Decision-level Fusion in Fingerprint Verification

Salil Prabhakar and Anil K. Jain

Dept. of Comp. Sci. and Eng., Michigan State University, East Lansing, MI 48824

Abstract. A scheme is proposed for classifier combination at decision level which stresses the importance of classifier selection during combination. The proposed scheme is optimal (in the Neyman-Pearson sense) when sufficient data are available to obtain reasonable estimates of the join densities of classifier outputs. Four different fingerprint matching algorithms are combined using the proposed scheme to improve the accuracy of a fingerprint verification system. Experiments conducted on a large fingerprint database ($\sim$ 2,700 fingerprints) confirm the effectiveness of the proposed integration scheme. An overall matching performance increase of $\sim 3\%$ is achieved. We further show that a combination of multiple impressions or multiple fingers improves the verification performance by more than 4% and 5%, respectively. Analysis of the results provide some insight into the various decision-level classifier combination strategies.

1 Introduction

A number of classifier combination strategies exist [1], [2], [3], however, a priori it is not known which combination strategy works better than the others and if so under what circumstances. In this paper we will restrict ourselves to a particular decision-level integration scenario where each classifier may select its own representation scheme and produces a confidence value as its output. A theoretical framework for combining classifiers in such a scenario has been developed by Kittler et al. [2]. However, the product rule for combination suggested in [2] implicitly assumes an independence of classifiers. The sum rule further assumes that the aposteriori probabilities computed by the respective classifiers do not deviate dramatically from the prior probabilities. The max rule, min rule, median rule, and majority vote rule have been shown to be special cases of the sum and the product rules. Making these assumptions simplifies the combination rule but does not guarantee optimal results and hinders the combination performance. We follow Kittler et al.'s framework without making any assumptions about the independence of various classifiers.

The contributions of this paper are two fold. Firstly, we propose a general system design for decision-level classifier fusion that uses the optimal Neyman-Pearson rule and outperforms the combination strategies based on the assumption of independence among the classifiers. Secondly, we propose a multi-modal biometric system design based on multiple fingerprint matchers. The use of the proposed combination strategy in combining multiple matchers significantly improves the overall accuracy of the fingerprint-based verification system. The effectiveness of the proposed integration strategy is further demonstrated by building multi-modal biometric systems that combine two different impressions of the same finger or fingerprints of two different fingers.

2 Biometrics

The biometric verification problem can be formulated as follows. Let the stored biometric signal (template) of a person be represented as S and the acquired signal (input) for authentication be represented by I. Then the null and alternate hypotheses are:

$H_0 : I \neq S$, input fingerprint is not from the same finger as the template,

$H_1 : I = S$, input fingerprint is from the same finger as the template.

The associated decisions are as follows: D_0 : person is an imposter, and D_1 : person is genuine. The verification involves matching S and I using a similarity measure. If the matching score is less than some decision threshold T, then decide D_0, else decide D_1. Then, $FAR = P(D_1|w_0)$, and $FRR = P(D_0|w_1)$, where w_0 is the class with $H_0 = true$ and w_1 is the class with $H_1 = true$.

Several biometric systems have been designed and tested on large databases. However, in some applications with stringent performance requirement, no single biometric can meet the requirements due to inexact nature of sensing, feature extraction, and matching processes. This has generated interest in designing multimodal biometric systems. Multimodal biometric systems may work in one of the following five scenarios: (i) Multiple sensors: the information obtained from different sensors for the same biometric may be combined. For example, optical, ultrasound, and capacitance based sensors are available to capture fingerprints. (ii) Multiple biometric system: multiple biometrics such as fingerprint and face may be combined [2]. (iii) Multiple units of the same biometric: one image each from both the iris, or both hands, or ten fingerprints may be combined. (iv) Multiple instances of the same biometric: for example multiple impressions of the same finger may be combined. (v) Multiple representation and matching algorithms for the same input biometric signal: for example, combining different approaches to feature extraction and matching of fingerprints [4]. The first two scenarios require several sensors and are not cost effective. Scenarios (iii) causes inconvenience to the user in providing multiple cues and has a longer acquisition time. In scenario (iv), only a single input is acquired during verification and matched with several stored templates acquired during the one-time enrollment process. Thus, it is slightly better than scenario (iii). In our opinion, scenario (v) is the most cost-effective way to improve biometric system performance.

We propose to use a combination of four different fingerprint-based biometric systems where each system uses different feature extraction and/or matching algorithms to generate a matching score which can be interpreted as the confidence level of the matcher. These different matching scores are combined to obtain the lowest possible FRR for a given FAR. We also compare the performance of our integration strategy with the sum and the product rules [2]. Even though we propose and report results in scenarios (iii), (iv) and (v), our combination strategy could be used for scenarios (i) and (ii) as well.

3 Optimal Integration Strategy

Let us suppose that pattern Z is to be assigned to one of the two possible classes, w_0 and w_1. Let us assume that we have N classifiers, and the ith classifier outputs a single confidence value θ_i about class w_1 (the confidence for the class w_0 will be $1-\theta_i$), $i = 1, 2, .., N$. Let us assume that the prior probabilities for the two classes are equal. The classifier combination task can now be posed as an independent (from the original N classifier designs) classifier design problem with two classes and N features (θ_i, $i = 1, 2, .., N$).

3.1 Classifier Selection

It is a common practice in classifier combination to perform an extensive analysis of various combination strategies involving all the N available classifiers. In feature selection it is well known that the most informative d-element subset of N conditionally independent features is not necessarily the union of the d individually most informative features. Cover [5] argues that no non-exhaustive sequential d-element selection procedure is optimal, even for jointly normal features. He further showed that all possible probability of error ordering can occur among subsets of features subject to a monotonicity constraint. The statistical dependence among features causes further uncertainty in the d-element subset composed of the individually best features. One could argue that the combination strategy itself should pick out the classifiers that should be combined. However, we know in practice that the "curse of dimensionality" makes it difficult for a classifier to automatically delete less discriminative features [6]. Therefore, we propose a classifier selection scheme prior to classifier combination. We propose to use the *class separation* statistic [7] as the feature effectiveness criterion. This statistic, CS, measures how well the two classes (imposter and genuine, in our case) are separated with respect to the feature vector, X^d, in a d-dimensional space, R^d.

$$CS(X^d) = \int_{R^d} |p(X^d|w_0) - p(X^d|w_1)| dx, \tag{1}$$

where $p(X^d|w_0)$ and $p(X^d|w_1)$ are the estimated distributions for the w_0 (imposter) and w_1 (genuine) classes, respectively. Note that $0 \leq CS \leq 2$.

We will use the class separation statistic to obtain the best feature subset using an exhaustive search of all possible $2^N - 1$ feature subsets.

3.2 Non-parametric density estimation

Once we have selected the subset containing d ($d \leq N$) features, we develop our combination strategy. We do not make any assumptions about the form of the distributions for the two classes and use non-parametric methods to estimate the two distributions. We use Parzen window density estimate to obtain the non-parametric distributions [8]. A Gaussian kernel was used and the window width was empirically determined.

3.3 Decision Strategy

We use the likelihood ratio $L = P(X^d|w_0)/P(X^d|w_1)$ to make the final decision for our two-class problem: Decide D_0 (person is an imposter) for low values of L; decide D_1 (person is genuine) for high values of L. If L is small, the data is more likely to come from class w_1; the likelihood ratio test rejects the null hypothesis for small values of the ratio. The Neyman-Pearson lemma states that this test is optimal, that is, among all the tests with a given significance level, α (FAR), the likelihood ratio test has the maximum power. For a specified α, λ is the smallest constant such that $P(L \leq \lambda) \leq \alpha$. The type II error, β (FRR), is given by $P(L > \lambda)$.

4 Matching Algorithms

We have developed four fingerprint verification systems which can be broadly classified into two categories: (i) minutiae-based, and (ii) filter-based. The three minutiae-based and one filter-based algorithms are summarized in this section.

4.1 Matcher *Hough*

The fingerprint matching problem can be regarded as template matching [10]: given two sets of minutia features, compute their matching score. The two main steps of the algorithm are: (1) Compute the transformation parameters δ_x, δ_y, θ, and s between the two images, where δ_x and δ_y are translations along x- and y- directions, respectively, θ is the rotation angle, and s is the scaling factor; (2) Align two sets of minutia points with the estimated parameters and count the matched pairs within a bounding box; (3) Repeat the previous two steps for the set of discretized allowed transformations. The transformation that results in the highest matching score is believed to be the correct one. The final matching score is scaled between 0 and 99. Details of the algorithm can be found in [10].

4.2 Matcher *String*

Each set of extracted minutia features is first converted into polar coordinates with respect to an anchor point. The two-dimensional (2D) minutia features are, therefore, reduced to a one-dimensional (1D) string by concatenating points in a increasing order of radial angel in polar coordinate. The string matching algorithm is applied to compute the edit distance between the two strings. The edit distance can be easily normalized and converted into a matching score. This algorithm [9] can be summarized as follows: (1) Rotation and translation are estimated by matching ridge segment (represented as planar curve) associated with each minutia in the input image with the ridge segment associated with each minutia in the template image. The rotation and translation that results in the maximum number of matched minutiae pairs within a bounding box is considered the correct transformation and the corresponding minutiae are labeled as anchor minutiae, $\mathcal{A}_1$ and $\mathcal{A}_2$, respectively. (2) Convert each set of minutia into a 1D string using polar coordinates anchored at $\mathcal{A}_1$ and $\mathcal{A}_2$, respectively; (3) Compute the edit distance between the two 1D strings. The matched pairs are retrieved based on the minimal edit distance between the two strings; (4) Output the normalized matching score which is the ratio of the number of matched-pairs and the number of minutiae points in the two sets.

4.3 Matcher *Dynamic*

This matching algorithm is a generalization of the above mentioned string algorithm. The transformation of a 2D pattern into a 1D pattern usually results in a loss of information. Chen and Jain [11] have shown that fingerprint matching using 2D dynamic time warping can be done as efficiently as 1D string editing while avoiding the above mentioned problems with algorithm *String*. The 2D dynamic time warping algorithm can be characterized by the following steps: (1) Estimate the rotation between the two sets of minutia features as in Step 1 of algorithm *String*; (2) Align the two minutia sets using the estimated parameters from Step 1; (3) Compute the maximal matched minutia pairs of the two minutia sets using 2D dynamic programming technique. The intuitive interpretation of this step is to warp one set of minutia to align with the other so that the number of matched minutiae is maximized; (4) Output the normalized matching score which is based on only those minutiae that lie within the overlapping region.

4.4 Matcher *Filter*

The four mains steps in the filter-based feature extraction algorithm are: (i) determine a reference point and region of interest for the fingerprint image. The reference point is taken to be the center point in a fingerprint which is defined as the point of maximum curvature of the ridges in a fingerprint. The region of interest is a circular area around the reference point. The algorithm rejects the fingerprint images for which the reference point could not be established. (ii) tessellate the region of interest. The region of interest is divided into sectors and the gray values in each sector are normalized to a constant mean and variance. (iii) filter the region of interest in eight different directions using a bank of Gabor filters (eight directions are required to completely capture the local ridge characteristics in a fingerprint while only four directions are required to capture the global configuration). Filtering produces a set of eight filtered images. (iv) compute the average absolute deviation from the mean (AAD) of gray values in individual sectors in each filtered image. AAD value in each sector quantifies the underlying ridge structures and is defined as a feature. A feature vector, which we call FingerCode, is the collection of all the features (for every sector) in each filtered image. Thus, the feature elements capture the local information and the ordered enumeration of the tessellation captures the invariant global relationships among the local patterns. The representation is invariant to translation of the image. It is assumed that the fingerprint is captured in an upright position and the rotation invariance is achieved by storing ten representations corresponding to the various rotations ($-45.0°$, $-45°$, $-33.75°$, $-22.5°$, $-11.25°$, $0°$, $11.25°$, $22.5°$, $33.75°$, $45.0°$) of the image. Euclidean distance is computed between the input representation and the ten templates to generate ten matching distances. Finally, the minimum of the ten distances is computed and inverted to give a matching score. The matching score is scaled between 0 and 99 and can be regarded as a confidence value of the matcher.

5 Experimental Results

Fingerprint images were collected in our laboratory from 167 subjects using an optical sensor manufactured by Digital Biometrics, Inc. (image size = 508×480, resolution = 500 *dpi*). A single impression each of the right index, right middle, left index, and left middle fingers for each subject was taken in that order. This process was then repeated to acquire a second impression. The fingerprint images were collected again from the same subjects after an interval of six weeks in a similar fashion. Thus, we have four impressions for each of the four fingers of a subject. This resulted in a total of $2,672$ ($167 \times 4 \times 4$) fingerprint images. We call this database MSU_DBI. A live feedback of the acquired image was provided and the subjects were guided in placing their fingers in the center of the sensor in an upright position. A total of 100 images (about 4% of the database) was removed from the MSU_DBI because the filter-based fingerprint matching algorithm rejected these images due to failure in locating the center or due to a poor quality of the images. We matched all the remaining $2,572$ fingerprint images with each other to obtain $3,306,306$ ($\frac{2572 \times 2571}{2}$) matchings and called the matchings genuine only if the pair are different impressions of the same finger. Thus, we have a total of $3,298,834$ ($3,306,306 - 7,472$) imposter and $7,472$ genuine matchings per matcher from this database. For the multiple matcher combination, we randomly selected half the imposter matching scores and half the genuine matching scores for training and the remaining samples for test. This process was repeated ten times to give ten different training sets and ten corresponding independent test sets. All performances will be reported in terms of ROC curves computed as an average from the ten ROC curves corresponding to the ten different training and test sets (two-fold cross validation repeated ten times). For the multiple impression and multiple finger combinations, the same database of $3,298,834$ imposter and $7,472$ genuine matchings computed using the *Dynamic* matcher was used.

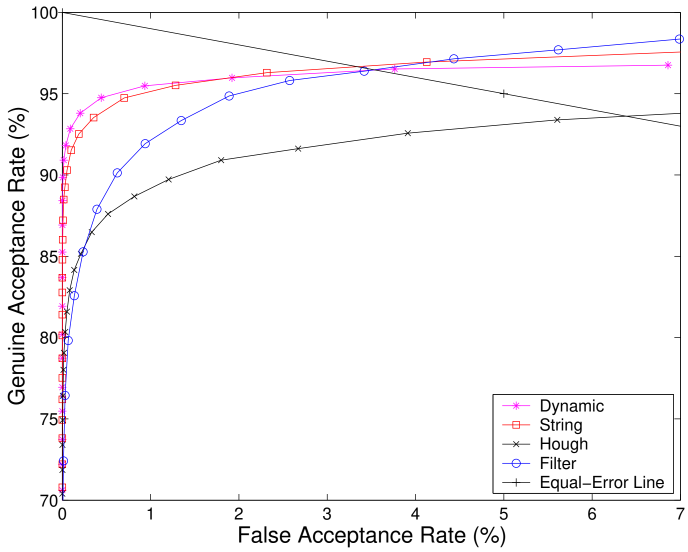

Fig. 1. Performance of individual fingerprint matchers.

The ROC curves computed from the test data for the four individual fingerprint matchers used in this study are shown in Figure 1. The class separation statistic computed from the training data was 1.88, 1.87, 1.85 and 1.76 for the algorithms *Dynamic*, *String*, *Filter*, and *Hough*, respectively, and is found to

Combination	CS (rank)	rank by ROC	ρ	rank by Δ ROC
$String + Filter$	1.95 (1)	1	0.52	2
$Dynamic + Filter$	1.95 (1)	2	0.56	3
$String + Dynamic$	1.94 (3)	2	0.82	3
$Hough + Dynamic$	1.93 (4)	4	0.80	6
$Hough + Filter$	1.91 (4)	6	0.53	1
$Hough + String$	1.90 (6)	5	0.83	5

Table 1. Combining two fingerprint matchers. CS is the class separation statistic. CS and ρ are computed from the training data. Ranks by ROC and ranks by Δ ROC are computed from the independent test data.

be highly correlated to the matching performance on the independent test set. Figure 1 shows that matchers $Dynamic$ and $String$ are comparable, $Filter$ is better than $Dynamic$ and $String$ at high FARs and slightly worse at very low FARs, and matcher $Hough$ ranks the last.

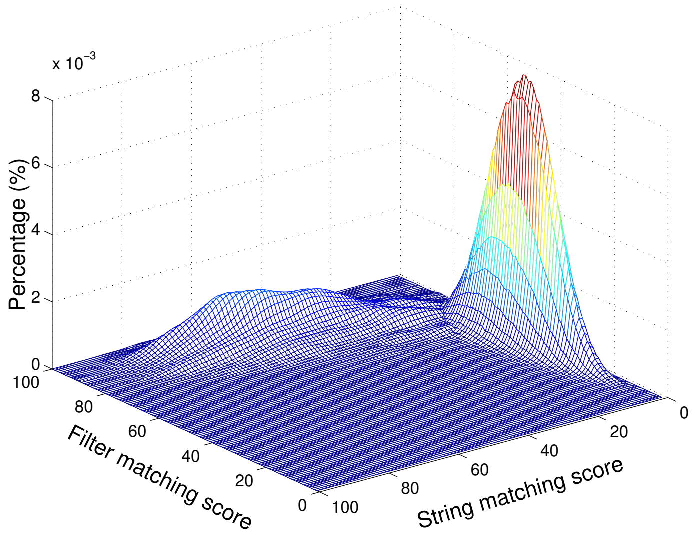

Fig. 2. Two-dimensional density estimates for the genuine and imposter classes for $String$+$Filter$ combination. Genuine density was estimated using Parzen window (h = 0.01) estimator and the imposter density was estimated using normalized histograms.

First, we combine the matchers in pairs of two. To combine two fingerprint matchers, we estimate the two-dimensional genuine and imposter densities from the training data. The two-dimensional genuine density was computed using the Parzen density estimation method. The value of window width (h) was empirically determined to obtain a smooth density estimate and was set at 0.01. We used the same value of h for all the two-matcher combinations. As a comparison, the genuine density estimates obtained from the normalized histograms were extremely peaky due to unavailability of sufficient data (only about $3,780$ genuine matching scores were available in the training set to estimate a two-dimensional distribution in $10,000$ (100×100) bins). However, for estimation of the two-dimensional imposter distribution, over 1.6 million matching scores were available. Hence, we estimated the two-dimensional imposter distribution by computing a normalized histogram using the following formula:

$$p(X^d|w_0) = \frac{1}{n}\sum_{j=1}^{n}\delta(X, X_j) \tag{2}$$

where δ is the delta function that equals 1 if the raw matching score vectors X and X_j are equal, 0 otherwise. Here n is the number of imposter matchings from the training data. The computation time for Parzen window density estimate depends on n and so, it is considerably larger than the normalized histogram method for large n. The estimates of the two-dimensional genuine and imposter densities thus computed for $String + Filter$ combination are shown in Figure 2. The class separation statistic for all pairs of matcher combination is shown in the second column of Table 1; the number in parenthesis is the predicted ranking of the combination performance based on CS. The actual ranking of performance obtained from the independent test set is listed in the third column marked ROC for ROC curves). As can be seen, the predicted ranking is very close to the actual rankings on independent test data.

The following observations can be made from the two-matcher combinations:

- Classifier combination improvement is directly related to the "independence" (lower values of ρ) of the classifiers.
- Combining two weak classifiers results in a large performance improvement.
- Combining two strong classifiers results in a small performance improvement.
- The two individually best classifier do not form the best pair.

Next, we combine the matchers in groups of three and then combine all the four matchers together. The class separation statistic is maximum (1.97) for the $String + Dynamic + Filter$ combination. From the tests conducted on the independent data set, we make following observations (see Figure 3).

- Adding a classifier may actually degrade the performance of classifier combination. This degradation in performance is a consequence of lack of independent information provided by the classifier being added and finite size of the training and test database.
- Classifier selection based on a "goodness" statistic is a promising approach.
- Performance of combination is significantly better than the best individual matcher.

Among all the possible subsets of the four fingerprint matchers, the class separation statistic is maximum for $String + Dynamic + Filter$ combination. Hence, our feature selection scheme selects this subset for the final combination and rejects the matcher $Hough$. This is consistent with the nature of the $Hough$ algorithm, which is basically the linear pairing step in algorithms $String$ and $Dynamic$, without the capability of dealing with elastic distortions. Therefore, $Hough$ does not provide "independent" information with respect to $String$ and

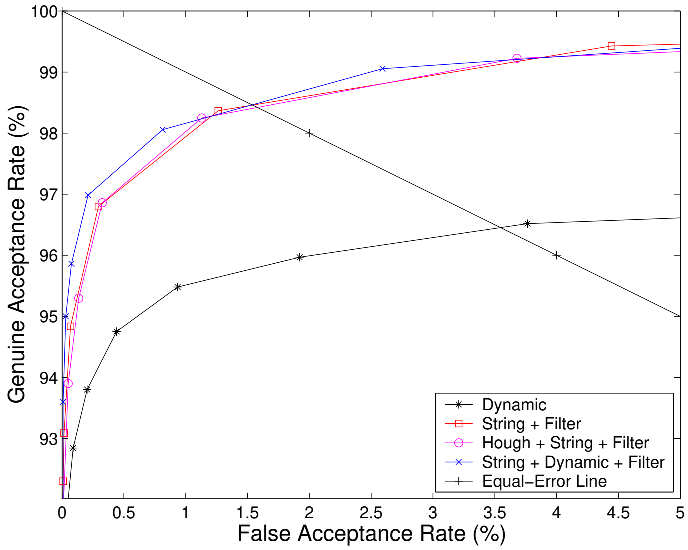

Fig. 3. The performance of the best individual matcher *Dynamic* is compared with the various combinations. The *String* + *Filter* is the best two-matcher combination and *String* + *Dynamic* + *Filter* is the best overall combination. Note that addition of the classifier *Hough* to the combination *String* + *Filter* results in a degradation of the performance.

Dynamic. The performance of the various matcher combinations on an independent test supports the prediction that *String* + *Dynamic* + *Filter* is the best combination. The proposed combination scheme either outperforms or matches the performance of the sum rule and outperforms the product rule in all the two- three- and four-matcher combinations because the proposed technique can produce a nonlinear complex decision boundary that is close to optimal.

The performance of the combined system is more than 3% better than the best individual matcher. The matcher combination takes about 0.02 seconds on an Sun Ultra 1 in the test phase. In an authentication system, this increase in time will have almost no effect on the verification time and the overall matching time is still bounded by the slowest individual matcher.

The performance improvement due to combination of two impressions of the same finger and the combination of two different fingers of the same person using the proposed strategy is 4% and 5%, respectively. The matcher *Dynamic* was used. The correlation coefficient between the two scores from two different impressions of the same finger is 0.42 and between two different fingers of the same person is 0.68 and is directly related to the improvement in the performance of combination. The CS for individual impressions is 1.84 and 1.87 respectively, and for the combination is 1.95. The CS for individual fingers is 1.87 and 1.86 respectively, and for the combination is 1.98. Combination of two impressions of the same finger or two fingers of the same person using the proposed combination strategy is extremely fast. Therefore, the overall verification time is same as the individual matcher *Dynamic*.

6 Conclusions and Discussions

We have presented a scheme for combining multiple matchers (classifiers) at decision-level in an optimal fashion. Our design emphasis is on classifier selection before arriving at the final combination. It was shown that one of the fingerprint matchers in the given pool of matchers is redundant and no performance improvement is achieved by utilizing this matcher. This matcher was identified and rejected by the matcher selection scheme. In case of a larger number of classifiers and relatively small training data, a classifier may actually degrade the performance when combined with other classifiers, and hence classifier selection is essential. We demonstrate that our combination scheme improves the performance of a fingerprint verification system by more than 3%. We also show that combining multiple instances of the same biometric or multiple units of the same biometric characteristics is a viable way to improve the verification system performance. We observe that independence among various classifiers is directly related to the improvement in performance of the combination.

References

1. A. K. Jain, R. P. W. Duin and J. Mao, "Statistical Pattern Recognition: A Review", *IEEE Transactions on Patt. Anal. and Machine Intell.*, Vol. 22, No. 1, pp. 4-37, 2000.
2. J. Kittler, M. Hatef, R. P. W. Duin, and J. Matas, "On Combining Classifiers", *IEEE Trans. on Patt. Anal. and Machine Intell.*, Vol. 20, No. 3, pp. 226-239, 1998.
3. T. K. Ho, J. J. Hull, and S. N. Srihari, "Decision Combination in Multiple Classifier Systems", *IEEE Trans. on Patt. Anal. and Machine Intell.*, Vol. 16, No. 1, pp. 66-75, 1994.
4. A. K. Jain, S. Prabhakar, and S. Chen, "Combining Multiple Matchers for a High Security Fingerprint Verification System", Pattern Recognition Letters, Vol 20, No. 11-13, pp. 1371-1379, November 1999.
5. T. M. Cover, "On the Possible Ordering in the Measurement Selection Problem", *IEEE Trans. on Systems, Man, and Cybernetics*, Vol. SMC-7, No. 9, pp. 657-661, 1977.
6. S. Raudys and A. K. Jain, "Small Sample Size Effects in Statistical Pattern Recognition: Recommendations for Practitioners", *IEEE Trans. on Patt. Anal. and Machine Intell.*, Vol. 13, No. 3, pp. 252-264, 1991.
7. I.-S. Oh, J.-S Lee, and C. Y. Suen, "Analysis of Class Separation and Combination of Class-Dependent Features for Handwriting Recognition", *IEEE Trans. Patt. Anal. and Machine Intell.*, Vol. 21, No. 10, pp. 1089-1094, 1999.
8. R. O. Duda, P. E. Hart, and D. G. Stork, Pattern Classification, 2nd Edition, John Wiley & Sons, November 2000.
9. A.K. Jain, L. Hong, S. Pankanti and R. Bolle, "An Identity Authentication System Using Fingerprints", *Proc. IEEE*, Vol. 85, No. 9, pp. 1365-1388, 1997.
10. N. K. Ratha, K. Karu, S. Chen, and A. K. Jain, "Real-time Matching System for Large Fingerprint Database", *IEEE Trans. on Patt. Anal. and Machine Intell.*, Vol. 18, No. 8, pp. 799-813, 1996.
11. S. Chen and A. K. Jain, "A Fingerprint Matching Algorithm Using Dynamic Programming", *Technical Report*, Department of Computer Science and Engineering, Michigan State University.
12. A. K. Jain, S. Prabhakar, L. Hong, and S. Pankanti, "Filterbank-based Fingerprint Matching", *IEEE Trans. on Image Proc.*, Vol. 9, No. 5, pp. 846-859, May 2000.