Cascaded Filtering for Fingerprint Identification using Random Projections

Atif Iqbal and Anoop Namboodiri
International Institute of Information Technology
Hyderabad, 500032, India
atif.iqbal@research.iiit.ac.in, anoop@iiit.ac.in

Abstract

Biometric identification often involves explicit comparison of a probe against each template stored in a database. This process becomes extremely time-consuming as the size of the database increases. Filtering approaches use a light weight comparison to select a smaller set of candidate templates from the database for explicit comparison. However, most existing filtering schemes use specific features that are hand-crafted for the biometric trait at each stage of the filtering. In this work, we explore the effectiveness of weak features in a cascade for filtering fingerprint databases. We start with a set of potential indexing features computed from minutiae triplets and minutiae quadruplets. Each stage of filtering consists of projecting the probe onto a specific line and the removal of database samples outside a window around the probe. The critical problem in this process is the selection of lines for projection at each stage of the filtering. We show that by using a set of random lines and the proposed fitness function, one can achieve better results that optimization methods such as PCA or LDA. Experimental results show that using an ensemble of projections we can reduce the penetration to 26% *at a hit rate of* 99%*. As each stage of the cascade is extremely fast, and filtering is progressive along the cascade, one can terminate the cascade at any point to achieve the desired performance. One can also combine this method with other indexing methods to improve the overall accuracy and speed. We present detailed experimental results on various aspects of the process on the FVC 2002 dataset.*

1. Introduction

Biometric identification is becoming a critical problem with the emergence of large scale biometric implementations across the world. Among the biometric traits, fingerprints are the most widely studied and accepted form in identification systems. Most biometric systems that scale to national population uses fingerprints as the one of the modalities for identification due to the ease of acquisition, amount of discriminative information available in fingerprints [18], acceptability in legal situations, as well as the availability of low cost devices for authentication purposes [10].

Matching of two fingerprint images is a computationally demanding task due to non-linear deformation of the skin during the acquisition process. The problem is compounded for identification tasks in large databases. To reduce the amount of matching to be performed, a common approach that is employed is the classification of the fingerprints into a set of basic classes [14, 12, 4]. All fingerprints in the database are classified into one of the basic classes (loops, whorl, arches) and stored in partially overlapping partitions. The input fingerprint is also classified, and is only compared against the fingerprints of the corresponding class in the partial database. If fingerprints were equally distributed into say five classes, the penetration rate would be reduced to $P = 0.2$. Therefore, the processing time and the False Identification Rate (FIR) would be reduced. However, as the number of classes is small and the fingerprints are unequally distributed among them (more than 90% of the fingerprints are either right loops, left loops or whorl [17]) the penetration is usually larger. Furthermore, the classification error and rejected fingerprints must be considered when classification is performed automatically. These factors reduce the effectiveness of classification based approach to narrow down the search space.

Fingerprint indexing algorithms reduce the number of comparison by selecting the most probable candidates and sorting them by the similarity to the input [5]. As indexing techniques perform better than exclusive classification considering the size of space that need to be searched [2], many indexing algorithms have been proposed recently. Germain *et al*. proposed a flash algorithm for fingerprint indexing [6]. Bebis *et al*. proposed the Delaunay triangulation of minutia points to perform fingerprint indexing [1]. Boer *et al*. used the registered directional field estimate, FingerCode and minutiae triplet along with their combination to index fingerprint databases [3]. Bhanu and Tan [2] generated minutiae triplets and used angles, handedness, type, direction, and maximum side as the features for indexing. They also applied some constraints on minutiae selection to avoid spurious minutiae. Jain *et al.* [11] use the features around a core point of a Gabor filtered image to realize indexing. Another indexing algorithm was proposed based on correlation of the robustly detected singular points in [15].

Most of the indexing methods available for the fingerprints are based on the detection of core and delta points. The accuracy of the entire system is dependent on the accuracy of detection of singular points. Other indexing schemes rely on alignment of the fingerprints for a compact representation, and the indexing accuracy is often dependent on the quality of alignment.

Random linear projections has already been used for the dimensionality reduction. It is en effective substitute for Principal Component Analysis and Linear Discriminant Analysis and is often computationally desirable. In other problems such as unsupervised learning, the objective function is either not defined or cannot be optimized analytically. The distance preserving nature of linear projections into random subspaces were explored by Johnson and Lindenstrauss [13] in 1984 (JL Theorem), who showed that random projections preserve the structure of high dimensional data well in lower dimensions. Specifically, the distortion in distances, when mapping n p-dimensional points into a q-dimensional random subspace, where $q \geq O(\log(n)/\epsilon^2)$ is less than a factor of $1+\epsilon$. Another advantage that random projection has over the other methods of dimensionality reduction is that we do not have to recalculate the objective function when a new sample is added to the database.

Random projections have also been used in biometric verification to derive lower dimensional feature representations of modalities such as face [7] and in palmprint and Iris Recognition[9]

Principal Component Analysis (PCA) preserves dimensions with maximum variance for the given data points (Y) and hence are potential candidates for projection for filtering. The first principal component, w_1 is obtained as:

$$w_1 = argmax_{||w||=1} Var\{Y^T w\} \tag{1}$$

While PCA is good for minimizing the error in representation of data in low dimensions, it does not promote the separation of classes in the projected subspace. This negatively affects the filtering performance on each projection. Linear Discriminant Analysis (LDA) on the other hand utilizes the class labels of the data and tries to maximize the ratio of between-class variance to the within-class variance in any particular data set, thereby guaranteeing maximal separability. LDA considers maximizing the following objective function:

$$J(w) = \frac{w^T S_B w}{w^T S_W w}, \tag{2}$$

where S_B and S_W are the between and within class scatter matrices. However, the objective function that we would like to maximize in the case of filtering is not that of LDA as it equally penalizes the increase in within class variance and decrease in between class variance. For a sequential cascade of filtering projection, one need to ensure that the samples within the same class are not filtered out even at the cost of reducing the filtering rate of non-matching classes.

In this work we propose a new objective function for evaluation of the suitability of projections for the filtering cascade and provide a method for designing a cascade based on random projections.

2. Feature Extraction

One of the major problems with fingerprint identification is that the feature vector is variable in length. Different samples of fingerprints from the same user can have different number of minutiae extracted. This prohibits us from using any indexing method that assumes that the pattern is a point in a Euclidean feature space. Another problem faced in fingerprint identification is the lack of alignment of the query with samples in the database.

To overcome the issue with minutiae representation, various fixed length feature representations have been proposed such as low order Delaunay triangulation [1], minutiae triplets [2], and Finger Code [11]. In this work, we start with the assumption that a set of fixed length features are available for representing each fingerprint sample, and derive a method for efficient filtering using the given set of features.

We have currently used two different sets of features and they are concatenated together so that every finger is represented using a fixed length feature vector. The first set of features are extracted from the triangles formed by minutiae in the image (referred to as minutiae triplets), and the second set of features are extracted from the quadrilaterals formed from the geometrical locations of minutiae, as proposed in [8].

To extract the features from triangles, the largest side, l (see Figure 1), and the two angles (α_1 and α_2) that involve l are computed. The feature representation of each triangle is (l, α_1, α_2), where $(\alpha_1 < \alpha_2)$.

The features from minutiae quadruplets is as proposed by Iloanusi *et al.* [8], which involves 7 features $< \varphi_1, \varphi_2, \delta_1, \delta_2, \rho_1, \rho_2, \eta >$ (see Figure 2). The two features φ_1 and φ_2 are the differences of two opposite angles in the quadrilateral, and δ_1,δ_2 are the lengths of the two diagonals. ρ_1 and ρ_2 are the heights of the parallelogram.

The last feature η is a global feature, and is a combination of sides and area of the quadrilaterals.

$$\eta = 100 \log_{10}(\tau\nu), \tag{3}$$

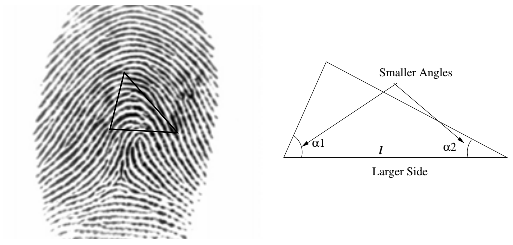

Figure 1. Triangles formed from the minutiae and the extracted features.

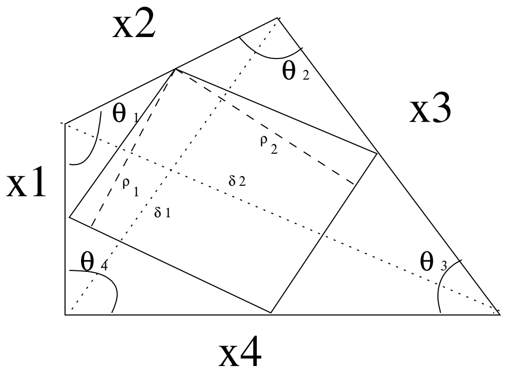

Figure 2. A minutia quadruplets

where

$$\tau = \sqrt{A_p} + \sqrt[4]{x_1 \times x_2 \times x_3 \times x_4} \tag{4}$$

$$\nu = \sqrt{A_q} + \sqrt{y_1 \times y_2} \tag{5}$$

A_p is the area of the parallelogram, x_1, x_2, x_3 and x_4 are the lengths of the sides of quadrilateral. A_q is the area of the quadrilateral, and y_1 and y_2 are the length of the sides of the parallelogram. We have removed concave quadrilaterals and all quadrilaterals with crossed edges were uncrossed to form regular convex quadrilaterals.

The above procedure gives us the features for each triangle and quadrilateral. The number of triangles and quadrilaterals vary considerably between images as it depends on the number of minutiae that are detected. In our experiment, we fixed the number of triangles and quadrilaterals that were selected, which were empirically set to be 800 and 1200 respectively. To reduce the influence of deformations in fingerprints, we concentrate on local minutiae structure, and hence only the smaller triangles and quadrilaterals are considered in computing the feature vectors. In short, the smallest 800 and 1200 triangles and quadrilaterals were used in construction of the final feature vector.

The final feature vector contains the frequency count of different triangles and quadrilaterals present in the fingerprint. To determine this, we first learn the most prominent k clusters are determined from the training samples for both triangles and quadrilaterals using $k-means$ clustering in the corresponding feature spaces. Let c_k denote the k^{th} cluster.

To extract the feature from images, its triplets $\{t_w | w = 1, 2, .., Q\}$ are assigned to the nearest cluster based on the Euclidean distance to the cluster centers.

$$\text{Assign } t_w \text{ to } c_k, \text{ if } k = argmin\{|t_w - n_j|, j = 1..k\} \tag{6}$$

Here, n_j is the centroid of the j^{th} cluster and c_k is the cluster id. Each triplet is assigned to a single cluster. The feature vector is constructed by counting the number of triplets assigned to each cluster. Thus, the feature vector of image Y is $F_t(Y) = \{a_1^Y, a_2^Y, \ldots, a_k^Y\}$, where a_i^Y is the number of triplets from image Y that are assigned to cluster i and k is the total number of clusters. We will refer to a_i as an accumulator for centroid i. The quadruplets based features are also converted to a histogram using the accumulation process as above.

In our experiments, the numbers of clusters k was empirically set to 50 for both triplets and quadruplets, resulting in feature vectors of length 50 for each. The two feature vectors are then concatenated to form a single 100 dimensional feature vector. The resulting feature vector may be written as the concatenation: $F(Y) = [F_t(Y)\ F_q(Y)]$. As the individual features are rotation invariant, there is no need of alignment between samples for the purpose of matching.

3. Filtering with Projections

As mentioned in the introduction, our goal is to develop a fast filtering strategy using a given set of features. All filtering methods make use of an efficient comparison strategy to match a query with all samples in the gallery and quickly eliminate any sample that is unlikely to match with the query during explicit comparison. Since our primary goal is efficiency, we explore the simplest form of comparison, where the representation of each sample is a single real number. In other words, we want to develop a function $\mathcal{P}_j()$ that maps the feature vector $F(x)$ of a fingerprint image x:

$$v_j(Y) = \mathcal{P}_j(F(Y)), \tag{7}$$

where $v_j(Y) \in \mathcal{R}$. $\mathcal{P}_j()$ may be thought of as a projection from the d-dimensional feature vector $F(Y)$ to a one-dimensional feature space. As we require a cascade of filtering stages, there should be a different projection for each stage and is indexed by the subscript j of $\mathcal{P}_j$. Using a one-dimensional representation for each sample in a filtering stage allows us to carry out extremely fast comparisons of samples at each stage.

We explore the use of linear projections due to i) the efficiency of projection, ii) the ability of linear projections to capture structure present in high-dimensional representations [13], and iii) robustness of linear mappings to avoid overfitting during the learning process.

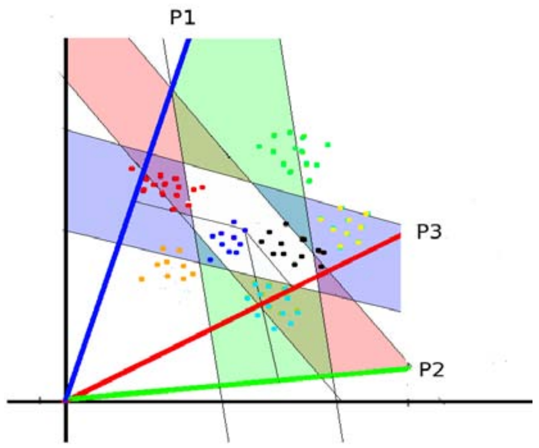

Figure 3. Cascading random projections: P1, P2 and P3 are three projections used in a sequence. Samples that are not falling within a window of the probe are removed at each stage.

A random projection is a weak but efficient representation of a biometric trait. If the samples of a class are within a bounding sphere of radius r in the feature space, they will be within a window of size $2r$ in any linear projection. Hence at each stage of the cascade, we discard all the gallery samples that are outside a window of size d around the projection of the query. Figure 3 shows the result of projection of a set of two dimensional samples on to three random lines and discarding the samples that lie outside a window. The white polygon in the middle represents the samples that are selected from this three-stage cascade. Although the final set of samples that are selected are independent of the order of projections, the efficiency of the cascade is clearly dependent on it. If we can use the projections that remove a large number of imposters in the initial stages of the cascade, the number of comparisons at later stages can be minimized.

The property that we like to maximize is close but not identical to the Fisher criterion: S_B/S_w (see Equation 2).

$$c_i = \frac{\sum\limits_{j \notin W} \neg S(j)}{\sum_j \neg S(j)}; f_i = \frac{\sum\limits_{j \notin W} S(j)}{\sum_j S(j)}, \tag{8}$$

where $S(j)$ is an indicator variable that takes a value 1, when j is of the same class as the probe. The score of the i^{th} projection is defined as the ratio:

$$Score_i = \frac{c_i}{1+f_i}. \tag{9}$$

The size of the window to be selected for each projections is first calculated from the training samples. The size is chosen in such a way that no sample that belongs to the same class as probe, should be left out. Note that this quantity is computed at each stage after leaving out the samples that are filtered out in the previous stages of the cascade. Algorithm 1 shows the algorithm for learning the cascade.

Algorithm 1 Learning the Projection Cascade.

```
for $i$ ⇐number of projections do
    for each projection $P_j$ do
        if the projections not selected then
            calculate the Score based on equation 9
        end if
    end for
    select the projection with the highest score and remove
    the samples that fall outside the window.
end for
Return the sorted order of projections
```

The indexing score is usually defined in two parameters– Hit Rate and Penetration Rate. The hit rate is defined as the probability that the correct user identity is retrieved. The penetration rate defines the fraction of user identities retrieved from the database upon presentation of the query print. At each stage of the cascade, as set of samples are filtered out, which might include some that belong to the same class as the query. We declare a miss if the set of samples after filtering does not contain any sample that belongs to the same class as the probe. After each stage, the miss rate tends to increase and the penetration rate decreases. One can decide to stop the filtering cascade after stage s, depending on these values.

In practice we note that the initial stages of the cascade results in large reduction in the penetration, while maintaining the hit rate close to 100%. As the cascade progresses, the filtering rate (reduction in penetration) at each stage tends to decrease and the hit rate also tends to decrease.

3.1. Query

For each query image Q, we determine $v_j(Q)$ for $j = 1..t$. For each stage j, we remove all the samples from gallery that falls outside the window W_j, which calculated during the training phase. The set of samples that remain after the cascade is returned for explicit comparison. The algorithm 2 shows the computation of the candidate list.

Algorithm 2 Computing Candidate list for a probe.

```
$CandidateList$ ⇐ {All templates in gallery}
for each projection $P_i$ do
    Retrieve projected values for $CandidateList$ for $P_i$
    Find the window around the projection of probe on $P_i$
    Remove templates outside window in $CandidateList$
end for
Return $CandidateList$
```

4. Experimental Results and Analysis

We used the FVC2002 (DB1,DB2,DB3,DB4) [16] for evaluating the proposed algorithm. Each DB consists of eight prints each of 100 distinct fingers captured by optical sensors (500 dpi). We separate each dataset into independent training and testing sets of equal size.

First the window size W_j for each projection is calculated from the training samples. The size of the window should be such that the samples of the same class falling outside the should be minimal and number of sample falling within the window should be maximum. For each probe template (testing sample), the features are extracted and projected into a lines. We will search only in the dataset which falls in the window centered around the probe as shown in the figure 3. To extract the features from fingerprints we have used 50 clusters for both quadrilaterals and triangles, giving us a combined feature vector of size 100. The number of projections to be used for cascading and size of the window is decided by the experiments and the dataset we are using.

To generate the cascade, we start with a set of 3000 random projections and select the best 100 projections. The final cascade is typically terminated within 50 projections. We also added 100 LDA and 100 PCA projections values for comparison purposes. However, there were no perceptible changes in the results, confirming the effectiveness of random projections for the problem.

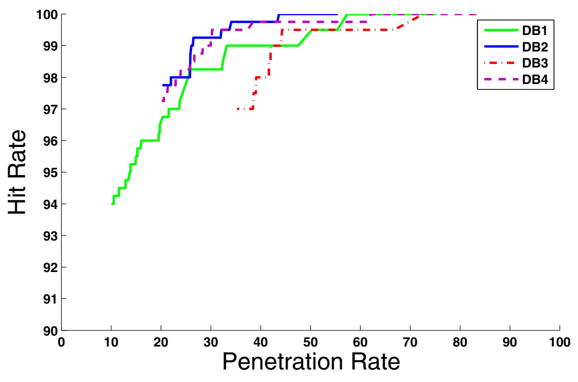

Figure 4. Hit Rate vs. Penetration Rate on FVC 2002 DB1,..,DB4.

Figure 4 shows the hit rate vs penetration rate on different DBs of FVC 2002. Figure 5 clearly shows our method is significantly better in terms of efficiency as compared to [8], and the difference increases with the size of database. Figure 6 shows the nature of filtering rate with increasing number of projections. We note that significant portion of the filtering takes place within the first 10 projections in the cascade, and one might stop there for efficiency purposes. We note that as the number of training samples increase, we can do a better estimation of the window size and the performance improves as seen in Figure 7.

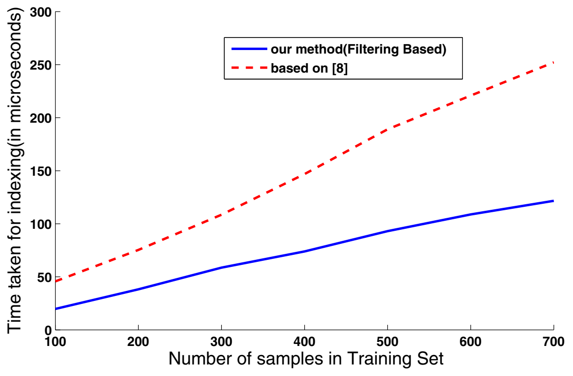

Figure 5. Time taken for indexing with increasing number of training samples.

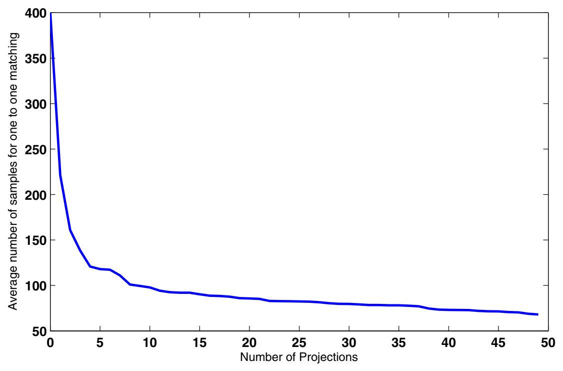

Figure 6. Decrease in penetration with increasing number of projections.

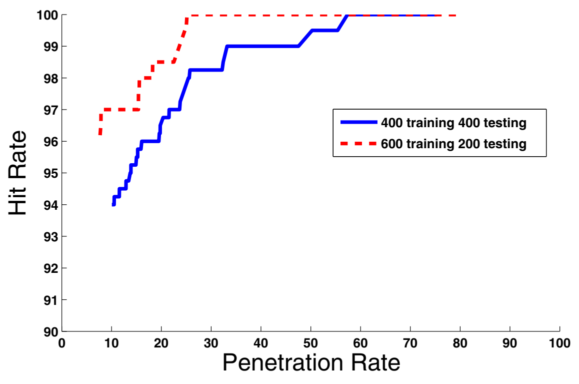

Figure 7. Effect of training sample size on filtering performance.

One of the advantages of the proposed method is that it efficiently and effectively combine different feature sets.

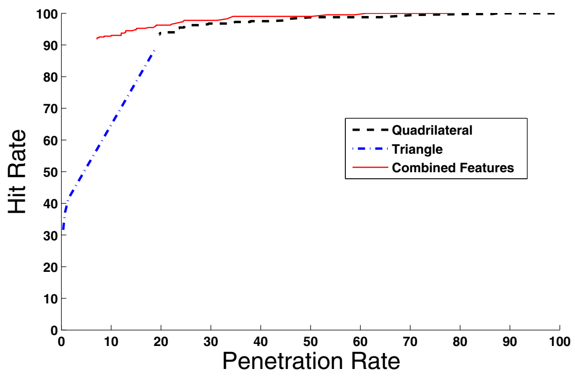

Figure 8. Effect of using the individual (triangles and quadrilaterals) and combined feature sets.

Method	Penetration at 99% Hit Rate	Time taken in μ secs.
Quadruplets[8]	**20%**	147
Combined Features	26%	**74**

Table 1. Result in FVC 2002 DB2 datasets, with equal number of training and testing samples.

Figure 8 shows the results of using the two feature sets considered independently and when combining the two. As the final cascade is based on projections, the length of the feature vector does affect the database and the effect on the filtering process is minimal.

5. Conclusions

We explore the method for fingerprint database filtering using cascaded projections. The cascade is built on a set of random projections applied to a given feature set. The results are comparable to the state of the art fingerprint indexing methods. Experiments show that the proposed method extremely efficient and can give a significant advantage when used as the first stage in identification. While we do not propose any new features for fingerprint indexing, our method is able to combine a large number of existing feature descriptors into a compact and efficient cascaded filter, irrespective of the feature vector size. This results in significant savings in time during identification of fingerprints. Due to it efficiency, the method may be used as the first stage while combining multiple indexing and filtering methods.

References

[1] G. Bebis, T. Deaconu, and M. Georgiopoulos. Fingerprint identification using delaunay triangulation. *IEEE International Conference on Intelligence, Information and Systems*, pages 452–459, 1999.

[2] B. Bhanu and X. Tan. Fingerprint indexing based on novel features of minutiae triplets. *IEEE Transactions on PAMI*, 25(5):616–622, 2003.

[3] J. D. Boer, A. M. Bazen, and S. H. Gerez. Indexing fingerprint databases based on multiple features. *Proceedings of the 12th ProRISC Workshop (Program for Research on Integrated Systems and Circuits)*, pages 300–306, November 2001.

[4] R. Cappelli, A. Lumini, D. Maio, and D. Maltoni. Fingerprint classification by directional image partitioning. *IEEE Transactions on Pattern Analysis Machince Intelligence*, 5(21):402–421, 1996.

[5] K. Choi, D. Lee, S. Lee, and J. Kim. An improved fingerprint indexing algorithm based on the triplet approach. *AVBPA*, pages 584–591, 2003.

[6] R. S. Germain, A. Califano, and S. Colville. Fingerprint matching using transformation parameter clustering. *IEEE Computing Science and Engineering*, 4(4):42–49, 1997.

[7] N. Goel, G. Bebis, and A. Nefian. Face recognition experiments with random projections. *SPIE Conference on Biometric Technology for Human Identification*, March 2005.

[8] O. Iloanusi, A. Gyaourova, and A. Ross. Indexing fingerprints using minutiae quadruplets. *IEEE Compter Society Conference Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 127–133, 2011.

[9] A. Iqbal and A. Namboodiri. Cascaded filtering for biometric identification using random projections. *In proceedings of National Conference on Communications*, pages 1–5, 2011.

[10] A. K. Jain, R. M. Bolle, and S. Pankanti, editors. *Biometrics, Personal Identification in Networked Society: Personal Identification in Networked Society*. Kluwer Academic Publishers, Norwell, MA, USA, 1998.

[11] A. K. Jain, S. Prabhakar, L. Hong, and S. Pankanti. Fingercode: A filterbank for fingerprint representation and matching. *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2, 1999.

[12] A. K. Jain, S. Prabhakar, and H. Lin. Multichannel approach to fingerprint classification. *IEEE Trans. Pattern Analysis Machine Intelligence*, 21(4):348–359, 1999.

[13] W. B. Johnson and J. Lindenstrauss. Extensions of lipschitz mappings into a hilbert space. *In* Contemporary Mathematics, 26:189–206, 1984.

[14] K. Karu and A. K. Jain. Fingerprint classification. *Pattern Recognition*, 29(3):389–404, 1996.

[15] T. Liu, G. Zhu, C. Zhang, and P. Hao. Fingerprint indexing based on singular point correlation. *IEEE International Conference on Image Processing*, 3:293–296, 2005.

[16] D. Maio, D. Maltoni, R. J. Cappelli, L. Wayman, and A. K. Jain. Fvc2002: Second fingerprint verification competition. *Proc. of International Conference on Pattern Recognition*, pages 811–814, August 2002.

[17] D. Maltoni, D. Maio, A. K. Jain, and S. Prabhakar. *"Handbook of Fingerprint Recognition"*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2003.

[18] S. Pankanti, S. Prabhakar, and A. K. Jain. On the individuality of fingerprints. *IEEE Trans. Pattern Analysis Machine Intelligence*, 24(8):1010–1025, aug 2002.