NISTIR 5493

Comparison of FFT Fingerprint Filtering Methods for Neural Network Classification

C.I. Watson
G.T. Candela
P.J. Grother

U.S. DEPARTMENT OF COMMERCE
Technology Administration
National Institute of Standards
and Technology
Advanced Systems Division
Gaithersburg, MD 20899

NIST

NISTIR 5493

Comparison of FFT Fingerprint Filtering Methods for Neural Network Classification

C.I. Watson
G.T. Candela
P.J. Grother

U.S. DEPARTMENT OF COMMERCE
Technology Administration
National Institute of Standards
and Technology
Advanced Systems Division
Gaithersburg, MD 20899

September 1994

U.S. DEPARTMENT OF COMMERCE
Ronald H. Brown, Secretary

TECHNOLOGY ADMINISTRATION
Mary L. Good, Under Secretary for Technology

NATIONAL INSTITUTE OF STANDARDS
AND TECHNOLOGY
Arati Prabhakar, Director

Contents

List of Figures

List of Tables

Abstract

Two types of Fourier Transform based filters are presented and used to enhance fingerprint images for use with a neural network fingerprint classification system developed at NIST [1][2]. With image enhancement the system is capable of achieving classification error rates of 8.65% with 10% rejects (average over volumes 1-5 of *NIST Special Database 9*), a 2 percentage point improvement in error rate versus using no fingerprint enhancement. Speed of the filters range from 2 to 9 seconds. Classification tests were performed with fingerprints from *NIST Special Database 9 Volumes 1-5* [3] using ridge-valley based feature extraction, Karhunen Loève transform, and a Probabilistic Neural Network (PNN) classifier. Improvements made to the classification system used include: a new segementor, use of non uniform feature vectors, and a faster version of the PNN classifier. The faster PNN classifier results in an average of four times faster classification with no change in resulting error rates. Also, the testing method used differs from past reports because no rolling of the same print is allowed to appear in both the training and testing set used by the Neural Network classifier.

Keywords: image enhancement, fast Fourier transform, fingerprint classification, Probabilistic Neural Network, Karhunen Loève transform, database, registration.

1Introduction

The current classification system used at NIST involves three main steps: pre-processing, feature extraction and classification. The current problem being presented is to accurately classify fingerprints into five major class groupings: Arch, Left Loop, Right Loop, Tented Arch and Whorl (see Figure 1a-e for example prints). A major problem that has occurred in trying to classify fingerprints is extracting features from poor quality images. The features extracted from poor quality images tend to have scattered ridge directions with low confidences. Poor ridge directions can result in erroneous registration points or, since some of the classes like arch and tented arch may have very slight differences, the classifier will have difficulty accurately separating the different classes. This report concentrates on using three different Fourier Transform based image filters to help reduce the noise present in the images. One hopes that by providing the feature extractor with less noisy images that it will be able to extract less ambiguous features to send into the classification stage resulting in more accurate classification. Results will show that the goal of extracting better features and improving classification was accomplished.

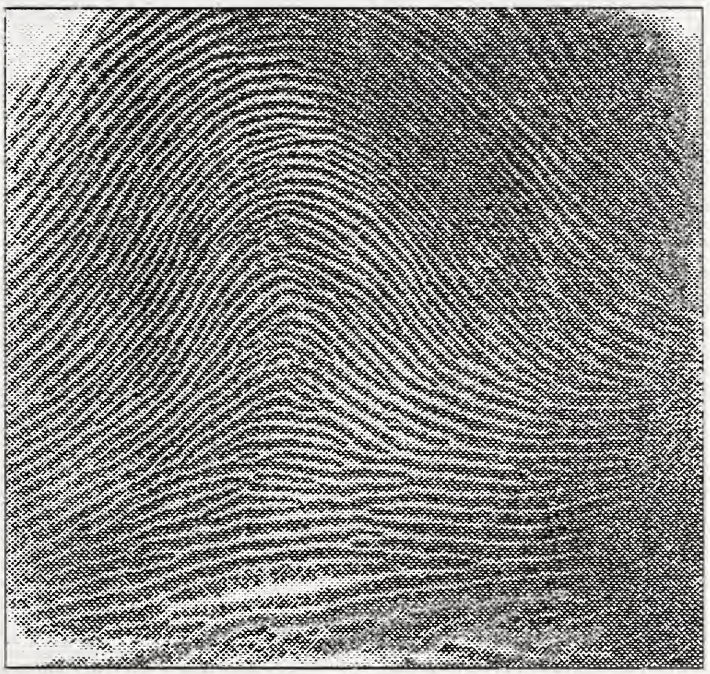

Figure 1a: Example of arch pattern.

Figure 1b: Example of left loop pattern.

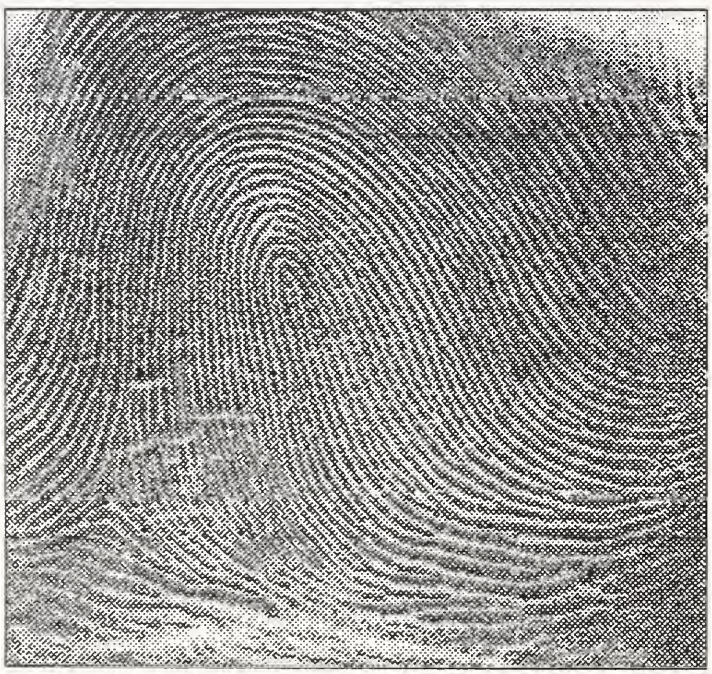

Figure 1c: Example of right loop pattern.

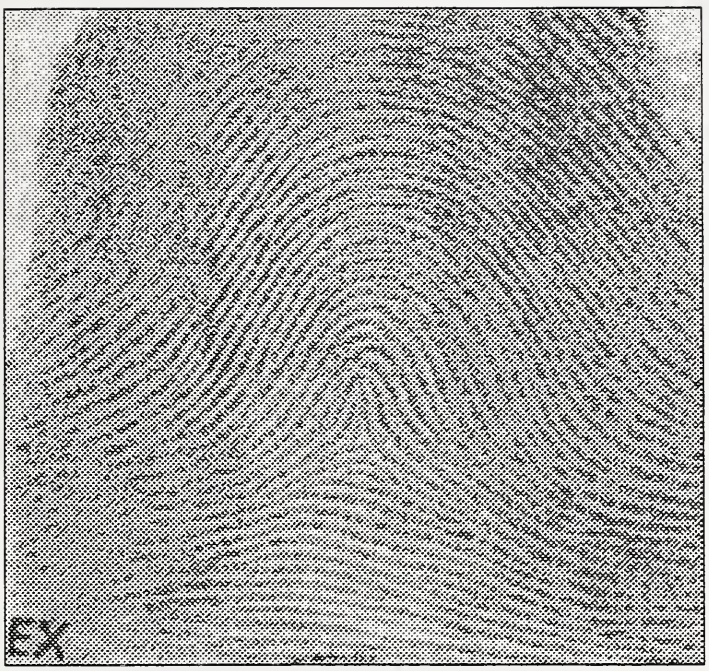

Figure 1d: Example of tented arch pattern.

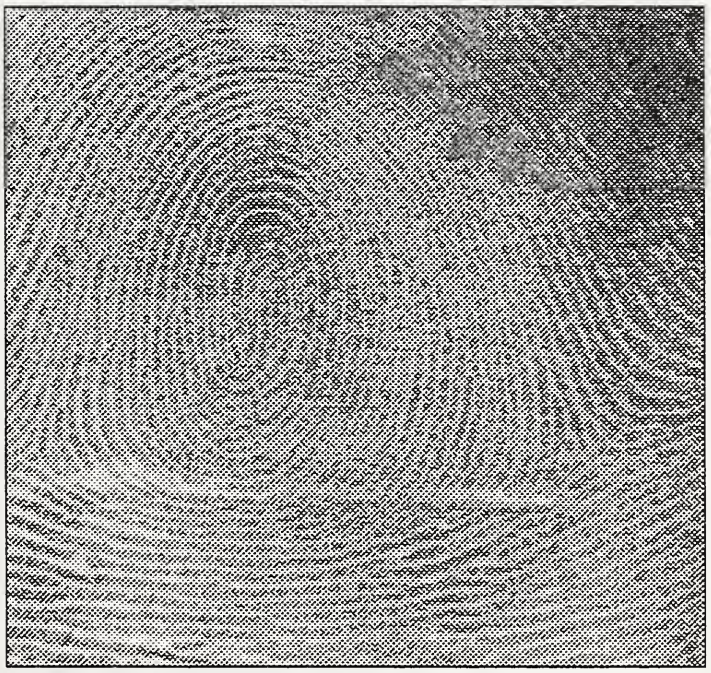

Figure 1e: Example of whorl pattern.

The images used, for training and testing, are from *NIST Special Database 9 Volumes 1 and 2* [3], which are 832 X 768 8-bit gray scale images. All reports to this point have reported results using *NIST Special Database 4* [4]; there are very significant differences between the two databases making comparison of results obtained from each database very difficult. Section 2 discusses

the important differences between *NIST Special Databases 4* and 9 such as method of scanning the data and quality of the data.

Another important difference from earlier testing methods is that in previous test the "f" rollings of the fingerprints in *Special Database 4* were used for training and the "s" prints were used for testing. This is very significant because for test in this report the "f" prints from one volume are used as the training set and the "s" of a different volume are used as the testing set. Test have shown there is a significant difference in classification error rates (3-4%) that occurs when the first rolling of a print appears in the training set (especially with a Probabilistic Neural Network) versus having different data in the training and testing sets. Knowing this one should not compare results reported in this report with results reported in earlier reports. For this reason, Section 8 contains results of classification at various stages of system improvement (i.e. no enhancement at all, adding registration and adding new feature extraction methods) for use in comparing the effects of applying different filters to the images.

Most of the original 832 X 768 images contain significant amounts of white background space which only increases processing time and does not help classification. Segmentation, as described in Section 3, is used to obtain the best 512 X 480 section of the original image for use by the rest of the classification system. Currently a section of 512 X 480 is used for compatibility with current algorithms and to help reduce computation time.

The next step is filtering of the fingerprint images, which is discussed in Section 4. As previously stated there are three different filters that will be applied to the image data. Each filter uses the fast Fourier transform to first convert the image into the frequency domain before applying filter masks. The first filter processes the image in subsections and reconstructs the filtered image from these sections. The other two filters use specially oriented masks which filter the image based on distinct orientations. They create new images based on each orientation and then reconstruct the filtered image from these orientation images.

After filtering, the image is ready for feature extraction. The current method being used, discussed in Section 5, is a ridge-valley feature extractor. The feature extractor provides more detail in important areas of the fingerprint print image such as cores and deltas by allowing more ridge directions in these areas at the expense of less ridge data near the edges of the image. At this stage the ridge directions are also registered. Figure 2 shows an example of a core location found by registration. Registration is used to move the core of each fingerprint to a common point and help reduce differences introduced by segmenting the fingerprint at different locations. The output of the feature extractor, an array of 840 ridge directions, is reduced to a much smaller set of input features by first calculating the covariance matrix of the training set feature vectors and then sending the principle eigenfunctions of the covariance matrix (calculated using EISPACK routines [5]) to a Karhunen Loève (KL) transform. The KL transform is a dimensionality reducing transform which takes the 840 ridge directions for each image and produces approximately 120 features for use as input to the Neural Network classifier. Another useful feature of the KL transform method is that the features are ranked in order of decreasing variance so it is simple to use fewer features than are actually found by selecting the first n features.

The final stage of the system is classification. For classification purposes the primary class of each print was used and no weight was given for any referenced classes at this time. Also, all scar prints were discarded from the dataset as it was not clear how to handle these prints. The classifier used for this report is a Probabilistic Neural Network [2][13] as described in Section 6. During classification the *a priori* probabilities of each class are applied to the output activations giving more weight to classes that have a more common occurrence in a natural distribution. Also, a "fast" implementation of PNN is used which reduces the computation time by approximately a factor of 4 with no change in classification accuracy. The method takes advantage of the KL feature set being in order of decreasing variance to limit the calculation time.

The results of the experiments performed are given in Section 8 along with the methods used for scoring and rejecting the fingerprints. Unlike previous work reported, the scoring does not use the *a priori* probabilities when scoring because after rejecting a certain number of prints it may be incorrect to assume the class distributions are still the same. At this point there is not sufficient data to estimate the class distributions after certain levels of rejection.

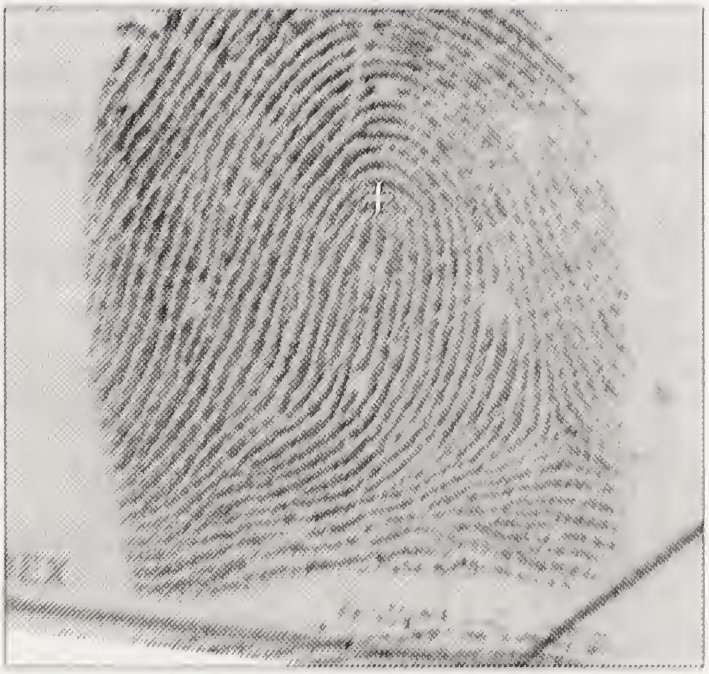

Figure 2: Example of a core location found by registration.

2 Experimental Fingerprint Database

To date most fingerprint classification results reported in NIST work were performed using *NIST Special Database 4* (SD4). The images used in this report for training and testing purposes were taken from *NIST Special Database 9 Volumes 1-5* (SD9). SD9 images are 8 bit per pixel gray scale images of mated fingerprint card pairs (270 card pairs per volume). This means the fingerprints are matched at the card level, and not every individual fingerprint from mated cards will necessarily have the same exact class. In contrast, SD4 was setup so that all matched fingerprints had the same class label. Every fingerprint in SD9 has a National Crime Information Center (NCIC) [6] class label assigned by classification experts. These assigned NCIC classes were converted to one of the following five major groups: Arch, Left Loop, Right Loop, Tented Arch and Whorl for classification purposes.

The most obvious difference between the two databases is that SD4 contains an equal number of fingerprints from the five major classes where as SD9 was randomly selected from current FBI work so that it approximated a natural distribution of the fingerprint classes. The "natural" probability of occurrence for each of the five major classes is shown in Table 1. These probabilities were calculated from a sample of fingerprint classes containing approximately 222 million fingerprint classes. Also shown in table 1 are the exact class distributions of volumes 1 and 2 of SD9. The variations between the exact and natural distributions are accounted for by weighting the output activations of the PNN classifier with the probabilities for each class (see Section 6).

Class	"Natural"	Volume 1	Volume 2
A	0.037	0.067	0.038
L	0.338	0.306	0.316
R	0.317	0.311	0.309
T	0.029	0.041	0.048
W	0.279	0.275	0.289

Table 1: Probability of occurrence of the five major class groups.

The random collection of data from current FBI work also results in a lower quality of images, although it is a more realistic sample of the classification work being done by humans. The quality is lower because the "s" rollings are from current search cards sent to the FBI which in most cases are of lower quality than the permanent file cards. The prints used in SD4 were taken from the permanent files of the FBI in which case if multiple cards have been collected on one individual the better quality cards are stored in the permanent file.

There was also a significant difference in the method used to collect the data for SD4 and SD9. In SD4 each image was scanned individually and some "eyeball" registration was done to center the image in the area being scanned as well as rotating the image into the upright position. SD9 was collected by first scanning all ten prints on a card into one large image (4096 X 1536 pixels) and then segmenting the individual images. The images were segmented at the same point for every card, so there was no "eyeball" registration or orientation correction occurring in SD9.

Taking all the factors of quality, registration, and segmentation into account, SD9 is a more realistic method of evaluating a complete classification system, where as SD4 is more useful in evaluating a simple feature extraction routine and classifier. The use of SD9 for evaluating the performance of the entire system should provide more realistic results than using SD4.

3 Image Segmenting

The fingerprints from *NIST Special Database 9,* present a new problem to the classification system because the images are 832 by 768 pixels in dimensions and contain significant amounts of white space in the image (see Figure 3). The segmentation routine described below is used to segment the fingerprint data for use by the rest of the classification system.

The segmentation routine takes as its input an original fingerprint image, which is an 8-bit gray raster of dimensions 832 pixels (width) by 768 pixels (height); its output is a smaller 8-bit raster, 512 by 480 in size, produced by snipping from the input raster a rectangular region, with the sides of the snipped rectangle not necessarily parallel to the corresponding sides of the original raster. Snipping out a smaller rectangle is helpful because it reduces the amount of data that has to undergo the compute-intensive filtering process, and also because it produces a raster whose size is well matched to our implementation of Wegstein's R92 registration routine. The segmentor also attempts to return fingerprints which are rotated to an upright position.

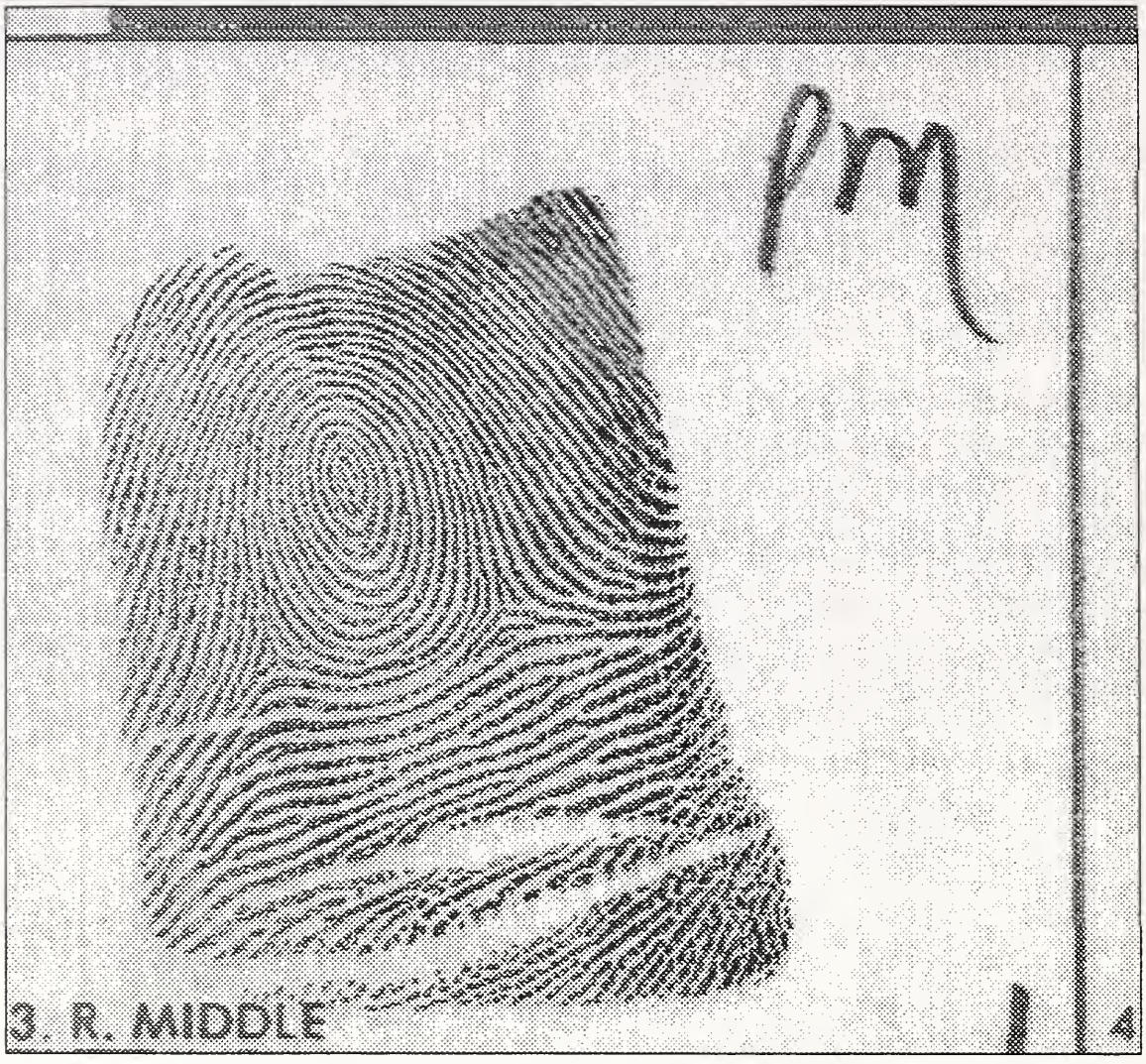

Figure 3: Original raster of image to be segmented.

The segmentor decides which rectangular region of the raster to snip out by performing the following steps (Figure 3 is an original fingerprint raster, and Figure 4a-d illustrate the processing as applied to this fingerprint):

1) Produce a 104x96-pixel binary raster whose pixels indicate which 8x8-pixel blocks of the original raster are considered to be "foreground":

   Find minimum pixel value for each block as well as the global minimum and maximum pixel values.

   For (several *factor* values between 0.0 and 1.0)

   {

   threshold = *global_min* + *factor* * (*global_max* - *global_min*)

   Set to "true" each pixel of candidate-foreground map whose corresponding pixel of the array of block minima is <= *threshold* and count resulting

candidate-foreground pixels.

Count the transitions between the true and false values in the candidate-foreground, counting along all rows and also along all columns. Keep track of minimum number of transitions.

}

Among those candidate-foregrounds whose number of true pixels is within specified limits, pick the one with the fewest transitions. (If *threshold* is too low, there tend to be many holes in what should be solid blocks of foreground; if the *threshold* is too high, there tend to be many spots on what should be solid background. If *threshold* is about right, there are few holes and spots, and hence relatively few transitions.

Figure 4a shows the foreground produced from the fingerprint of figure 3.

Figure 4a: Foreground of Figure 3.

2) Clean up and center the foreground-map:

Perform three erosions on foreground-map. Each erosion consists of changing to false each true pixel that is next to a false pixel.

Find the connected sets ("blobs") of true pixels, and change to false all pixels except those belonging to the largest blob.

Change to true any pixel that has true pixels both to its left and to its right, or both above and below itself.

Calculate centroid of foreground-map and translate foreground-map moving its centroid to the middle pixel position of its raster.

Figure 4b shows the result of cleaning up and centering the foreground.

Figure 4b: Foreground of Figure 3, "cleaned" and centered.

3) Find the left, top, and right edges of the foreground:

Move upward from middle row and find left-most true pixel of each row, but stop when horizontal difference between current row's and previous

row's left-most true pixel is > 1.

Repeat process, moving downward from middle row.

These two processes find the left edge of the foreground. The limit of one on the horizontal change prevents the supposed edge from going around a corner of the foreground.

Similarly, find top and right edges.

The three lines in the center of Figure 4c are the edges of the foreground.

Figure 4c: Edge detection of Figure 3.

4) Fit straight lines to foreground edges:

For each of the three edges, use linear regression to produce a straight line that most closely fits the points comprising the edge.

Naturally, the left and right edges are fitted to lines of the form x = m * y + b.

The top edge is fitted to a line of the form y = m * x + b.

The straight lines in the right part of Figure 4c are the fitted lines.

5) Calculate overall slope of foreground:

Calculate the average of the slopes of the left edge, the right edge, and a line perpendicular to the top edge (negative the slope of the fitted line). This average slope is the overall slope of the foreground.

6) Find top of foreground:

Make a histogram from the rows of a rectangle whose width corresponds to the output raster width, whose height is large, whose center is at the center of the foreground's raster, and which is rotated so that its sides have the same slope as the foreground.

Move downward in the histogram, stopping at the first row which both fits entirely into the foreground raster and has a threshold number of true pixels. (Note that the resulting foreground top is not generally the same as the top edge found earlier, because its slope is the average of the slopes corresponding to the three edges found, rather than being the slope of just the top edge.)

7) Finish deciding the snipping parameters:

The overall slope computed earlier determines the angle of snipping which nullifies any rotation of the fingerprint.

As for the position of snipping, that is chosen so that the top of the snipped rectangle corresponds to the foreground top found in the preceding step. (Having the snipped rectangle hang from the top of the foreground, instead of centering it on the foreground center, produces a bias in favor of the last joint of the finger, which is the only interesting part of the finger as far as classification is concerned.)

The box superimposed on the foreground, in the left part of Figure 4c, shows the snipping rectangle that has been decided on.

8) Snip smaller raster from the original raster:

Produce the output raster by copying the appropriate pixels of the input raster, applying the translation and rotation that correspond to the snipping parameters that were computed.

Figure 4d shows the output raster snipped from the input raster. Its edges correspond to the box in the left part of Figure 4c.

Figure 4d: Segmented image of Figure 3.

4 Fingerprint Image Enhancement

This section describes the two filtering techniques used to enhance the quality of the fingerprint images. Both filter techniques use the fast Fourier transform (FFT) to compute the discrete Fourier transform (DFT) when filtering the image. The difference in the methods is that the first filter operates on 32 X 32 pixel sub-regions of the image and the second filter acts globally over the entire image. Also, the second filter enhances the image in distinct directions where as the first just does simple noise reduction. Figure 5 shows the original unfiltered fingerprint raster that has been through the segmenting process.

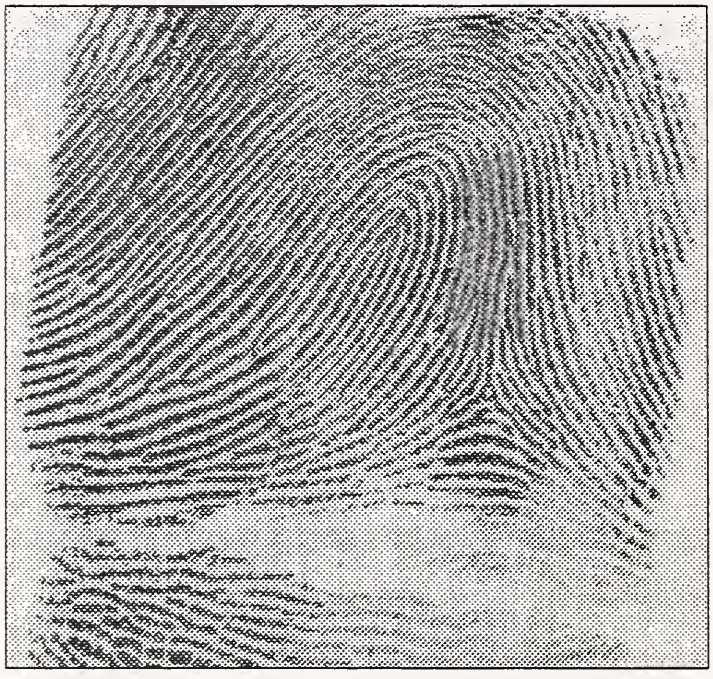

Figure 5: Original image **f0000048**.pct.

4.1 Localized FFT Fingerprint Filter

The first filter used to improve the quality of the fingerprint images is based on the algorithm in [7]. This filter processes the image in 32 X 32 pixels, beginning in the upper left hand corner of the image. After processing a tile it shifts right 24 pixels and to obtain the next 32 X 32 tile, resulting in the first 8 columns of the tile being common with the last 8 columns of the previous tile. After reaching the right side of the image the filter shifts down 24 pixels, resulting in the 8 rows of common data with vertically adjacent tiles, and restarts at the left side of the image. Processing continues until reaching the bottom right side of the image. The common data between the horizontally and vertically adjacent tiles helps reduce the artifacts (visible in Figure 6) created by processing the image in tiles.

Each tile is treated as a matrix of real numbers. The first step in filtering a tile is to compute the two-dimensional DFT, defined as follows (B set to zeros):

$$X_{jk} + iY_{jk} = \sum_{m=1}^{32} \sum_{n=1}^{32} (A_{mn} + iB_{mn}) \exp(-2\pi i (\frac{(j-1)(m-1)}{32} + \frac{(k-1)(n-1)}{32})) \quad (1)$$

The FFT is used, rather than using formula (1) directly. The filtering of some of the high and low spatial frequencies is done using a mask to set these frequencies to zero. Next the power spectrum P of the FFT is computed:

$$P_{jk} = X_{jk}^2 + Y_{jk}^2 \quad (2)$$

The elements of the power spectrum (P_{jk}) are then raised to a power α (0.3 was used) and multiplied by the FFT elements $X + iY$ producing the new elements $U + iV$:

$$Q_{jk} = P_{jk}^{\alpha} \quad (3)$$

$$U_{jk} = Q_{jk} X_{jk} \quad (4)$$

$$V_{jk} = Q_{jk} Y_{jk} \quad (5)$$

Finally, the inverse transform of $U + iV$ is computed, and its real part becomes the filtered tile. In reconstructing the image the filter keeps only the center 24 X 24 pixels, accounting for the 8 pixel overlap, and discards the 4 outer edge rows/columns of the tile. The multiplication of the FFT elements by a power of the power spectrum has the effect of amplifying the dominant frequencies in the tile. Presumably, the dominant frequencies of the tile are those corresponding to the ridges thereby increasing the ratio of ridge information to non-ridge noise and adapting to variations in ridge frequency from one tile to the next. Figure 6 is a result of applying this filter to the raster of Figure 5.

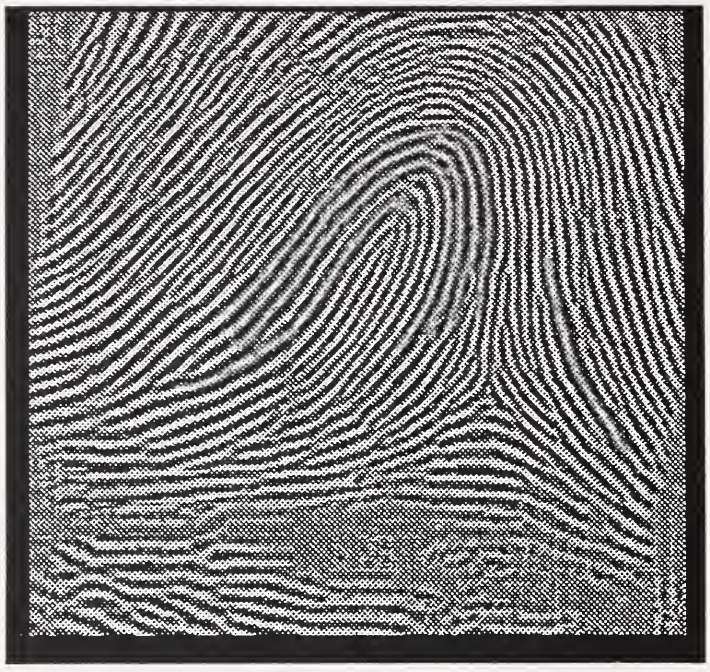

Figure 6: Image filtered using localized FFT filter.

4.2 Directional FFT Filter

The directional FFT filter was designed to do better filtering with respect to the ridge flow in the fingerprint image [8]. The filter uses a predefined orientation mask designed to filter the fingerprint image in a primary ridge direction while preserving the detail of the minutiae. Another advantage of the filter is that it does not produce artifacts as seen with the localized FFT filter.

The filter processes the image by first calculating the FFT of the image. Next, the directional mask is applied by rotating it to ten distinct orientations, creating ten different images with the ridge flow enhanced in each of ten distinct directions. Then the inverse FFT for each direction filtered image is computed (see Figure 7a). The pixel orientations of the filtered image are determined by comparing the ten direction filtered images, pixel by pixel, and recording the direction with the largest squared magnitude at each pixel as the pixel orientation in the filtered image. A histogram smoothing function is applied to the recorded pixel orientations to help smooth directions in local neighborhoods.The filtered image is then reconstructed using the recorded pixel directions to determine from which direction filtered image to select each pixel value. Figure 7b shows the results of filtering the fingerprint in Figure 5 with this method.

After some experimentation it was determined that using ten orientations was probably not necessary, so adjustments were made to the kernel mask and a second version used only six orientations (see Figure 8a and Figure 8b).

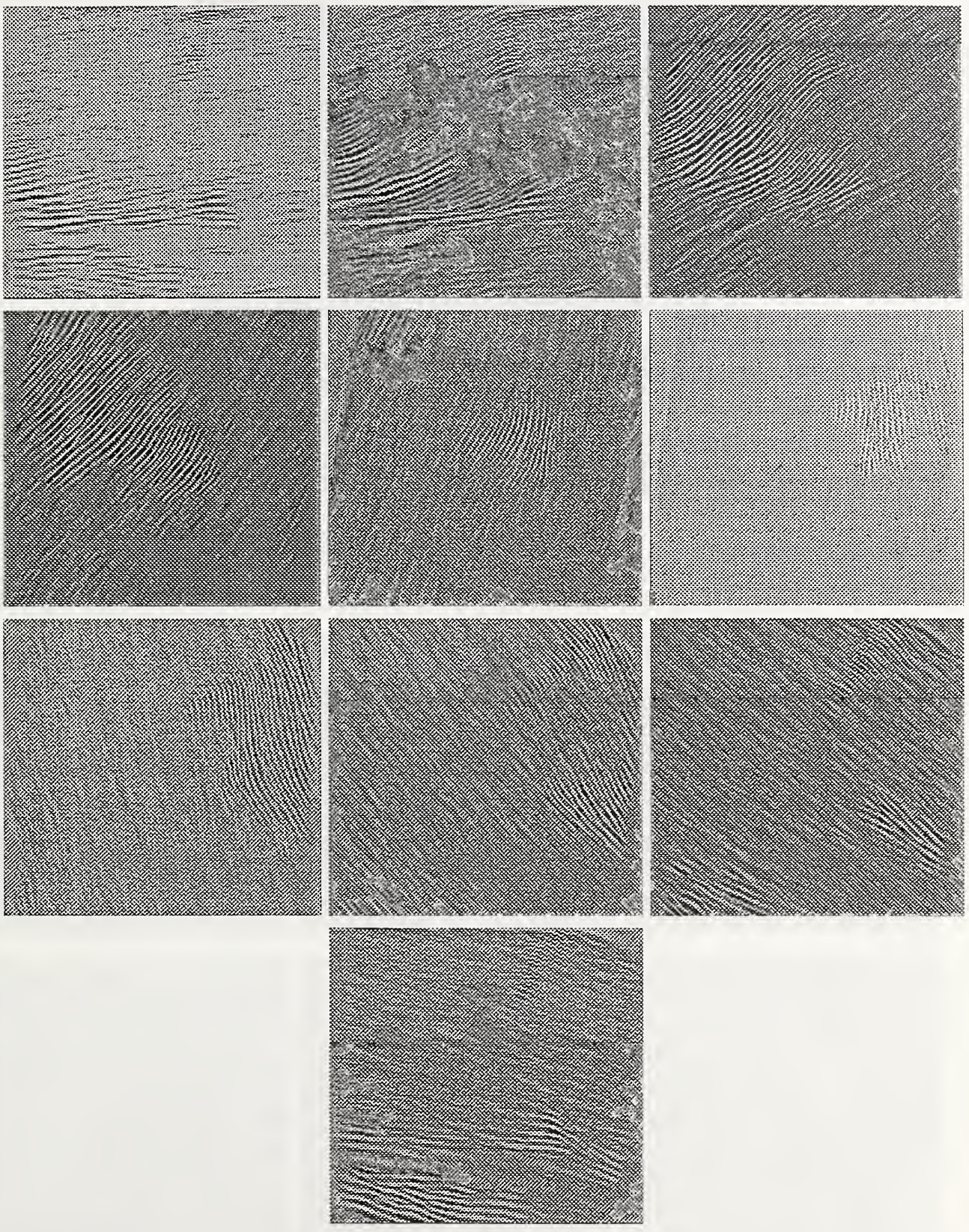

Figure 7a: Orientation images for direction filter version1.

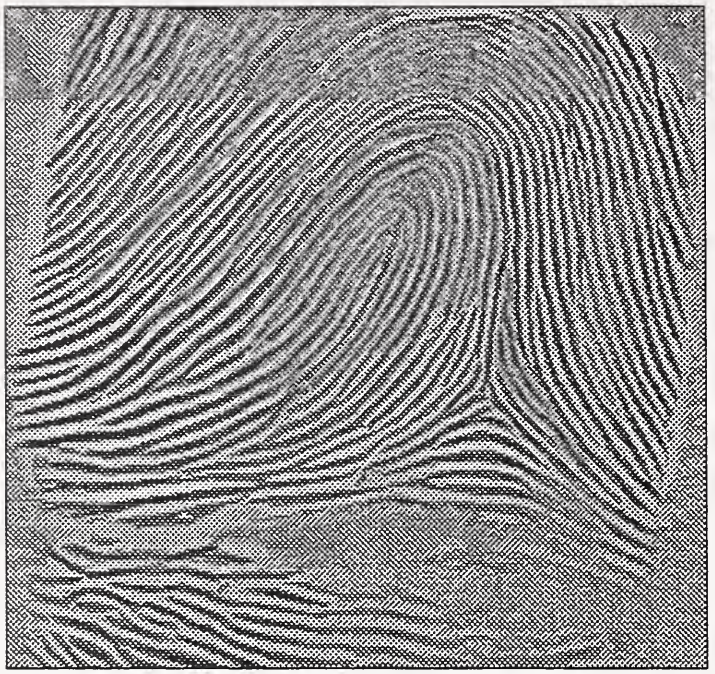

Figure 7b: Image filtered using version 1 of the directional filter (ten orientation masks).

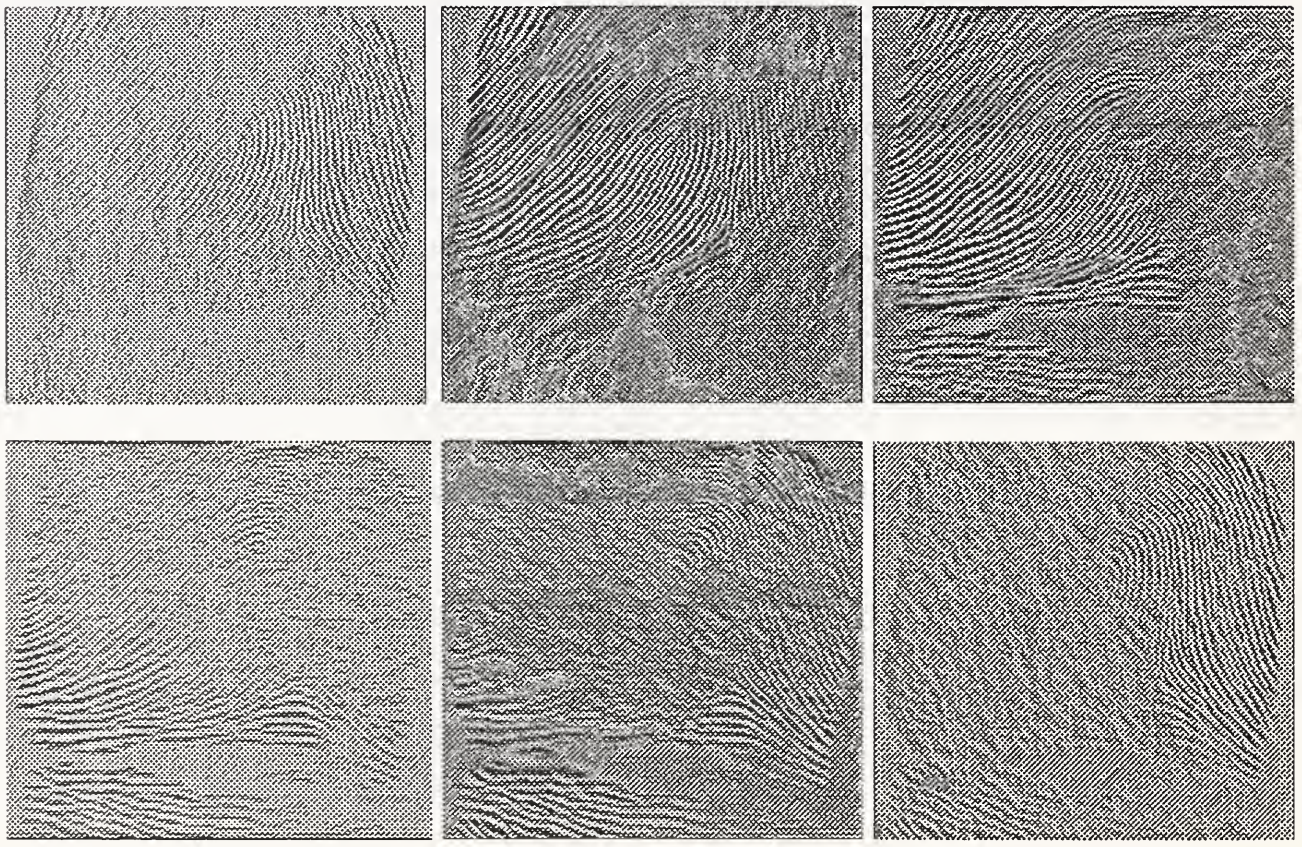

Figure 8a: Orientation images for direction filter version2.

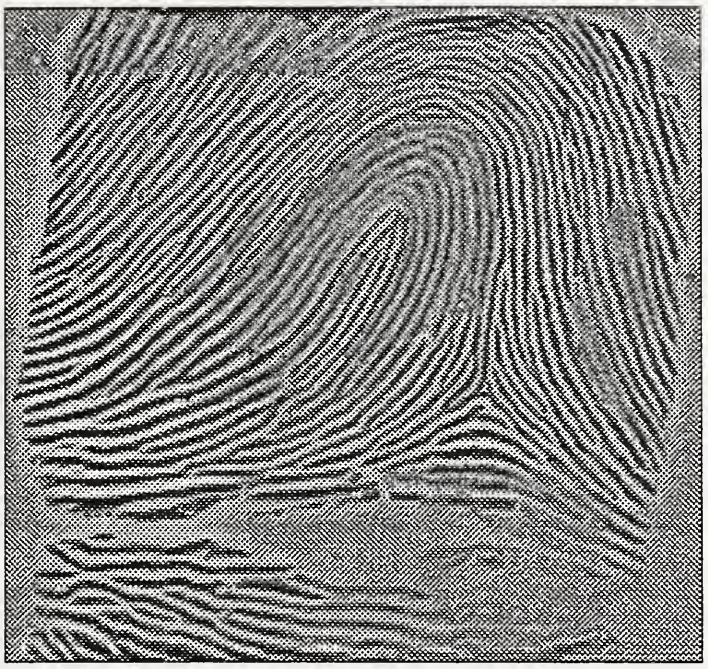

Figure 8b: Image filtered using version 2 of the directional filter (six orientation masks).

5 Feature Extraction

An earlier version of the direction finder, based on the ridge-valley fingerprint binarizer described in [9], produced a grid of directions spaced 16 pixels apart horizontally and vertically, for a total of 840 (28 X 30) vectors as shown in Figure 9. The ridge directions were then registered by shifting the fingerprint "core" to a location which is the median core location from a larger sample of handmarked core data. Wegsteins' [10] routine was used to find the core location for each fingerprint. Figure 10 shows an example of a incorrect registration point found before filtering and Figure 11 shows that after filtering a correct registration point was found.

The current version of the direction finder [11] produces better classification results by using the same number of vectors, but arranged in a fixed unequally spaced pattern which concentrates the vectors in certain areas at the expense of less important regions (see Figure 12).The location of the dense ridge directions was determined by hand marking the location of cores and deltas in a large sample of images and then adding up the number of cores and deltas located in each 32 X 32 grid of the image. A mapping of the most dense core and delta regions was used to determine where the dense ridge regions should be located. Each 32 X 32 pixel tile of the raster gets either 1, 4, or 16 direction vectors. First, a grid is produced with the vectors spaced every 8 pixels (but still using 16 X 16 pixel averaging windows); this grid has 16 vectors per tile. Grids with 4 vectors/tile and 1 vector/tile are produced from this original grid by two averaging steps. Then, some tiles receive their vectors from the coarse grid. some from the medium grid, and some from the fine grid, according to a pattern produced as follows. Let the number of tiles that receive 1, 4, and 16 vectors be n_1, n_4, and n_{16}. There are 15 X 16 = 240 tiles, so $n_1 + n_4 + n_{16} = 240$. The total number of vectors is fixed at 840 for comparability with the earlier version, so $n_1 + 4n_4 + 16n_{16} = 840$. Using these two equations in three variables, integer values of n_{16} with $0 <= n_{16} <= 40$ produce n_1 and n_4 values that are non-negative integers. Meaningful values for the three variables were produced by simply picking n_{16} values and solving for the other two variables, since there is not a unique meaningful solution. Through experimentation it was determined that the best classification error rate was obtained using a n_{16} value of 10.

The 840 output vectors were then reduced, using a Karhunen Loève (KL) [12] transform, to approximately 120 features for use in with the Neural Network classifier. The dimensionality reduction was accomplished by first calculating the covariance matrix of the training data and determining the principle eigenfunction set using EISPACK routines. The KL transform uses the output vectors along with the mean output vector (calculated from the training data) and principle eigenfunctions to produce the reduced feature set for each image. In the transform, the mean output vector is first subtracted from the output vector and then the result is multiplied by a matrix containing the principle eigenvectors. Since the KL features are ranked in order of decreasing variance it is simple to reduce the number of features used by selecting the first *n* features. Through testing it was determined that no difference in error rate was seen when using more than 96 input features.

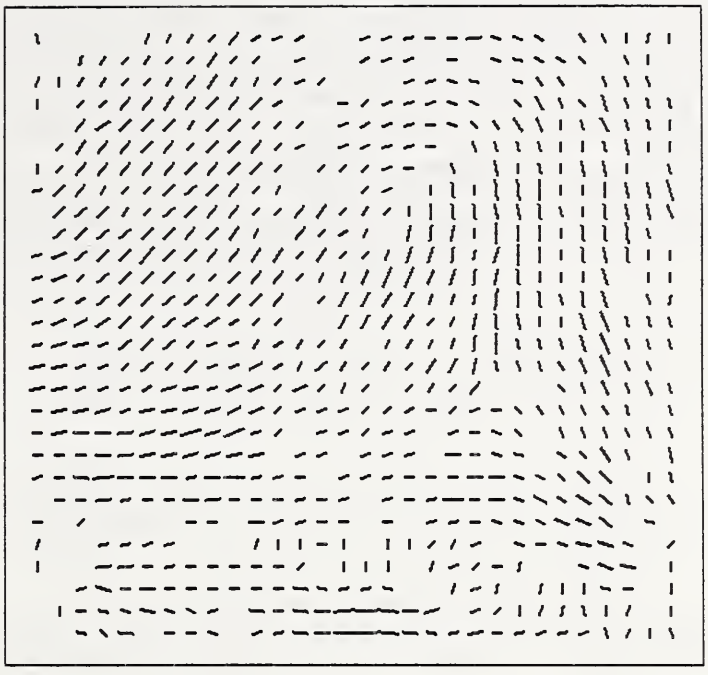

Figure 9: Equally spaced direction vectors of non-filtered image.

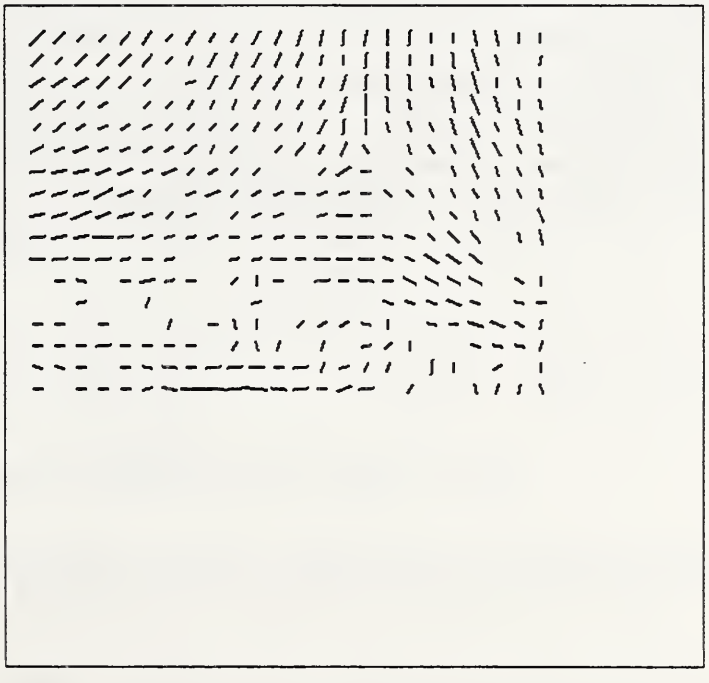

Figure 10: Registered equally spaced direction vectors of non-filtered image.

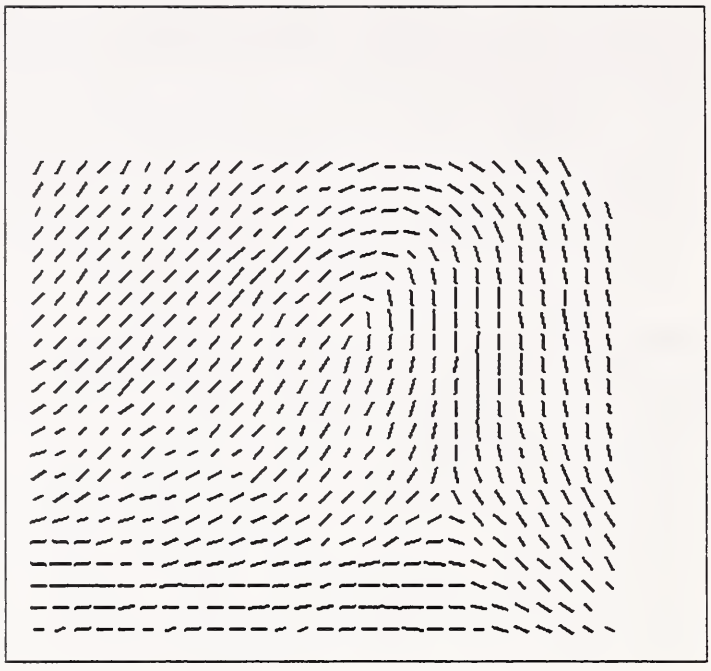

Figure 11: Registered equally spaced direction vectors of filtered image.

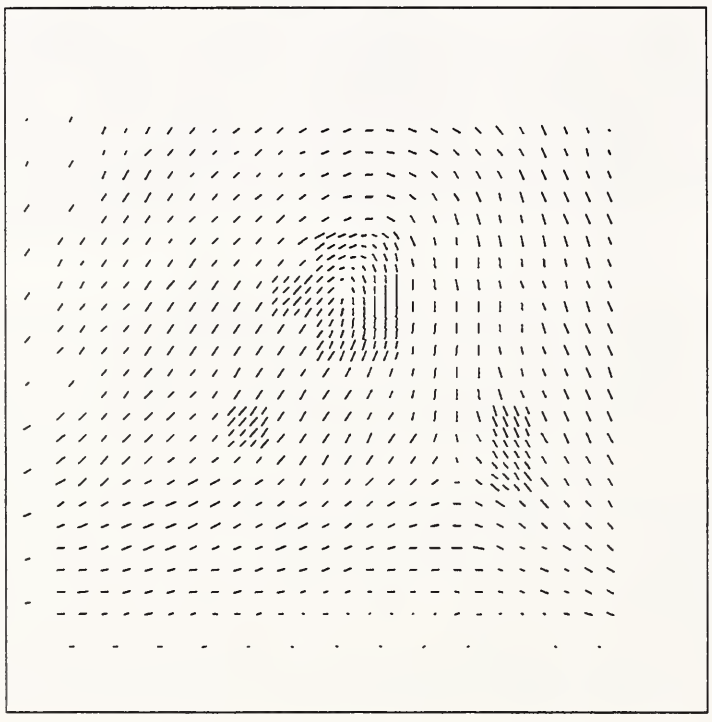

Figure 12: Registered non-equally spaced direction vectors of filtered image.

6 PNN Classifier

The classification algorithm used was a Probabilistic Neural Network (PNN)[2][13]. The unknowns are classified by summing the values of the kernel functions of the prototypes for each output class i, and then weighting these "output activations", $D_i(y)$, by a compensating factor involving the *a priori* probability of each output class, $p(i)$ and the number of examples for each class, M_i. The activations are then normalized and the highest activation is selected as the hypothesized class. The kernel function used is a radially symmetric Gaussian kernel parameterized by a smoothing variable σ that was optimized by trial and error.

$$D_i(y) = \frac{p(i)}{M_i} \sum_{j=1}^{M_i} exp\left(-\frac{1}{2\sigma^2} d^2(x_j^{(i)}, y)\right) \tag{6}$$

where the euclidean distance of the unknown $\boldsymbol{y}$ to the j^{th} prototype $\boldsymbol{x}_j$ is:

$$d^2(x_j^{(i)}, y) = \sum_{k=1}^{n} (y(k) - x(k)_j^{(i)})^2 = \sum_{k=1}^{n} d(k)^2 \tag{7}$$

A modification was made to the classifier which decreased the time of classification by a factor of 4 with no cost to classification accuracy. The method takes advantage of the KL features being ranked in order of decreasing variance by applying a threshold factor which keeps only those prototypes which make a significant contribution to the computation of the discriminant function shown above. The exponential in the discriminant function results in the closer prototypes having by far the most significant contribution to the summation. Taking this into account, the function can be approximated by discarding those prototypes with exponential terms contributing less than $10^{-\lambda}$ times the largest term. Meaning, a prototype of any given class, $x_j^{(i)}$, can be deleted from its discriminant summation if:

$$exp\left(-\frac{1}{2\sigma^2} d^2(x_j^{(i)}, y)\right) < 10^{-\lambda} exp\left(-\frac{1}{2\sigma^2} d^2(x_c, y)\right) \tag{8}$$

where x_c is the closest prototype without regard to class.

By taking logs and changing sign this condition can be expressed more usefully in the squared distance domain. If we define the set of eligible prototypes of class i as

$$S^{(i)} = \{j \mid d^2(x_j^{(i)}, y) \le 2\sigma^2 \lambda ln10 + d^2(x_c, y)\} \tag{9}$$

then the discriminant summation of (6) can be abbreviated,

$$D_i(y) = \frac{p(i)}{M_i} \sum_{j \in S^{(i)}} exp\left(-\frac{1}{2\sigma^2} d^2\left(\boldsymbol{x}_j^{(i)}, \boldsymbol{y}\right)\right) \tag{10}$$

so that only those prototypes whose squared distance is less than or equal to the distance of the closest prototype, $\boldsymbol{x}_c$, plus the factor controlled by λ as defined in (9). Note that $\boldsymbol{x}_c$ is the closest prototype without regard to class. The error associated with this approximation is controlled by setting λ to a sufficiently large value. The value used in these experiments was $\lambda = 4$, insuring error rates did not change between traditional PNN and the optimized PNN.

One advantage to this calculation is that an outer limit distance is determined by the current closest prototype's squared distance and the $2\sigma^2\lambda ln10$ factor. If a new prototype becomes the closest, the threshold criteria is reapplied and the set is redefined. The main execution time is saved by the fact that as soon as any distance summation (7) is larger than the criteria set by $2\sigma^2\lambda ln10$ in (4), the calculation can be stopped with $k < n$ and the prototype discarded. This becomes very useful with the KL transform because the expected value for the contribution of a given feature is proportional to the variance of that feature. Formally, over all prototypes, $\boldsymbol{x}_j$, the expected value of $d(k)^2$ in equation (7) for a given unknown $\boldsymbol{y}$ is:

$$E\left(d(k)^2\right) = E\left(x_j(k)^2\right) - 2E\left(x_j(k)\, y(k)\right) + E\left(y(k)^2\right) \tag{11}$$

Then, by substituting in the sample estimates:

$$E\left(d(k)^2\right) \cong \frac{1}{N}\sum_{j=1}^{N} x_j(k)^2 - 2y(k)\frac{1}{N}\sum_{j=1}^{N} x_j(k) + y(k)^2 \tag{12}$$

For KL features the mean value of $x(k)$ is zero, so the expression reduces to:

$$E\left(d(k)^2\right) \cong Var\left(x(k)\right) + y(k)^2 \tag{13}$$

and if the unknown feature vectors are identically distributed as the prototypes then:

$$E\left(d(k)^2\right) \cong 2Var\left(x(k)\right) \tag{14}$$

Since the KL transform ranks the features in order of decreasing variance, the first few features contribute most to the distance calculation. Normally, only 4 or 5 features are used in the distance calculation before the distance to the prototype exceeds the deletion criteria ($2\sigma^2\lambda ln10$) and the calculation can be stopped.

Each filter process was tested twice, the first time using the f rollings from volume 1 as the prototype set and the s rollings of volume 2 as the test set and the second time using the f rollings of volume 2 as the prototypes and the s prints of volume 1 as the testing. This testing method insured that no other rollings of a print in the prototype set occurred in the testing set making the results using the PNN classifier more realistic. It also checked for some consistency in the results over the two sets of data.

7 Method of Rejection

After selecting the class based on the highest output activation as described in Section 6, the highest activation is used a confidence measure to determine wether or not to reject the fingerprint as unclassifiable. Rejecting fingerprints was done by comparing a threshold value to the highest output activation and any output activation below the confidence threshold level is rejected as unclassifiable. The reason for doing this is to discard any prints that appear ambiguous to the classifier resulting in a low output activation.

8 Results

8.1 Accuracy

As is shown in Table 2 an improvement of approximately 2 percentage points was seen in the overall classification error rate when filtering was applied to the fingerprint data. No one filtering method seemed to do significantly better than the other suggesting that the classifier is not extremely sensitive to the technique used to reduce noise in the image. Since most of the prints misclassified at high reject levels are not of bad quality one would not expect more filtering to result in better error rates at high levels of reject. Improved filtering could still help reduce error rates at lower reject levels. Figure 13 and Figure 14 show plots of the error rate versus the percent reject for volumes 1 and 2 of *NIST Special Database 9*.

The scoring method used to present these results was a simple method of dividing the number of wrong prints by the number of accepted prints shown in the equation below. This differs from some previous work reported which used the *a priori* probabilities to calculate the error rates [2]. Using the *a priori* probabilities after rejecting some of the prints may actually be invalid because it assumes that after rejecting a percentage of the prints the probability of occurrence for each class has not changed. This may actually be true but at this point we do not have the data to compute these probabilities.

$$E_i = 100.0 \times \left(\frac{W_i}{A_i} \right) \tag{15}$$

8.2 Speed

There was a significant difference seen in the time required to filter images with the three different filters. The fastest time seen for the localized FFT filter was approximately 2 seconds per image when run on a DAP 510C[1] massively parallel architecture. The fastest times for the direc-

tional FFT filters were approximately 9 seconds per image (version 1) and 5 seconds per image (version 2) when run on a i860XP 50 MHz processing board[1]. The execution times on a SUN sparc 2 workstation[1] were approximately 30 seconds (localized FFT filter), 5 minutes (version 1 directional filter) and 3 1/2 minutes (version 2 directional filter).

	Volume 1 Prototypes Volume 2 Testing			Volume 2 Prototypes Volume 1 Testing		
Image Enhancement	% error	% error with 10% rejects	σ	% error	% error with 10% rejects	σ
Equally Spaced Grids						
No Filter or Registration	18.91	14.68	2.19	21.33	16.67	2.10
Registered	17.05	12.81	2.01	18.39	13.56	1.83
Non-Equally Spaced Grids						
Registered	15.63	11.32	2.18	15.79	11.12	1.93
Localized FFT Filter	13.73	9.33	2.47	13.23	8.85	2.54
Directional FFT filter 1	14.89	10.45	2.23	14.04	9.31	2.21
Directional FFT filter 2	14.81	10.66	2.43	14.37	10.01	2.83

Table 2: Classification results for *NIST Special Database 9 Volumes 1 and 2.*

1. Certain commercial equipment is identified in order to adequately describe the subject matter of this work. In no case does such identification imply recommendation or endorsement by the National Institute of Standards and Technology, nor does it imply that the equipment identified is necessarily the best available for the purpose.

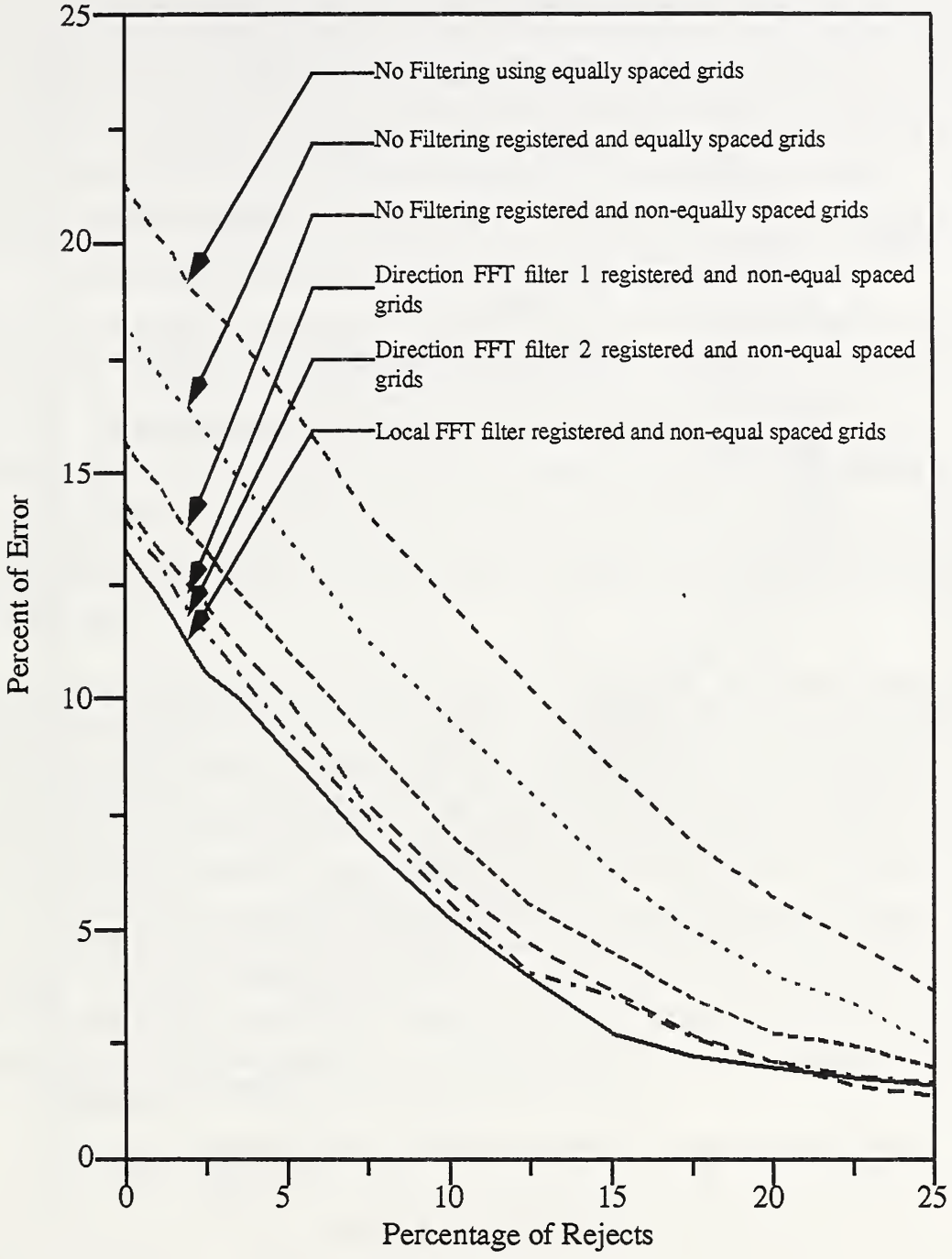

Figure 13: Error vs. reject plot for Volume 1 of *NIST Special Database 9*.

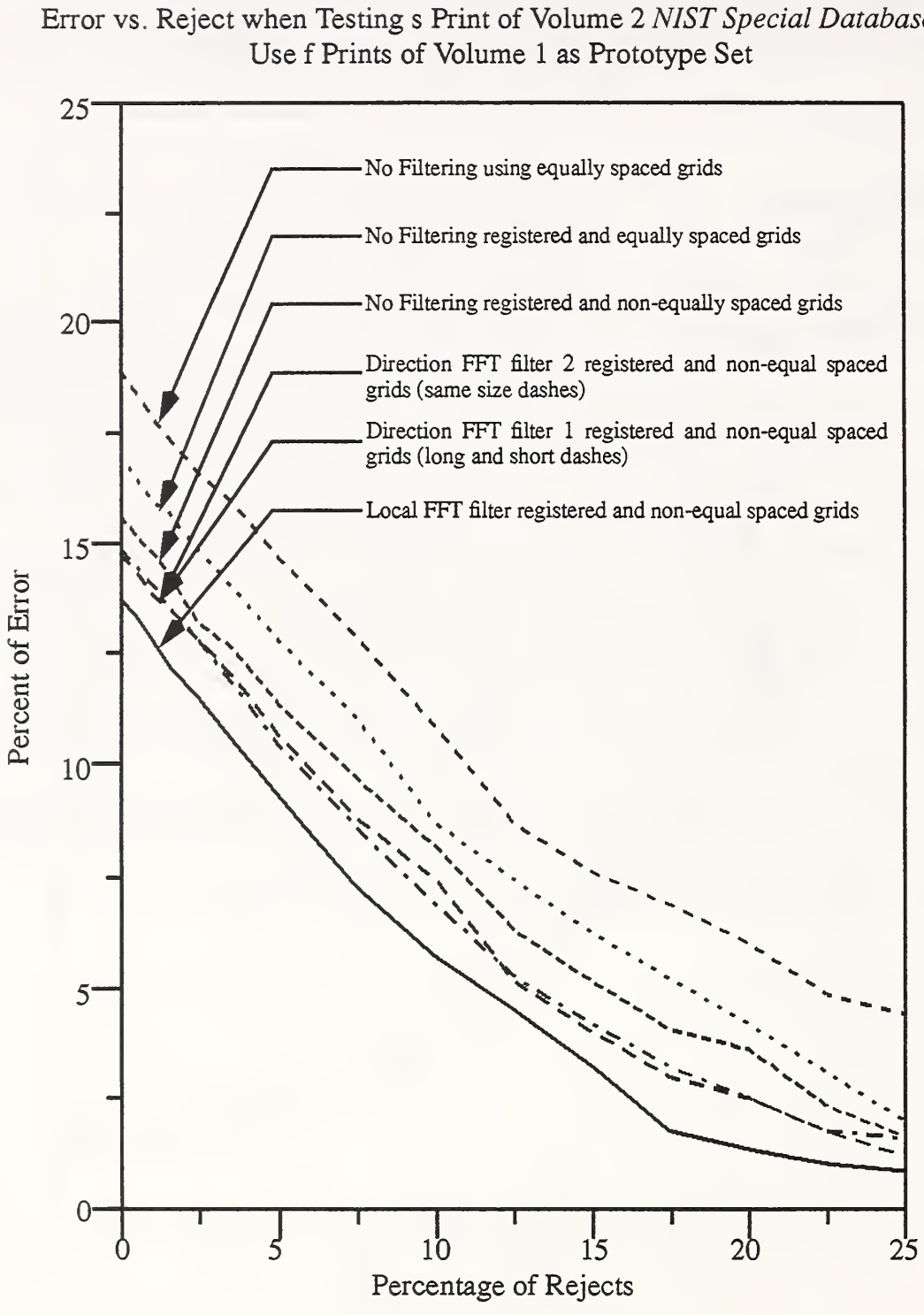

Figure 14: Error vs. reject plot for Volume 2 of *NIST Special Database 9*.

9 Conclusions

The first point that needs to be made about the results are that they can not be compared to previous results reported using *NIST Special Database 4* for two main reasons. First the data sets were collected using different techniques which affected the quality and orientations of the images. Second the test performed in this report do not use the first rollings of a set of prints as the training set and the second rollings of the same prints as the testing set as was done in previous tests. Test have shown that using different rollings of a print in the prototype and testing sets result in significantly better error rates (3-4%) versus using different prototype and testing data with a PNN classifier. For this reason, all tests in this report are done using a set of prototypes that does not contain any rollings of the prints in the testing set. This testing method produces more realistic results since one can not always expect a rolling of a fingerprint in the testing data to appear in the prototype data.

The use of filtering accomplished the main goal of providing better features the classifier as shown by improved registration and ridge flow data (Figure 9-Figure 12) and improved error rates (Table 2). The improvement from the feature vectors in Figure 9 to those in Figure 11 is shown by two facts. First the feature vectors have smoother flow from one orientation vector to the next. Second the length of the lines in the figures shows the amount of confidence that the orientation is correct, the confidences are clearly better in Figure 11. There was also an improvement of approximately 2% consistently observed for rejection rates up to 50%. Currently our system uses the localized FFT filter because it is more than twice as fast as the next fastest filter and provides the best error rates.

After carefully observing the results it was also determined that further filtering will result in very little gain in overall error rate with the current system. In a separate test a printout was made of all the fingerprints incorrectly classified after rejecting 35% of the classified prints. The printouts showed that approximately 45% were double loop whorls with accurate ridge flow data and correct registration points (as defined by the current algorithm). The problem is that the classifier is having trouble distinguishing between certain double loop whorls (Figure 15a and Figure 15b show an example print with registration point and corresponding ridge features) and loops. The same problem was occurring approximately 15% of the time with central pocket whorls that had a small number of ridges completely circling the center core (see Figure 16a and Figure 16b). The classifier also had difficulty with tented arches confusing them with loops and arches (10% of the errors). Taking into account these three cases approximately 70% of the error occurring needs to be solved by some other method than improving image enhancement.

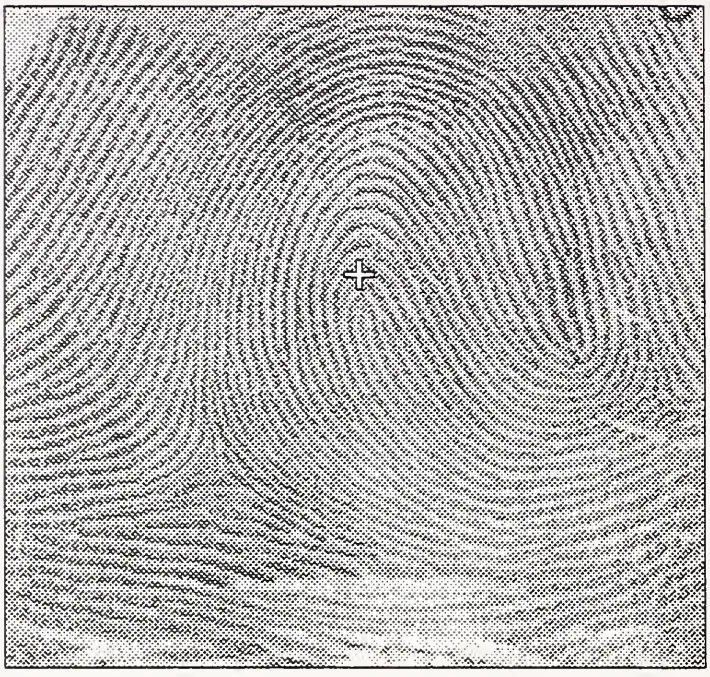

Figure 15a: Example of misclassified double loop whorl with marked registration point.

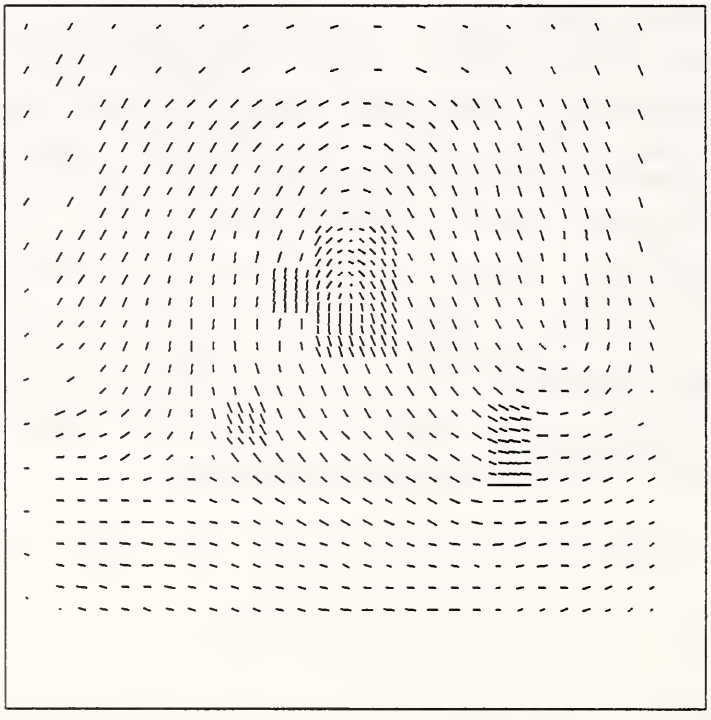

Figure 15b: Feature vectors for fingerprint image in Figure 15a.

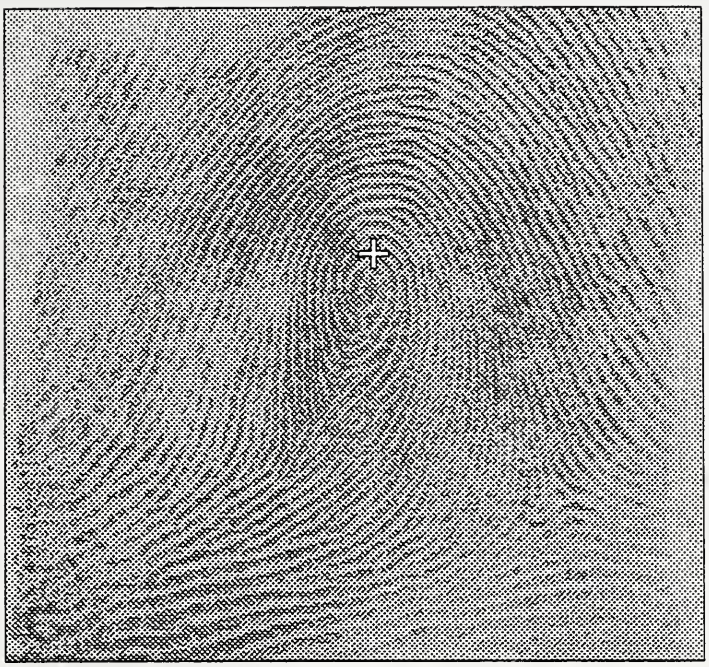

Figure 16a: Example of misclassified central pocket whorl with marked registration point.

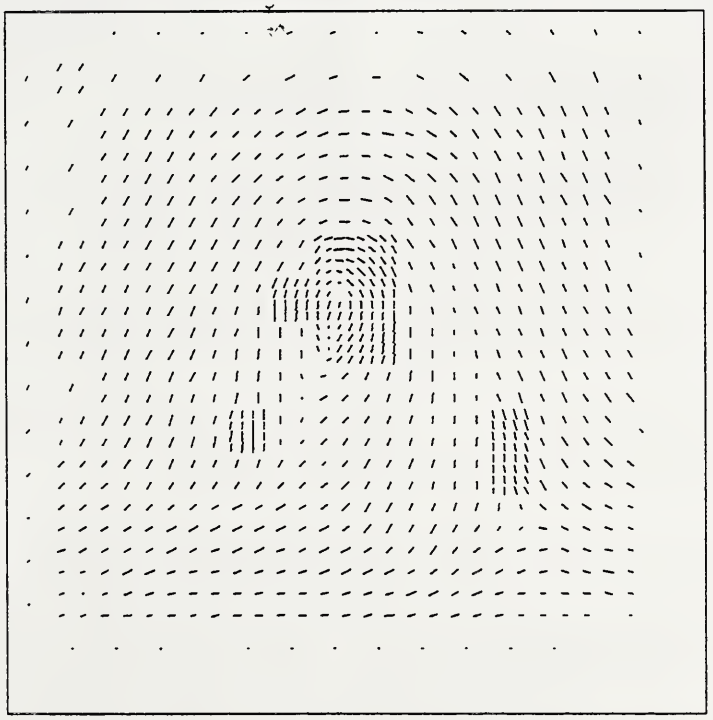

Figure 16b: Feature vectors for fingerprint image in Figure 16a.

References

[1] C.L. Wilson, G.T. Candela, P.J. Grother, C.I. Watson, and R.A. Wilkinson. Massively Parallel Neural Network Fingerprint Classification System. Technical Report **NISTIR 4880**, *National Institute of Standards and Technology*, July 1992.

[2] G. T. Candela, R. Chellappa. Comparative Performance of Classification Methods for Fingerprints. Technical Report **NISTIR 5163**, *National Institute of Standards and Technology*, April 1993.

[3] C.I. Watson. Mated Fingerprint Card Pairs. Technical Report Special Database 9, **MFCP**, *National Institute of Standards and Technology*, February 1993.

[4] C. I. Watson, C. L. Wilson, Fingerprint Database, *National Institute of Standards and Technology*, Special Database 4, **FIGS**, March 1992.

[5] B. T. Smith, J. M. Boyle, J. J. Dongarra, B. S. Garbow, Y. Ikebe, V. C. Klema, C. B. Moler, *Matrix System Routines - EISPACK Guide,* Springer-Verlag, 1976.

[6] *The Science of Fingerprints*. U. S. Department of Justice, Washington, DC, 1984.

[7] Automated classification system reader project (ACS). Technical report, DeLaRue Printrak Inc., February 1985.

[8] D. Wasson, Masters Thesis University of Maryland, 1994.

[9] R. M. Stock and C. W. Swonger. Development and evaluation of a reader of fingerprint minutiae. *Cornell Aeronautical Laboratory*, Technical Report CAL No. XM-2478-X-1:13-17, 1969.

[10] J. H. Wegstein. An automated fingerprint identification system. *National Institute of Standards and Technology*, NBS Special Publication 500-89, February 1982.

[11] C. L. Wilson, G. T. Candela, and C. I. Watson. Neural network fingerprint classification. *Journal of Artificial Neural Networks*, 1992. to be published

[12] P. J. Grother. Karhunen Loève feature extraction for neural handwritten character recognition. In *Proceedings: Applications of Artificial Neural Networks III*. Orlando, SPIE, April 1992.

[13] Donald F. Specht. Probabilistic neural networks. *Neural Networks*, 3(1):109-118, 1990.