2009-01-01

# Assessment Of Super-Resolution For Face Recognition From Very-Low Resoution Images

James Roger Roeder
*University of Texas at El Paso*, jrroeder@gmail.com

# ASSESSMENT OF SUPER-RESOLUTION FOR FACE RECOGNITION FROM VERY-LOW RESOUTION IMAGES

JAMES ROGER ROEDER

Department of Electrical & Computer Engineering, ECE

APPROVED:

Sergio D. Cabrera, Ph.D., Chair

Jose Gerardo Rosiles, Ph.D.

Christian Meissner, Ph.D.

Patricia D. Witherspoon, Ph.D.
Dean of the Graduate School

## Dedication

*To my family,*

*for always believing in me, encouraging me and supporting me throughout this endeavor.*

# ASSESSMENT OF SUPER-RESOLUTION FOR FACE RECOGNITION FROM VERY-LOW RESOUTION IMAGES

by

JAMES ROGER ROEDER, BSEE

THESIS

Presented to the Faculty of the Graduate School of

The University of Texas at El Paso

in Partial Fulfillment

of the Requirements

for the Degree of

MASTER OF SCIENCE

Department of Electrical & Computer Engineering, ECE

THE UNIVERSITY OF TEXAS AT EL PASO

May 2009

# Acknowledgements

I would like to express my gratitude and appreciation to the following people whose help during this research I found invaluable. I would like to thank two professors in the Wireless Sensor Networks Group. I need to thank Dr. Sergio Cabrera for introducing me to digital signal processing, image processing, and computer vision and inspiring me to pursue work in this area and for his guidance during this thesis. I also want to thank Dr. Cabrera for providing funding for my degree through the Texas Instruments Foundation Endowed Scholarship to help with the tuition. I would like to thank Dr. Gerardo Rosiles for accepting me as into the Wireless Sensor Networks Group.

Many of my colleagues were also instrumental to the completion of this thesis. Joel Quintana reminded me to work hard, but always take a break to not become overworked, and Ricardo Zapeda kept me relaxed even during the most stressful times.

This material is based upon work supported by the National Science Foundation under Grant No. CNS-0709438. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation. Network switches were provided by Cisco Systems, Inc.

# Abstract

Super-Resolution (SR) involves the registration of multiple images/frames and reconstruction of a single higher resolution image. The goal of this research is to use multiple, very-low resolution images, such as those produced from a video sequence in a wireless sensor network system, as input to the super-resolution process in a face recognition system. The algorithm used for face recognition is the Fisherfaces method with a nearest neighbor classifier used for the recognition decision. Super-resolution consists of two stages, a registration stage and a reconstruction stage.

The testing images were segmented using a simple skin color detection approach. After the cropping, the images were combined into groups of four from a sliding window that would take the current image and the following three images repeating this process by moving to the next image in the sequence and the subsequent three images until the end of the current class, or person, is reached. This same sliding window was used for the super-resolution algorithm using faces from the three people or classes. Each group of four images was used as an input to the Keren registration algorithm where the rotational and translation information was saved that was then entered into the robust super-resolution reconstruction algorithm to create a single high quality image, which was processed by the face recognition algorithm. The methods tested to compare were the average of the same groups of four, the centroid shifted average and the minimum of the four faces in the group. The comparison was based on nearest neighbor classifier and on classification rates. The results were not in favor of the super-resolution method but instead, the centroid shifted average was the best in this study.

# Table of Contents

# List of Tables

# List of Figures

# Chapter 1: Introduction

This section covers topics that introduce the general information regarding cameras, face recognition, and super-resolution. A literature review of face recognition, super-resolution, and the combination of face recognition and super-resolution is given to describe what has been done and what is currently being done, and the scope of the project is covered in the section as well.

## 1.1 CAMERA

A camera captures the certain range of electromagnetic waves that are reflected from an object and allows those wavelengths to be seen by a human eye. Electromagnetic waves are classified into different ranges depending on their wavelength such as visible light and infrared to name a few. The visible wavelength, which is what most cameras operate at, is broken down into the colors violet, blue, green, yellow, orange, and red and the wavelengths are between 400 $nm$ to 700 $nm$.

Most camera systems in use are CCD, charge-coupled device, or CMOS, complementary metal oxide semiconductor, devices. The CCD sensor has every pixel's charge transferred through a very limited number of output nodes that are then converted to voltage, buffered, and sent off-chip as an analog signal as shown in Figure 1.1. With the CMOS sensor each pixel has its own charge-to-voltage conversion, and the sensor often also includes amplifiers and other processing circuits resulting in the chip outputs being digital bits as shown in Figure 1.2 [1].

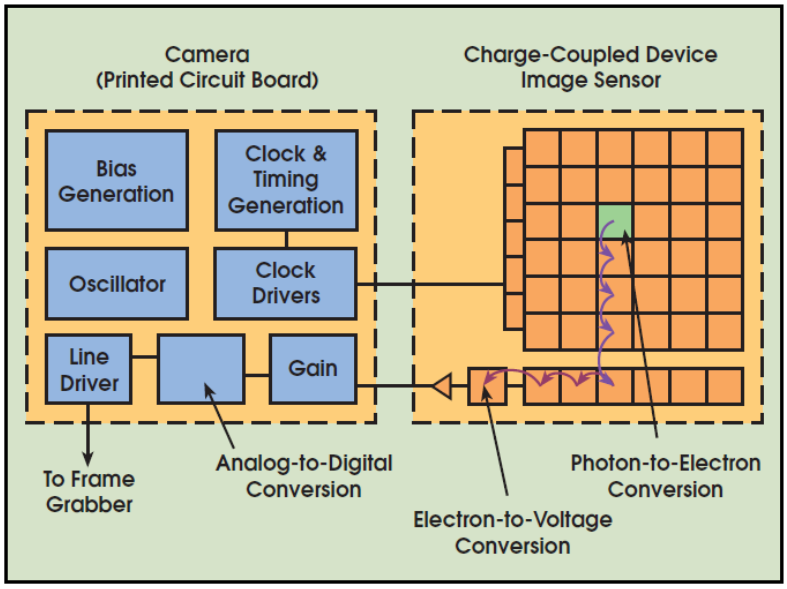


Figure 1.1: CCD Camera [1]

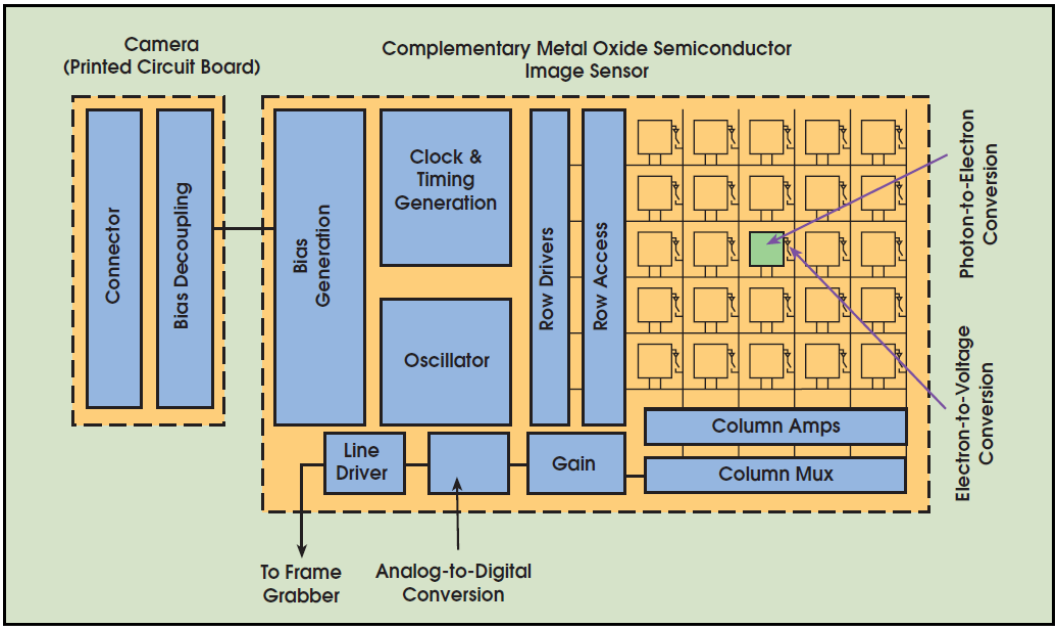


Figure 1.2: CMOS Camera [1]

## 1.2 WHAT IS FACE RECOGNITION?

Automated face recognition is clearly an important application of computer imaging though it remains a difficult problem and is not as reliable as would be desirable. Most advances have been made that include only small variations in lighting and in pose [2]. Some faces with medium to large

variations due to lighting are difficult for humans to identify and even harder for a computer to identify. The two most common methods for face recognition are Eigenfaces and Fisher Linear Discriminant (FLD) or Fisherfaces. Both methods take the images from a higher dimensionality to a lower dimensionality for classification. This lower dimensionality is useful in that fewer features are needed and therefore it is an ideal simplification for being sent through a low-capacity data channel like those encountered in wireless sensor networks. Each method reduces the dimensionality by using a projection matrix that is created using a small amount of images referred to as the training set of images. The training sets allows for the algorithm to know the different faces for each class.

### 1.3 WHAT IS SUPER-RESOLUTION?

When images are taken from a camera or a video source, the highest resolution is not always possible. Two main constraints in a wireless sensor network system are the low available data transmission bandwidth, and the need to use low-cost hardware. Some trade-offs must be made and the quality of the images is what will be sacrificed. However, the ability to take multiple images or a video sequence in a short period of time may allow for low quality images to be sufficient. High-resolution images can be reconstructed out of a series of low-resolution images. High-resolution or Super-Resolution (SR) involves registration of multiple images/frames and reconstruction of a single higher resolution image. This SR resulting image, called the high-resolution (HR) image, contains information from all of the images used and can be created to have multiple times as many pixels as the original images. The low-quality images used must have slight differences or the reconstructed image will not contain any new information and will be an interpolated version of any one of the identical input images.

Super-Resolution consists of two stages, a registration stage and a reconstruction stage. The registration stage is where the images are compared to each other to find any differences between the images in terms of rotations and linear shifts as shown in Figure 1.3. These types of differences can be from camera and/or object motion as shown in Figure 1.4. If the motion between the low-resolution images is a shift by an integer number of units, the resulting SR of HR image will not contain any new information. A good SR image will contain new information if the shifts have sub-pixel accuracy and are sampled at the same rate, that is, they are ideally subsampled and aliased versions of a higher

resolution ideal image which is what the SR method would attempt to recover [3]. Registration can be done either in the spatial domain or in the frequency domain. The spatial domain approach estimates the relative motion between the low resolution images, using nonuniform interpolation to produce a high quality image, and removes any blur or additional noise that was introduced. The frequency domain approach is based on the shifting property of the Fourier Transform, the aliasing relationship between the continuous Fourier transform of the Super-Resolution image and the discrete Fourier transform of the low resolution images, and the assumption that the signal is bandlimited.

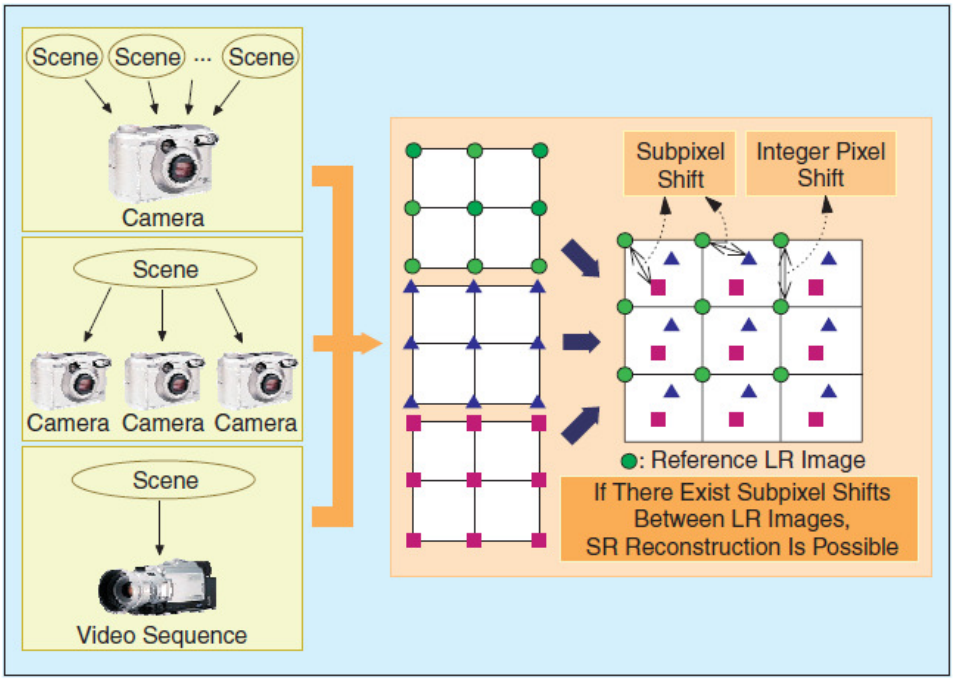


Figure 1.3: Sub-pixel Shifts Between Input Images [3]

The reconstruction stage is where the estimated registration parameters are taken and used to create a new, higher-resolution image. The higher resolution images are then mapped to an image that has 4 times as many pixels using interpolation as shown in Figure 1.5.

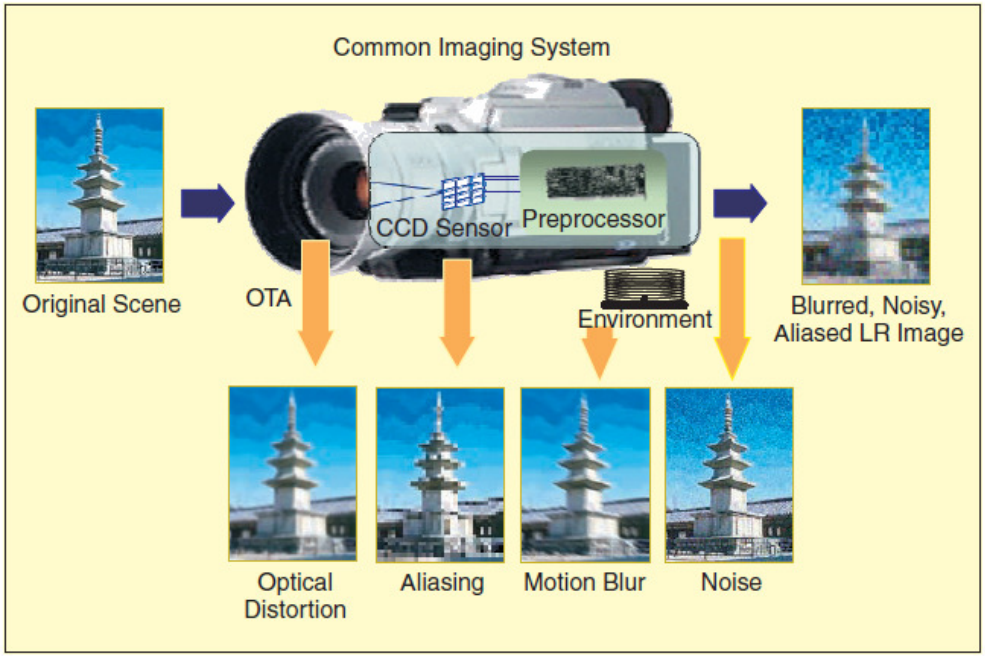


Figure 1.4: Common Sources of Sub-pixel Shifts [3]

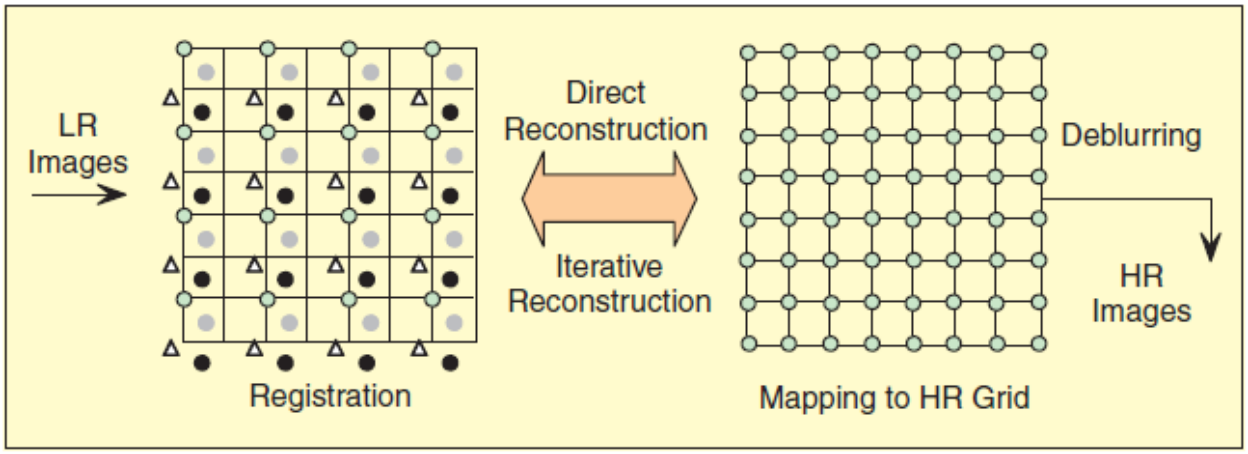


Figure 1.5: High-Resolution Interpolation [3]

## 1.4 Literature Review

### 1.4.1 Face Recognition

Face recognition has had many different methods proposed to solve the problem of having a computer classify a complex feature, a human face. A typical face recognition algorithm consists of three steps: face detection, feature extraction, and face recognition. In most cases the face detection and the feature extraction stages are done simultaneously, however, this is not always the case.

The face detection stage detects a face and performs a rough normalization of the face. Until the middle 1990's most segmentation was focused on segmenting a single face from a simple or complex background. The approaches used included a whole face template, a deformable feature-based template, skin color segmentation, and a neural network [4]. Recently, appearance or image based methods [5], [6] train machine systems using a large number of samples have achieved the best results. Some applications of the face detection stage are face tracking and pose estimation.

The feature extraction stage takes the face that has been detected and obtains features that are then inputted into the face classification system. The features that are extracted can be local features such as lines or points of reference or facial features such as eyes, nose, and mouth. There are three types of feature extraction methods currently: generic methods that are based on edges, lines and curves, feature template based methods detect facial features, and structural matching methods that consider geometrical constraints on the features [4]. Some applications of face detection alone are face tracking and pose estimation where feature extraction applications include face feature tracking, emotion recognition, and gaze estimation.

For face recognition there are three different types of approaches: holistic templates, feature geometry, and a hybrid between holistic templates and feature geometry. These approaches are used with intensity images and give a high-level categorization similar to how humans use holistic and local features. A guideline suggested by the psychological study of how humans use holistic and local features is shown in Figure 1.6.

| Approach |
| --- |
| Holistic Methods |
| *Principal-component analysis* (PCA) |
| Eigenfaces |
| Probabilistic Eigenfaces |
| Fisherfaces/subspace LDA |
| SVM |
| Evolution pursuit |
| Feature Lines |
| ICA |
| *Other representations* |
| LDA/FLD |
| PDBNN |
| Feature Based Methods |
| Pure geometry methods |
| Dynamic link architecture |
| Hidden Markov Model |
| Convolution Neural Network |
| Hybrid Methods |
| Modular Eigenfaces |
| Hybrid LFA |
| Shape-normalized |
| Component-based |

Figure 1.6: Further Classification of Face Recognition Methods [4]

The holistic methods use a whole face region while the feature matching methods use local features like the eyes, nose, and mouth which are extracted first while their locations and geometries are inputted subsequently into a structural classifier. The hybrid methods use a combination of the two methods similar to a human's perception.

### 1.4.2 Super-Resolution

Super-resolution is a signal processing technique to create a high resolution image from a sequence of low resolution images. Super-resolution consists of two separate and important stages: the registration stage and the reconstruction stage. Large amounts of work have been done in each stage and brief overviews of the most common methods are described.

The registration process for super-resolution can be done in either the spatial domain or the frequency domain. The advantage to using the spatial domain is that more general motion models are allowed [3]. Some spatial domain registration methods are by [7], which is an iterative planar motion estimation algorithm that is based on Taylor expansion using a pyramid scheme to increase the precision for large motion parameters. The frequency domain approaches are limited by the properties of the Fourier transform resulting in global motion models [3]. Only planar shifts and planar rotation and scale are considered in the frequency domain approaches. The advantage to these approaches is that it is much easier to describe and handle aliasing in the frequency domain than in the spatial domain. Most of the frequency domain registration approaches are based on the fact two shifted images differ by a phase shift only in the frequency domain, which can be found from their correlation [8]. A log-polar transform of the magnitude of the frequency spectrum, image rotation and scale can be converted into horizontal and vertical shifts. [8] apply a phase correlation technique to estimate the planar shifts. To minimize the errors due to aliasing, [8] uses the low-frequency part of the images since that part of the frequency domain is almost alias free.

Once the images have been properly registered, they can be reconstructed into a single high resolution image. Spatial domain approaches usually use interpolation based approaches [Keren] and others use an estimate of the image using an imaging model that is then updated during each iteration until an acceptable amount of error is reached or the number of preset iterations has been reached [9]. [9] uses the median of the errors improving the results and making the algorithm more robust when outliers are present. Another reconstruction method is to use a stochastic approach. Stochastic super-resolution usually involves a Bayesian approach that is flexible and a convenient way to model a priori knowledge [3]. A maximum a priori (MAP) estimator is used to recreate the high resolution image that will be stable because of the priori constraints [3], [8], [10], [11]. The most common priori assumptions are Gaussian Markov random fields (GMRF) or Huber Markov random fields (HMRF) [10]. HMRF's are used because the GMRF priors are too smooth and sharp edges are lost. The HRMF's use a Gibb's prior

$$p(x) = \begin{cases} x^2 & if\ |x| \leq \alpha \\ 2\alpha|x| - \alpha^2 & otherwise \end{cases}$$

where $x$ is the first derivative of the image and $\alpha$ is a scaling factor shown in Figure 1.7.

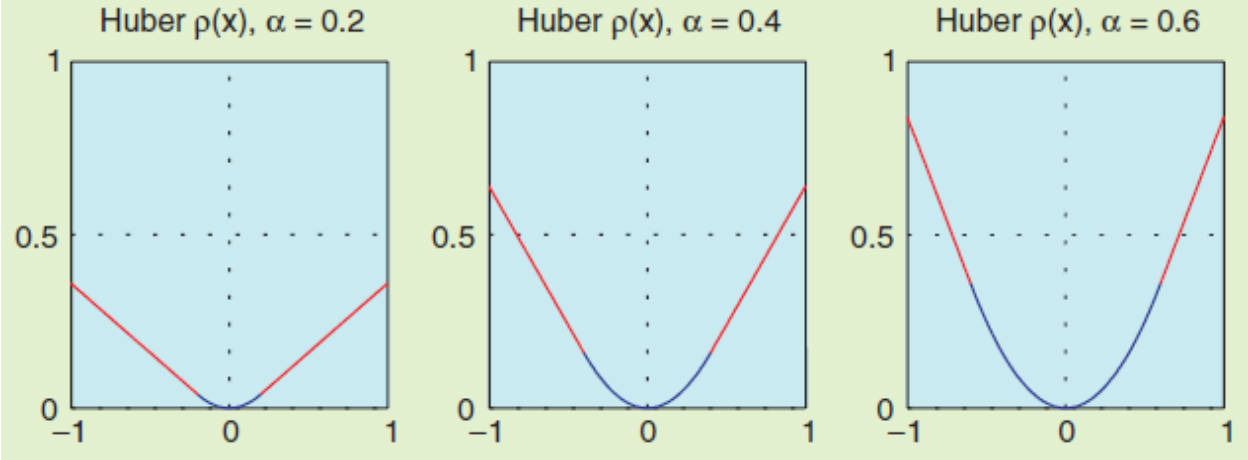


Figure 1.7: Different Values of $\boldsymbol{\alpha}$ [10]

### 1.4.3 Combining Face Recognition and Super-Resolution

Combining super-resolution and face recognition is usually a simple and straight forward procedure as shown in Figure 1.8. This combination takes the low resolution images and applies super-resolution to the images creating a single high-resolution image which is then fed into the face recognition algorithm. This method is the most intuitive and most common approach when using super-resolution.

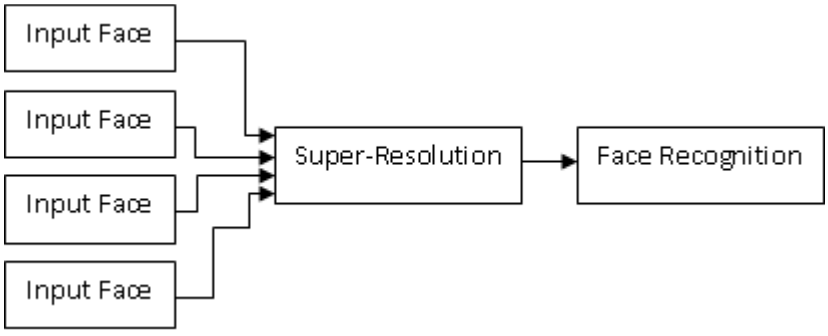


Figure 1.8: Normal Combination for Face Recognition and Super-Resolution

An alternative approach to combining face recognition and super-resolution is to take the low resolution input face images and transform those images into the face space and perform the face recognition in that domain instead of the spatial or pixel domain before super-resolution is applied as

shown in Figure 1.9. An advantage to using this method is that the computational complexity is greatly reduced from the traditional combination of face recognition and super-resolution [11].

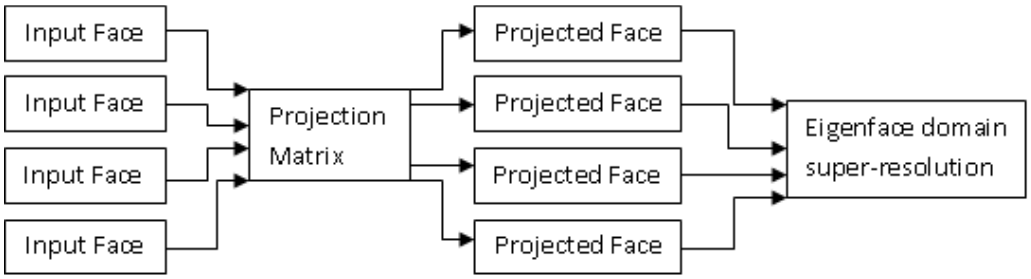


Figure 1.9: Alternative Combination for Face Recognition and Super-Resolution

## 1.5 SCOPE OF PROJECT

The problem with face recognition is that it is unreliable when large changes are made to the face such as lighting and pose [12]. If the change in lighting or pose is not in the training set, the algorithm will not be able to accurately identify that person. This thesis addresses the problem of variations in pose using super-resolution and other techniques to try to consistently and accurately detect the correct person by having the person rotate their head to see if the images chosen in the training set will be enough to accurately recognize or identify the face. The training set will contain some rotated faces and try to estimate the remaining rotated faces found in the testing set. The other techniques used to compare against super-resolution will be an average of the cropped faces, a minimum of the four faces, and the centroid shifted average.

# Chapter 2: Theoretical Background

This section covers topics that are necessary to understand how the face recognition and super-resolution algorithms work as well as some of the morphological processes used to enhance some of the images. A review is provided for face recognition, Eigenfaces and Fisherfaces, super-resolution, registration and reconstruction, the color segmentation algorithm including the color spaces used, as well as morphology to improve the results of the color segmentation algorithm.

## 2.1 FACE RECOGNITION

### 2.1.1 Eigenfaces

Eigenfaces is a technique based on linearly projecting the image space to a low dimensional feature space, the face space. The Eigenfaces method uses the principle component analysis (PCA) also known as the Karhunen-Loeve transformation for dimensionality reduction whose projection maximizes the scatter of all the projected images. This approach transforms the faces into a small set of feature images that are the principle components of the training set images [13]. The faces are then classified by comparing the position of the current face in the face space with the positions of known individuals, the training faces. One disadvantage of using Eigenfaces is the projection matrix tends to smear the classes together making them unable to be linearly separable.

Let us say we have a set of $N$ training images $\boldsymbol{x}_1, \boldsymbol{x}_2, \dots, \boldsymbol{x}_N$ ordered as a vector in a $n$-dimensional image space and each image belongs to one of $c$ classes $\boldsymbol{X}_1, \boldsymbol{X}_2, \dots, \boldsymbol{X}_c$. A linear transformation that maps the original $n$-dimensional image space to a new $m$-dimensional feature space that is less than the original image space, $m<n$ is

$$\boldsymbol{y}_i = \boldsymbol{W}^T \boldsymbol{x}_i \qquad\qquad i = 1, 2, \dots, N$$

where $W \in \mathbb{R}^{nxm}$ is a matrix with each of the columns being orthonormal. The total scatter matrix is found by

$$S_T = \sum_{i=1}^{N} (\boldsymbol{x}_i - \boldsymbol{\mu})(\boldsymbol{x}_i - \boldsymbol{\mu})^T$$

where $N$ is the total number of trainings images. The matrix $\boldsymbol{\mu}$ is the sample mean of the training images and is calculated by

$$\boldsymbol{\mu} = \frac{1}{N}\sum_{i=1}^{N} \boldsymbol{x}_i$$

where $N$ is the total number of training images. After applying the transformation $\boldsymbol{W}^T$ the total scatter of the feature vectors $\boldsymbol{y}_1, \boldsymbol{y}_2, \ldots, \boldsymbol{y}_N$ becomes $\boldsymbol{W}^T S_T \boldsymbol{W}^T$, which is used to find the optimum projection matrix. The optimum projection matrix is

$$W_{opt} = \arg\max_{W} |\boldsymbol{W}^T S_T \boldsymbol{W}^T|$$

$$= [\boldsymbol{w}_1\ \boldsymbol{w}_2 \ldots \boldsymbol{w}_m]$$

where $\boldsymbol{w}_1\ \boldsymbol{w}_2 \ldots \boldsymbol{w}_m$ are the set of eigenvectors of the scatter matrix $S_T$ that correspond to the $m$ largest eigenvalues.

The disadvantages of the Eigenfaces method is the projection will actually smear the classes together. This smearing prevents the classes from being linearly separable in the projected space. In good cases the smearing of the classes does not interfere with the classification, but that is not always the case.

### 2.1.2 Fisherfaces

Fisherfaces is another algorithm used for face recognition that tries to re-shape the scatter in feature space to make classification more reliable. While using PCA projections are optimal for representation of all classes together, they are not always optimal for discriminating between two or more classes [12]. PCA actually smears the classes together while the Fisherfaces method increases the between-class scatter making classification simpler and more reliable. The Fisherfaces method combines the Fisher Linear Discriminant with the PCA to achieve optimal dimensionality reduction and improved classification

Let us say we have a set of $N$ training images $\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_N$ ordered as a vector in a $n$-dimensional image space and each image belongs to one of $c$ classes $\boldsymbol{X}_1, \boldsymbol{X}_2, \ldots, \boldsymbol{X}_c$. A linear transformation that maps the original $n$-dimensional image space to a new $m$-dimensional feature space with $m < n$ is implemented by the projection:

$$\boldsymbol{y}_i = \boldsymbol{W}^T \boldsymbol{x}_i \qquad\qquad i = 1, 2, \ldots, N$$

The between class scatter is the scatter found between the different classes. If the between class scatter is large, classification is easier. The between class scatter is found by

$$S_B = \sum_{i=1}^{c} N_i (\boldsymbol{\mu}_i - \boldsymbol{\mu})(\boldsymbol{\mu}_i - \boldsymbol{\mu})^T$$

where $\boldsymbol{\mu}_i$ are the sample means for each class, $\boldsymbol{\mu}$ is the sample mean of the whole training set, $N_i$ is the number of images in the i[th] class, and $c$ is the total number of classes in the training set. The within class scatter is the amount of scatter that is found in each class. Each difference in the training images for a specific class adds to the within class scatter. The total within class scatter is calculated by

$$S_W = \sum_{i=1}^{c} \sum_{\boldsymbol{x}_k \in X_i} (\boldsymbol{x}_k - \boldsymbol{\mu}_i)(\boldsymbol{x}_k - \boldsymbol{\mu}_i)^T$$

where $\boldsymbol{x}_k$ are the images found in the training set. The projection matrix for Fisherfaces is such that the projection matrix has orthonormal columns as well as maximizing the ratio between the determinant of the projected between class scatter to the determinant of the projected within class scatter. Thus, the optimum projection matrix is found by

$$W_{opt} = \arg\max_W \frac{|W^T S_B W|}{|W^T S_W W|}$$

$$= [\boldsymbol{w}_1 \; \boldsymbol{w}_2 \; \ldots \; \boldsymbol{w}_m]$$

where $\boldsymbol{w}_1 \; \boldsymbol{w}_2 \; \ldots \; \boldsymbol{w}_m$ are the generalized set of eigenvectors of $S_B$ and $S_W$ corresponding to the m largest generalized eigenvalues. The most important property of this matrix is its direction not the magnitude. One advantage of Fisherfaces is that the dimensionality can be reduced to N-c in the feature space [Belhumeur]. This dimensionality reduction is achieved by using PCA to reduce the dimensionality of the feature space then applying the projection for Fisherfaces. The additional reduction is calculated by

$$W_{opt}^T = W_{FLD}^T W_{PCA}^T$$

where

$$W_{PCA} = \arg\max_W |W^T S_T W|$$

$$W_{FLD} = \arg\max_W \frac{|W^T W_{PCA}^T S_B W_{PCA} W|}{|W^T W_{PCA}^T S_W W_{PCA} W|}$$

where $S_T$ is the total scatter found in Eigenfaces and can be calculated from [14] as

$$S_T = S_W + S_B$$

## 2.2 SUPER-RESOLUTION

### 2.2.1 Keren Registration Method

The Keren registration method uses sub-pixel shifts from a number of low resolution images to create a high resolution image. In this method the image is kept in the spatial or image domain instead of the frequency domain because the Fourier transform is sensitive to noise in an already noisy environment [7]. Another reason to keep the image in the spatial domain is that most cameras sample the image. In the frequency domain, the frequencies that were lost due to sampling cannot be recovered; however, in the spatial domain lines, edges, or other features will remain and can be located easily.

Sub-pixel accuracy is used to find shifts that are less than a full pixel apart. A high resolution image can still be reconstructed with inaccurate registration, but the image will not be an accurate or good representation of the low resolution images. An accurate sub-pixel registration from [7] is found by

$$g(x,y) = f(x\cos\theta - y\sin\theta + a, y\cos\theta + x\sin\theta + b)$$

where $f$ and $g$ are two frames that are considered two functions, $a$ is the horizontal shift, $b$ is the vertical shift, and $\theta$ is the angle of rotation around the origin. An approximation to find $g(x,y)$ can be made using the first two terms of a Taylor series expansion of $f$ yielding

$$g(x,y) \approx f(x + a - y\theta - x\theta^2/2\,, y + b + z\theta - y\theta^2/2)$$

If $f$ is now expanded to the first term of its Taylor series it gives

$$g(x,y) \approx f(x,y) + (a - y\theta - x\theta^2/2)\frac{df}{dx} + (b + x\theta - y\theta^2/2)\frac{df}{dy}$$

The error of the approximation between $f$ and $g$ after the rotation $\theta$ and the shifts $x, y$ takes the form

$$E(a,b,\theta) = \sum\left[f(x,y) + (a - y\theta - z\theta^2/2)\frac{df}{dx} + (b + x\theta - y\theta^2/2)\frac{df}{dy} - g(x,y)\right]^2$$

where the summation is over the overlap between $f$ and $g$. This process is repeated with a new $g$ until the error is very small or a set number of iterations have been reached.

### 2.2.2 Robust Super-Resolution

When reconstructing the image, the model used in the registration stage needs to be accurate. Any inaccuracies during the registration stage could cause the reconstructed image to degrade instead of being enhanced. Robust Super-Resolution is a reconstruction technique that uses the pixel-wise median filter to achieve robustness from outliers [9] and minimizes the error under a norm that is more robust than the $\ell_2$ norm. Robust Super-Resolution is based on the model initially used by [15] and keeping the same vector image framework. The input images, $g_1, g_2, \ldots, g_n$, reshaped each from a $n \times m$ matrix to a $(n * m)x1$ vector, can be assumed to be created as follows

$$\vec{Y}_k = D_k C_k F_k \vec{X} + \vec{E}_k$$

where $\vec{X}$ is the high resolution image as a vector, $\vec{Y}_k$ is the $k^{\text{th}}$ input image $g_k$ as a $(n * m)x1$ vector, $\vec{E}_k$ is the normally distributed additive noise as a $(n * m)x1$ vector, $F_k$ is the geometric warp matrix, $C_k$ is the blurring matrix, and $D_k$ is the decimation matrix. The total squared error of the high resolution image is

$$L(\vec{X}) = \frac{1}{2}\sum_{k=1}^{n} \left\| \vec{Y}_k - D_k C_k F_k \vec{X} \right\|_2^2$$

which can be minimized by taking the derivative of $L$ with respect to $\vec{X}$. The gradient of $L$ is the sum of the gradients over all of the input images

$$\nabla L(\vec{X}) = \sum_{k=1}^{n} \vec{B}_k$$

where $\vec{B}_k$ is the back-projection difference image. The back-projection difference images are found by

$$\vec{B}_k = F_k^T C_k^T D_k^T \left( D_k C_k F_k \vec{X} - \vec{Y}_k \right)$$

A simple gradient-based iterative minimization method is used to update the estimated solution in each iteration by

$$\vec{X}^{n+1} = \vec{X}^n + \lambda \nabla L(\vec{X})$$

where $\lambda$ is a scale factor defining the step size in the direction of the gradient. In the image domain this is a version of the Iterated Back Projection method [16]. During each iteration, the high resolution (HR) image estimated is resampled in the lattices of the input image. The difference between the resampled HR image and the input image is then projected back to the high resolution lattice. To make the

algorithm more robust, a pixel-wise median filter is applied to a scaled median back-projected difference image to approximate the gradient of $L$ by

$$\nabla L(\vec{X}) \approx n \cdot median\{\vec{B}_k(x,y)\}_{k=1}^{n}$$

The median filter can approximate the mean accurately if the distribution is symmetric and the median is more robust to outliers than the mean. If the distribution is non-symmetric with respect to the mean, the median estimate can be biased. A bias detector is then used to detect most of the outliers that could bias the median estimator. The bias detector mask is created by

$$e_k^j = g_k - P_j(f^{(j)})$$

where $e_k^j$ is the error image, $f^{(j)}$ is the current estimate of the high resolution image, $g_k$ is the input image of the $j^{\text{th}}$ iteration, and $P_j(f^{(j)})$ is the resampling of $f^{(j)}$ on the lattice of image $g_k$ using the blur, warp, and decimation operators. In a typical case $e_k^j$ is nonzero due to aliasing and the intensity of the pixels in the error image, which has zero mean noise, is concentrated near intensity edges. The intensity edges are positive and have a neighboring negative value, so the local mean will be close to zero.

## 2.3 COLOR SEGMENTATION

Color segmentation is an important step in utilizing any sort of visual biometric scan. In face recognition the ability to distinguish the difference between a person's skin and the background is an important step in improving the accuracy of the algorithm. By getting rid of the background and keeping only skin pixels, the algorithm will not be confused or have to deal with different backgrounds that might interfere with classification. There are three different types of skin models: parametric, nonparametric, and explicit skin cluster definition models [17]. The parametric model assumes that the skin color distribution can be modeled as with an elliptical Gaussian joint probability density function in the color space. The nonparametric method estimates the skin color distribution from the histogram of the training data with no skin color model. The explicit skin color is the simplest method and most commonly used method that pre-specifies the boundaries of the skin cluster in different color spaces.

### 2.3.1 RGB Color Space

The RGB color space is best defined as a cube where the three axes are the red, green, and blue colors. The RGB space can be defined as all possible colors made from the three primary colors shown

in Figure 2.1. Displaying a color using the RGB color space is defined as a three coordinate point in the cube, (R,G,B). Depending on the application and the amount of information being sent through the channel, RGB can have anywhere from 0 to 255 values for 8-bit values or 0 to 65535 values for 16-bit channels. The RGB color space has been used in photography, television, and computer monitors. An example of some colors from the vertices of the cube is given in Figure 2.2.

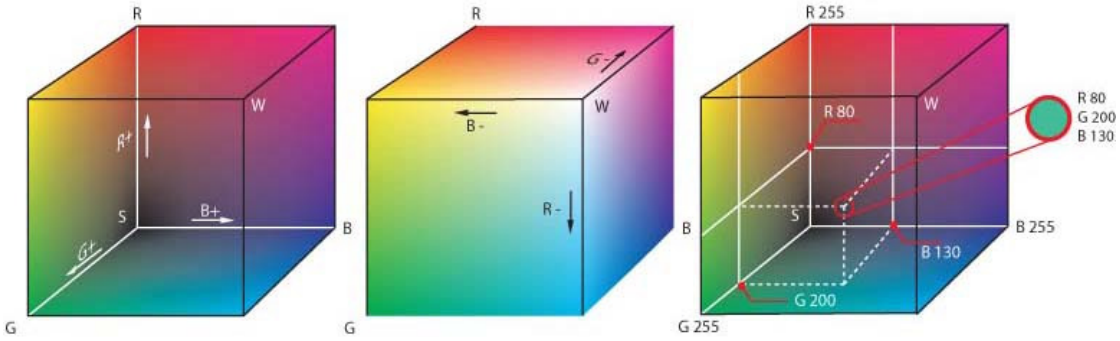


Figure 2.1: RGB Color Cube [18]

The RGB color space is what is found in most common in cameras, and there are two different boundary conditions depending on the type of light the subject is under. For uniform daylight the boundaries from [17] are

$$R > 95, G > 40, B > 20$$

$$Max\{R, G, B\} - min\{R, G, B\} < 15$$

$$|R - G| > 15, R > G, R > B$$

If the light sources are a flashlight or there is lateral illumination caused by daylight, the boundaries from [17] are found to be

$$R > 220, G > 210, B > 170$$

$$|R - G| \leq 15, B < R, B < G$$

The problem with using the RGB color space is that the algorithm will have to know what kind of lighting is present in the image being processed. This would be a difficult problem to solve; however, another solution would be to use boundaries from a different color space.

| | Nominal Range | White | Yellow | Cyan | Green | Magenta | Red | Blue | Black |
|---|---|---|---|---|---|---|---|---|---|
| **R** | 0 to 255 | 255 | 255 | 0 | 0 | 255 | 255 | 0 | 0 |
| **G** | 0 to 255 | 255 | 255 | 255 | 255 | 0 | 0 | 0 | 0 |
| **B** | 0 to 255 | 255 | 0 | 255 | 0 | 255 | 0 | 255 | 0 |

Figure 2.2: Some Values from the RGB Color Cube

### 2.3.2 YCbCr Color Space

The YCbCr color space is used in video and digital images to represent color. The Y component is the luminance, intensity values, the Cb component is the blue difference (B-Y), and the Cr component is the red difference (R-Y). While the RGB color space defines three component values for each pixel of the image, the YCbCr color space can be more flexible depending on the encoding scheme. YCbCr is more adaptable due to the fact that the human eye is more sensitive to variations in intensity than in variations in color [19]. Some examples of different YCbCr encoding schemes are YCbCr 1:1:1, where there is one sample for each component, YCbCr 2:1:1, where there are two samples for Y and 1 sample for each of the other two components, and YCbCr 4:1:0, where a 4×4 block of pixels is represented by 16 Y samples, 1 for each pixel, but all 16 pixels share a Cb sample and a Cr sample.

For skin segmentation in our application, the YCbCr is used for simplicity instead of the RBG color space. The suitable boundaries in the YCbCr color space from [17] are found to be

$$77 \leq Cb \leq 127 \; and \; 133 \leq Cr \leq 173$$

The YCbCr color space works better than the RGB boundaries because there is no additional information about lighting required and the ranges are easier to calculate.

## 2.4 BINARY IMAGE CENTROID

A binary image is an image that is composed of zeros and ones. The zeros correspond to the black pixels while the ones correspond to the white pixels as shown in Figure 2.3.

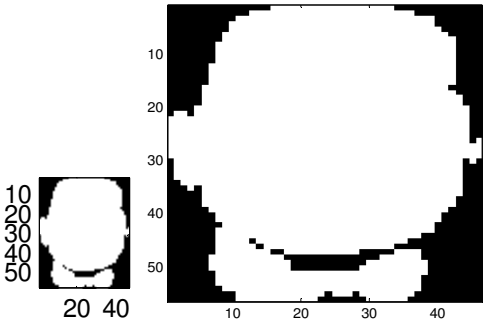


Figure 2.3: Binary Image and a Larger Version of the Same Image

The centroid of a binary image is the midpoint along each row and column axis corresponding to the "middle" based on the spatial distribution of pixels within the object [20]. This means that the centroid is the pixel that is in the center of the rows and the pixel that is at the center of the columns in the binary image. The equation to find the centroid is given by

$$\bar{r}_i = \frac{1}{A_i}\sum_{r=0}^{N-1}\sum_{c=0}^{M-1} rI_i(r,c) \quad ; \qquad \bar{c}_i = \frac{1}{A_i}\sum_{r=0}^{N-1}\sum_{c=0}^{M-1} cI_i(r,c)$$

where $i$ is the $i^{\text{th}}$ object in the binary image, $I(r,c)$ is the value of the binary image at the point $(r,c)$, and $A_i$ is the area of the $i^{\text{th}}$ object. The area of the $i^{\text{th}}$ object is calculated by

$$A_i = \sum_{r=0}^{N-1}\sum_{c=0}^{M-1} I_i(r,c)$$

In this case $i$=1 because of the assumption that the face will be the only object in the binary image. The centroid of Figure 2.3 is found to be $(\bar{r},\bar{c}) = (24,27)$ because the pixels in any image are integers.

## 2.5 MORPHOLOGY

Morphological filtering is done to simplify a segmented image so objects of interest can be found. Some examples of what morphology can do to an image are smoothing out object outlines, filling in small holes, and erasing small projections to name a few. Morphology can be done on binary images and grayscale images. In this project only the binary images will have the morphological filters applied to them. The two principal morphological operations are dilation and erosion. Both dilation and erosion are performed by using a structuring element to determine if the pixel of interest changes or remains the same. This structuring element is a smaller matrix usually but not always with odd dimensions such as a $3 \times 3$ or a $5 \times 5$. The structuring element is placed over the matrix with the center pixel being compared to the current pixel in the image. The two pixels are compared and either an

operation is performed or the next pixel is chosen until the entire image has gone through this process. When the structuring element is beyond the boundaries of the image, those values are either skipped or a buffer row or column is placed containing zeros to account for the mismatch.

### 2.5.1 Dilation

Dilation is a morphological operation that allows an object to expand allowing for small holes to be filled or separate objects to be joined. Dilation is described as

1. If the origin of the structuring element coincides with a "0" in the image, there is no change; move to the next pixel.
2. If the origin of the structuring element coincides with a "1" in the image, perform the OR logic operation on all pixels within the structuring element [20].

The OR logic operation is defined by Table 2.1.

Table 2.1: Logical OR Operation

| **X** | **Y** | **Result** |
|---|---|---|
| **0** | 0 | 0 |
| **0** | 1 | 1 |
| **1** | 0 | 1 |
| **1** | 1 | 1 |

An example of a dilation operation on an image is given by

$$\begin{bmatrix} 1&1&1&1&1&1&1 \\ 1&0&0&1&0&0&1 \\ 1&1&1&0&0&1&1 \\ 1&0&0&0&0&0&1 \\ 1&1&1&1&0&0&1 \\ 1&1&0&0&0&1&1 \\ 1&1&0&0&1&1&1 \end{bmatrix}$$

using the structuring element,

$$\begin{bmatrix} 1 & 0 & 0 \\ 1 & 1 & 1 \\ 1 & 0 & 0 \end{bmatrix}$$

the resulting image after the dilation operation if the boundary pixels are ignored becomes

$$\begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 1 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix}$$

The dilation operation filled in the small holes, and will also expand the boundary of any object. Some of the holes will not be filled in with the current structuring element. The structuring element determines how much of the holes will be filled out. Too aggressive the structuring element and unwanted elements could be added to the image, while a weaker structuring element will not fill in the holes.

### 2.5.2 Erosion

Erosion is a morphological operation that shrinks objects by etching away boundaries. Erosion is similar to dilation but instead of trying to produce ones the erosion operation tries to produce zeros. Erosion can be described as:

1. If the origin of the structuring element coincides with a “0” in the image, there is no change; move on to the next pixel.
2. If the origin of the structuring element coincides with a “1” in the image, and any of the “1” pixels in the structuring element extend beyond the object (“1” pixels) in the image, then change the “1” pixel in the image, whose location corresponds to the origin of the structuring element to a “0” [20].

An example of an erosion on an image is given by

$$\begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 1 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 1 & 1 & 1 \end{bmatrix}$$

using the structuring element,

$$\begin{bmatrix} 1 & 0 & 0 \\ 1 & 1 & 1 \\ 1 & 0 & 0 \end{bmatrix}$$

the resulting image after the erosion operation becomes

$$\begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 1 & 0 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 & 1 & 1 & 1 \end{bmatrix}$$

The erosion operation keeps only elements that coincide to the origin of the structuring element where the entire structuring element was contained in the image.

### 2.5.3 Opening and Closing

The two basic morphological operations, erosion and dilation, can be combined to form a more complex morphological operation like an opening or a closing. An opening consists of an erosion followed by a dilation. The opening is used to remove regions too small to be on object of interest, expands holes, and erodes edges. The erosion and the dilation operations can have a different structuring element for each individual operation or the same structuring element can be used for both operations. The closing consists of a dilation followed by an erosion and is used to fill in holes and small gaps.

# Chapter 3: Methodology

## 3.1 OVERVIEW

Figure 3.1 shows the block diagram for the face recognition algorithm and the individual algorithms that are compared to help improve the Fisherfaces face recognition algorithm. Three people referred to as classes will be tested. The video sequence will have all of the frames extracted, and those frames will be cropped and resized. The resized images will then be used to create the training set, which will be the small number of images that all the other images are compared against, the W or projection matrix that transforms the images from the spatial domain to the face domain, and the testing set that contains all the images that were not chosen for the training set. The testing set will then have face recognition performed to it as well as super-resolution, the average, the minimum of the group of four, and the centroid shifted algorithms to determine which one improved face recognition the most. The faces shown have been artificially brightened for clarity and the original brightness values were used during the experiments.

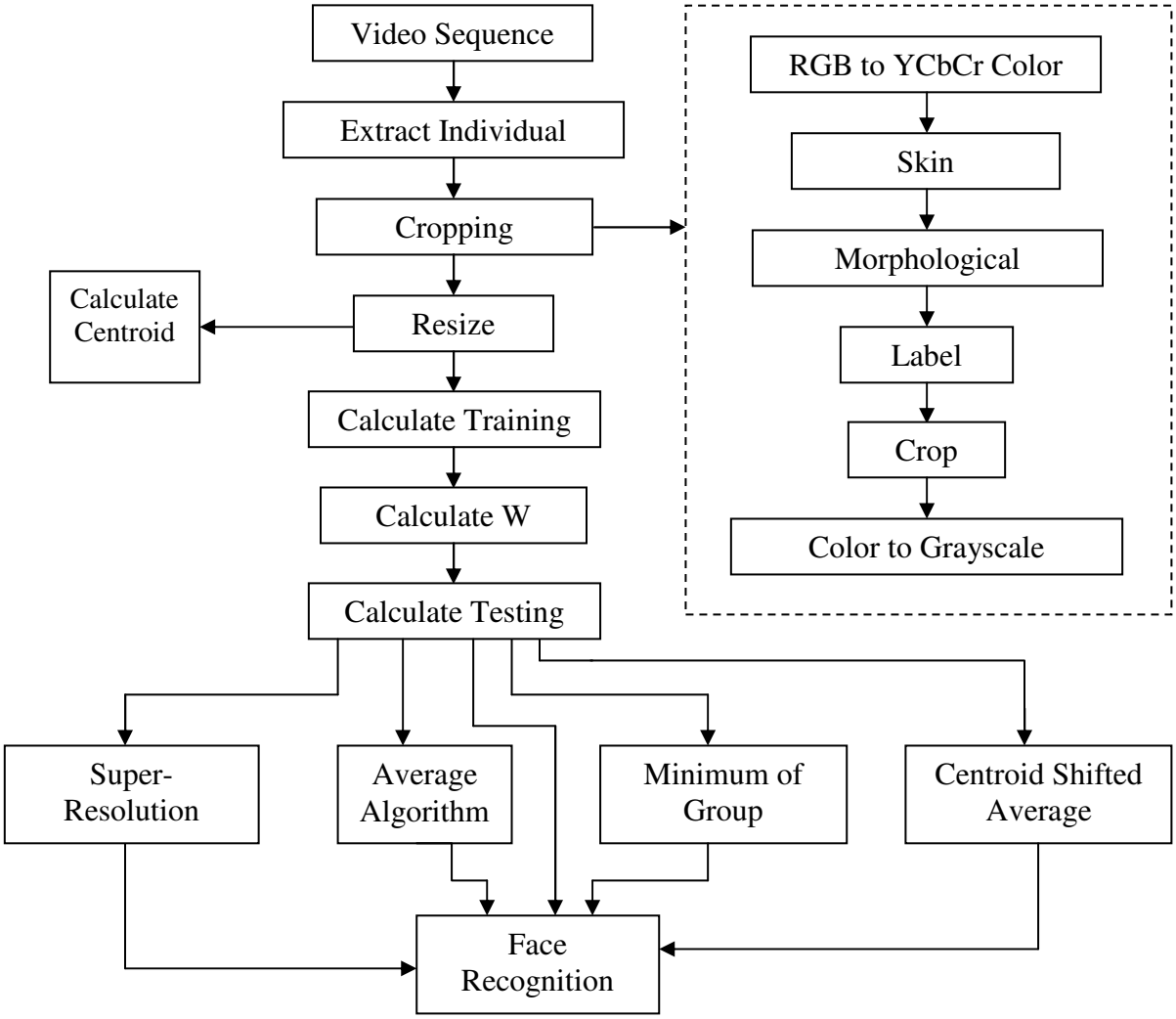


Figure 3.1: Block Diagram

## 3.2 DATA COLLECTION

### 3.2.1 Veo Personal Web Camera

The Veo Personal Web Camera was chosen because of its 320x240 resolution output at 30 frames per second. The setup for capturing the video was taken with low external lighting except for the light source used to illuminate the face at a distance of 2 feet. The lighting was chosen to try to be as uniform as possible to avoid causing too many bright spots or reflections along the face compared to the rest of the image. An example of the setup for the taking the video is shown in Figure 3.2.

Figure 3.2: Data Collection Setup

The subject will then rotate their head from left to right for three complete rotations then go back to the front.

### 3.2.2 Experimental Setup for Data Collection

The experiment consists of three people, or classes, with similar skin tones for a homogeneous experiment ensuring the results are accurate by having subjects with the least amount of distinct differences as possible. Each person will be sitting in the same chair about two feet away from a light source and rotate their heads from right to left three times. The heads will be rotated 90° from the center in both the left and right directions. The room will be kept dark to allow for only the light source to illuminate the face to try to get as uniform lighting as possible. Uniform lighting is needed so the face does not reflect too much light, which will affect the skin segmentation algorithm. The background has a black sheet to prevent the background from interfering with the experiment. If the background is incorrectly classified as part of the face, the results can influence how the face is cropped and contain extra information that will be kept when the face is reduced and skew the results for that group of faces.

### 3.2.3 RGB Color Video

MATLAB takes video outputs and displays the result as a $row \times column \times dimension \times frame$ structure. MATLAB is unique in the way it stores the values of video and still images. Other software programs and programming languages store video and images using the $column \times row$ format, but MATLAB reverses the convention by storing video and images using the $row \times column$

format. The row and column refer to the total number of rows and columns where the dimension is three for color schemes and one for grayscale. The frame dimension is used only in video and is the current frame being processed in the entire video sequence. The default color scheme in MATLAB is 8 bit RGB. Each dimension in the RGB color scheme has 8 bits resulting in a total of 24 bits color. The output from the camera will be a video with 240x320 pixels in 24 bits color. Each frame will then be extracted to have further processing done before any face recognition or super-resolution can be applied.

### 3.2.4 Video to Still Images

After the video is loaded and read into MATLAB using the `mmreader()` command. After the video is loaded into MATLAB, the video is read to get the individual frames and their respective colormap, which is RGB for all frames taken from the camera. After all of the video frames are read, the total number of frames are then counted. In a *.wmv file the number of frames is variable and will give a warning in MATLAB but will give an error in previous versions.

```
readerobj=mmreader('ricardo.wmv');
vidFrames=read(readerobj);
numFrames=get(readerobj,'numberOfFrames');
```

After the total number of frames are counted, the individual color frames must be taken and processed individually from each other. The frames and the colormaps are stored in a structure in MATLAB and must be loaded in a particular way to successfully extract the frames.

```
mov(k).cdata=vidFrames(:,:,:,k);
mov(k).colormap=[];
```

The individual frame can now be processed by the next step, the skin segmentation algorithm.

## 3.3 CROPPING AND RESIZING

### 3.3.1 Skin Segmentation

The frame then goes into the skin segmentation algorithm where it is converted from the RGB color space into the YCbCr color space. The image is then compared to the skin boundary conditions to

see if the pixel is a skin pixel or not. After the image has been processed, an initial mask is created as shown in Figure 3.3.

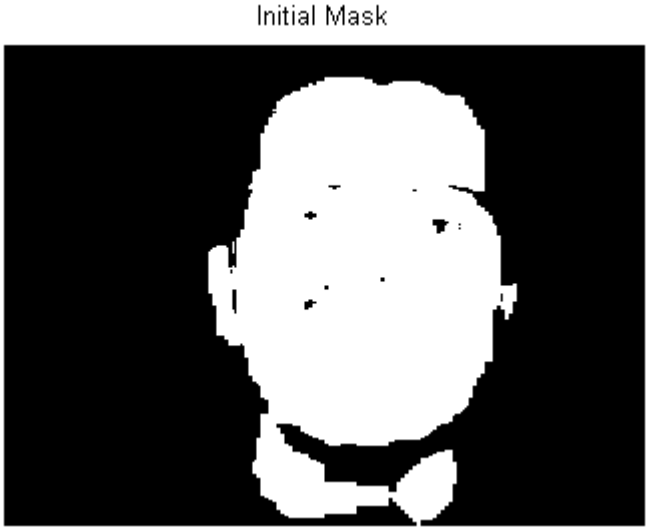


Figure 3.3: Initial Mask

### 3.3.2 Morphology

After the initial mask is created, an opening is performed to remove any pixels that are not connected to the face and to clean up the mask. After the opening, a dilation is performed to fill in any small holes and connect any pixels that are connected to the face but were not classified as part of the face from the segmentation. The face is assumed to be the largest component in the initial binary image and any smaller components are either parts of the face that were not segmented or are parts of the background that was incorrectly classified. The dilation operation is used to combine the largest component, the face, and any components that are close to the face, which are assumed to be part of the face that was not correctly classified. This new component that has the largest area is labeled as the face and an erosion operation is used to remove the spreading of the boundaries from the dilation. Figure 3.4 shows the result after the morphological filtering is performed and Figure 3.5 shows the result of the final mask minus the initial mask.

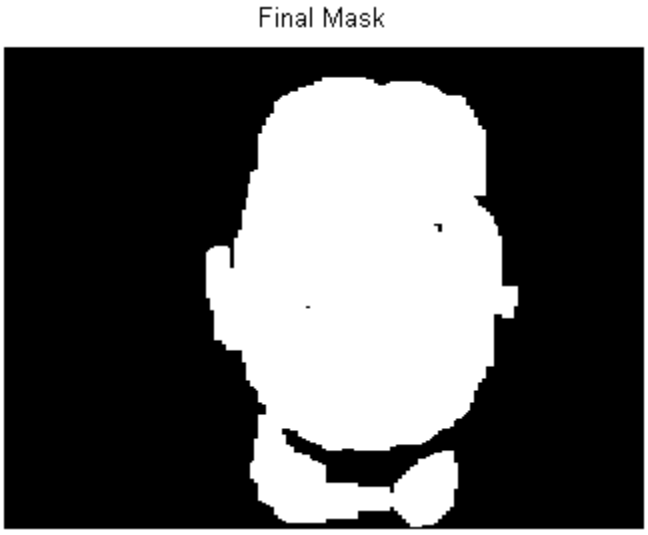


Figure 3.4: Final Mask

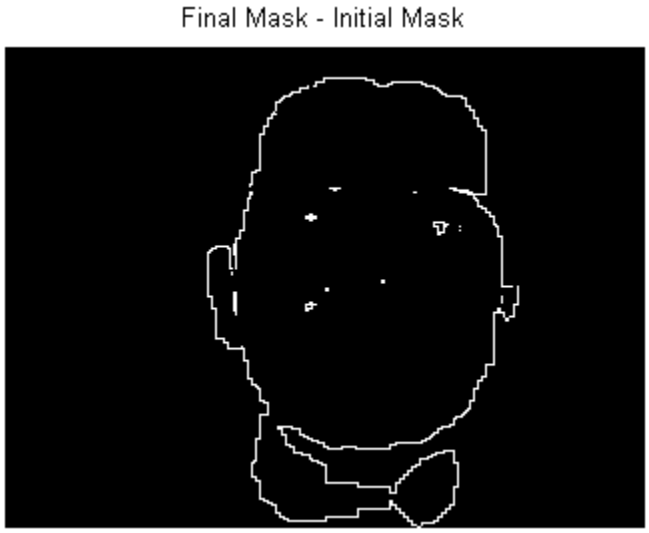


Figure 3.5: Final Mask minus the Initial Mask

After the final mask is created, the mask is then ANDed to the image to keep the face. The advantage of using the YCbCr color space is that a conversion to grayscale is simply the Y- component of the color space.

### 3.3.3 Cropping the Image

The resulting image is then cropped to the size of the mask. The number of columns of the new image is determined by the difference between the right most pixel and the left most pixel and the

number of rows is determined by the difference between the bottom most pixel and the top most pixel. After the cropped image is created, the image is then resized to 56x46 pixels. This size will then be the standard size that all frames and their respective cropped images will be further processed on as shown in Figure 3.6.

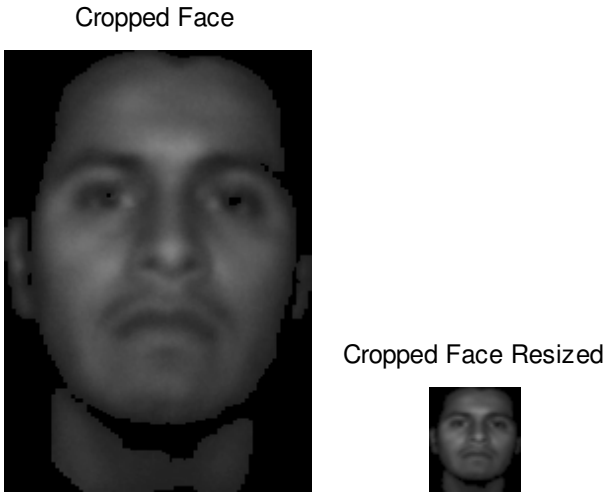


Figure 3.6: Resulting Cropped Face and Resized Cropped Face (Artificially Brightened)

## 3.4 TRAINING THE SYSTEM

### 3.4.1 Creating the Training Set

For face recognition to work, the system must have a sample of all the images that are going to be tested. In this case of all the fames being tested, only 10 or 11 total samples from each class or person will be used as the training set. The total number of frames for that particular class is calculated and that result is divided by 10. The frames that are used as the training set are the first frame skipping the value that was calculated until the next frame until the end of the images.

```
[row col]=size(francisco_info);
 skip=round(row/10);
 k=1;
 for i=1:skip:row
     training_set{k,2}=francisco_info{i,1};
     training_set{k,1}=1;
     skipped_images_francisco(k)=i;
     k=k+1;
 end
```

This process is then repeated for all three classes and combined to create the entire training set, and the rest of the images are regarded as the testing images.

### 3.4.2 Creating the W Matrix

After the training images are compiled some preprocessing is needed for the face recognition algorithm to work correctly. The W matrix or the projection matrix needs to be calculated in order for the images to be transformed into the face space. After the W matrix is calculated the Eigen reduction can begin. To reduce the matrix, the singular value decomposition was taken reducing the U matrix from $2576 \times 2576$ to a $2576 \times (N - c)$ matrix.

```
[u s v]=svd(Sw);
W(:,1:face-max_class)=u(:,1:face-max_class);
```

The singular value decomposition was used because it is more accurate than the Eigen decomposition in MATLAB. The U matrix forms a set of orthonormal output basis vectors for the W matrix. This reduced matrix then becomes the projection matrix used to transform the images from the pixel space to the face space.

## 3.5 TESTING THE SYSTEM

### 3.5.1 Applying Face Recognition to the Group of Images

The face recognition algorithm will take the images in the testing set and check which class, or person, each image belongs to. Some initialization is required before actually performing face recognition. First, the total number of training faces must be known as well as the total number of classes in the training set. Second, the total number of faces in each class of the testing set must be known so all the faces can be properly identified. Third, the training faces are then transformed into the face space using the reduced projection matrix, W, and the average of each class is then calculated in the face space.

A nearest neighbor classifier will be used to classify which face belongs to which class. The classifier works by calculating the distance from each class by multiplying the transpose of the projection matrix multiplied by the current face after it is concatenated where that result is then subtracted from the sample mean of all the classes. The result will be in the form of a matrix which does not give a scalar distance, so the result will be transformed into a scalar by the `norm()` function in MATLAB. The `norm()` of a matrix is a scalar that gives some measure of the magnitude of the elements of the matrix. In this case the largest singular value of the singular value decomposition will be the distance.

```
for ii=1:(max_class)
distanze_pesi(ii)=norm(double(W')*double(face_test{xx,1}(:))-distances(:,ii));
end
```

After the three distances have been calculated, the class that corresponds to the minimum distance will be the class that the face belongs to. The values that are saved will be the face that was tested, the class that was chosen, the smallest distance, and all the distances calculated. These values will be saved so some statistical analysis can be performed and conclusions can be based.

### 3.5.2 Applying Super-Resolution to the Group of Images

The super-resolution procedure is broken down into two steps. The first step is the registration step where the four input images will be properly aligned using the Keren registration method. The second step is the reconstruction step where the estimated values are used to align the images and the four input images are used to create an image that has twice the amount of rows and twice the amount of columns.

A subroutine `keren()` will be called to register, or align, the images to get correct information. The pyramid scheme is first constructed to be able to account for large motion differences between the images. After the pyramid is set up the $g(x, y)$ can be approximated along with the partial derivatives of f, $\frac{\partial f}{\partial x}$ and $\frac{\partial f}{\partial y}$. The first image in the series is treated as the reference image and the other three will be compared to that image. At each level in the pyramid scheme the translational shifts and rotational values are calculated until the pyramid is at the last level. These shifts and rotations are stored at each

level where the values are refined at the last level. The values kept from the registration stage include the rotational information and the translational information.

The reconstruction stage uses a subroutine called `robustSR()` which takes the output from the `keren`() subroutine and uses those along with the same four images from the registration stage, and the interpolation factor, which in this case is two, for its inputs. The `robustSR()` has a maximum of 50 iterations with a scale factor, or step size, of $\lambda = 0.05$. The algorithm then takes the registration stage information from each image and creates a temporary image that is then blurred with the matrix

$$\begin{bmatrix} 0 & 1 & 0 \\ 1 & 2 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

then rescaled back to the original size of $56 \times 46$, which is the decimation matrix, where the back-projection difference image is created. The back-projected difference image is then rescaled to the high resolution image size and a sharpening filter is applied with the elements

$$\begin{bmatrix} 0 & -0.25 & 0 \\ -0.25 & 2 & -0.25 \\ 0 & -0.25 & 0 \end{bmatrix}$$

The median is then taken over the new matrix element by element when all four images have undergone this process resulting in a single image. This process is then repeated in the next iteration using the iteration equation of

$$\vec{X}^{n+1} = \vec{X}^{n} + \lambda \nabla L\left(\vec{X}\right)$$

until all 50 iterations have been applied or the error is less than $0.0001$.

The resulting image with twice as many rows and columns, $112 \times 92$, as seen in Figure 3.7 will then be reduced to the standard size of $56 \times 46$ shown in Figure 3.8. This new image will still contain enough new information even though half of the information was discarded to help to improve the classification accuracy. This new image will then be sent to the face recognition subroutine with the procedure being the same as the individual faces.

The remaining images will then have a length four sliding window to determine the next four images. If images 1, 2, 3, and 4 from the training set are used for super-resolution, the next four images used are 2, 3, 4, and 5. This repeats until the end of the current class as shown in Figure 3.9. After the window has reached the end of the class, the window then starts the procedure again using the image

number for the remaining class. The values saved during the super-resolution section are the four images being tested, their respective rotational and translational information from the registration stage, the resulting image from the super-resolution stage, the class that the image was closest to, the minimum distance that was calculated, and all of the distances calculated.

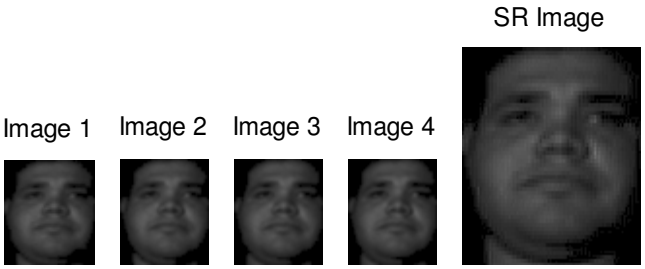


Figure 3.7: Four Input Images and Resulting Super-Resolution Image (Artificially Brightened)

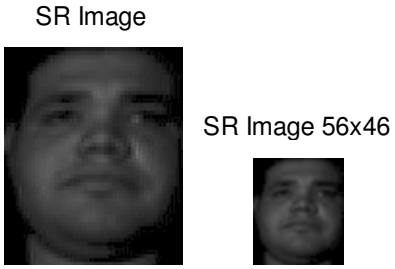


Figure 3.8: Original Super-Resolution Image and Resized Super-Resolution Image (Artificially Brightened)

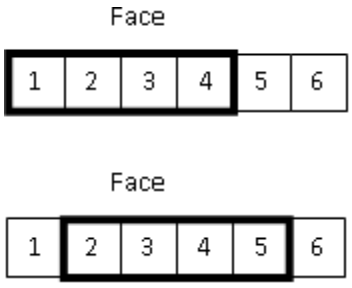


Figure 3.9: Sliding Window

### 3.5.3 Calculating the Average in Pixel Space for the Group of Images

Another method tested in this paper to improve the accuracy of face recognition is to take the average of the same images that were used in the super-resolution algorithm. The average face will be in the pixel domain meaning no transformations or any additional processing will be utilized. The group of four images will be used resulting in an image that is the same size as the four input images. The averaging process takes four consecutive images from the training set and takes the average of all the images as follows

$$\bar{x} = \frac{1}{N}\sum_{i=1}^{N} x_i$$

The resulting image is then the average face as shown in Figure 3.10. The rest of the images are then processed the same way using the same sliding window that was used in the super-resolution procedure. The resulting averaged images are then put through the face recognition algorithm where they are to be classified. The results that are kept for comparison are the four input images, the averages output image, the class that was closest, the minimum distance that was calculated, and all the distances that were calculated.

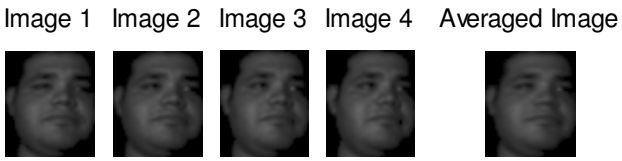


Figure 3.10: The Four Faces and the Resulting Average (Artificially Brightened)

### 3.5.4 Calculating the Minimum of Four Faces to the Group of Images

Another method tested for possible improvement to super-resolution that does not involve and addition processing is to take four input face images and calculate the distance across each face. The sliding window is applied to the same face images during the super-resolution and average procedures. The face recognition stage is performed differently here than in any other procedure. Each face image has its distance calculated from each class and the minimum distance is saved. This is repeated three more times for the remaining faces in the sliding window. After the four faces have been identified, the minimum of the minimum distances is kept and used as the actual minimum distance. Great care has to

be taken with this procedure and almost all of the data must be kept to ensure that the correct class is chosen. If one image is classified incorrectly and has the minimum distance, the result will affect those images and possible future images until the sliding window has completely passed that image. This error has the capability of multiplying each mistake fourfold resulting in up to four times as many total errors. The values that are saved are the four input images, the four classes that were closest, the four minimum distances, and all of the calculated distances for each of the four images.

### 3.5.5 Calculating the Centroid Shifted Average for the Group of Four Faces

The centroid shifted average is another possible inexpensive SR approach which is an alternative to compare with full super-resolution to try to improve the accuracy of face recognition. The centroid shifted average is similar to the average except that the images are shifted according to a computed reference centroid. The first image in the group of four faces is considered the reference image and the centroids of the other three are found and saved. The horizontal and vertical shifts from the first image centroid are calculated creating a difference matrix for the centroids where the first column is the horizontal difference and the second column is the vertical difference and each row corresponds to the individual images. That difference matrix is used to shift the images to have the centroids aligned when the average is performed on the images. If the centroid shifts result in the face exceeding the bounds of the image, those values are ignored in the averaging calculation. Only the values that stay in the image bounds are kept and averaged. Figure 3.11 shows the four input faces and the resulting centroid shifted average. The resulting face is then saved and the process is repeated with the sliding window moving on face ahead and repeating the process with the new set of four faces. The values that are stored are the four faces used, the class that was closest, the minimum distance that was calculated, and all the distances that were calculated.

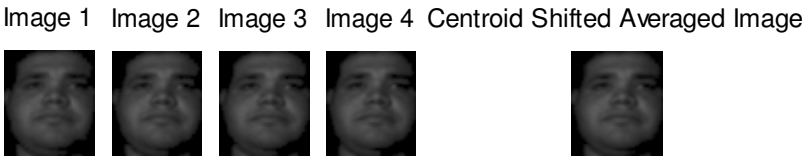


Figure 3.11: The Four Faces and the Resulting Centroid Shifted Average (Artificially Brightened)

### 3.5.6 Manual Cropping

The face images that are automatically cropped will then be tested against the same images, but this time the face images will be manually cropped. This manual cropping will try to get as much of the head as possible while removing as much of the background near the face. This cropping is under the assumption that keeping just the face and a minimum amount of the background, the algorithm will improve because of the large amount of face pixels in the image.

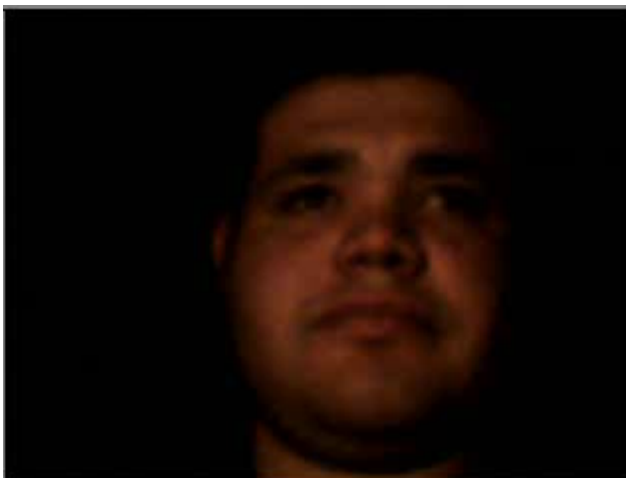

Figure 3.12: Original Image

Figure 3.13: Cropped Image

Figure 3.12 shows the original size of the image while Figure 3.13 shows the resulting image after it has been cropped. The cropped images are then resized to the standard $56 \times 46$ dimensions and converted

into grayscale images using the `rgb2ycbcr()` command. After the conversion only the luminance portion of the new image is kept, resulting in the wanted grayscale face image. After the images are converted to grayscale, the images are then trained the same way as the automatically cropped images, having the same testing set and training set images. The only difference between the two training sets and testing sets will be the way that the grayscale image was cropped. The testing set will then have the same procedure as the automatically cropped images: face recognition will be applied to the face images, super-resolution will be applied with the length 4 sliding window using the Keren reconstruction method and the Robust Super-Resolution reconstruction method, the average of the four faces in the pixel space with the same length 4 sliding window, and the minimum distance of the four images in the length 4 sliding window. The centroid shifted average will not be performed because the skin segmentation algorithm used to create the mask of the face for the centroid. The manual cropping has no face mask, and a centroid value will not correspond to the face but somewhere in the cropped image, which may or may not be the face.

## 3.6 Approaches to Evaluate Improvements

### 3.6.1 Distance from Training Set Means

The first approach to evaluate any improvements is the minimum distance that is calculated from the projected training set images. For each group of faces the methods of super-resolution, average, minimum of the four images, and centroid shifted average will be compared and whichever method consistently has the lowest distance from all of the grouped faces will be considered the most reliable method. The only major problem with the distance measurement is the fact that an incorrect classification will provide a false minimum distance for that method or in the worst case scenario all the distances for all the methods will be incorrect.

### 3.6.2 Distance from Mean Centroid

An approach to evaluate the centroid shifted average is to compare the distances from each image centroid in each class to the average centroid for each individual class. This is the distance that is calculated by taking the average centroid value from all the images in a single class and calculates the resulting distance from the individual centroid values, and this process is repeated for all three classes.

The mean and variance are then calculated for the distances for each class to measure how spread out the distances are from the found average distance. The larger the variance, the more spread out the distances are from the average distance, and the smaller the variance, the less spread out or closer to the average distance.

### 3.6.3 Centroid Statistics for All Algorithms

The second approach to evaluate the centroid shifted average is to calculate the mean and variance for the row and column for each centroid for each class. These statistics will give an idea as to how spread apart the individual centroids are in addition to the distance measures of the centroids. A small variance in either the row or column for each class indicates the centroids for the faces are close to the average centroid value, while a large variance in either the row or column for each class indicates the centroids are farther apart from the average centroid value.

### 3.6.4 Motion Vector Analysis for Super-Resolution and Centroid Shifted Average for Resized Images

The motion vectors will be calculated to compare the registrations between the centroid shifted average and super-resolution to determine why a certain algorithm performed better than the other. The motion vectors will show the vertical and horizontal shifts from the second to fourth images in the sliding window. The first image will be ignored because of it being the reference image and will always have motion vectors being zero. Besides the motion vectors being calculated, the magnitude of the motion vectors will be calculated to provide a single value or distance and is another distance to determine which registration algorithm did not shift the images enough or shift the images too much. The "correct" motion vectors will be the ones that produced the smallest distance to the mean. There is no ground truth for determining exactly how much the image was shifted, but it is assumed that the motion vectors that correspond to the smallest distance are close to the correct shift, and the motion vectors that are farther away will have a larger distance to the mean.

## 3.7 Evaluation of Super-Resolution and Centroid Shifted Average on Full Size Faces

The centroid shifted average and super-resolution will be tested on the full sized cropped images. This test should determine whether the centroid shifted average or super-resolution performs better

using larger images. These are the faces that were kept from the skin segmentation algorithm before the resizing of $56x46$. The images will be padded with zeros until the final image size of $240x300$ is reached because the super-resolution registration algorithm needs images with the same size to work. The size of $240x300$ is chosen because that is the largest cropped image has those dimensions. Those larger images will then go through the super-resolution and centroid shifted average algorithms with the resulting image resized to the standard $56x46$ size and those resized images will be inputted through the face recognition algorithm. A memory node with 64GB of RAM was used for this section to speed up the amount of time for the super-resolution algorithm to register the $240x300$ images and reconstruct the $480x600$ resulting image.

# Chapter 4: Analysis of Results

Throughout the experiments, the four algorithms super-resolution, the average, the centroid shifted average, and the minimum of four faces were compared against each other to determine which algorithm performed better across all three classes. Centroid statistics and motion vectors from the centroid shifted average were compared to determine why one algorithm performed better than another. A manual cropping was also tested using the same three classes with the same video to compare the automatic cropping algorithm to cropping done by hand. A comparison between the centroid shifted average and super-resolution was compared using the full sized images from the skin segmentation without the resizing was performed. This comparison was done to determine if the distances decrease and which algorithm performs better with more information.

## 4.1 AUTOMATIC CROPPING

### 4.1.1 Distance from Training Set Means

The evaluation of how well an algorithm performed will be determined by the distance for each group of four faces. A similar measure of how well the algorithm performed will be the accuracy rate for each individual class. The accuracy will give an indication of how well the algorithm performs with a large number of faces by analyzing if the smallest distance corresponded to the correct class. If too many faces are incorrectly classified by a certain algorithm, it is more likely that the algorithm did not perform as well as it could have.

Table 4.1 shows the results for face recognition by itself and Table 4.2 shows the confusion matrix for face recognition. Class 1, or Francisco, has all 200 faces correctly classified, while class 2, or Gabriel, has 284 faces correctly classified and 17 incorrectly classified as class 1, Francisco, and class 3, or Ricardo, has 154 faces correctly classified and 4 incorrectly classified as class 1, Francisco. The accuracy rates for the three classes, 100%, 94.35%, and 97.47% are at acceptable levels of over 90% as well as the statistics of the classes.

Table 4.1: Statistics for Face Recognition

| Class | | Accuracy (%) | Mean | Variance |
|---|---|---|---|---|
| 1 | 200/200 | 100 | 596.7527 | 11499.52 |
| 2 | 284/301 | 94.35216 | 1052.385 | 16414.45 |
| 3 | 154/158 | 97.46835 | 749.3441 | 13150.46 |

Table 4.2: Confusion Matrix for Face Recognition

| | Class 1 | Class 2 | Class 3 |
|---|---|---|---|
| Class 1 | 200 | 0 | 0 |
| Class 2 | 17 | 284 | 0 |
| Class 3 | 4 | 0 | 154 |

Table 4.3 shows the results for face recognition combined with Super-Resolution and Table 4.4 shows the confusion matrix for the combination of Super-Resolution. Class 1, or Francisco, has all 197 faces correctly classified, while class 2, or Gabriel, has 274 faces correctly classified and 24 incorrectly classified as class 1, Francisco, and class 3, or Ricardo, has 153 faces correctly classified and 2 incorrectly classified as class 1, Francisco. The accuracy rates for the three classes, 100%, 91.95%, and 98.71% are at acceptable levels of over 90% as well as the statistics of the classes.

Table 4.3: Statistics for Face Recognition with Super-Resolution

| Class | | Accuracy (%) | Mean | Variance |
|---|---|---|---|---|
| 1 | 197/197 | 100 | 575.3298 | 12035.59 |
| 2 | 274/298 | 91.94631 | 1046.559 | 18098.74 |
| 3 | 153/155 | 98.70968 | 733.7816 | 13831.6 |

Table 4.4: Confusion Matrix for Super-Resolution

| | Class 1 | Class 2 | Class 3 |
|---|---|---|---|
| Class 1 | 197 | 0 | 0 |
| Class 2 | 24 | 274 | 0 |
| Class 3 | 2 | 0 | 153 |

Table 4.5 shows the results for face recognition combined with the average face and Table 4.6 shows the confusion matrix for the combination of the average face. Class 1, or Francisco, has all 197 faces correctly classified, while class 2, or Gabriel, has 283 faces correctly classified and 15 incorrectly classified as class 1, Francisco, and class 3, or Ricardo, has all 155 faces correctly classified. The accuracy rates for the three classes, 100%, 94.97%, and 100% are at acceptable levels of over 90% as well as the statistics of the classes.

Table 4.5: Statistics for Face Recognition with Average

| Class | | Accuracy (%) | Mean | Variance |
|---|---|---|---|---|
| 1 | 197/197 | 100 | 552.8521 | 13921.07 |
| 2 | 283/298 | 94.96644 | 1029.339 | 21648.09 |
| 3 | 155/155 | 100 | 705.9207 | 14653.3 |

Table 4.6: Confusion Matrix for Average

| | Class 1 | Class 2 | Class 3 |
|---|---|---|---|
| Class 1 | 197 | 0 | 0 |
| Class 2 | 15 | 283 | 0 |
| Class 3 | 0 | 0 | 155 |

Table 4.7 shows the results for face recognition combined with the minimum of four faces and Table 4.8 shows the confusion matrix for the combination of the minimum of four faces. Class 1, or Francisco, has all 197 faces correctly classified, while class 2, or Gabriel, has 276 faces correctly classified and 22 incorrectly classified as class 1, Francisco, and class 3, or Ricardo, has 146 faces correctly classified and 9 incorrectly classified as class 1, Francisco. The accuracy rates for the three classes, 100%, 92.62%, and 94.19% are at acceptable levels of over 90% as well as the statistics of the classes.

Table 4.7: Statistics for Face Recognition with Minimum of Four Faces

| Class | | Accuracy (%) | Mean | Variance |
|---|---|---|---|---|
| 1 | 197/197 | 100 | 554.0585 | 10426.33 |
| 2 | 276/298 | 92.61745 | 1023.093 | 18327.76 |
| 3 | 146/155 | 94.19355 | 694.5252 | 10589.02 |

Table 4.8: Confusion Matrix for Minimum of Four Faces

| | Class 1 | Class 2 | Class 3 |
|---|---|---|---|
| Class 1 | 197 | 0 | 0 |
| Class 2 | 22 | 276 | 0 |
| Class 3 | 9 | 0 | 146 |

Table 4.9 shows the results for face recognition combined with the centroid shifted average and Table 4.10 shows the confusion matrix for the combination of the centroid shifted average. Class 1, or Francisco, has all 197 faces correctly classified, while class 2, or Gabriel, has 284 faces correctly classified and 14 incorrectly classified as class 1, Francisco, and class 3, or Ricardo, has 151 faces correctly classified and 4 incorrectly classified as class 1, Francisco. The accuracy rates for the three classes, 100%, 95.93%, and 97.42% are at acceptable levels of over 90% as well as the statistics of the classes.

Table 4.9: Statistics for Face Recognition with Centroid Shifted Averaging

| Class | | Accuracy (%) | Mean | Variance |
|---|---|---|---|---|
| 1 | 197/197 | 100 | 536.5645 | 13842.50 |
| 2 | 284/298 | 95.93020 | 1006.720 | 25309.93 |
| 3 | 151/155 | 97.41936 | 690.2111 | 15187.03 |

Table 4.10: Confusion Matrix for Centroid Shifted Averaging

| | Class 1 | Class 2 | Class 3 |
|---|---|---|---|
| Class 1 | 197 | 0 | 0 |
| Class 2 | 14 | 284 | 0 |
| Class 3 | 4 | 0 | 151 |

A comparison of all the different methods were compared against each other and the individual faces. Figures 4.1 and 4.2 show the results of Class 1, Francisco. Out of the 197 faces that are combined through the sliding window, super-resolution had 28 combinations that has the smallest distance, the average had 47 combinations, the centroid shifted average had 104 combinations, and the minimum of the four faces had 18 combinations.

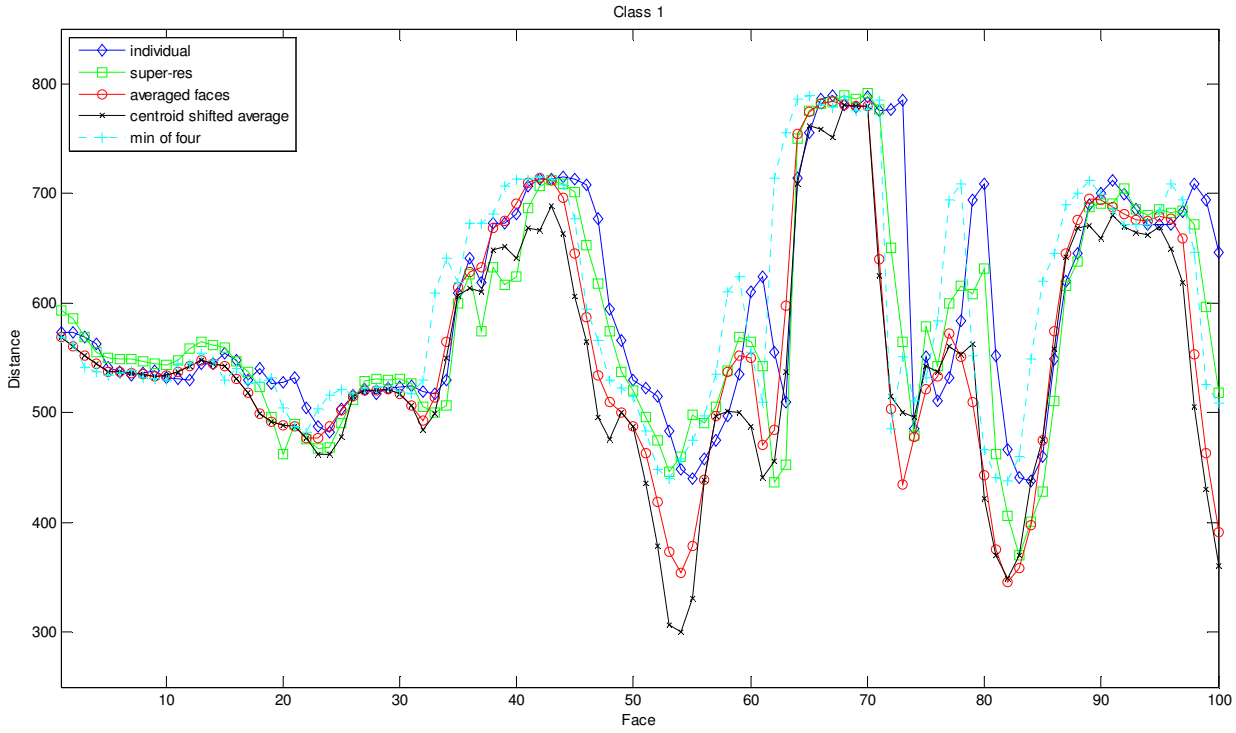


Figure 4.1: Plot of Distances for Class 1

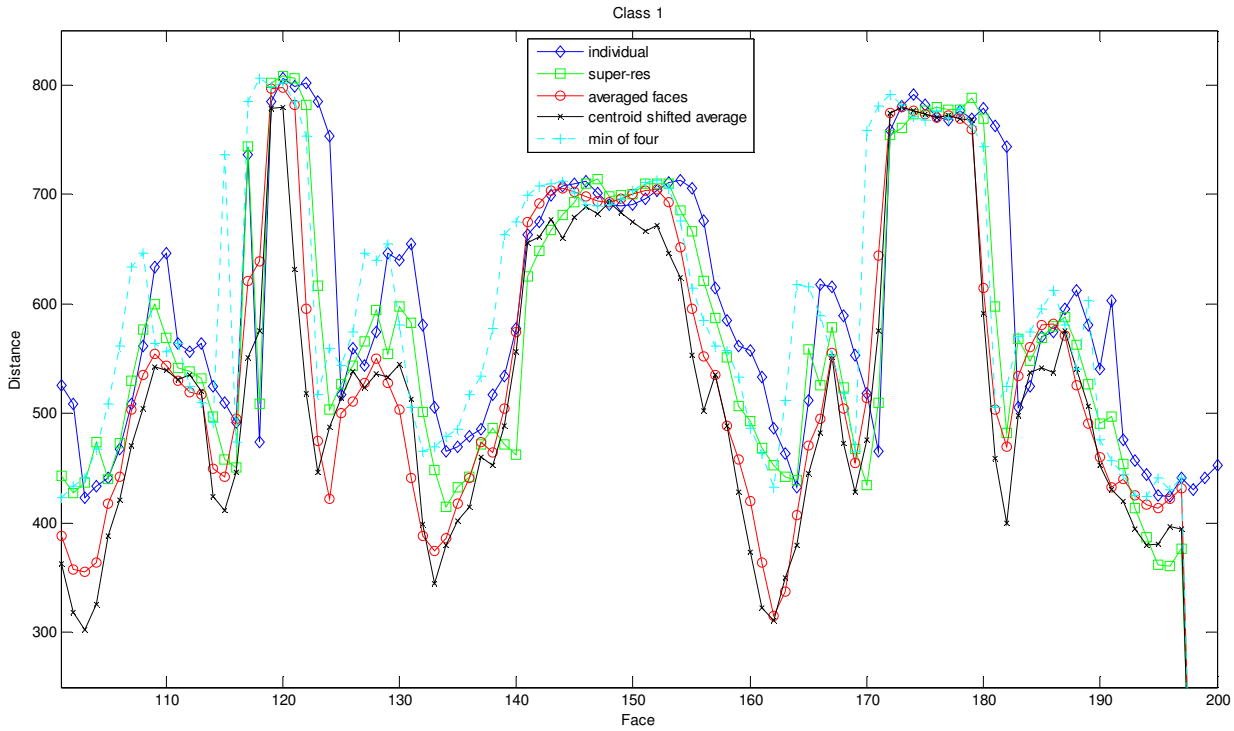


Figure 4.2: Plot of Distances for Class 1

Figures 4.3 and 4.4 show the results of Class 2, Gabriel. Out of the 298 faces that are combined through the sliding window, super-resolution had 53 combinations that has the smallest distance, the average had 62 combinations, the centroid shifted average had 136 combinations, and the minimum of the four faces had 47 combinations.

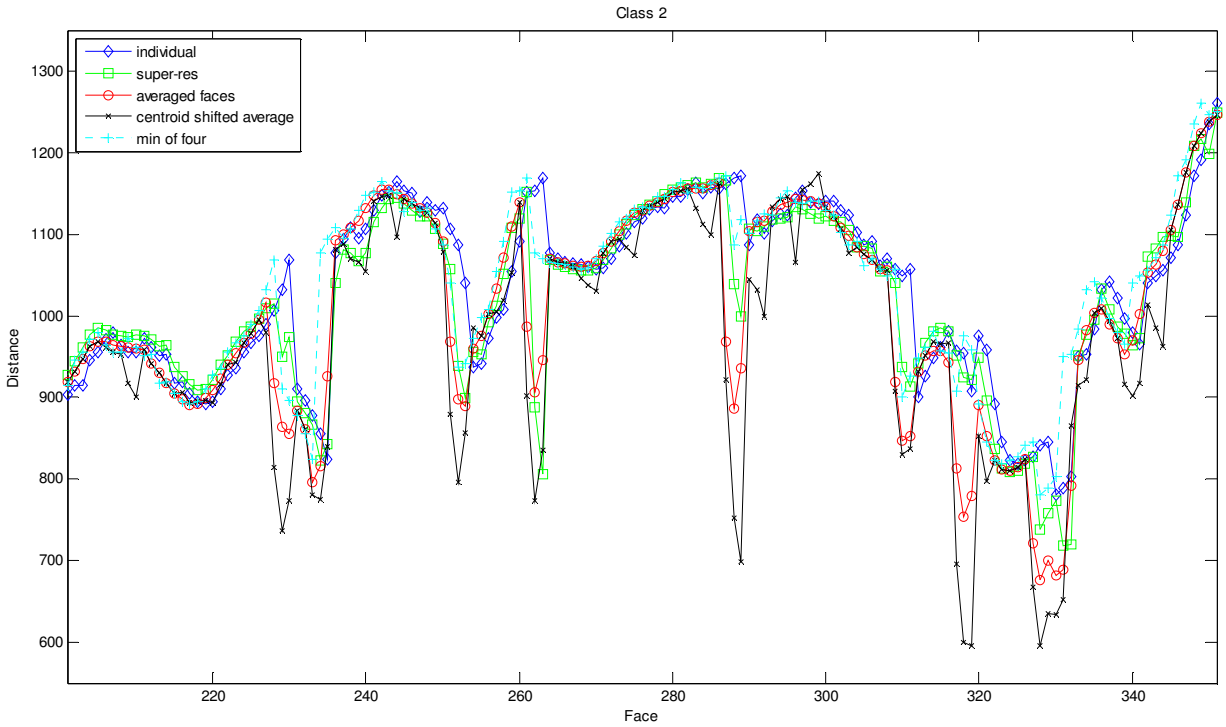


Figure 4.3: Plot of Distances for Class 2

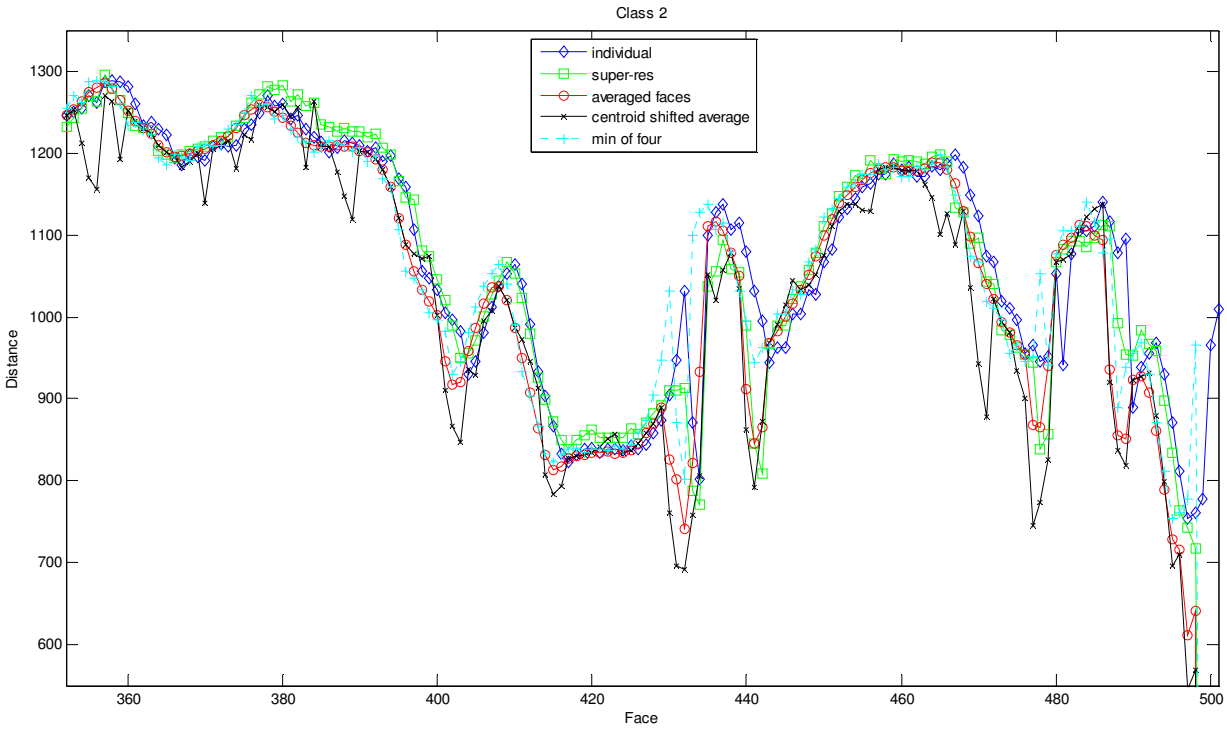


Figure 4.4: Plot of Distances for Class 2

A comparison of all the different methods were compared against each other and the individual faces. Figures 4.5 and 4.6 show the results of Class 3, Ricardo. Out of the 155 faces that are combined through the sliding window, super-resolution had 31 combinations that has the smallest distance, the average had 26 combinations, the centroid shifted average had 80 combinations, and the minimum of the four faces had 18 combinations.

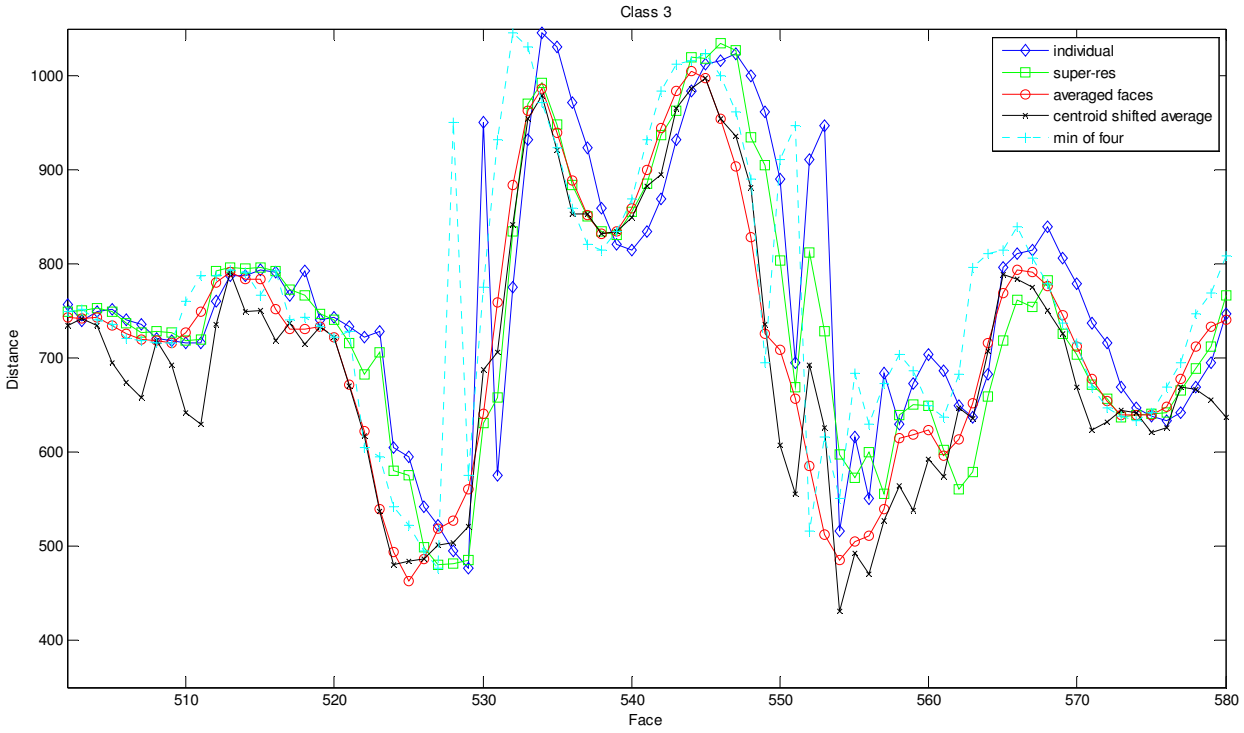


Figure 4.5: Plot of Distances for Class 3

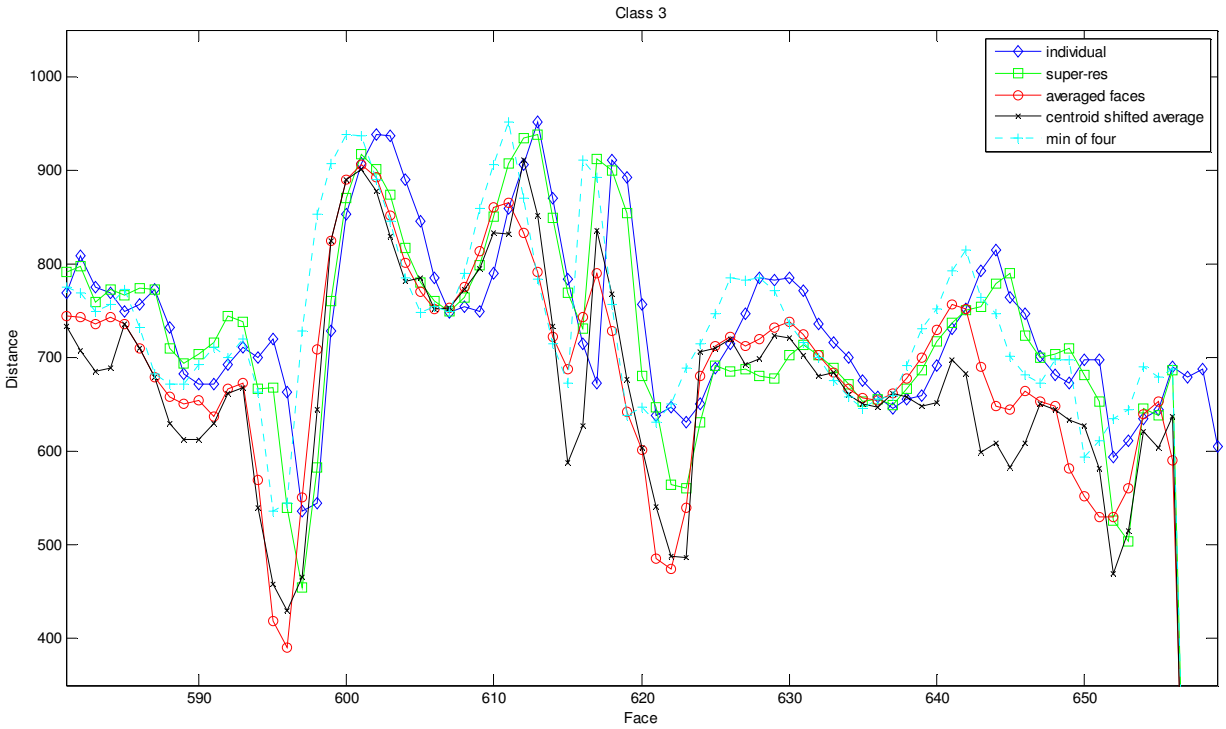


Figure 4.6: Plot of Distances for Class 3

Table 4.11 shows the overall results for the automatic cropping algorithm. The super-resolution algorithm had 112 out of the possible 650 combinations, 12.23%, as the algorithm with the smallest distance. The average had 135 out of 650 combinations, 20.77%, as the algorithm with the smallest distance. The centroid shifted average had the most, 320 out of 650, combinations or 49.23% as the algorithm with the smallest distance. The minimum of four faces had the least, 83 out of 650, combinations or 12.77% as the algorithm with the smallest distance.

Table 4.11: Overall Results for Automatic Cropping

| | Smallest Distance | Percent |
|---|---|---|
| Super-Resolution | 112 | 17.23077 |
| Average | 135 | 20.76923 |
| Centroid Shifted Average | 320 | 49.23077 |
| Minimum of Four Faces | 83 | 12.76923 |

A check was done on the testing set to determine how well these faces did when tested. All of the faces had an accuracy of 100%. Table 4.12 shows the statistics of the self test which has a higher mean than the individual methods in addition to a smaller variance. Table 4.13 shows the confusion matrix which has all the images classified as the correct class.

Table 4.12: Statistics for the Training Set

| Class | | Accuracy (%) | Mean | Variance |
|---|---|---|---|---|
| 1 | 10/10. | 100 | 665.6766 | 8208.221 |
| 2 | 11/11. | 100 | 1089.257 | 12570.78 |
| 3 | 10/10. | 100 | 870.9981 | 10427.28 |

Table 4.13: Confusion Matrix for Training Set

| | Class 1 | Class 2 | Class 3 |
|---|---|---|---|
| Class 1 | 10 | 0 | 0 |
| Class 2 | 0 | 11 | 0 |
| Class 3 | 0 | 0 | 10 |

### 4.1.2 Distance from Mean Centroid and Centroid Statistics

Another measure of how well the centroid shifted average performed would be to compare the distance between the centroids of the individual faces to that of the mean centroid of that class. Figure 4.7 shows the centroid distances of Class 1, Francisco, with the mean centroid distance being 1.26 with a variance of 0.45. These distances show a pattern that is similar, but does not exactly correspond with the rotation of the head. The distances for Class 1 are small and show that there is not a large amount of motion between the centroids with the largest distance being less than 3 and the smallest distance being around 0.25.

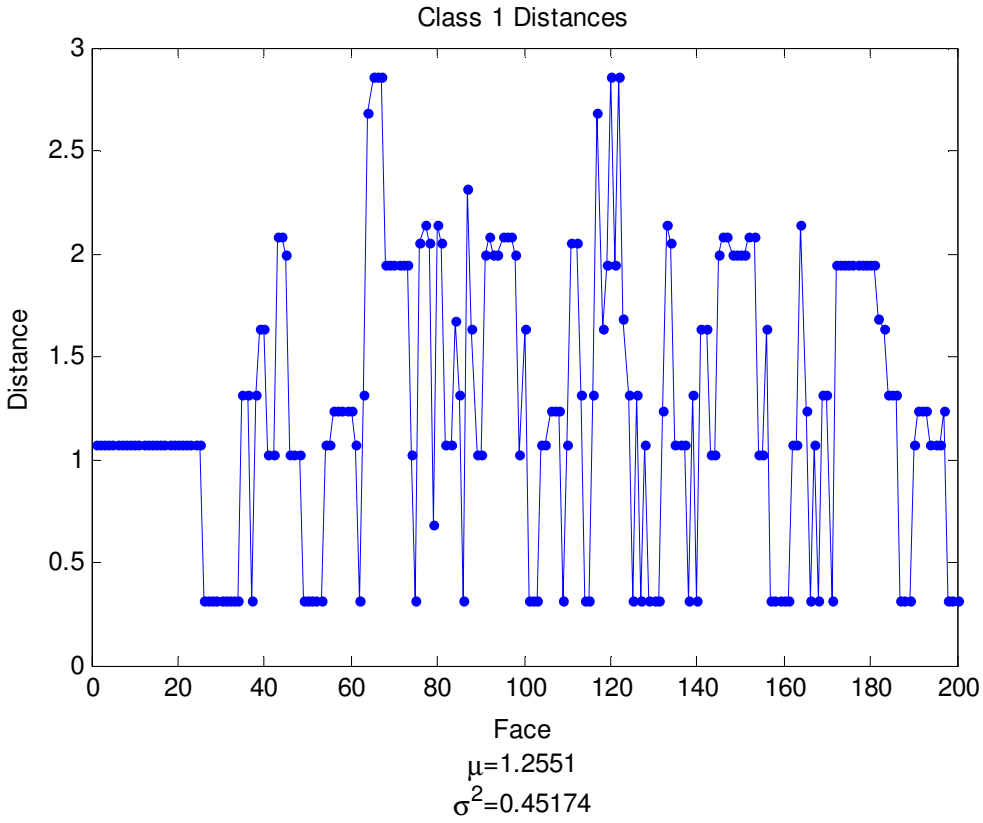


Figure 4.7: Distance Between Face Centroid from Mean Centroid of Class 1 and its Statistics

Figure 4.8 shows the centroid distances of Class 2, Gabriel, with the mean centroid distance being 2.48 with a variance of 1.56. These distances show a pattern that is similar, but does not exactly correspond with the rotation of the head. The distances for Class 2 are larger than for Class 1 and show that there some motion between the centroids with the largest distance being over 6 and the smallest distance being around 0.25.

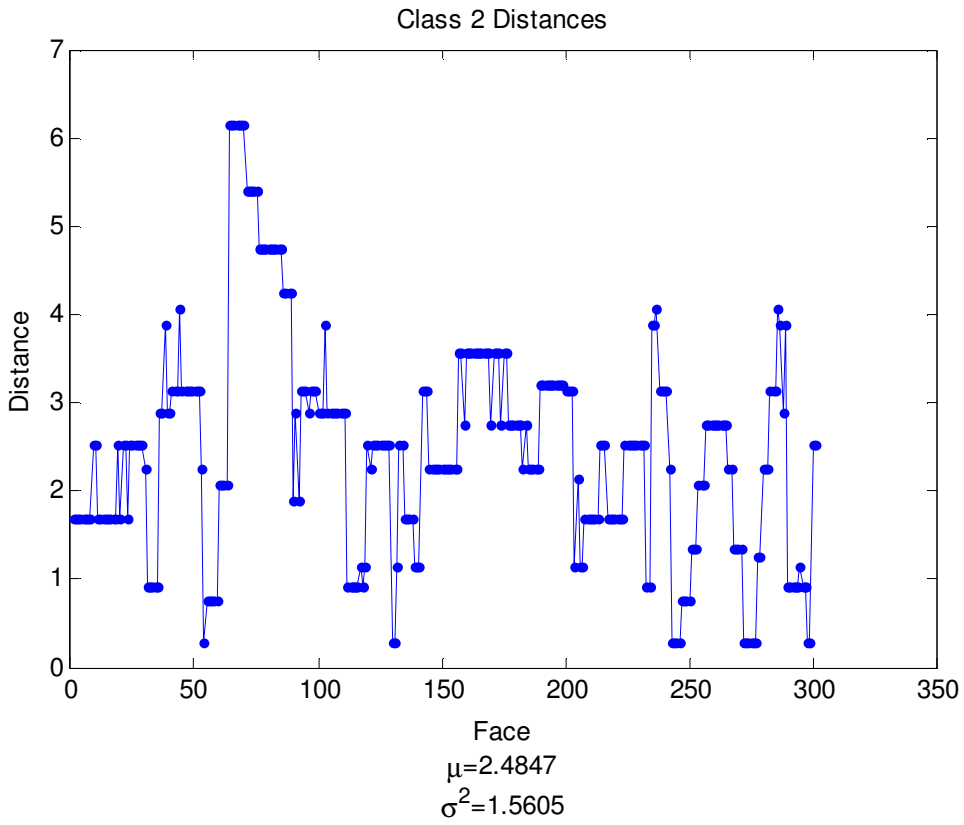


Figure 4.8: Distance Between Face Centroid from Mean Centroid of Class 2 and its Statistics

Figure 4.9 shows the centroid distances of Class 3, Ricardo, with the mean centroid distance being 1.66 with a variance of 0.32. These distances show a pattern that is similar, but does not exactly correspond with the rotation of the head. The distances for Class 3 are small and show that there is not a large amount of motion between the centroids with the largest distance being less than 3 and the smallest distance being less than 0.5.

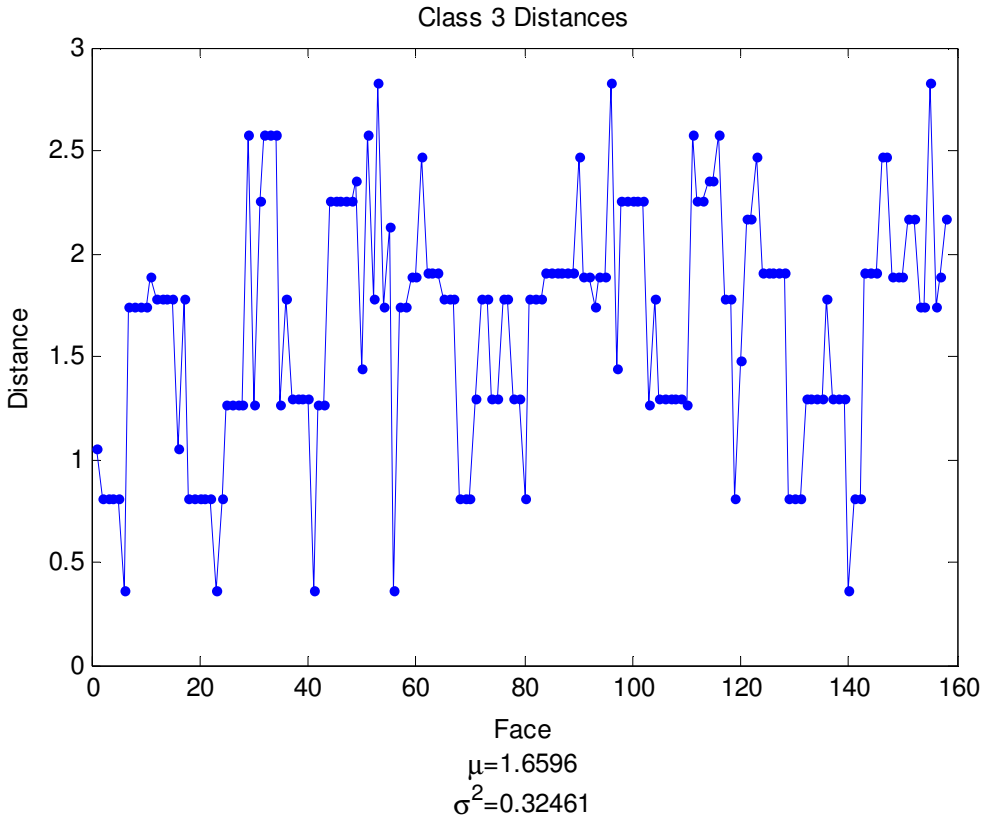


Figure 4.9: Distance Between Face Centroid from Mean Centroid of Class 3 and its Statistics

## 4.2 MANUAL CROPPING

The alternative method to the automatic cropping was to test the same images done with the automatic cropping by manually cropping the images. Table 4.14 shows the results for face recognition by itself and Table 4.15 shows the confusion matrix for face recognition. Class 1, or Francisco, has all 200 faces correctly classified, while class 2, or Gabriel, has 286 faces correctly classified and 15 incorrectly classified as class 1, Francisco, and class 3, or Ricardo, has 158 faces correctly. The accuracy rates for the three classes, 100%, 95.02%, and 100% are at acceptable levels of over 90% as well as the statistics of the classes.

Table 4.14: Statistics for Face Recognition

| Class | | Accuracy (%) | Mean | Variance |
|---|---|---|---|---|
| 1 | 200/200 | 100 | 438.918 | 3093.088 |
| 2 | 286/301 | 95.01661 | 851.0562 | 20804.84 |
| 3 | 158/158 | 100 | 670.1112 | 8211.129 |

Table 4.15: Confusion Matrix for Face Recognition

| | Class 1 | Class 2 | Class 3 |
|---|---|---|---|
| Class 1 | 200 | 0 | 0 |
| Class 2 | 0 | 286 | 15 |
| Class 3 | 0 | 0 | 158 |

Table 4.16 shows the results for face recognition combined with Super-Resolution and Table 4.17 shows the confusion matrix for the combination of Super-Resolution. Class 1, or Francisco, has all 197 faces correctly classified, while class 2, or Gabriel, has 282 faces correctly classified and 16 incorrectly classified as class 3, Francisco, and class 3, or Ricardo, has 155 faces correctly classified. The accuracy rates for the three classes, 100%, 94.63%, and 100% are at acceptable levels of over 90% as well as the statistics of the classes.

Table 4.16: Statistics for Face Recognition with Super-Resolution

| Class | | Accuracy (%) | Mean | Variance |
|---|---|---|---|---|
| 1 | 197/197 | 100 | 440.789 | 3905.734 |
| 2 | 282/298 | 94.63087 | 853.3526 | 22390.01 |
| 3 | 155/155 | 100 | 668.146 | 7519.951 |

Table 4.17: Confusion Matrix for Face Recognition with Super-Resolution

| | Class 1 | Class 2 | Class 3 |
|---|---|---|---|
| Class 1 | 197 | 0 | 0 |
| Class 2 | 0 | 282 | 16 |
| Class 3 | 0 | 0 | 155 |

Table 4.18 shows the results for face recognition combined with the average face and Table 4.19 shows the confusion matrix for the combination of the average face. Class 1, or Francisco, has all 197 faces correctly classified, while class 2, or Gabriel, has 283 faces correctly classified and 15 incorrectly classified as class 3, Ricardo, and class 3, or Ricardo, has all 155 faces correctly classified. The accuracy rates for the three classes, 100%, 94.97%, and 100% are at acceptable levels of over 90% as well as the statistics of the classes.

Table 4.18: Statistics for Face Recognition with Average

| Class | | Accuracy (%) | Mean | Variance |
|---|---|---|---|---|
| 1 | 197/197 | 100 | 409.9608 | 4868.924 |
| 2 | 283/298 | 94.96644 | 835.0851 | 21544.78 |
| 3 | 155/155 | 100 | 637.966 | 10367.68 |

Table 4.19: Confusion Matrix for Face Recognition with Average

| | Class 1 | Class 2 | Class 3 |
|---|---|---|---|
| Class 1 | 197 | 0 | 0 |
| Class 2 | 0 | 283 | 15 |
| Class 3 | 0 | 0 | 155 |

Table 4.20 shows the results for face recognition combined with the minimum of four faces and Table 4.21 shows the confusion matrix for the combination of the minimum of four faces. Class 1, or Francisco, has all 197 faces correctly classified, while class 2, or Gabriel, has 286 faces correctly classified and 12 incorrectly classified as class 3, Ricardo, and class 3, or Ricardo, has all 155 faces correctly classified. The accuracy rates for the three classes, 100%, 95.97%, and 100% are at acceptable levels of over 90% as well as the statistics of the classes.

Table 4.20: Statistics for Face Recognition with Minimum of Four Faces

| Class | | Accuracy (%) | Mean | Variance |
|---|---|---|---|---|
| 1 | 197/197 | 100 | 413.5058 | 2817.894 |
| 2 | 286/298 | 95.97315 | 815.8965 | 19947.38 |
| 3 | 155/155 | 100 | 629.2864 | 7484.16 |

Table 4.21: Confusion Matrix for Minimum of Four Faces

| | Class 1 | Class 2 | Class 3 |
|---|---|---|---|
| Class 1 | 197 | 0 | 0 |
| Class 2 | 0 | 286 | 12 |
| Class 3 | 0 | 0 | 155 |

A comparison of all the different methods were compared against each other and the individual faces. Figures 4.10 and 4.11 show the results of Class 1, Francisco. Out of the 197 faces that are combined through the sliding window, super-resolution had 52 combinations that had the smallest distance, average has 111 combinations, and the minimum of the four faces has 34 combinations.

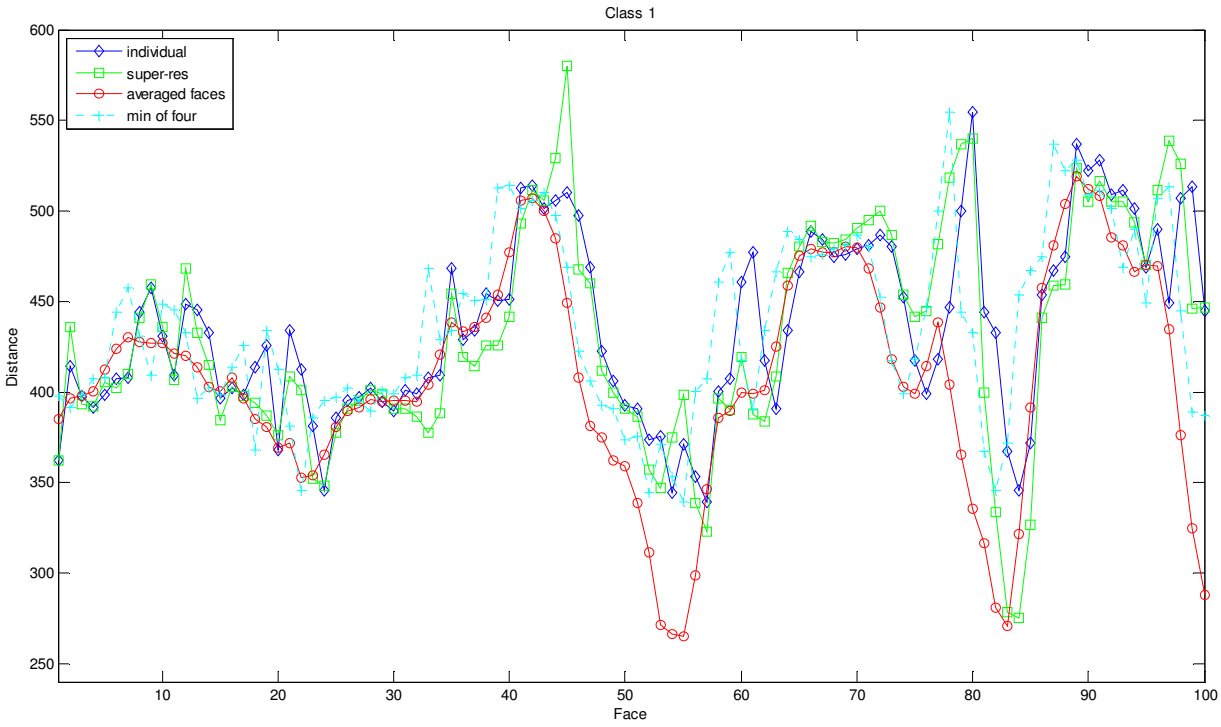


Figure 4.10: Plot of Distances for Class 1

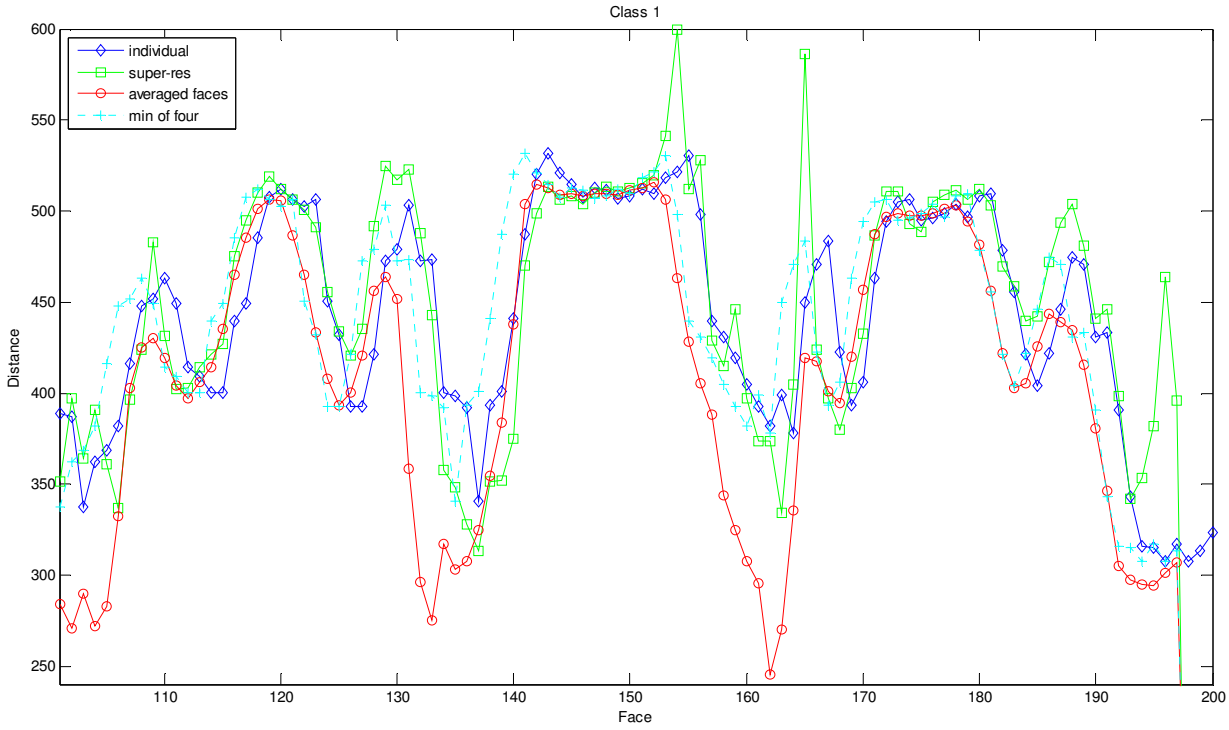


Figure 4.11: Plot of Distances for Class 1

Figures 4.12 and 4.13 show the results of Class 2, Gabriel. Out of the 298 faces that are combined through the sliding window, super-resolution had 52 combinations that had the smallest distance, average has 167 combinations, and the minimum of the four faces has 79 combinations.

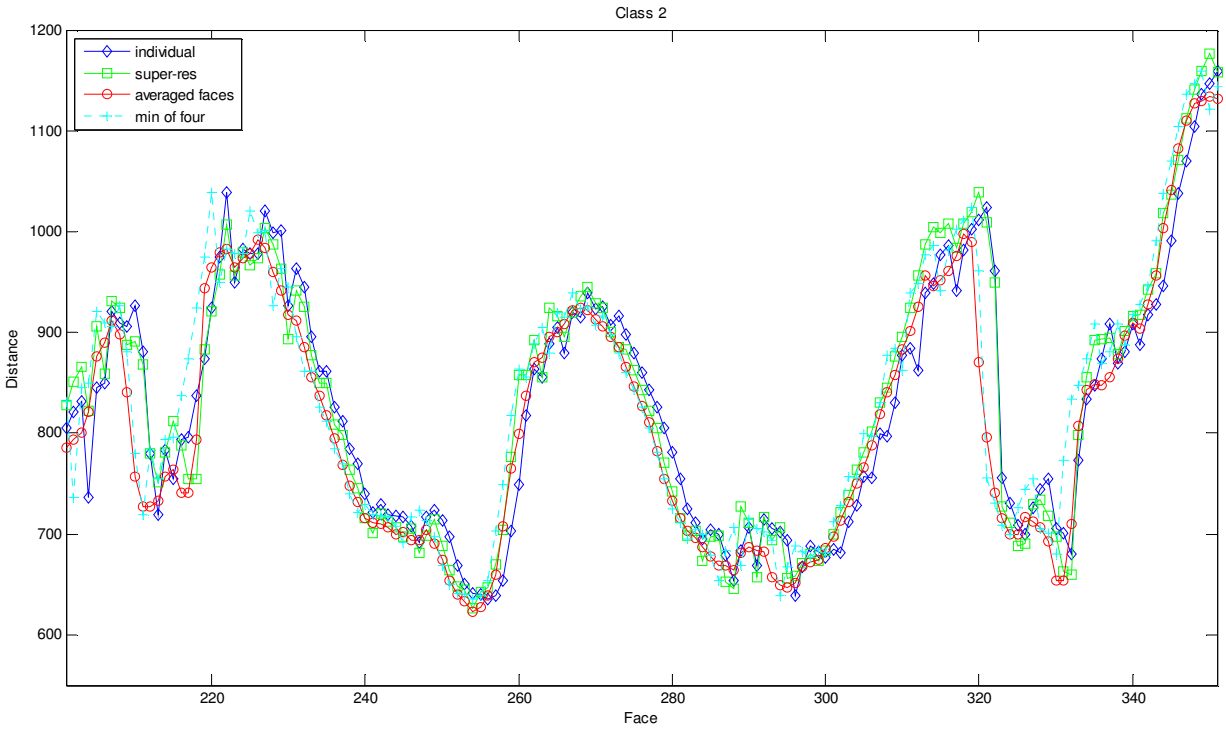


Figure 4.12: Plot of Distances for Class 2

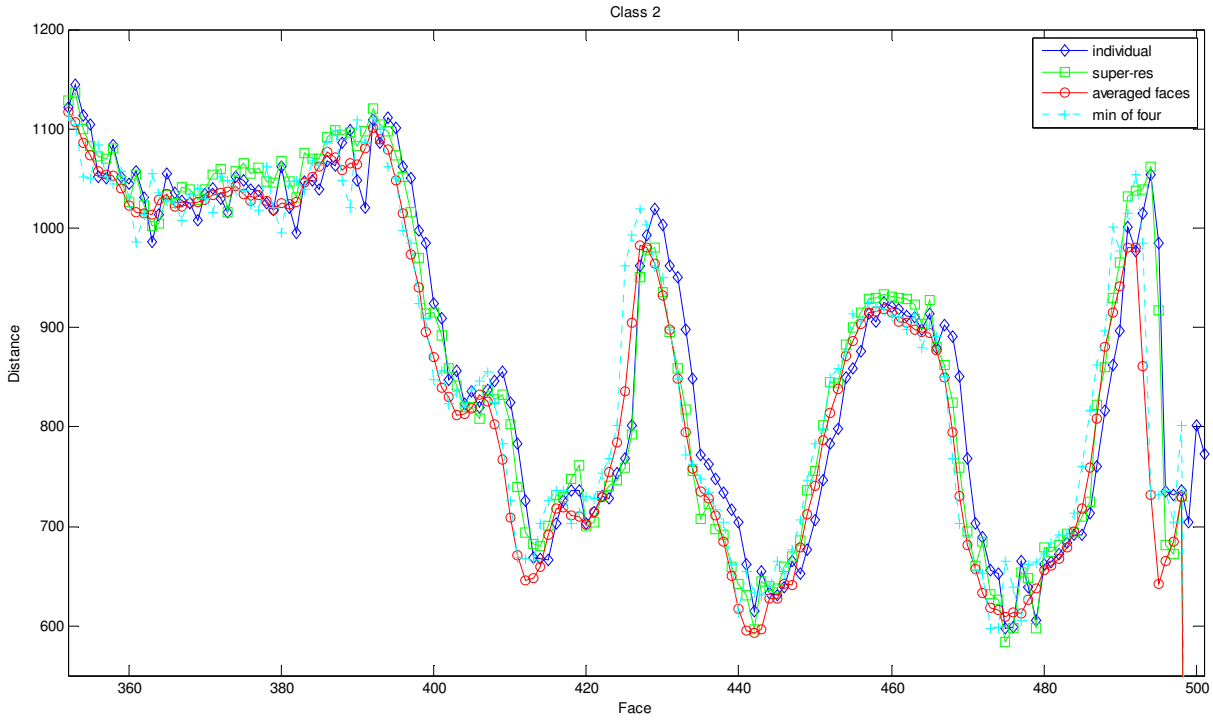


Figure 4.13: Plot of Distances for Class 2

A comparison of all the different methods were compared against each other and the individual faces. Figures 4.14 and 4.15 show the results of Class 3, Ricardo. Out of the 155 faces that are combined through the sliding window, super-resolution had 37 combinations that had the smallest distance, average has 94 combinations, and the minimum of the four faces had 24 combinations.

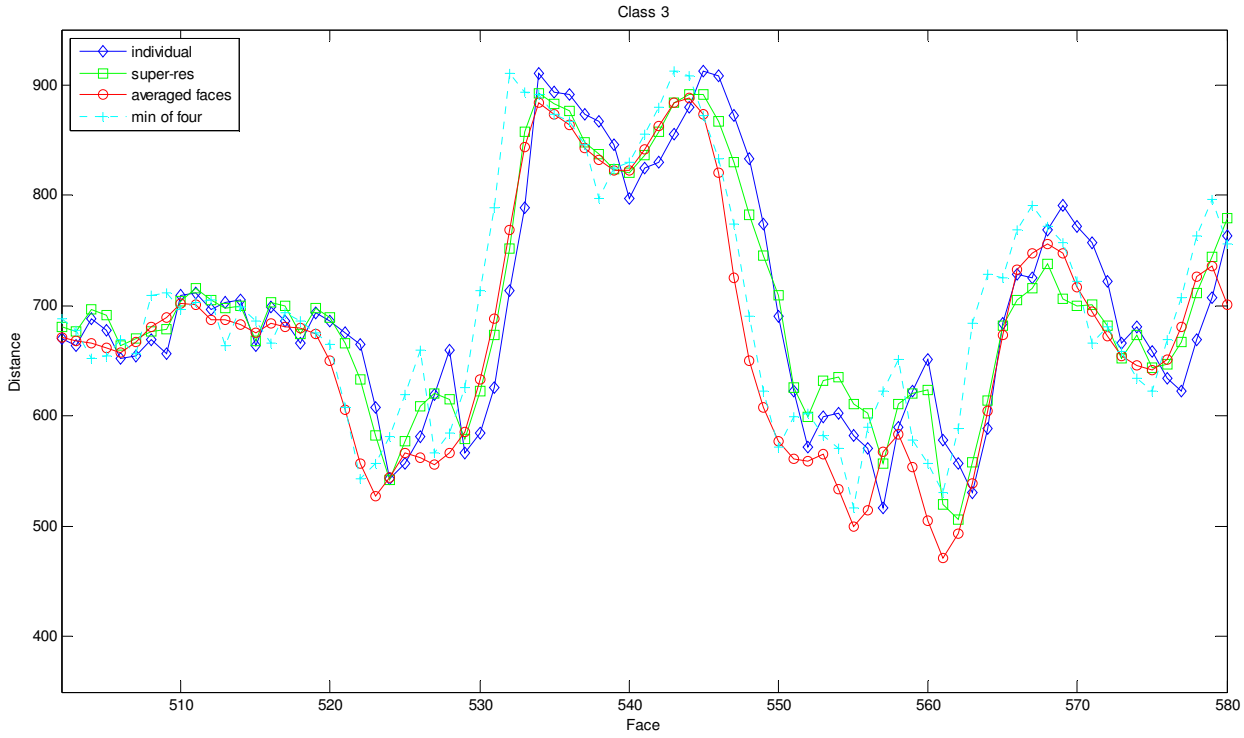


Figure 4.14: Plot of Distances for Class 3

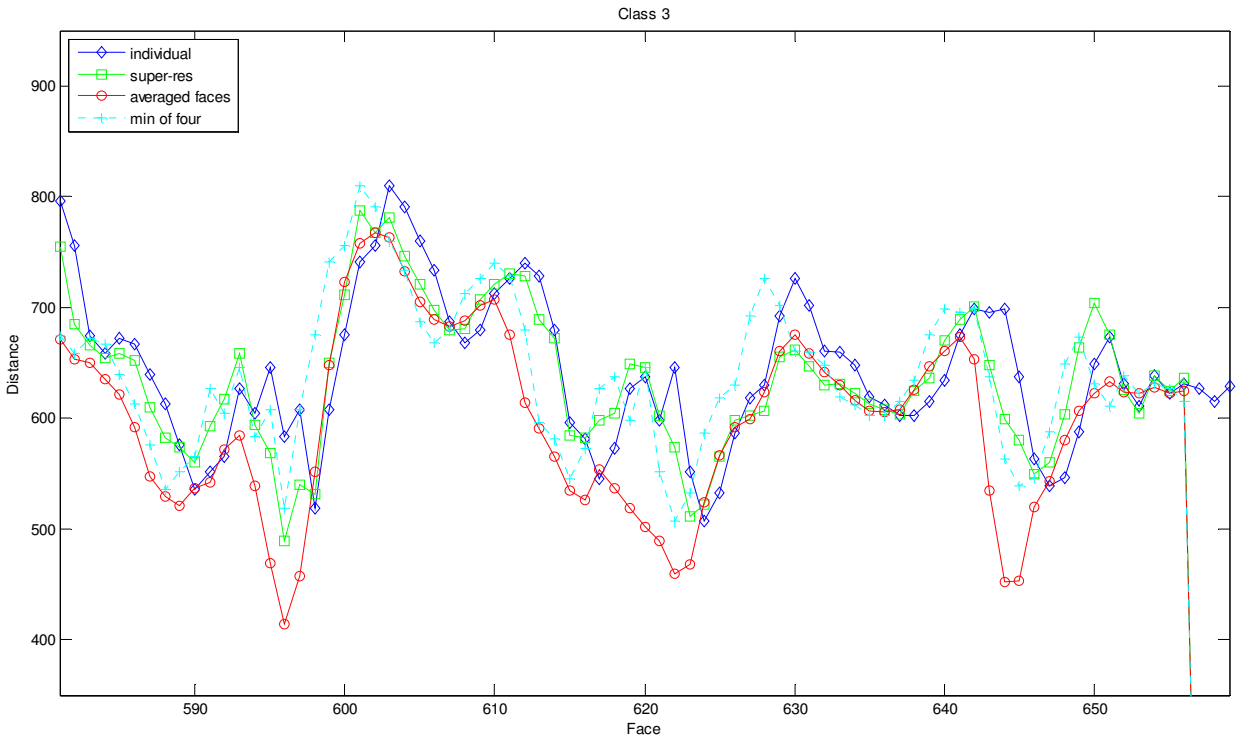


Figure 4.15: Plot of Distances for Class 3

Table 4.22 shows the overall results for the automatic cropping algorithm. The super-resolution algorithm had 141 out of the possible 650 combinations, 21.69%, as the algorithm with the smallest distance. The average had the most, 372 out of 650, combinations or 57.23% as the algorithm with the smallest distance. The minimum of four faces had the least, 137 out of 650, combinations or 21.08% as the algorithm with the smallest distance.

Table 4.22: Overall Results for Manual Cropping

| Overall Results | Smallest Distance | Percent |
|---|---|---|
| Super-Resolution | 141 | 21.69231 |
| Average | 372 | 57.23077 |
| Minimum of Four Faces | 137 | 21.07692 |

A check was done on the testing set to determine how well these faces did when tested. All of the faces had an accuracy of 100%. Table 4.23 shows the statistics of the self test which has a higher

mean than the individual methods in addition to a smaller variance. Table 4.24 shows the confusion matrix which has all the images classified as the correct class

Table 4.23: Statistics for the Training Set

| Class | | Accuracy (%) | Mean | Variance |
|---|---|---|---|---|
| 1 | 10/10. | 100 | 486.9734 | 2915.472 |
| 2 | 11/11. | 100 | 888.4116 | 25229.16 |
| 3 | 10/10. | 100 | 733.4317 | 4359.917 |

Table 4.24: Confusion Matrix for Training Set

| | Class 1 | Class 2 | Class 3 |
|---|---|---|---|
| Class 1 | 10 | 0 | 0 |
| Class 2 | 0 | 11 | 0 |
| Class 3 | 0 | 0 | 10 |

## 4.3 MOTION VECTOR ANALYSIS OF CENTROID SHIFTED AVERAGE AND SUPER-RESOLUTION REGISTRATION PARAMETERS

A comparison of the registration parameters between the centroid shifted average and the super-resolution was initiated to try to determine why the centroid shifted average greatly outperformed super-resolution. The motion vectors that were calculated using both algorithms have remarkably different results for the same set of faces. The magnitudes of the motion vectors were compared with each other and the mean and standard deviation was calculated for the second through fourth images. The direction of the motion vectors was ignored because there was no common shift between the faces. The first image in both registration techniques is considered the reference image, meaning no vectors are calculated. The magnitude was chosen because it gave the clearest indication as to why one registration method worked better than the other. Faces, 20, 38, 55, 63, 103, and 104 were compared to find a reason why one method outperformed the other.

In Table 4.25 to Table 4.30, the super-resolution algorithm outperformed the centroid shifted average algorithm. The motion vectors of the two algorithms are similar in Table 4.29, but the super-

resolution's sub-pixel accuracy seemed to improve the reconstructed image. The full pixel accuracy of the centroid shifted average was too large a shift for the images and that shift prevented the best alignment for the averaged face. In other cases, the motion vectors and the corresponding magnitudes from the centroid shifted average were too large and gave inaccurate registration compared to super-resolution.

Table 4.25: Motion Vectors for Face 20

| Face 20 Motion Vectors | X | Y | X | Y | X | Y |
|---|---|---|---|---|---|---|
| Super-Resolution | -0.3132 | -0.1909 | -0.6191 | -0.3027 | -0.7274 | -0.3436 |
| Centroid Shifted Average | 0 | 0 | 0 | 0 | 0 | 0 |

Table 4.26: Motion Vector Magnitudes for Face 20

| Face 20 | | | | Mean | Std |
|---|---|---|---|---|---|
| Super-Resolution | 0.366766 | 0.68911 | 0.804448 | 0.620108 | 0.226853 |
| Centroid Shifted Average | 0 | 0 | 0 | 0 | 0 |

Table 4.27: Motion Vectors for Face 38

| Face 38 Motion Vectors | X | Y | X | Y | X | Y |
|---|---|---|---|---|---|---|
| Super-Resolution | -0.10224 | -0.10592 | 0.084334 | -0.02811 | 0.319754 | -0.06198 |
| Centroid Shifted Average | 0 | -1 | 0 | -1 | 1 | -1 |

Table 4.28: Motion Vector Magnitudes for Face 38

| Face 38 | | | | Mean | Std |
|---|---|---|---|---|---|
| Super-Resolution | 0.147213 | 0.088896 | 0.325705 | 0.187272 | 0.123382 |
| Centroid Shifted Average | 1 | 1 | 1.414214 | 1.138071 | 0.239146 |

Table 4.29: Motion Vectors for Face 63

| Face 63 Motion Vectors | X | Y | X | Y | X | Y |
|---|---|---|---|---|---|---|
| Super-Resolution | 4.191695 | 0.003576 | 4.212668 | -0.00399 | 4.428698 | 0.040031 |
| Centroid Shifted Average | 4 | 0 | 4 | -1 | 4 | -1 |

Table 4.30: Motion Vector Magnitudes for Face 63

| Face 63 | | | | Mean | Std |
|---|---|---|---|---|---|
| Super-Resolution | 4.191696 | 4.21267 | 4.428879 | 4.277748 | 0.131302 |
| Centroid Shifted Average | 4 | 4.123106 | 4.123106 | 4.08207 | 0.071075 |

In Table 4.31 to Table 4.36, the centroid shifted average algorithm outperformed the super-resolution algorithm. The super-resolution algorithm seemed to shift the images too much and created a reconstructed image that did not perform as well as the centroid shifted averaged face. The means and standard deviations of the magnitude of the motion vectors are more centralized than the super-resolution, meaning there was little motion from the motion vectors. In this case, the sub-pixel accuracy of super-resolution over compensated for the faces rather than the full pixel accuracy of the centroid shifted average.

Table 4.31: Motion Vectors for Face 55

| Face 55 Motion Vectors | X | Y | X | Y | X | Y |
|---|---|---|---|---|---|---|
| Super-Resolution | 1.497674 | 0.312602 | 2.593723 | 0.474032 | 3.335138 | 0.503654 |
| Centroid Shifted Average | 1 | 0 | 1 | 0 | 1 | 0 |

Table 4.32: Motion Vector Magnitudes for Face 55

| Face 55 | | | | Mean | Std |
|---|---|---|---|---|---|
| Super-Resolution | 1.52995 | 2.636685 | 3.372953 | 2.513196 | 0.927686 |
| Centroid Shifted Average | 1 | 1 | 1 | 1 | 0 |

Table 4.33: Motion Vectors for Face 103

| Face 103 Motion Vectors | X | Y | X | Y | X | Y |
|---|---|---|---|---|---|---|
| Super-Resolution | 0.420263 | 0.159005 | 1.647536 | 0.392462 | 2.804909 | 0.42398 |
| Centroid Shifted Average | 0 | 1 | 0 | 1 | 1 | 1 |

Table 4.34: Motion Vector Magnitudes for Face 103

| Face 103 Magnitudes | | | | Mean | Std |
|---|---|---|---|---|---|
| Super-Resolution | 0.449336 | 1.693635 | 2.836772 | 1.659914 | 1.194075 |
| Centroid Shifted Average | 1 | 1 | 1.414214 | 1.138071 | 0.239146 |

Table 4.35: Motion Vectors for Face 104

| Face 104 Motion Vectors | X | Y | X | Y | X | Y |
|---|---|---|---|---|---|---|
| Super-Resolution | 1.307095 | 0.204165 | 2.650526 | 0.105813 | 3.393289 | 0.007849 |
| Centroid Shifted Average | 0 | 0 | 1 | 0 | 1 | 0 |

Table 4.36: Motion Vector Magnitudes for Face 104

| Face 104 Magnitudes | | | | Mean | Std |
|---|---|---|---|---|---|
| Super-Resolution | 1.322944 | 2.652637 | 3.393298 | 2.456293 | 1.049049 |
| Centroid Shifted Average | 0 | 1 | 1 | 0.666667 | 0.57735 |

## 4.4 LARGE FACE COMPARISON

A comparison was done using the cropped images before the resizing to see the effects of super-resolution and the centroid shifted average on the larger images. This comparison was to test the registration parameters of both algorithms on a full size image and if any improvements are made in classification accuracy and if any improvements are made in the distance measurement. Full size faces refer to the original size of the faces that are zero padded to a standard size. This section was done separately from the others using a memory node to speed up the computational time required for both the centroid shifted average and especially super-resolution.

Figure 4.16 shows a group of four faces that have been artificially brightened for viewing but the original images were used for the experiment. Those faces were the entered into the super-resolution algorithm producing the resulting face shown in Figure 4.17. The resulting face shows some of the rotation that was seen by the Keren registration algorithm while in Figure 4.18, the resulting centroid shifted average resulting image there is no visible shift.

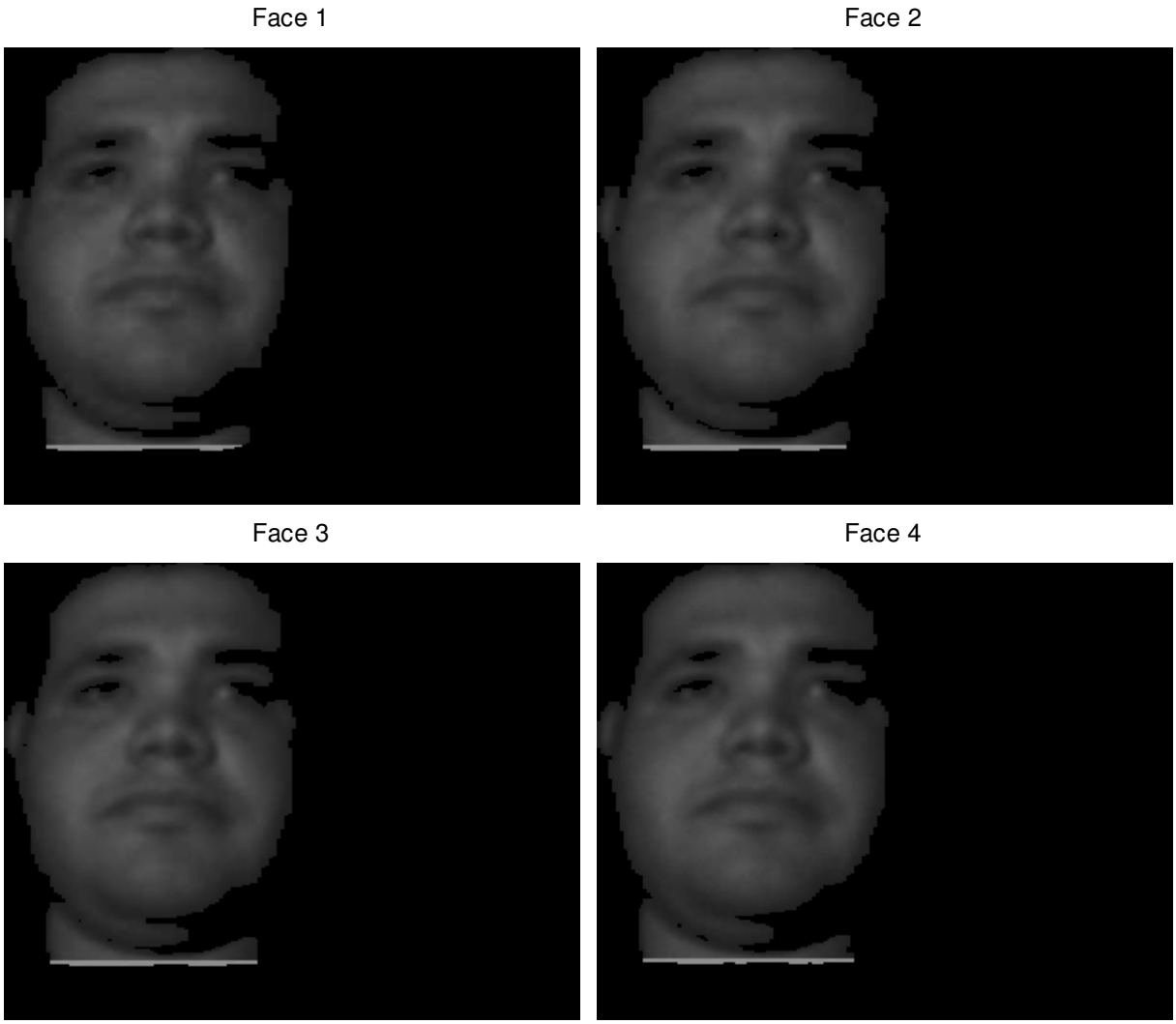


Figure 4.16: Four Faces of the Large Face Comparison (Artificially Brightened)

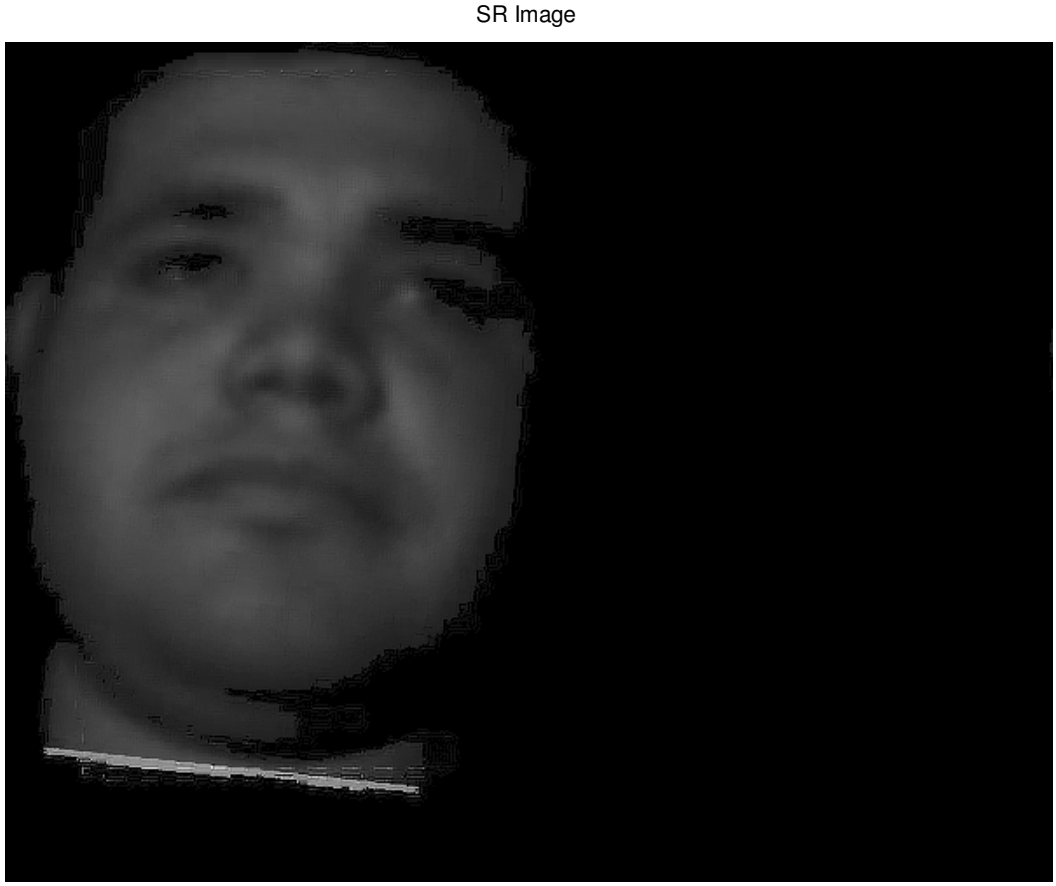


Figure 4.17: Resulting SR Image of Large Face Comparison (Artificially Brightened)

Figure 4.18: Resulting Centroid Shifted Average of Large Face Comparison (Artificially Brightened)

Figures 4.19 to 4.24 show the results of the centroid shifted average and super-resolution distances from the mean. The centroid shifted average outperformed super-resolution for all three

classes. For Classes 2 and 3 the centroid shifted average overwhelmingly outperformed super-resolution. There are a few cases in Class 2 where super-resolution performed better and no cases in Class 3. For Class 1 there are a noticeable number of cases where super-resolution outperforms the centroid shifted average, but not enough to be significant.

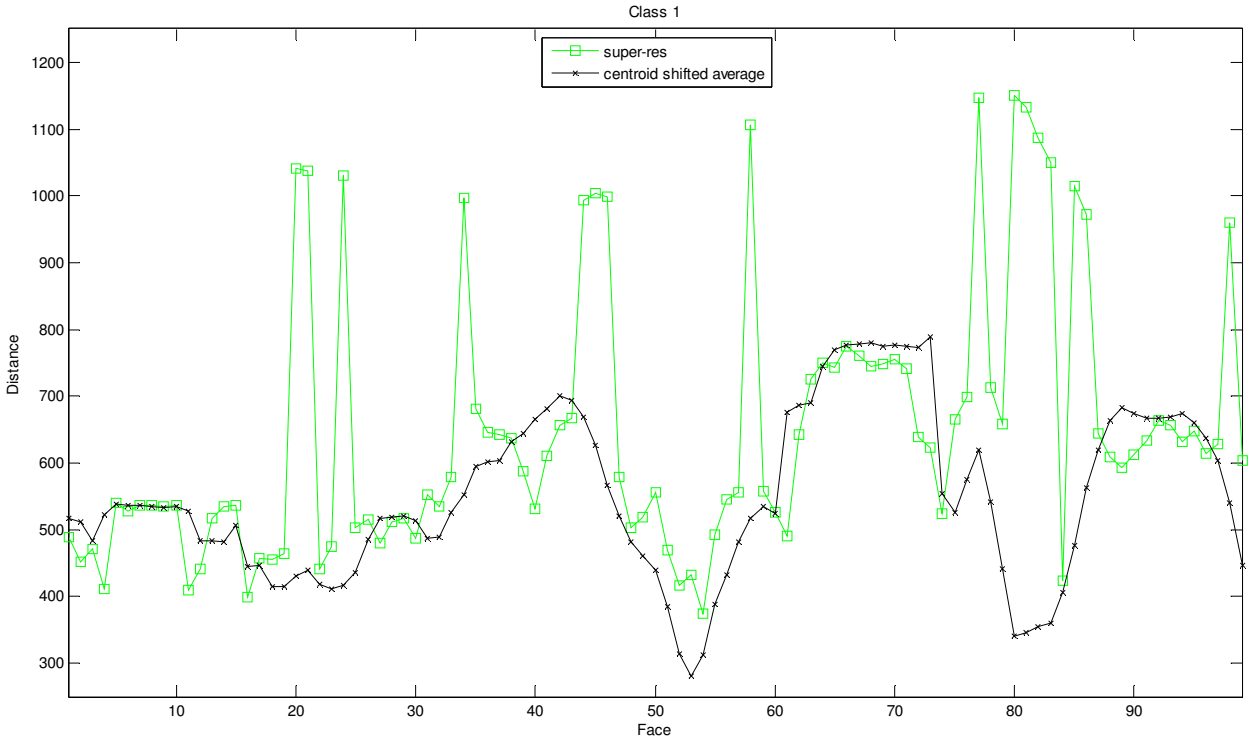


Figure 4.19: Plot of Distances for Class 1 Full Size

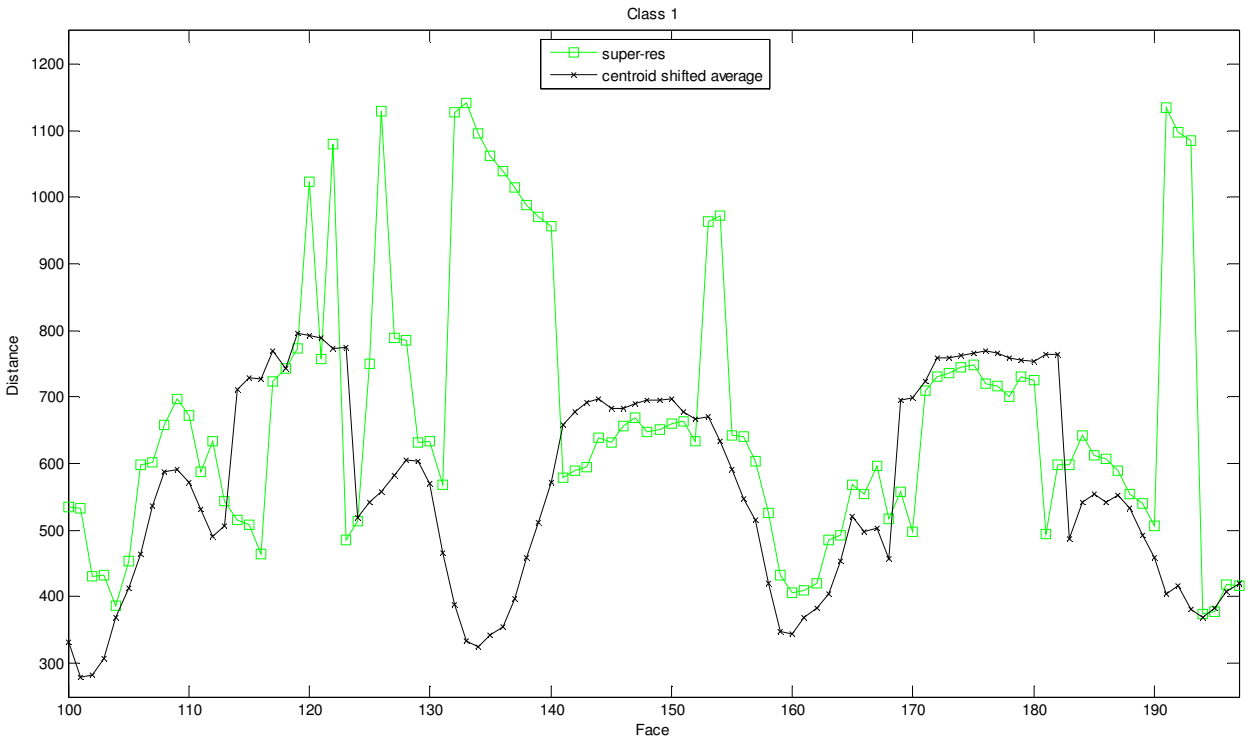


Figure 4.20: Plot of Distances for Class 1 Full Size

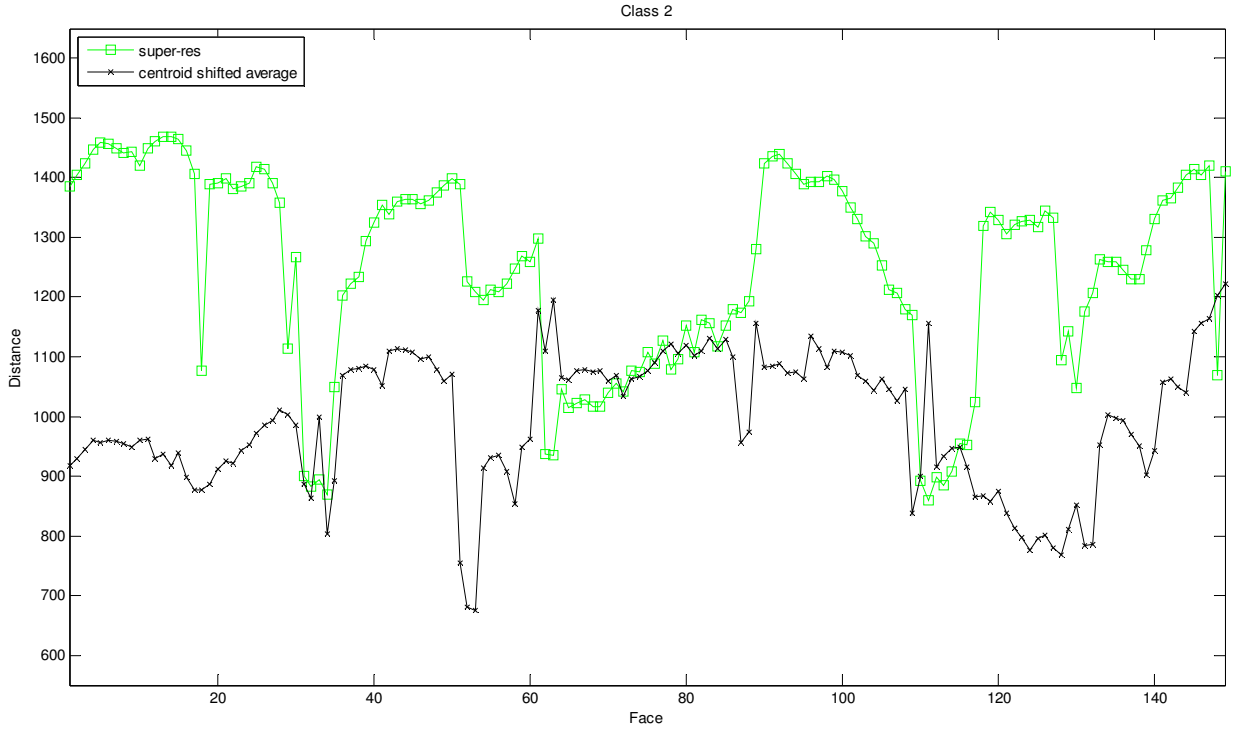


Figure 4.21: Plot of Distances for Class 2 Full Size

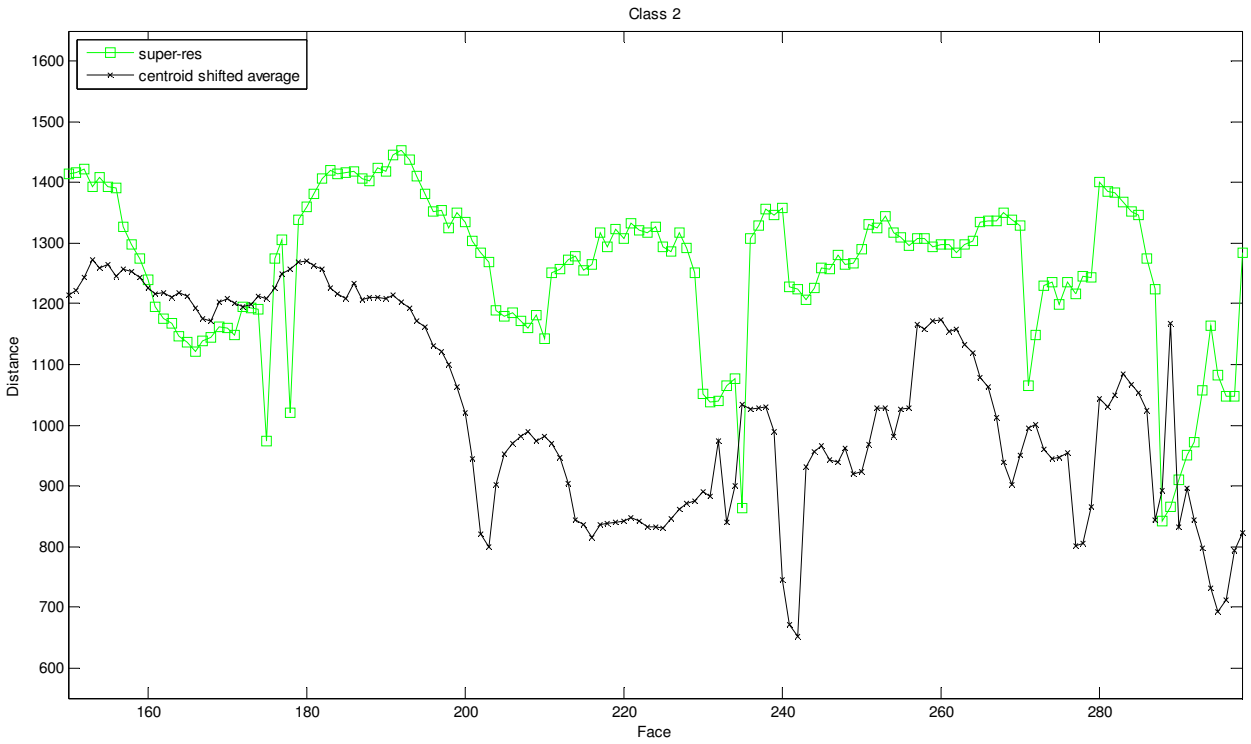


Figure 4.22: Plot of Distances for Class 2 Full Size

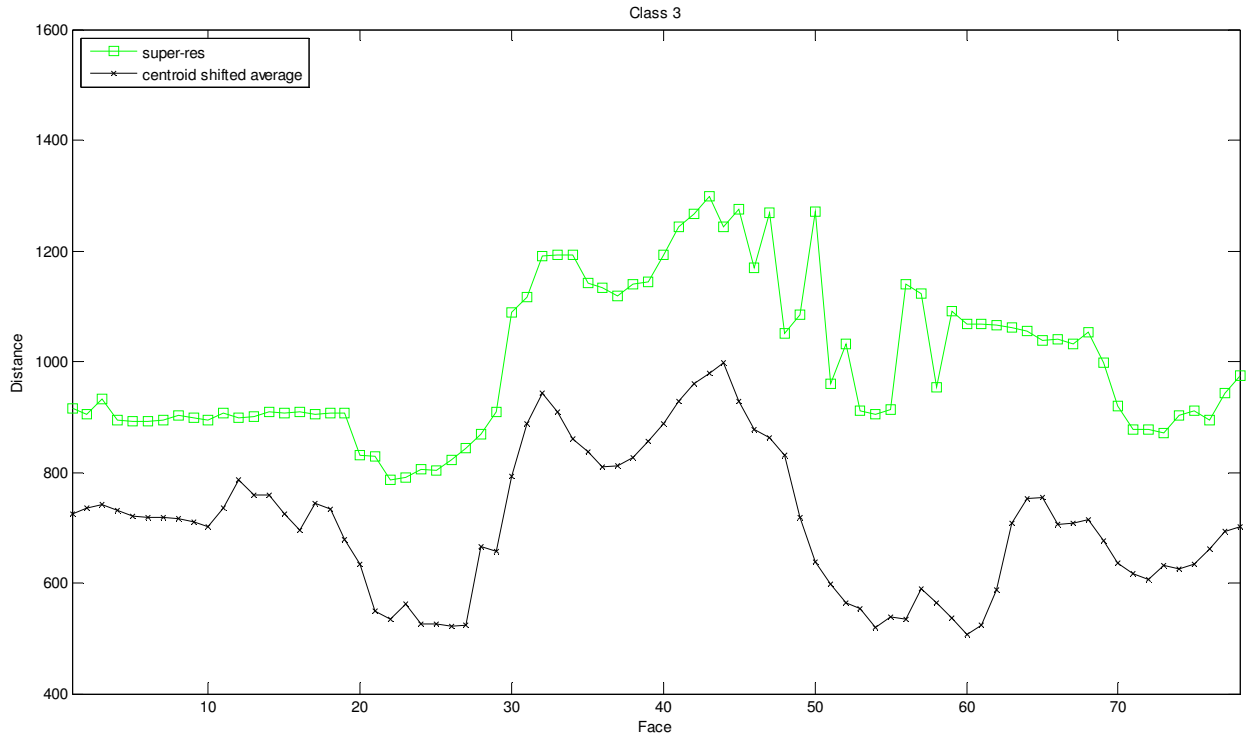


Figure 4.23: Plot of Distances for Class 3 Full Size

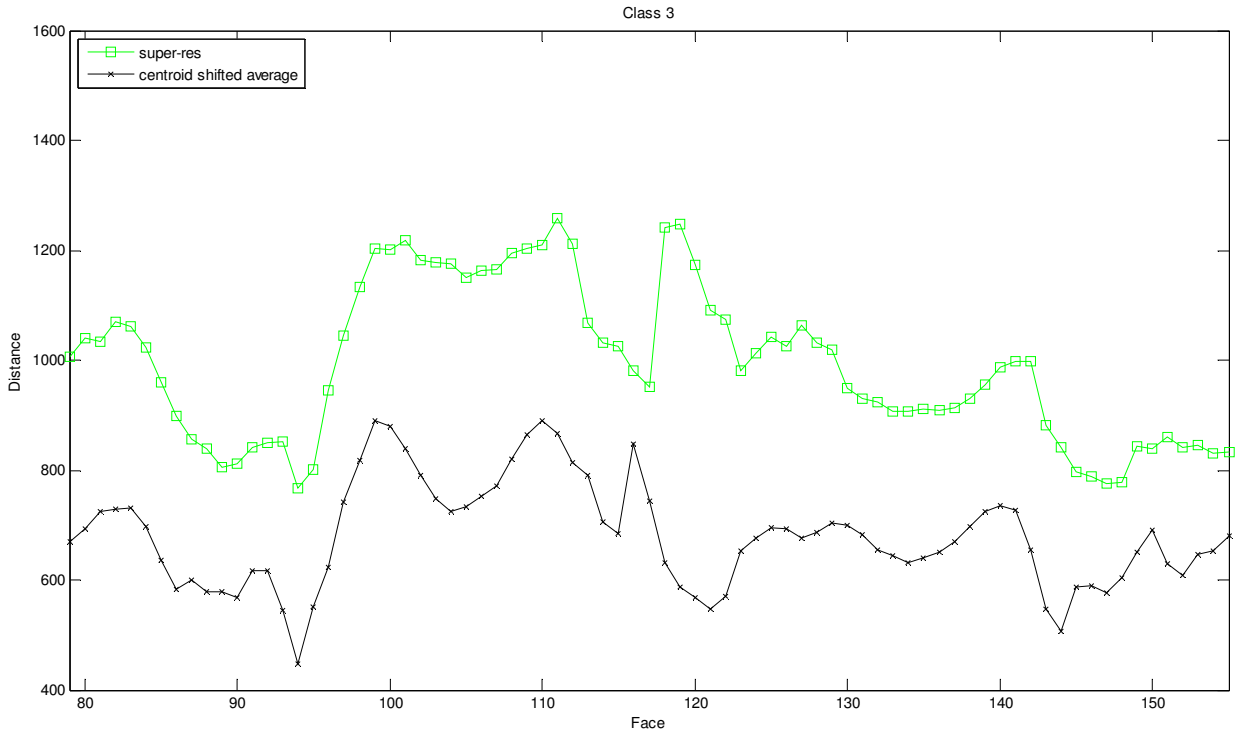


Figure 4.24: Plot of Distances for Class 3 Full Size

Table 4.37 and shows the statistics for the large face super-resolution and Table 4.38 shows the corresponding confusion matrix. The accuracies for the large faces are worse than the resized faces having lower accuracies and higher means and variances. The accuracies of these larger faces start to fall below 90%, which was the value considered acceptable with the resized faces. Class 1 is the only class that was above 90% while Class 2 and Class 3 were above 80%. Class 1 had 10 faces incorrectly classified as Class 3. Class 2 had 51 faces incorrectly classified as Class 1 and 7 faces incorrectly classified as Class 3. Class 3 had 23 faces incorrectly classified as Class 1.

Table 4.37: Statistics for the Large Face Super-Resolution

| Class | | Accuracy (%) | Mean | Variance |
|---|---|---|---|---|
| 1 | 187/197 | 94.92386 | 659.709 | 40423.66 |
| 2 | 240/298 | 80.53691 | 1067.654 | 20800.98 |
| 3 | 132/155 | 85.16129 | 862.0387 | 66933.4 |

Table 4.38: Confusion Matrix for the Large Face Super-Resolution

| | Class 1 | Class 2 | Class 3 |
|---|---|---|---|
| Class 1 | 187 | 0 | 10 |
| Class 2 | 51 | 240 | 7 |
| Class 3 | 23 | 0 | 132 |

Table 4.39 and shows the statistics for the large face centroid shifted average and Table 4.40 shows the corresponding confusion matrix. The accuracies for the large faces are very similar to the resized faces having slightly lower accuracies for Class 1 and Class 2 while Class 3 remained the same. The means for the large faces were similar with very little difference while the variances were slightly larger. Class 3 is the only class to fall below 90%, which was the value considered acceptable with the resized faces, but only slightly. While the results were similar for the resized faces and the full sized faces, the full sized faces on average had smaller distances. Class 1 had 3 faces incorrectly classified as Class 3, Class 2 had 31 faces incorrectly classified as Class 1, and Class 3 had 1 face incorrectly classified as Class 1.

Table 4.39: Statistics for the Large Face Centroid Shifted Average

| Class | | Accuracy (%) | Mean | Variance |
|---|---|---|---|---|
| 1 | 194/197 | 98.47716 | 557.2442 | 18893.67 |
| 2 | 267/298 | 89.59732 | 1011.654 | 20011.77 |
| 3 | 154/155 | 99.35484 | 693.5857 | 12715.65 |

Table 4.40: Confusion Matrix for the Large Face Centroid Shifted Average

| | Class 1 | Class 2 | Class 3 |
|---|---|---|---|
| Class 1 | 194 | 0 | 3 |
| Class 2 | 31 | 267 | 0 |
| Class 3 | 1 | 0 | 154 |

One reason why super-resolution performed so poorly on the large face images was the fact that super-resolution uses circular shifts. Circular shifts are where the pixels that would be shifted out of the

image are place on the opposite side of the image. Figure 4.25 shows the reconstructed image with the ear visible on the far right of the image where the centroid shifted average does not suffer from the circular shift as shown in Figure 4.26. When images such as these are resized for face recognition the entire image is resized instead of the face and the results are not as accurate as they could be.


Figure 4.25: Large Face 134 Circular Shifting Seen on Reconstructed Image (Artificially Brightened)

Face 134

Figure 4.26: Large Face 134 Using Centroid Shifted Average (Artificially Brightened)

Table 4.42 shows the motion vectors for Large Face 134, and that circular shift affects the motion vectors. The centroid shifted average outperformed super-resolution for this example and is the primary reason for the large spikes in distances that occur especially for Class 1. The motion vectors for super-resolution are going in the wrong direction and the magnitudes are shifting the image too far.

Table 4.41: Motion Vectors for Large Face 134

| Face 134 Motion Vectors | X | Y | X | Y | X | Y |
|---|---|---|---|---|---|---|
| Super-Resolution | 4.196402 | -7.8993 | 6.539887 | -13.587 | 6.593701 | -17.813 |
| Centroid Shifted Average | 0 | -5 | -3 | -8 | -4 | -10 |

Table 4.42: Motion Vector Magnitudes for Large Face 134

| Face 134 Magnitudes | | | | Mean | Std |
|---|---|---|---|---|---|
| Super-Resolution | 8.944748 | 15.0792 | 18.99485 | 14.3396 | 5.065707 |
| Centroid Shifted Average | 5 | 8.544004 | 10.77033 | 8.104778 | 2.910132 |

Figure 4.27 shows Large Face 143 that was reconstructed using super-resolution and Figure 4.28 shows Large Face 143 that was reconstructed using the centroid shifted average. For this face super-resolution performed better than the centroid shifted average

Face 143

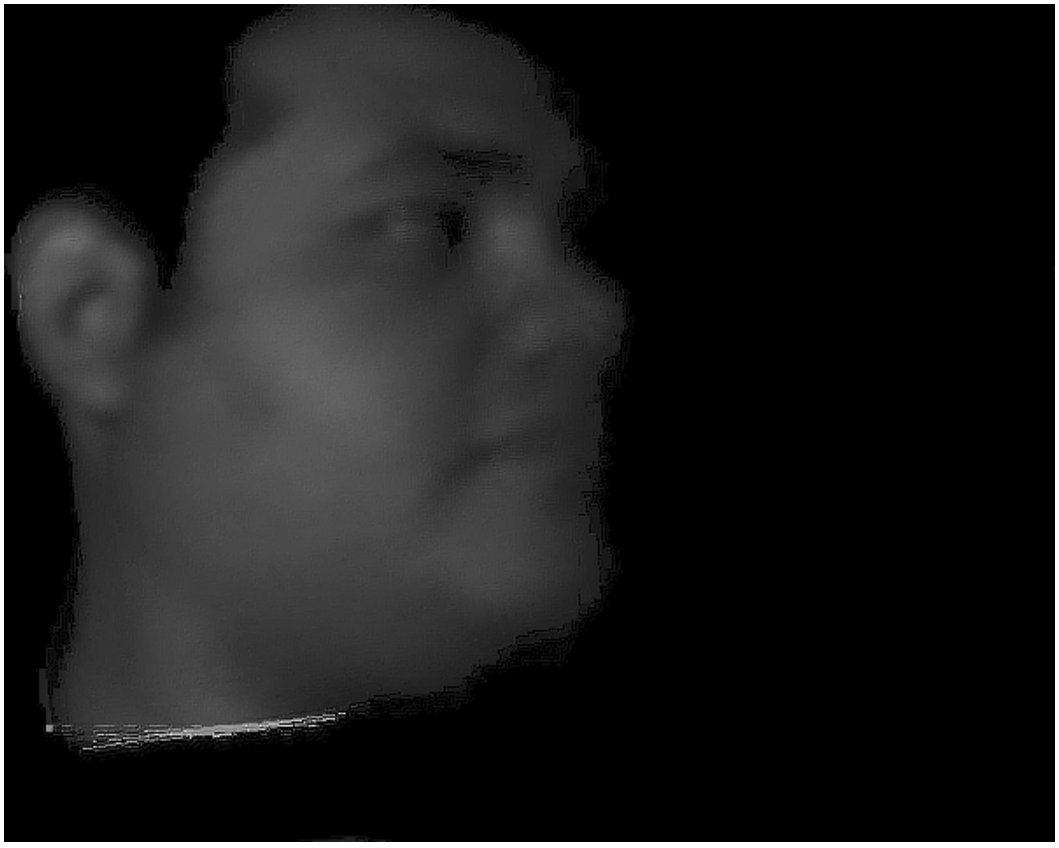

Figure 4.27: Super-Resolution Large Face 143 (Artificially Brightened)

Face 143

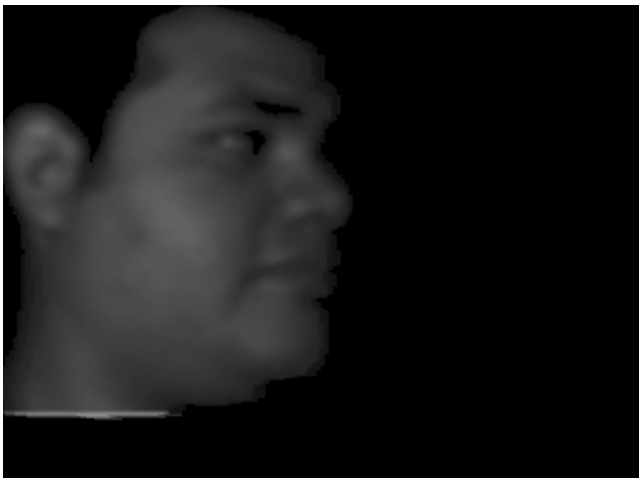

Figure 4.28: Large Face 143 Using Centroid Shifted Average (Artificially Brightened)

Table 4.43 shows the motion vectors for Large Face 143 and Table 4.44 shows the motion vector magnitudes for Large Face 143 where the super-resolution algorithm outperformed the centroid shifted average algorithm. The centroid shifted average did not shift the image far enough in the horizontal direction and that shift prevented the reconstruction to be more accurate that what was reconstructed. The magnitudes for the first face are close, but the subsequent faces show the shifts had to be larger.

Table 4.43: Motion Vectors for Large Face 143

| Face 143 Motion Vectors | X | Y | X | Y | X | Y |
|---|---|---|---|---|---|---|
| Super-Resolution | -5.3588 | 2.2545 | -8.5064 | 4.047039 | -12.271 | 4.513082 |
| Centroid Shifted Average | 0 | 2 | -1 | 3 | -1 | 3 |

Table 4.44: Motion Vector Magnitudes for Large Face 143

| Face 143 Magnitudes | | | | Mean | Std |
|---|---|---|---|---|---|
| Super-Resolution | 5.813728 | 9.420061 | 13.07433 | 9.436038 | 3.630325 |
| Centroid Shifted Average | 2 | 3.162278 | 3.162278 | 2.774852 | 0.671041 |

# Chapter 5: Conclusions and Future Work

## 5.1 CONCLUSIONS

While all of the algorithms used to improve face recognition improved the distance, the centroid shifted average outperformed all the other methods with 320 faces shown in Table 5.1. The drawback from the centroid shifted average is the amount of time that is taken to compute the centroid and to align the images. Over thirty minutes was used to compute the centroid shifted average compared with the twenty minutes for the super-resolution and two minutes for the average, which came in second for having the smallest distance with 135 faces. The average also helped improve face recognition more than super-resolution; however, a potential problem would be the zeros that are part of the background that are averaged with the actual faces. This could be affecting the results. Super-Resolution performed the third worst with 112 faces, but it is image dependant and could have had a problem with the reduced image size and low resolution. An increase in resolution could improve the image quality for super-resolution to outperform the average and centroid shifted average. Another change would be to use a different registration algorithm. The minimum of four faces performed the worst out of all the algorithms when it came to having the smallest distance with 83 faces. Table 5.2 shows the means for each class using the different algorithms with automatic cropping.

Table 5.1: Summary of Results for Automatic Cropping

| Overall Results | Smallest Distance | Percent |
|---|---|---|
| Super-Resolution | 112 | 17.23077 |
| Average | 135 | 20.76923 |
| Centroid Shifted Average | 320 | 49.23077 |
| Minimum of Four Faces | 83 | 12.76923 |

Table 5.2: Class Means for the Different Algorithms using Automatic Cropping

| Method | Class 1 Means | Class 2 Means | Class 3 Means |
|---|---|---|---|
| Face Recognition | 596.7527304 | 1052.38467 | 749.3440564 |
| Super-Resolution | 575.3297785 | 1046.558858 | 733.7816365 |
| Centroid Shifted Average | 528.6123027 | 987.8687158 | 660.226063 |
| Average | 552.852112 | 1029.33887 | 705.9206828 |
| Minimum of Four Faces | 554.058529 | 1023.093139 | 694.5252172 |

For the manual cropping, the average outperformed the other algorithms with 372 faces, with super-resolution second with 141 faces, and the minimum of four faces last with 137 faces as seen in Table 5.3. The manually cropped images had distances that where significantly smaller than the automatically cropped faces as shown in Table 5.4. This means that the automatically cropped faces could be better cropped using a different color space or a more complex algorithm to locate the face. This means that there was either information lost during the skin segmentation that was accounted for during the manual scan, or the background that was kept during the manual cropping was not background but actually a part of the face.

Table 5.3: Summary of Results for Manual Cropping

| Overall Results | Smallest Distance | Percent |
|---|---|---|
| Super-Resolution | 141 | 21.69231 |
| Average | 372 | 57.23077 |
| Minimum of Four Faces | 137 | 21.07692 |

Table 5.4: Class Means for the Different Algorithms using Manual Cropping

| Method | Class 1 Means | Class 2 Means | Class 3 Means |
|---|---|---|---|
| Face Recognition | 438.917994 | 851.0562049 | 670.1111967 |
| Super-Resolution | 440.7890269 | 853.3526247 | 668.1459562 |
| Average | 409.9607912 | 835.0850796 | 637.9660281 |
| Minimum of Four Faces | 413.505827 | 815.8965465 | 629.2864257 |

## 5.2 FUTURE WORK

One way to help improve the results of face recognition would be to use a different procedure for super-resolution. Another method for super-resolution registration would be to use the Vandevalle registration method instead of the Keren method. The Vandevalle method is claimed to be more accurate than the Keren method [8]. Also, instead of using the Robust Super-Resolution algorithm, a maximum a priori statistical method could be used to reconstruct the super-resolved image. Larger image sizes could also be used to determine if the image size was too small and too much data was lost.

Another possible improvement would be to use a more complex face identifier. The one used in this experiment was a simple skin segmentation algorithm that used different color spaces to empirically find boundaries that are related to human skin color. A more complex face identifier would be to use a statistical model of the face [21].

Another addition would be to investigate why the average performed so well. The average performed second best and is a linear operation meaning it would correspond to an average in the face space. This average in the face space could be the reason why it performed so well and see of any slight modifications could produce even better results.

The faces used in the experiment were very similar in skin tones and gender, males. For the future more varied types of faces will be used. In addition to adding more faces with different skin tones, adding females and different ages to the experiment will be beneficial to see if the Fisherfaces algorithm can accurately identify the wide variety of faces.

# Bibliography


[1] D. Litwiller, "CCD vs. CMOS: Facts and Fiction," *Photonics Spectra*, no. 1, January 2001

[2] Shashua, "Geometry and Photometry in 3D Visual Recognition," PhD diss., Massachusetts Institute of Technology, 1992.

[3] S.C. Park, M.K. Park, and M.G. Kang, "Super-Resolution Image Reconstruction: A Technical Overview," *IEEE Signal Processing Magazine*, vol. 20, no. 3, pp. 21–36, May 2003.

[4] W. Zhao, R. Chellappa, P.J. Phillips, and A. Rosenfeld, "Face Recognition: A Literature Survey," *ACM Comput. Surveys,* vol. 35, no. 4, pp. 399-458, December 2003.

[5] H. A. Rowley, S. Baluja, and T. Kanade, "Neural network-based face detection," *IEEE Trans. Pattern Anal. Machine Intell.,* vol. 20, no. 1, pp. 23-38, January 1998.

[6] K. Sung, and T. Poggio, "Example-based learning for view-based human face detection," *IEEE Trans. Pattern Anal. Machine Intell.,* vol. 20, no. 1, pp. 39-51, January 1997.

[7] D. Keren, S. Peleg, and R. Brada, "Image sequence enhancement using sub-pixel displacements," in *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, June 1988, pp. 742–746.

[8] P. Vandewalle, S. Süsstrunk, and M. Vetterli, "A frequency domain approach to registration of aliased images with application to super-resolution,**"** *EURASIP Journal on Applied Signal Processing*, vol. 2006, pp. 233-233, 2006.

[9] A. Zomet, A. Rav-Acha, and S. Peleg, "Robust superresolution," in *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, vol. 1, 2001, pp. 645–650.

[10] D. Capel and A. Zisserman, "Computer Vision Applied to Super-Resolution," *IEEE Signal Processing Magazine*, vol. 20, no. 3, pp. 75–86, May 2003.

[11] B.K. Gunturk, A.U. Batur, Y. Altunbasak, M.H. Hayes, and R.M. Mersereau, "Eigenface-Domain Super-Resolution for Face Recognition," *IEEE Transactions on Image Processing*, vol. 12, no. 5, pp. 597–606, May 2003.

[12] P.N. Belhumeur, J.P. Hespanha, and D.J. Kriegman, "Eigenfaces vs. Fisherfaces: Recognition Using Class Specific Linear Projection," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 19, no. 7, pp. 711-729, July 1997.

[13] M. Turk and A. Pentland, "Eigenfaces for Recognition," *J. Cognitive Neuroscience*, vol. 3, no. 1, 1991.

[14] R.O. Duda, P.E. Hart, and D.G. Stork, *Pattern Classification*, 2d ed, New York: Wiley-Interscience, 2001.

[15] M. Elad and A. Feuer, "Restoration of a single super-resolution image from several blurred, noisy, and undersampled measured images," *IEEE Transactions on Image Processing*, vol. 6, no. 12, pp. 1646–1658, December 1997.

[16] M. Irani and S. Peleg, "Improving resolution by image registration," *CVGIP: Graphical Models and Image Processing*, vol. 53, no. 3, pp. 231–239, 1991.

[17] F. Gasparini and R. Schettini, "Skin Segmentation Using Multiple Thresholding," in *Proc. SPIE*, vol. 6061, pp. 60610F-1 - 60610F-8, January 2006.

[18] http://upload.wikimedia.org/wikipedia/commons/0/03/RGB_farbwuerfel.jpg

[19] http://blog.its.ac.id/masgandhul/2008/11/07/ycbcr-formats/

[20] S.E. Umbaugh, *Computer Imaging Digital Image Analysis and Processing*, New York: CRC Press, 2005.

[21] S.Z. Li and A.K. Jain, Ed., *Handbook of Face Recognition*, New York: Springer, 2005.

# Appendix

## Representative MATLAB Code

Here is the main algorithm for the automatic cropping algorithm. This code is the main shell with calls to other functions to crop the images, perform the face recognition algorithms, and the modifications to the faces as well as other function calls.

```
function
[result,result_superres,result_average,result_min_of_four,result_centroid_shifted_a
verage,compare_all]=automated_cropping()
    clear all;close all;clc;
%%%%%class 1
    if (exist('francisco_info.dat')==2)
        load('francisco_info.dat','-mat');
    else
        readerobj=mmreader('francisco.wmv');
        vidFrames=read(readerobj);
        numFrames=get(readerobj,'numberOfFrames');
        for k=1:numFrames
            mov(k).cdata=vidFrames(:,:,:,k);
            mov(k).colormap=[];
            temp=cropping(mov(k).cdata);
            temp=imresize(temp,[56 46],'bicubic');
            francisco_info{k,1}=temp(:);
            save('francisco_info.dat','francisco_info');
        end
    end
%%%%%%%class 2
    if (exist('gabriel_info.dat')==2)
        load('gabriel_info.dat','-mat');
    else
        readerobj=mmreader('gabriel.wmv');
        vidFrames=read(readerobj);
        numFrames=get(readerobj,'numberOfFrames');
        for k=1:numFrames
            mov(k).cdata=vidFrames(:,:,:,k);
            mov(k).colormap=[];
            temp=cropping(mov(k).cdata);
            temp=imresize(temp,[56 46],'bicubic');
            gabriel_info{k,1}=temp(:);
            save('gabriel_info.dat','gabriel_info');
        end
    end
%%%%%class 3
    if (exist('ricardo_info.dat')==2)
        load('ricardo_info.dat','-mat');
    else
        readerobj=mmreader('ricardo.wmv');
        vidFrames=read(readerobj);
        numFrames=get(readerobj,'numberOfFrames');
        for k=1:numFrames
            mov(k).cdata=vidFrames(:,:,:,k);
            mov(k).colormap=[];
            temp=cropping(mov(k).cdata);
            temp=imresize(temp,[56 46],'bicubic');
```

```
            ricardo_info{k,1}=temp(:);
            save('ricardo_info.dat','ricardo_info');
        end
    end

%calculate training set
%%%%%class 1
    [row col]=size(francisco_info);
    skip=round(row/10);
    k=1;
    for i=1:skip:row
        training_set{k,2}=francisco_info{i,1};
        training_set{k,1}=1;
        skipped_images_francisco(k)=i;
        k=k+1;
    end
%%%%%class 2
    [row col]=size(gabriel_info);
    [row1 col1]=size(training_set);
    skip=round(row/10);
    k=1;
    for i=1:skip:row
        training_set{k+row1,2}=gabriel_info{i,1};
        training_set{k+row1,1}=2;
        skipped_images_gabriel(k)=i;
        k=k+1;
    end
%%%%%class 3
    [row col]=size(ricardo_info);
    [row1 col1]=size(training_set);
    skip=round(row/10);
    k=1;
    for i=1:skip:row
        training_set{k+row1,2}=ricardo_info{i,1};
        training_set{k+row1,1}=3;
        skipped_images_ricardo(k)=i;
        k=k+1;
    end
%%%%%
    training_set{1,3}=k+row1-1;

    %calculate testing sets
%%%%%%%%class 1
    [row col]=size(skipped_images_francisco);
    [row1 col1]=size(francisco_info);
    i=1;
    l=1;
    while i<col
        for j=skipped_images_francisco(i)+1:skipped_images_francisco(i+1)-1
            testing_set_francisco{l,1}=francisco_info{j,1};
            l=l+1;
        end
        i=i+1;
    end
    for j=skipped_images_francisco(col)+1:row1
        testing_set_francisco{l,1}=francisco_info{j,1};
        l=l+1;
    end
```

```
    % clear row col row1 col1;
%%%%%%%%class 2
    [row col]=size(skipped_images_gabriel);
    [row1 col1]=size(gabriel_info);
    i=1;
    l=1;
    while i<col
        for j=skipped_images_gabriel(i)+1:skipped_images_gabriel(i+1)-1
            testing_set_gabriel{l,1}=gabriel_info{j,1};
            l=l+1;
        end
        i=i+1;
    end
    for j=skipped_images_gabriel(col)+1:row1
        testing_set_gabriel{l,1}=gabriel_info{j,1};
        l=l+1;
    end
%%%%%%%%class 3
    [row col]=size(skipped_images_ricardo);
    [row1 col1]=size(ricardo_info);
    i=1;
    l=1;
    while i<col
        for j=skipped_images_ricardo(i)+1:skipped_images_ricardo(i+1)-1
            testing_set_ricardo{l,1}=ricardo_info{j,1};
            l=l+1;
        end
        i=i+1;
    end
    for j=skipped_images_ricardo(col)+1:row1
        testing_set_ricardo{l,1}=ricardo_info{j,1};
        l=l+1;
    end
%%%%%%%%

    [row col]=size(testing_set_francisco);
    [row1 col1]=size(testing_set_gabriel);
    [row2 col2]=size(testing_set_ricardo);
    disp(row);
    disp(row1);
    disp(row2);
    k=1;
    for i=1:row
        testing_set{k,1}=testing_set_francisco{i,1};
        k=k+1;
    end
    for i=1:row1
        testing_set{k,1}=testing_set_gabriel{i,1};
        k=k+1;
    end
    for i=1:row2
        testing_set{k,1}=testing_set_ricardo{i,1};
        k=k+1;
    end
%     save('training_set.dat','training_set');
%     save('testing_set.dat','testing_set');

    if (exist('W.dat')==2)
```

```
        load('W.dat','-mat');
    else
    W=W_matrix(training_set,testing_set);
    end

    result=facerec(training_set,testing_set,W);
    save('result.dat','result');
    if (exist('result_superres.dat')==2)
        load('result_superres.dat','-mat');
    else result_superres=super_res(result,row,row1,row2);
        save('result_superres.dat','result_superres');
    end
    result_superres=facerec_superres(training_set,result_superres,row,row1,W);
    save('result_superres.dat','result_superres');
    result_average=average(result_superres,row,row1);
    result_average=facerec_average(training_set,result_average,row,row1,W);
    save('result_average.dat','result_average');
    result_min_of_four=min_of_four(training_set,testing_set,row,row1,W);
    save('result_min_of_four.dat','result_min_of_four');
%%%%%%%%%new part%%%%%%%%%%%%%%%%%
    load('centroid_shifted_average.dat','-mat');

result_centroid_shifted_average=facerec_centroid_shifted_average(training_set,centr
oid_shifted_average,row,row1,W);
    save('result_centroid_shifted_average.dat','result_centroid_shifted_average');
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

    display_result=zeros(row+row1+row2,1);
    display_result_superres=zeros(row+row1+row2,1);
    display_result_average=zeros(row+row1+row2,1);
    display_result_centroid_shifted_average=zeros(row+row1+row2,1);
    display_result_min_of_four=zeros(row+row1+row2,1);
    for i=[1:row-3 row+1:row+row1-3 row+row1+1:row+row1+row2-3]
        display_result_superres(i,1)=result_superres{i,10};
        display_result_average(i,1)=result_average{i,7};

display_result_centroid_shifted_average(i,1)=result_centroid_shifted_average{i,3};
        display_result_min_of_four(i,1)=result_min_of_four{i,11};
    end
    for i=1:row+row1+row2
        display_result(i,1)=result{i,3};
    end
%%%%%%%%%Class 1%%%%%%%%%%%%%%%%%
    figure;plot(1:round(row/2),display_result(1:round(row/2)),'-d');xlim([1
round(row/2)]);ylim([250 850]);xlabel('Face');ylabel('Distance');hold on;
    plot(1:round(row/2),display_result_superres(1:round(row/2)),'-sg');
    plot(1:round(row/2),display_result_average(1:round(row/2)),'-or');
    plot(1:round(row/2),display_result_centroid_shifted_average(1:round(row/2)),'-
xk');
    plot(1:round(row/2),display_result_min_of_four(1:round(row/2)),':+c');
    legend('individual','super-res','averaged faces','centroid shifted
average','min of four','Location','NW');title('Class 1');hold off;

    figure;plot(round(row/2)+1:row,display_result(round(row/2)+1:row),'-
d');xlim([round(row/2)+1 row]);ylim([250
850]);xlabel('Face');ylabel('Distance');hold on;
    plot(round(row/2)+1:row,display_result_superres(round(row/2)+1:row),'-sg');
    plot(round(row/2)+1:row,display_result_average(round(row/2)+1:row),'-or');
```

```
plot(round(row/2)+1:row,display_result_centroid_shifted_average(round(row/2)+1:row)
,'-xk');
    plot(round(row/2)+1:row,display_result_min_of_four(round(row/2)+1:row),':+c');
    legend('individual','super-res','averaged faces','centroid shifted
average','min of four','Location','N');title('Class 1');hold off;
%%%%%%%%%Class 2%%%%%%%%%%%%%%%%%

figure;plot(row+1:row+round(row1/2),display_result(row+1:row+round(row1/2),1),'-
d');xlim([row+1 row+round(row1/2)]);ylim([550
1350]);xlabel('Face');ylabel('Distance');hold on;
plot(row+1:row+round(row1/2),display_result_superres(row+1:row+round(row1/2),1),'-
sg');
plot(row+1:row+round(row1/2),display_result_average(row+1:row+round(row1/2),1),'-
or');
plot(row+1:row+round(row1/2),display_result_centroid_shifted_average(row+1:row+roun
d(row1/2),1),'-xk');
plot(row+1:row+round(row1/2),display_result_min_of_four(row+1:row+round(row1/2),1),
':+c');
legend('individual','super-res','averaged faces','centroid shifted average','min of
four','Location','NW');title('Class 2');hold off;


figure;plot(row+round(row1/2)+1:row+row1,display_result(row+round(row1/2)+1:row+row
1,1),'-d');xlim([row+round(row1/2)+1 row+row1]);ylim([550
1350]);xlabel('Face');ylabel('Distance');hold on;
plot(row+round(row1/2)+1:row+row1,display_result_superres(row+round(row1/2)+1:row+r
ow1,1),'-sg');
plot(row+round(row1/2)+1:row+row1,display_result_average(row+round(row1/2)+1:row+ro
w1,1),'-or');
plot(row+round(row1/2)+1:row+row1,display_result_centroid_shifted_average(row+round
(row1/2)+1:row+row1,1),'-xk');
plot(row+round(row1/2)+1:row+row1,display_result_min_of_four(row+round(row1/2)+1:ro
w+row1,1),':+c');
legend('individual','super-res','averaged faces','centroid shifted average','min of
four','Location','N');title('Class 2');hold off;
%%%%%%%%%Class 3%%%%%%%%%%%%%%%%%
figure;plot(row+row1+1:row+row1+round(row2/2),display_result(row+row1+1:row+row1+ro
und(row2/2),1),'-d');xlim([row+row1+1 row+row1+round(row2/2)]);ylim([350
1050]);xlabel('Face');ylabel('Distance');hold on;
plot(row+row1+1:row+row1+round(row2/2),display_result_superres(row+row1+1:row+row1+
round(row2/2),1),'-sg');
plot(row+row1+1:row+row1+round(row2/2),display_result_average(row+row1+1:row+row1+r
ound(row2/2),1),'-or');
plot(row+row1+1:row+row1+round(row2/2),display_result_centroid_shifted_average(row+
row1+1:row+row1+round(row2/2),1),'-xk');
plot(row+row1+1:row+row1+round(row2/2),display_result_min_of_four(row+row1+1:row+ro
w1+round(row2/2),1),':+c');
legend('individual','super-res','averaged faces','centroid shifted average','min of
four','Location','NE');title('Class 3');hold off;


figure;plot(row+row1+round(row2/2)+1:row+row1+row2,display_result(row+row1+round(ro
w2/2)+1:row+row1+row2,1),'-d');xlim([row+row1+round(row2/2)+1
row+row1+row2]);ylim([350 1050]);xlabel('Face');ylabel('Distance');hold on;
plot(row+row1+round(row2/2)+1:row+row1+row2,display_result_superres(row+row1+round(
row2/2)+1:row+row1+row2,1),'-sg');
```

```
plot(row+row1+round(row2/2)+1:row+row1+row2,display_result_average(row+row1+round(r
ow2/2)+1:row+row1+row2,1),'-or');
plot(row+row1+round(row2/2)+1:row+row1+row2,display_result_centroid_shifted_average
(row+row1+round(row2/2)+1:row+row1+row2,1),'-xk');
plot(row+row1+round(row2/2)+1:row+row1+row2,display_result_min_of_four(row+row1+rou
nd(row2/2)+1:row+row1+row2,1),':+c');
    legend('individual','super-res','averaged faces','centroid shifted
average','min of four','Location','NE');title('Class 3');hold off;

    compare_all=zeros(353,8);
    for i=[1:row-3 row+1:row+row1-3 row+row1+1:row+row1+row2-3]
        compare_all(i,1)=result_superres{i,10};
        compare_all(i,3)=result_average{i,7};
        compare_all(i,5)=result_centroid_shifted_average{i,3};
        compare_all(i,7)=result_min_of_four{i,11};
    end
    avg=0;min_4=0;super=0;centroid=0;
    for i=[1:row-3 row+1:row+row1-3 row+row1+1:row+row1+row2-3]
        temp=compare_all(i,1:2:7);
        [x,y]=min(temp);
        switch (y)
        case 1
            super=super+1;
        case 2
            avg=avg+1;
        case 3
            centroid=centroid+1;
        case 4
            min_4=min_4+1;
        end
    end
    compare_all(1,2)=super;
    compare_all(1,4)=avg;
    compare_all(1,6)=centroid;
    compare_all(1,8)=min_4;
end
```

This portion of the code shows the cropping algorithm. The cropping algorithm takes an image and converts the face form RGB to YCbCr and segments the face. The remaining mask is enhanced through morphology and finally cropped to keep only the face portion and returns the cropped face back to the main algorithm.

```
function [cropped_image]=cropping(X)

YCBCR=rgb2ycbcr(X);
[row col]=size(YCBCR(:,:,1));
for i=1:row
    for j=1:col
        if (YCBCR(i,j,2)>=77 && YCBCR(i,j,2)<=127)
            if (YCBCR(i,j,3)>=133 && YCBCR(i,j,3)<=173)
                mask(i,j)=true;
                else mask(i,j)=false;
            end
```

```
        else mask(i,j)=false;
        end
    end
end
mask_open=bwmorph(mask,'open');
mask_new=bwmorph(mask_open,'dilate');

L=bwlabel(mask_new,8);
L1=label2rgb(L);
S = regionprops(L, 'FilledArea');
[row col]=size(S);
for i=1:row
    temp(i)=S(i,1).FilledArea;
end
region=find(temp==max(temp));
head_area=ismember(L,region);

[mask_row_col(:,1) mask_row_col(:,2)]=find(head_area==1);
[row col]=size(mask_row_col);

min_mask_row=min(mask_row_col(:,1));
max_mask_row=max(mask_row_col(:,1));
min_mask_col=min(mask_row_col(:,2));
max_mask_col=max(mask_row_col(:,2));

cropped_image=uint8(zeros(max_mask_row-min_mask_row+1,max_mask_col-
min_mask_col+1));

for i=1:row
    cropped_image(mask_row_col(i,1)-min_mask_row+1,mask_row_col(i,2)-
min_mask_col+1)=YCBCR(mask_row_col(i,1),mask_row_col(i,2),1);
end
```

This portion of the code shows the face recognition without any enhancement. The training set, testing set, projection matrix, are needed to correctly identify the faces.

```
function [result]=facerec(face_info,face_test,W)
    face=face_info{1,3};
    [row col]=size(face_test);
    max_test=row;
    for i=1:face
        temp(i,1)=face_info{i,1};
    end
    max_class=max(temp);
%%%%%%%%%begin Luigi code%%%%%%%%
    for xx=[1:max_test]
        numero_elementi_classe = zeros(1,max_class);
        pesi_database          = zeros(face-max_class,max_class);
        for i=1:face
            current_face=double(face_info{i,2});
            current_class=double(face_info{i,1});
            pesi_correnti=W'*(double(current_face));

pesi_database(:,current_class)=pesi_database(:,current_class)+pesi_correnti;
```

```
numero_elementi_classe(current_class)=numero_elementi_classe(current_class)+1;
        end
        for ii=1:(max_class)
            distances(:,ii)=pesi_database(:,ii)/numero_elementi_classe(ii);
        end

        distanze_pesi=zeros(max_class,1);
%%%%%%Nearest Neigbor Classifier%%%%%%%%
        for ii=1:(max_class)
            distanze_pesi(ii)=norm(double(W')*double(face_test{xx,1}(:))-
distances(:,ii));
        end
        [minimo_pesi,posizione_minimo_pesi]=min(distanze_pesi);
        result{xx,1}=face_test{xx,1};
        result{xx,2}=posizione_minimo_pesi;
        result{xx,3}=min(distanze_pesi);
        result{xx,4}=distanze_pesi;
    end
```

Here is the code for the face recognition using super-resolution. Besides the training set, testing set, and projection matrix, the amount of faces for classes one and two are needed to compensate for the jumps.

```
function [result_superres]=facerec_superres(face_info,result_superres,row1,row2,W)
    face=face_info{1,3};
    [row col]=size(result_superres);
    max_test=row;
    for i=1:face
        temp(i,1)=face_info{i,1};
    end
    max_class=max(temp);
%%%%%%%%%begin Luigi code%%%%%%%%
    for xx=[1:row1-3 row1+1:row1+row2-3 row1+row2+1:max_test]
        numero_elementi_classe = zeros(1,max_class);
        pesi_database          = zeros(face-max_class,max_class);
        for i=1:face
            current_face=double(face_info{i,2});
            current_class=double(face_info{i,1});
            pesi_correnti=W'*(double(current_face));

pesi_database(:,current_class)=pesi_database(:,current_class)+pesi_correnti;

numero_elementi_classe(current_class)=numero_elementi_classe(current_class)+1;
        end
        for ii=1:(max_class)
            distances(:,ii)=pesi_database(:,ii)/numero_elementi_classe(ii);
        end

        distanze_pesi=zeros(max_class,1);
        for ii=1:(max_class)
            %%%%%%Nearest Neigbor Classifier%%%%%%%%
            distanze_pesi(ii)=norm(double(W')*double(result_superres{xx,8})-
distances(:,ii));
```

```
        end
        [minimo_pesi,posizione_minimo_pesi]=min(distanze_pesi);
        result_superres{xx,9}=posizione_minimo_pesi;
        result_superres{xx,10}=min(distanze_pesi);
        result_superres{xx,11}=distanze_pesi;
    end
```

Here is the main algorithm for the manually cropped algorithm. This code is the main shell with calls to other functions to perform the face recognition algorithms and the modifications to the faces as well as other function calls similar to the automatic cropping except without the automatic cropping.

```
function
[result,result_superres,result_average,result_min_of_four,compare_all]=automated_no
_crop()
    clear all;close all;clc;
%%%%%class 1
    for k=1:210
        cd('C:\Documents and Settings\James\Desktop\Grad School\Thesis\paper
files\three classes\No cropping\francisco');
        frame=imread([num2str(k) '.tif']);
        cd('C:\Documents and Settings\James\Desktop\Grad School\Thesis\paper
files\three classes\No cropping');
        temp=rgb2ycbcr(frame);
        temp=imresize(temp(:,:,1),[56 46],'bicubic');
        francisco_info{k,1}=temp(:);
    end
%%%%%%%class 2
    for k=1:312
        cd('C:\Documents and Settings\James\Desktop\Grad School\Thesis\paper
files\three classes\No cropping\gabriel');
        frame=imread([num2str(k) '.tif']);
        cd('C:\Documents and Settings\James\Desktop\Grad School\Thesis\paper
files\three classes\No cropping');
        temp=rgb2ycbcr(frame);
        temp=imresize(temp(:,:,1),[56 46],'bicubic');
        gabriel_info{k,1}=temp(:);
    end
%%%%%class 3
    for k=1:168
        cd('C:\Documents and Settings\James\Desktop\Grad School\Thesis\paper
files\three classes\No cropping\ricardo');
        frame=imread([num2str(k) '.tif']);
        cd('C:\Documents and Settings\James\Desktop\Grad School\Thesis\paper
files\three classes\No cropping');
        temp=rgb2ycbcr(frame);
        temp=imresize(temp(:,:,1),[56 46],'bicubic');
        ricardo_info{k,1}=temp(:);
    end

    %calculate training set
%%%%%class 1
    [row col]=size(francisco_info);
    skip=round(row/10);
    k=1;
    for i=1:skip:row
```

```
        training_set{k,2}=francisco_info{i,1};
        training_set{k,1}=1;
        skipped_images_francisco(k)=i;
        k=k+1;
    end
%%%%%class 2
    [row col]=size(gabriel_info);
    [row1 col1]=size(training_set);
    skip=round(row/10);
    k=1;
    for i=1:skip:row
        training_set{k+row1,2}=gabriel_info{i,1};
        training_set{k+row1,1}=2;
        skipped_images_gabriel(k)=i;
        k=k+1;
    end
%%%%%class 3
    [row col]=size(ricardo_info);
    [row1 col1]=size(training_set);
    skip=round(row/10);
    k=1;
    for i=1:skip:row
        training_set{k+row1,2}=ricardo_info{i,1};
        training_set{k+row1,1}=3;
        skipped_images_ricardo(k)=i;
        k=k+1;
    end
%%%%%
    training_set{1,3}=k+row1-1;

    %calculate testing sets
%%%%%%%%class 1
    [row col]=size(skipped_images_francisco);
    [row1 col1]=size(francisco_info);
    i=1;
    l=1;
    while i<col
        for j=skipped_images_francisco(i)+1:skipped_images_francisco(i+1)-1
            testing_set_francisco{l,1}=francisco_info{j,1};
            l=l+1;
        end
        i=i+1;
    end
    for j=skipped_images_francisco(col)+1:row1
        testing_set_francisco{l,1}=francisco_info{j,1};
        l=l+1;
    end
    % clear row col row1 col1;
%%%%%%%%class 2
    [row col]=size(skipped_images_gabriel);
    [row1 col1]=size(gabriel_info);
    i=1;
    l=1;
    while i<col
        for j=skipped_images_gabriel(i)+1:skipped_images_gabriel(i+1)-1
            testing_set_gabriel{l,1}=gabriel_info{j,1};
            l=l+1;
        end
```

```
        i=i+1;
    end
    for j=skipped_images_gabriel(col)+1:row1
        testing_set_gabriel{l,1}=gabriel_info{j,1};
        l=l+1;
    end
%%%%%%%%class 3
    [row col]=size(skipped_images_ricardo);
    [row1 col1]=size(ricardo_info);
    i=1;
    l=1;
    while i<col
        for j=skipped_images_ricardo(i)+1:skipped_images_ricardo(i+1)-1
            testing_set_ricardo{l,1}=ricardo_info{j,1};
            l=l+1;
        end
        i=i+1;
    end
    for j=skipped_images_ricardo(col)+1:row1
        testing_set_ricardo{l,1}=ricardo_info{j,1};
        l=l+1;
    end
%%%%%%%%

    [row col]=size(testing_set_francisco);
    [row1 col1]=size(testing_set_gabriel);
    [row2 col2]=size(testing_set_ricardo);
    k=1;
    for i=1:row
        testing_set{k,1}=testing_set_francisco{i,1};
        k=k+1;
    end
    for i=1:row1
        testing_set{k,1}=testing_set_gabriel{i,1};
        k=k+1;
    end
    for i=1:row2
        testing_set{k,1}=testing_set_ricardo{i,1};
        k=k+1;
    end
    save('training_set.dat','training_set');
    save('testing_set.dat','testing_set');

    if (exist('W.dat')==2)
        load('W.dat','-mat');
    else
    W=W_matrix(training_set,testing_set);
    end

    result=facerec(training_set,testing_set,W);
    save('result.dat','result');
    if (exist('result_superres.dat')==2)
        load('result_superres.dat','-mat');
    else result_superres=super_res(result,row,row1,row2);
        save('result_superres.dat','result_superres');
    end
    result_superres=facerec_superres(training_set,result_superres,row,row1,W);
    save('result_superres.dat','result_superres');
```

```
    result_average=average(result_superres,row,row1);
    result_average=facerec_average(training_set,result_average,row,row1,W);
    save('result_average.dat','result_average');
    result_min_of_four=min_of_four(training_set,testing_set,row,row1,W);
    save('result_min_of_four.dat','result_min_of_four');

    display_result=zeros(row+row1+row2,1);
    display_result_superres=zeros(row+row1+row2,1);
    display_result_average=zeros(row+row1+row2,1);
    display_result_min_of_four=zeros(row+row1+row2,1);
    for i=[1:row-3 row+1:row+row1-3 row+row1+1:row+row1+row2-3]
        display_result_superres(i,1)=result_superres{i,10};
        display_result_average(i,1)=result_average{i,7};
        display_result_min_of_four(i,1)=result_min_of_four{i,11};
    end
    for i=1:row+row1+row2
        display_result(i,1)=result{i,3};
    end
%%%%%%%%%Class 1%%%%%%%%%%%%%%%%%
figure;plot(1:round(row/2),display_result(1:round(row/2)),'-d');xlim([1
round(row/2)]);ylim([240 600]);xlabel('Face');ylabel('Distance');hold on;
plot(1:round(row/2),display_result_superres(1:round(row/2)),'-sg');
plot(1:round(row/2),display_result_average(1:round(row/2)),'-or');
plot(1:round(row/2),display_result_min_of_four(1:round(row/2)),':+c');
legend('individual','super-res','averaged faces','min of
four','Location','NW');title('Class 1');hold off;

figure;plot(round(row/2)+1:row,display_result(round(row/2)+1:row),'-
d');xlim([round(row/2)+1 row]);ylim([240
600]);xlabel('Face');ylabel('Distance');hold on;
plot(round(row/2)+1:row,display_result_superres(round(row/2)+1:row),'-sg');
plot(round(row/2)+1:row,display_result_average(round(row/2)+1:row),'-or');
plot(round(row/2)+1:row,display_result_min_of_four(round(row/2)+1:row),':+c');
legend('individual','super-res','averaged faces','min of
four','Location','NW');title('Class 1');hold off;
%%%%%%%%%Class 2%%%%%%%%%%%%%%%%%
figure;plot(row+1:row+round(row1/2),display_result(row+1:row+round(row1/2),1),'-
d');xlim([row+1 row+round(row1/2)]);ylim([550
1200]);xlabel('Face');ylabel('Distance');hold on;
plot(row+1:row+round(row1/2),display_result_superres(row+1:row+round(row1/2),1),'-
sg');
plot(row+1:row+round(row1/2),display_result_average(row+1:row+round(row1/2),1),'-
or');
plot(row+1:row+round(row1/2),display_result_min_of_four(row+1:row+round(row1/2),1),
':+c');
legend('individual','super-res','averaged faces','min of
four','Location','NW');title('Class 2');hold off;

figure;plot(row+round(row1/2)+1:row+row1,display_result(row+round(row1/2)+1:row+row
1,1),'-d');xlim([row+round(row1/2)+1 row+row1]);ylim([550
1200]);xlabel('Face');ylabel('Distance');hold on;
plot(row+round(row1/2)+1:row+row1,display_result_superres(row+round(row1/2)+1:row+r
ow1,1),'-sg');
plot(row+round(row1/2)+1:row+row1,display_result_average(row+round(row1/2)+1:row+ro
w1,1),'-or');
plot(row+round(row1/2)+1:row+row1,display_result_min_of_four(row+round(row1/2)+1:ro
w+row1,1),':+c');
```

```
legend('individual','super-res','averaged faces','min of
four','Location','NE');title('Class 2');hold off;
%%%%%%%%%Class 3%%%%%%%%%%%%%%%%%
figure;plot(row+row1+1:row+row1+round(row2/2),display_result(row+row1+1:row+row1+ro
und(row2/2),1),'-d');xlim([row+row1+1 row+row1+round(row2/2)]);ylim([350
950]);xlabel('Face');ylabel('Distance');hold on;
plot(row+row1+1:row+row1+round(row2/2),display_result_superres(row+row1+1:row+row1+
round(row2/2),1),'-sg');
plot(row+row1+1:row+row1+round(row2/2),display_result_average(row+row1+1:row+row1+r
ound(row2/2),1),'-or');
plot(row+row1+1:row+row1+round(row2/2),display_result_min_of_four(row+row1+1:row+ro
w1+round(row2/2),1),':+c');
legend('individual','super-res','averaged faces','min of
four','Location','NW');title('Class 3');hold off;

figure;plot(row+row1+round(row2/2)+1:row+row1+row2,display_result(row+row1+round(ro
w2/2)+1:row+row1+row2,1),'-d');xlim([row+row1+round(row2/2)+1
row+row1+row2]);ylim([350 950]);xlabel('Face');ylabel('Distance');hold on;
plot(row+row1+round(row2/2)+1:row+row1+row2,display_result_superres(row+row1+round(
row2/2)+1:row+row1+row2,1),'-sg');
plot(row+row1+round(row2/2)+1:row+row1+row2,display_result_average(row+row1+round(r
ow2/2)+1:row+row1+row2,1),'-or');
plot(row+row1+round(row2/2)+1:row+row1+row2,display_result_min_of_four(row+row1+rou
nd(row2/2)+1:row+row1+row2,1),':+c');
legend('individual','super-res','averaged faces','min of
four','Location','NW');title('Class 3');hold off;

    compare_all=zeros(353,6);
    for i=[1:row-3 row+1:row+row1-3 row+row1+1:row+row1+row2-3]
        compare_all(i,1)=result_superres{i,10};
        compare_all(i,3)=result_average{i,7};
        compare_all(i,5)=result_min_of_four{i,11};
    end
    tie=0;avg=0;min_4=0;super=0;
    for i=[1:row-3 row+1:row+row1-3 row+row1+1:row+row1+row2-3]
        temp=compare_all(i,1:2:5);
        [x,y]=min(temp);
        switch (y)
        case 1
            super=super+1;
        case 2
            avg=avg+1;
        case 3
            min_4=min_4+1;
        end
    end
    compare_all(1,2)=super;
    compare_all(1,4)=avg;
    compare_all(1,6)=min_4;
end
```

# Curriculum Vita

James Roger Roeder was born on February 7th, 1985 in El Paso, Texas. He is the first son of Robert Roeder and Elva Roeder. He received his Bachelor's degree from The University of Texas at El Paso (UTEP) in the spring of 2007. He continued his studies as a graduate student at UTEP and was accepted into the Wireless Sensor Networks Group, by his advisor Dr. Sergio Cabrera. During this time, he worked as a research assistant working on improving face recognition by using super-resolution. He published a paper at the SPIE Defense, Security, and Sensing conference titled "Assessment of super-resolution for face recognition from very-low resolution images in sensor networks" in April 2009. He obtained his Master's degree from the University of Texas at El Paso in the spring of 2009.

Permanent address: 11721 Gwen Evans
El Paso, TX, 79935

This thesis/dissertation was typed by James Roger Roeder.