THEME ARTICLE

Fingerprint Matching Using Transformation Parameter Clustering

ROBERT S. GERMAIN, ANDREA CALIFANO, AND SCOTT COLVILLE
IBM Thomas J. Watson Research Center

Fingerprint matching is used in many noncriminal identification applications. Flash, a similarity-searching algorithm akin to geometric hashing, proves suitable for one-to-many matching of fingerprints on large-scale databases.

Biometrics is the science of identifying individuals by a particular physical characteristic such as voice, eye color, fingerprints, height, facial appearance, iris texture, or signature. Fingerprints are arguably the most popular biometric currently in use because of their long history in law enforcement applications. The pattern of ridges on each person's fingers uniquely characterizes that individual and contains sufficient information to distinguish that person from any other.

Fingerprint matching addresses two general classes of problems. The first involves situations for which it is necessary to verify or authenticate an individual's identity. Such one-to-one matching problems are of interest here primarily as a conceptual basis for one-to-many matching.

The second, more challenging class of problems occurs when a particular database requires a single entry for any given individual. Examples include a social services database, wherein individuals must be prevented from using multiple aliases, and identity card issuance. This identification problem necessitates a large database search of individuals to determine whether a person is already in the database.

Previous work in automatic fingerprint identification systems has concentrated on criminal justice applications. In the criminal justice arena, the cost of missing a potential match is high—if a wanted criminal is released, for example—so trained fingerprint officers are employed to inspect a large number of candidate matches.

Criminal justice fingerprinting systems retain images of all 10 fingers. Queries of such systems almost always involve a large amount of filtering, effectively reducing the size of the searched database. This filtering might involve classification based on general ridge pattern, but also includes demographic filtering based on such factors as age, race, and geographic location.

The work reported here addresses the requirements of a noncriminal identification application. The challenges for this application are to support the large throughput required for enrollment of a large population over a limited period of time and to minimize the time that a clerk must spend investigating ambiguous cases. In addition, to simplify the enrollment process and minimize storage requirements, the system must achieve acceptable performance with as few fingerprint impressions from each individual as possible. Commonly, imprints are taken only of the two index fingers.

Large-scale social service or national identity registry applications require searches of databases containing imprints from a large fraction of the total population of a state or country. An understanding of the systematic changes in the error rates of a fingerprint identification system relative to database size helps build the framework for extrapolating measurements from small benchmarking or sample databases. Criminal justice fingerprint systems are not characterized in a manner that

1070-9924/97/$10.00 © 1997 IEEE

allows extrapolation of measured performance data to large database sizes. In particular, they focus on characterization in terms of the frequency with which the correct result appears in the top-ranked position or in the top 10 positions, metrics which are useful for comparing results achieved on a particular database but which are ill-suited for making estimates of the identification error rates on larger databases. The existing American National Standards Institute (ANSI) and International Association for Identification (IAI) standard[1] for the benchmarking of fingerprint identification systems focuses on criminal justice applications and on the relative performance of competing systems.

Flash and geometric hashing

The straightforward approach to searching a large database is to scan the entire database and to compare the query against each reference model. The increased efficiencies obtained from generating index tables to speed access are well known in the database community.[2] An index can be formed from a subset of the feature points in a model instance or from the generation of multiple indices for a single model instance from subsets that redundantly include feature points. The indexing scheme allows retrieval of models that differ from the query by one or more feature points. Redundant indexing schemes in computer vision applications, the earliest example of which is geometric hashing,[3] are robust in the presence of partial occlusion.

The Flash algorithm[4] uses a higher dimensional indexing scheme than geometric hashing by adding invariant properties of the feature subset to the index. Scalar properties such as color might be appropriate in some vision applications, while in fingerprint recognition the relationship of the chosen subset of features to the local ridge pattern provides additional distinguishing power. The second stage of the Flash algorithm uses transformation parameter clustering to accumulate evidence.[5]

Object instances are represented by a collection of feature points, which might be points of maximum curvature in a vision application, minutiae in a fingerprint application, or an ASCII character in a string-matching application. When adding a model to the database, invariant information computed from each subset of feature points forms a key or index. The key labels an entry that is added to a multimap or bag,[2] a variant form of associative memory permitting more than one entry to be stored with the same key value. This entry minimally contains the identifier of the model that generated the key and information concerning the feature subset, as shown in Figure 1.

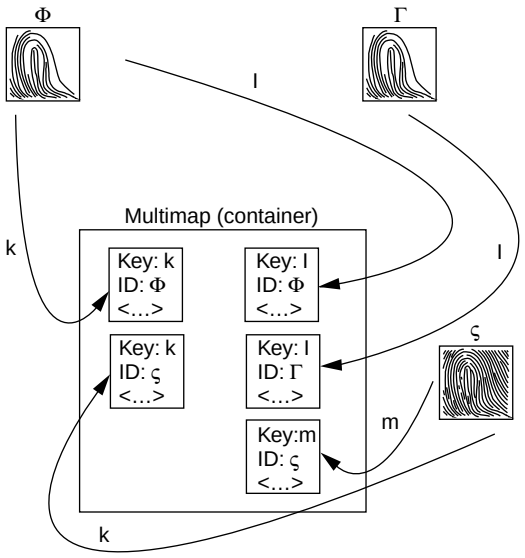

Figure 1. The extracted features from each fingerprint are used to generate keys or indices. For each key generated, an entry is added to the multimap data structure. For example, fingerprint Φ generates keys k and l, fingerprint Γ generates key l, and fingerprint ζ generates keys k and m.

When servicing a query, each key generated by the query object instance is used to retrieve any items in the multimap that are stored under the same key. Each item retrieved represents a hypothesized match between subsets of features in the query object instance and the reference model instance that created the item stored in the multimap. This hypothesized match is labeled by the reference model identifier and, possibly, by parameters characterizing the geometric transformation bringing the two subsets of features into closest correspondence.

Votes for these hypothesized matches accumulate in another associative memory structure, keyed by the model fingerprint identifier and the transformation parameters as shown in Figure 2. This structure—a map or keyed set—serves as a container that permits a single item to be stored under a given key. With the construction of each hypothesis, the program checks to see if a hypothesis with the same label already exists in the hypothesis table (map container). If the hypothesis already exists, the score of the existing entry updates appropriately. If the hypothesis does not exist, a new entry is added to

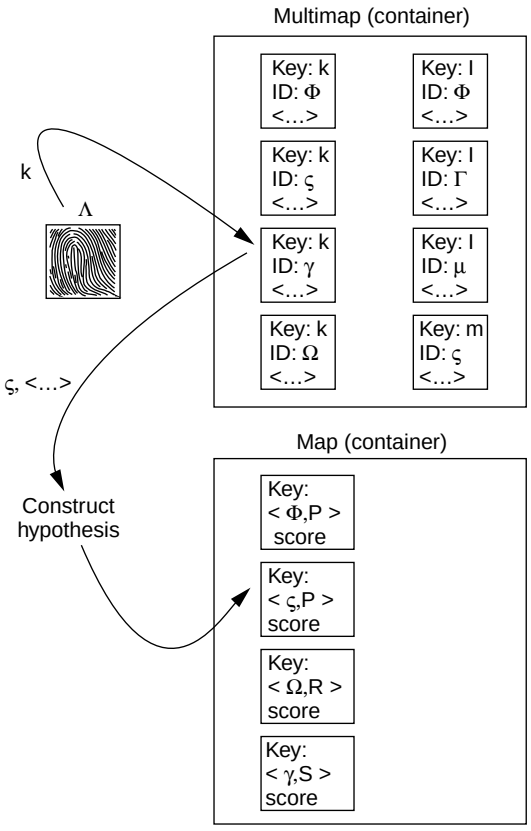

Figure 2. Retrieval.

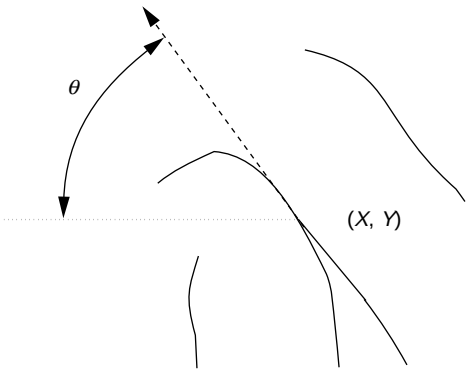

Figure 3. An example of a minutia point in a fingerprint. The *X*, *Y* coordinates provide the location of the minutia in the reference frame of the print, while θ is the angle that the ridge makes with respect to the *X*-axis of the reference frame. With the appropriate convention for choosing this direction, θ has an unambiguous value in the range of [0,2π) radians.

the hypothesis table with its score set to an initial value. Finally, a sorted list of hypotheses whose scores exceed some threshold can be used to determine whether either a match to the query object instance exists in the database or as input to another stage of matching machinery.

If a globally parameterizable transformation can be computed from each set of local feature correspondences, then when a good match exists, many of the local feature subset correspondences located during the index lookup phase will generate the same parameters for the geometric transformation and accumulate a large number of votes for that hypothesized match. Examples of globally parameterizable transformations include affine, similarity, or rigid transformations in one, two, or three dimensions. The computation of transformation parameters and the accumulation of evidence in bins or cells that span a range of values for these parameters are instances of the pose-clustering technique.[6] It is only necessary to compute transformations or relative poses for pairs of corresponding feature subsets that are mapped to the same invariant index, permitting parallel accumulation of evidence at a fine granularity.[7] Verification of consistency in the correspondence of different local feature sets is implicit in the evidence accumulation process; large numbers of consistent relative poses are only generated when the relative positions of many local feature sets are consistent in both the query and model object instances.

Application to fingerprint matching

In the fingerprint application, the class of transformations that connects different object instances is assumed to be that of two-dimensional distance preserving (rigid) transformations. A least-squares estimation methodology is used to solve the overconstrained pose estimation problem for each hypothesized local correspondence generated by the index lookup process.

Data abstraction and index generation

Both automatic and manual fingerprint recognition schemes use the feature points determined by singularities in the finger ridge pattern. Unlike the general shape recognition problem,[5] in fingerprint matching the singularities in the ridge pattern known as *minutiae* provide a natural choice for feature points. These features, which consist of points where a ridge either ends or splits into two ridges, form the basis of most fingerprint matching applications.

A triplet of numbers (X, Y, θ) represent each feature point, as shown in Figure 3. A typical "dab" impression has approximately 40 minutiae that are recognized by the feature extraction software, but the number of minutiae varies from zero to over 100 depending on the finger morphology and imaging conditions. Not all of these minutiae are reproducible from imprint to imprint, so redundancy in the combinatorial index formation process is essential.

The Flash matcher uses one additional piece of information. Part of the output of the feature extraction process is a skeletonized version of the ridge pattern on the finger. If a line is drawn between each pair of minutiae, the number of ridges crossed by this line can be counted, as shown in Figure 4. This ridge-counting procedure repeats for each pair of minutiae in the fingerprint, and the results become part of the Flash index.

The Flash algorithm uses redundant combinations of three feature points when forming indices. This gives some immunity against noise (insertions and deletions of feature points) and provides more uniquely descriptive information than is available from a single feature point. An exhaustive listing of the possible combinations of three feature points requires

$$\binom{n}{3} = \frac{n!}{(n-3)!\,3!}$$

entries, where n is the number of minutiae. To keep the number of indices generated within bounds, the algorithm restricts the "acceptable" combinations of feature points used to form an index; it only uses triplets for which the distances separating each pair of points fall into the specified range.

Even the restriction on pairwise separations does not prevent large variations in the number of indices generated by different fingerprints. To guarantee a relatively constant number of indices generated, the algorithm uses a deterministic selection process to select a sampling of those indices whose generating triangles satisfy the imposed side length constraints.

The search engine requires the generation of indices used for table lookup that are simultaneously descriptive of the objects stored in the database and invariant under the transformations to which an object might be subjected. Each component of the index is invariant under rotations and translations.

While the model used here assumes that mated fingerprint impressions might be mapped onto each other by a rigid transformation, the realities of the imaging and feature extraction process are such that some uncertainty is associated with the minutia coordinates. The implementation of the Flash algorithm requires that the index take on discrete values; hence, some binning mechanism must be used. The bin size allows appropriate tolerance for irreproducibility in minutiae positions.

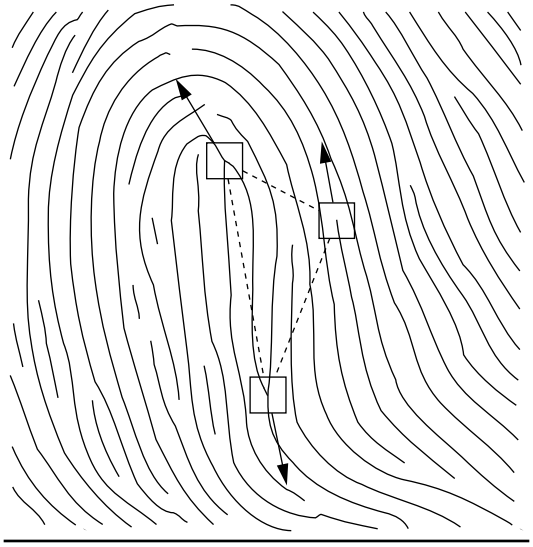

Figure 4. Triplet of minutiae on skeletonized image of a fingerprint, with the direction of ridges at minutiae (minutiae angles) shown.

A reproducible choice for the ordering of the sides is made by traversing the triangle in a consistent sense (clockwise in this implementation), as shown in Figure 5. This procedure is invariant under rotations and translations, but not under reflection. The full index consists of nine components: the length of each side, the ridge count between each pair, and the angles measured with respect to the fiducial side. S_i are the lengths of the three sides. θ_i are the minutiae angles encoded in a transformation-invariant fashion. RC_i are the number of ridges crossed by a line connecting a pair of minutiae.

Accumulating evidence

The index generated by the Flash framework serves as the key to identifying triangles that "resemble" one another. During the storage phase, each index generated by a fingerprint causes the storage of a data object containing the identity of the fingerprint and information concerning the triplet of feature points that generated the index.

During the query phase, each index generated by the query fingerprint is used to retrieve all model

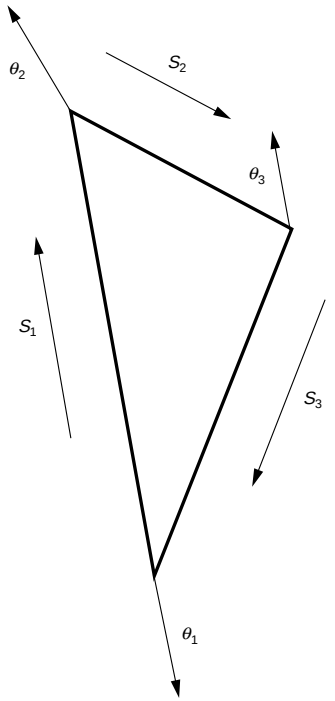

Figure 5. Geometry of an ordered triangle without the background of a fingerprint. The sides (with lengths S_i) are ordered so that the largest side appears first, and successive sides proceed in a defined orientation—for example, clockwise. The order of the ridge counts (the number of ridges crossed by a line connecting a pair of minutiae), the minutiae, and the minutia angles θ_i is the same as that of the sides, with the first being in a well-defined orientation (for example, the most counterclockwise point) with respect to the first side. Thus the ordering of the sides enables unambiguous expression of the other essential quantities.

objects stored in the table that are labeled with the same index. Each of these retrieved model objects represents a hypothesized correspondence between three points in the query print and three in the model print. Given this correspondence, the coordinate transformation that best maps the query triplet onto the model triplet is computed. The algorithm that computes the coordinate transformation does so in a way that minimizes the sum of the squared distances between the transformed query points and their corresponding model points. The essentials of the retrieval procedure are outlined in Figure 6.

The computed transformation parameters, X and Y translation and rotation θ, are binned and, along with the reference fingerprint ID, form a key that indexes the map (keyed set) used for evidence accumulation, as shown in Figure 2.

If a large number of feature points can be brought into correspondence by a rigid transformation of the coordinate system, all of the indices generated by the combinations of three feature points belonging to this set generate the same coordinate transformation parameters. Hence a large number of votes for a correct match are tabulated. There might be a number of random correspondences between triplets of points in the query print and some arbitrary reference print, but the likelihood of a number of consistent transformation parameters being generated by such random correspondences is quite small.

After the system generates all of the indices from the query fingerprint and computes all of the relevant hypotheses, it sorts by score the entries in the hypothesis table exceeding some threshold. This ranked list of scores can be provided in response to the original query or used in other decision-making machinery that might combine the results from queries using imprints taken from additional fingers.

Accuracy issues

Consider the problem of determining whether or not two fingerprints were made by the same finger (verification). This problem amounts to assigning the pair to either the mated or nonmated pair populations while making the smallest number of mistakes in a large number of trials. This problem has a long history[8] in statistical decision making. Where one of two mutually exclusive hypotheses, H_0 and H_1, must be selected, two classes of errors can be made. Suppose that H_0 is the hypothesis that the prints belong to the nonmated population and that H_1 is the hypothesis that the prints belong to the mated population. Four scenarios are possible:

- H_0 is true, and test says H_0 is true
- H_0 is false, but test says H_0 is true
- H_1 is true, and test says H_1 is true
- H_1 is false, but test says H_1 is true

The test breaks down in two of the four scenarios. Two distinct types of errors can be made: a false negative or miss, in which a mated pair is incorrectly assigned to the nonmated population; and false positive or false alarm, in which a nonmated pair is incorrectly assigned to the mated population.

The number of matching triangles that generate a consistent rigid transformation between two prints serves as the basis for assigning pairs of fingerprints to the mated or nonmated pair population. Histograms of the scores achieved by the matcher on the two test populations can be used as estimates for the conditional probability densities of the score, $f_{mated}(x)$ and $f_{nonmated}(x)$. It is natural to use a threshold x_{th} to assign a pair of imprints to one of the two possible populations. Any pair whose score exceeds x_{th} is assigned to the mated pair population, and other pairs are assigned to the nonmated pair population. There is a trade-off between the false-positive error rate (*FPR*) and the false-negative error rate (*FNR*). The *FNR* can be reduced to an arbitrarily small value by decreasing x_{th} sufficiently, but a large number of false alarms results in a corresponding increase in the *FPR*.

With this decision criterion, it is straightforward to determine the two error rates from the

conditional probability densities computed from the test populations. The error rate for incorrectly assigning a mated pair to the nonmated population (*FPR*) is given by this distribution function:

$$FPR = F_{mated}(x_{th}) = \int_0^{x_{th}} f_{mated}(t)dt .$$

Similarly, the error rate for incorrectly assigning a nonmated pair to the mated pair population (*FNR*) is given by the following function of the conditional probability distribution function:

$$FNR = 1 - F_{nonmated}(x_{th}) = 1 - \int_0^{x_{th}} f_{nonmated}(t)dt .$$

Insofar as the mated and nonmated pair test populations form representative samples of real populations, the estimates can be used to extrapolate behavior in real populations. The measured accuracy of a matcher is a strong function of the database from which estimates of the error rates are derived. Indeed, the variation of the *FNR* with threshold depends on the care with which fingerprint images were acquired. These estimates of the error rates are independent of the size of the test database used, although the uncertainties in the estimates depend on the sizes of the sample pair populations.

Consider a one-to-many identification query, which might be viewed as a series of one-to-one verifications executed against every print in the database. The candidate list of hypothesized matches is formed by taking all prints from the reference database whose verification-matching scores with the query print exceed some fixed threshold. Assuming the presence of at most one mate to the query, the *FPR* and the *FNR* for an identification search against a database of *N* individuals are as follows:

$$FNR(N) = FNR(1)$$
$$FPR(N) = 1 - (1 - FPR(1))^N \approx N \times FPR(1) \text{ for } FPR(1) << 1/N$$

The *FPR* increases drastically with database size because each additional entry in the database provides another opportunity to randomly achieve a high score. A matcher operating with a false positive verification rate of 1 percent might be satisfactory in a verification application, but in even a small-scale identification application the error rate becomes unacceptable. For example, when used on a 10-person database, this matcher generates false matches at a rate of $1 - 0.99^{10}$ or 9.5 percent. On a 100-person database, this matcher's *FPR* becomes $1 - 0.99^{100}$ or 63 percent. Figure 7 shows the extrapolated *FPR* versus population size for a variety of one-to-one error rates. To keep the *FPR* within reasonable bounds for large population sizes, a matcher must operate with an *FPR* in the range of $10^{-9} - 10^{-6}$. Model-independent estimates of false-positive error rates in this range require a correspondingly large sample population of mismatched pairs of prints. Because of the trade-off between *FPR* and *FNR*, the need to operate at very small values of the false-positive rate in identification applications might lead to unacceptable miss rates when using a single finger. The system miss rate can be reduced dramatically by executing searches using two different query fingers and considering a match on either finger to be a hit while causing a modest increase in the *FPR*.

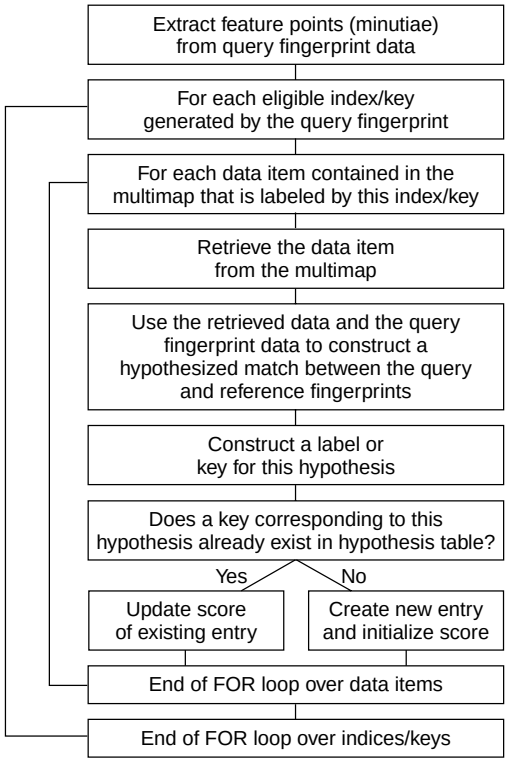

Figure 6. A broad outline of the main portion of the query phase. Hypothesis key construction involves estimating the rigid transformation parameters, rotation and translation, using the hypothesized correspondence between triplets of minutiae in the query and the model fingerprint.

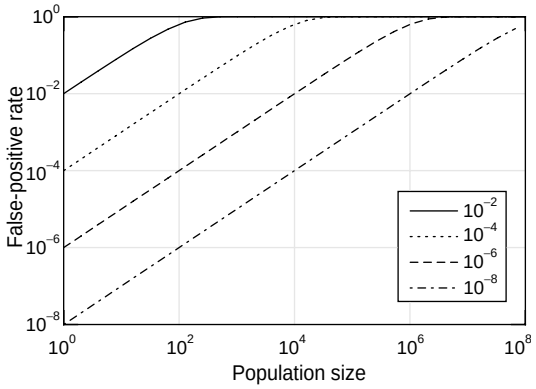

Figure 7. Extrapolated false-positive error rates plotted versus population size for a series of verification false-positive error rates.

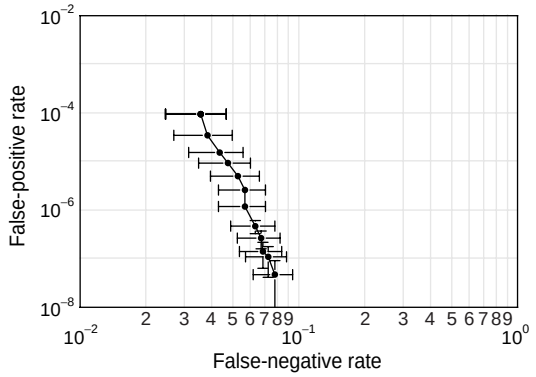

Figure 8. False positive error rate (*FPR*) versus false negative error rate (*FNR*) for a variety of verification decision thresholds. The error bars represent the 90-percent confidence intervals for the estimates of the corresponding error rates obtained from this set of experiments. This presentation is similar to the receiver operating curve[7] used to characterize the ability of a statistical test to distinguish between two alternative hypotheses.

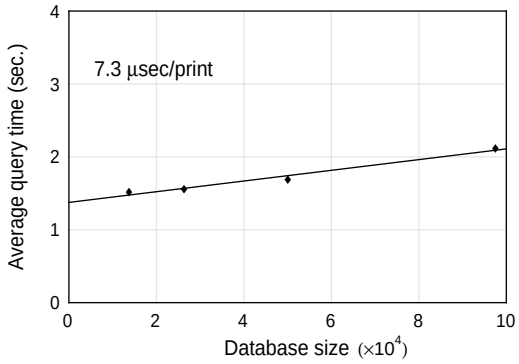

Figure 9. Average query times for databases of varous sizes: 13,726, 26,317, 50,047, and 97,492 entries. Each point represents an average of 657 queries, and the line is a least-squares fit to the data. No front-end filtering was used in these tests; each data point represents a similarity search of the entire database. The hardware and software configuration used for each test was the same. The nonzero intercept is a consequence of the requirement for doing index lookups. In this series of runs, 32 disks were used to spread the I/O burden.

Results

Two important aspects of the matching engine are accuracy and speed (see Figures 8 and 9).

To characterize system accuracy, we constructed a reference database of model prints from 97,492 inked dab images acquired in 1995. These were processed by the feature extraction code developed by the Exploratory Computer Vision Group at the IBM Thomas J. Watson Research Center. We executed 657 queries against this database. The query set of prints was a subset of the 97,492 models. Conceptually, we made $657 \times 97{,}492$ comparisons of pairs. These pairs can be divided into three groups:

- identical fingerprints (657 pairs)
- different impressions of the same finger (768 pairs)
- impressions of different fingers (64,050,819 pairs)

The pairs in the first group are excluded from the analysis of results because they represent an experimental artifact. There are 768 pairs in the second group because some query prints had two or even three mates in the reference database.

The distributed Flash algorithm as implemented for fingerprint matching requires a few thousand I/O operations to look up the indices generated by each query. These I/O operations can be spread over a large number of disks to keep down the elapsed time. With 32 disks distributed over an 8-node IBM SP2 system, the incremental addition to the average query time caused by an additional print in the database is approximately 7 microseconds, as shown in Figure 9. Thus, the system configuration used for these trials could search a database of 10 million prints in approximately 70 seconds. Additional experiments indicate that the load balancing of

the I/O is such that disk parallelism can be used effectively to reduce the I/O contribution to the query time. Identification searches of very large databases of fingerprints without prefiltering are thus possible through indexing subsets of features on the fingerprints.

The work described here was completed in mid-1996. Research and development into refinements of the technology, including tests on larger databases, is continuing in collaboration with IBM Hursley (UK) Laboratories. No inferences concerning the current state of technology resulting from continued research and development for commercial application should be drawn from this article. ♦

Acknowledgments

We would like to acknowledge the contributions to this work of our collaborators in the Exploratory Computer Vision Group, S. Pankanti, N. Ratha, M. Yao, and R. Bolle.

References

1. ANSI/IAI, "Automated Fingerprint Identification Systems—Benchmark Tests of Relative Performance,"*Tech. Report ANSI/IAI 1-1988,* Am. Nat'l Standards Inst., New York, 1988.
2. J.D. Ullman and J. Widom, *A First Course in Database Systems,* Prentice Hall, Upper Saddle River, N.J., 1997.
3. Y. Lamdan and H.J. Wolfson, "Geometric Hashing: A General and Efficient Model-Based Recognition Scheme," *Proc. Second Int'l Conf. Computer Vision,* 1988, pp. 238–249.
4. A. Califano and R. Mohan, *Generalized Shape Autocorrelation for Shape Acquisition and Recognition,* US patent 5,351,310, Sept. 1994.
5. A. Califano and R. Mohan, "Multidimensional Indexing for Recognizing Visual Shapes," *IEEE Trans. Pattern Analysis and Machine Intelligence*, Vol. 16, No. 4, Apr. 1994, pp. 373–392.
6. G. Stockman, "Object Recognition and Localization via Pose Clustering," *Computer Vision, Graphics, & Image Processing,* Vol. 40, 1987, pp. 361–387.
7. I. Rigoutsos and A. Califano, "Searching in Parallel for Similar Strings (Biological Sequences)," *IEEE Computational Science & Eng.,* Vol. 1, No. 2, Summer 1994, pp. 60–75.
8. E.L. Lehmann, *Testing Statistical Hypotheses,* Second ed., Springer-Verlag, New York, 1986.

Robert S. Germain manages the Scalable Similarity Searching Group at the IBM T.J. Watson Research Center. His current research interests include the parallelization of pattern matching and pattern discovery and their applications in biometrics. Germain received his AB in physics from Princeton University and his MS and PhD in physics from Cornell University. After receiving his doctorate in 1989, he became a research staff member at the Watson Research Center. He is a member of the IEEE Computer Society, the IEEE, and the American Physical Society.

Andrea Califano received the Laurea in physics from the University of Florence, Italy, in 1985. In 1986, he was a visiting scientist at the Information Mechanics Group at the Massachusetts Institute of Technology. From the end of 1986 to 1991, he was a research staff member in the Exploratory Computer Vision Group at the IBM T.J. Watson Research Center, where he was manager of the Computer Biology and Pattern Matching Group and program director of the IBM Research Center for Computational Biology. A senior member of the IEEE since 1993, Califano is currently interested in computer vision and pattern matching and their applications to molecular biology and genetics.

Scott Colville is pursuing an MS in computer science at the University of Wisconsin-Madison. His current research involves memory caches; however, his interests also include computer architecture, networking, systems, and databases—for example, data mining, pattern matching, and real-time queries. Scott received his BS in computer science from Carnegie Mellon University and then worked in the Computer Biology and Pattern Matching Group at the IBM T.J. Watson Research Center from 1995 to 1996.

Readers may reach the authors in care of Germain, IBM T.J. Watson Reasearch Center, 30 Saw Mill River Road, Hawthorne, NY 10532; e-mail, germain@watson.ibm.com.