2015

Validating the Representativeness of Test Samples for Performance Prediction of Face Recognition

Jason Hooks
West Virginia University

Follow this and additional works at: <https://researchrepository.wvu.edu/etd>

Part of the [Computer Engineering Commons](#)

Recommended Citation

Hooks, Jason, "Validating the Representativeness of Test Samples for Performance Prediction of Face Recognition" (2015). *Graduate Theses, Dissertations, and Problem Reports*. 8154.
<https://researchrepository.wvu.edu/etd/8154>

This Thesis is protected by copyright and/or related rights. It has been brought to you by the The Research Repository @ WVU with permission from the rights-holder(s). You are free to use this Thesis in any way that is permitted by the copyright and related rights legislation that applies to your use. For other uses you must obtain permission from the rights-holder(s) directly, unless additional rights are indicated by a Creative Commons license in the record and/ or on the work itself. This Thesis has been accepted for inclusion in WVU Graduate Theses, Dissertations, and Problem Reports collection by an authorized administrator of The Research Repository @ WVU. For more information, please contact researchrepository@mail.wvu.edu.

Validating the Representativeness of Test Samples for Performance Prediction of Face Recognition

Jason Hooks

Thesis submitted to the College of Engineering and Mineral Resources at West Virginia
University

In partial fulfillment of the requirements for the degree of
Master of Science in Computer Science

Bojan Cukic PhD, Chair
Jeremy Dawson PhD
Donald Adjero PhD

Lane Department of Computer Science and Electrical Engineering

Morgantown, West Virginia
2015

Keywords: Biometrics, Facial Recognition, Performance Prediction, Sequestered Datasets,
Subset Generation, ROC curve emulation

Copyright 2015 Jason Hooks

ABSTRACT

Validating the Representativeness of Test Samples for Performance Prediction of Face Recognition

Jason Hooks

One of the reasons testing biometric systems is difficult lays in the fact that the test sample available during technology evaluation may not be sufficiently similar to the usage profile that the system will encounter in operations. As the result, performance expectations derived from testing prior to system deployment may not match actual performance after the deployment. A specific instance of this problem occurs when the data from the field in which biometric system will be deployed is sequestered.

In this study, we simulated the stated scenario using two datasets, originally claimed to be similar and adequate for performance prediction; dataset A, assembled by RAND Corporation and the “sequestered” dataset B, collected in a multi-year project at West Virginia University, sponsored by the FBI. Our objective was to select biometric samples from A that yield match scores that best represent the performance of B, thus enabling accurate prediction.

We developed two groups of approaches. In the first, we select biometric samples from data set A randomly. The subset of samples with overall match score distributions most similar to data set B then represent the best test sample. We use the Probability Mass Function to make distributions based on percentage of score vectors. Kullback-Leibler Divergence, a statistical method, measures the similarity of Probability Mass Functions. The second group of approaches adds carefully chosen weights to match scores. The samples are added into a sorted list by weight, and a number of them are picked to make a test sub set most similar. ROC curve is used to measure how well test performance over subsets of data set A compare to data set B. Weighted selection approaches allow us to construct test data sets that successfully predict match performance of the sequestered biometric data set.

ACKNOWLEDGEMENTS

I would like to thank Dr. Cukic for being my research advisor throughout my studies here at WVU. Thanks to Dr. Adjero and Dr. Dawson for being on my committee. Thanks to Mounica Kamireddy for generating the FBI face data. Thanks to James Searls for implementing the KL divergence matlab code and for the idea of generating weights for subjects. I would like to thank Mark Culp and Kenneth Ryan for their statistical guidance. I would like to thank Karla Buckel for delivering the RAND dataset to us. I would like to thank Cameron Whitelam for working with me when I first started doing research.

I would also like to thank my mom, dad, brother and sisters – Mary Lee, Steve, John, Jenna, and Caroline Hooks, for their love and support.

Table of Contents

ABSTRACT.....	II
ACKNOWLEDGEMENTS.....	III
Table of Contents.....	IV
List of Figures.....	V
1. Use Case for this work	1
2. Introduction to Face Recognition:	1
3. Related Work in Performance Prediction:	9
4. Software Components of the Analysis:.....	14
5. Analysis Procedure/Methodology:	17
5.1 Probability Mass Function.....	18
5.2 Kullback-Leibler Divergence measure.....	19
5.3 Making Subsets	20
5.3.1 Exchange	22
5.3.2 Exchange Group	24
5.3.3 Weighted Group.....	26
5.3.4 Weighted Pair.....	29
5.3.5 Weighted Group Single	32
5.3.6 – Finding the optimal number of templates automatically	34
5.4 – Looking at Performance and Comparison Tests	34
5.5 - Time taken to run each algorithm	45
6. Explanation of results for FBI and RAND:	46
7. Match Score and Quality Score Correlation:	48
8. Conclusion.....	50
9. Future Work	51
Bibliography:	56

Table of Figures

Figure 1 - Standard Eigenfaces.....	4
Figure 2 - Example of six classes using LDA [12]	5
Figure 3 - PCA vs LDA layout [15]	5
Figure 4 - Elastic Graph Bunch Matching.....	6
Figure 5 - Comparison of PCA, LDA, EBGM [20].....	7
Figure 6 - Comparison between LBP and PCA [21]	7
Figure 7 - Face Recognition Algorithms [19].....	8
Figure 8 - Finding the optimal gallery size	11
Figure 9 - Comparison – Digital Camera vs Webcam	13
Figure 10 - Probability Mass Function [9]	18
Figure 11 - RAND and FBI pmfs	21
Figure 12 - RAND and FBI quality pmfs	22
Figure 13 - Exchange, RAND, and FBI pmfs.....	23
Figure 14 - Exchange, RAND, and FBI - Quality pmfs	24
Figure 15 - Exchange Group, RAND, and FBI pmfs.....	25
Figure 16 - Exchange Group, RAND, and FBI - Quality pmfs	26
Figure 17 - Weighted Group, RAND, and FBI pmfs	28
Figure 18 - Weighted Group, RAND, and FBI – Quality pmfs.....	28
Figure 19 - Weighted Pairs, RAND, and FBI pmfs.....	31
Figure 20 - Weighted Pairs, RAND, and FBI - Quality pmfs.....	32
Figure 21 - Weighted Group Single, RAND, and FBI pmfs.....	33
Figure 22 - Weighted Group Single, RAND, and FBI - Quality pmfs	33
Figure 23 - 25 points RAND, FBI ROC	36
Figure 24 - 500 Points RAND, FBI ROC	36
Figure 25 - 4722 points RAND, FBI ROC	37
Figure 26 - 4722 points RAND, 5730 points FBI ROC	38
Figure 27 - Exchange Group Subset vs FBI ROC	40
Figure 28 - Exchange Subset vs FBI ROC	41
Figure 29 - Weighted Group Subset vs FBI ROC.....	42
Figure 30 - Weighted Group Single vs FBI.....	43
Figure 31 - Weighted Pairs Subset vs FBI ROC	44
Figure 32 - Quality versus Match Score - Genuine	49
Figure 33 - Quality versus Match Score - Impostor	50

1. Use Case for this work

This work could be used in cases where there is a dataset that is not allowed to be released to an outside entity, and consequently, the outside entity wants something that performs in a similar manner. We can pick subjects from another dataset that is allowed to be released, to make a subset that performs close to the way of the dataset that is not allowed to be released. This subset can then be given to the outside entity to be used in place of the other dataset.

2. Introduction to Face Recognition:

Security has always been a paramount concern in computer systems. People do not want certain information or privileges belonging to them to be used, viewed, or stolen by another person. While traditional computer systems have long used passwords/PINs (something you know) and/or identification cards (something you possess), a need for a newer system based on who you are intrinsically, has arisen.

Biometric recognition is the automatic recognition of individuals based on their physical or behavioral characteristics. Humans have used properties such as face, voice, gait, etc for thousands of years to recognize one another. In the late 19th century, many law enforcement agencies began to “book” the fingerprints of criminals and store them in a database. Though biometrics emerged from use in law enforcement applications, today it is also being used in a number of civilian applications. Today biometric systems exist in a variety of modalities

including face, voice, iris, fingerprint, DNA, Ear, Gait, Hand geometry, hand vein, retina, odor, keystroke, palm print, and signature.

A typical biometric system operates in either verification or identification mode. In verification mode, captured biometric data (from a scanner or camera) is compared against an existing template stored in the database. In verification mode, the person uses a PIN, username, smart card to claim that they are a particular person. After this claim, they must scan one of their physical characteristics to verify that they are who they claim to be. The comparison is one-to-one.

In identification mode, the system recognizes an individual by searching the templates of every user in the database for a match. The system performs a one-to-many search on the database. The search succeeds if it finds the user in the database, or fails if the person is not enrolled in the database. Identification can be used to establish that a person is someone they deny to be (negative identification), or used to automatically recognize someone without claiming to be a specific person (positive recognition), for the sake of convenience. For purposes of this paper, we will be using the identification mode of a matching system, because as a whole it executes in less time than verification.

Face recognition is non-intrusive, and is the most common characteristic by which humans identify each other. [1] Many biometric modalities require the cooperation of the person being examined. For example, fingerprints require a person's finger to be pressed on a sensor; iris requires a person to open their eyes widely in front of a scanner; and voice requires a person to speak a phrase. Face images have an advantage, in that they can be captured from a distance without touching the person and do not require interacting with the person. [11] Face images also have the advantage of capturing the emotion of the person when their picture is taken. Many

people are willing to share their face in the public domain (e.g. Facebook), but not other traits like fingerprint and iris. Possible inputs to a face recognition system include conventional 2D photographs, 3D range or depth images, and video. [18]

There are three widely used implementations of biometric face recognition: appearance based, feature (also referred to as model), and texture based. [19] Feature-based uses the location and shape of facial attributes such as eyes, eyebrows, nose, lips, and chin and the spatial relationships between them. [16, 12] The first semi-automated facial recognition technique required an administrator to actually mark the location of features (e.g. eyes, ears, nose, and mouth) before calculating distances and ratios between them. In the 1970s, Goldstein, Harmon, and Lesk developed a method using 21 subjective markers, including hair color and lip thickness, to automate the recognition. The problem with these early solutions was that some aspects still needed to be manually computed.

In 1988, Kirby and Sirovich applied Principle component analysis, a statistical procedure, to the face recognition problem. Information that is not useful is removed, and images are transformed into orthogonal components known as eigenfaces.[12] The goal with eigenfaces is to have an intra-face distance (between images of the same person) that is lower than the inter-face distance (between images of different people), but there is no guarantee by the algorithm that this will be true. [17] In 1991, Turk and Pentland discovered while using the eigenfaces technique that residual error could be used to detect a face in an image. This enabled reliable real-time automated face recognition systems, for the first time.

Figure 1- Standard Eigenfaces

PCA (Principal Component Analysis) is an appearance based model that turns images into feature vectors with a 1000:1 reduction in size. Each face image can be represented as a weighted sum of the eigenfaces. PCA uses a gallery of training images from various subjects. Once the probe image feature vector and a gallery image feature vector are created, the distance between the two is measured to see if they are a match. The gallery images with the least distance from the probe image will be chosen. PCA requires the facial images to be frontal, otherwise it performs poorly. [12] PCA is a powerful way to represent faces, because it ensures data variance is maintained while having no existing correlations among the original features. [15] After the invention of PCA, a number of other face recognition algorithms were developed.

Linear Discriminant Analysis (LDA), another appearance based facial recognition algorithm, works by classifying unknown samples based on training samples of known classes. [16, 12] LDA aims to maximize the between user variation and minimize within user variation. LDA is best utilized when the goal is classification rather than representation. An example of a class is a subject (a group of information about an individual).

Figure 2 - Example of six classes using LDA [12]

LDA outperforms PCA in class discrimination. To illustrate the difference in class discrimination, see the figure below:

Figure 3 - PCA vs LDA layout [15]

One drawback of LDA is the small sample size problem. This occurs whenever the number of samples is smaller than the dimensionality of the samples. [14] LDA outperforms PCA when the sample size is large. PCA outperforms LDA when the sample size is small. [13]

Elastic Bunch Graph Matching (EBGM), a feature based algorithm, accounts for the fact that faces have many non-linear characteristics that are not addressed by the previously mentioned algorithms. [16, 12] These characteristics are variations in illumination (outdoor

lighting vs indoor fluorescents), pose (standing straight up vs leaning over) and facial expression (smile vs neutral). A Gabor wavelet transform is used to project the face onto an elastic grid. Nodes on the grid, shown as circles, describe the image behavior around a given pixel. The nodes are known as Gabor Jets and are the result of running a convolution of a Gabor filter on the image, which is used to detect features and extract shapes. Recognition is implemented by checking the Gabor filter response at each node. The challenging part of implementing this method is the requirement of accurate landmark localization.

Figure 4 - Elastic Graph Bunch Matching

EBGM has the highest accuracy and the slowest computational time out of the previous mentioned methods. PCA has the lowest accuracy and the fastest computational time. LDA is somewhere in the middle on both measures. [20]

TABLE I
TEST RESULT FOR DATABASE ASU: ACCURACY RATE AND
COMPUTATIONAL TIME

	Accuracy rate (%)	Computational time (s)
PCA	66.67	1.638
LDA	84.44	1.891
EBGM	97.78	1251

TABLE II
TEST RESULT FOR DATABASE FERET: ACCURACY RATE AND
COMPUTATIONAL TIME

	Accuracy rate (%)	Computational time (s)
PCA	57.00	5.238
LDA	64.00	5.548
EBGM	76.00	3422

Figure 5 - Comparison of PCA, LDA, EBGM [20]

The final type of face recognition algorithm is texture based. Texture based algorithms look for robust local features that are not affected by posing or lighting variations. Two such algorithms are Local Binary Pattern and Gradient Orientation. Local Binary Pattern is generally faster than Principal Component Analysis, but it is also less accurate.

LBP	PCA
Less accurate	More accurate
When identifying, it sometimes returns the wrong face	Gives the correct face even if the facial expression varies
Fast recognition speed	Slower recognition speed

Figure 6 - Comparison between LBP and PCA [21]

Below is a listing of the most common facial recognition algorithms:

Figure 7 - Face Recognition Algorithms [19]

For any algorithm or matching system, accommodations should be made for race, gender, age as they can significantly affect performance.

Face recognition is used today to fight passport fraud, support law enforcement, identify missing children, minimize benefit/identity fraud, and control access to facilities. [12]

Since face recognition in a standard setting with multiple objects and people is very difficult, biometric matchers and datasets typically restrict one or more aspects. These aspects include the environment where the recognition system will take place (fixed lighting, fixed location, uniform background, single face, etc), the allowable face change (neutral expression, negligible aging, etc), the number of individuals in the image, and the viewing condition (frontal view, profile view, occlusion, no occlusion, etc). [2]

The ideal biometrics have five characteristics: robustness, distinctiveness, availability, accessibility, and acceptability. To evaluate performance of biometric systems, we typically use

False Alarm Rate (FAR), False Reject Rate (FRR), and Hit Rate. False alarms are when a match between two separate subjects yields a score higher than the threshold. False alarm rate is the percent of the time this happens for a particular threshold. False Rejects are when a match between two templates of the same subject yields a score lower than the threshold. False reject rate is the percent of the time this happens for a particular threshold. Hit Rate is defined as: 1 minus the False Reject Rate (FRR). The FAR and Hit Rates vary depending on the system threshold. The system threshold is a discrete variable. A match score below the threshold is considered to be a non-match. A match score above the threshold is considered to be a match. The Receiver Operating Curve (ROC) is a plot of the FAR versus the hit rate (GAR), with each point at a different threshold. The area under the ROC curve is proportional to the system performance. The ideal biometrics system has an area under the ROC curve that is near 1.0. [3]

3. Related Work in Performance Prediction:

Performance Prediction for Biometrics Recognition Systems by Rong Wang presents a binomial model to predict large population performance from a small gallery. Wang's method uses the genuine and impostor matches from small gallery to predict a larger population. This work goes over finding the optimum small gallery size, which is made of the images used to model the performance of a larger population. Error is measured by the difference between the predicted large population performance and the actual performance of an experimental population. Wang's work also covers optimizing sensor fusion combinations. The small gallery starts with 50 fingerprints/subjects, and as the quantity of small gallery fingerprints/subjects increases (e.g. 70, 120), the error decreases. $P(N, 1, t)$ is the prediction model with a threshold of

12. $P(N, 1)$ is the prediction model without a threshold. The error contributed by the prediction model $P(N, 1, t)$ is much smaller than the error contributed by $P(N, 1)$. Beyond a certain population size, the error decreases. The dataset consists of 2000 pairs of fingerprints.

To find the optimal gallery size an initial size of 50 samples is utilized. The expectation maximization algorithm is used to estimate the large population performance. The error between the prediction performance and actual performance is computed. If the error is greater than the acceptable margin of error, then the small gallery size is increased and the previous steps are run again. Otherwise, the large population size is increased, and the previous steps are repeated until the large population size is equal to the total amount of subjects/fingerprints. The Chernoff and Chebychev inequalities are used, given a margin of error, to find the relationship between small gallery size and the prediction confidence interval. Figure 8 below illustrates the concept:

Figure 8 - Finding the optimal gallery size

Face Recognition Performance: Role of Demographic Information

The performance of face recognition biometric algorithms can be biased on gender, age, and ethnic group participation in the sample. This paper presents results from different face recognition algorithms (Commercial, Trainable, Non-Trainable). The female, African American, and younger cohorts are generally more difficult to recognize. One contributing factor for females being more difficult to recognize is the use of makeup/cosmetics. Recognition generally improves when training exclusively on a cohort. It is observed that particular biases exist in commercial matchers, but it is not known why, because they are black boxes. To discover why, non-commercial trainable and non-trainable face recognition algorithms are used. If a trainable face recognition algorithm is trained on one cohort and tested on another, it will not work as well.

The Multi-Scenario Multi- Environment BioSecure Multimodal Database (BMDB)

This study is a comparison of multiple multi-modal biometric databases. It is shown in this paper that using higher quality face images results in better performance. Figure 9 below shows that digital camera images (high quality) perform better than webcam images (low quality):

Figure 9 - Comparison – Digital Camera vs Webcam

Chen, Y.; Dass, S.; Jain, A., “Fingerprint Quality Indices for Predicting Authentication Performance” – The performance of an automatic fingerprint authentication system relies heavily on the quality of the captured fingerprint image. [6] Fingerprint scans consist of minutiae (island, dot, bifurcation, and ridge ending) and singularity points (loop, delta, and whorl). [7] Fingerprint quality is dependent on the measure of the ridge and valley structures in the fingerprint examined, and also the extractability of these features (minutiae and singularity points). Poor quality fingerprint images increase false reject rate and false accept rate. This paper proposes two new quality metrics for fingerprints. One metric is in the frequency domain, and the other is in the spatial domain.[6]

4. Software Components of the Analysis:

Dupeguru Picture Edition

This is a free piece of software that can scan a directory of images and find any duplicate image files by examining the underlying image data. After looking at the RAND face scores, it was found that there were many scores above 9000. High scores like this can be caused by duplicate image pairs. After inspecting these high score pairs, it was found that they belonged to duplicate images. Dupeguru was used to remove the duplicate image files from the RAND face data set. Dupeguru works with a variety of picture formats and is available for Mac, Windows, and Linux. The software is available at <http://www.hardcoded.net/dupeguru/>.

Neurotec Biometric 5.0 SDK

Neurotec Biometric is the SDK that was used for matching. This is a commercial product and must be purchased. The SDK takes in an image file as input, and either outputs a template or rejects the image. The SDK can perform a variety of functions (template creation, enrollment, identification, verification, etc) with different biometric data including generating quality scores, running identification, and running verification. The SDK works with voice, finger, face, iris, and palm print. The SDK is also capable of fusing one or more of these modalities into a subject. The MegaMatcher SDK allows for 360 degrees of head roll, 15 degrees of pitch in direction from the frontal position, and 45 degrees of yaw in each direction from the frontal position. Also, the SDK has the ability to identify different facial expressions such as smiles, open mouth, closed eyes, and determine if an individual is wearing glasses. [23]The SDK can be found online at <http://www.neurotechnology.com/megamatcher.html>.

This Study's Custom SDK Module

This study uses a custom SDK module that interacts with the Neurotec Biometric SDK. The identification and verification modules of the Neurotec Biometric SDK were evaluated, and the identification module was shown to take less time to accomplish the same tasks. This study attempts to create a template for each face image. A template is either created, or an exception is thrown and it is skipped. All templates are put into a gallery. The first template is removed from the gallery to be used as the probe. Matches are generated from this process using the identity module of the SDK.

Data output from this analysis includes match scores, quality scores, and a flag for whether matches are genuine or impostor (evaluated by a subject id equality check). This process is performed again by removing the second template from the gallery and using it as the probe. This process is repeated until there is only one image left in the gallery. ID_Probe, Quality_Probe, ID_Gallery, Quality_Gallery, Match_Score, and Genuine columns and data are written to a csv file for further analysis. This module is written in Java.

Matlab

Matlab reads in csv files that were generated from our custom sdk module. This data is organized through a number of separate scripts. Matlab performs the analysis and generates graphs of the data.

All components for this study are run on DELL PRECISION T7610 systems. These systems contain 64GB of ram, an Intel Xeon CPU E5-2630 v2 @ 2.60GHz (2 processors), and are running Windows7.

Study Datasets

The RAND face dataset contains 31,771 images. After eliminating duplicates, there are 24,071 images. After feeding the 24,071 images into the SDK module, there are 8,757 templates created and 1899 valid unique subjects. There are 21,951 genuine matches and 38,316,195 impostor matches. The minimum match score of the RAND dataset is 0, and the maximum match score of the RAND dataset is 4721.

There are 13,976 images in the FBI dataset. There are 13,830 templates created from the SDK module for the FBI face dataset. There are 1,199 valid unique subjects. There are 26,555,173 impostor matches and 165,080 genuine matches. The minimum match score of the FBI dataset is 0 and the maximum match score of the FBI dataset is 5730.

5. Analysis Procedure/Methodology:

The most obvious way to make a subset of dataset A that represents B is to run brute force. To do this one must go through every possible template combination of A and pick the combination that is the best the representation of B. There are so many possible combinations that this would take many years to finish on the top computer systems. As an example, if we had 8757 possible templates in A, and went through all combinations of 2000 (8757 C 2000), the number of possible combinations would be $2.722 * 10^{2068}$.

Since brute force is not an option at this point, we will instead use several different algorithms to make the best subset of dataset A that we can. We will cover each algorithm we use, the advantages and disadvantages of it, and the results of using the algorithm on our datasets.

The first step of the workflow employed in this study is to run Dupeguru on the files in the dataset. Then the custom SDK module is run, specifying the correct input directory and modality (face, iris, or finger) to execute the score output process.

The output from the Custom SDK module is a collection of match scores, quality scores, ground truth status, and filename strings. Each entry is made from a pair of two templates. An example of a filename string from RAND is "2162_5f4fdc0a-4b3b-4edc-a228-aa0697daa65c.10.17.jpg ". The first part of this filename before the underscore - "2162" is the subject id. The subject id is a unique number identifying the actual person pictured in the image. By comparing the subject ids of the probe and gallery strings, the ground truth of the match can be determined. The ground truth specifies whether both parts of the pair belong to the same person, or if they are of different people. If the probe and gallery images belong to the same person, it is designated as genuine. Otherwise, it is designated as an impostor.

All match scores will be classified as genuine or impostor matches. Subjects will be picked by relying on the genuine distributions. The impostor distributions for two separate biometric databases are usually pretty similar, so they only need to be examined in the final step of the analysis.

5.1 Probability Mass Function

Both match scores and quality scores are discrete values. A probability mass function is run on the datasets. This is a fundamental part of the comparison. The probability mass function works by iterating over a vector, and calculating the probability at each index. The probability mass function (PMF) tells us the probability of each match score or quality score. [8]

Figure 10 depicts an example of output from a probability mass function:

Figure 10 - Probability Mass Function [9]

In this study, each data point is a match score. The probability of each data point is calculated as:

X / Y

Where

X = the number of occurrences of each score

Y = the total number of score across the whole set

Summing all of the probabilities (X / Y) on a correctly distributed vector from a PMF yields a value of 1.0. [8]

5.2 Kullback-Leibler Divergence measure

As a baseline to compare two probability mass distributions, a slightly modified version of the Kullback-Leibler divergence measure is used. The Kullback-Leibler (KL) divergence measure gives us a non-symmetric way to compare two PMF vectors and output a distance score. Since it is non-symmetric, reversing the order of the inputs will yield different results. That is P, Q will be different from Q, P .

The code to implement the comparison is as follows:

```
for i = 1:length(P)

    if P(i) ~= 0 && Q(i) ~= 0

        dist = dist + P(i) * abs(log(P(i)) - log(Q(i)));

    end

end
```

A comparison is only made at a data point if both P and Q are non-zero. Without this protection, one or more of the numbers (P, Q or Q, P) would be negative infinity, and this would render the analysis unusable. This means if there are no non-zero data points in the subset that match dataset B, the score (distance) will be 0.0.

Before running the KL divergence test, the min and max of our vectors are examined. If needed, zeroes are padded onto the A distribution (or the subset) to match the sequestered database B's distribution. It is very important to do this, because without this step, the KLD.m code will crash.

For this study, the Kullback-Leibler divergence test is run on dataset A (or a subset of it) and dataset B. A score as close to 0.0 as possible is desirable. The lower the score, the better. The Kullback-Leibler divergence test is run in both directions (P, Q) and (Q, P) and the outputs are averaged. [10]

5.3 Making Subsets

In this section, methods for making subsets of dataset A will be discussed. The subset distributions will be reviewed in order of worst to best (based on our FBI and RAND datasets). These methods were found after much experimentation with the data.

The probability mass distributions of RAND and FBI are graphed in Figure 11 below:

Figure 11 - RAND and FBI pmfs

The Kullback-Leibler divergence between the datasets here is 1.7224

The graph shows that the two distributions are quite different. These results are a baseline to improve upon. The quality score distributions for RAND and FBI are graphed in Figure 12:

Figure 12 - RAND and FBI quality pmfs

It can be seen that the FBI generally has better quality templates compared to the RAND. The KL divergence here between RAND and FBI is 0.9067.

5.3.1 Exchange

The first algorithm that will be used for picking a subset is the exchange algorithm. With the exchange algorithm, 10 bins of 100 templates each are created by picking randomly from dataset A (without replacement). 5 bins are selected from the 10 to make the template vector. This means all 252 combinations will be processed (10 Choose 5).

For each combination, first a template vector containing all the templates across the five bins is made. Second, a score vector is made, containing all genuine scores belonging to the templates in our template vector. Third, the probability mass function distribution is created as previously described in the paper. Fourth, the Kullback-Leibler Divergence code is called. The first combination and score are stored, and then subsequent combinations are checked for a lower

KL score. If a combination has a lower KL score than the current one, the current score is replaced with the new score, the current score vector is replaced with the new score vector, and the current subject vector is replaced with the new subject vector.

The exchange algorithm is not very intelligent. To get the best score possible from it, the number of bins and the size of the bins need to be tweaked for the databases. The proper number of bins and bin size depends on the distribution of the sequestered and non-sequestered databases. The exchange algorithm does usually yield a distribution that is more similar to the sequestered database than the original. Since templates are picked, it is probable that some templates will not result in scores (if another template is not paired with it). Below is a graph showing the exchange versus FBI versus RAND distributions.

Figure 13 - Exchange, RAND, and FBI pmfs

The Kullback-Leibler Divergence for Figure 13, between Exchange and FBI, is 1.3765.

Looking at the quality distributions:

Figure 14 - Exchange, RAND, and FBI - Quality pmfs

Figure 14 illustrates that the exchange subset has a higher peak and lower lows than RAND. The distribution is quite jagged, due to the random picking of templates and low template count (494 templates). The exchange quality distribution is still quite different from the FBI distribution. Between the subset quality distribution and the FBI quality distribution, the KL divergence is 0.8757.

5.3.2 Exchange Group

Next, the exchange algorithm is modified in an attempt to realize a closer kl divergence between distributions. By picking entire subjects instead of templates, all of our templates end up with scores. We make 10 bins of 25 subjects, and use the same concept as with the exchange algorithm. Figure 15 shows the pmf distribution of match scores. Figure 16 shows the pmf distribution of quality scores:

Figure 15 - Exchange Group, RAND, and FBI pmfs

The Kullback-Leibler Divergence between Exchange Group and FBI is 1.3298.

Figure 16 - Exchange Group, RAND, and FBI - Quality pmfs

The Kullback-Leibler Divergence between Exchange Group and FBI quality distributions is 0.8552.

5.3.3 Weighted Group

Moving away from the exchange group algorithm, now the weighted group algorithm will be utilized to process the datasets. The approach of the weighted group algorithm is to group the templates by which subject they belong to (using the subject id string). There are 1683 subjects that have genuine scores in the RAND face dataset, so there are 1683 groups. After they are grouped together, weights are calculated and subjects are inserted into a sorted list by their weight.

The Matlab code for calculating the weight is as follows:

```
Indexdensity = score + 1;
```

```
one = pmfgenFBI(indexdensity);
```



```
two = pmfgenRAND(indexdensity);
```

```
if(one > two)
```

```
    quantity = two / one;
```

```
else
```

```
    quantity = one / two;
```

```
sum = sum + (quantity / nscores)
```

where nscores is the number of pairs in the group, score is the match score, pmfgenFBI is the FBI distribution, and pmfgenRAND is the RAND distribution. The comparison based assignment of quantity yields better results than just assigning a single value for both situations. With the comparison, having equal values in the distributions gives a higher weight than having different values. After the subject groups are sorted, the algorithm starts at the highest weighted position and iterates through, generating a genuine score vector, making a probability mass function vector, and checking the kl score after adding each subject. The best number of templates for weighted group was found to be 713 (this subset yields the lowest kl score). Figure 17 shows the distributions:

Figure 17 - Weighted Group, RAND, and FBI pmfs

The Kullback-Leibler Divergence here between Weighted Group and FBI is 0.8074.

The quality scores for FBI, RAND, and Weighted Group are graphed in Figure 18:

Figure 18 - Weighted Group, RAND, and FBI – Quality pmfs

Between the subset quality distribution and the FBI quality distribution the KL divergence is 0.8734.

5.3.4 Weighted Pair

The next method of obtaining a subset is the weighted pair algorithm. To employ the weighted pair algorithm, the RAND genuine vector is looped through and each entry is assigned a weight. To make a weight for each pair, the following Matlab code is used:

```
indexdensity = score + 1;

one = pmfgenFBI(indexdensity);

two = pmfgenRAND(indexdensity);

%%scoring for a pair

quantity = one / two;
```

To evaluate a weighted pair, the code takes the probability of the current score for FBI divided by the probability of the current score for RAND. After the weight is determined, the score, the probe id, the gallery id, and the weight are stored in vectors. The position each variable is inserted into each vector depends on the current weight and match score. If an entry does not exist for a certain match score, one is created at the correct position based on weight. By doing this, the vectors are already sorted after all the data is processed. There may be multiple gallery templates for a match score, so the gallery ids are stored as a vector within a vector. There also may be multiple probe templates for a match score, so the probe ids are stored as a vector within a vector.

After the vectors are sorted, they are iterated through. A new vector for templates is

created. For each position in the vectors, all of the probe template ids and gallery template ids are examined. If the template ids do not exist in the new templates vector, they are added in. Using these ids all the corresponding genuine scores are determined, a probability mass distribution is created, and the Kullback-Leibler divergence test is run between the immediate probability mass distribution and the FBI probability mass distribution. The algorithm starts at the highest weighted position and iterates through, checking the kl score after adding each template. 2177 was found to be the best number of templates (this subset has the lowest kl score) for RAND and FBI. Running the same process on other datasets will most likely yield a different best number of templates.

The disadvantage of weighted pairs is that by picking templates, pairs can be brought in that weren't intended to. For example, if the pairs of template 1 and template 3, and pairs of template 1 and template 2 are desired, weighted pairs brings in the pair of template 2 and template 3. For these datasets, we have 2872 pairs, with 1415 being intentional.

Figure 19 - Weighted Pairs, RAND, and FBI pmfs

The Kullback-Leibler Divergence here between Weighted Pairs and FBI is 0.5410. The quality distributions for weighted pairs, RAND, and FBI are shown in Figure 20:

Figure 20 - Weighted Pairs, RAND, and FBI - Quality pmfs

The weighted pair quality distribution is still quite different from the FBI distribution, however. Between the subset quality distribution and the FBI quality distribution the KL divergence is 0.9043.

5.3.5 Weighted Group Single

Another method that uses weights, called weighted group single, records the highest weighted score for each subject/person. Since this method only records and uses one score per subject, all pairs are intentional. There is one probe template and one gallery template per subject. There are 1683 subjects that have genuine scores in the RAND face dataset, so there are 1683 groups. For each template that is added, a genuine score vector is generated and a probability mass function vector is generated from the genuine score vector. After this, the Kullback Leibler divergence test is run between the immediate pmf vector and the FBI pmf vector. For our datasets, the top 1581 subjects (1581 gallery templates and 1581 probe templates) yield the lowest Kullback-Leibler Divergence score. This yields 1581 intentional pairs, and the KL score between Weighted Group Single and FBI is 0.5228.

Figure 21 - Weighted Group Single, RAND, and FBI pmfs

The quality distributions for Weighted Group Single, FBI, and RAND are as follows:

Figure 22 - Weighted Group Single, RAND, and FBI - Quality pmfs

This algorithm yields the lowest Kullback-Leibler divergence for our datasets, producing the subset most similar to the FBI database. While weighted group single yields a more similar genuine distribution than the weighted pairs for FBI face and RAND face, this may not be the case for other datasets. Between the subset quality distribution and the FBI quality distribution the kl divergence is 0.8883.

5.3.6 – Finding the optimal number of templates automatically

For weighted group single, weighted group, and weighted pairs we have a collection of templates/subject ordered by weight. For each algorithm, there is a number of templates picked from the collection that yields the lowest Kullback-Leibler Divergence score. This optimal number of templates can be found automatically.

The method of doing this is to start at the end of the list with the highest weight and proceed down the list adding templates. We start with 1 desired pair or group and record the Kullback Leibler score. We repeat the process with 2 desired pairs or groups, then 3, and so on. After we have looked at the Kullback Leibler score at each possible subset from the weighted entries, we record the template ids and the distribution yielding the lowest Kullback-Leibler score. Having this procedure available allows anyone to use our process of picking a subset (on any dataset) without needing manual tuning in respect to the number of templates.

5.4 – Looking at Performance and Comparison Tests

The Receiver Operating Characteristic curve (ROC) gives a summary of the performance of the matching performance of a biometric system. It is a graph of the genuine accept rate (1 - False Reject Rate) versus the false accept rate. To determine the final performance, the ROC

curve of the subset distribution we have created is compared with the ROC curve of the sequestered. [22]

The max score of impostor RAND is 106. The max score of impostor FBI is 974. 99.5% of the impostor scores for FBI are between 0 and 25 inclusively. 99.7% of the impostor scores for RAND are between 0 and 25 inclusively.

The max score of genuine RAND is 4721. The max score of genuine FBI is 5730. 99% of the genuine FBI scores are between 0 and 3000. 99% of the genuine RAND scores are between 0 and 2500. Due to the range difference between the impostor and genuine distributions in both datasets, the ROC curves appear very steep, and there is not much on the impostor dependent side.

As the number of points rises, the distance between the ROC curves for the two distributions decreases. The following three figures demonstrate this. The space between the yellow and blue lines should be scrutinized:

Figure 23 - 25 points RAND, FBI ROC

Figure 24 - 500 Points RAND, FBI ROC

Figure 25 - 4722 points RAND, FBI ROC

Figure 26 - 4722 points RAND, 5730 points FBI ROC

There is an optimum number of points where increasing beyond will yield no benefit. The optimal number of points is then determined by the max of the score vector with the highest score + 1. By using this many points, each threshold will have a distance of 1 between itself and the next threshold. FAR and GAR are calculated at each threshold point. An ROC curve increases in height when increasing the value of x .

The ROC curve is a good indicator/measure of performance at specific points, but it provides a spectrum of performance assessments. How can this be boiled down to a simple, single-number measure of performance? The answer, is to measure the *area under the ROC curve* (abbreviated *AUC*, or less frequently, *AUROC*).

Assuming that one is not interested in a specific trade-off between true positive rate and false positive rate (that is, a particular point on the ROC curve), the AUC is useful in that it

aggregates performance across the entire range of trade-offs. Interpretation of the AUC is easy: the higher the AUC, the better, with 0.50 indicating random performance and 1.00 denoting perfect performance.

The x vectors for both RAND and FBI decrease or hold the same as the threshold increases. The y vectors for both RAND and FBI also decrease or hold the same as the threshold increases. We calculate the AUC by first calculating width and height between points.

```
height = (ys(index) + ys(index-1)) / 2;
```

```
width = xs(index-1) - xs(index);
```

Where xs is our x vector for RAND and ys is our y vector for RAND. Multiplying height by width produces the AUC between the points. If the x values are the same, we have a width of 0, and this produces an area of 0. If the y-values are the same, it produces a positive height.

The AUC values between the points can be added up to produce AUC for the entire curve. The AUC for RAND is 0.9468 and the AUC for FBI is 0.9985.

To generate the ROC curve for each of the subset picking algorithms, an impostor vector must first be generated based on the template vector. The following figures shows the performance difference between the exchange group algorithm and the FBI dataset:

Figure 27 - Exchange Group Subset vs FBI ROC

The difference between the AUROC values for the Exchange Group Subset (.9471) and the FBI (.9985) is -.0514 or -5.15 percent.

The following figure shows the performance difference between the exchange algorithm and the FBI dataset:

Figure 28 - Exchange Subset vs FBI ROC

The difference between AUROC values for the Exchange subset (.9481) and FBI (0.9985) is - .0504 or -5.05 percent.

The following figure shows the performance difference between the weighted group algorithm and the FBI dataset:

Figure 29 - Weighted Group Subset vs FBI ROC

The difference between AUROC values for the Weighted Group subset (.9972) and FBI (.9985) is -0.0013 or -.13 percent. The Weighted Group ROC curve has an area value of .9972, while the FBI has an area value of .9985.

The following figure shows the comparison of using the weighted group single on the RAND dataset mentioned previously in this paper. It can be seen that the layout of the ROC curves/segments as a whole are very close together.

Figure 30 - Weighted Group Single vs FBI

The difference between AUROC values for the Weighted Group Single subset (1.0) and FBI (0.9985) is 0.0015. This is a difference of .15 percent. Though the weighted group single had a lower kl score than pairs, its performance is further away from the FBI. By picking the highest weighted score from each subject group, we eliminate a lot of low scores. The lack of low scores pushes the ROC curve over to the right some. We end up with a ROC curve that is actually better than the FBI, and can be considered perfect (it's 1.0).

The following figure shows the difference between the Weighted Pairs algorithm and the FBI dataset:

Figure 31 - Weighted Pairs Subset vs FBI ROC

The difference between AUROC values for the Weighted Pairs subset (.9993) and FBI (0.9985) is 0.0008 or .08 percent. This makes the weighted pairs algorithm the winner for our datasets.

The difference between AUROC values for RAND (0.9468) and FBI (0.9985) is 0.0517. The Weighted Pairs algorithm produces a dataset that is much closer to the sequestered dataset (FBI) than the original non-sequestered dataset. The difference by percent of the FBI value is .08 percent for Weighted Pairs and 5.2 percent for the original. By using an algorithm to pick subjects, the subset is much closer in performance to the sequestered dataset than the original non-sequestered dataset.

5.5 - Time taken to run each algorithm

It has been shown how each algorithm performed in terms of how close the subset it made was to the FBI dataset. It is also important to examine how long it takes for each algorithm to make subsets, and create and display an ROC Curve. All times presented here are assuming the the csv files with the java module have already been generated, and the csv data has already been read in with matlab, and organized into template ids, genuine vectors, and impostor vectors.

For the exchange algorithm it takes 10 minutes and 34.958211 seconds to generate the subjects and genuine scores of the subset. 40 minutes and 20.863201 seconds to generate the impostor score vector. 2 minutes and 45.483163 seconds to generate and graph the ROC curve. Overall, it takes 53 minutes and 41.304575 seconds to run the entire process.

For the exchange group algorithm it takes 10 minutes and 33.844211 seconds to generate the subjects and genuine scores of the subset. 41 minutes and 7.838188 seconds to generate the impostor score vector. It takes 2 minutes and 45 seconds to generate and graph the ROC curve. Overall, it takes 54 minutes and 27.183454 seconds to run the entire process.

For the weighted group single algorithm it takes 18.559230 seconds to find the highest weighted score for each subject. It takes 3.07 seconds to make weighted group single pairs. It takes 45 minutes 52.575688 seconds to generate the subjects and genuine scores of the subset. It takes 9 hours 50 minutes and 2 seconds to generate the impostor vector. It takes 2 minutes and 44.838877 seconds to generate and graph the ROC curve. Overall, it takes 10 hours 39 minutes and 1.608965 seconds to run the entire process.

For the weighted pairs algorithm it takes 1 minute and 34.438094 seconds to make the pairs. It takes 4 hours 7 minutes and 50.159257 seconds to generate the subjects and genuine

scores of the subset. It takes 2 hours 51 minutes and 11.201010 seconds to generate the impostor vector. It takes 2 minutes 43.876338 seconds to generate and graph the roc curve. Overall, it takes 7 hours 2 minutes and 19.674699 seconds.

For the weighted group algorithm it takes 31.101627 seconds to split into groups sorted by weight. It takes 1 hour 8 minutes and 53.762943 seconds to generate the subjects and genuine scores of the subset. It takes 43 minutes 25.928399 seconds to generate the impostor vector. It takes 2 minutes and 45.998681 seconds to generate and graph the roc curve. Overall, it takes 1 hour 55 minutes and 36.79165 seconds.

The algorithms perform from fastest to slowest in this order: Exchange, Exchange Group, Weighted Group, Weighted Pairs, Weighted Group Single.

6. Explanation of results for FBI and RAND:

The FBI mass distribution has more than twice the number of non-zero percentages than the RAND mass distribution. This means in the probability mass distribution of RAND, each non-zero data point will have on average a probability 2x as large as the FBI. For samples of RAND, the average non-zero data point probability will be even higher. The average non-zero percentage for RAND is 6.135×10^{-4} , and the average non-zero percentage for FBI is 2.983×10^{-4} . The average KL divergence at a single match score comes out to 3.1×10^{-4} .

Adding subjects beyond a certain point causes more comparisons and makes the

percentage smaller on one side in the comparisons. To get the most similar distributions, each non-zero data point must be as close as possible to the non-zero data point in the sequestered set. For the datasets used in this study, this point is at 2177 templates for weighted pairs and 3162 templates for weighted group single. If the RAND dataset had proper amounts of match scores at these zero-percent RAND, positive percent FBI points, then a subset that is more similar to the sequestered could be made.

The RAND and FBI datasets genuine score distributions are quite different. The RAND genuine distribution starts as a spike and decreases from there. The FBI genuine distribution increases, goes flat, and then decreases. The first point of intersection (where the probabilities are equal) between the two distributions is around 200. The FBI genuine dataset score vector has around 8 times as many scores as the RAND score vector. 73 percent of the RAND genuine scores are low scores (below 200), while only 9 percent of the FBI genuine scores are low (below 200).

By going through our RAND genuine scores, storing the highest match score for each template, and copying the template to a specific directory, images can be examined by their best match score. Quality scores are added onto the end of the filename, and can also be examined. Low match scores are caused by the following factors: off angle on one side of the match pair (part of one eye may be cut off), bad background, drastically different background than other images, and low-resolution/bad camera.

Low quality scores of individual templates are caused by face angle (frontal gives a much higher quality score than an image with the face at an angle), background (a solid color wall or canvas gives a higher quality score than no wall/canvas), low resolution/bad camera, and lighting (good lighting produces better quality templates).

If the subjects responsible for low match scores (below 100) are removed from both FBI and RAND, the distributions are much closer. The KL score between the distributions (FBI vs RAND) with the low scores removed is 0.63. If the Weighted Pairs algorithm is run with 2500 subjects on these new distributions, the resulting score is 0.4678.

To summarize, the RAND (or a subset of RAND) and FBI pmf distributions show a degree of dissimilarity due to the significant amount of bad images/low score pairs in the RAND, and the fact that the RAND has less score data points.

7. Match Score and Quality Score Correlation:

One of the goals of this study was to identify the biometric quality and population factors that drive the distribution of match scores in data sets A and B. After looking over these factors, it was found that the templates follow a normal curve in regard to quality (see quality distribution graph shown previously in this paper - Figure 12). There aren't very many subjects at the low end (ex quality of 50) and high end (ex quality of 95). Due to the normal distribution of quality scores (ex: quality of 50) and high end (ex: quality of 95), it is difficult to see the relationship between quality score and match score on datasets A and B.

To show the relationship in a transparent manner, the first step is to loop through our genuine score vector. At each index, the probe and gallery quality scores are averaged together and used as the quality variable (X on the graph). The sum of the match scores for each quality score and the number of elements for each quality score is stored. After finishing the iterations, the average match score is computed at each quality score (ex. Average match score at quality of 55, average match score at quality of 56, etc.), and this is used as our average match score

variable (Y on the graph).

Figure 32 depicts this:

Figure 32 - Quality versus Match Score - Genuine

Figure 32 shows that the correlation is mostly positive for both RAND and FBI. So, according to the face datasets used in this project, higher quality templates, for the most part, generate higher genuine match scores. The correlation is not completely positive most likely because of other factors that influence match score, such as age, gender, race, etc. Genuine match scores affect FRR. This in turn affects GAR ($\text{GAR} = 1 - \text{FRR}$). Higher scores lead to higher GAR's across the board. This leads to higher peaks and more AUC, which in turn affects performance.

```
ni = int32(i);
for j = 1:(ni-1) % <
    gensum = genuinescores(j) + gensum;
end
for k = ni:(maxScore+1) % >=
    impsum = impostorscores(k) + impsum;
end;
far = double(impsum) / double(length(thesescoresimp));
frr = double(gensum) / double(length(thesescores));
```

If we repeat the process for the impostor scores of RAND and FBI, we get the following:

Figure 33 - Quality versus Match Score - Impostor

It appears that there is no correlation between the quality of the subjects and match scores for the impostor distribution.

8. Conclusion

Overall, this study has determined that it is practical to make a subset of one dataset that closely represents the performance of another dataset. Out of our methods, weighted pairs generated ROC curves that were closest to the FBI ROC Curves. Weighted Group Single and Weighted Group also generated ROC curves that were close to the FBI ROC curve.

The exchange and exchange group algorithms generated ROC curves that were far away from the FBI ROC curve. These algorithms should not be used for generating a subset because

they are not accurate enough.

If we use different datasets for A and B, we may have a different weighted algorithm as the winner. By using a weighted algorithm, we can generate a subset that performs close to the performance of the sequestered dataset.

Our method of generating a similar dataset to the sequestered dataset will be useful in cases where a dataset is not allowed to be released to an outside entity, but the entity wants something similar in performance. The generated dataset will then be used in place of the sequestered dataset.

9. Future Work

There are various different new approaches we can try to improve our results. It may be possible to eliminate the Kullback-Leibler divergence test entirely from our subset generation methodology. The Kullback Leibler is being used to see the difference between genuine score distributions, but it has some disadvantages. It only makes comparisons if both datasets have scores present at a score index, we are only running the comparison on genuine data, and it is very different from the ROC performance test. The KL test is especially inaccurate when the length of the vectors between datasets is different (for example, if A had scores from 0 to 200, and B had scores from 0 to 5000). Weights are inaccurate in this scenario as well. A new idea would be to create weights and do the kl test would be to divide the larger dataset into bins. For example, 0 to 24 inclusive of B would correspond to 0 of A. ROC curves work in a way that is relative to the range of the score vectors, so they should work fine as is in this scenario. Luckily, the range of our datasets for FBI and RAND are similar, so this problem did not really come up in our runs.

The Kullback-Leibler Divergence test is a middleman, and if it could be eliminated, we would obtain better results. It would also make the process of making a subset easier to understand, with one less concept/step. The problem with eliminating the KL test right now is that generating an impostor score vector takes a very long time (around 3 hours for weighted pairs), and we would need to generate impostor scores with each template pair we add.

However, it is this researcher's opinion that this could be done with some crafty programming. If we add in only the corresponding impostor scores for each template pair we add, it may be possible to do this in a reasonable amount of time. The impostor scores, genuine scores, and templates would need to be organized in a very intelligent way using Map/Dictionary data structures, so we can get all impostor scores corresponding to a template. We would only add in the impostor scores that go along with the templates we have in our template vector. After we have the genuine and impostor scores for a subset, we will run the ROC curve process, which takes only .685736 seconds. The entire run through may take longer (or, it could take less time, there is no way to know until it is tried), but should give better results.

It has been seen through testing that a lower Kullback-Leibler divergence score does not always lead to better performance. Instead of choosing a subset based on the Kullback-Leibler Divergence scores, the new method will choose based on the difference between the FBI AUROC and the subset AUROC.

PseudoCode for add:

```
//Assuming we have a map (or a map of maps) for impostors and a map of genuines
```

```
//where template ids are the keys. These maps would be generated one time, when we
```

```
//read in our csv files
```

```
existingtemplates;
```

```
newtemplate id;
```

```
Loop through existing templates{
```

```
    probeScoreImp = impostorMap[new template][existing template];
```

```
    If(probeScoreImp exists):
```

```
        Add probeScoreImp to impostorScoresVector;
```

```
    galleryScoreImp = impostorMap[existing template][new template];
```

```
    If(galleryScoreImp exists):
```

```
        Add galleryScoreImp to impostorScoresVector;
```

```
    probeScoreGen = genuineMap[new template][existing template];
```

```
    If(probeScoreGen exists):
```

```
        Add probeScoreGen to genuineScoresVector;
```

```
    galleryScoreGen = genuineMap[existing template][new template];
```

```
    If(galleryScoreGen exists):
```

```
        Add galleryScoreGen to genuineScoresVector;
```

```
}
```

```
Add new template to existing templates;
```

Check the $\text{abs}(\text{AUROCFBI} - \text{AUROCRAND}_{\text{subset}})$;

Store subset if it is the lowest difference seen;

Repeat;

Once we have the genuine and impostor vectors, it takes .685736 seconds to make a ROC curve. We make a count vector at each score index for the impostor vector and genuine vector. This count could likely be made as we go along, inside the loop above (on each impostor score add and genuine score add) for an even quicker execution. The ROC curve generation itself as-is is very quick and not a problem.

If it is desired to continue using the Kullback-Leibler divergence test (instead of eliminating it), it could be improved. One thing to try would be to add a penalty for the Kullback-Leibler Divergence measure when one dataset score index has a zero percentage and the other dataset has a positive percentage. This may provide better results for all of our algorithms, or it may provide worse results. This needs to be retested for all algorithms. This penalty process would yield much higher Kullback-Leibler Divergence scores, but it is all relative. Once a subset is picked due to its low kl score, we move on to performance.

With weighted pairs, all the intentional pairs have positive percentages on both datasets, but the unintentional pairs could be 0 percent on the FBI (sequestered) end. By adding a penalty for these instances, we would most likely end up with a different subset and a different ROC curve.

The same concept can apply to the weighted group algorithm. By adding in a penalty, we

may select a different number of templates and we may end up with a different ROC Curve. For exchange/exchange group, we may select a different combination of bins and may end up with a different ROC Curve.

Weighted Group Single is unlikely to benefit much from adding a penalty. All pairs are intentional, and all weights above 0.0 do not have a penalty. It cannot be completely ruled out though. Certain datasets may yield a performance improvement from introducing penalties with Weighted Group Single.

At some point it may be desired to use a different matcher and datasets. To do this, some modifications to our java code and matlab code will have to be made. There are also most likely new ideas that could be used to develop smarter subset picking algorithms.

Bibliography:

- [1] Jain, A. K.; Ross, A.; Prabhakar, S., “An Introduction to Biometric Recognition”, January 2004, URL: http://www2.citer.wvu.edu/members/publications/files/RossBioIntro_CSVT2004.pdf
- [2] Weng, J. J.; Swets, D. L., “Face Recognition”, URL: <http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.43.9866&rep=rep1&type=pdf>
- [3] Wang, R, “Performance Prediction for Biometrics Recognition Systems”, September 2007
- [4] Klare, B., et al. “Face Recognition Performance: Role of Demographic Information.” IEEE T-IFS, V7(6), 2012
- [5] The Multi-Scenario Multi- Environment BioSecure Multimodal Database (BMDDB), IEEE T-PAMI, V32(6), 2010, URL: [http://www.cistib.org/gestpub/attachments/The%20Multi-Scenario%20Multi-Environment%20BioSecure%20Multimodal%20Database%20\(BMDDB\).pdf-d0f01f21a585f247e78f924562ea565c.pdf](http://www.cistib.org/gestpub/attachments/The%20Multi-Scenario%20Multi-Environment%20BioSecure%20Multimodal%20Database%20(BMDDB).pdf-d0f01f21a585f247e78f924562ea565c.pdf)
- [6] Chen, Y.; Dass, S.; Jain, A., “Fingerprint Quality Indices for Predicting Authentication Performance” URL: http://www.cse.msu.edu/rgroups/biometrics/Publications/ImageQuality/ChenDassJain_FpQuality_AVBP_A05.pdf
- [7] Marasco, Emanuela, “Fingerprint Recognition” URL: http://www.csee.wvu.edu/~gdguo/courses/fa14/biom426/Slides/Lecture1_Fingerprint_Recognition.pdf
- [8] “Probability Mass Function”. Statlect, the digital textbook
- [9] “Probability Mass Function”. Wikipedia
- [10] “Kullback Leibler (KL) Distance”. Monash University
- [11] “Face Recognition.” NEC URL: http://www.nec.com/en/global/solutions/biometrics/technologies/face_recognition.html
- [12] “Face Recognition”. NISTC Subcommittee on Biometrics URL: <http://www.biometrics.gov/Documents/facerec.pdf>
- [13] “Face Recognition Using Dimensionality Reduction PCA and LDA” URL: http://www.cse.unr.edu/~bebis/CS790Q/Lect/FR_PCA_LDA.ppt

[14] “A new LDA-based face recognition system which can solve the small sample size problem”

[15] Sarkar, Sudeshna “LECTURE 10: Linear Discriminant Analysis”

URL: <http://www.facweb.iitkgp.ernet.in/~sudeshna/courses/ML06/lda.pdf>

[16] Stergiou, Andreas, “Elastic Bunch Graph Matching Face Recognition: Performance and Comparison with Subspace Project Methods” URL:

http://www.academia.edu/958325/Elastic_bunch_graph_matching_face_recognition_performance_and_comparison_with_subspace_projection_methods

[17] “Face recognition 101: Eigenfaces.” OpenIMAJ

URL: <http://www.openimaj.org/tutorial/eigenfaces.html>

[18] Guo, Guodong, “Human Face.”

URL: <http://www.csee.wvu.edu/~gdguo/courses/fa14/biom426/Slides/Lecture8.pdf>

[19] Guo, Guodong, “Face Detection and Recognition”

URL: <http://www.csee.wvu.edu/~gdguo/courses/fa14/biom426/Slides/Lecture9.pdf>

[20]

Zhang, Chaoyang, “Comparison of Three Face Recognition Algorithms”

[21]

Bilal, Muhammad, “Comparison of Local Binary Patterns (LBP) and Principal Comp”

URL: <https://prezi.com/nldeppmeqyo3/comparison-of-local-binary-patterns-lbp-and-principal-comp/>

[22] Guo, Guodong, “Performance Measures.”

URL: <http://www.csee.wvu.edu/~gdguo/courses/fa14/biom426/Slides/Lecture5.pdf>

[23] “Megamatcher Technology Features and Capabilities.”

URL: <http://www.neurotechnology.com/megamatcher-technology.html>