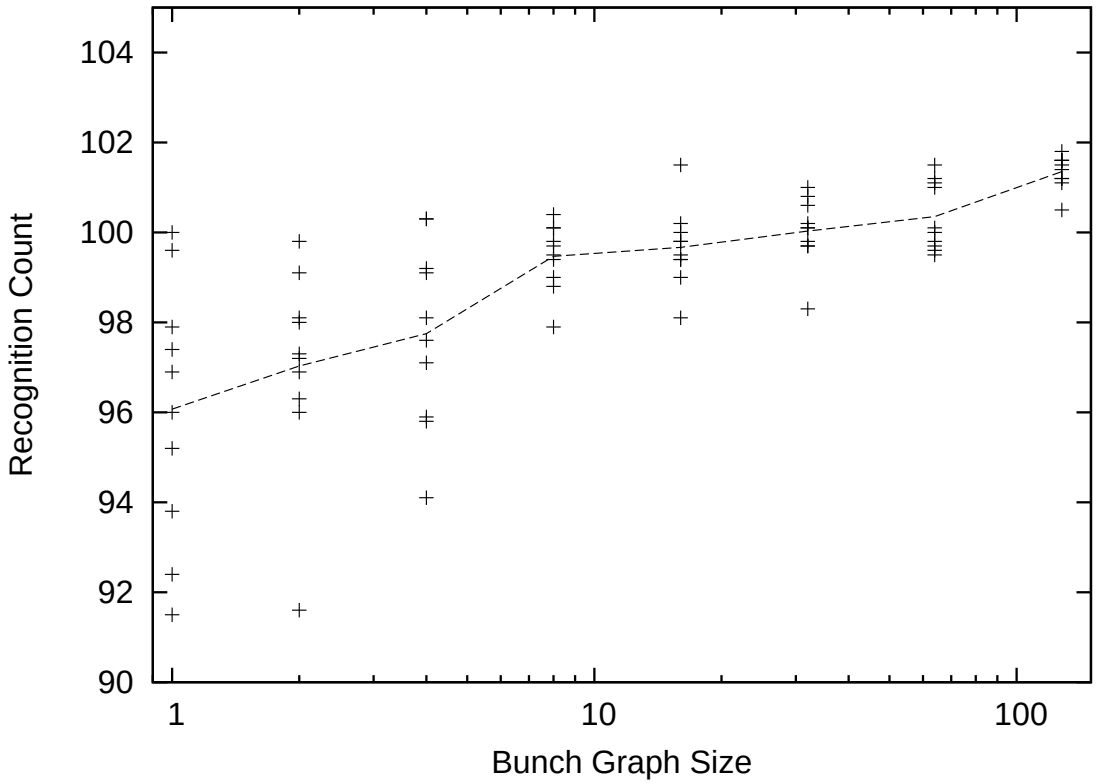


Figure 6.3: Model Set Size Results: Recognition rate out of 160.

least 82% overlap in the model sets. It is impossible to tell from this experiment if the smaller distributions are an effect of set size or set overlap.

The second part of this configuration experiment tests whether using model images that are also in the testing set is an important factor in algorithm performance. Of the set of 150 images used for models, 51 (representing 49 subjects) are also in the set of 640 (representing 160 subjects) used for testing. The standard configuration for the model set uses 25 images from the testing set and 45 other images from the FERET database. This experiment will run the algorithm twice. The first run will use the 51 images also in the testing set as models. The second run will use 51 randomly chosen images that are not in the testing set as models. These two configurations will then be compared to each other and to the base configuration.

Tables 6.6, 6.5, 6.7, and 6.8 show the recognition rates for the permutation test. The results show that there is hardly any difference in recognition performances for the jet based similarity measure. The geometry based results show that there is a significant advantage to using images of the same subjects as models.

Another interesting result of this experiment is that the consistency of the localization process does not seem

Table 6.5: The table shows the Mean, Mode, and 95% range for the number of images correctly recognized at rank 1. The corresponding recognition rates are shown where rate is just the count divided by 160: the number of images in this experiment. The base configuration is shown in italics.

MODEL SET SELECTION: FGPREDICTIVESTEP

Tested Localization Method	Mean	Mode	Range	Mean	Mode	Upper 95%
Base (70)	*99.8*	*100*	*91-108*	*62.4%*	*62.5%*	*56.9%-67.5%*
Disjoint (51)	101.0	101	93-109	63.1%	63.1%	58.1%-68.1%
Subset (51)	99.6	99	92-108	62.2%	61.9%	57.5%-67.5%

Table 6.6: The table shows the Mean, Mode, and 95% range for the number of images correctly recognized at rank 1. The corresponding recognition rates are shown where rate is just the count divided by 160: the number of images in this experiment. The base configuration is shown in italics.

MODEL SET SELECTION: GEOLEASTSQUARES

Tested Localization Method	Mean	Mode	Range	Mean	Mode	Upper 95%
Base(70)	*36.8*	*37*	*29-45*	*23.0%*	*23.1%*	*18.1%-28.1%*
Disjoint (51)	35.9	36	27-45	22.4%	22.5%	16.9%-28.1%
Subset (51)	47.0	47	39-56	29.4%	29.4%	24.4%-35.0%

to effect the recognition performance for the jet based similarity measure. The geometry based results show a significant difference in favor of using a model set that is a subset of the testing set, instead of using disjoint set. The jet based measure actually favors the disjoint set over the subset. It is possible that the displacement correction in the FGPredictiveStep measure compensates for any inconsistencies in the localization process. The result is that there seems to be very little effect of the localization process on algorithm performance when using the standard jet based face graph similarity measure.

Table 6.7: These results show the difference in recognition rates for the standard, subset, and disjoint bunch graph sets using feature based distance measure FGPredictiveStep. None of the results are statistically significant.

Algorithm 1	Algorithm 2	Mean	Range	P(Alg1 ¿ Alg2)
Disjoint (51)	*Base (70)*	1.21	-4:7	0.5966
Base (70)	Subset (51)	0.17	-6:6	0.4577
Disjoint (51)	Subset (51)	1.37	-4:7	0.6148

Table 6.8: These results show the difference in recognition rates for the standard, subset, and disjoint bunch graph sets using feature based distance measure GeoLeastSquares. Statistically significant results are shown in bold.

Algorithm 1	Algorithm 2	Mean	Range	P(Alg1 ¿ Alg2)
Base (70)	Disjoint (51)	0.92	-9:11	0.5245
Subset (51)	***Base (70)***	**10.18**	**1:20**	**0.9751**
Subset (51)	**Disjoint (51)**	**11.09**	**0:22**	**0.9745**

6.4 Localization Methods

There are four displacement estimator (DE) algorithms tested in this study: DEPredictiveStep, DEPredictiveIter, DEFixedLocalSearch, and DENarrowingLocalSearch. Complete descriptions of these algorithms can be found in Section 3.5.

The basic configuration of the algorithm uses DEPredictiveStep during the localization phase to refine the landmark locations. In this experiment the DEPredictiveStep will be replaced by the other DE algorithms and the results will be compared using a permutation test.

For this experiment the algorithm is run four different times, corresponding to the DE algorithms: DEPredictiveStep, DEPredictiveIter, DEFixedLocalSearch, DENarrowingLocalSearch. Each run produces two different distance matrices based on the feature based measure and the geometry based measure. The distance matrices are analyzed using a permutation test.

The geometry based measure will measure the similarity of the landmark locations. It does not indicate that the landmark locations are placed correctly. Instead the measure indicates if the landmark locations are placed consistently. For example, if the DE places the point for the nose tip wrong on one image of a subject, the measure should still perform well as long as it makes the same mistake on all of the other images of that subject.

Figure 6.4 shows the some samples of automatically placed landmark locations. There is nothing in the figure that makes one DE algorithm stand out. Errors made by one method seem to be replicated in the others. The landmark jet based results in Table 6.9 show that the DE method has very little effect on overall algorithm performance. There is only a one percent difference from the best to worst performance. Also, the results of the algorithm comparison in Table 6.10 do not really show any bias toward any of the DE methods.

Table 6.11 shows a small difference between the predictive and local search methods. The local search methods seem to be more consistent when locating the landmarks. These results lead to the same conclusions as the previous section: the consistency of the localization process does not seem to effect the FGPredictiveStep

Table 6.9: The table shows the Mean, Mode, and 95% range for the number of images correctly recognized at rank 1. The corresponding recognition rates are shown where rate is just the count divided by 160: the number of images in this experiment. The base configuration is shown in italics.

LOCATION REFINEMENT: FGPREDICTIVESTEP

Tested Localization Method	Mean	Mode	Range	Mean	Mode	Upper 95%
DEPredictiveStep	*99.7*	*100*	*91-108*	*62.3%*	*62.5%*	*56.9%-67.5%*
DEPredictiveIter	99.1	100	91-107	61.9%	62.5%	56.9%-66.9%
DEFixedLocalSearch	98.6	99	90-107	61.8%	61.9%	56.3%-66.9%
DENarrowingLocalSearch	98.0	99	90-106	61.3%	61.9%	56.3%-66.3%

Table 6.10: These results show the difference in recognition rates for each pair of landmark point refinement methods using the feature based distance measure FGPredictiveStep. None of the results are statistically significant.

Algorithm 1	Algorithm 2	Mean	Range	P(Alg1 ¿ Alg2)
DEPredictiveStep	DEPredictiveIter	0.69	-6:7	0.52810
DEPredictiveStep	DEFixedLocalSearch	1.15	-6:7	0.58880
DEPredictiveStep	DENarrowingLocalSearch	1.76	-5:9	0.66310
DEPredictiveIter	DEFixedLocalSearch	0.45	-6:7	0.49310
DEPredictiveIter	DENarrowingLocalSearch	1.06	-5:7	0.58240
DEFixedLocalSearch	DENarrowingLocalSearch	0.61	-4:6	0.51930

face graph similarity measure.

6.5 Face Graph Similarity

The final experiment will test all of the different face graph similarity measures, while holding the early part of the algorithm constant. The experiment will show how the different similarity measures discussed in Chapter 4 compare to each other.

The first part of the similarity study examines the feature based similarity measures: FGMagnitude, FGPhase, FGPredictiveStep, FGPredictiveIter, FGFixedLocalSearch, FGNarrowingLocalSearch. FGGridSample is excluded because the computation does not complete in a reasonable amount of time.

The Bochum/USC algorithm used the magnitude similarity measure for similarity computation because it performed better on FERET fb probe set. The CSU implementation has shown similar results; however, FGPredictiveStep was selected for the control in these experiments because it performs much better on other FERET probe sets. The performance on fb was only slightly worse.

Table 6.13 shows the rank one results for all of the similarity measures. The displacement corrected similarity

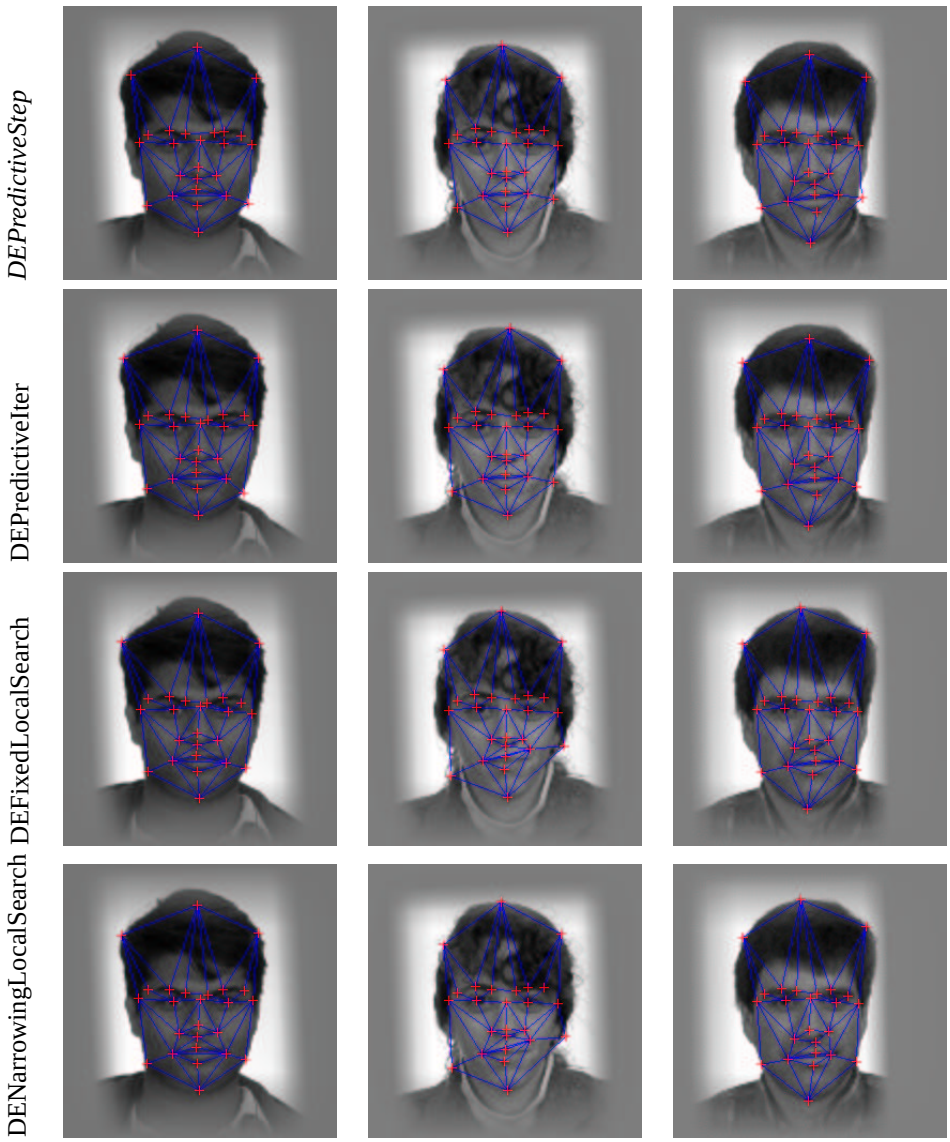


Figure 6.4: Samples of automatically placed graphs nodes.

Table 6.11: The table shows the Mean, Mode, and 95% range for the number of images correctly recognized at rank 1. The corresponding recognition rates are shown where rate is just the count divided by 160: the number of images in this experiment. The base configuration is shown in italics.

LOCATION REFINEMENT: GEOLEASTSQUARES

Tested Localization Method	Mean	Mode	Range	Mean %	Mode %	Range %
DEPredictiveStep	*36.8*	*37*	*29-45*	*23.0%*	*23.1%*	*18.1%-28.1%*
DEPredictiveIter	37.6	38	30-46	23.5%	23.8%	18.8%-28.8%
DEFixedLocalSearch	41.1	40	33-50	25.7%	25.0%	20.6%-31.3%
DENarrowingLocalSearch	40.5	41	32-49	25.3%	25.6%	20.0%-30.6%

Table 6.12: These results show the difference in recognition rates for each pair of landmark point refinement methods using the geometry based distance measure GeoLeastSquares. None of the results are statistically significant.

Algorithm 1	Algorithm 2	Mean	Range	P(Alg1 ¿ Alg2)
DEPredictiveIter	*DEPredictiveStep*	0.77	-9:11	0.52390
DEFixedLocalSearch	*DEPredictiveStep*	4.30	-6:15	0.78480
DENarrowingLocalSearch	*DEPredictiveStep*	3.73	-7:14	0.75290
DEFixedLocalSearch	DEPredictiveIter	3.53	-6:13	0.75870
DENarrowingLocalSearch	DEPredictiveIter	2.96	-7:12	0.71350
DEFixedLocalSearch	DENarrowingLocalSearch	0.56	-8:9	0.49990

Table 6.13: The table shows the Mean, Mode, and 95% range for the number of images correctly recognized at rank 1. The corresponding recognition rates are shown where rate is just the count divided by 160: the number of images in this experiment. The base configuration is shown in italics.

FACE GRAPH SIMILARITY: GABOR JET

Tested Similarity Measure	Mean	Mode	Range	Mean %	Mode %	Range %
FGMagnitude	93.0	94	85-101	58.1%	58.8%	53.1%-63.1%
FGPhase	89.4	89	81-98	55.9%	55.6%	50.6%-61.3%
FGPredictiveStep	*99.7*	*100*	*91-108*	*62.3%*	*62.5%*	*56.9%-67.5%*
FGPredictiveIter	101.2	100	93-109	63.3%	62.5%	58.1%-68.1%
FGFixedLocalSearch	103.9	104	96-112	64.9%	65.0%	60.0%-70.0%
FGNarrowingLocalSearch	103.9	104	96-112	64.9%	65.0%	60.0%-70.0%

measures did much better than FGMagnitude and FGPhase.

The results here would seem to confirm that the displacement compensated measures perform the best. The Local search method averages 6.8% better than the Magnitude measure. Within the displacement compensated measures, Local Search is the best. The Local Search methods are much slower than the FGPredictiveIter method for determining displacements. If performance is an issue, FGPredictiveIter is a good alternative.

The second part of the similarity study examines the Geometry based similarity measures. These measures include the standard Euclidean distance (L2), the least squares method, and the corrected least squares measures (GeoLeastSquaresPS is least squares corrected with DEPredictiveStep). It is obvious from the results in Tables 6.15 and 6.16 that the geometry based similarity measures are much worse than the feature based measures. Like the feature based measures the displacement corrected geometry is much better than either of the geometry only measures.

It is important to understand that the corrected geometry measure is also effected by the features. The features

Table 6.14: These results show the difference in recognition rates for each pair of feature based distance measures. Statistically significant results are shown in bold.

Algorithm 1	Algorithm2	Mean	Range	P(Alg1 ¿ Alg2)
FGMagnitude	FGPhase	3.58	-5:12	0.7706
FGPredictiveStep	**FGMagnitude**	**6.77**	**-1:14**	**0.9524**
FGPredictiveIter	**FGMagnitude**	**8.25**	**1:16**	**0.9820**
FGFixedLocalSearch	**FGMagnitude**	**10.91**	**4:18**	**0.9991**
FGNarrowingLocalSearch	**FGMagnitude**	**10.91**	**4:18**	**0.9986**
FGPredictiveStep	**FGPhase**	**10.35**	**4:17**	**0.9980**
FGPredictiveIter	**FGPhase**	**11.83**	**5:19**	**0.9991**
FGFixedLocalSearch	**FGPhase**	**14.49**	**7:22**	**0.9997**
FGNarrowingLocalSearch	**FGPhase**	**14.49**	**7:22**	**0.9998**
FGPredictiveIter	*FGPredictiveStep*	1.48	-3:6	0.6799
FGFixedLocalSearch	*FGPredictiveStep*	4.14	-1:9	0.9276
FGNarrowingLocalSearch	*FGPredictiveStep*	4.14	-1:9	0.9233
FGFixedLocalSearch	FGPredictiveIter	2.66	-1:7	0.8470
FGNarrowingLocalSearch	FGPredictiveIter	2.67	-2:7	0.8459
FGNarrowingLocalSearch	FGFixedLocalSearch	0.01	-2:2	0.2083

are used to change the geometry of the graph. Therefore, the features will have a large effect on the similarity.

6.6 Standard and Optimal Configuration

This last set of tests will use the results from the previous sections to create optimized configurations for the algorithm. There are two new configurations for the algorithm designed to be more accurate in recognition: EBGM Standard and EBGM Optimal.

EBGM Standard: This version of the algorithm is the standard configuration for the CSU EBGM algorithm

Table 6.15: The table shows the Mean, Mode, and 95% range for the number of images correctly recognized at rank 1. The corresponding recognition rates are shown where rate is just the count divided by 160: the number of images in this experiment.

FACE GRAPH SIMILARITY: GEOMETRY

Tested Similarity Measure	Mean	Mode	Range	Mean %	Mode %	Range %
GeoL2	43.7	45	35:53	27.3%	28.1%	28.121.9-33.1
GeoLeastSquares	36.8	37	29-45	23.0%	23.1%	18.1-28.1%
GeoLeastSquaresPS	65.1	64	56-75	40.7%	40.0%	36.0-46.9%
GeoLeastSquaresPI	68.3	69	59-78	42.7%	43.1%	36.9-48.8%
GeoLeastSquaresFLS	71.7	71	62-81	44.8%	44.4%	38.8-50.6%
GeoLeastSquaresNLS	73.6	75	64-83	46.0%	46.9%	40.0-51.9%

Table 6.16: These results show the difference in recognition rates for each pair of geometry based distance measures. Statistically significant results are shown in bold.

Algorithm 1	Algorithm 2	Mean	Range	P(Alg1 ¿ Alg2)
GeoL2	GeoLeastSquares	6.92	-2:16	0.9323
GeoLeastSquaresFLS	**GeoL2**	**27.94**	**17:39**	**1.0000**
GeoLeastSquaresNLS	**GeoL2**	**29.92**	**19:41**	**1.0000**
GeoLeastSquaresPI	**GeoL2**	**24.63**	**14:35**	**1.0000**
GeoLeastSquaresPS	**GeoL2**	**21.40**	**11:32**	**1.0000**
GeoLeastSquaresFLS	**GeoLeastSquares**	**34.86**	**25:45**	**1.0000**
GeoLeastSquaresNLS	**GeoLeastSquares**	**36.84**	**27:47**	**1.0000**
GeoLeastSquaresPI	**GeoLeastSquares**	**31.55**	**21:42**	**1.0000**
GeoLeastSquaresPS	**GeoLeastSquares**	**28.32**	**19-38**	**1.0000**
GeoLeastSquaresNLS	GeoLeastSquaresFLS	1.982500	-4:8	0.6955
GeoLeastSquaresFLS	GeoLeastSquaresPI	3.308600	-5:11	0.7488
GeoLeastSquaresFLS	GeoLeastSquaresPS	6.541400	-2:15	0.9192
GeoLeastSquaresNLS	GeoLeastSquaresPI	5.291100	-3:13	0.8790
GeoLeastSquaresNLS	**GeoLeastSquaresPS**	**8.523900**	**0:17**	**0.9658**
GeoLeastSquaresPI	GeoLeastSquaresPS	3.232800	-3:10	0.7874

Table 6.17: Three algorithm configurations.

Configuration	Wavelet	Model Set	Localization	Similarity
Base	*Wiskott*	*ebgm70.srt*	*DEPredictiveStep*	*FGPredictiveStep*
EBGM Standard	Bolme	ebgm70.srt	DEPredictiveIter	FGPredictiveIter
EBGM Optimal	Bolme	ebgm70.srt	DENarrowingLocalSearch	FGNarrowingLocalSearch

Table 6.18: The table shows the Mean, Mode, and 95% range for the number of images correctly recognized at rank 1. The corresponding recognition rates are shown where rate is just the count divided by 160: the number of images in this experiment. The base configuration is shown in italics.

OPTIMAL CONFIGURATION RESULTS

Algorithm	Mean	Mode	Range	Mean %	Mode %	Range %
EBGM Optimal	106.3	105	98-114	66.4%	65.6%	61.2%-71.2%
EBGM Standard	105.0	104	97-113	65.6%	65.0%	60.6%-70.6%
Base	*99.8*	*100*	*91-108*	*62.4%*	*62.5%*	*56.9%-67.5%*

distributed with the CSU Face Recognition Evaluation System version 5.0. It performs better than the base algorithm used in the earlier sections and is still relatively fast. The configuration uses the Bolme wavelet set because it showed the most improvement in recognition rate in the earlier configuration experiments. The DEPredictiveIter is used for both the localization and similarity measurement because it seems to be slightly better than DEPredictiveStep method and is still relatively fast.

EBGM Optimal: This configuration attempts to produce the best recognition results possible based on the configuration experiments presented in this chapter. Like the standard configuration, it also uses the Bolme wavelet set. The DENarrowingLocalSearch showed a slight advantage of the four displacement estimators tested.

The new configurations still uses the same model as the base configuration. It was not clear that changing the model set would have any clear advantage in recognition performance. The experiments conducted indicate that 70 model images is a reasonable number to serve as templates, which offers a good balance of accuracy and speed. If a specific model set was chosen that performed better on the permutation experiment, it is more likely that it would be optimizing the algorithm for the 640 testing images.

Table 6.18 compares the recognition rates of the base configuration to the two new configurations in the permutation test. The results show that the two new configurations do perform better than the base configuration by a three to four percent increase in recognition rate. Table 6.19 shows the results of a direct algorithm comparison. The results indicate that both of the new configurations are better than the base. There is only a small improvement in the optimal over the standard.

Figures 6.5 and 6.6 show the results of the three configurations on the FERET probe sets. These results show that the optimal configurations perform better on all but the fc probe set. Interestingly, the base configuration performs much better on fc. The fc probe set tests the effect of variations on lighting. It is possible that variations in lighting have less effect on Wiskott based wavelets.

Table 6.19: These results show the difference in recognition rates for each pair of configurations. Statistically significant results are shown in bold.

Algorithm 1	**Algorithm 2**	**Mean**	**Range**	**P(Alg1 ¿ Alg2)**
EBGM Optimal	EBGM Standard	1.34	-3:6	0.6379
EBGM Optimal	***Base***	**6.54**	**0:13**	**0.9643**
EBGM Standard	*Base*	5.20	-2:12	0.9121

This chapter has revealed a few ways to improve the performance of the EBGM algorithm. Even though the performance has improved on three of the four FERET datasets, the CSU implementation still falls short of the Bochum/USC algorithm (see Figures 1.2 and 1.3). It is very possible that algorithm improvement may come from other sources not tested in this thesis.

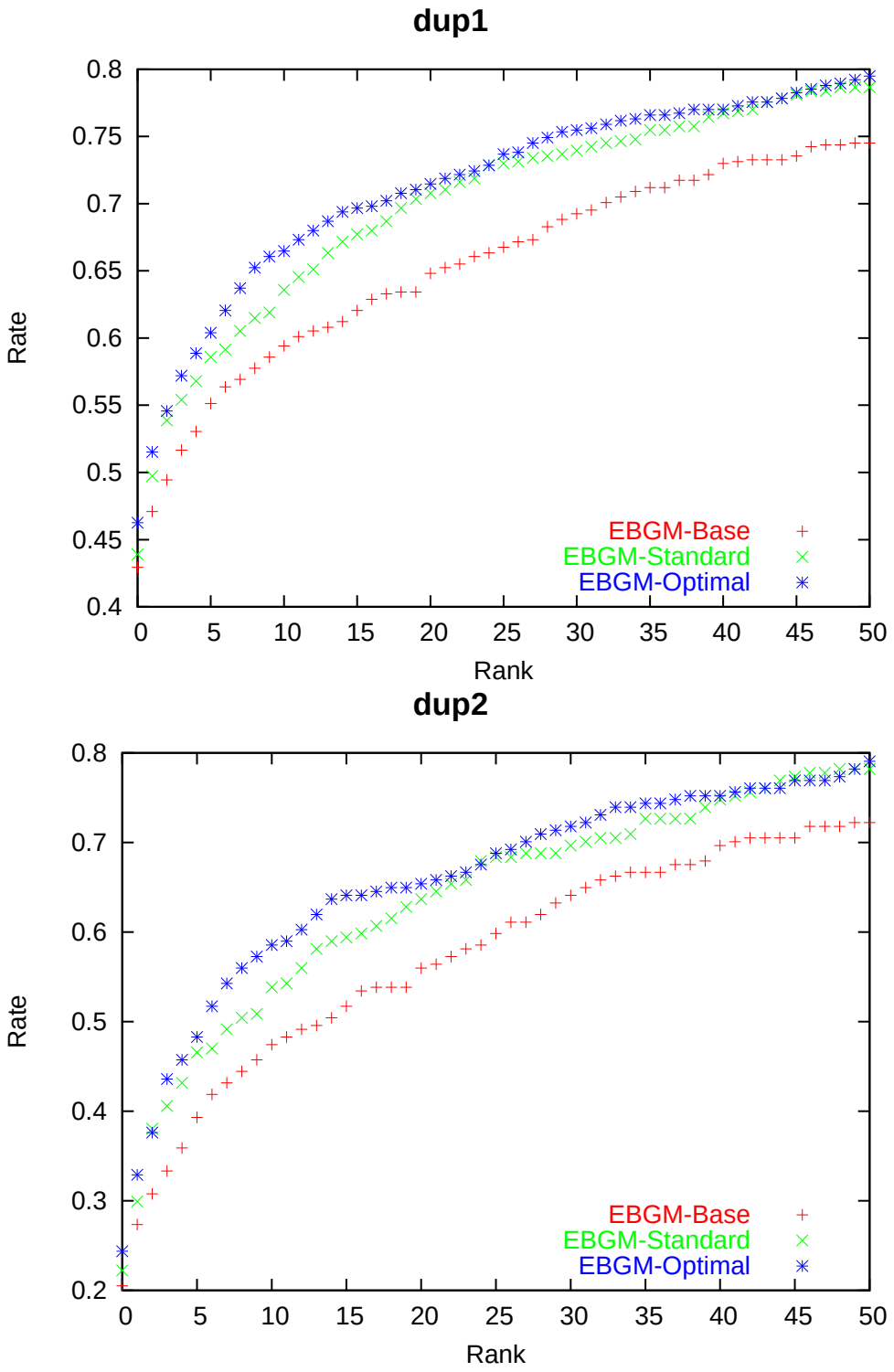


Figure 6.5: Comparison of the CSU EBGM algorithm to the base configuration in the FERET test dup1 and dup2.

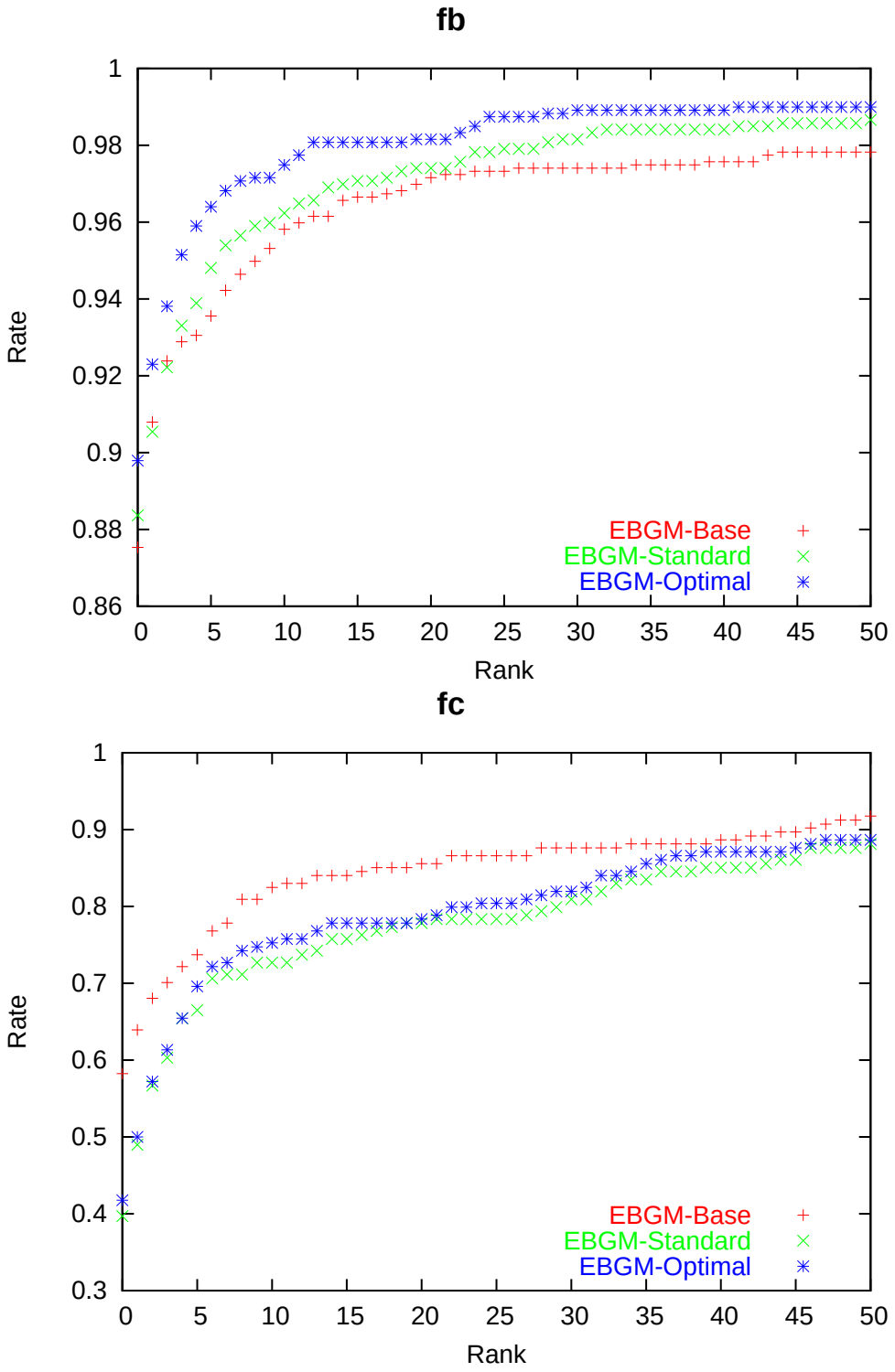


Figure 6.6: Comparison of the CSU EBGM algorithm to the base configuration in the FERET test fb and fc.

Chapter 7

Conclusions

7.1 Summary

This thesis describes my implementation of the Bochum/USC Elastic Bunch Graph Matching algorithm used in the FERET evaluations [12]. The CSU EBGM algorithm performs well on the FERET probe sets fb, fc, dup1, and dup2. In these tests the algorithm performance is comparable to other algorithms evaluated in the original FERET test. The CSU EBGM algorithm performed much better that the FERET eigenfaces (PCA) and has results that are comparable to the MIT Bayesian Interpersonal/Extra-personal Classifier (BIC) and the UMD Linear Discriminant Analysis (LDA) implementation. For results see Section 1.3.

Although the CSU EBGM performed well compared to those two algorithms, the CSU implementation performs significantly worse than the Bochum/USC implementation. The Bochum/USC algorithm performed better on each of the four FERET probe sets. There are many possible areas that could account for differences in recognition performance.

The CSU EBGM implementation attempts to capture the spirit of the algorithm produced by Bochum/USC. It is not a copy of that algorithm. As a result there are many very subtle differences in the two algorithms and a few substantial differences in the two algorithms. To test each difference would be very difficult; however, a few tests have been conducted that have not shown any significant improvements on recognition performance (Section 1.3). A few more suggestions are discussed under future work (Section 7.2).

The CSU EBGM algorithm has also been compared to the other algorithms included in the CSU Face Recognition Evaluation System version 5.0[3]: PCA and BIC. We have not had much success with the CSU LDA algorithm. The performance of that algorithm has never been spectacular, probably due to improper training. For that reason CSU EBGM will not be compared to LDA.

The CSU EBGM algorithm performs better than PCA on the fb, dup1, and dup2 probe sets. PCA performs

much better on the fc probe set. The fc probe set is designed to test the face recognition algorithms when the probe and gallery images are under different illumination. It is possible that the Gabor wavelets make the CSU EBGM algorithm sensitive to an illumination change.

Another interesting finding of the CSU EBGM to PCA comparison is that the PCA performed much better on the permutation test than the CSU EBGM algorithm. This is a surprising result because the permutation test selects probe and gallery images that should be similar to the pairings found in fb, dup1, and dup2. It is unknown why the CSU EBGM algorithm would perform better on fb, dup1, and dub2, and worse on the permutation test. One guess is that the performance difference is an effect of the gallery size. The permutation test selects its probe and gallery out of 160 subjects with 4 images each. Each iteration of this test will have 160 probe images and 160 gallery images. When producing results for the FERET probe sets, the probe sets vary in size, but the gallery stays fixed at 1196 images. It is possible that that PCA performs better with a smaller gallery, and its performance degrades more rapidly as gallery size increases.

The BIC algorithm shows good performance in all of the tests. On the standard FERET probe sets, BIC outperforms the CSU EBGM on dup1 and dup2, the results are comparable on fc, and the CSU EBGM does better on fb. On the permutation test BIC performs much better than CSU EBGM. It is unclear if the permutation results show BIC is clearly better or if there is a similar effect to that seen in the CSU EBGM to PCA comparison.

One of the most interesting results presented in this thesis is that accuracy of the landmark localization process seems to have very little effect on the final algorithm performance. The CSU EBGM algorithm landmark localization process finds the landmarks; however, it is not very precise. For example, the algorithm could find the nose in an image, but the location of that landmark could drift around a few pixels. It is my understanding from personal correspondence with Kazunori Okada that the Bochum/USC algorithm had similar results for landmark localization.

Experiments in Sections 6.3 and 6.4 specifically tested the effects of modifications to the localization process. Both of these tests show that the modifications had very little effect on the overall performance of the algorithm. In the model set selection experiments, the geometry only distance measure showed a significant improvement in accuracy when the model set was a subset of the testing set. This improvement in localization did not improve the jet based performance. If anything, it made the jet based performance worse.

The results in Section 6.2 indicate that the wavelet set used in the algorithm seems to have the most effect on algorithm performance. Each set of wavelets had a large effect on the performance of the algorithm. Interestingly the Bolme wavelets improved the recognition performance on the fb, dup1, and dup2 probe sets while hurting performance on the fc probe set.

It is my belief that the best way to effect the performance of the algorithm is through modifications to the wavelets, the normalization process, and similarity measurement. Such modifications seem to have a larger effect on the similarity of the face graphs than any other changes tested thus far. The model set and localization process appear to be adequate, and it is doubtful that any improvements to this early part of the algorithm will have a significant effect on algorithm performance.

7.2 Future Work

7.2.1 Replicating Bochum/USC Performance

One possibility for future work is to determine which aspects of the original Bochum/USC algorithm led to its good performance on the FERET probe sets. Section 1.3 briefly discussed a few experiments of that nature that have already been conducted. The experiments presented in Chapter 6 have also tested different configurations of the algorithm. None of these experiments have indicated that any part of the CSU algorithm that could benefit from further modification.

These experiments did show that good landmark localization does not correlate to good algorithm performance. Although the jet based displacement estimation technique was able to track landmarks, it did not show outstanding accuracy. Also, all of the experiments that tested landmark localization showed the accuracy had very little effect on the overall system performance.

It is also known that the Bochum/USC algorithm used the Wiskott wavelet set and the FGMagnitude similarity measure. Any experiments that would try to replicate the USC results should be able to do so using that configuration. Although the CSU System has tested both the wavelets and the similarity measure, there are still a few differences in the systems that could be tested.

The first is the way that the wavelet convolution is performed. The Bochum/USC system performs wavelet convolutions over the entire image by computing the discrete Fourier transform of the image and then convolving the Fourier space with a Gaussian. This method introduces a few differences that will have an effect on the convolution values.

The discrete Fourier transform operates under the assumption that the image repeats itself. This effect is illustrated in Figure 7.1. A wavelet convolution will still be primarily effected by the area close to the center; however, the convolution is also influenced by the repeated images.

The FGMagnitude similarity measure is the same as the one used in the USC system. Other factors that effect the similarity may vary, such as the landmarks. For the CSU algorithm I chose 25 landmarks and 55

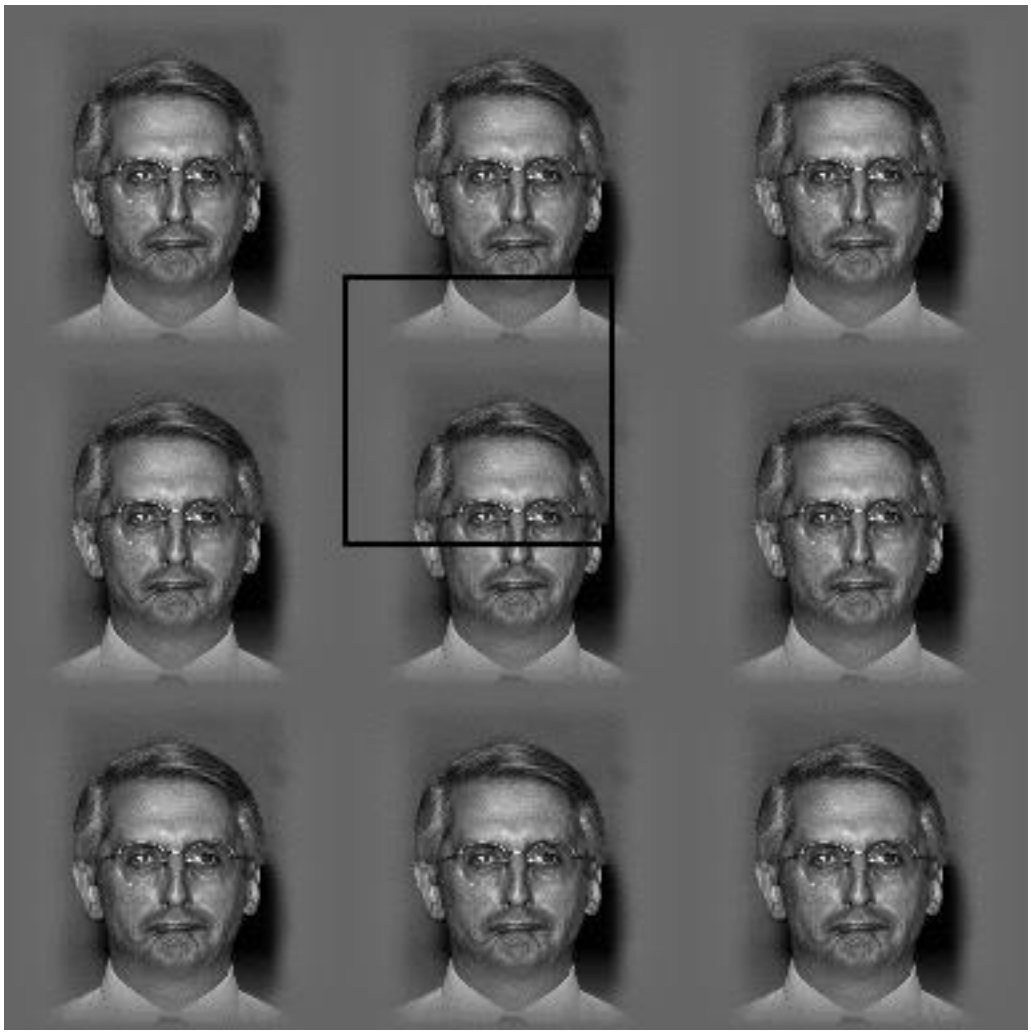


Figure 7.1: Convolution as produced by the Fourier transform method.

interpolated jets distributed throughout the face. Communications with Okada suggested the Bochum/USC system used a total of 80 jets extracted from landmark and interpolated locations. I am also under the impression that the Bochum/USC system would ignore the jets extracted from the outside of the head for the final similarity computation. The current version of the CSU algorithm does not use the same set of landmarks and makes no attempt to discard jets from the edge of the face. Both of these differences may lead to improved performance.

One last area of the algorithm that may offer possibilities for improvement is the normalization process. Even though the edge of the source image is smoothed, it is still clearly visible in the normalized imagery (See Figure 5.3). This artifact could still have a large effect on the Gabor wavelet convolutions. Reducing this effect even further could have a significant impact on performance. In addition to this, other normalization techniques could be tried to eliminate other lighting and image effects that may be confusing the algorithm.

7.2.2 Face Graph Similarity Measures

One area of the algorithm that could benefit from more study is testing new face graph similarity measures. This thesis has focused on similarity measures inspired by the Bochum/USC documentation. There are two modifications to the face graph similarity computations that may improve algorithm performance.

The first is to combine the use of jet based similarity measures and geometry based similarity measures. This could be accomplished by using a new similarity equation of the form:

$$L_{JetGeo}(G_1, G_2) = \alpha L_{Jet}(G_1, G_2) + (1 - \alpha) L_{Geo}(G_1, G_2)$$

where α is a value between 0 and 1. Images of the same subject should have both a high jet similarity and a high geometry similarity. By using both similarity measures, the chance that images from different subjects would have both a high jet similarity and a high geometry similarity would be reduced. Determining a good value for α is non trivial and is, therefore, reserved for future work.

A second possible improvement to the algorithm would be to use a weighted average of jet similarities. The current version of the algorithm assumes that all jets in the face graph are weighted equally. It is possible that some landmarks are better for recognition than others and should, therefore, carry more weight in the similarity computations.

A similar weighting method could be used to focus the algorithm on certain wavelets. It could be that certain orientations and frequencies of wavelets are good for face recognition. The algorithm could weight individual wavelet coefficients based on their recognition ability.

An example of this might be that low frequency wavelets are much more discriminating for face recognition than high frequency wavelets. If this is the case, the Gabor jet similarity computations could be weighted so that the lower frequencies carried more weight. In the same way certain orientations of wavelets could be more discriminating and would carry more weight in jet similarity measurement. Just like the combined jet and geometry similarity measures, determining weights for landmarks or Gabor coefficients is a non trivial process and is beyond the scope of this thesis.

REFERENCES

[1] J. Ross Beveridge, *Evaluation of face recognition algorithms*, WWW Page: http://www.cs.colostate.edu/evalfacerec, 2003.

[2] J. Ross Beveridge, Kai She, Bruce Draper, and Geof H. Givens, *A nonparametric statistical comparison of principal component and linear discriminant subspaces for face recognition*, Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, December 2001, pp. 535 – 542.

[3] Ross Beveridge, *Evaluation of face recognition algorithms web site*, `http:\\cs.colostate.edu\evalfacerec`.

[4] David S. Bolme, J. Ross Beveridge, Marcio L. Teixeira, and Bruce A. Draper, *The CSU Face Identification Evaluation System: Its Purpose, Features and Structure*, Proc. 3rd International Conf. on Computer Vision Systems (Graz, Austria), April 2003.

[5] Michael Kirby, *Geometric data analysis*, John Wiley & Sons, Inc., 2001.

[6] M. Kirby and L. Sirovich, *Application of the Karhunen-Loeve Procedure for the Characterization of Human Faces*, IEEE Trans. on Pattern Analysis and Machine Intelligence **12** (1990), no. 1, 103 – 107.

[7] B. Moghaddam, C. Nastar, and A. Pentland, *A bayesian similarity measure for direct image matching*, ICPR **B** (1996), 350–358.

[8] O. Nestares and R. Navarro and J. Portilla and A. Tabernero, *Efficient Spatial-Domain Implementation of a Multiscale Image Representation based on Gabor Functions*, submitted to Multidimensional Systems and Signal Processing, 1995.

[9] Kazunori Okada, Johannes Steffens, Thomas Maurer, Hai Hong, Egor Elagin, Hartmut Neven, and Christoph von der Malsburg, *The Bochum/USC Face Recognition System And How it Fared in the FERET Phase III test*, Face Recognition: From Theory to Applications (H. Wechsler, P. J. Phillips, V. Bruce, F. Fogeman Soulié, and T. S. Huang, eds.), Springer-Verlag, 1998, pp. 186–205.

[10] N. Petkov and P. Kruizinga, *Computational models of visual neurons specialised in the detection of periodic and aperiodic oriented visual stimuli: Bar and grating cells*, 1997, pp. 83–96.

[11] J. Phillips, *The feret database web site*, `http://www.itl.nist.gov/iad/humanid/feret/`.

[12] J. Phillips, H. Moon, S. Rizvi, and P. Rauss, *The FERET Evaluation*, Face Recognition: From Theory to Application (H. Wechslet, J. Phillips, V. Bruse, F. Soulie, and T. Hauhg, eds.), Springer-Verlag, Berlin, 1998.

[13] _____, *The FERET Evaluation Methodology for Face-Recognition Algorithms*, Tech. Report Technical Report Number 6264, NIST, 1999.

[14] P.J. Phillips, H.J. Moon, S.A. Rizvi, and P.J. Rauss, *The FERET Evaluation Methodology for Face-Recognition Algorithms*, T-PAMI **22** (2000), no. 10, 1090–1104.

[15] Laurenz Wiskott, Jean-Marc Fellous, Norbert Kruger, and Christoph von der Malsburg, *Face Recognition by Elastic Bunch Graph Matching*, Tech. Report 96-08, Ruhr-Universitat Bochum, April 1996.

[16] Wendy S. Yambor, *Analysis of pca-based and fisher discriminant-based image recognition algorithms*, Master's thesis, Colorado State University, 2000.

[17] W. Zhao, R. Chellappa, and A. Krishnaswamy, *Discriminant analysis of principal components for face recognition*, In Wechsler, Philips, Bruce, Fogelman-Soulie, and Huang, editors, Face Recognition: From Theory to Applications, 1998, pp. 73–85.

[18] W. Zhao, R. Chellappa, and P.J. Phillips, *Subspace linear discriminant analysis for face recognition*, UMD, 1999.

Appendix A: Matlab Code For Wavelet Experiment

```
function f = waveletexp()
%% This function illustrates how Gabor wavelets are used
%% to analyze the frequency information in a function.
% Create a function that is a sum of 10 sinusoids.
x = 0:.05:50;
f = 0*x;
for i = 1:10
    r1 =(2*rand-1);
    r2 =(2*rand-1);
    f = f + r1*sin(15*x/(2^i))+r2*cos(15*x/(2^i));
end
% Create wavelet masks that is sensitive to the third
% sinusoid.
y = -12:0.05:12;
for i = 1:max(size(y))
    w1(i) = exp(-0.025*y(i)^2)*cos(15*y(i)/(2^3));
    w2(i) = exp(-0.025*y(i)^2)*sin(15*y(i)/(2^3));
end
% Convolve the function with the wavelet masks.
c1 = conv(f,w1);
c2 = conv(f,w2);
% Resize the convolution arrays to only include the
% parts corresponding to the function.
[n,m] = size(f);
[p,q] = size(c1);
lowb = (q-m)/2;
highb = lowb+m-1;
c1 = c1(lowb:highb);
c2 = c2(lowb:highb);
% Compute the magnitude and phase transformation of
% the convolution values.
for i = 1:max(size(c1))
    cmag(i) = sqrt(c1(i)^2 + c2(i)^2);
    if ( c1(i) > 0 )
        cphase(i) = atan(c2(i)/c1(i));
    else
        cphase(i) = pi + atan(c2(i)/c1(i));
    end
```

```
end
% Plot the function
subplot(5,1,1);
plot(x,f,'-');
axis manual
axis( [min(x) max(x) min(f) max(f)] );
% Plot the wavelets
subplot(5,1,2);
plot(y,w1,'g-',y,w2,'b-');
axis manual
axis( [-25 25 -1 1] )
% Plot the raw convolution values
subplot(5,1,3);
plot(x,c1 ,'g-',x,c2,'b-');
axis manual
axis([min(x) max(x) min(c1) max(c1)]);
% Plot the magnitude of the values
subplot(5,1,4);
plot(x,cmag,'g-');
axis manual
axis([min(x) max(x) min(cmag) max(cmag)]);
% Plot the phase of the values
subplot(5,1,5);
plot(x,cphase,'b.');
axis manual
axis([min(x) max(x) min(cphase) max(cphase)]);
```

Appendix B: Similarity Computation Run Times

One issue in face recognition is how many face comparisons can be made in a short amount of time. The following table shows the estimated time required to compute a 3368X3368 distance matrix using a variety of different EBGM similarity measures. The test was run on an AMD Athlon 1800+ processor with 266MHz DDR ram running Mandrake Linux 9.1. These results show the relative speed of the similarity measures.

Similarity Measure	Estimated Time	Images/Second
GeoLeastSquares	3 min	63019
FGMagnitude	14 min	13504
FGPredictiveStep	1.5 hours	2100
FGPredictiveIter	2.5 hours	1260
FGNarrowingLocalSearch	34.5 hours	91
FGGridSample	3950 hours	0.79

Appendix C: Wavelet Specifications

Bolme

PARAMETER	SYMBOL	VALUES
Orientation	θ	$\{ 0, \pi/8, 2\pi/8, 3\pi/8, 4\pi/8, 5\pi/8, 6\pi/8, 7\pi/8 \}$
Wavelength	λ	$\{ 4, 4\sqrt{2}, 8, 8\sqrt{2}, 16 \}$
Phase	φ	$\{ -\pi/4, \pi/4 \}$
Gaussian Radius	σ	$\sigma = 3 * \lambda/4$
Aspect Ratio	γ	1

Nestares

PARAMETER	SYMBOL	VALUES
Orientation	θ	$\{ 0, 2\pi/8, 4\pi/8, 6\pi/8 \}$
Wavelength	λ	$\{ 4, 8, 16, 32 \}$
Phase	φ	$\{ 0, \pi/2 \}$
Gaussian Radius	σ	$\sigma \approx 0.575 * \lambda$ (see [8] for details)
Aspect Ratio	γ	1

Wiskott

PARAMETER	SYMBOL	VALUES
Orientation	θ	$\{ 0, \pi/8, 2\pi/8, 3\pi/8, 4\pi/8, 5\pi/8, 6\pi/8, 7\pi/8 \}$
Wavelength	λ	$\{ 4, 4\sqrt{2}, 8, 8\sqrt{2}, 16 \}$
Phase	φ	$\{ 0, \pi/2 \}$
Gaussian Radius	σ	$\sigma = \lambda$
Aspect Ratio	γ	1