Evolutionary Facial Feature Selection

Aaron K. Baughman
IBM Global Services
Fairfax VA, USA
703-653-7525
baaron@us.ibm.com

ABSTRACT

With the growing number of acquired physiological and behavioral biometric samples, biometric data sets are experiencing tremendous growth. As database sizes increase, exhaustive identification searches by matching with entire biometric feature sets become computationally unmanageable. An evolutionary facial feature selector chooses a set of features from prior contextual or meta face features that reduces the search space. This paper discusses and shows the results of an evolutionary computing approach with agglomerative k-means cluster spaces as input parameters into a LDA evaluation function to select facial features from the Carnegie Mellon University Pose, Illumination, and Expression database of human faces (PIE).

Categories and Subject Descriptors

I.5.3 [**Pattern Recognition**]: Clustering – *algorithms.*

General Terms

Algorithms, Performance, Experimentation, Theory.

Keywords

Application, Artificial Intelligence, Genetic algorithms, Pattern recognition and classification, Search.

1. INTRODUCTION

Traditional automated biometrics matured from the late 1960's with the first Integrated Automated Fingerprint Identification System (IAFIS) fielded by the Federal Bureau of Investigation in 1972. Within the United States Government, other biometric systems include the US-VISIT program, Transportation Workers Identification Credentials (TWIC) program and the Registered Traveler (RT) program. A few international biometric programs include the United Arab Emirates border control, Colombia's La Registraduria Nacional del Estado national population registry, Germany's Federal Office of Administration visa program, and Pakistan's passport database.

The National Institute of Standards and Technology (NIST) has run Facial Recognition Vendor Tests (FRVT) beginning with the Facial Recognition Technology (FERET) program in 1993. NIST's independent evaluations identified the most promising approaches to facial recognition. The evaluations have achieved two orders of magnitude improvement in performance from 1993, 0.79 False Rejection Rate (FRR) at 1/1000 False Acceptance Rate (FAR), to 2006, 0.01 FRR at 1/1000 FAR. The FRVT 2006 achieved the Facial Recognition Grand Challenge (FRGC) of achieving a 0.02 FRR at 1/100 FAR from FRVT 2002, established the first 3D face recognition benchmark, showed progress of facial recognition technologies among changes in lighting and that face recognition algorithms are capable of performing better than humans [8]. The number of participants increased from 5 in FRVT 2000 to 22 in FRVT 2006.

This paper presents the design, parameter selection and experimental results of an evolutionary facial feature selector algorithm. First, the field of facial recognition as related to the proposed algorithm is introduced in Section 2. Then, in Section 3, related works are presented. Next, in Section 4, the evolutionary algorithm is discussed while in Section 5 the results are given. Finally, a discussion and future work is presented in Section 6.

2. BACKGROUND

Semi automated facial recognition technology has been around since the 1960's where technicians manually marked facial features. In the 1970's, Goldstein et al used 21 markers for facial recognition [2]. The first automated facial recognition techniques evolved in the 1980's with Kohonen's method of a self organizing map or Neural Network to recognize faces. Facial recognition technology is a widely accepted biometric, in part, because humans use facial recognition on a daily basis to recognize other individuals.

Biographical information provides Meta Data that describes facial images. The CMU PIE database provides facial expression, glasses, talking, pose, flash, light, sex, age, race and etc. describing each subject [10]. Similar biographical data is used by the FBI to build database indices and binning structures to limit the scope of a biometric search. For latent fingerprint searches, the FBI requires that probe penetration rates must not exceed 30% of the entire gallery [11]. Subject biographical information is used to decrease the probe penetration rate.

3. RELATED WORKS

Previous works [1], [4], [5], [6], [7], [12] and [13] presented several methods such as inverse file indexing, clustering and matching algorithms to support biometric searching. This paper expands evolutionary computing as applied to facial recognition search. Several related works within the fields of k-means clustering, linear discriminant analysis (LDA), and evolutionary computing support the work within this paper.

GECCO '08, July 12–16, 2008, Atlanta, Georgia, USA.
ACM 978-60558-131-6/08/07.

3.1 Biometric Searching

Biometric searching addresses two types of problems. First, 1:1 or biometric verification is not presented within this paper. The more challenging class concerns 1:N or M:N where an identity has multiple biometrics. Within Germain's work, an algorithm of transformation parameter clustering attempts to build fingerprint database index maps [4]. Triangular minutiae constellations are created from

$$(n \text{ choose } 3) = \frac{n!}{3!(n-3)!} \quad (1)$$

minutia points. A bound is established so that not every triangular shape is computed. The full index or key consists of nine components: length of each side, ridge counts between the pair and angles between the pair [4]. Each index is placed into a multimap or container. If the key is already within with a container, the entry is added. Through a search, accumulation of evidence for a potential match is generated based upon the members that are found within the probe keyset. On average, the algorithm achieved 7.3 μsec/print with a 1/10 False Negative Rate (FNR) and 1/1000000 False Positive Rate (FPR) [4].

The L1 Identix Automated Biometric Identification System (ABIS) 4.1 provides several modalities for facial recognition search [1]. Each modality is a tradeoff between speed and accuracy. The vendor provides search capability for proprietary face representations as vector, full template, two pass and two pass with skin luminescence [1].

3.2 Clustering

Grouping objects into clusters based upon feature variances among objects create collections of associated items. K-means clustering is a grouping algorithm that minimizes the within class scatter or the sum squared error.

Step 1 consists of optionally initializing the K class centers. Step 2, each member X_k is assigned to a class based on the minimization of the Euclidean distance.

$$Dist(u) = \sqrt{(\mu - u)}^2 \quad (2)$$

The distance matrix D_m is

$$D_m = [Dist_1 \quad \ldots \quad Dist_n] \quad (3)$$

while the assigned class for u maintains

minimum (Dm)

Step 3, each cluster's mean is calculated with

$$Center_l^a = \frac{1}{N_l}\sum_{k=1}^{N_l} member_k \quad (4)$$

$$Center_b^a = \begin{matrix} center_1^1 & \ldots & center_1^a \\ \ldots & \ldots & \ldots \\ center_b^1 & \ldots & center_b^a \end{matrix} \quad (5)$$

where a is the center vector for each cluster and b is each cluster.

Step 4, step 2 is repeated if any class centers changed.

Step 5, calculate a clustering threshold.

A method described by Zhao et al [13] calculates the threshold value as

$$Threshold = \frac{1}{2}\left(Center_k + Center_{k-1}\right) \quad (6)$$

Where k is the highest valued center and k-1 is the second highest valued center.

3.3 Linear Discriminant Analysis (LDA)

Within Linear Discriminant Analysis, a search within a given space yields vectors that best linearly discriminant classes. Martinez et al [7] describe LDA as the maximization of the ratio of between class scatter and within class scatter also known as the Fisher criterion [7]. The analysis technique strives to maintain tightly coupled members within a class while having high separation between members of differing classes.

The within class scatter matrix is defined by

$$S_w = \sum_{j=1}^{c}\sum_{i=1}^{N_j}(x_i^j - \mu_j)(x_i^j - \mu_j)^T \quad (7)$$

where x_i^j is the ith sample of class j, μ_j is the mean of class j, c is the number of classes and N_j is the number of samples in class j [7]. The between class scatter matrix is defined by

$$S_b = \sum_{j=1}^{c}(\mu_j - \mu)(\mu_j - \mu)^T \quad (8)$$

Where μ represents the mean of all classes [7]. The maximization of between class scatter and the minimization of within class scatter is defined by

$$\frac{S_b}{S_w} \qquad (9)$$

Often, LDA is used for both class discrimination and an algorithm for dimensionality reduction. With respect to facial recognition, the Fisherface algorithm projects feature vectors of Principal Component Analysis onto the Fisherface matrix [5]. The most discriminating features that inherently maintain the Fisher property are kept. The predefined feature vectors are used for facial recognition.

3.4 Evolutionary Computing

Learning the face space of a face database maintains or reduces the dimensionality of data. The reduction of data complexity reduces the amount of computational complexity. The evolutionary pursuit seeks to learn an optimal face space for the purpose of pattern classification and data compression [12]. Evolution is driven by a fitness function. An example fitness function combines performance accuracy, $\xi_a(F)$ with the predicted risk, $\xi_b(F)$, to evaluate a face space [12].

$$\xi(F) = \xi_a(F) + \lambda\xi_b(F) \qquad (10)$$

Chengjun Liu et al [6] examined the application of genetic algorithms to face recognition. The fitness function includes $\xi_a(F)$ that defines facial recognition accuracy and $\lambda\xi_b(F)$ that defines class scatter.

The evolutionary computing method reduced a 30 principal component analysis space to a 26 vector face space.

4. NOVEL EVOLUTIONARY FEATURE SELECTION APPROACH

An evolutionary genetic algorithm was implemented with Evolutionary Computation in Java (ECJ), LDA fitness evaluation and an agglomerative K-Means clustering algorithm to find a maximum class scatter. All of the faces from the CMU PIE database were loaded into an evolutionary environment.

The evolutionary automaton contains 10 randomly generated chromosomes of length 12. Each bit position on the chromosome is either a 1 for including a face trait or a 0 for excluding a trait. For each chromosome, an agglomerative clustering algorithm generates a cluster space. All of cluster spaces' fitness values are measured by an LDA evaluation function. During a total of 25 generations, genetic operators are the independent variables for experimental analysis. A formal definition follows:

Evolutionary Automaton = $\{x_i, X, CP(x), G, E(x_i), O\}$

Where x_i = a chromosome, X = population, CP(X) = cluster space, G = Generation, $E(x_i)$ = evaluation function, O = operator.

For the purpose of experimental setup:

$$|x_j| = 12 \quad |X| = 10 \quad |G| = 25 \quad \frac{S_b}{S_w}$$
$$CP(X) = \{c_0 \ldots c_n\}$$
$$O_o = crossoverType(x_j, x_i, P(crossover))$$
$$O_1 = mutate(x_j, P(mutation))$$

Figure 1: Experimental setup

The 12 bit chromosome represents 0 and 1 weightings for face features. Figure 2 depicts the bit positional mapping to features.

(Blinking, Smiling, Neutral, Glasses, Talking, Pose, Light, Flash, Needs Glasses, Gender, Mustache, Beard)

Figure 2: Chromosome bit position to face feature mapping.

4.1 Data

The Carnegie Mellon University Pose, Illumination, and Expression database consists of 41,368 images of 68 people from 13 camera angles with 43 illumination conditions and 4 expressions [10]. Over a 3 month period in the year 2000, all of the images were acquired in a specialized photo room. Each image contained descriptions for the following fields: facial expression, glasses, talking, pose, flash, light, glasses, sex, mustache, beard, age, and race [10]. The traits blinking, smiling, neutral, talking, glasses, mustache, beard, light, pose, flash, gender, and needs glasses were used in experimentation [10].

4.2 Agglomerative K-Means

An agglomerative K-Means clustering algorithm was implemented during experimentation. Initially, each data element was itself a cluster [8]. Sequentially, the data elements were evaluated for cluster membership by the Euclidean distance between its feature vector and each cluster's epicenter. If the sample's smallest distance from a cluster is less than an empirically determined threshold, the data sample is placed into the cluster. Otherwise, a new cluster is formed. The clustering algorithm continued until none of the epicenters moved within the cluster space.

Agglomerative K-means clustering pseudo code

Step 1: Through supervised learning, determine a threshold value

Step 2: Assign each member to a cluster

a) For each member:

i) Determine the minimum Euclidean distance to a cluster's 12 dimensional centroid

ii) If the distance is greater than the threshold,
a. Create a new cluster

b. Initialize the cluster's epicenter to the new feature vector

iii) Otherwise, add the member to the closest cluster

a. Recalculate the cluster 12 dimensional centroid

b) Determine if the cluster epicenters have changed

i) If true, goto step a.

ii) If false, goto step 3.

Step 3: Return the cluster space

The clustering threshold was calculated from Equation 11. A mean vector for the entire PIE dataset was calculated. Next, a mean distance from the data epicenter described the average member scatter. The α within equation 11 is a scalar that is multiplied with the PIE data's standard deviation, σ, to select as a threshold. A series of agglomerative K-Means threshold experiments were executed to determine the appropriate selection of alpha.

$$Threshold = \alpha\sigma, (\sigma \varepsilon \Re) \qquad (11)$$

4.3 Linear Discriminant Analysis

A LDA evaluation function maximized the ratio of the between cluster scatter and within cluster scatter. Martinez et al describe LDA as the ratio maximization of between class scatter and within class scatter also known as the Fisher criterion [7]. The evaluation function specified in Equation 12, measured the fisher property of cluster spaces generated by the agglomerative K-Means clustering algorithm.

$$\frac{\sum_{j=1}^{cb} (\mu_j - \mu)(\mu_j - \mu)^T}{cb} \qquad (12)$$

$$\sum_{j=1}^{Ca} \sum_{i=1} (x_i^b - \mu_j^a)(x_i^b - \mu_j^a)^T$$

Linear Discriminant Analysis pseudo code

Step 1: Retrieve all of the cluster spaces

Step 2: Calculate the LDA score for each cluster:

i) Calculate the within class scatter

ii) Calculate the between class scatter

iii) Compute the ratio of step ii to step i

Step 3: Return LDA scores

4.4 Evolutionary Computation in Java

Evolutionary Computation in Java (ECJ) is a research evolutionary computing system developed by George Mason University's Evolutionary Computation Laboratory [3]. ECJ provides genetic algorithms, generational evolution, flexible breeding architecture, multiple selection operators and population evolution that are defined within a parameter file and loaded at runtime [3].

4.5 Evolutionary Approach

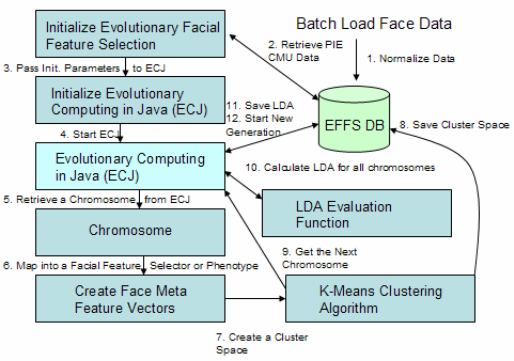


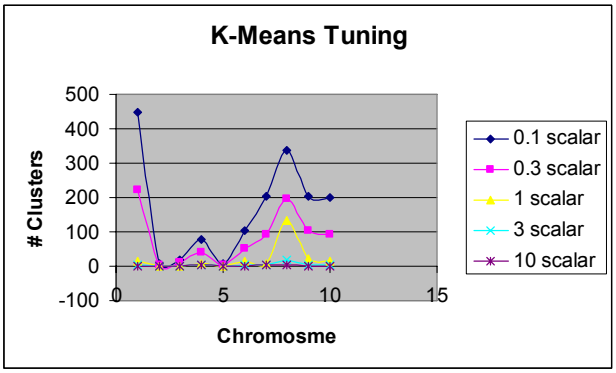
Figure 3: Evolutionary System Architecture

Figure 3 shows the overall system architecture of the evolutionary approach within this paper. All of the CMU PIE data is normalized and loaded into a database. The feature vectors of each face are retrieved from the database and placed into a system context. An ECJ parameter file is written to disk that is used to initialize the evolutionary environment. Initialization includes the generation of chromosomes. Each chromosome is retrieved from ECJ and mapped into a facial feature selector vector. The vectors are input into the agglomerative K-Means clustering algorithm. A cluster space is generated and tagged to a facial feature selector with the associated chromosome. The tagging and cluster spaces are saved into a database. After all cluster spaces are created and saved, the LDA evaluation function scores each cluster space. The LDA scores of the cluster spaces are stored into the database. ECJ crossed the top two chromosomes based on LDA score rank and continued in a pair wise mating scheme. The generation continued until all generations completed.

5. RESULTS

5.1 Agglomerative K-Means Results

A series of tuning experiments were run on all of the images from the PIE database to determine an appropriate agglomerative K-Means threshold. The threshold was the product of a weight and the standard deviation of the data set from a mean vector. Graph 1 depicts the results of changing the α weight.



Graph 1: K-Means tuning experiments

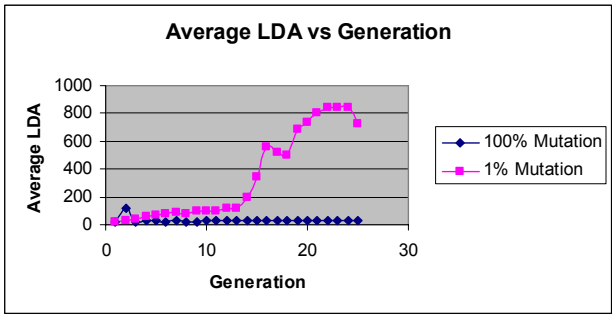
The 10 initial chromosomes were identical throughout all experiments.

The floor of the number of clusters was established with the 10 scalar while the ceiling was found with a 0.1 scalar. A scalar selection between 0.1 and 10 was desired to compromise between run time and the number of clusters. Since most of the elements of the centroid vector were in the interval [0-1], a scalar of 0.3 was chosen empirically. The 0.3 scalar produced a threshold of 1.02.

5.2 Genetic Algorithm Results

This paper provides results obtained on an evolutionary algorithm on a face dataset. Two sets of experiments were run on the full PIE database. Each experiment consisted of two batches.

Within each batch, one genetic operator was selected as an independent variable while all other genetic algorithm parameters were held constant. Between batches, the affect of the chosen genetic algorithm parameters were examined with respect to LDA and the average number clusters for each generation. In total, 4 different sets of parameters were chosen for the evolutionary facial feature selection.



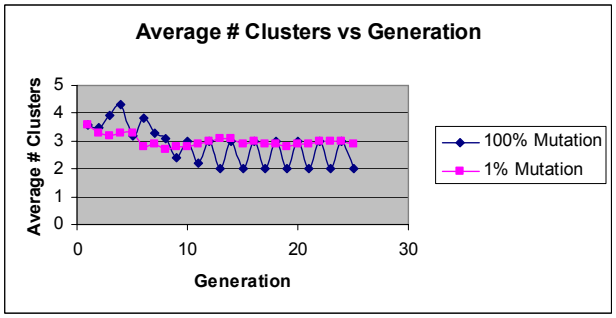
Graph 2: Mutation and LDA results

|G| = 25

Independent Variable: mutate(x_j,P(1)) or mutate(x_j,P(0.01))

Constant: onePoint(x_j,x_i,P(1))

Dependent Variables: LDA



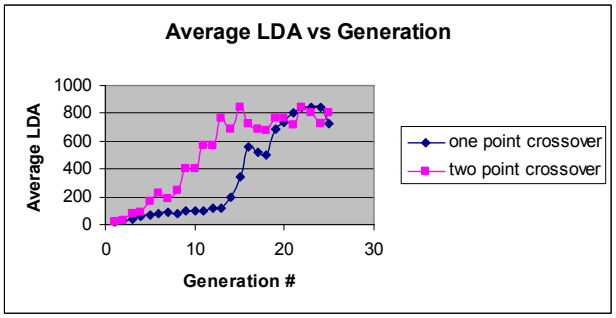
Graph 3: Mutation and average clusters results

|G| = 25

Independent Variable: mutate(x_j,P(1)) or mutate(x_j,P(0.01))

Constant: onePoint(x_j,x_i,P(1))

Dependent Variables: Average # clusters



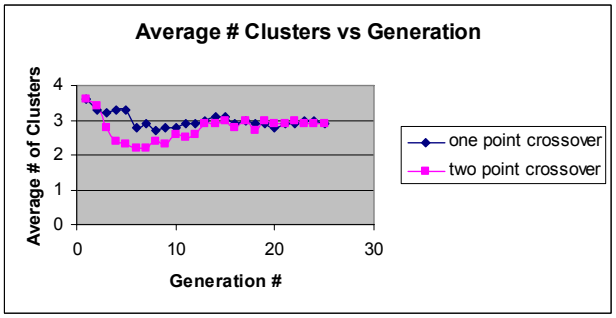
Graph 4: Crossover type and LDA results

|G| = 25

Independent Variable: onePoint(x_j,x_i,P(1)) or twoPoint(x_j,x_i,P(1))

Constant: mutate(x_j,P(0.01))

Dependent Variables: LDA



Graph 5: Crossover type and average clusters results

|G| = 25

Independent Variable: onePoint(x_j,x_i,P(1)) or twoPoint(x_j,x_i,P(1))

Constant: mutate(x_j,P(0.01))

Dependent Variables: Average # clusters

With one point crossover, 0.01 mutation and 25 generations, the best converged solution included the chromosome with blinking, smiling and pose. By generation 14, the best fit chromosome was evaluated to 845.5 and appeared once. The entire population consisted of the best fit chromosome by generation 22. Two point crossover yielded an identical best chromosome as one point crossover. By the fifth generation, the best fit chromosome appeared once. By generation 15, the best fit chromosome was the entire population. Two point cross over oscillated between the most fit chromosome and a second member that was three times less fit.

Two point crossover achieved the best fit chromosome faster than one point crossover. However, after achievement, the two point crossover operator diverged more than the single point crossover.

With a mutation of 1, one pointer crossover and 25 generations, the best chromosome oscillated between one that only contained blinking with a score of 25.42 and another that contained all traits except light with a score of 33.53. Neither score was close to a previous best of 845.5. When the mutation rate was changed to 0.01, the best converged solution contained blinking, smiling and pose with a score of 845.5. From table 1, the LDA evaluation results with a mutation rate of 0.01 was over a magnitude better than with 1.

The LDA evaluation score was 25.42 as contrasted to a previous best of 845.5. The first ranked chromosome was present 5 times. The second ranking chromosome contained blinking and neutral weights. A mutation rate of 0.01 with 10 generations produced a chromosome of blinking and smiling of rank 1 twice. The second ranking chromosome was present twice and contained weights for neutral, pose, flash and needs glasses. Both mutation rates of 0.01 and 1 contained chromosome 101000000 within the top two.

This indicates that a lower mutation rate is better than an extremely high rate with achieving the best chromosome. In addition, table 2 depicts the oscillation characteristic of mutation 1. Local optimum solutions were not annealed.

6. CONCLUSION

6.1 Conclusion and Future Works

This paper provides results obtained on facial meta data features from an evolutionary genetic algorithm. The algorithm used 12 face meta features as marked during face image acquisition. The experiments indicate that a mutation rate of 1% with a two point crossover converge to an optimal chromosome.

Future experiments will include the analysis of the effects of crossover and mutation selection as a whole. Additional work will include the examination of false positives versus true positives with the implementation of a facial recognition matcher. The comparison of receiver operator curves with and without the evolutionary algorithm provides a natural extension of this paper. For example, the classification of face images might decrease the number of false positives while decreasing the number of true positives.

7. ACKNOWLEDGMENTS

Special thanks to Jason Pelecanos, Nalini Ratha and Andrew Senior from IBM Research for guidance and reviews. In addition, special gratitude to Nelson Dilallo, IBM GS, for support and guidance through all phases of the paper process. The following people provided guidance and support through the publication process: Douglas Lashmit, Glenn Galfond, Fred Maymir-Ducharme, Barry Graham, Wanda Kiernan, Russell Patrick, Milton Harrison, Michael Hoffman and IBM management. The foundational work started at Johns Hopkins University with Assistant Research Professor John Sheppard and Graduate Student John Brown. Thank you Karen, Tim, Zach and Angelica.

8. REFERENCES

[1] "ABIS System Developer's Guide", L1, 050-138, Revision A, Software Version 4.1, 2005.

[2] "Biometrics Foundation Documents". NSTC, Subcommittee on Biometrics.

[3] ECJ. http://cs.gmu.edu/~eclab/tools.html. 12/04/2006.

[4] Germain, Robert S., Andrea Califano, Scott Colville. "Fingerprint Matching Using Transformation Parameter Clustering". IEEE CSE, 1070-9924/97.

[5] Lee, Hyung-Ji, Wan-Su Lee, Jae-Ho Chung. "Face Recognition Using Fisherface Algorithm and Elastic Graph Matching." IEEE 0-7803-6725-1/01.

[6] Liu, Chengjun. "Evolutionary Pursuit and Its Application to Face Recognition." IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 22, No. 6, June 2000.

[7] Martinez, Aleix, Avinash C. Kak. "PCA versus LDA". IEEE TPAM Vol. 23, No. 2, 02/2001.

[8] Mohanta, Partha Paratim, D.P. Mukherjee, Scott T. Acton. "Agglomerative Clustering for Image Segmentation". 1051-4651/02, IEEE 2002.

[9] Phillips, P. Johathan, W. Todd Scruggs, Alice O'Toole, Patrick J. Flynn, Kevin W. Bowyer, Cathy L. Schott, Matthew Sharpe. "FRVT 2006 and ICE 2006 Large-Scale Results". NISTIR 7408 2007.

[10] Sim, Terence and Simon Baker, Maan Bsat. "The CMU Pose, Illumination, and Expression (PIE) Database of Human Faces". CMU-RI-TR-01-02.

[11] "Universal Latent Workstation Manual" CJIS FBI Version 2.96.

[12] Wayman, James, Anil Jain, Davide Maltoni, Dario Maio. Biometric Systems: Technology, Design and Performance Evaluation, Springer 2005.

[13] Zhao, Bo, Zhongziang Zhu, Enrong Mao, Zhenghe Song. "Image Segmentation Based on Ant Colony Optimization and K-Means Clustering". IEEE ICAL 08/2007.