

A Real-Time Matching System for Large Fingerprint Databases

Nalini K. Ratha, *Student Member, IEEE*, Kalle Karu, Shaoyun Chen, and Anil K. Jain, *Fellow, IEEE*

Abstract—With the current rapid growth in multimedia technology, there is an imminent need for efficient techniques to search and query large image databases. Because of their unique and peculiar needs, image databases cannot be treated in a similar fashion to other types of digital libraries. The contextual dependencies present in images, and the complex nature of two-dimensional image data make the representation issues more difficult for image databases. An invariant representation of an image is still an open research issue. For these reasons, it is difficult to find a universal content-based retrieval technique. Current approaches based on shape, texture, and color for indexing image databases have met with limited success. Further, these techniques have not been adequately tested in the presence of noise and distortions. A given application domain offers stronger constraints for improving the retrieval performance. Fingerprint databases are characterized by their large size as well as noisy and distorted query images. Distortions are very common in fingerprint images due to elasticity of the skin. In this paper, a method of indexing large fingerprint image databases is presented. The approach integrates a number of domain-specific high-level features such as pattern class and ridge density at higher levels of the search. At the lowest level, it incorporates elastic structural feature-based matching for indexing the database. With a multilevel indexing approach, we have been able to reduce the search space. The search engine has also been implemented on Splash 2—a field programmable gate array (FPGA)-based array processor to obtain near-ASIC level speed of matching. Our approach has been tested on a locally collected test data and on NIST-9, a large fingerprint database available in the public domain.

Index Terms—Image database, fingerprint matching, minutiae points, image registration, indexing, field programmable gate array.

1 INTRODUCTION

A multimedia system is expected to efficiently handle various types of data such as text, image, audio, and video. Current database technology can easily manage alphanumeric data. However, the techniques used in conventional text-oriented databases cannot handle image retrievals based on content. An obvious approach to content-based retrieval based on static annotation of images by text, followed by available database methods has a limited usage. The main drawback of this method is that the scene description can be different at different times for the same image depending on the query context. Secondly, for large databases, annotating each and every image can be a highly time consuming task. Researchers in the past had shown substantial interest in pictorial databases. But, the prevailing hardware at the time posed serious limitations. With recent advances in hardware technology, nontextual databases are receiving more attention. For textual data, the representation and retrieval issues have been addressed extensively. The same analysis cannot be extended to images, audio, and video data because of the difficulties in obtaining an invariant representation of nontextual data. For example, an object is represented in a textual database

by its name, color, and other attributes that define the object. The image of the same object may vary depending on its orientation, ambient light, and the sensor. The sensed information is of much higher dimensionality than the textual information. In a digital library, there are primarily three components:

- 1) data capture,
- 2) storage management, and
- 3) search and query techniques.

For nontextual data, the data capture stage is context dependent and highly sensitive to sensing modalities. Hence, the other two stages warrant special treatment.

Early image databases were primarily used for remotely sensed data [13], [15]. Recently, many researchers have examined various design issues for image databases [30], [32], [40], [50], [4], [47], [48]. The content-based retrieval techniques used in these systems are based on many visual cues such as color [43], texture [33], and shape [9]. Iconic indexing [14], [11] has been used for representation and retrieval. Orenstein and Manola [31] describe an image database for spatial and temporal data modeling. A data structure called “point sets” has been used to represent images and an operator called the “geometric filter” has been used to optimize spatial queries. Joseph and Cardenas [22] proposed a high-level query language, called PICQUERY for pictorial database management. This was later extended to PICQUERY⁺ by Cardenas et al. [10]. They describe a high-level domain-independent query language that supports imprecise and fuzzy descriptors. Roussopolous et al. [38] use R - and R^+ trees and a query language PSQL for cartographic database applications. Iconic indexing was first proposed

• N. Ratha, S. Chen, and A.K. Jain are with the Department of Computer Science, Michigan State University, East Lansing, MI 48824.

E-mail {ratha, chenshao}@pixel.cps.msu.edu; jain@cps.msu.edu.

• K. Karu is with the Department of Mathematics, Boston University, Boston, MA 02215.

E-mail kllkr@bu.edu.

Manuscript received July 17, 1995; revised Apr. 22, 1996. Recommended for acceptance by R.W. Picard.

For information on obtaining reprints of this article, please send e-mail to: transpami@computer.org, and reference IEEECS Log Number P96049.

by Chang et al. [14] for representing images as 2D strings. Querying using iconic indexing then reduces the problem to 2D subsequence matching. Extending this idea, Chang and Lee [11] describe a retrieval method which is based on transforming each image into a set of ordered triplets representing relationship between objects in the image and then constructing a hash table for all the triplets for a faster access.

Index-based object recognition has become popular in the computer vision community [18], [9], [42], [37]. Features extracted from an image are compared with features in the model database of objects. Geometric hashing [25] has been extensively used for this purpose. Rigoutsos and Hummel [37] have implemented geometric hashing on a parallel machine for object recognition. Recently, Califano and Mohan [9] have described an analytical framework for multidimensional indexing. Color histograms of multicolored objects proposed by Swain and Ballard [43] provide another method of indexing into a large database of models. The QBIC project [30] integrates many cues such as color, texture, and shape to efficiently retrieve contents from the image database. MIT's Photobook [32] describes a set of interactive tools for browsing and searching images and image sequences. A set of perceptually-significant coefficients are used for image matching. Similar approaches have been taken by Picard and Minka [33] for annotating a digital library of images based on image texture. For video data indexing and retrieval, Smoliar and Zhang [40] use color, texture, and shape as features and construct index tables using knowledge representation techniques. Yoshitaka, et al. [50] used a knowledge assisted content-based retrieval technique based on an object-oriented data model called MORE. Chang and Hsu [12] discuss many research issues in an image information system and suggest the use of generalized icons and active indices for content-based retrieval. Jain and Vailaya [21] describe an image retrieval technique based on color and shape with excellent results on trademark image database.

There has been a considerable amount of interest in face databases. Bach et al. [4] integrate ideas from image understanding for image retrieval and update it with example-based querying from database technology for querying. They use edge maps and other topological features from faces (e.g., nose length) as features for face image retrieval. Wu and Narasimhalu [47] describe similar methods for face retrieval based on neural networks for iconic indexing. Though a general purpose content-based retrieval system is hard to design, Wu et al. [48] attempt to integrate many basic requirements in a content-based search engine, called CORE.

For small databases (on the order of hundreds of images) most of these techniques have been shown to work well. As the database size grows, these techniques are not very successful. The performance does not scale as the cues used are primarily global in nature. Secondly, these techniques have been tested under controlled environments. Hence, their robustness to noise and distortions in real-life environments is yet to be demonstrated.

Our interest in image database representation and content-based retrieval stems from the application domain of fingerprint matching. A fingerprint database is character-

ized by a large number of records (in the order of millions). The FBI database size has grown from over 0.8 million fingerprint cards (10 fingerprints per card) in 1924 to over 114 million fingerprint cards in 1994. The storage requirements for such a large collection of images runs into 1,140 terabytes without compression [7]. Moreover, the type of queries such a system is expected to handle radically differs from the other application domains of image databases. A typical query of the type "find the best possible matches from the database to the input image" is different for a fingerprint database in the following ways:

- 1) we may not have an exact match (yes/no type);
- 2) the input image can be substantially different from the stored image in the database even though they represent the same finger;
- 3) the input image may be noisy and distorted, and finally
- 4) the input may contain only a partial image with severe distortions as in the case of a scene-of-crime fingerprint.

Both the uniqueness of the application domain as well as an inadequate treatment of the associated difficult issues warrant a thorough treatment.

The problems associated with fingerprint matching are very complex and an inappropriate representation scheme can make it intractable. This paper addresses many critical issues involved in large-scale fingerprint matching. The purpose of this paper is three fold.

- First, we describe a content-based retrieval method that can tolerate noisy and distorted query images.
- Second, an integrated multilevel approach is described to handle very large databases.
- Finally, in order to meet "real-time" matching requirements in large databases, we have implemented our search engine on Splash 2—a custom computing machine.

Splash 2 is a field programmable gate array (FPGA)-based custom computing platform. By a suitable mapping of an algorithm on FPGA-based processing elements, near-ASIC level speeds can be achieved [8].

The rest of the paper is organized as follows. The fingerprint matching problem is briefly introduced in Section 2. The various high-level features extracted for use in the retrieval stage are described in Section 3. The integrated multilevel retrieval technique is presented in Section 4. The proposed method has been implemented and tested on the NIST-9 database. The results of our tests are discussed in Section 5. In order to meet the computational needs of this content-based retrieval technique, we have mapped our algorithm on a novel custom computing machine. In Section 6, the architecture of Splash 2 and the mapping of the matching algorithm onto Splash 2 is described. Our conclusions and future work are described in Section 7.

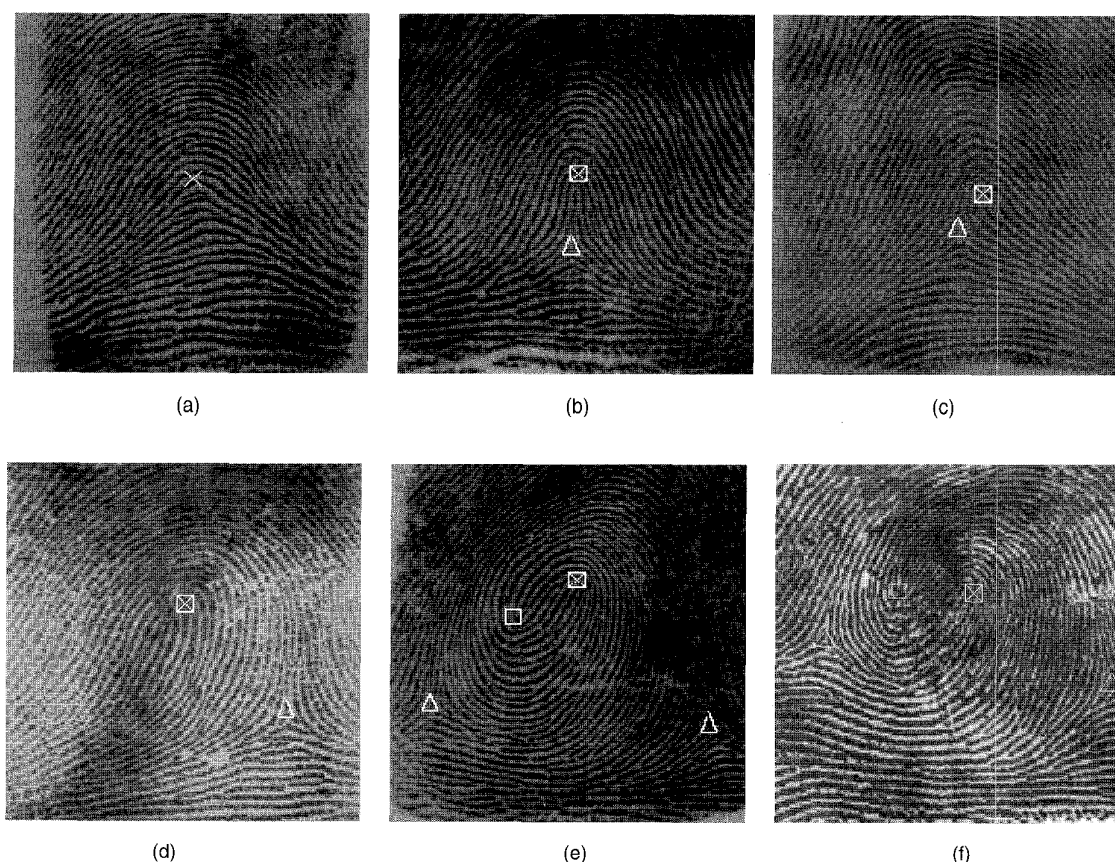


Fig. 1. Gray level fingerprint images of different types of patterns with core (□) and delta (△) points: (a) arch; (b) tented arch; (c) right loop; (d) left loop; (e) whorl; (f) twin loop.

2 FINGERPRINT MATCHING

Fingerprint matching is one of the most popular biometric techniques used in automatic personal identification [28]. Law enforcement agencies use it routinely for criminal identification. Now it is also being used in several other applications such as access control for high security installations, credit card usage verification, and employee identification [28]. The main reason for the popularity of fingerprints as a method of identification is that the fingerprint of a person is unique and features used in matching remain invariant with age. The law enforcement agencies have developed a standardized method for manually matching rolled fingerprints and latent or partial fingerprints (lifted from the scene of a crime). However, the manual matching of fingerprints is a highly tedious task for the following reasons. The fingerprint matching complexity is a function of the size of the image database, which can vary from a few hundred records to several million records. Even though the standard Henry formula [26] for fingerprint recognition can be used to cut down the search time, the manual matching can still take several days in some cases. The manual classification method makes the distribution of records uneven resulting in more work for commonly occurring fingerprint classes. These problems can be easily overcome by automating the finger-

print-based identification process.

For the purpose of automation, a suitable representation (feature extraction) of fingerprints is essential. This representation should have the following desirable properties:

- 1) retention of the discriminating power (uniqueness) of each fingerprint at several levels of resolution (detail),
- 2) easy computability,
- 3) amenable to automated matching algorithms,
- 4) stable and invariant to noise and distortions, and
- 5) efficient and compact representation.

The compactness property of representation often constrains its discriminating power. Clearly, the raw digital image of a fingerprint itself does not meet these representational requirements. Hence, high-level structural features are extracted from the image for the purpose of representation and matching.

The ridges and valleys in a fingerprint alternate, flowing in a local constant direction (see Fig. 1). A closer analysis of the fingerprint reveals that the ridges (or the valleys) exhibit anomalies of various kinds, such as ridge bifurcations, ridge endings, short ridges, and ridge crossovers. Eighteen different types of fingerprint features have been enumerated in [16]. Collectively, these features are called *minutiae*. For automatic feature extraction and matching, the set of fingerprint features is restricted to two types of minutiae:

ridge endings and *ridge bifurcations*. Ridge endings and bifurcations are shown in Fig. 2.

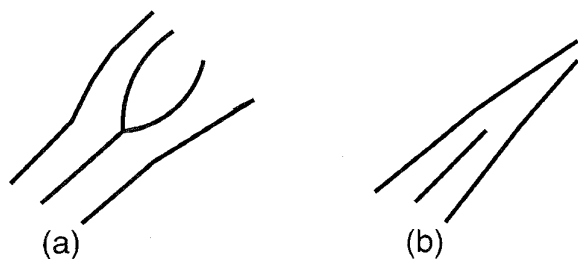


Fig. 2. Two commonly used fingerprint features: (a) ridge bifurcation; (b) ridge ending.

In a good quality, rolled fingerprint image, there are about 70 to 80 minutiae points. In a latent, or partial fingerprint, the number of minutiae is much less (approximately 20 to 30). More complex fingerprint features can be expressed as a combination of these two basic features. For example, an *enclosure* can be considered a collection of two bifurcations and a *short ridge* can be considered a pair of ridge endings as shown in Fig. 3.

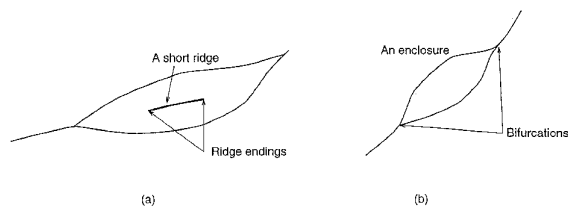


Fig. 3. Complex features as a combination of simple features: (a) short ridge; (b) enclosure.

The structural features which are commonly extracted from the gray level input fingerprint image are the ridge bifurcations and ridge endings. Each of the two types of minutiae has three attributes, namely, the x -coordinate, the y -coordinate, and the local ridge direction (θ) as shown in Fig. 4. Many other features have been derived from this basic three-dimensional feature vector [1].

Given the minutiae representation of fingerprints, matching a fingerprint against a database reduces to the problem of point matching. Due to the large size of the fingerprint database, only a small subset from the database are selected using some of the image indexing techniques described in the literature. A variety of fingerprint matching techniques have been proposed. The early work by Moayer and Fu [29] was based on syntactic pattern recognition. Later, many geometrical approaches have been adopted. The matching problem can be defined as finding a degree of match between a query and reference fingerprint feature set. The minutiae sets can be matched using many techniques, including

- 1) point set matching [46],
- 2) graph matching [20], and
- 3) sub-graph isomorphism [1].

These generic techniques are known to be intractable problems. Suboptimal solutions have been obtained using iterative procedures such as relaxation [44], simulated annealing, and genetic algorithms [3]. The large computational requirement of matching is primarily due to the following three factors:

- 1) a query fingerprint is usually of poor quality,
- 2) the fingerprint database is very large, and
- 3) structural distortion of the fingerprint images requires powerful matching algorithms.

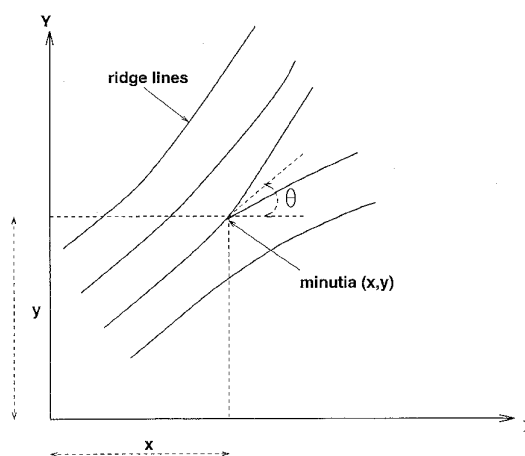


Fig. 4. Components of a minutiae feature.

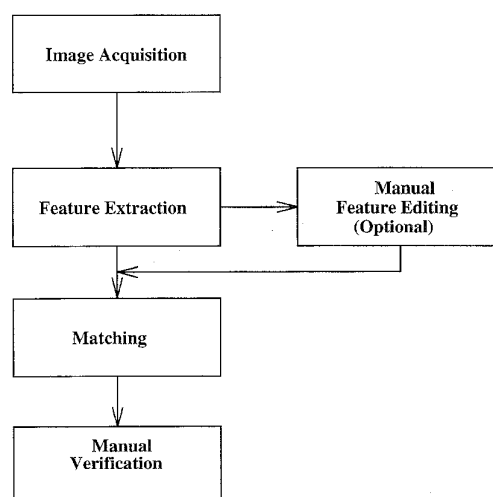


Fig. 5. Stages in an AFIS.

An automatic fingerprint identification system (AFIS) consists of various processing stages as shown in Fig. 5. Commercially available fingerprint identification systems typically use ridge bifurcations and ridge endings as features. Because of the large size of the fingerprint database and the noisy fingerprints encountered in practice, it is very difficult to achieve a reliable one-to-one matching in all the test cases. Therefore, the commercial systems provide a

ranked list of possible matches (usually the top 10 matches) which are then verified by a human expert. Details of commercial fingerprint recognition systems from NEC, PRINTRAK, and MORPHO are presented in [26].

3 FEATURE EXTRACTION

In addition to minutiae features described earlier, there are other high-level features that can be used in reducing the search space during a match. A very important feature for this purpose is the pattern class of a fingerprint. Fingerprints are classified into five main categories:

- arch,
- tented arch,
- left loop,
- right loop, and
- whorl.

The pattern class may be ambiguous in partial fingerprints and indeterminate for noisy fingerprints. An alternate class or wild-card class is assigned in these cases. Yet another high-level feature is the ridge density in a fingerprint. Ridge density can be defined as the number of ridges per unit distance. In order to make it invariant to position, the ridge density between two singular points in a fingerprint is computed. The singular points of interest to us are the *core* and *delta* points. The core point is the top most point on the inner most ridge and a delta point is the triradial point with three ridges radiating from it. Our definition of core and delta point differs from a fingerprint expert's point of view. Often, they are the same points except that in case of arch and tented arch classes, core and delta points are not defined for a fingerprint expert. We still identify these points to suit our algorithms even for arch and tented arch classes. The core and delta points have been marked for some commonly observed classes and shown in Fig. 1.

Central to our feature extraction process is the computation of ridge directions in non-overlapping windows in the fingerprint image. The predominant direction is computed as an optimal estimate of the direction vectors at each pixel in the window. Fingerprint images can be considered as an oriented texture pattern. As per the taxonomy described in [34], fingerprints can be classified as a weakly-ordered texture. The orientation field [34] is used to compute the optimal dominant ridge direction in each 16×16 window or block. The following steps are involved in the computation of the orientation field for each window:

- 1) Compute the gradient of the smoothed block. Let $G_x(i, j)$ and $G_y(i, j)$ be the gradient magnitude in x and y directions, respectively, at pixel (i, j) obtained using the 3×3 Sobel masks [5].
- 2) Obtain the dominant direction in a 16×16 block using the following equation:

$$\theta_d = \frac{1}{2} \tan^{-1} \left(\frac{\sum_{i=1}^{16} \sum_{j=1}^{16} 2G_x(i, j)G_y(i, j)}{\sum_{i=1}^{16} \sum_{j=1}^{16} (G_x(i, j)^2 - G_y(i, j)^2)} \right), \quad (1)$$

$G_x \neq 0$ and $G_y \neq 0$

Note that if either G_x or G_y is zero then the estimate of the dominant direction is trivial (0° or 90°). The angle θ_d is quantized into 16 directions.

3.1 Fingerprint Classification

The fingerprint classification problem deals with assigning a fingerprint image to one of the following classes:

- arch,
- tented arch,
- whorl,
- left loop,
- right loop, and
- reject.

A human expert can perform this classification relatively easily. For an automatic system, the problem is much more difficult because the system has to take into account the global ridge directions and also their local connectivity. Many researchers have addressed the fingerprint classification problem [35], [41], [24], [6]. Rao and Black [35] and Kawagoe and Tojo [24] used syntactic techniques for classification. Srinivasan and Murthy [41] used a classification method based on singular point detection. Blue et al. [6] used a neural network based method for classification. Our approach is similar to the approach taken in [41] except that we use Poincaré index to locate the singular points. Our algorithm [23] has three main steps,

- 1) block direction computation,
- 2) smooth directional image, and
- 3) location of singular points.

Based on the number of singular points located, a simple set of rules can be used to obtain the class label. The block diagram of the approach is shown in Fig. 6.

The singular points of interest to us are the core and delta points. The input to the classification algorithm is a 64×64 image of ridge directions. Directions are given not as angles but as vectors [6]. For each pixel, we compute its direction α in degrees ($\alpha \in [0, 180^\circ]$), multiply this by two, and represent it as a unit vector in this direction $v = (\cos 2\alpha, \sin 2\alpha)$. As a byproduct, we also obtain the confidence value in the estimated direction as the length of the averaged vector. A directional image in the vector representation can be smoothed by averaging the two components of the vectors separately. We have used a 3×3 averaging box filter. The filter can be implemented very efficiently, and by applying it several times we get a Gaussian-like smoothing of the reduced vector image. One of the most crucial problems in the fingerprint classification algorithm is to determine the amount of smoothing which should be applied to the 64×64 vector image. We have taken an iterative approach, where we smooth the image once, and then try to classify it. If the classifier fails, we smooth the image once more and

try to classify the image again. This process eventually terminates because any image, when smoothed sufficiently many times, becomes a constant directional image, and this image is classified as an arch pattern.

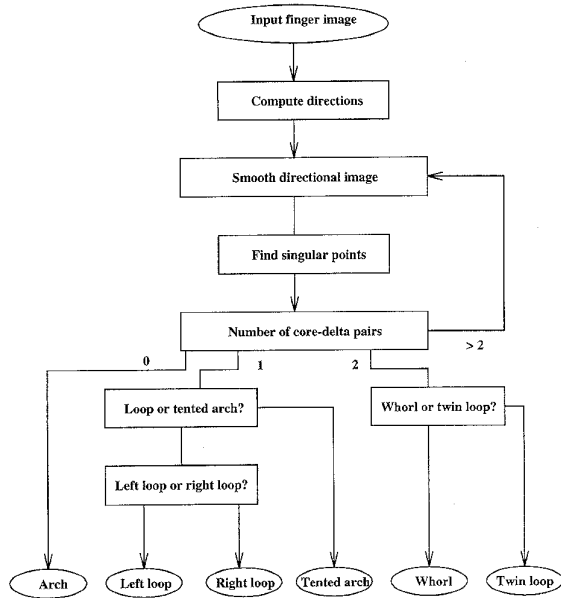


Fig. 6. Block diagram of the fingerprint classification algorithm.

A point in the directional image is classified as an ordinary point, core, or delta by computing the Poincaré index along a small closed curve around the point. The Poincaré index is computed by summing up the changes in the direction angle around the curve. When making a full counter-clockwise turn around the curve in a directional image, we see that the direction angle turns 0° , $\pm 180^\circ$, $\pm 360^\circ$, ... during this trip. A point is called ordinary if the angle has turned 0° , core if it has turned 180° , and delta if it has turned -180° (Fig. 7).

After locating all the core and delta points, we classify the fingerprint image based on the number and locations of these points. As can be seen in Fig. 8, an arch fingerprint image contains no cores or deltas; loops and tented arches contain one core and one delta; and whorls and twin loops have two cores and two deltas. We discriminate a tented arch from a loop by connecting the core and delta points with a straight line. In a tented arch image, this line's orientation is along the local direction vectors, while in a loop image the line intersects local directions transversally. Let β be the slope of the line connecting the core and delta points, and let $\alpha_1, \alpha_2, \dots, \alpha_n$ be the local direction angles on this line segment. If the averaged sum $\frac{1}{n} \sum_{i=1}^n \sin(\alpha_i - \beta)$ is less than a threshold (0.2 was used in our experiments), then the image is classified as tented arch, otherwise it is a loop image. The same technique can be used to distinguish a whorl from a twin loop. In a whorl image, the two core points can be connected along direction vectors, while in a twin loop

image they cannot be connected. Left loops are discriminated from right loops as follows. When starting from a core point and moving along the direction vectors, the delta point remains to the left in a left loop image, and to the right in a right loop image. A confidence measure is determined for the assigned class and in case of low confidence values, an alternate class or wildcard class is used. The details of the approach including classification results on NIST-4 and NIST-9 databases have been reported in [23].

3.2 Minutiae Extraction

One of the main problems in extracting structural features is the presence of noise in the fingerprint image. Commonly used methods for taking fingerprint impressions involve applying a uniform layer of ink on the finger and rolling the finger on paper. This causes the following types of problems:

- over-inked areas of the finger create smudgy areas in the image,
- breaks in ridges are created by under-inked areas, and
- the skin being elastic in nature can change the positional characteristics of the fingerprint features depending upon the pressure being applied on the fingers.

Although inkless methods for taking fingerprint impressions are now available, these methods still suffer from the positional shifting caused by the skin elasticity. Often, the non-cooperative attitude of suspects or criminals leads to smearing in parts of the fingerprint impressions. Thus, a substantial amount of research reported in the literature on fingerprint identification is devoted to image enhancement techniques.

We briefly describe our feature extraction method presented in [36]. While our approach uses many of the well-known ideas proposed in the earlier studies [39], [27], [19], [49], the ridge flow orientations form the basis for adapting parameters in all the stages of our feature extraction algorithm. We view a fingerprint image as a flow pattern with a definite texture. To accurately determine the local orientation field, the input image is divided into equal-sized blocks (windows) of 16×16 pixels. Each block is processed independently. The gray level projection along a scanline perpendicular to the local orientation field provides the maximum variance. We locate the ridges using the peaks and the variance in this projection. The ridges are thinned and the resulting skeleton image is enhanced using an adaptive morphological filter. The feature extraction stage applies a set of masks to the thinned and enhanced ridge image. The postprocessing stage deletes noisy feature points. The overall process can be divided into three main operations;

- 1) preprocessing and segmentation,
- 2) thinning and feature extraction, and
- 3) postprocessing.

The purpose of preprocessing and segmentation is to obtain a binary segmented fingerprint ridge image from an input gray scale fingerprint image, where the ridges have a value '1' (white) and rest of the image has value '0.' This is achieved through the following four steps:

- 1) computation of orientation field,
- 2) foreground/background separation,

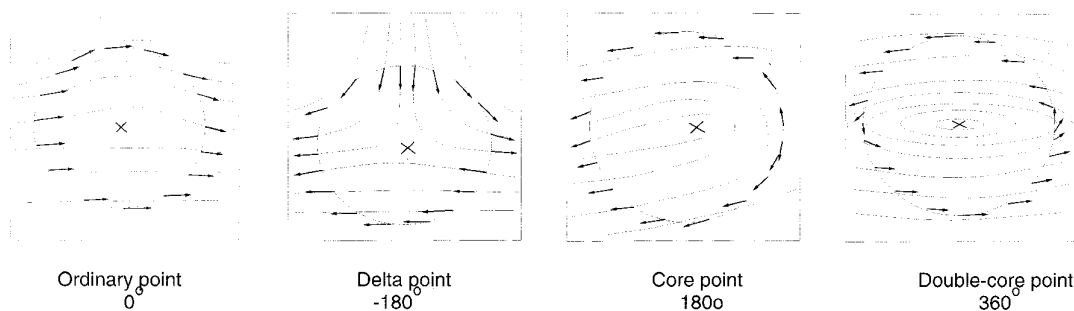


Fig. 7. Computation of the Poincaré index and the definition of ordinary, core, and delta points. The circle is centered at the point of interest, denoted as 'X.'

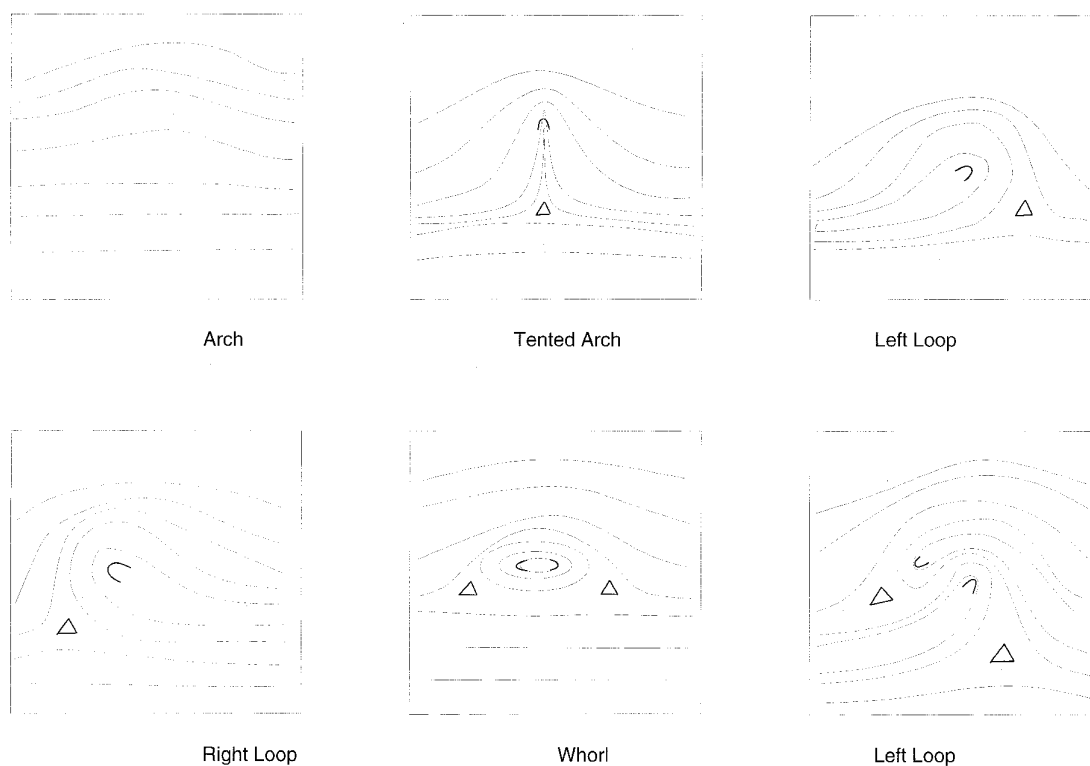


Fig. 8. Cores (C) and deltas (Δ) in fingerprint images belonging to different classes.

- 3) ridge segmentation, and
- 4) directional smoothing of ridges.

Details of our feature extraction algorithm are available in [36].

The results of intermediate stages and the detected minutiae features for a typical fingerprint are shown in Fig. 9. The performance of this approach has been evaluated by comparing the minutiae obtained using this algorithm and ground truth as obtained by a human expert. The evaluation results, including a comparative analysis with other algorithms are available in [36].

3.3 Ridge Counting

The singular point detection is a significant part of our clas-

sification procedure. Having obtained the core-delta points from the classifier module, to obtain the ridge density, we make use of the skeleton of the fingerprint used in our feature extraction stage. The number of points cutting the imaginary line between the core point and delta point on the skeleton gives us the number of ridges between the core and delta. The core and delta are highly stable points in a fingerprint. Hence, this ridge count is also a good estimate of ridge density. Manual fingerprint matching uses a similar approach [16].

4 INTEGRATED MULTILEVEL MATCHING

This section describes the various stages in our matching

for database retrieval. The matching pyramid consists of four levels, namely,

- 1) text-based,
- 2) class-based,
- 3) ridge density-based, and
- 4) minutiae-based.

The computational complexity increases as we traverse down the pyramid where as the number of records selected from the database reduces in the same direction. At the top of the pyramid, we have the whole fingerprint database available for matching. Every level of filtering/matching discards a large number of undesirable records. At the lowest level, the minutiae-based matching is used to obtain a matching score between the query fingerprint feature vector and a database fingerprint record.

A high-level description of the multilevel matching is described as follows:

- 1) Let the search space be the whole database of fingerprints.
- 2) Filter records that match the text range.
- 3) Obtain the fingerprint class information. Filter records that match with the query class or an alternate class. In case of wildcard class, include all the records obtained from previous step.
- 4) Obtain the ridge density for the query and retain the records with ridge count within a tolerance limit of the query ridge count.
- 5) Perform elastic matching on the query feature set and each of the database records retained in the previous step.

Skin elasticity demands that we accommodate elastic dis-

tortions in our minutiae matching. The text-based filters need a range-based retrieval, commonly available in standard database packages. The textual fields that can be used are the last name, age range, and colors of hair and eye. Steps 2, 3, and 4 are self explanatory after the associated features have been extracted. We describe the elastic matching algorithm in more detail.

4.1 Elastic Matching

Matching a query and a database fingerprint is equivalent to matching their minutiae sets. Each query fingerprint minutia is examined to determine whether there is a corresponding database fingerprint minutia. There are three steps involved in the elastic matching process:

- 1) registration,
- 2) minutiae pairing, and
- 3) matching score computation.

The first two steps are explained in the following subsections, followed by the overall matching algorithm.

4.1.1 Registration

In order to match two point sets with unknown orientation, scale, and translation, the two point sets must be registered with respect to each other. The orientation, scale, and translation parameters are estimated using a generalized Hough Transform [5].

The input to the registration algorithm consists of two sets of minutiae points \mathcal{P} and \mathcal{Q} extracted from two fingerprint images. For convenience, we use the following notation:

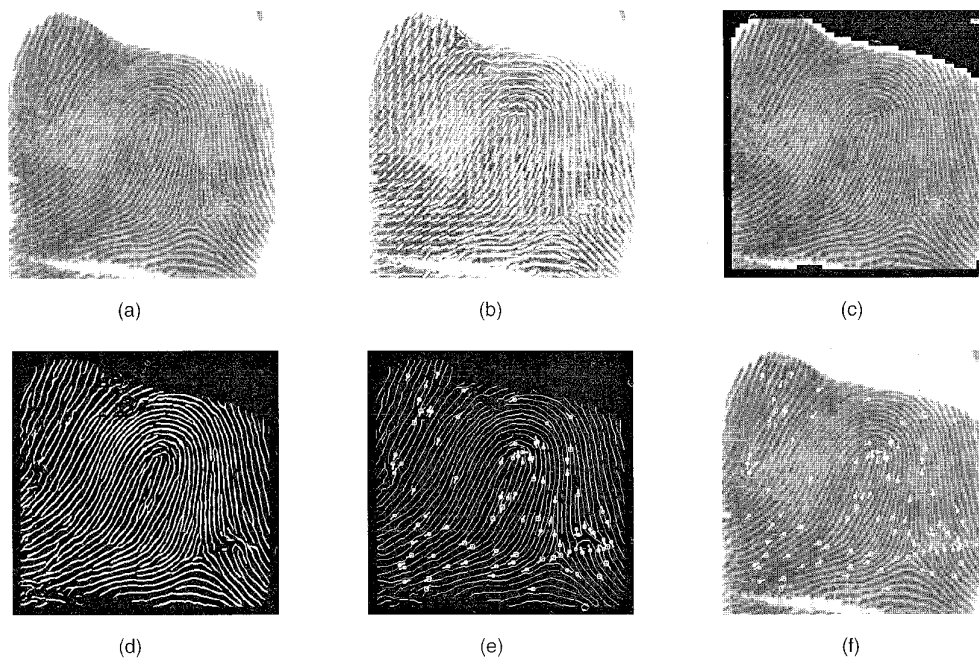


Fig. 9. Results of various stages in feature extraction: (a) original image; (b) orientation field; (c) foreground/background segmentation; (d) ridges after masking out the background; (e) skeleton with minutiae points marked; (f) minutiae superimposed on the input gray level image.

$$\mathcal{P} = \left\{ (p_x^1, p_y^1, \alpha^1), \dots, (p_x^P, p_y^P, \alpha^P) \right\},$$

$$\mathcal{Q} = \left\{ (q_x^1, q_y^1, \beta^1), \dots, (q_x^Q, q_y^Q, \beta^Q) \right\}$$

where $|\mathcal{P}| = P$, $|\mathcal{Q}| = Q$, and (p_x^i, p_y^i, α^i) are the three features (spatial position, orientation) associated with the i th minutia in set \mathcal{P} . We assume that the second fingerprint image can be obtained by applying a similarity transformation (rotation, scaling, and translation) to the first image. The second point set \mathcal{Q} is then a rotated, scaled, and translated version of the set \mathcal{P} , where points may be shifted by a random noise, some points may be added and some points deleted. Fig. 10 shows two different images of the same finger and the extracted minutiae point sets overlaid on each other. The task of fingerprint registration is to recover this unknown transformation. Since we do not know whether the two fingerprints are the same or not (i.e., images of the same finger), we attempt to find the 'best' transformation in the sense that when applying the transformation to the minutiae points of the set \mathcal{P} , as many of these points as possible overlap with the minutiae points from the set \mathcal{Q} . Two overlapping points are considered as a match only if they have the same direction. There may be minutiae points in either set that do not match with any point in the other set.

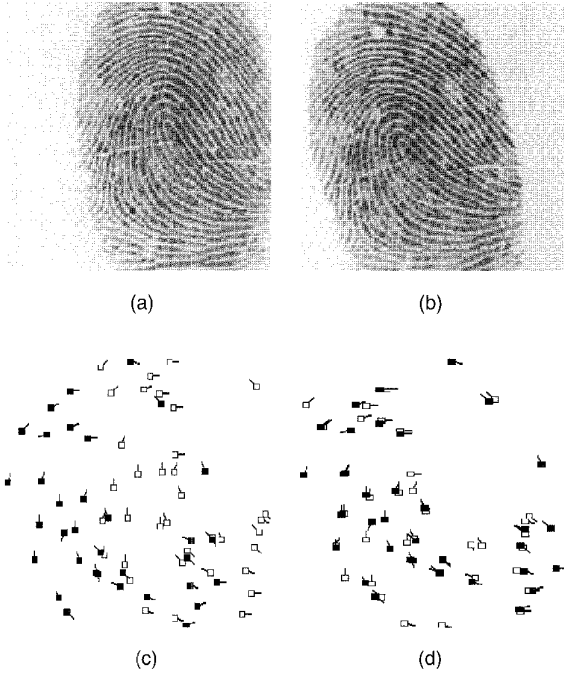


Fig. 10. Two images of the same fingerprint and overlay of their minutiae point sets: (a) first image, (b) second image, (c) overlay of minutiae points before registration, (d) overlay of minutiae points after registration.

The usual Hough transform for line detection can be generalized for point matching. We discretize the set of all allowed transformations, and for each transformation, the matching score is computed. The transformation with the maximal score is believed to be the correct one. We consider

transformations $F_{s,\theta,\Delta x,\Delta y}: R^2 \rightarrow R^2$ given by

$$F_{s,\theta,\Delta x,\Delta y} \begin{pmatrix} x \\ y \end{pmatrix} = s \begin{pmatrix} \cos \theta & \sin \theta \\ -\sin \theta & \cos \theta \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} \Delta x \\ \Delta y \end{pmatrix},$$

where s , θ , and $(\Delta x, \Delta y)$ are the scale, rotation, and shift parameters, respectively. The space of transformations consists of quadruples $(s, \theta, \Delta x, \Delta y)$, where each parameter is discretized into a finite set of values:

$$s \in \{s_1, \dots, s_K\}, \theta \in \{\theta_1, \dots, \theta_L\}, \Delta x \in \{\Delta x_1, \dots, \Delta x_M\},$$

and

$$\Delta y \in \{\Delta y_1, \dots, \Delta y_N\}.$$

Matching scores for the transformations are collected in the accumulator array A , where the entry $A(k, l, m, n)$ counts the evidence for the transformation $F_{s_k, \theta_l, \Delta x_m, \Delta y_n}$. The array A is filled as follows. For each pair (\mathbf{p}, \mathbf{q}) , where $\mathbf{p} = (p_x^i, p_y^i)$ is a point in the set \mathcal{P} and $\mathbf{q} = (q_x^j, q_y^j)$ is a point in the set \mathcal{Q} , we find all possible transformations that map \mathbf{p} to \mathbf{q} , and increment the evidence for these transformations in the array A (see Fig. 11). For every pair of values, (s_k, θ_l) , there is exactly one shift vector $(\Delta x, \Delta y)^t$ such that $F_{s_k, \theta_l, \Delta x, \Delta y}(\mathbf{p}) = \mathbf{q}$, and it can be found as:

$$\begin{pmatrix} \Delta x \\ \Delta y \end{pmatrix} = \mathbf{q} - s_k \begin{pmatrix} \cos \theta_l & \sin \theta_l \\ -\sin \theta_l & \cos \theta_l \end{pmatrix} \mathbf{p}.$$

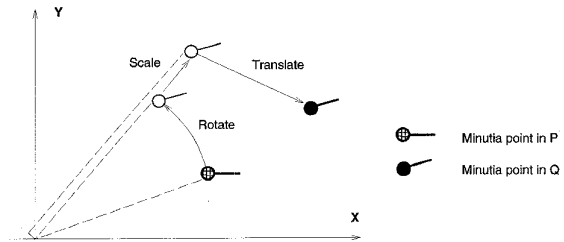


Fig. 11. Applying a similarity transformation to a minutia point.

The values Δx and Δy need to be quantized to the nearest bins corresponding to Δx_m and Δy_n . It is common in Hough transform to cast a vote not only in the correct bin $A(k, l, m, n)$, but also to its nearest neighbors. The result of applying the registration algorithm on the minutiae sets of Fig. 10c is shown in Fig. 10d. In the above description, we have disregarded the directions of minutiae points. This can be incorporated by checking if the direction of \mathbf{p} , when rotated by θ_l degrees, is the same as the direction of \mathbf{q} . The complete algorithm is given in Fig. 12.

4.1.2 Minutia Pairing

After registering the two point sets, the minutiae need to be paired. Two minutiae are said to be *paired or matched* if their components or features (x, y, θ) are equal (within some tolerance) after registration. Three situations arise as shown in Fig. 13:

- 1) A database fingerprint minutia matches the query fingerprint minutia in all the components (paired minutiae);

```

PROCEDURE Hough
  A(k, l, m, n) := 0,
  k = 1, ..., K;
  l = 1, ..., L;
  m = 1, ..., M;
  n = 1, ..., N
  FOR (Px, Py, α) ∈ P DO
    FOR (qx, qy, β) ∈ Q DO
      FOR θ ∈ {θ1, ..., θL} DO
        IF α + θ = β THEN
          FOR s ∈ {s1, ..., sv} DO
            (Δx) := (qx) - sk (cos θl sin θl) + (px)
            (Δy) := (qy) - sk (-sin θl cos θl) + (py)
            Add evidence for Fsk, θl, Δx, Δy
          END FOR
        END FOR
      END IF
    END FOR
  END FOR
  END FOR
  Result := arg maxk, l, m, n A(k, l, m, n)
END PROCEDURE

```

Fig. 12. Registration algorithm.

- 2) A database fingerprint minutia matches the query fingerprint minutia in the x and y coordinates, but does not match in the direction (minutiae with unmatched angle);
- 3) No database fingerprint minutia matches the query fingerprint minutia (unmatched minutia).

Of the three cases described above, the minutiae are said to be paired only in the first case.

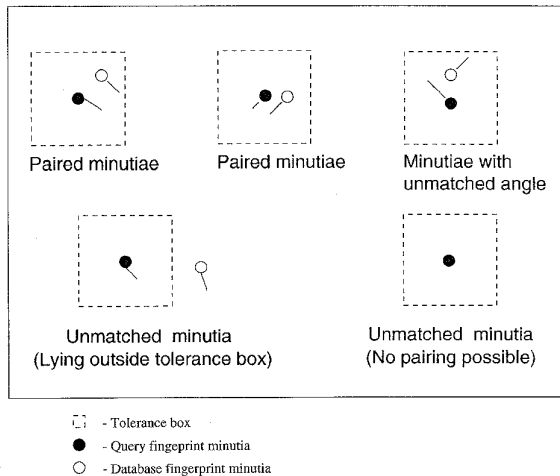


Fig. 13. Possible outcomes in minutia matching.

4.1.3 Matching Algorithm

The following notation is used in the sequential and parallel matching algorithms described below. Let the query fingerprint be represented as a set of n_q minutiae points

$f^q = (f_1^q, f_2^q, \dots, f_{n_q}^q)$. Note that each of the n_q elements is a feature vector containing three components,

$$f_i^q = (f_i^q(x), f_i^q(y), f_i^q(\theta)).$$

The components of a feature vector are shown geometrically in Fig. 4. Similarly, let the r th reference (database) fingerprint be represented as a set of n_r minutiae points $f^r = (f_1^r, f_2^r, \dots, f_{n_r}^r)$. Let (x_q^t, y_q^t) and (x_q^b, y_q^b) define the bounding box for the query fingerprint, where x_q^t is the x-coordinate of the top left corner of the box and x_q^b is the x-coordinate of the bottom right corner of the box. Quantities y_q^t and y_q^b are defined similarly. A bounding box is the smallest rectangle that encloses all the feature points.

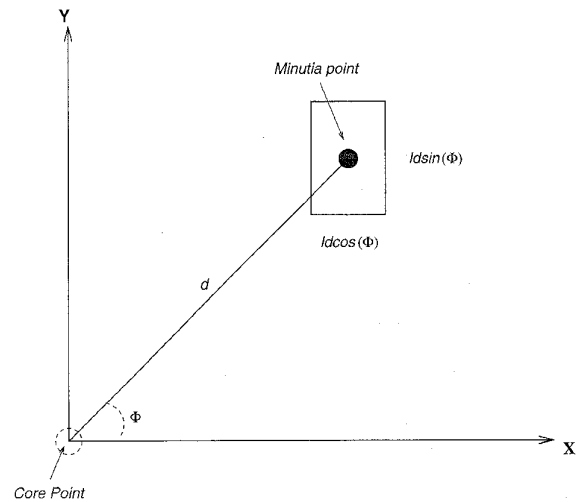


Fig. 14. Tolerance box for X- and Y-components.

The matching algorithm is based on finding the number of paired minutiae between each database fingerprint and the query fingerprint. A tolerance box is shown graphically in Fig. 14. In order to reduce the amount of computation, the matching algorithm takes into account only those minutiae that fall within a common bounding box. The common bounding box is the intersection of the bounding box for query and reference (database) fingerprints. Once the count of matching minutiae is obtained, a matching score is computed. The matching score is used for deciding the degree of match. Finally, a set of top 10 scoring reference fingerprints is obtained as a result of matching. Note that the query fingerprint f^q may or may not belong to the fingerprint database f^D .

In order to accommodate the shift in the minutia features, a tolerance box is created around each feature (see Fig. 14). The size of the box depends on the ridge widths and distance from the core point in the fingerprint. The sequential matching algorithm is described in Fig. 15. Depending on the desired matching accuracy, more than one finger could be used in matching. In that case, a composite score is computed for each set of fingerprints for a person.

5 EXPERIMENTAL RESULTS

To test our approach, we created the following databases:

- *Test Set 1*: Two hundred live scan images of fingerprints captured using inkless fingerprint scanners. The database contains 10 different versions of each fingerprint that have been rotated and translated randomly during acquisition.
- *Test Set 2*: A database of 1,800 fingerprints chosen from NIST-9 [45] database. The NIST-9 database consists of a total of 1,350 mated fingerprint card pairs (10 fingerprints per card) with a total of 27,000 fingerprints. For each fingerprint there are two different impressions available in the database. Our database has been created with the feature vector sets from the first set and tested using the second set.

On a SPARCstation 20, our feature extraction algorithm takes 31.7 seconds for a 768×832 image. The registration step takes less than one second for a query fingerprint and the matching time for a query fingerprint with a single database record is about 15 msecs. For Test Set 1, a leave-one-out method has been used for testing the matching technique. A fingerprint is said to be matched if the correct match appears in the shortlist (of 10) obtained by the matching algorithm. We achieved 100% accuracy with this method.

A random sample of 100 fingerprints was chosen to test the matching algorithm on the NIST-9 database. With a reject rate of 10%, we could achieve an 80% accuracy on this database. The overall accuracy can be improved by using more than one fingerprint per person. It can be easily seen that by using two fingerprints, the error rate falls sharply to less than five percent. Using the fingerprint class information, the search space could be reduced to 25% of the database and the use of ridge counts reduced the number of records by another 50%.

Although the matching step takes about one second per match, the total time taken to match against a large database can be very high since the registration has to be done first for each record in the database. For example, for our database of 1,800 records, registration with the query image can take up to 30 minutes without using the fingerprint class and ridge count information. To carry out a leave-one-out method over this database will take several days! In order to speedup the matching process, we need to use special hardware accelerators. Many parallel implementations of Hough Transform have been reported, including one on Splash 2 [2]. Hence, we chose to implement the matching algorithm on Splash 2—a novel custom computing machine.

6 PARALLEL MATCHING ALGORITHM ON SPLASH 2

Custom computing machines are becoming popular with the advent of field-programmable gate arrays (FPGAs). Using FPGA-based custom computing machines, application specific instructions can be generated on the hardware [17]. We have used Splash 2—a Xilinx 4010 based custom computing machine developed by the Supercomputing Research Center. In this section, the parallel matching algo-

Input: A set of n_q minutiae points in the query fingerprint \mathbf{f}^q and the rolled fingerprint database $\mathbf{f}^D = \{\mathbf{f}^r\}_{r=1}^N$. Let the r th database fingerprint have n_r minutiae points $\mathbf{f}^r = (f_1^r, f_2^r, \dots, f_{n_r}^r)$.

Output: A list of top 10 records from the database with matching score greater than a threshold T .

Begin

FOR $r = 1$ to N DO

1. Register the database fingerprint with respect to the query fingerprint.
2. Compute the common bounding box for the query and reference fingerprints. Let the query print have n_q^b and reference print have n_r^b minutiae in this box.
3. Set the number of paired minutiae for the r th database fingerprint m^r to zero.

FOR $i = 1$ to n_q^b DO

Compute the tolerance vector for the i th minutiae points in the r th database fingerprint feature vector \mathbf{f}_i^r .

If it can be paired with a query minutiae, then increment m^r and mark the query minutiae paired. A paired query minutiae will not be paired again.

END FOR

4. Compute the matching score ($MS(q, r)$):

$$MS(q, r) = \frac{m^r * m^r}{n_r^b * n_q^b}.$$

5. Update a list of top 10 scoring database fingerprints.

END FOR

END

Fig. 15. Sequential fingerprint matching algorithm.

rithm on Splash 2 is described after a brief introduction to Splash 2 architecture.

6.1 Splash 2 Architecture

The Splash 2 system consists of an array of Xilinx 4010 FPGAs [8]. Fig. 16a shows a system-level view of the Splash 2 architecture. Splash 2 is connected to the host through an interface board that extends the address and data buses. The Sun host can read/write to memories and memory-mapped control registers of Splash 2 via these buses. Each Splash 2 processing board has 16 Xilinx 4010s as PEs ($X_1 - X_{16}$) in addition to a 17th Xilinx 4010 (X_0) which controls the data flow into the processor board. Each PE has 512 KB of memory. The Sun host can read/write this memory. There is a 36-bit linear data path (SIMD Bus) running through all the PEs. The PEs can read/write data from their respective memory through a private address and data bus after setting appropriate control signals. The PEs are connected through a crossbar that is programmed by X_0 . A broadcast path also exists by

suitably programming X_0 . The processor organization for a PE is shown in Fig. 16b.

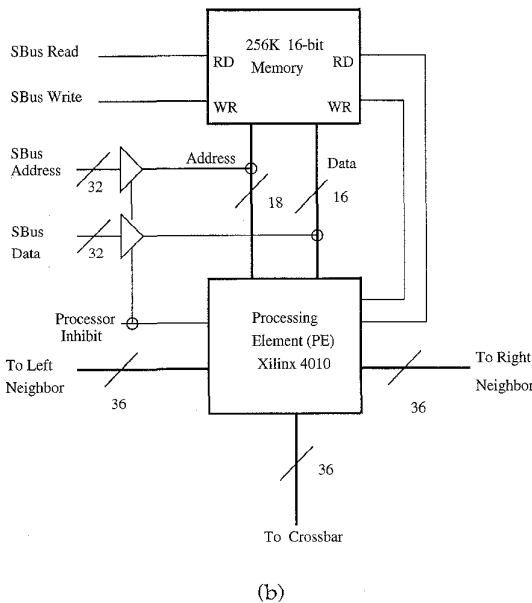
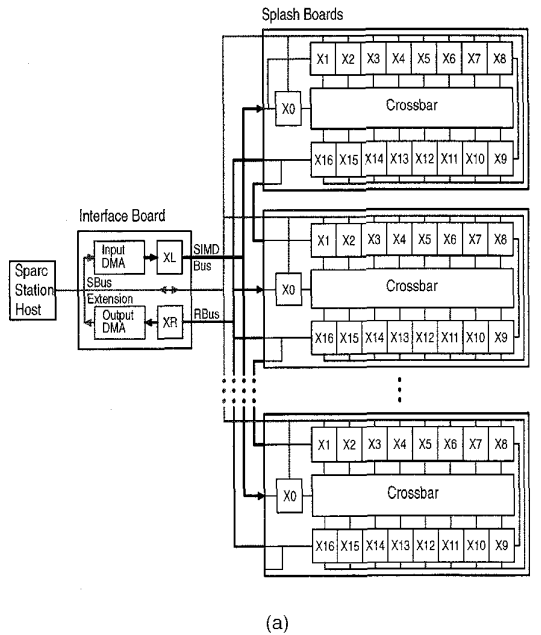


Fig. 16. Splash 2: (a) architecture; (b) one processing element.

The Splash 2 system supports several models of computation, including PEs executing the single instruction on multiple data (SIMD mode) and PEs executing multiple instructions on multiple data (MIMD mode). It can also execute the same or different instructions on single data by receiving data through the global broadcast bus. The most common mode of operation is systolic in which the SIMD Bus is used for data transfer. Individual memory available with each PE makes it convenient to store temporary results and tables.

The Xilinx 4010 consists of 400 (20×20) configurable logic blocks (CLB). Programming an FPGA-based computer

is different from usual high-level programming. The design automation process consists of two steps: simulation and synthesis. The programming flow for Splash 2 is shown in Fig. 17. In simulation, the logic designed using VHDL is verified. This involves comparing the results of the VHDL simulation with those obtained manually or by a sequential program. In synthesis, the main concern is to achieve the best placement of the logic in an FPGA in order to minimize the timing delay.

To program Splash 2, we need to program each of the PEs (X_1 - X_{16}), the crossbar, and the host interface. The crossbar sets the communication paths between PEs. In case the crossbar is used, X_0 needs to be programmed. The host interface takes care of data transfers in and out of the Splash 2 board. A special library is available for these facilities for VHDL programming as described in [8]. The synthesis process involves the following steps:

- Obtain a vendor specific netlist from the VHDL source code;
- Partition, Placement, and Routing: to fit the logic generated onto a physical PE;
- Delay and Timing Analysis: to analyze the timing and delay;
- Control bit stream generation; and
- Bit stream to raw file generation.

The raw file is loaded onto each PE, and a configuration file for the crossbar is used to describe the crossbar usage. The host uses these files to control the attached processor through a set of routines callable by a C program.

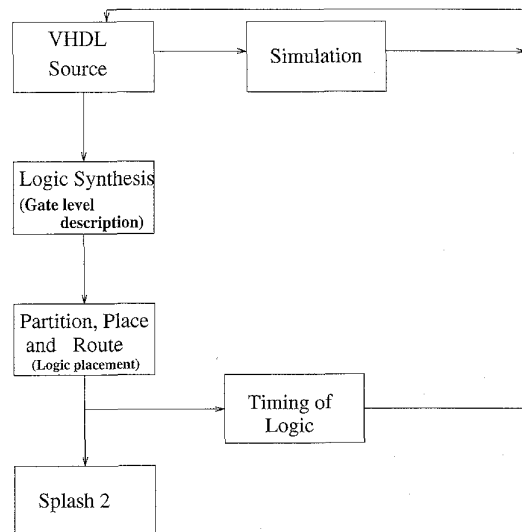


Fig. 17. Programming flow for Splash 2.

6.2 Parallel Elastic Matching Algorithm

We parallelize the matching algorithm by exploiting the specific characteristics of Splash 2 architecture. While performing this mapping, we need to take into account the limitations of the available FPGA technology. Any pre-processing needed on the query minutiae set is an one-

time operation, whereas reference fingerprint minutiae matching is a repetitive operation. Computing the matching score involves floating point division. The floating point operations and one-time operations are performed in software on the host whereas the repetitive operations are delegated to the FPGA-based PEs of Splash 2. The parallel version of the algorithm involves operations on the host, on X_0 , and on each PE.

One of the main constructs of the parallel algorithm is a lookup table which consists of all possible points within the tolerance box that a feature may be mapped to.

6.3 Preprocessing on the Host

The host processes the query and database fingerprints as follows. The query fingerprint is read first and the following preprocessing is undertaken:

- 1) For each query feature (minutiae point) f_j^q , $j = 1, 2, \dots, n_q$, generate a tolerance box. Enumerate a total of $(t_x \times t_y \times t_\theta)$ grid points in this box, where t_x is the tolerance in x , t_y is the tolerance in y and t_θ is tolerance in θ .
- 2) Allocate each feature to one PE in Splash 2. Repeat this cyclically, i.e., features 1-16 are allocated to PEs X_1 to X_{16} , features 17-32 are allocated to PEs X_1 to X_{16} , and so on.
- 3) Initialize the lookup tables by loading the grid points within each tolerance box in Step 1 into the memory.

In this algorithm, the tolerance box is computed with respect to the query fingerprint features. The host then reads the database of fingerprints and sends their feature vectors for matching to the Splash 2 board.

For each database fingerprint, the host performs the following operations:

- 1) Read the feature vectors.
- 2) Register the features as described in Step 1 of the sequential algorithm in Fig. 15.
- 3) Send each of the feature vectors over the Broadcast Bus to all PEs if it is within the bounding box of the query fingerprint.

For the r th database fingerprint, the host then reads the number of paired features m^r , $r = 1, \dots, N$, that was computed by the Splash 2 system, where N is the number of records in the database. Finally, the matching score is computed as in the sequential method.

6.4 Computations on Splash

The computations carried out on each PE of Splash 2 are described below. As mentioned earlier, X_0 plays a special role in controlling the crossbar in Splash 2.

- 1) Operations on X_0 :
Each database feature vector received from the host is broadcast to all PEs. If it is matched with a feature in a lookup table, the PE drives the Global OR Bus high. When this OR Bus is high, X_0 increments a counter. The host reads this counter value (m^r) after all the feature vectors for the current database fingerprint have been processed.
- 2) Operations on each PE:

On receiving the broadcast feature, a PE computes its address in the lookup table through a hashing function. If the data at the computed address is a '1', then the feature is paired, and the PE drives the Global OR Bus high.

The Splash 2 data paths for the parallel algorithm are shown in Fig. 18.

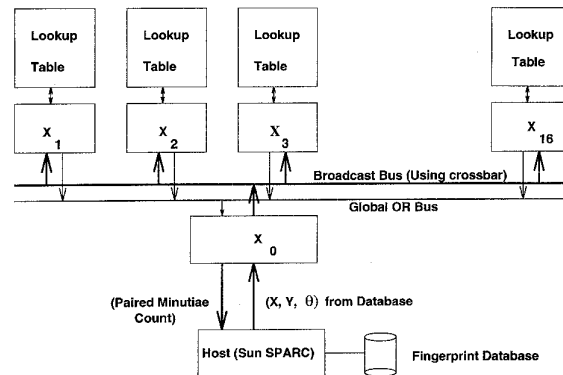


Fig. 18. Data flow in parallel algorithm.

6.5 Performance

Using Splash 2, the point pattern matching under the conditions described earlier can be executed at the rate of 110,000 matches per second. This high speed has been achieved by using an instruction set specifically designed for our matching algorithm and an optimal mapping of the sequential algorithm on the Splash 2 architecture. Details of the implementation are described in [8].

7 CONCLUSIONS AND FUTURE WORK

We have described a method for structural feature-based indexing into large fingerprint image databases. The approach has been tested using a large public domain database from NIST. Our multilevel matching algorithm employs an elastic matching technique as a central component. The elastic matching has been implemented on a special-purpose hardware to show that desired matching speeds needed to access large fingerprint databases can be achieved. Although the feature extraction, indexing, and retrieval techniques described here have been tested with fingerprint images, we strongly believe that these techniques can be applied to other similar images such as fluid flow images and lumber images.

Currently, we are in the process of enlarging our database. In feature extraction, we are examining measures to assign a confidence level to a feature. Often, poor quality images lower the system accuracy. Hence, an evaluation of image quality at the input stage to accept or reject an input is being considered. More complex and robust matching algorithms are also under evaluation.

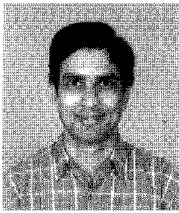
ACKNOWLEDGMENTS

We thank Dr. Sharath Pankanti and Dr. Ruud Bolle of the

IBM T. J. Watson Research Center for their valuable comments and suggestions.

REFERENCES

- [1] "Application Briefs: Computer Graphics in the Detective Business," *IEEE Computer Graphics and Applications*, vol. 5, no. 4, pp. 14–17, Apr. 1985.
- [2] A.L. Abbott, P.M. Athanas, L. Chen, and R.L. Elliott, "Finding Lines and Building Pyramids with Splash 2," *Proc. Second IEEE Workshop FPGAs for Custom Computing Machines*, Napa Valley, Calif., pp. 155–164, Apr. 1994.
- [3] N. Ansari, M.-H. Chen, and E.S.H. Hou, "A Genetic Algorithm for Point Pattern Matching," *Dynamic, Genetic, and Chaotic Programming*, B. Soucek, ed., pp. 353–371. New York: John Wiley and Sons, 1992.
- [4] J.R. Bach, S. Paul, and R. Jain, "A Visual Information Management System for the Interactive Retrieval of Faces," *IEEE Trans. Knowledge and Data Engineering*, vol. 5, no. 4, pp. 619–628, Aug. 1993.
- [5] D.H. Ballard and C.M. Brown, *Computer Vision*. Englewood Cliffs, NJ: Prentice-Hall, 1982.
- [6] J.L. Blue, G.T. Candela, P.J. Grother, R. Chellappa, and C.L. Wilson, "Evaluation of Pattern Classifiers for Fingerprint and OCR applications," *Pattern Recognition*, vol. 27, no. 4, pp. 485–501, Apr. 1994.
- [7] J.N. Bradley and C.M. Brislawn. The Wavelet/Scalar Quantization Standard for Digital Fingerprint Image Compression. *Proc. IEEE ISCAS-94*, London, vol. 3, pp. 205–208, 1994.
- [8] *Splash 2: FPGAs for Custom Computing Machines*, D.A. Buell, J.M. Arnold, and W.J. Kleinfelder, eds. Los Alamitos, Calif.: IEEE Computer Society Press, 1996.
- [9] A. Califano and R. Mohan, "Multidimensional Indexing for Recognizing Visual Shapes," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 16, no. 6, pp. 373–392, Apr. 1994.
- [10] A.F. Cardenas, I.T. Jeong, R.K. Taira, and C.M. Breant, "The Knowledge-Based Object-Oriented PICQUERY Language," *IEEE Trans. Knowledge and Data Engineering*, vol. 5, no. 4, pp. 644–657, Aug. 1993.
- [11] C.C. Chang and S.Y. Lee, "Retrieval of Similar Pictures on Pictorial Databases," *Pattern Recognition*, vol. 24, no. 7, pp. 675–680, July 1991.
- [12] S.-K. Chang and A. Hsu, "Image Information Systems: Where Do We Go from Here?" *IEEE Trans. Knowledge and Data Engineering*, vol. 4, no. 5, pp. 431–442, Oct. 1992.
- [13] S.-K. Chang and S.-H. Liu, "Picture Indexing and Abstraction Techniques for Pictorial Databases," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 6, no. 4, pp. 475–483, July 1984.
- [14] S.-K. Chang, Q.-Y. Shi, and C.-W. Yan, "Iconic indexing by 2D Strings," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 9, no. 3, pp. 413–428, May 1987.
- [15] M. Chock, A.F. Cardenas, and A. Klinger, "Database Structure and Manipulation Capabilities of a Picture Database Management System (PICDMS)," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 6, no. 4, pp. 484–492, July 1984.
- [16] *The Science of Fingerprints: Classification and Uses*, Federal Bureau of Investigation. Washington, D.C.: U.S. Government Printing Office, 1984.
- [17] M. Gokhale, W. Holmes, A. Kopser, S. Lucas, R. Minnich, D. Sweely, and D. Lopresti, "Building and Using Highly Parallel Programmable Logic Array," *Computer*, vol. 24, no. 1, pp. 81–89, Jan. 1991.
- [18] W.I. Gorsky and R. Mehrotra, "Index-Based Object Recognition in Pictorial Data Management," *Computer Vision, Graphics, and Image Processing*, vol. 52, pp. 416–436, 1990.
- [19] D.C.D. Hung, "Enhancement and Feature Purification of Fingerprint Images," *Pattern Recognition*, vol. 26, no. 11, pp. 1,661–1,671, Nov. 1993.
- [20] D.K. Iseñor and S.G. Zaky, "Fingerprint Identification Using Graph Matching," *Pattern Recognition*, vol. 19, no. 2, pp. 113–122, 1986.
- [21] A.K. Jain and A. Vailaya, "Image Retrieval Using Color and Shape," *Proc. ACCV*, vol. 2, pp. 529–533, Singapore, 1995.
- [22] T. Joseph and A.F. Cardenas, "PICQUERY: A High Level Query Language for Pictorial Database Management," *IEEE Trans. Software Engineering*, vol. 14, no. 5, pp. 630–638, May 1988.
- [23] K. Karu and A.K. Jain, "Fingerprint Classification," *Pattern Recognition*, vol. 29, no. 3, pp. 389–404, Mar. 1996.
- [24] M. Kawagoe and A. Tojo, "Fingerprint Pattern Classification," *Pattern Recognition*, vol. 17, no. 3, pp. 295–303, 1984.
- [25] Y. Lamdan and H.J. Wolfson, "Geometric Hashing: A General and Efficient Model-Based Recognition Scheme," *Proc. Second Int'l Conf. Computer Vision*, pp. 238–249, 1988.
- [26] *Advances in Fingerprint Technology*, H.C. Lee and R.E. Gaensslen, eds. New York: Elsevier, 1991.
- [27] B.M. Mehtre, "Fingerprint Image Analysis for Automatic Identification," *Machine Vision and Applications*, vol. 6, pp. 124–139, 1993.
- [28] B. Miller, "Vital Signs of Identity," *IEEE Spectrum*, vol. 31, no. 2, pp. 22–30, Feb. 1994.
- [29] B. Moayer and K.S. Fu, "A Syntactic Approach to Fingerprint Pattern Recognition," *Pattern Recognition*, vol. 7, pp. 1–23, 1975.
- [30] W. Niblack, R. Barber, W. Equitz, M. Flickner, E. Glasman, D. Petkovic, P. Yanker, C. Faloutsos, and G. Taubin, "The QBIC Project: Querying Image by Content Using Color, Texture, and Shape," *Proc. SPIE Storage and Retrieval for Image and Video Databases*, pp. 173–187, San Jose, Calif., 1993.
- [31] J.A. Orenstein and F.A. Manola, "PROBE Spatial Data Modeling and Query Processing in an Image Database Application," *IEEE Trans. Software Engineering*, vol. 14, no. 5, pp. 611–629, May 1988.
- [32] A. Pentland, R.W. Picard, and S. Sclaroff, "Photobook: Tools for Content-Based Manipulation of Image Databases," *Proc. SPIE Storage and Retrieval for Image and Video Databases II*, pp. 34–47, San Jose, Calif., 1994.
- [33] R.W. Picard and T.P. Minka, "Vision Texture for Annotation," *Multimedia Systems*, vol. 3, pp. 3–14. Springer-Verlag, 1995.
- [34] A.R. Rao, *A Taxonomy for Texture Description and Identification*. New York: Springer-Verlag, 1990.
- [35] C.V.K. Rao and K. Black, "Type Classification of Fingerprints," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 2, no. 223–231, pp. 31, 1980.
- [36] N.K. Ratha, S. Chen, and A.K. Jain, "Adaptive Flow Orientation Based Texture Extraction in Fingerprint Images," *Pattern Recognition*, vol. 28, no. 11, pp. 1,657–1,672, Nov. 1995.
- [37] I. Rigoutsos and R. Hummel, "Massively Parallel Model Matching," *Computer*, vol. 25, no. 2, pp. 33–42, Feb. 1992.
- [38] N. Roussopolous, C. Faloutsos, and T. Sellis, "An Efficient Pictorial Database System for PSQL," *IEEE Trans. Software Engineering*, vol. 14, no. 5, pp. 639–650, May 1988.
- [39] B.G. Sherlock, D.M. Monro, and K. Millard, "Fingerprint Enhancement by Directional Fourier Filtering," *Proc. Visual Image Signal Processing*, vol. 141, no. 2, pp. 87–94, Apr. 1994.
- [40] S.W. Smoliar and H. Zhang, "Content-Based Video Indexing Retrieval," *IEEE Multimedia*, pp. 62–72, Summer 1994.
- [41] V.S. Srinivasan and N.N. Murthy, "Detection of Singular Points in Fingerprint Images," *Pattern Recognition*, vol. 25, no. 2, pp. 139–153, Feb. 1992.
- [42] F. Stein and G. Medioni, "Structural Indexing: Efficient 2D Object Recognition," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 14, no. 12, pp. 1,198–1,204, Dec. 1992.
- [43] M.J. Swain and D.H. Ballard, "Color Indexing," *Int'l J. Computer Vision*, vol. 7, no. 1, pp. 11–32, 1991.
- [44] J. Ton and A.K. Jain, "Registering Landsat Images by Point Matching," *IEEE Trans. Geoscience and Remote Sensing*, vol. 27, no. 5, pp. 642–651, Sept. 1989.
- [45] C.I. Watson, *NIST Special Database 9: Mated Fingerprint Card Pairs*, Advanced Systems Division, Image Recognition Group, National Institute for Standards and Technology, Feb. 1993.
- [46] J.H. Wegstein, "An Automated Fingerprint Identification System," Technical Report 500-89, National Bureau of Standards, Bethesda, Md., 1982.
- [47] J.K. Wu and A.D. Narasimhalu, "Identifying Faces Using Multiple Retrievals," *IEEE Multimedia*, pp. 27–38, Summer 1994.
- [48] J.K. Wu, A.D. Narasimhalu, B.M. Mehtre, C.P. Lam, and Y.J. Gao, "CORE: A Content-Based Retrieval Engine for Multimedia Information Systems," *Multimedia Systems*, vol. 3, no. 25, pp. 25–41, Springer-Verlag, 1995.
- [49] Q. Xiao and H. Raafat, "Fingerprint Image Post-processing: A Combined Statistical and Structural Approach," *Pattern Recognition*, vol. 24, no. 10, pp. 985–992, Oct. 1991.
- [50] A. Yoshitaka, S. Kishida, M. Hirakawa, and T. Ichikawa, "Knowledge-Assisted Content Based Retrieval for Multimedia Databases," *IEEE Multimedia*, pp. 12–21, Winter 1994.



Nalini K. Ratha received his BTech degree in electrical engineering and the MTech degree in computer science and engineering from the Indian Institute of Technology, Kanpur, in 1982 and 1984, respectively. He is currently a PhD student in the Department of Computer Science at Michigan State University.

His current research interests include computer vision, pattern recognition, and special architectures for vision and image processing. He is a student member of the IEEE.



Kalle Karu received a diploma from Tartu University, Estonia, in 1983, and an ME degree in computer science from Michigan State University. He is currently a graduate student at Boston University.



Shaoyun Chen received his BS degree from Xiamen University in 1988, and the ME degree from Tsinghua University, China, in 1991, both in computer science. He is currently a PhD student at Michigan State University.

His current research interests include pattern recognition, machine vision, and their applications to robotics.



Anil K. Jain is a university distinguished professor and chairs the Department of Computer Science at Michigan State University.

His current research interests are computer vision, image processing, neural networks, and pattern recognition. He has made significant contributions and published a large number of papers on the following topics: statistical pattern recognition, exploratory pattern analysis, neural networks, Markov random fields, texture analysis, interpretation of range images, and 3D object recognition.

Several of his papers have been reprinted in edited volumes on image processing and pattern recognition. He received the best paper awards in 1987 and 1991, and received certificates for outstanding contributions in 1976, 1979, and 1992 from the Pattern Recognition Society.

Dr. Jain served as editor-in-chief of the *IEEE Transactions on Pattern Analysis and Machine Intelligence* (1991-1994), and currently serves on the editorial boards of *Pattern Recognition Journal*, *Pattern Recognition Letters*, *Journal of Mathematical Imaging*, *Journal of Applied Intelligence*, and the *IEEE Transactions on Neural Networks*. He is co-author of the book *Algorithms for Clustering Data* (Prentice-Hall, 1988). He has edited the book, *Real-Time Object Measurement and Classification* (Springer-Verlag, 1988), and co-edited the books, *Analysis and Interpretation of Range Images* (Springer-Verlag, 1989), *Neural Networks and Statistical Pattern Recognition* (North-Holland, 1991), *Markov Random Fields: Theory and Applications* (Academic Press, 1993), and *3D Object Recognition* (Elsevier, 1993).

Dr. Jain is a Fellow of the IEEE. He was co-general chair of the 11th International Conference on Pattern Recognition, The Hague (1992); general chair of the IEEE Workshop in Interpretations of 3D Scenes, Austin (1989); director of the NATO Advanced Research Workshop on Real-Time Object Measurement and Classification, Maratea (1987); and co-directed National Science Foundation-supported Workshops on "Future Research Directions in Computer Vision," Maui (1991), "Theory and Applications of Markov Random Fields," San Diego (1989), and "Range Image Understanding," East Lansing (1988).

Dr. Jain was a member of the IEEE Publications Board (1988-1990) and served as a Distinguished Visitor of the IEEE Computer Society (1988-1990). He is currently a distinguished lecturer of the IEEE Computer Society's Asia-Pacific Lectureship Program.