U.S. Department of Commerce

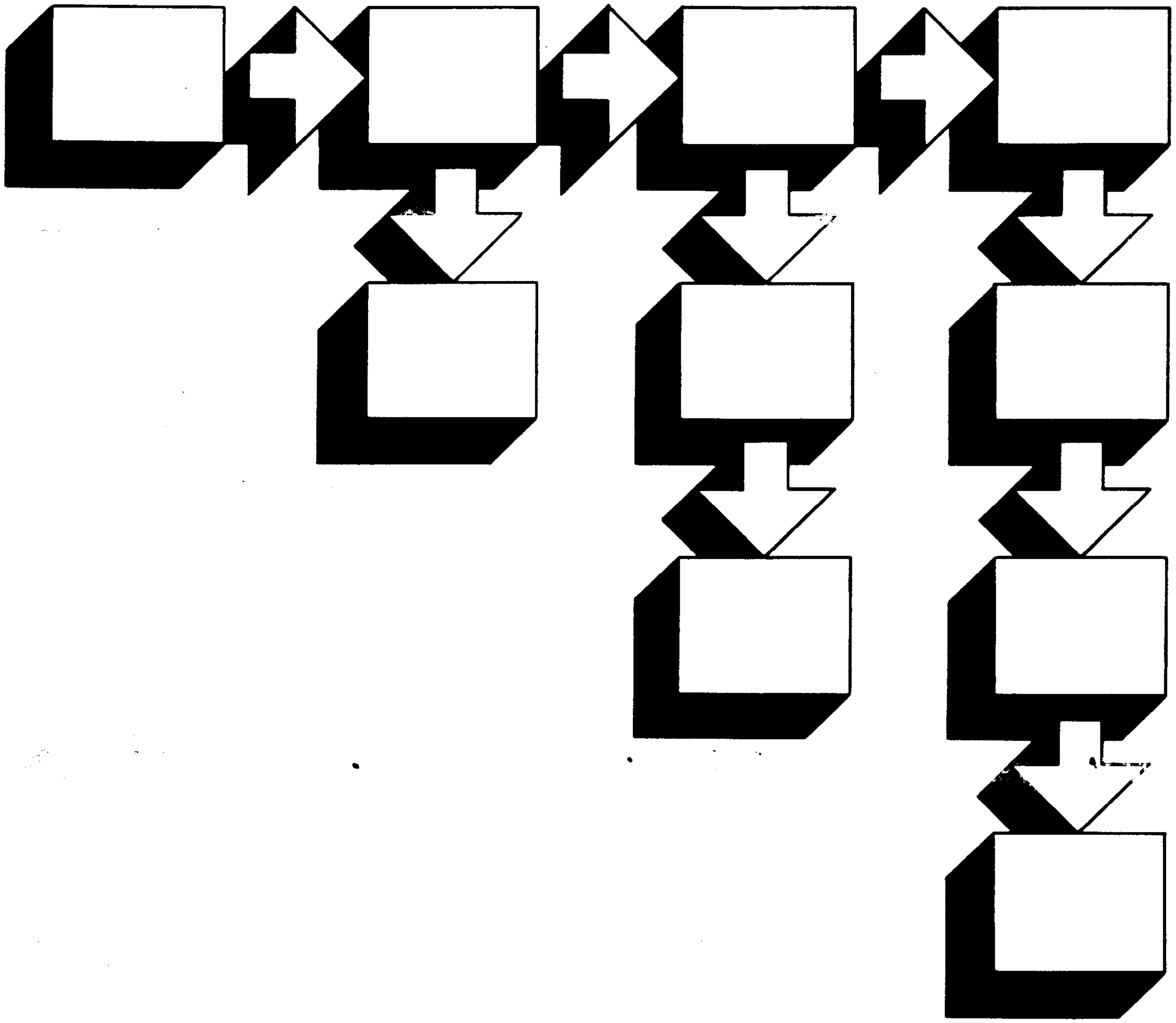
National Bureau of Standards

Computer Science and Technology

NBS Special Publication 500-124

A Topological Approach to the Matching of Single Fingerprints: Development of Algorithms for Use on Rolled Impressions

Malcolm K. Sparrow and Penelope J. Sparrow



Computer Science
and Technology

NBS Special Publication 500-124

A Topological Approach to the Matching of Single Fingerprints: Development of Algorithms for Use on Rolled Impressions

Malcolm K. Sparrow
Penelope J. Sparrow

Center for Computer Systems Engineering
Institute for Computer Sciences and Technology
National Bureau of Standards
Gaithersburg, Maryland 20899

U.S. DEPARTMENT OF COMMERCE
Malcolm Baldrige, Secretary

National Bureau of Standards
Ernest Ambler, Director

Issued May 1985

Reports on Computer Science and Technology

The National Bureau of Standards has a special responsibility within the Federal Government for computer science and technology activities. The programs of the NBS Institute for Computer Sciences and Technology are designed to provide ADP standards, guidelines, and technical advisory services to improve the effectiveness of computer utilization in the Federal sector, and to perform appropriate research and development efforts as foundation for such activities and programs. This publication series will report these NBS efforts to the Federal computer community as well as to interested specialists in the academic and private sectors. Those wishing to receive notices of publications in this series should complete and return the form at the end of this publication.

Library of Congress Catalog Card Number: 85-600541

National Bureau of Standards Special Publication 500-124
Natl. Bur. Stand. (U.S.), Spec. Publ. 500-124, 75 pages (May 1985)
CODEN: XNBSAV

U.S. GOVERNMENT PRINTING OFFICE
WASHINGTON: 1985

For sale by the Superintendent of Documents, U.S. Government Printing Office, Washington, DC 20402

ACKNOWLEDGEMENTS

The authors are indebted to the Institute for Computer Sciences and Technology, National Bureau of Standards, for its hospitality and for the use of its excellent facilities. More particularly they are most grateful to R. T. Moore (ICST, NBS) for sharing the benefits of his vast experience in this field; to Robert M. McCabe (ICST, NBS) for his invaluable advice on computational problems and his expert assistance in negotiating the foibles of Fortran; to Robert Stock (FBI, Fingerprint Automation and Research) and his staff for their advice, their cooperation in providing the many hundreds of required fingerprints, and for allowing access to and use of their automatic scanning facilities.

TABLE OF CONTENTS

Page

LIST OF FIGURES

A TOPOLOGICAL APPROACH TO THE MATCHING OF SINGLE FINGERPRINTS: DEVELOPMENT OF ALGORITHMS FOR USE ON ROLLED IMPRESSIONS

Malcolm K. Sparrow and Penelope J. Sparrow

ABSTRACT

The motivation for seeking topological descriptions of single fingerprints is provided by the elasticity of the human skin; successive rolled impressions from the same finger will invariably have suffered a degree of relative distortion (translation, rotation and stretching). Topology based systems should be free from the detrimental effects of plastic distortion.

Systems are described for the extraction of simple topological codes from rolled impressions of the pattern types 'loops,' 'whorls' and 'arches.' The generated codes take the form of vectors or simple digital arrays.

The nature and frequency of changes that may occur in such codes is investigated and fingerprint comparison algorithms, based on these topological codes, are developed. The objective of such algorithms is to draw a score derived from the degree of 'nearness' of the topological codes in such a manner that it intelligently reflects similarity or dissimilarity in the two prints under comparison.

Detailed analysis of the performance of such algorithms is given, making extensive use of the results of investigation into the 'match' and 'mismatch' score distributions produced by each one. A final test is described in which the most effective 'topology-based' algorithm was directly tested against one of the best existing 'spatial' algorithms. Topology-based coding, with the inclusion of a crude 'distance measures,' is found to be an extremely accurate and efficient basis for the comparison of rolled impressions.

Key words: Automated comparison; distortion independence; fingerprints; minutiae; ridge-tracing; topology.

INTRODUCTION

The motivation for seeking topological descriptions of single fingerprints is provided by the elastic nature of the human skin. That elasticity causes substantial variation in the spatial descriptions of successive impressions of the same finger. Consequently, comparison algorithms based on spatial information (i.e., information which principally records distances and directions) have to fall into one of two broad categories: either they will be unreliable or they will be sufficiently sophisticated to recognize, and compensate for, the innumerable types of twisting, stretching, tilting and translation caused by changes in the physical circumstances under which the impressions are formed. Such sophistication will produce comparison algorithms that are highly complex statistically and which will invariably be expensive to implement.

The theoretical appeal of coding fingerprints topologically (in a way that omits reference to distances and directions) lies in the expectation that such topological information will be relatively free from the effects of plastic distortion. Detailed explanation of this motivation has already been given in an earlier paper.[1]

This phase of the experimental work seeks to establish whether or not a topological coding system can be found, together with a suitable matching algorithm, that provides a sound basis for reliable and efficient single-print comparison. If such a scheme can be found, then we will certainly want to know under what circumstances, if any, it will perform better than coding/comparison techniques based on the more traditional (spatial) approaches.

It is exactly these questions that we will hope to answer in the following chapters.

1. BACKGROUND, AIMS AND ANTICIPATED PROBLEMS

1.1 Aims of the Work on Rolled Impressions

There are already a variety of automated systems available that appear to adequately perform the function of comparing rolled impressions. This is usually in the context of the comparison of record cards each showing clear rolled impressions of all ten fingers of one individual. (These will have been taken, for example, when a suspect was arrested or, in the case of some civilian applications, when an individual applied for a job of a particular kind.) Matching these cards by comparison of some or all of those ten impressions forms an important part of the 'identification of persons' problem; that problem faces a wide range of law-enforcement agencies. The fingerprint comparison part of the procedure becomes of prime importance when the stated name and date of birth cannot be relied upon.

For example the FBI (Federal Bureau of Investigation) Automated Fingerprint Division (based in Washington, D.C.) handles in the order of 20,000 10-print card enquiries per day. The database to be searched in each case comprises some 23 million cards. With such a colossal workload, it is necessary to use all the available demographic information (such as the physical description of the individual, his stated name, address and date of birth, the type of crime and the geographical area of its commission) to reduce the field of search quite drastically before any fingerprint comparisons are performed. The field of search is reduced, in fact, from the 23 million possibilities, to a maximum of 259 most likely candidates. Only then does computerized fingerprint comparison take place.

When a 'match' is indicated (by a high score from the comparison algorithms), the appropriate card is extracted from the collection and checked against the 'search' (enquiry) card manually by a fingerprint expert. If a 'match' is not suggested by the comparison scores, there is absolutely no question of a manual search being conducted to ensure that the individual represented by the search card is 'not known' (i.e. that his prints do not appear in the collection).

The FBI system is almost certainly the largest automated collection in the world. Smaller (more local) agencies have a variety of similar systems available to them, and several are in use.

With this situation in mind, one could surmise that research interest should now confine itself to the problems of 'latent mark' identification (that is the matching of marks left at the scenes of crime) against a file collection of rolled images. (Latent marks tend to be fragmentary and of relatively poor quality. They have to be 'developed' by chemical or other means and then are normally photographed to facilitate comparison.)

There are several good reasons why research into topological coding must start with its application to rolled impressions:

(a) Topological coding may well provide neat and concise digital codes that would provide a more economical, and perhaps more reliable, basis for ten-print systems. If an appropriate degree of simplicity and speed can be achieved then such methods could make feasible the use of inexpensive microcomputers for the storage and searching of small (local) collections.

(b) Contemplation of massive collections (e.g., for international missing person identification) only becomes possible with extremely fast comparison methods (witness the operational constraints imposed on searching by the FBI's workload). The advent of parallel processing systems should suggest that we look for comparison methods which consist largely, or entirely, of sequences of array operations. Algorithms so composed, when run on parallel processing facilities, should be capable of achieving phenomenal speeds. The type of comparison algorithms explored here do consist almost entirely of sequences of array operations--indeed they have all been designed with the capabilities of array processors in mind.

(c) A proper knowledge and understanding of the behaviour of topological codes under the ordinary plastic distortions can best be gained in experiments free from any other difficulties or complications. Such investigation is therefore, in a sense, preparatory to any later application of topological coding to latent mark identification.

(d) Other commercial applications (security access devices and personal authorization verification) may well benefit from a quick and effective single-print comparison technique.

1.2 Selection of Raw Data Rather Than Enhanced Images

The process of automatic scanning of fingerprints and automatic extraction of ridge detail therefrom necessarily involves an image enhancement step. The original grey-scale image (in matrix form from the scanners) is ultimately converted to a binary picture. This involves some 'smoothing' operations using 'ridge-valley' filters, and some steps to compensate for ink-density variations. These methods, and their continuing development, are not the subject of this paper even though the ideas expressed here cannot lead to any operational systems without the use and further sophistication of digital image-interpretation techniques.

The availability of enhanced images, however, poses an early question for this research: should experiments be based on images read and interpreted by machine (i.e., enhanced prints) or should raw fingerprints be used? Despite the obvious appeal of working from clear binary images, raw fingerprints were selected for this reason: automatic enhancement algorithms have not been developed with topological coding in mind. The systems in use by the FBI and by the Home Office research team (London) do not discriminate between ridge-endings and bifurcations. They simply identify the presence of a ridge-flow irrregularity and record its coordinates (after application of various tests to make sure it really is a genuine characteristic). The enhancement stages of the algorithm will most probably have a degree of bias towards some types of topological structure in its interpretation. As that degree of bias is both unknown and undocumented it was deemed unwise to incorporate it into experimental databases at source.

Although election for raw prints made for very slow and tedious data collection (direct from projected images of fingerprint cards), that penalty was mitigated somewhat by the value of much good practice in fingerprint interpretation. That experience was to prove invaluable, later, when attention was turned to latent marks.

The skill of the human brain in pattern recognition as a noise elimination filter during manual encoding also goes to set a standard by which automatic interpretative algorithms can be measured hereafter. The extent to which these experimental results could be reproduced when using machine-gathered data would be a significant test of the data collection process's ability to make the correct topological decisions.

1.3 Selection of Ulnar Loops for Initial Experiments

Of all the various single-print pattern types the category of 'Loops' is by far the largest. It accounts for roughly 64% of all fingerprints. Next most common are the 'Whorls' (30%) and then the 'Arches' (5%). Approximately 1% of prints have some other, more complex, pattern type--being known variously as 'accidentals' or 'composites.'

The 'Loops' are divided into 'radial loops' and 'ulnar loops' depending on the direction of the ridge flow from the base of the loop. Ulnar loops account for the vast majority of loops and are certainly the most common pattern class. (The ulnar loops have the delta on the thumb-side of the finger.)

For this reason ulnar loops were selected as the basis for initial experiments and the developed techniques were applied to both radial loops, whorls and arches at a later stage.

1.4 Selection of Line Based System

The stated aims of the work on rolled impressions (paragraph 1.1) make it plain that a quick, easy coding method is sought. It should provide sufficient information to identify each single print uniquely, and should be easily reproducible. The coding method selected was the 'ordering of graphical information by lines' as described in an earlier paper.[1] This is a simple process which leads to formulation of an ordered digital sequence (vector).

(a) Rules are established, dependent on the pattern type, for the superposition of a line on each print.

(b) The placing of lines forms an ordered set of intersection points (where the line crosses a ridge), each one located on one of the ridges of the print.

(c) Each point of intersection gives two 'directions' for topological exploration of that ridge: imagining oneself (just for a moment) to be a tiny insect capable of 'walking along a ridge'--then one could walk each ridge in each of two directions from the point of intersection. We stipulate that the walking (or exploration) will cease as soon as one of a number of specific 'events' is found. These events could be ridge-flow irregularities (characteristics); they could be the coming upon scarred tissue where the ridge flow pattern has been completely destroyed; they could be the 'walking off' the edge of the visible print.

(d) Assignment of digital codes to the different possible ridge-exploration 'events' leads to formation of a pair of digits for each point of intersection. Writing them down in order generates a digital vector of length equal to twice the number of points of intersection.

In theory it would be desirable for the rules governing the line placement to be entirely independent of spatial considerations so that the points of intersection used to generate the vector were themselves free from the effects of spatial distortion.

In practice it is much quicker and simpler to allow some spatial concepts to be used in placing the lines, and it will be seen that the actual position and orientation of the lines (relative to the print) is not critical provided it runs roughly orthogonal to the ridge flow.

The exact orientation of the line would be far more important if the line's direction was close to that of the ridge flow. The effect of small changes in relative orientation of line and print would then be to shift the points of intersection considerable distances, and perhaps move some of them to the opposite side of some characteristics which were close to the line. Severe corruption of the generated vectors would then occur.

A line placement rule that satisfies the requirements fairly well for loops is this:

(a) By looking at the whole available print, and with particular reference to the first flexion crease and the directions of ridges which run close to it, estimate a 'horizontal' orientation for a straight line. ('Horizontal' means parallel to the apparent direction of the flexion crease.)

(b) Place a 'horizontal' line through the loop core-center, using the conventional rules for precise location of the core-point.[2]

Figure 1 shows a typical ulnar loop pattern with horizontal line superimposed according to these rules.

These line placement rules are by no means the only ones possible for loops. There are innumerable possibilities; some centered on the core, some on the delta, and some independent of both. Experiments using the selected line placement are, however, quite sufficient to answer most of the pertinent questions as to the behaviour of topological codes under spatial distortion.

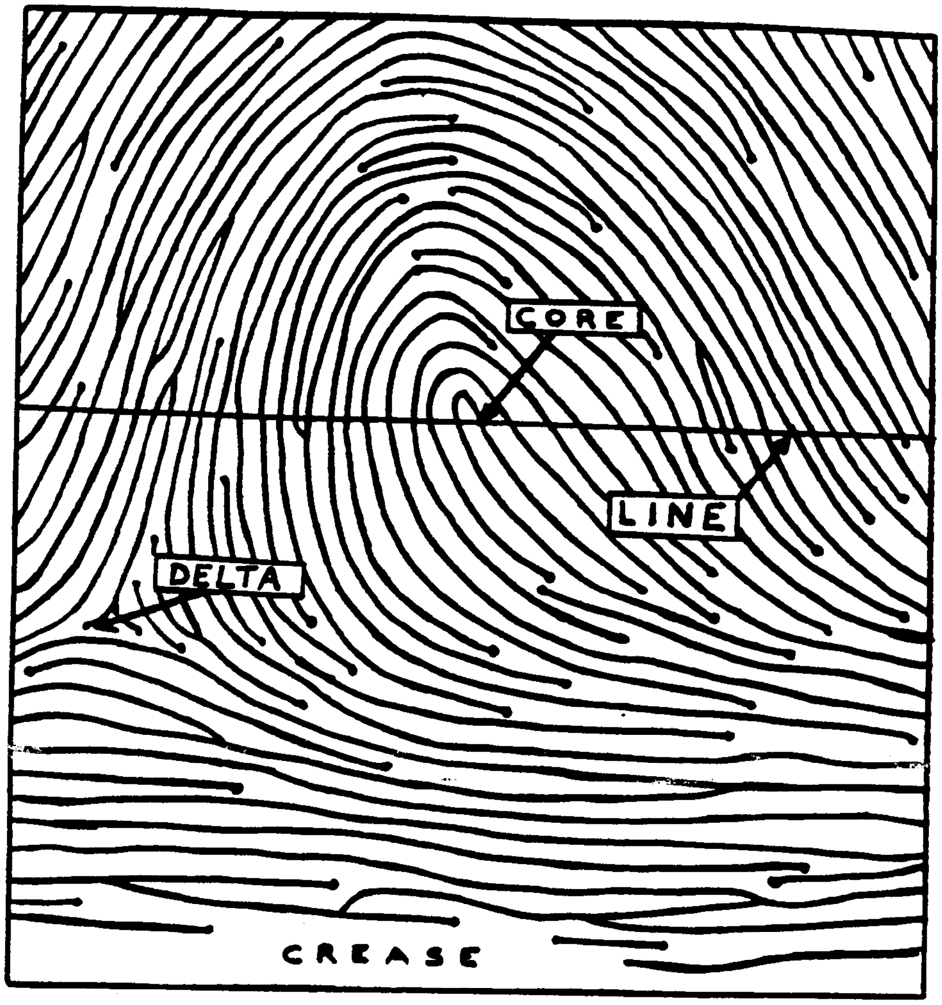


Figure 1. A complete tracing of an ulnar loop pattern, with horizontal line through core center superimposed.

1.5 Selection of Digital Codes

Figure 2 shows the digital codes selected to correspond to possible ridge-exploration 'events.' In each case the ridge being explored is marked with an arrow to show the direction of the exploration. In excess of 500 prints have been coded using these codes, and they have been found to cover all eventualities.

Code 5 (where the exploration returns to its starting point without having encountered any other 'event') was not encountered in any patterns other than whorls and then only very rarely.

The digital codes take the form of hexadecimal integers, and are always processed as such. Storage space required for each one is therefore

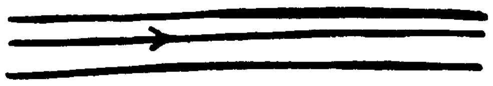
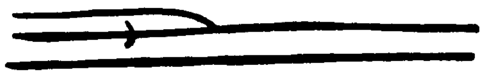
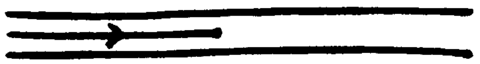
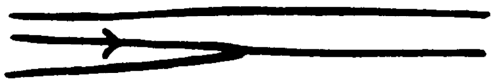
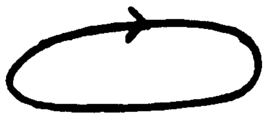
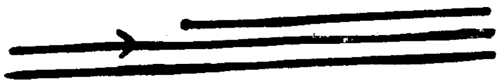
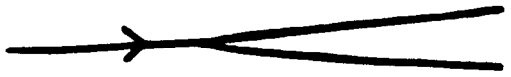
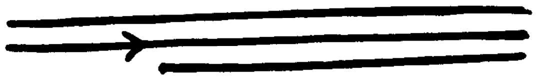
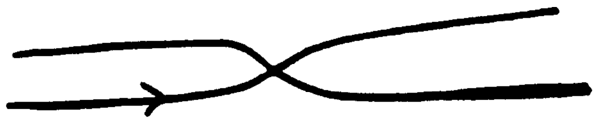
Code	Description	Diagram
0	Ridge runs out of sight without any event occurring.	
2	Ridge meets a bifurcation with fork appearing from left.	
3	Ridge ends.	
4	Ridge meets a bifurcation with fork appearing from right.	
5	Ridge returns to starting-point without any event occurring.	
6	Ridge meets a new ridge starting on the left.	
7	Ridge bifurcates.	
8	Ridge meets a new ridge starting on the right.	
A	Ridge encounters scarred tissue.	
B	Ridge encounters blurred print.	
C	Ridge encounters a compound (e.g. a crossover).	

Figure 2. Table of ridge-exploration "events" and their corresponding digital codes.

only 4 bits, making it possible to compress one pair of digits into one byte. Not all 16 hex-digits are used; 1, 9, D and E being 'spare.' 'F' is used for padding the vectors up to a certain length for storage in a standardized data format.

Codes 6 and 8 record events that do not actually occur on the ridge being explored. They record the start of a new ridge either on the immediate left or the immediate right of it. The main reason for their inclusion in the scheme is that they record the presence of ridge-endings which would otherwise be ignored by the coding process. (This is because the ridge-ending belongs to a ridge that does not have a point of intersection with the generating line.)

The allocation of particular digits to particular events is not quite arbitrary. The tendency of inking and pressure differences between successive impressions of a print to cause topological change is well known. Bifurcations will mutate to ridge-endings, and vice-versa. This occurs when short connecting ridge segments (e.g., the segment AB in Figure 3) either appear or disappear. In anticipation of this phenomenon the digital codes are selected in order that some sense of 'closeness' is carried over to them. The extent of that 'closeness' is only that event 3 is liable to change to or from either of events 2 or 4; likewise event 7 is liable to change to or from events 6 or 8.

The frequencies of these topological variations, and their effects on digital code vectors, are among the objects of this study.

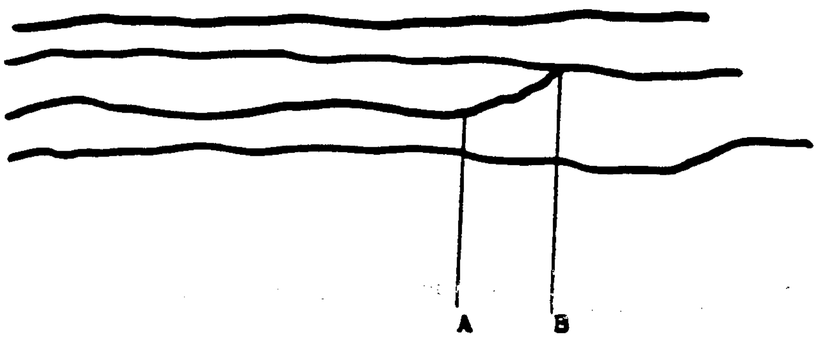


Figure 3. Ridge-ending/bifurcation mutation.

1.6 Method/Apparatus for Tracing and Coding Prints

The original data took the form of inked impressions on standard FBI ten-print cards. These were positioned in the projection plane of the 'Graphic-pen' (a device built at NBS for semi-automated data entry from a projected image; see ref. 3). This projects an enlarged image (10x enlargement) of a single print onto a horizontal screen. The available window size on the screen is 7.1" wide and 7.8" high. The cores of loops (and, later, the centers of whorls) were located at a fixed reference point on the screen 3" from the top of the screen and equidistant from the left and right edges. Prints were positioned 'upright' by reference to the flexion crease--no regard whatever being paid to the orientation of the print within the relevant printed "box" on the fingerprint record card.

Prints were positioned once and once only--so the portion of each print viewed was a rectangle measuring 0.71" by 0.78". Anything outside this rectangle was regarded as 'out of sight' and ignored by the coding process.

The projected image was traced manually onto tracing paper--the tracing of each ridge being continued only as far as was necessary to establish which of the possible 'events' was encountered FIRST in the exploration of that ridge. The traced image, therefore, gives a clear indication of just how much of the print pattern (and which characteristics) are accessed by a particular coding process.

Figure 4 shows the tracing of an ulnar loop generated by exploration from a horizontal line through the core. Points of intersection are shown numbered outwards from the core, and characteristics accessed are highlighted with a small 'blob.'

With print positioning as described, a horizontal line through the core rarely intersected more than 20 different ridges on either side of the core. Consequently a standard length for digital vectors was set at 82 digits. That is 41 pairs--of which 20 pairs represent up to 20 ridges on the left hand side of the core, one pair represents the ridge on which the core itself is located, and the other twenty pairs represent up to 20 ridges intersected on the right of the core. Whenever less than twenty ridges were intersected on the left or the right side of the core (which was usually the case) the 82 digit code was padded with 'F's, as mentioned above, to bring it up to the standard length. The padding was done at the extreme ends of the vector in such a way that the digit pair representing the core-ridge remained in the central position (i.e., the 21st digit pair).

The convention was established that the digit representing exploration along a ridge 'upwards' from the line was to be written first (of the pair), and the digit representing exploration 'downwards' along the same ridge would be written second. Adhering to that convention, the 82 digit vector generated from the tracing referred to above (Figure 4) is shown in Figure 5. To facilitate interpretation the intersection

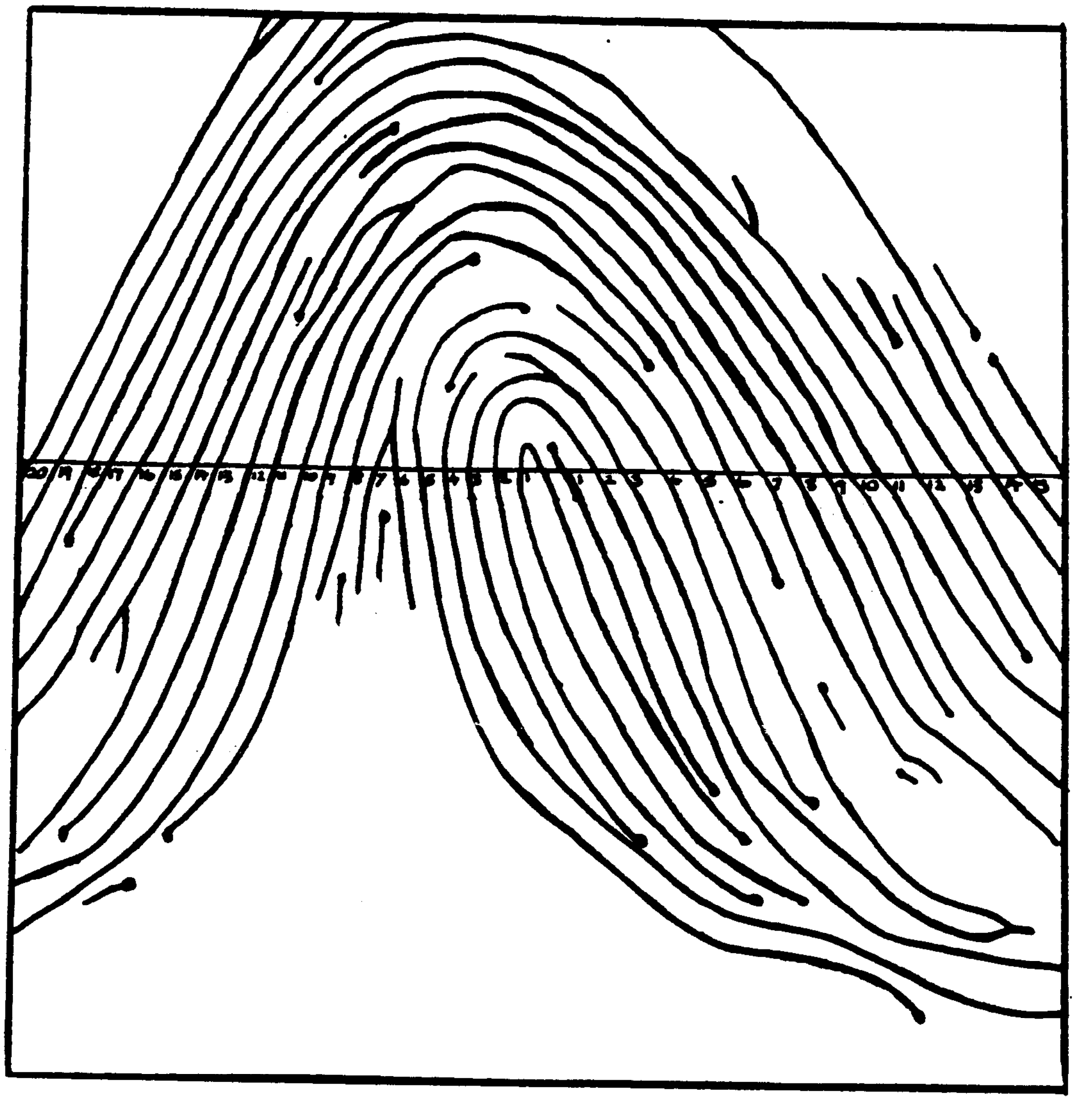


Figure 4. Ulnar loop ridge tracing as generated by coding from a horizontal line through the core. Points of intersection numbered outwards from the core.

Ridge number:	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Code.	70	00	30	63	00	00	37	80	63	20	36	33	36	46	28	83	60	23	83	63	33

Ridge (cont.):	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
Code:	33	30	72	83	86	63	78	30	83	60	00	73	80	60	80	30	FF	FF	FF	FF

Figure 5. 82-digit Vector Generated from Figure 4--(showing also the intersection point numbers that correspond to each digit pair (for easy interpretation).

point numbers (from Figure 4) are also shown with their corresponding digit pairs. (These intersection point numbers are not normally recorded, and they form no part of the topological code.) Digit pairs are juxtaposed, and each pair separated from the next. It is important to remember that each digit pair is just that--a pair of digits; they should never be interpreted together as being one number.

After some experience had been gained, the average total time taken to trace a print and to generate and record the 82 digit vector from it, was roughly seven minutes.

1.7 "Dependent Pairs"

"Dependent pairs" of digits occur in a line-generated vector whenever the same characteristic is observed during the exploration of two adjacent ridges. If, for instance, one ridge runs into a bifurcation arriving as if from the 'left hand fork' (coded '2'), then it is quite likely that an adjacent ridge (on the appropriate side) will run into the same bifurcation as if from the 'right hand fork' (coded '4'). Such pairing is not guaranteed, however, as some other 'event' may occur first on either of the two ridges in question and effectively stop the exploration getting as far as that bifurcation.

Consequently, there is a marked tendency for '2's and '4's to occur within the vectors in the combinations "4* 2*" or "*2 *4". (The asterisks simply mean 'any code.') The first combination ("4* 2*") appears when a bifurcation is doubly accessed ABOVE the generating line, and the second combination when it happens BELOW the generating line. Similarly combinations "8* 6*" and "6* 8*" are also 'dependent pairs'; they appear when a ridge-ending which faces towards the generating line is doubly accessed from two adjacent ridges.

In comparing two vectors the aim will be to identify digital substrings which are identical (or almost identical). The occurrence of dependent pairs requires that any scoring system adopted (as a measure of "closeness") allow for the fact that a dependent pair of digits refers to only one characteristic, rather than two. Their preservation (i.e., appearance in the same combination in the other vector) is therefore less significant (as an indication of a possible 'match') than preservation of two non-dependent digits in similar circumstances.

1.8.1 Frequency Analysis: Aims

Some sort of frequency analysis experiment is a necessary preliminary to development of any effective vector comparison algorithms. The aims of such analysis are:

(a) To establish the frequencies with which the various selected "event codes" occur within the vectors.

(b) To determine if those frequencies are uniform over different physical regions of the prints--and, if they are not, to determine the extent of the variation.

(c) To determine how often ridge event codes appear in dependent pairs.

1.8.2 <u>Frequency Analysis: Results</u>

One-hundred and fifty-two prints were coded according to the scheme described above (paragraph 1.6). All of those prints were ulnar loops from right hand fingers. No selection was made on the basis of ridge-count or of any other characteristics. Analysis of the generated vectors by computer yielded the following results:

(a) Length of vectors: ignoring the padding('F's) the average number of digit pairs from the left hand side of the core was 18.7 (maximum 20, minimum 14). On the right hand side of the core the mean length was 15.2 pairs (maximum 20, minimum 10). These figures give an indication of how many ridges were inter sected by the generating line within the the confines of the central rectangle (0.78" x 0.71"--see paragraph 1.6).

(b) Dependent pairs: Altogether the code '2' appeared 1078 times. Of those appearances it was accompanied by a code '4' in the dependent position in 63.5% of cases. Conversely, code '4' appeared 1111 times and had an accompanying dependent '2' 61.7% of the time. Code '6' appeared 1235 times, with a dependent '8' in 60.7% of cases. Code '8' appeared 1241 times, with a dependent '6' in 60.4% of cases.

(c) Global frequencies: Figure 6 shows the global frequencies of the various event codes. ('Global' here means without any breakdown into different physical regions of the print.)

Figure 7 shows those frequencies divided into two clases--looking 'upwards' from the generating line and looking 'downwards.' That division shows up significant variations--the most obvious being the frequency with which ridges run 'out of sight' (code "0"). It is 8.1% when exploring upwards, and 38.8% when exploring downwards.

More detailed analysis was performed for four distinct physical areas of the print: Figure 8 deals with the ridges on the left of the core, looking downwards--Figure 9 with those same ridges but looking up-

CODE.	MEANING (Summary)	FREQUENCY.	PERCENTAGE.
0	Runs out of sight.	2415	23.5
1	-----Not allocated.	0	0.0
2	Meets bifurcation (left fork)	1078	10.5
3	Ridge ends.	1676	16.3
4	Meets bifurcation(right fork)	1111	10.8
5	-----Not allocated	0	0.0
6	Faces ridge-ending (left)	1235	12.0
7	Ridge bifurcates ahead.	1280	12.4
8	Faces ridge-ending (right)	1241	12.1
9	-----Not allocated	0	0.0
A	Runs into scarred area.	93	0.9
B	Runs into unclear area.	123	1.2
C	Meets compound.	38	0.4
D	-----Not allocated	0	0.0
E	-----Not allocated	0	0.0
F	-----Not allocated	0	0.0
TOTAL		10290	100.0

Figure 6. Global code frequencies.

FREQUENCY

CODE.	UPWARDS.	DOWNWARDS.
0	419	1996
1	0	0
2	707	371
3	906	770
4	719	392
5	0	0
6	703	532
7	816	464
8	725	516
9	0	0
A	47	46
B	74	49
C	29	9
D	0	0
E	0	0
F	0	0
TOTAL	5145	5145

PERCENTAGE

CODE.	UPWARDS.	DOWNWARDS.
0	8.1	38.8
1	0.0	0.0
2	13.7	7.2
3	17.6	15.0
4	14.0	7.6
5	0.0	0.0
6	13.7	10.3
7	15.9	9.0
8	14.1	10.0
9	0.0	0.0
A	0.9	0.9
B	1.4	1.0
C	0.6	0.2
D	0.0	0.0
E	0.0	0.0
F	0.0	0.0
TOTAL	100.0	100.0

Figure 7. Upwards and downwards code frequencies.

wards. Figure 10 deals with ridges on the right of the core, looking upwards--and Figure 11 with the same ridges, looking downwards. In each case the ridges were divided into 4 separate ridge-bands (using numbering from the core outwards). The upper table in each of the figures is a simple frequency 'count,' and the lower table is the expression of those counts as a percentage of the total number of times that a code (rather than an 'F') was found in that ridge band.

1.8.3 Frequency Analysis: Conclusions

Detailed scrutiny of these tables can be interesting. For instance, one observes in Figure 8 the peaking of the incidence of facing ridge-endings (codes '6' and '8') when looking downwards from the generating line on the left hand side of the core. This occurs in ridge-bands 6-10 and 11-15. The physical interpretation of this is the high incidence of ridges which run from the area of the delta and which end as they approach the area of the core. Also notice that the frequency of the code '0' varies from 85.8% (Figure 11, ridge-band 16-20) to just 1.2% (Figure 10, ridge-band 1-5).

There is only one general and importrant conclusion to be drawn from these results and it is: variation of code frequencies over different areas of the print is so marked that it will be impossible to construct one single global frequency chart (or a derived global scoring system) that bears any meaningful relationship to actual code frequencies. The two directions ('upwards' and 'downwards') need to be distinguished in any scoring system as do the different ridge-bands. It is clear that different score 'tables' will have to be constructed for each different area of the print.

1.9 Anticipated Problems in Vector Comparison

There are various types of change that should be expected to occur between topological vector codes representing successive impressions of the same finger. Some have already been touched upon.

There are four principal causes of change:

(a) Graphical mutation: the changing of characistic types from bifurcation to ridge-ending and vice-versa (due to inking and pressure differences). This change will alter some digits from one vector to the other. However, its effect will be 'local' to one or two ridges.

(b) Core misplacement: the core may be placed or interpreted differently, especially if the two impressions are coded by different operators. Such core misplacement will produce 'shifting' of the entire vector either to the left or to the right. For example, a misplacement by two ridges to the right, will produce a shift of the entire vector by two digit-PAIRS to the left.

FREQUENCY ANALYSIS FOR CODES FOUND ON THE LEFT HAND SIDE OF THE CORE, LOOKING DOWNWARDS

	RIDGE-BANDS (NUMBERED FROM THE CORE CENTRE)							
CODE	1-5	6-10	11-15	16-20	1-10	11-20	1-20	CODE
0	79	59	266	451	138	717	855	0
1	0	0	0	0	0	0	0	1
2	72	68	22	8	140	30	170	2
3	163	133	51	34	296	85	381	3
4	67	70	17	7	137	24	161	4
5	0	0	0	0	0	0	0	5
6	102	140	157	33	242	190	432	6
7	180	111	87	6	291	93	384	7
8	84	151	135	17	235	152	387	8
9	0	0	0	0	0	0	0	9
A	8	11	15	2	19	17	36	A
B	3	17	8	5	20	13	33	B
C	2	0	0	0	2	0	2	C
D	0	0	0	0	0	0	0	D
E	0	0	0	0	0	0	0	E
F	0	0	0	0	0	0	0	F
TOTAL	760	760	758	563	1520	1321	2841	TOTAL

FREQUENCIES EXPRESSED AS PERCENTAGE OF TOTALS FOR EACH RIDGE-BAND

	RIDGE-BANDS (NUMBERED FROM THE CORE CENTRE)							
CODE	1-5	6-10	11-15	16-20	1-10	11-20	1-20	CODE
0	10.4	7.8	35.1	80.1	9.1	54.3	30.1	0
1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1
2	9.5	8.9	2.9	1.4	9.2	2.3	6.0	2
3	21.4	17.5	6.7	6.0	19.5	6.4	13.4	3
4	8.8	9.2	2.2	1.2	9.0	1.8	5.7	4
5	0.0	0.0	0.0	0.0	0.0	0.0	0.0	5
6	13.4	18.4	20.7	5.9	15.9	14.4	15.2	6
7	23.7	14.6	11.5	1.1	19.1	7.0	13.5	7
8	11.1	19.9	17.8	3.0	15.5	11.5	13.6	8
9	0.0	0.0	0.0	0.0	0.0	0.0	0.0	9
A	1.1	1.4	2.0	0.4	1.3	1.3	1.3	A
B	0.4	2.2	1.1	0.9	1.3	1.0	1.2	B
C	0.3	0.0	0.0	0.0	0.1	0.0	0.1	C
D	0.0	0.0	0.0	0.0	0.0	0.0	0.0	D
E	0.0	0.0	0.0	0.0	0.0	0.0	0.0	E
F	0.0	0.0	0.0	0.0	0.0	0.0	0.0	F

Figure 8. <u>Detailed frequency analysis table--LHS downwards</u>

FREQUENCY ANALYSIS FOR CODES FOUND ON THE LEFT HAND SIDE OF THE CORE, LOOKING UPWARDS

	RIDGE-BANDS (NUMBERED FROM THE CORE CENTRE)							
CODE	1-5	6-10	11-15	16-20	1-10	11-20	1-20	CODE
0	34	47	79	80	81	159	240	0
1	0	0	0	0	0	0	0	1
2	66	163	109	76	329	185	514	2
3	142	214	192	92	356	284	640	3
4	174	154	119	81	328	200	528	4
5	0	0	0	0	0	0	0	5
6	54	61	89	67	115	156	271	6
7	101	47	66	68	148	134	282	7
8	71	61	83	75	132	158	290	8
9	0	0	0	0	0	0	0	9
A	0	9	5	4	9	9	18	A
B	2	3	16	20	5	36	41	B
C	16	1	0	0	17	0	17	C
D	0	0	0	0	0	0	0	D
E	0	0	0	0	0	0	0	E
F	0	0	0	0	0	0	0	F
TOTAL	760	760	758	563	1520	1321	2841	TOTAL

FREQUENCIES EXPRESSED AS PERCENTAGE OF TOTALS FOR EACH RIDGE-BAND

	RIDGE-BANDS (NUMBERED FROM THE CORE CENTRE)							
CODE	1-5	6-10	11-15	16-20	1-10	11-20	1-20	CODE
0	4.5	6.2	10.4	14.2	5.3	12.0	8.4	0
1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1
2	21.8	21.4	14.4	13.5	21.6	14.0	18.1	2
3	18.7	28.2	25.3	16.3	23.4	21.5	22.5	3
4	22.9	20.3	15.7	14.4	21.6	15.1	18.6	4
5	0.0	0.0	0.0	0.0	0.0	0.0	0.0	5
6	7.1	8.0	11.7	11.9	7.6	11.8	9.5	6
7	13.3	6.2	8.7	12.1	9.7	10.1	9.9	7
8	9.3	8.0	10.9	13.3	8.7	12.0	10.2	8
9	0.0	0.0	0.0	0.0	0.0	0.0	0.0	9
A	0.0	1.2	0.7	0.7	0.6	0.7	0.6	A
B	0.3	0.4	2.1	3.6	0.3	2.7	1.4	B
C	2.1	0.1	0.0	0.0	1.1	0.0	0.6	C
D	0.0	0.0	0.0	0.0	0.0	0.0	0.0	D
E	0.0	0.0	0.0	0.0	0.0	0.0	0.0	E
F	0.0	0.0	0.0	0.0	0.0	0.0	0.0	F

Figure 9. Detailed frequency analysis table--LHS upwards

FREQUENCY ANALYSIS FOR CODES FOUND ON THE RIGHT HAND SIDE OF THE CORE, LOOKING UPWARDS

CODE	RIDGE-BANDS (NUMBERED FROM THE CORE CENTRE)							
	1-5	6-10	11-15	16-20	1-10	11-20	1-20	CODE
0	9	36	89	45	45	134	179	0
1	0	0	0	0	0	0	0	1
2	66	61	57	9	127	66	193	2
3	72	85	89	20	157	109	266	3
4	67	56	57	11	123	68	191	4
5	0	0	0	0	0	0	0	5
6	133	176	103	20	309	123	432	6
7	266	145	100	23	411	123	534	7
8	134	177	104	20	311	124	435	8
9	0	0	0	0	0	0	0	9
A	2	17	10	0	19	10	29	A
B	1	7	20	5	8	25	33	B
C	10	0	0	2	10	2	12	C
D	0	0	0	0	0	0	0	D
E	0	0	0	0	0	0	0	E
F	0	0	0	0	0	0	0	F
TOTAL	760	760	629	155	1520	784	2304	TOTAL

FREQUENCIES EXPRESSED AS PERCENTAGE OF TOTALS FOR EACH RIDGE-BAND

CODE	RIDGE-BANDS (NUMBERED FROM THE CORE CENTRE)							
	1-5	6-10	11-15	16-20	1-10	11-20	1-20	CODE
0	1.2	4.7	14.1	29.0	3.0	17.1	7.8	0
1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1
2	8.7	8.0	9.1	5.8	8.4	8.4	8.4	2
3	9.5	11.2	14.1	12.9	10.3	13.9	11.5	3
4	8.8	7.4	9.1	7.1	8.1	8.7	8.3	4
5	0.0	0.0	0.0	0.0	0.0	0.0	0.0	5
6	17.5	23.2	16.4	12.9	20.3	15.7	18.8	6
7	35.0	19.1	15.9	14.8	27.0	15.7	23.2	7
8	17.6	23.3	16.5	12.9	20.5	15.8	18.9	8
9	0.0	0.0	0.0	0.0	0.0	0.0	0.0	9
A	0.3	2.2	1.6	0.0	1.3	1.3	1.3	A
B	0.1	0.9	3.2	3.2	0.5	3.2	1.4	B
C	1.3	0.0	0.0	1.3	0.7	0.3	0.5	C
D	0.0	0.0	0.0	0.0	0.0	0.0	0.0	D
E	0.0	0.0	0.0	0.0	0.0	0.0	0.0	E
F	0.0	0.0	0.0	0.0	0.0	0.0	0.0	F

Figure 10. **Detailed frequency analysis table-RHS upwards**

FREQUENCY ANALYSIS FOR CODES FOUND ON THE RIGHT HAND SIDE OF THE CORE, LOOKING DOWNWARDS

CODE	RIDGE-BANDS (NUMBERED FROM THE CORE CENTRE)							
	1-5	6-10	11-15	16-20	1-10	11-20	1-20	CODE
0	173	342	493	133	515	626	1141	0
1	0	0	0	0	0	0	0	1
2	109	74	15	3	183	18	201	2
3	193	140	53	3	333	56	389	3
4	105	100	21	5	205	26	231	4
5	0	0	0	0	0	0	0	5
6	45	36	17	2	81	19	100	6
7	64	10	5	1	74	6	80	7
8	61	44	16	8	105	24	129	8
9	0	0	0	0	0	0	0	9
A	2	4	4	0	6	4	10	A
B	7	4	5	0	11	5	16	B
C	1	6	0	0	7	0	7	C
D	0	0	0	0	0	0	0	D
E	0	0	0	0	0	0	0	E
F	0	0	0	0	0	0	0	F
TOTAL	760	760	629	155	1520	784	2304	TOTAL

FREQUENCIES EXPRESSED AS PERCENTAGE OF TOTALS FOR EACH RIDGE-BAND

CODE	RIDGE-BANDS (NUMBERED FROM THE CORE CENTRE)							
	1-5	6-10	11-15	16-20	1-10	11-20	1-20	CODE
0	22.8	45.0	78.4	85.8	33.9	79.8	49.5	0
1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1
2	14.3	9.7	2.4	1.9	12.0	2.3	8.7	2
3	25.4	18.4	8.4	1.9	21.9	7.1	16.9	3
4	13.8	13.2	3.3	3.2	13.5	3.3	10.0	4
5	0.0	0.0	0.0	0.0	0.0	0.0	0.0	5
6	5.9	4.7	2.7	1.3	5.3	2.4	4.3	6
7	8.4	1.3	0.8	0.6	4.9	0.8	3.5	7
8	8.0	5.8	2.5	5.2	6.9	3.1	5.6	8
9	0.0	0.0	0.0	0.0	0.0	0.0	0.0	9
A	0.3	0.5	0.6	0.0	0.4	0.5	0.4	A
B	0.9	0.5	0.8	0.0	0.7	0.6	0.7	B
C	0.1	0.8	0.0	0.0	0.5	0.0	0.3	C
D	0.0	0.0	0.0	0.0	0.0	0.0	0.0	D
E	0.0	0.0	0.0	0.0	0.0	0.0	0.0	E
F	0.0	0.0	0.0	0.0	0.0	0.0	0.0	F

Figure 11. Detailed frequency analysis table-RHS downwards

(c) 'Subsidiary' (or 'incipient') ridges: appearance or disappearance of these extra ridges between principal ridges will, if they intersect the generating line, cause introduction or deletion of one digit pair and a resulting shift of all digit pairs 'outside' (i.e., further from the center).

(d) Line placement errors: it would be foolish to expect orientation of the horizontal line to be exactly repeatable as it involved some subjective judgments. It also depends to an extent on the clarity of the flexion crease in the print. It would be more reasonable to expect it to be repeatable within, say, 20 degrees. [In fact, when substantial numbers of mated prints had been coded using this scheme, (albeit by the same operator) it was found that line orientation differed by less than 5 degrees in 73% of pairs, by 5-10 degrees in 21% of pairs, by 10-15 degrees in 5% of pairs, and by over 20 degrees in only 1% of pairs.] If the line is drawn to pass the 'wrong' side of a ridge-ending or bifurcation, then the number of ridges intersected by the generating line will change (from 1 to 2 or vice-versa in the case of a bifurcation, and from 0 to 1 or vice-versa in the case of a ridge-ending). Exactly the same effect will be produced if plastic distortion of the print causes some characteristics to move from one side of the generating line to the other.

Any comparison algorithms must be capable of recognizing similarity between two vectors while allowing for any or all of these types of change. Most importantly, the combined effect of several different, but superimposed, substring shifts must be catered for in the formulation of any score (or other indication of 'closeness') when two vectors are compared.

1.10 Description of Databases

The evaluation of various matching algorithms has to be conducted by testing them on various databases. This is a brief description of the preparation and use of the early databases.

The first database (hereafter called 'TESTSET.1') comprised 100 mated pairs of coded ulnar loops. All were taken from right hand fingers; all were manually encoded from FBI record cards. The encoding processes and data entry steps were all checked for accuracy. One hundred fingers were selected where two different impressions of that finger were available. In every case the two impressions had been taken by different officials, with the intervening time lapse varying from a few days to 9 years (and with an average of 2.2 years). Relatively clear prints (i.e., ones that were not badly smudged) were chosen to give maximum information. Scarred prints were not avoided; in fact several badly scarred prints were included in TESTSET.1.

The prints were divided into an 'A' set and a 'B' set. Each of the 100 prints in the 'A' set has a 'mate' in the 'B' set. ('Mate print' or 'matching print' are useful brief ways of saying 'a different impression from the same finger'). Each impression is identified by its set (i.e., 'A' or 'B'), by its card number (representing the owner of the finger in question), and by its finger number (numbers 1-10; fingers 1-5 are on the right hand, 6-10 on the left. Numbers 1 and 6 are the thumbs.) Each of these 200 prints was traced and coded, and the resulting vectors also referred to by these same indices (e.g., Card 32 A, finger 3).

In each test the 'B' set of vectors would be used as the 'file' set as if they were an established fingerprint collection. The 'A' set would be treated as 'search' enquiries, being taken one at a time and compared with every one of the 'B' set in turn. 'A' set vectors were never compared with other 'A' set vectors; nor 'B's with other 'B's.

Each experimental algorithm test using TESTSET.1 therefore involved ten thousand vector comparisons--of which 100 were 'matches' and the other 9900 were 'mismatches'.

2. DESCRIPTION OF BASIC MATCHING ALGORITHM

2.1 Relationship Between the Various Matching Algorithms

What follows here is a description of the original vector comparison algorithm. It served well as a basis on which to build all later improvements. A proper understanding of each of the distinct stages of this algorithm will serve well as a framework within which to understand all subsequent developments.

The algorithm described in paragraph 2.2 is, in some particulars, comparatively crude. It is, nevertheless, surprisingly effective. It is called 'MATCH1' and will be referred to as such.

2.2 Description of 'MATCH1'

There are seven distinct phases to this algorithm; two are preliminary and five form the actual comparison process. Each will be described in turn.

2.2.1 Preliminary Stage 1--Fileset Analysis

Suppose that the statistical analysis of paragraph 1.8 had led to the creation of a fixed, permanent scoring system. Suppose further that an algorithm incorporating that scoring system had been tested against the same dataset that had been used to devise the scoring system. Quite proper objections could then be raised as to the 'correctness' of scientific procedure. Parameters derived from one set of data should not be tested against that same set. It would seem objectionable, therefore, to use a scoring system derived from one file set of prints on that same set of prints. Indeed so--UNLESS that was to be the approach taken IN PRACTICE.

There is no reason at all why an operational fingerprint system should not periodically reevaluate its scoring system in the light of the information (prints) currently stored within its memory. In fact, one would expect any 'intelligent' system to do just that. The frequency with which such reevaluations should take place would depend on the rate of change of the collection's size and content. As the collection became larger the various code frequencies would tend, asymptotically, towards certain stable limits. Those limits would correspond to the 'natural' distribution (i.e., over ALL fingerprints) of code frequencies. Consequently once the collection had attained a certain size (i.e., large enough) periodic reevaluation of the scoring system would become unnecessary).

Fileset analysis is the first preliminary operation conducted by MATCH1 before any individual vector comparisons are made. The analysis is of the fileset alone (the 'B' set) with no knowledge of the search enquiries (the 'A' set) being assumed. The vectors stored within the fileset are of length 82 digits, representing up to 41 ridges. No specific distinction is made hereafter between ridges that fall to the left of the core and those that fall to the right of it: rather the 82 ridges (in order, from left to right) are divided into ridge bands.

The ridge-band width for this analysis is to be a parameter of the program. (It was '5' when the tables in Figures 9-11 were produced.) Let us suppose that this parameter (which will be called 'BANDWIDTH') is set at 5. Then, with vectors of length 82 digits, derived from 41 ridge intersection points, there will be 9 ridge bands. (These cover ridges 1-5, 6-10, 11-15,... 36-40, and 41-45 respectively. Ridges 42-45 do not 'exist,' and so the ninth ridge band only contains the last (41st) pair of digits in each vector.)

Each ridge band is to be analyzed separately, as are the two directions ('upwards' and 'downwards' from the horizontal line). Simple code frequency analysis conducted on all the vectors stored in the fileset ultimately yields a real matrix P, of three dimensions thus:

P (J,K,L)	: J=0,15	J represents one of the hexadecimal 'event' codes.
	: K=1,9	K is the ridge-band number. (They are numbered from left to right.)
	: L=1,2	L shows one of two 'directions.' (L=1 for 'upwards':i.e., first digit of a pair.) (L=2 for 'downwards': i.e., 2nd digit of a pair.)

The combination of any value of K with a value of L specifies one of 18 possible 'ridge areas'. P(J,K,L) is the proportion of codes in the (K,L) ridge area that had the value J.

Clearly $0 \leq P(J,K,L) \leq 1$ for all (J,K,L). Also the sum of P(J,K,L) over J for any fixed pair (K,L) is 1.0.

2.2.2 Preliminary Stage 2--Setting Up Score-Reference Matrix

From the three dimensional frequency matrix P, a four dimensional Score-reference matrix S is constructed. S is to be regarded as a 'look-up table' of initial scores to be awarded during the vector comparison process.

A score S(i,j,k,l) will be awarded initially when code 'i' appears in the search vector opposite code "j" in the file vector, in corresponding (digit) positions which fall in the (k,l) ridge area.

That score S(i,j,k,l) is an indication of the value of such a coincidence in indicating that the search and file vectors under comparison are 'matched.' It could also be regarded as a measure of the 'unlikelihood' of that coincidence occurring by chance had the file vector been selected completely at random from the population of 'all fingerprints'.

The calculation of the matrix S is done according to these rules:

(a) For each i,j,k,l such that i=j and i,j belong to the set [0,2,3,4,6,7,8,C] then

$$S(i,j,k,l) = \text{minimum} \left[\text{"BOUND"}, \frac{1}{P(j,k,l)} \right]$$

Where "BOUND" is another parameter--it is an imposed upper bound on the values taken by elements of the matrix S: it's purpose is to limit the possible effect of isolated, but rare, coincidences.

These elements of S are the 'exact match' scores.

(b) For all i,j,k,l such that at least one of i and j is either 10, 11 or 12 (i.e., hexadecimal A, B or C) then

$S(i,j,k,l) = 1.0$

These elements of S represent all the appearances (either in the file vector or in the search vector) of the codes for 'scarred' or 'unclear' areas, and for 'compounds.' The reason for allocation of a score of 1.0 will become apparent in paragraph 2.2.5.

(c) Paragraph 1.5 described the phenomenon of 'graphical mutation' and related this to the selection of event codes. The pairs of codes [(2,3), (3,4), (6,7), (7,8)] can be regarded as 'close matches' as they could be observed in corresponding positions within mated vectors as a result of such graphical mutations.

Consequently if the comparison algorithm is to recognize 'close matches' as indications of a possible match (albeit not as strong an indication of this as 'exact matches' would be) that policy can be effected by allocating positive values to the subset of S defined:

[S(i,j,k,l) such that the unordered pair (i,j) belongs to the set of unordered pairs [(2,3), (3,4), (6,7), (7,8)].

This set of elements within S are hereafter called the 'close match' scores. For any particular (k,l) they will appear as entries in the (i,j) table which are just off the leading diagonal. The entries of the leading diagonal itself are the 'exact match' scores.

(d) For all i,j,k,l not covered by one of the rules a, b or c above:

S(i,j,k,l) = 0

The matrix S (when there are 9 ridge bands) could be regarded as 18 different comparison 'tables' each one of which might typically appear as shown in Figure 12. (In Figure 12 the close match scores have been set to 2.0 and an upper bound of 15.0 applied.)

2.2.3 <u>Comparison Stage 1--Formation of File and Search Matrices</u>

The vector comparison process itself begins with a file vector (B(i), i=1,82), a search vector (A(i), i=1,82) and the established score reference matrix S.

An important parameter not yet introduced is "MAXSHIFT." "MAXSHIFT" is the maximum number of ridge shifts (either to left or right) that is to be anticipated by the comparison algorithm. Such shifts are likely to have occurred as a result of the types of distortion described in paragraph 1.9 subparagraphs (b), (c) and (d).

Let us suppose that up to 5 ridge shifts should be anticipated (i.e., MAXSHIFT=5). Then comparison of vector A with vector B will need to allow for relative shifting by up to five digit-PAIRS. This is accomplished by use of standard array processing techniques thus:

	j values															
i values	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	2.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	1.0	1.0	0.0	0.0	0.0
1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
2	0.0	0.0	8.0	2.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	1.0	1.0	0.0	0.0	0.0
3	0.0	0.0	2.0	9.0	2.0	0.0	0.0	0.0	0.0	0.0	1.0	1.0	1.0	0.0	0.0	0.0
4	0.0	0.0	0.0	2.0	7.0	0.0	0.0	0.0	0.0	0.0	1.0	1.0	1.0	0.0	0.0	0.0
5	0.0	0.0	0.0	0.0	0.0	15.0	0.0	0.0	0.0	0.0	1.0	1.0	1.0	0.0	0.0	0.0
6	0.0	0.0	0.0	0.0	0.0	0.0	10.0	2.0	0.0	0.0	1.0	1.0	1.0	0.0	0.0	0.0
7	0.0	0.0	0.0	0.0	0.0	0.0	2.0	11.0	2.0	0.0	1.0	1.0	1.0	0.0	0.0	0.0
8	0.0	0.0	0.0	0.0	0.0	0.0	0.0	2.0	9.0	0.0	1.0	1.0	1.0	0.0	0.0	0.0
9	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
A	1.0	0.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	0.0	1.0	1.0	1.0	0.0	0.0	0.0
B	1.0	0.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	0.0	1.0	1.0	1.0	0.0	0.0	0.0
C	1.0	0.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	0.0	1.0	1.0	15.0	0.0	0.0	0.0
D	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
E	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
F	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

Table of S(i,j,k,l) for a fixed (k,l) with upper bound 15.0 and close match scores 2.0.
(Codes 1,9,D,E are not allocated.)

Figure 12. Typical S(i,j,k,l) table for fixed (k,l) showing close match scores of 2. Real number entries rounded to 1 decimal place.

(a) The search vector A is used to construct the SEARCH MATRIX "C". C will have 82 columns and the number of rows will be given by (2 x MAXSHIFT) + 1. Each row will be a copy of the vector A, but the copy will be progressively shifted to the left or right by from 0 to MAXSHIFT digit pairs. The central row will be an exact copy of A. The top (first) row will show A shifted 5 digit pairs to the left; the second ... 4 digit pairs to the left; the bottom row ... 5 digit pairs to the right. Some digits of A may be 'lost' off the ends of some of the rows--and gaps caused by the shifting are padded with pairs of 'F's. Such a search matrix can be seen in Figure 13.

(b) The file vector B is used to create a FILE MATRIX D, of identical dimensions to C. It is formed by faithful duplication of the vector B, without shifting, the appropriate number of times. Every row of D is an exact copy of the vector B. No padding is needed and no digits are lost from row ends. Figure 13 also shows such a FILE MATRIX.

2.2.4 Comparison Stage 2--Comparison of File and Search Matrices

The search and file matrices, C and D, are then compared element by element, and the INITIAL SCORE MATRIX, is formed as the result. The initial score matrix will be called E. E has the same dimensions as C and D.

For each r,s the element E(r,s) depends only on C(r,s) and D(r,s). Each element E(r,s) is evaluated by 'looking up' C(r,s) and D(r,s) in the score reference matrix S:

```
E(r,s) = S(I,J,K,L) where I = C(r,s)
                          J = D(r,s)
                          (K,L) are determined by s
```

K and L are picked, for each s, to represent the 'ridge-area' to which the 's'th element of a vector would belong. Thus K will increase from 1 to 9 as s varies from 1 to 82, and L will be 1 if s is odd, 2 if s is even.

In other words C(r,s) and D(r,s) are 'looked up in the 'book' of comparison tables called "S." The values (K,L) are evaluated (from s) just to make sure that the appropriate table is 'looked up'.

2.2.5 Properties of the Initial Score Matrix

The feature of the initial score matrix E that begins to suggest whether or not vectors A and B are a matching pair is the presence (or absence)

```
          SEARCH SERIES (VECTOR) IS:
FF 70 30 80 60 30 30 30 30 80 66 38 86 68 36 36 48 26 08 87 32 64 80 82 64 33 86 88 63 63 30 80 63 37 20 B0 FF FF FF FF FF

           FILE SERIES (VECTOR) IS:
FF FF 30 80 60 60 30 40 30 70 36 38 86 48 36 36 48 27 07 87 32 64 70 82 64 34 86 68 73 63 30 80 60 36 B8 80 00 FF FF FF FF

          SEARCH MATRIX IS :-

30 30 30 30 80 66 38 86 68 36 36 48 26 08 87 32 64 80 82 64 33 86 88 63 63 30 80 63 37 20 B0 FF FF FF FF FF FF FF FF FF FF
60 30 30 30 30 80 66 38 86 68 36 36 48 26 08 87 32 64 80 82 64 33 86 88 63 63 30 80 63 37 20 B0 FF FF FF FF FF FF FF FF FF
80 60 30 30 30 30 80 66 38 86 68 36 36 48 26 08 87 32 64 80 82 64 33 86 88 63 63 30 80 63 37 20 B0 FF FF FF FF FF FF FF FF
30 80 60 30 30 30 30 80 66 38 86 68 36 36 48 26 08 87 32 64 80 82 64 33 86 88 63 63 30 80 63 37 20 B0 FF FF FF FF FF FF FF
70 30 80 60 30 30 30 30 80 66 38 86 68 36 36 48 26 08 87 32 64 80 82 64 33 86 88 63 63 30 80 63 37 20 B0 FF FF FF FF FF FF
FF 70 30 80 60 30 30 30 30 80 66 38 86 68 36 36 48 26 08 87 32 64 80 82 64 33 86 88 63 63 30 80 63 37 20 B0 FF FF FF FF FF
FF FF 70 30 80 60 30 30 30 30 80 66 38 86 68 36 36 48 26 08 87 32 64 80 82 64 33 86 88 63 63 30 80 63 37 20 B0 FF FF FF FF
FF FF FF 70 30 80 60 30 30 30 30 80 66 38 86 68 36 36 48 26 08 87 32 64 80 82 64 33 86 88 63 63 30 80 63 37 20 B0 FF FF FF
FF FF FF FF 70 30 80 60 30 30 30 30 80 66 38 86 68 36 36 48 26 08 87 32 64 80 82 64 33 86 88 63 63 30 80 63 37 20 B0 FF FF
FF FF FF FF FF 70 30 80 60 30 30 30 30 80 66 38 86 68 36 36 48 26 08 87 32 64 80 82 64 33 86 88 63 63 30 80 63 37 20 B0 FF
FF FF FF FF FF FF 70 30 80 60 30 30 30 30 80 66 38 86 68 36 36 48 26 08 87 32 64 80 82 64 33 86 88 63 63 30 80 63 37 20 B0

(NOTE: THIS IS A HEXADECIMAL, SINGLE-DIGIT MATRIX.)
(DO NOT BE CONFUSED BY JUXTAPOSITION.)

           FILE MATRIX IS :-

FF FF 30 80 60 60 30 40 30 70 36 38 86 48 36 36 48 27 07 87 32 64 70 82 64 34 86 68 73 63 30 80 60 36 B8 80 00 FF FF FF FF
FF FF 30 80 60 60 30 40 30 70 36 38 86 48 36 36 48 27 07 87 32 64 70 82 64 34 86 68 73 63 30 80 60 36 B8 80 00 FF FF FF FF
FF FF 30 80 60 60 30 40 30 70 36 38 86 48 36 36 48 27 07 87 32 64 70 82 64 34 86 68 73 63 30 80 60 36 B8 80 00 FF FF FF FF
FF FF 30 80 60 60 30 40 30 70 36 38 86 48 36 36 48 27 07 87 32 64 70 82 64 34 86 68 73 63 30 80 60 36 B8 80 00 FF FF FF FF
FF FF 30 80 60 60 30 40 30 70 36 38 86 48 36 36 48 27 07 87 32 64 70 82 64 34 86 68 73 63 30 80 60 36 B8 80 00 FF FF FF FF
FF FF 30 80 60 60 30 40 30 70 36 38 86 48 36 36 48 27 07 87 32 64 70 82 64 34 86 68 73 63 30 80 60 36 B8 80 00 FF FF FF FF
FF FF 30 80 60 60 30 40 30 70 36 38 86 48 36 36 48 27 07 87 32 64 70 82 64 34 86 68 73 63 30 80 60 36 B8 80 00 FF FF FF FF
FF FF 30 80 60 60 30 40 30 70 36 38 86 48 36 36 48 27 07 87 32 64 70 82 64 34 86 68 73 63 30 80 60 36 B8 80 00 FF FF FF FF
FF FF 30 80 60 60 30 40 30 70 36 38 86 48 36 36 48 27 07 87 32 64 70 82 64 34 86 68 73 63 30 80 60 36 B8 80 00 FF FF FF FF
FF FF 30 80 60 60 30 40 30 70 36 38 86 48 36 36 48 27 07 87 32 64 70 82 64 34 86 68 73 63 30 80 60 36 B8 80 00 FF FF FF FF
FF FF 30 80 60 60 30 40 30 70 36 38 86 48 36 36 48 27 07 87 32 64 70 82 64 34 86 68 73 63 30 80 60 36 B8 80 00 FF FF FF FF

(NOTE: THIS IS A HEXADECIMAL, SINGLE-DIGIT MATRIX.)
(DO NOT BE CONFUSED BY JUXTAPOSITION.)

INITIAL  SCORE  MATRIX IS :-

00 00 51 01 01 C0 40 00 00 00 44 05 04 05 00 50 00 00 00 00 D0 00 00 00 60 D0 50 50 00 00 12 00 00 00 00 00 00 00 00 00 00
00 00 51 01 01 02 00 00 00 00 44 40 00 00 00 00 00 00 00 A0 00 00 00 50 60 00 00 00 06 00 02 11 00 00 00 00 00 00 00 00 00
00 00 51 01 01 02 02 00 40 00 00 40 04 45 04 00 00 00 00 A0 05 65 00 50 00 00 00 00 00 56 A0 01 11 00 00 00 00 00 00 00 00
00 00 01 01 01 02 42 02 00 00 04 05 04 00 00 07 08 05 00 00 00 30 00 00 00 00 00 50 00 00 00 00 01 10 00 00 00 00 00 00 00
00 00 01 01 01 02 42 02 02 00 40 00 00 00 44 00 00 00 05 00 00 00 00 00 00 00 50 50 06 00 02 00 00 00 10 00 00 00 00 00 00
00 00 51 71 01 02 42 02 42 02 04 45 F4 05 44 57 58 40 E0 A5 D5 65 05 55 65 D0 5F 0F 06 56 A2 91 70 70 10 11 00 00 00 00 00
00 00 01 01 01 C2 42 02 42 02 00 00 00 00 00 57 00 00 00 00 00 00 00 50 00 05 00 00 00 56 00 01 01 00 10 11 11 00 00 00 00
00 00 00 01 01 02 02 02 42 02 40 00 04 05 04 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 01 00 10 10 01 00 00 00 00
00 00 00 00 01 02 02 02 42 02 40 40 F0 00 40 07 08 00 00 00 00 00 00 05 65 00 50 50 06 00 00 00 70 70 10 10 00 00 00 00 00
00 00 00 00 00 02 42 02 02 02 40 40 00 00 04 50 00 00 00 00 00 00 00 50 00 05 50 00 00 06 00 90 70 00 10 11 00 00 00 00 00
00 00 00 00 00 00 02 02 02 02 40 40 00 00 00 07 08 00 00 00 D0 00 00 00 00 D0 00 00 00 50 A0 90 00 00 10 11 01 00 00 00 00

(NOTE: THIS IS A HEXADECIMAL, SINGLE-DIGIT MATRIX.)
(DO NOT BE CONFUSED BY JUXTAPOSITION.)
```

Figure 13. Creation of search and file matrices (para. 2.2.3), and initial score matrix (para.2.2.4).

of horizontal strings of non-zero scores. Such a string within one row of E represents similarly placed rows within matrices C and D that were similar, or identical. Such strings, in turn, represent parts of the vectors A and B that were similar or identical. Where a high scoring continuously non-zero string occurs in the central row of E, then vectors A and B are probably mates and are correctly aligned. If such a high scoring string appears in one of the other rows of E, then A and B were probably mates but incorrectly aligned (i.e., there had been some 'shifting' error).

If, on the other hand, the matrix E appears to be a random scattering of scores with no discernible concentrations of non-zero scores, then it is likely that A and B were not mates. Figure 13 shows a typical INITIAL SCORE MATRIX which, in this case, has come from a mated pair of vectors. (For demonstration purposes, and to facilitate 'writing down' this matrix, all the elements of E have been rounded to the nearest integer and written as hexadecimal digits. Otherwise display would be exceedingly cumbersome.)

The task facing the remainder of the algorithm is to calculate a single score which will show whether 'significant' strings are present in the matrix E, or not--and thus provide an indication of whether A and B are mated vectors.

The methods used to do this are based on the idea of 'multiplying together' all the digits of each continuously non-zero horizontal string within E. Remember that the scores allocated (S(r,s)) for each 'exact match' (when C(r,s)=D(r,s)) were measures of the 'unlikelihood' of such coincidence [i.e.reciprocals of the probability of occuring by chance]. Consequently the product of a continuous series is a measure of the unlikelihood of that whole series occurring by chance. Typically non-matches are unlikely to display any continuously non-zero series of length greater than 6 digits. Matches can produce such series of lengths up to 50 or 60 digits.

Anticipation of this 'multiplying together' was the origin of the rules used in setting up the score matrix S. The significance of scores of "1.0" (rule (b) in paragraph 2.2.2) is that their appearances within the initial score matrix E do nothing to the product of a series, but they do preserve its continuity. Thus, appearance of scars, or inability to determine what does happen first during ridge exploration, is not given any significance in indicating a match, but it is not allowed to break up an otherwise continuous non-zero sequence that would be indicative of a match. Hence, the "1.0" allocation to any comparison involving codes "A" or "B." Comparisons involving code "C" were also allocated scores of 1.0, because true compounds are very rare and what normally appears as a compound is usually an ambiguous characteristic of some other sort.

2.2.6 Comparison Stage 3--Filtering for Dependent Pairs

As explained in paragraph 1.7 the repetition (from the search vector to the file vector) of a 'dependent pair' of digits is less significant in indicating a possible match than independent repetitions of those two codes would have been. There may then be scores E(r,s) and E(r,s+2) within the matrix E that form part of a continuously non-zero series, but whose appearance stems from repetition of a dependent pair of codes. Whenever such scores occur, their product (E(r,s) x E(r,s+2)) is more weighty than is appropriate in view of the dependence.

The matrix E is therefore 'filtered,' and the FILTERED SCORE MATRIX (F) created. F has exactly the same dimensions as E, D and C. The filtering step involves a reduction of scores stemming from repetitions of dependent code-pairs. It is accomplished by reference to the matrices C and D (to identify exactly where such pairs appeared in both).

The rule for score reduction is:

where E(r,s) and E(r,s+2) are exact-match scores derived from a dependent pair then

$$F(r,s) = \text{minimum}\ (E(r,s),\ E(r,s+2))$$
$$F(r,s+2) = 2.0$$

Elsewhere F(r,s) = E(r,s).

This reduction of scores gives a more reasonable weighting to the scores derived from dependent pairs, in the light of the results of the analysis on pair dependency given in paragraph 1.8.2 (b). The step typically reduces about 2 entries per row of the matrix E.

2.2.7 Comparison Stage 4--Condensing Digit Pairs to a Single Score

Careful examination of a large number of FILTERED SCORE MATRICES derived from mated vector pairs revealed that the fairly long continuously non-zero strings were not the most telling feature of the matrices; as well as revealing these completely non-zero strings they also exhibited much longer 'mostly non-zero' strings. These longer strings, even though they were interrupted by isolated zeros, seemed to be a better indication of 'match' or 'mismatch' by their presence or absence.

Often one digit of a pair (e.g., the 2nd digit) would be positive for several successive digit pairs--while the other digit of each pair scored zero. This will happen whenever the ridge pattern on one side of the generating line is well preserved, while being corrupted on the other side.

Prior to product evaluation, the matrix F is therefore 'condensed' into a matrix G (which has the same number of rows, but only half as many columns) in a manner which moves the emphasis onto the much longer 'mostly non-zero' strings.

The condensing rule applied in MATCH1 is:

$$G(r,s) = 0 \text{ if } F(r,2s-1) \text{ and } F(r,2s) \text{ are BOTH ZERO}$$

$$= \text{Maximum } (F(r,2s-1), F(r,2s)) \text{ if one, and only one, is non-zero}$$

$$= F(r,2s-1) \times F(r,2s) \text{ if BOTH ARE NON-ZERO}$$

Thus isolated zeros cease to 'break up' the long series that result from mated vectors. The products of these long series from matches are expected to far outweigh the products of any continuously non-zero series which occur 'by chance' (i.e., from a vector mismatch).

A condensed matrix from a mismatch is shown in Figure 14. (Once again the integer equivalent is displayed for ease of presentation). Note that there are no non-zero horizontal strings of length greater than 4.

2.2.8 Comparison Stage 5--Product Calculation and Score Formulation

Formulating a score from the condensed matrix G provides a further variety of options. MATCH1 calculates the product of each continuously non-zero string, and then SUMS those PRODUCTS for all strings detected in G.

Derivation of final score from the condensed matrix in Figure 14 is also shown in Figure 14.

2.3.1 Performance of MATCH1

At this stage there are two obvious 'performance indicators' for a comparison algorithm available after each test:

(a) The percentage of 'mates' ranked 1st (abbreviated to "MR1" hereafter). A mate is ranked 1st if, for a given search vector, its mate vector (in the fileset) scored higher than any other vectors in the fileset.

(b) The lowest rank obtained by a mate (hereafter "LMR").

Both of these have some practical significance. Conducting a search inquiry against a file collection on a computerized system should, hopefully, throw out the 'mate' as the top score; if not top, then it should be close to the top in order that little or no manual checking is required to identify it. The number of mates ranked first, and the lowest rank obtained by a 'mate' are clearly crucial questions in determining the efficiency of the system as a labor saving device.

```
FILTERED SCORE  MATRIX IS :-

01 51 01 01 00 00 0B 00 00 00 42 04 00 00 06 00 07 AF A0 00 00 02 0F 00 60 00 00 06 00 F2 12 00 00 00 00 00 00 00 00 00 00
01 51 01 01 00 00 00 0B 00 00 44 40 00 00 00 00 50 00 A0 00 00 00 00 2F 50 00 00 00 06 00 02 11 00 00 00 00 00 00 00 00 00
71 01 01 01 00 00 00 00 00 00 00 40 00 00 00 00 00 00 00 0F 00 00 00 50 00 00 00 00 00 00 A0 01 00 00 00 00 00 00 00 00 00
01 01 01 01 00 00 00 00 00 00 02 04 00 00 00 00 00 A0 00 00 00 00 00 00 00 00 00 06 00 02 00 00 00 00 00 00 00 00 00 00 00
01 51 01 01 00 00 00 00 00 00 40 00 00 00 40 00 00 00 A5 00 00 00 00 00 00 00 00 06 06 02 02 00 00 00 00 00 00 00 00 00 00
00 01 01 01 00 00 00 00 00 00 02 44 F0 00 40 00 00 00 00 00 00 00 00 20 50 00 00 00 06 00 A2 91 00 00 00 00 00 00 00 00 00
00 00 01 01 00 00 00 00 00 00 00 00 00 F0 00 00 50 00 00 00 0F 00 00 50 05 00 00 00 00 00 00 01 00 00 00 00 00 00 00 00 00
00 00 00 01 00 00 00 00 00 00 40 00 00 00 00 F0 50 00 00 00 00 00 00 00 00 02 05 06 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 40 40 F0 00 40 00 00 00 00 20 50 00 00 00 60 00 00 00 06 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 40 40 00 F0 00 00 00 00 00 00 00 02 0F 50 05 00 00 00 00 00 00 90 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 40 40 00 00 00 F0 50 A0 00 00 00 00 00 0F 00 02 05 00 00 00 A0 90 00 00 00 00 00 00 00 00 00

(NOTE: THIS IS A HEXADECIMAL, SINGLE-DIGIT MATRIX.)
(DO NOT BE CONFUSED BY JUXTAPOSITION.)

CONDENSED MATRIX IS :-

 1  5  1  1  0  0 11  0  0  0  8  4  0  0  6  0 7150 10  0  0  0  0  2 15  0  6  0  0  6  0 30  2  0  0  0  0  0  0  0  0  0  0
 1  5  1  1  0  0  0 11  0  0 16  4  0  0  0  0  5  0 10  0  0  0  0  0  0 30  5  0  0  0  6  0  2  1  0  0  0  0  0  0  0  0  0
 7  1  1  1  0  0  0  0  0  0  0  4  0  0  0  0  0  0  0 15  0 15  0  0  0  5  0  0  0  0  0  0 10  1  0  0  0  0  0  0  0  0  0
 1  1  1  1  0  0  0  0  0  0  2  4  0  0  0  0  0 10  0  0  0  0  0  0  0  0  0  0  0  6  0  2  0  0  0  0  0  0  0  0  0  0  0
 1  5  1  1  0  0  0  0  0  0  4  0  0  0  4  0  0  0 50  0  0  0  0  0  0  0  0  0  0  6  6  2  2  0  0  0  0  0  0  0  0  0  0
 0  1  1  1  0  0  0  0  0  0  2 16 15  0  4  0  0  0  0  0  0  0  0  0  0  2  5  0  0  0  6  0 20  9  0  0  0  0  0  0  0  0  0
 0  0  1  1  0  0  0  0  0  0  0  0  0 15  0  0  5  0  0  0 15  0 15  0  0  5  5  0  0  0  0  0  0  1  0  0  0  0  0  0  0  0  0
 0  0  0  1  0  0  0  0  0  0  4  0  0  0  0 15  5  0  0  0  0  0  0  0  0  0  0  2  5  6  0  0  0  0  0  0  0  0  0  0  0  0  0
 0  0  0  0  0  0  0  0  0  0  4  4 15  0  4  0  0  0  0  2  5  2  5  0  0  0  6  0  0  0  6  0  0  0  0  0  0  0  0  0  0  0  0
 0  0  0  0  0  0  0  0  0  0  4  4  0 15  0  0  0  0  0  0  0  0  0  2 15  5  5  0  0  0  0  0  0  9  0  0  0  0  0  0  0  0  0
 0  0  0  0  0  0  0  0  0  0  4  4  0  0  0 15  5 10  0  0  0  0  0  0  0 15  0  2  5  0  0  0 10  9  0  0  0  0  0  0  0  0  0

(NOTE: THIS IS A MATRIX OF DECIMAL INTEGERS.)

ANALYSIS OF CONTINUOUS-STRING PRODUCTS FOUND IN   CONDENSED MATRIX.

STRING LENGTH.    OCCURRENCES.        TOTAL   SCORE.

        1               31                267.
        2               18                789.
        3                6              12031.
        4                7                917.

TOTAL SCORE IS :           14004.
```

Figure 14. Sample filtered score matrix, condensed matrix (mismatch) and derived score evaluation.

When MATCH1 was run on TESTSET1 (the 100 pairs of ulnar loops) with these parameter values:

BOUND = 50.0	(upper bound on entries in the score reference matrix S)
MAXSHIFT = 5	(max. number of ridge-shifts anticipated)
CLOSE MATCH SCORES = 1	
BAND = 5	(ridge band width for frequency analysis, and basis for scoring system)
The results were:	:90% of mates ranked 1st (MR1)
	:lowest mate rank (LMR) = 25

To appreciate the nature of the scores produced by MATCH1, it is worth pointing out that the highest mate score achieved was in the order of 10 to the power 43. The lowest mate score was 3.9 x 10 ↑ 5. Most mate scores lay between 10 ↑ 15 and 10 ↑ 25. The range of the mismatch scores was from 100 to 10 ↑ 13, with most around 10 ↑ 4.

2.3.2 Parameter Variation

The performance was improved with adjustment of the parameters. The best results for MATCH1 on TESTSET1 were:

Parameters:	Performance:
BOUND = 15.0	MR1 = 95%
MAXSHIFT = 2	LMR = 8
BAND = 2	
CLOSE MATCH SCORES = 0	

A complete table of parameters/performance for MATCH1 is given in Figure 15.

2.3.3 Conclusions

The fact that the parameters given in paragraph 2.3.2 should give the best results is quite revealing:

(a) That MAXSHIFT = 2 gives better performance than MAXSHIFT = 5 suggests that ridge-shifting errors had not been too severe.

	Maxshift	Band	Bound	CloseMatch	MR1	LMR	Ranks not equal to 1.
Test 1	5	5	50	1	90	25	25,17,11,3,3,3,2,2,2,2.
Test 2	5	5	50	0	93	15	15,6,5,3,3,2,2.
Test 3	2	5	50	1	91	14	14,13,9,3,2,2,2,2.
Test 4	2	5	50	0	94	10	10,4,3,3,2,2.
Test 5	2	2	50	0	94	9	9,4,3,2,2,2.
Test 6	2	2	15	0	95	8	8,3,3,2,2.
Test 7	2	2	5	0	95	8	8,3,3,3,2.

Figure 15. Table of parameters/performance for MATCH1.

(b) Use of a smaller ridge band width (2, rather than 5) produces 21 different ridge bands (rather than 9). The degree of variation of code frequencies over these ridge bands can be seen from Figure 16 which is part of the program output which shows the 'exact-match' scores within the score reference matrix S after the fileset analysis. The table displays ONLY the exact match scores (i.e. S(I,J,K,L) for which I=J) and could be imagined to be a diagonal slice out of the S matrix. The presence of the two tables is a consequence of the two 'directions' (L = 1 or 2).

(c) Reducing the close match scores to zero aided performance. This is somewhat surprising--but shows that the predominant effect of allowing for graphical mutation in the scoring system is to boost mismatch scores.

The overall performance of MATCH1 is encouraging. Any "MR1" value greater than 90% is very good. (See Chapter 5 for comparison with a matching algorithm based on the traditional 'spatial' approach.)

2.4 Series Length/Density Experiment

Examination of some of the higher scoring mismatches, in detail, showed that when mismatches achieved high scores it was often as a result of very long strings of relatively low scores (notably containing a lot of '1's).

In order to find out if a string 'score-density' test could be used to aid discernment between matches and mismatches, statistical analysis of the products from strings of different lengths was conducted--both for matches and mismatches--and the results compared. The mean product yielded by a string of length n in the case of matches only varied significantly from the equivalent mean for mismatches when n exceeded 6. Therefore, for all values of n from 6 to 41 (the greatest series length possible in a condensed matrix) cutoff scores M(n) were evaluated such that a product less than M(n), from a string of length n, was significantly less likely to have come from a match than from a mismatch.

MATCH1 was adapted to implement these cutoff values for all values of n greater than 6--the rule being applied was that any product from a series of length n which scored less than M(n) was ignored (i.e. it was not added into the final score).

The performance of MATCH1 with such a score-density test incorporated was no improvement: in fact it was worse than before. Consequently score-density testing was rejected, for the time being, as an aid to differentiating between matches and mismatches. The details of the series-length analysis and calculation of cutoff points are not included here.

```
                        EXACT MATCH SCORES ALLOCATED AFTER FREQUENCY ANALYSIS OF FILE SET.

                          SCORES FOR THE FIRST DIGIT IN EACH PAIR, BY RIDGE BANDS

                                                         RIDGE BANDS.
CODE.  1- 2  3- 4  5- 6  7- 8  9-10 11-12 13-14 15-16 17-18 19-20 21-22 23-24 25-26 27-28 29-30 31-32 33-34 35-36 37-38 39-4

  0     8.7   7.4   9.4  10.9  11.6  14.3  15.0  15.0  15.0  15.0  15.0  15.0  15.0  15.0  15.0  11.6   9.5   5.8   4.1   2.
  1     0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0
  2     9.6   6.7   7.8   6.8   6.2   5.0   3.8   4.4   3.8   4.8   5.7  10.5  12.5  15.0  11.8  14.1   7.5  13.8  15.0  15
  3     5.6   7.0   3.8   4.6   3.5   3.6   4.1   3.9   6.7   4.4   4.3  14.3  10.0   9.1   9.1   6.4   6.9   8.6   6.4   8
  4     6.9   7.4   7.0   6.4   5.2   4.9   4.3   4.3   3.8   5.3  12.5   9.1  10.5  15.0  11.1   8.6  11.2  13.8  14.0   5
  5     0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0
  6     8.0   6.2   8.2   7.0  11.0  11.8  12.5  15.0  15.0  15.0   8.7   6.9   4.9   3.8   4.5   7.0   4.7   5.8   7.0   3
  7     5.6  12.3  14.5   9.8  10.4  15.0  15.0  10.5   7.1   8.3   3.8   2.9   3.6   4.8   4.9   6.0   8.2   5.3   5.8  15
  8     7.4   6.4   6.3   8.6  15.0  11.1  15.0  14.3  15.0   8.7  10.0   5.0   4.4   3.9   5.4   4.8   6.4   7.3   7.0  15
  9     0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0
  A     1.0   1.0   1.0   1.0   1.0   1.0   1.0   1.0   1.0   1.0   1.0   1.0   1.0   1.0   1.0   1.0   1.0   1.0   1.0   1.
  B     1.0   1.0   1.0   1.0   1.0   1.0   1.0   1.0   1.0   1.0   1.0   1.0   1.0   1.0   1.0   1.0   1.0   1.0   1.0   1
  C    15.0  15.0  15.0  15.0  15.0  15.0  15.0  15.0  15.0  15.0  15.0  15.0  15.0  15.0  15.0  15.0  15.0  15.0  15.0  15
  D     0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0
  E     0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0
  F     0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0


                              SCORES FOR SECOND DIGIT IN EACH PAIR.

                                                         RIDGE BANDS.
CODE.  1- 2  3- 4  5- 6  7- 8  9-10 11-12 13-14 15-16 17-18 19-20 21-22 23-24 25-26 27-28 29-30 31-32 33-34 35-36 37-38 39-4

  0     1.1   1.2   1.6   2.5   3.5   8.0  15.0   8.7   9.5   9.1  10.5   4.8   3.6   3.2   2.1   1.5   1.3   1.3   1.2   1.
  1     0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0
  2    15.0  15.0  15.0  15.0  15.0  14.3   7.1  10.5  10.0  10.5   8.7   7.4   5.6   7.7  10.5  15.0  15.0  15.0  15.0  15
  3    15.0  15.0  11.1  15.0  15.0   6.3   6.5   5.0   5.4   4.8   3.7   3.6   3.9   4.3   6.1   7.3  15.0  15.0  15.0  15
  4    15.0  15.0  15.0  15.0  15.0  15.0   8.0  10.5  10.5   9.5   8.0   5.6   7.4   6.3   7.4  15.0  15.0  15.0  15.0  15
  5     0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0
  6    15.0  15.0   7.0   5.8   4.8   4.1   6.5   5.7   7.1   6.7  13.3  15.0  15.0  12.5  15.0  15.0  15.0  15.0  15.0  15
  7    15.0  15.0  15.0   9.4   6.2   6.9   7.7   5.7   4.9   4.0   5.9  15.0  15.0  15.0  15.0  15.0  15.0  15.0  15.0  15
  8    15.0  15.0  15.0   5.5   4.8   5.0   5.1   7.7   5.9  13.3   8.0  14.3  15.0  15.0  15.0  15.0  15.0  15.0  15.0  15
  9     0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0
  A     1.0   1.0   1.0   1.0   1.0   1.0   1.0   1.0   1.0   1.0   1.0   1.0   1.0   1.0   1.0   1.0   1.0   1.0   1.0   1
  B     1.0   1.0   1.0   1.0   1.0   1.0   1.0   1.0   1.0   1.0   1.0   1.0   1.0   1.0   1.0   1.0   1.0   1.0   1.0   1
  C    15.0  15.0  15.0  15.0  15.0  15.0  15.0  15.0  15.0  15.0  15.0  15.0  15.0  15.0  15.0  15.0  15.0  15.0  15.0  15
  D     0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0
  E     0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0
  F     0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0
```

Figure 16. Samplè 'exact-match' score table for MATCH1 with BAND = 2 showing variation.

3. ALGORITHM DEVELOPMENTS: MATCH2 and MATCH3

3.1 Need for New Performance Measures

Testing MATCH1 with different parameter sets produced a variation in performance as described in paragraph 2.3. That variation in performance was signalled by the two performance measures in use at this stage, namely:

MR1 - Percentage of mates ranked first
LMR - Lowest mate rank

These measures would have been quite efficient in showing changes in performance had the original performance been much worse. If early tests had given MR1 values of 40% or so, then a change in the algorithm that raises MR1 to 65% is quite clearly a significant improvement. However, with MR1 already at 95%, it is questionable whether future adaptations can be properly assessed by, say, a change in MR1 to 97%. Such a small increment in MR1 is quite probably not significant statistically.

Moreover LMR can be changed dramatically by a program alteration that just happens to boost the one match score on which the 'worst ranking' depended. Such a change could be sheer fluke--from which one could not reasonably infer that performance on a much larger collection would be improved by that particular amendment.

For algorithms producing MR1 values greater than 90%, changes in MR1 and LMR actually depend on very few of the 10,000 comparisons done in each test. They depend only on the lowest match scores, and on the highest few mismatch scores. They are inadequate bases from which to draw meaningful conclusions about the value, or otherwise, of various algorithm adaptations.

3.2.1 Desirable Basis for Performance Measures

The most reliable performance indicators to use on data from (necessarily) limited tests would take into account a large part of the available data--if not all of it. In considering match and mismatch scores the points of critical interest, however, are the right hand tail of the mismatch score distribution and the left hand tail of the match score distribution. Especially important is the extent of their overlap, and this will only concern (hopefully) relatively few data points.

The proper way out of this apparent dilemma is to base performance measures on deductions about the behavior of the tails that can be made from the whole observed distributions. Such deductions can only be made if the natures (shapes) of the underlying distributions are known. (We are assuming that mismatch scores are from a population of independent identically distributed random variables. The same assumption is made about match scores.) If the shapes of these distributions are known then predictions about the behaviour of the tails and their overlap can be made with some degree of confidence.

For these reasons significant efforts have gone into the study of the distributions of match and mismatch scores. Most interesting is the question whether the match and mismatch score distributions are examples of probability density functions that are already known and understood. If they are, then percentiles and other details of the tails can be read from tables or calculated. If they are not pdf's with which we are familiar, then study of the distributions may not, ultimately, be much help.

3.2.2 MATCH1: Match and Mismatch Score Distributions

Each test with MATCH1 on TESTSET1 produced 100 match scores and 9900 mismatch scores. The match scores can be presumed to be independent of each other, but there is certainly a degree of dependence within the mismatch scores as they are derived from a total of just 200 different prints.

The vast range of scores from MATCH1 (from 10 ** 2 to 10 ** 43) would make nonsense of any attempt to plot histograms or density functions, and so the exponents alone were used for this purpose. [In effect each score was re-expressed as its logarithm (base 10).]

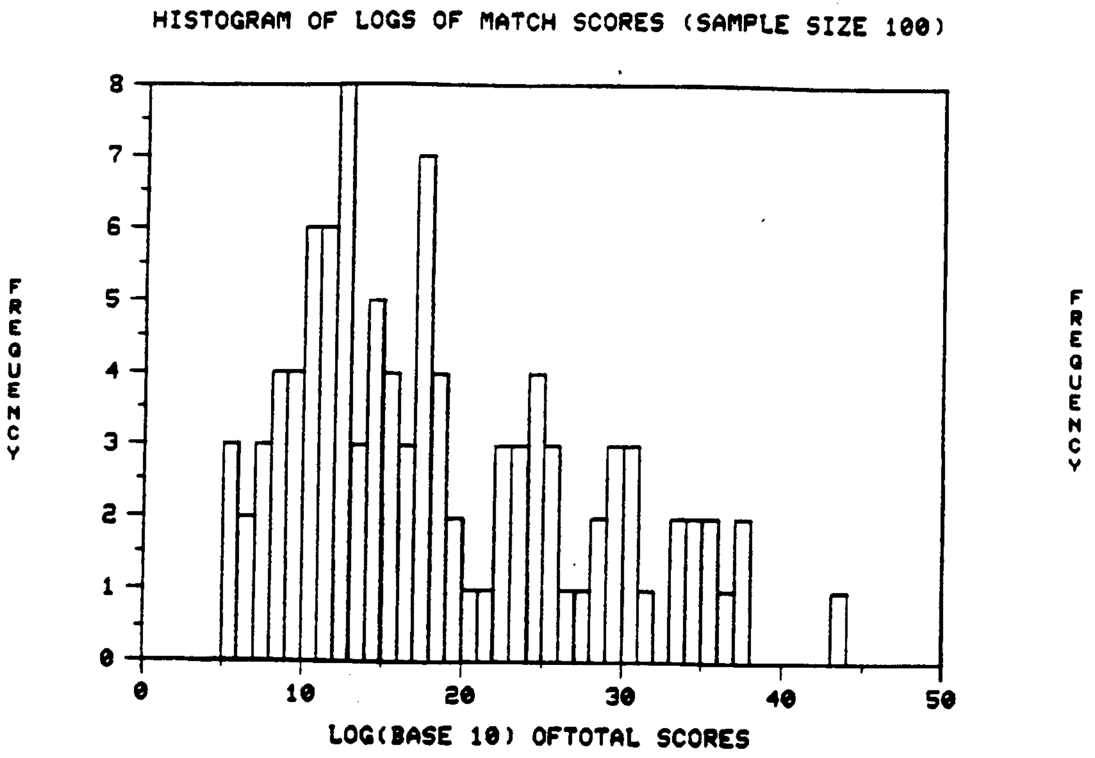
Histograms of the logs (base 10) of the match and mismatch scores from MATCH1 are shown in Figure 17.

The histograms were then converted into density functions and attempts made to 'fit' known pdf's to the plot. The most likely known pdf's (judging by the shape of the histograms and raw density plots) were the gamma, lognormal and Weibull distributions. Of these three, a lognormal was found to give a fairly good approximation to the observed mismatch score distribution--but no reasonable fit was found for the match score distribution.

The best lognormal fit for the observed mismatch score distribution is shown in Figure 18. Sadly the right hand tail (which is the crucial area) is the part of the distribution most badly fitted. Figure 18 shows an enlarged section of the right hand tail.

Fortunately one of the earliest amendments to MATCH1 (the 'score-normalization' procedures described in paragraph 3.4.3) altered the mismatch scores in such a way that a lognormal curve became a handsomely good fit (as was later confirmed by use of the 'Chi-square goodness of fit' test). However, it still did not improve the situation for match scores.

Even where familiar probability density functions could not be fitted, behavior in the tails of score distributions could be estimated by use of non-parametric density estimation techniques; (these are described in detail in reference 4) a non-parametric density function can be derived from the observations by summing a series of small 'kernal' distributions centered on each observed value. The sum of the kernal functions approximates the underlying distribution. Use of this technique would be laborious--and its value in drawing inferences from just 100 observations



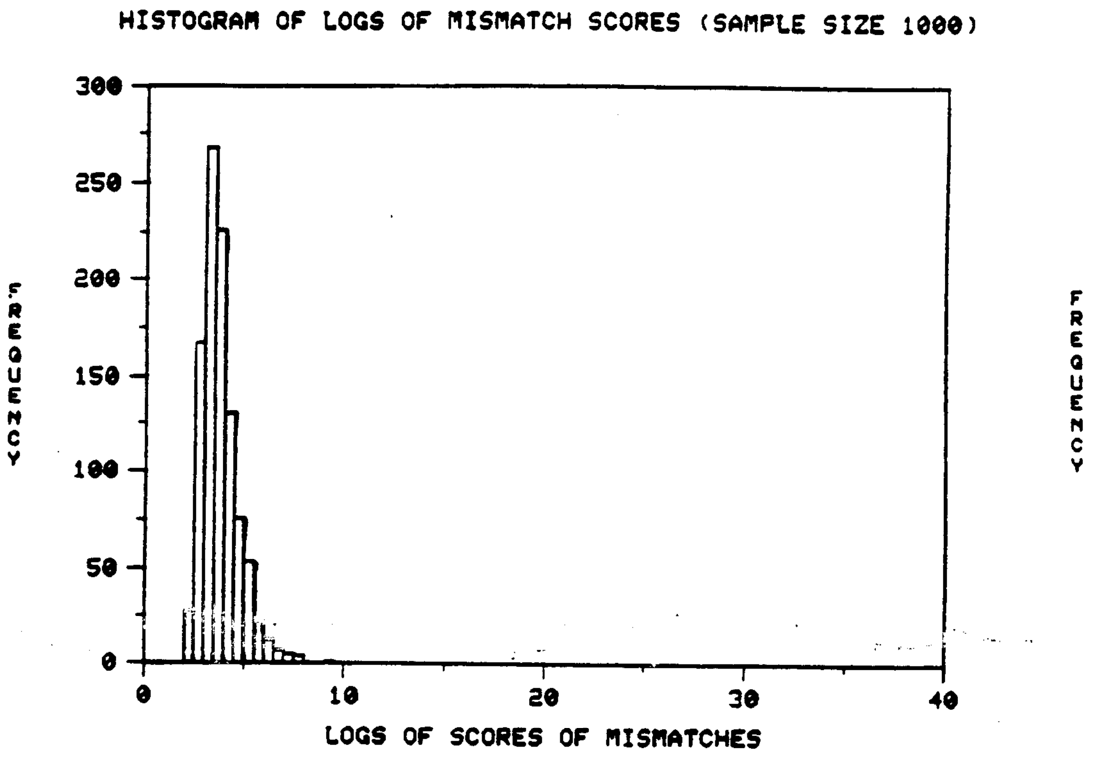
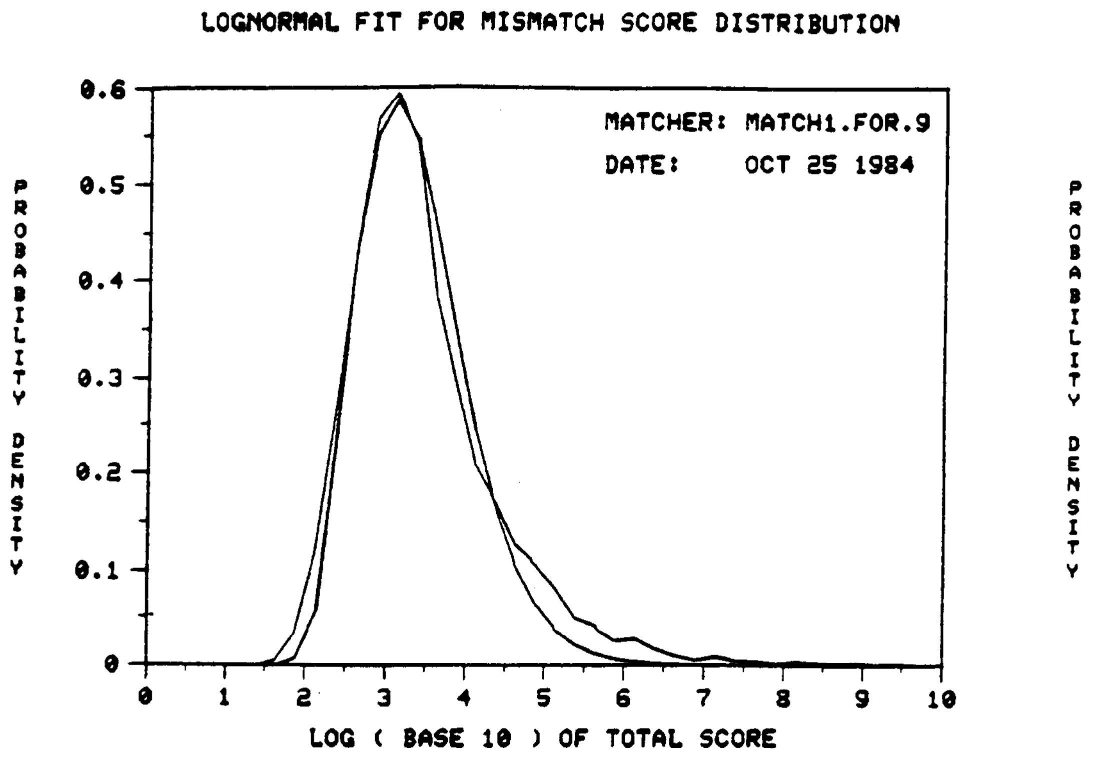


Figure 17. Histograms of match and mismatch scores (log(10)) from MATCH1



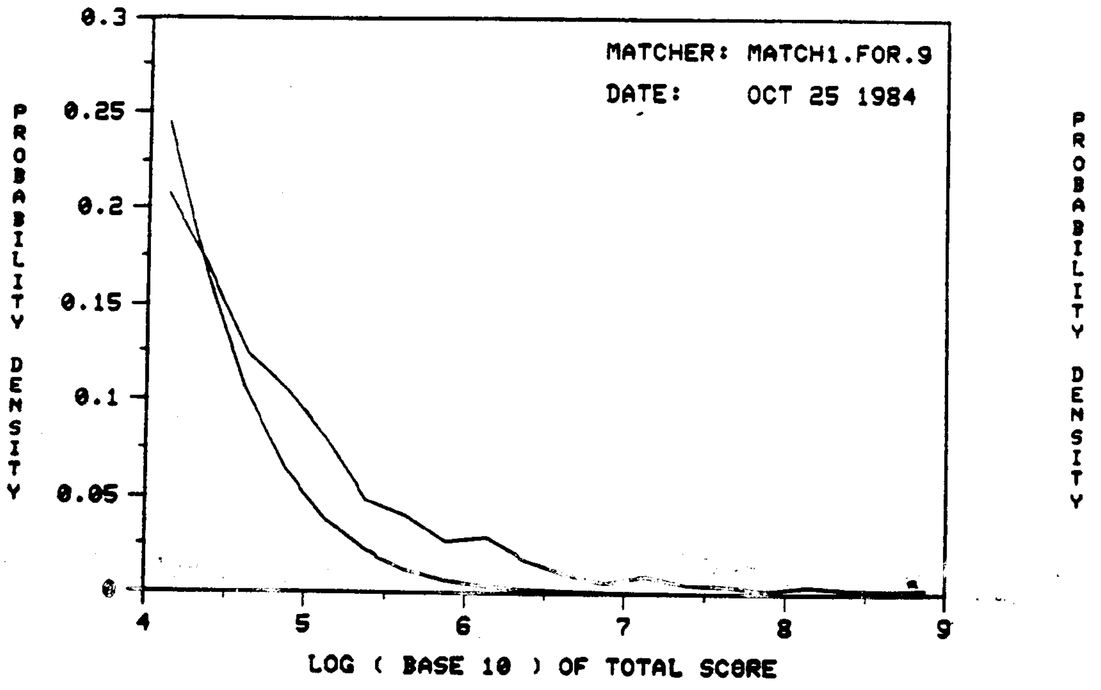


Figure 18. Lognormal fitting for mismatch score distributions with blown up portion showing the badly fitting right hand tail.

would depend somewhat on a fairly arbitrary choice of kernal shape. Use of this technique might have been essential, nevertheless, had not the mismatch scores behaved 'nicely' in turning out to be lognormally distributed.

3.3.1 Performance Measures Adopted

The performance measures eventually used to evaluate changes in the matching algorithms were:

(a) MR1 and LMR as already described.

(b) MINIMUM TOTAL ERROR (MTE)--see paragraph 3.3.2.

(c) The percentage of observed match scores exceeding the 99th percentile of the lognormal distribution that best 'fitted' the observed mismatch score distribution (P99). See paragraph 3.3.3.

(d) The percentage of observed match scores exceeding the 99.9th percentile of the 'fitted' lognormal mismatch distribution (P999). See paragraph 3.3.3.

3.3.2 Minimum Total Error (MTE)

Operational computerized fingerprint comparison schemes often employ a 'threshold' score; for a given search print, any fileprint scoring above the threshold in comparison is considered a likely candidate to be a true mate of the search print. Normally any file print scoring below the threshold would not be examined.

Such a system has two types of error--namely 'rejection' and 'substitution' errors (known variously as type 1 and 2 errors, or as 'misses' and 'false drops' in the fingerprint world). Substitution errors occur when a mismatch score exceeds the threshold. 'Rejection' errors occur when the true mate scores below the threshold.

For each test run with any particular matching algorithm, we can define the percentage substitution error to be the observed percentage of mismatches which scored above a given threshold value. Likewise define the percentage rejection error to be the percentage of match scores below it. These two percentages will vary as the threshold score is altered. The MINIMUM TOTAL ERROR is defined as the minimum value taken by the SUM of the percentage substitution error and percentage rejection errors--as the threshold score varies over the whole possible range.

The OPTIMUM 'CUTOFF' POINT is the threshold score for which the minimum total error is achieved. The optimum cutoff point corresponds exactly to the point at which the match and mismatch score density functions cross. See Figure 19 for a pictorial representation of this.

It is important to remember that minimum total error (MTE) is calculated from the observed match and mismatch scores only--not from any 'fitted' probability density functions.

3.3.3 <u>P99 and P999</u>

These are based on the lognormal probability density function best fitting the observed mismatch scores. They still depend on the raw MATCH scores, however, as no curve has fitted the match score distribution with any degree of reliability in the tails.

P99 and P999 are the observed percentage of match scores that exceed the 99th and 99.9th percentiles, respectively, of the lognormal curve fitted to the observed mismatch scores.

These two measures do, once again, have some practical significance for an operational system: there may well be a specified upper limit on the number of possible 'candidates' that can be manually examined for any one search inquiry (due to constraints on time and labour). Suppose one was not prepared to examine more than one print per thousand in the collection. Then the threshold would have to be set at least as high as the 99.9th percentile of the mismatch score distribution. Then the point of concern becomes what proportion of matches will be missed by selecting such a threshold score. P999 represents the percentage of matches that would not have been missed, had such a threshold been set during the particular test run.

Higher percentiles would be relevant to larger collections (i.e., the 99.99th and 99.999th percentiles) but to use these experimentally would be to stretch the reliability of the lognormal 'fitting' beyond reasonable limits.

3.4 <u>Description of MATCH2 Improvements</u>

MATCH2 used the same basic techniques as MATCH1, but several important modifications were made. They are described in paragraphs 3.4.1 to 3.4.4.

3.4.1 <u>Array Operations Made Integer Addition</u>

It was clear from the results of MATCH1 tests that it made better sense to use the logarithms of scores produced than the raw scores themselves. It would also make very good computing sense if all the array operations that involved multiplication of real numbers could be transformed into additive operations on integers.

All this 'good sense' is realized in MATCH2 by the use of 'log-based' integers in the array operation stages of the comparison algorithm (i.e., from the 'initial score matrix' onwards). Product evaluation is to be replaced by summation. It is a far quicker and simpler approach.

Theoretical Match And Mismatch Score Distributions

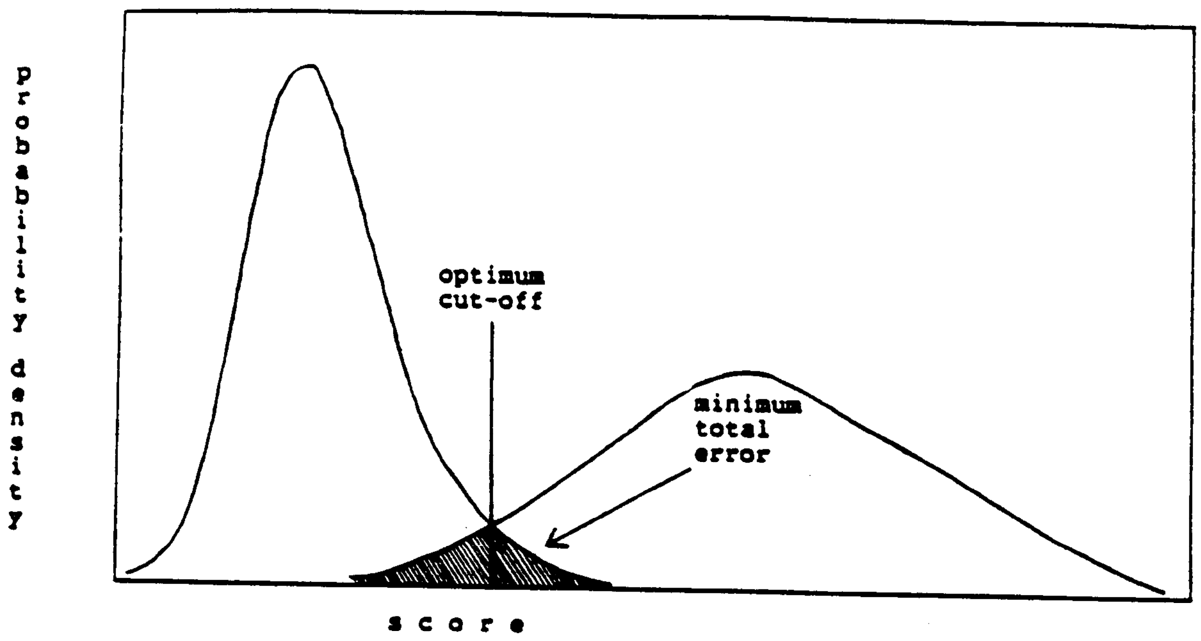


Figure 19. Theoretical match and mismatch score distributions showing the relationship of the optimum cutoff point to the density functions, and with the minimum total error area shaded in.

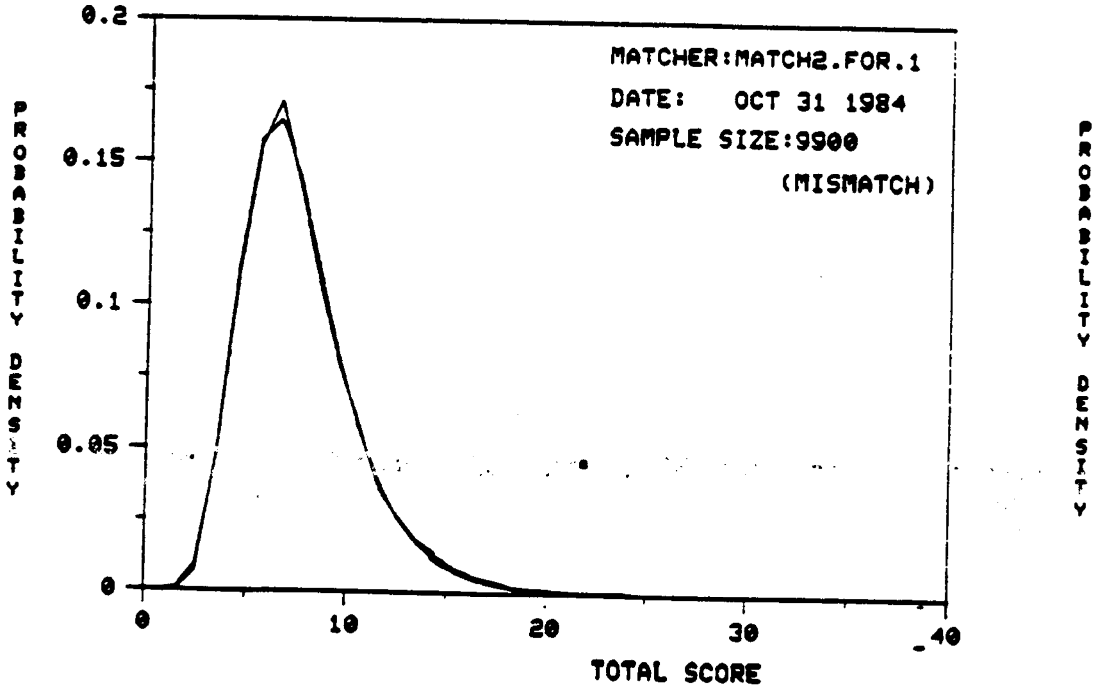


Figure 20. Lognormal fit for the mismatch scores from MATCH2.

The particular details required to effect this change are:

(a) In the score reference matrix S the 'exact match' scores [S(i,j,k,l): i=j] are now defined thus:

$$S(i,j,k,l) = \text{minimum}(\text{BOUND}, \text{INT}[10 \times -\text{Log}(P(j,k,l))])$$

where INT[...] means the integer part of [...]. The factor 10 appears to avoid all the exact match scores being either 0 or 1. The inclusion of this factor gives a reasonable spread of exact match scores based on code frequencies, despite the integer rounding. Typically these scores range from 1 to 15 or so. (Logs used are base 10).

(b) In the score reference matrix S all entries that were 1.0 are now changed to zero.

(c) In the score reference matrix S all entries that were zero are now set to an arbitrary negative number (-1) which will be recognized as 'no score' by the algorithm.

(d) The condensing step rules are appropriately altered to ADD the two digits together, or to take the non-negative one if only one is non-negative.

(e) Evaluation of any string product now becomes evaluation of the string sum. The ends of strings are marked by negative entries rather than by zeros.

3.4.2 <u>Final Score Evaluation</u>

Final score evaluation is made dependent on the single highest-scoring series in the condensed matrix rather than on the sum of all the different string 'products.' The 'best' series invariably scored so much higher than all the others that it rendered them almost insignificant. Ignoring strings other than the 'best' one is most unlikely to affect mate rankings at all. (It also obviates the need to take antilogs, add, and then reconvert to logs.)

The final score thus obtained is already logarithmic in nature. It is left in that form (i.e., the antilog is not taken) in order that the score distributions can be plotted and analyzed as already described.

3.4.3 Score Normalisation Procedure

Examination of the lower match scores from MATCH1 showed that they were often produced when the search prints had been of relatively low quality: some were badly scarred (producing many 'A's in their vectors) and others were not clear in parts (producing many 'B's). With high proportions of 'A's and 'B's present, and perhaps with a high proportion of ridges running 'out of sight'--large scores were just not possible, even if that vector had been faithfully reproduced within the file set.

The intention of score-normalization was to adjust scores from each comparison according to the amount of, or lack of, good information in the search print. The justification for such a procedure lies in this argument: if a search vector contains little information and a large part of it is found in a file vector, then this may be just as significant (in indicating a possible match) as had the search vector had plenty of information, only a little of which had appeared in the file vector. A mediocre score from a poor print is better than a mediocre score from a good print.

How then can the quantity of information in a search vector be measured? The method used in MATCH2 was to compare the search vector WITH ITSELF (using the matching algorithm) and see what score was obtained. That score is a very meaningful indication of the quality (i.e., rarity) and quantity of information in the search vector. It represents the sum of one continuous 'string' in the condensed matrix which covers the whole length of the search vector. It is, for that vector, the 'perfect' score. It is the maximum that could possibly be achieved by any file set vector compared to it.

All subsequent comparisons of that search vector with fileset vectors have their final scores expressed as a percentage of that 'perfect' score. Scores thus normalized appear as real numbers in the range 0 to 100. Real numbers are only used at this very last stage of the comparison process. The raw score (before normalization) was an integer.

This normalization cannot, of course, alter any rankings as all scores for any one search vector are expressed as percentages of the same 'perfect' score.

A notable effect of the change, however, is that it does make the overall distribution of mismatch scores appear to be genuinely lognormal. Figure 20 shows a lognormal curve superimposed on a raw density function of mismatch scores from MATCH2. This change (in the shape of the mismatch distribution) gives a good basis for use, hereafter, of the performance measures P99 and P999.

3.4.4 'Hopping' in the Condensed Matrix

Final score evaluation in MATCH2 depends on the single highest-scoring series found within the condensed matrix. One possible effect of this is that some matches may have produced very long strings which were broken up by isolated negative entries or ridge-shifts.

These string 'breaks' may have occurred as a result of two graphical mutations (one on either side of the generating line) that happened to affect the same ridge; that would cause an isolated negative entry in an otherwise continuously non-negative string in the condensed matrix. Alternatively 'ridge-shifting' (with its variety of causes) may have occurred; this will 'break' the string as a result of inclusion or deletion of a digit pair from one of the vectors under comparison. The result will be that part of the string in the condensed matrix is displaced either to the row above, or the row below (as shown in Figure 21).

An 'intelligent' algorithm would recognise this phenomenon, and would be able to put these broken strings together again (i.e., to evaluate their sums as if they had not been broken). To this end a 'HOPPING' section is introduced to the algorithm after formation of the condensed matrix, but before final score evaluation. A parameter "HOPS" is used--which indicates the maximum number of breaks which can be overlooked in evaluation of any one series score.

The score evaluation will then find the highest scoring string that can be found in the condensed matrix if up to "HOPS" number of breaks (of specified kind) can be ignored in each string.

The parameter is called "HOPS" because, in effect, the programme is allowed to hop from the right hand end of a series onto another point where that string is thought to be continuing. The permissible hops in the condensed matrix G are from any point g(r,s) to any one of these three points:

(a) G(r,s+2): this simply bypasses an isolated negative element in an otherwise continuously non-negative series.

(b) G(r+1,s+2) or G(r-1,s+1): these are the hops required to repair a string break caused by insertion or deletion of one digit pair from the search or file vector. (To see why these particular hops are appropriate one must study the effect of ridge shifting on the staggered search matrix C.)

These three particular 'hops' are not the only ones that could have been allowed; hopping from G(r,s) to either of G(r+1,s+3) or G(r-1,s+2) can be useful in repairing 'breaks' caused when the generating line passes the wrong side of a bifurcation. The selection of the three described above, however, has been found to be the most effective selection in aiding match scores without unnecessarily aiding mismatch scores.

These three different types of hop can be combined in any one string--although compounding hops simultaneously to make 'longer' hops is not allowed! If, for example, HOPS = 5, then the final score should represent the sum of the highest scoring string that can be found in the condensed matrix G, allowing up to five different hops per string, any one of which can be of any one of the three types described.

```
.........-1  1 -1 -1 -1  1  2 -1 -1  0  0 -1 -1 -1  0.............
......... 0  2  5  6 -1 -1  2 -1  0  0 -1  0 -1 -1 -1.............
.........75 30 50 10  5  6 45 30  5  3 -1  0 -1 -1 -1.............
.........-1 10 -1 -1 -1  3  0 -1  2 -1 -1 26 75 30 11.............
......... 1  0  3  4 -1 -1  2  1  0 -1 -1 -1  0  0 -1.............
```

Figure 21. Part of a condensed matrix showing a suitable 'hopping' place.

The calculation of such scores is accomplished by a further series of simple array operations. They are not described here. It is worth pointing out that the number of operations required for this step increases LINEARLY with the value of HOPS, and not exponentially as might have been expected. In the algorithm for MATCH2 the hopping section is one single iterative loop, which is repeated "HOPS" times. It is bypassed whenever "HOPS" is set at zero.

3.5.1 MATCH2 Performance on Loops

MATCH2 was run on TESTSET1 with a variety of different parameter sets. A table of results is shown in Figure 22. (It includes tests run on other sets of data.)

Particular observations that can be made from the results are:

(a) That 'MR1' was highest when "HOPS" = 0, i.e., when no hopping was allowed. (95% in first place on test 1).

(b) That 'LMR' was lowest (i.e., best) with "HOPS" = 1 (lowest mate rank was 3 on test 2).

(c) That more than one HOP seemed to worsen the results by boosting mismatch scores too much (presumably joining up bits of disconnected noise into high-scoring series).

(d) That the rankings for MATCH2 in test 1 were not significantly different from those for MATCH1 in test 6. This shows that the conversion from real number multiplication to log-based integer addition preserved the discriminating power of the algorithm--moreover that use of the highest scoring series only, as opposed to summing all the products, had little practical effect.

(e) That the four different performance indicators (MTE, P99, P999, MR1) are fairly consistent (with each other) in their appraisal of performance.

[Tests were also conducted with a variety of different 'condensing' rules, i.e., rules for forming the condensed matrix from the filtered score matrix. None were found that gave better results than the rule originally adopted, and so it was retained.]

Parameters fixed throughout these tests are: Bound=15 Close match score=0
Band=2 Vector length=82

Test	Testset	Testset Size	Hops	Maxshift	MR1	LMR	Optimum Cut-off	MTE %	P99 %	P999 %
1	1	100	0	2	95	8	17.54	4.13	97.00	87.00
2	1	100	1	2	93	3	19.95	5.13	95.00	84.00
3	1	100	2	2	95	10	24.93	5.44	93.00	86.00
4	1	100	3	2	95	22	27.76	6.83	92.00	83.00
5	2	53	0	2	52	16	15.66	3.59	94.34	86.79
6	2	53	1	2	52	9	21.42	4.90	96.23	88.68
7	2	53	2	2	52	10	21.46	5.95	94.34	86.79
8	2	53	3	2	52	8	25.21	5.59	88.68	86.79
13	3	23	0	5	23	1	16.52	4.55	95.65	82.61
14	3	23	1	5	23	1	20.76	5.93	91.30	86.96
15	3	23	2	5	22	2	26.48	3.95	95.65	73.91
16	3	23	1	2	21	2	20.76	8.10	86.96	82.61
17	3	23	1	10	23	1	20.76	8.50	91.30	82.61

Table to show MATCH2 results on various test sets and with various parameters. (Tests 9 - 12 were experiments using different condensing rules.)

Figure 22. Summary of MATCH2 performance--tests 1-17.

3.5.2 MATCH2 Performance With 'Whorls'

TESTSET2 comprised core-centred vectors from 53 pairs of mated whorls--and it had the same form as TESTSET1. The precise location of the 'core' of the whorl was determined by a simple adaptation of the rules used for loops. A sample whorl tracing generated during the coding process is shown in Figure 23.

The performance of MATCH2 when applied to TESTSET2 was very similar to its performance with loops (TESTSET1). Again some of the performance measures suggested the best value for HOPS was 1; others suggested the best value was 0. (Refer to Figure 22).

All but one of the mates were ranked 1st in every test conducted on TESTSET2. The lowest value achieved for the minimum total error was 3.59% (which comprised 1.71% substitution error and 1.89% rejection error around an optimum cutoff point of 15.66).

3.5.3 MATCH2 Performance With 'Plain Arches'

A few 'plain arch' prints were available--and 23 mated pairs were selected. These were to form TESTSET3.

The method of 'placing a line' on an arch is quite different to that used for both loops and whorls. There is no central reference point (such as a core). Instead the print is oriented, once again, so that the flexion crease appears horizontal. Then a FLEXIBLE line is drawn vertically through successive summits of the ridges--as shown in Figure 24. The line starts at the lowest visible ridge above the flexion crease and follows the 'summit' route to the top of the available picture. The digit pairs for each ridge intersection point were formed by looking 'left' and 'right' along the ridges, rather than 'up' and 'down.' The same set of event codes were used. The digit pairs were ordered from the 'bottom up'--i.e. in the order of the numbered intersection points shown in Figure 24. The resulting vectors varied in length, and were padded up to the standard length of 82 digits (with 'FF's). On this occasion, however, the padding was a single-ended operation rather than double-ended, because the vector was not generated around any fixed central reference point (as the vectors for loops and whorls had been).

Because of this lack of any central referencing a larger value for the parameter MAXSHIFT is anticipated--as comparative ridge shifting of the whole vector (caused by changes in the starting point of the generating line) may well be more severe.

The performance of MATCH2 on TESTSET3 is shown in Figure 22 (tests 13-17). All 23 mates were ranked first when MAXSHIFT was 5 or more (tests 13 and 14) indicating that relative mate-vector alignment had not been 'out' by more than five ridges in any case.

Again the various performance measures favour either HOPS=0 or HOPS=1.

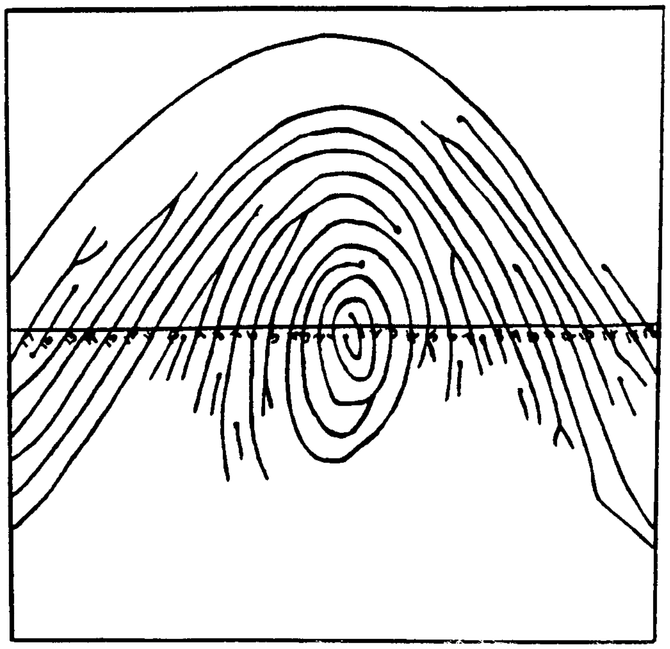


Figure 23. Sample tracing of whorl generated during coding process.

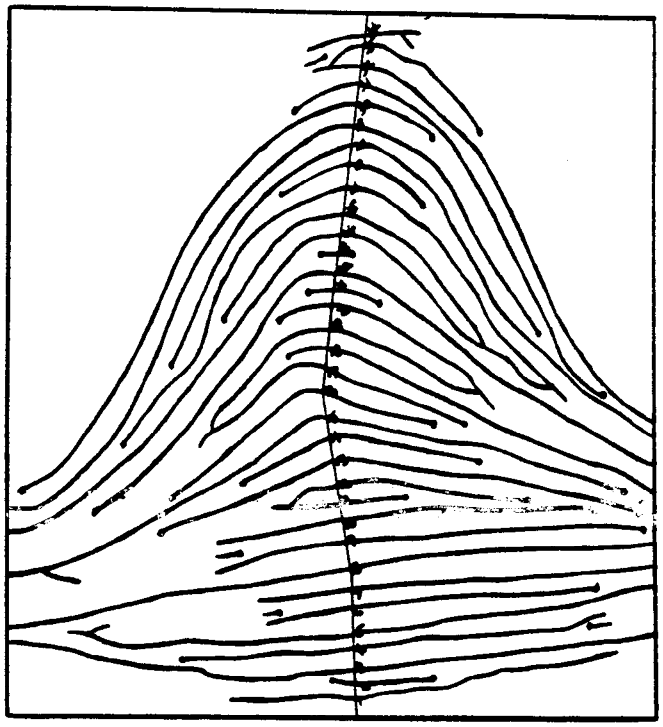


Figure 24. Sample tracing of plain arch generated during coding process.

3.6 Description of MATCH3 Improvements

The score normalisation procedure described in paragraph 3.4.3 adjusted each comparison score by reference to the amount of information contained in the search vector. That 'amount of information' was determined by self-matching the search vector to give a 'perfect' score; subsequent comparison scores involving that search vector were expressed as a percentage of that 'perfect' score.

The one thing that such a score normalisation scheme clearly fails to do is to take account of the amount of information in the FILE vector.

MATCH3 was an attempt to redress the balance, and to include a second correction factor based on the file vector. The amount of information in the file vector was measured just as it had been for the search vector in MATCH2--by self-matching. This meant that another preliminary stage, to be executed before any search vectors were processed, was introduced to the algorithm. This preliminary step was to self-match each file vector in turn and record the 'perfect' score obtained in each case. (This would not need to be done every time a search was conducted; each file vector would have its 'self-mate' score calculated just once when it was introduced to the collection; the self-mate score would then be stored along with the file vector, and it would be referenced each time that file vector was used in comparison. A file vector's 'self-mate' score would have to be recalculated only when the scoring system, for that file, was reappraised by a new 'fileset analysis'.)

Suppose there were n vectors in the file--called B[1] to B[n]. Suppose perfect scores obtained for each by self-matching were called R(i), i=1,n. Let the calculated mean of the R(i)'s be Rm. Suppose, further, that a particular search vector A[j] gave a 'perfect' self-match score of Q(j), and that A[j] compared with B[i] gave a raw score (i.e., not normalised in any way) of T(ij).

Then the normalisation described in paragraph 3.4.3 gave a final normalised score of:

$$\frac{T(ij) \times 100}{Q(j)}$$

which is a 'percentage.'

Two different ways of incroporating R(i) into this normalisation formula were tried. MATCH3 version 1 used the formula:

$$\frac{T(ij) \times 100 \times Rm}{Q(j) \times R(i)}$$

(where the ratio of R(j) to the mean file Perfect score (Rm) is used as the second correction factor.)

MATCH3 version 2 used the formula:

$$\frac{T(ij) \times 100}{SQR(Q(j) \times R(i))}$$

Both these formulae give final 'percentage' scores, although the first one is capable of producing scores over 100% (which suggests 'over correction'). The second cannot produce scores over 100% as T(ij) cannot possibly exceed either Q(j) or R(i).

3.7 Performance of MATCH3--Versions 1 and 2

The normalisation procedure used in MATCH3 version 1 seemed to over-compensate for print quality. It succeeded in bringing mate ranks for poor quality prints to the top (i.e., to mate rank 1)--but it also boosted some mismatch scores involving poor quality prints so that they scored higher than matches involving good prints.

The approach used in MATCH3 version 2 seemed to be a more balanced one altogether, and performance was improved by its use. A complete table of results using MATCH3 versions 1 and 2 is shown in figure 25.

The best values achieved for MTE were:

3.48% on TESTSET1 (Loops)--test no. 3.
3.08% on TESTSET2 (Whorls)--test no. 5.
1.78% on TESTSET3 (Arches)--test no. 9.

With MATCH3 version 2 P999 values above 90% were achieved on all three testsets.

However, any algorithm improvements that would raise MR1 above 95% (i.e., put the remaining five mate-scores into top place) had,thus far, eluded us.

4. THE INTRODUCTION OF DISTANCE MEASURES

4.1 Motivation For So Doing

Despite the various improvements, described in Chapter 3, designed to improve discrimination between matches and mismatches--it was noticeable that some mismatched pairs consistently scored high, and did so whichever matching algorithm was used.

Examination of some of these high-scoring mismatched vector pairs showed that, on occasions, they really did have very similar sequences within them. For example here are short sections of the vectors representing two different prints (from different fingers):

Card 32, set A, finger 2	.......73 71 22 74 41 21 81......
Card 21, set B, finger 4	.......73 71 23 73 41 21 83......

Parameters fixed throughout these tests are: Bound=15 Close match score=0
Band=2 Vector length=82
Maxshift=2

Test	Match3 Version	Testset	Testset Size	Hops	MR1	LMR	Optimum Cut-off	MTE %	P99 %	P999 %
1	1	1	100	0	94	6	16.06	5.17	95.00	87.00
2	1	1	100	1	90	10	24.15	6.30	94.00	80.00
3	2	1	100	0	95	4	16.67	3.48	97.00	91.00
4	2	1	100	1	92	7	22.81	4.88	96.00	88.00
5	2	2	53	0	51	14	16.38	3.08	98.11	92.45
6	2	2	53	1	52	9	24.32	4.03	96.23	94.34
7	2	3	23	0	23	1	17.41	2.57	95.65	91.30
8	2	3	23	1	23	1	21.88	3.75	95.65	91.30
9	2	3	23	2	23	1	27.90	1.78	95.65	91.30

Figure 25. Table of MATCH3 performance--showing which of the two versions was used for each test.

In comparison of these two vectors these substrings scored very highly indeed (approximately 25% of the 'perfect' score for the search vector).

The actual prints represented by such high scoring mismatches were scrutinised to see if they really were so similar. Topologically speaking they were indeed very similar. However, they could easily be told apart by the very crudest of spatial measurements.

It was hoped, therefore, that incorporation of some single spatial measure into the topological coding scheme could be used to 'break up' these high scoring mismatch series.

Recognition of this need, is perhaps, recognition that topology ALONE is not quite 'strong enough.' Introduction of some sort of crude distance measure is not reversion to a 'spatial' approach--as will be seen. It is the 'taking of a little help' from distance measurement to enhance the performance of a topology based system.

4.2.1 Methods of Coding and Recording Distance

The 'measuring' scheme adopted is quick and simple. It gives one hexadecimal integer as a 'distance measure' for each hexadecimal event-code.

The measurement was performed on the ridge tracings generated during the original coding process. The distance was measured from each 'ridge event' to the generating line. The measuring was not 'as the crow flies' but rather 'as the insect walks' (assuming that insects 'walk along' ridges). Distances are measured along the relevant ridge from generating line to ridge-event. A FLEXIBLE ruler is therefore required for the manual operation!

The distance was measured (on the 10 x enlargements) in centimetres, and was then rounded down to the nearest integer, and an upper bound of 15 imposed. On the actual print, therefore, the distance measures would represent the distance, measured along ridges, from generating line to ridge-event, rounded down to the nearest millimetre. Thus the only possible distance measures are the integers 0,1,2,...15.

If the ridge-event codes were any of the set [0,A or B] then the corresponding distance measures were set to a default value of 15. These codes 0 ('out of sight'), A ('scarred tissue') and B ('unclear') cannot really have meaningful distance measures associated with them; all the other event codes can.

Restriction to hexadecimal distance measures does mean that an event code, together with its distance measure, can be stored in 1 byte of memory. The storage requirement for each print code is therefore 82 bytes.

4.2.2 The New Databases

All of the TESTSETS were reformulated to incorporate one hexadecimal distance measure for every event code in the original vector. A single print was thus represented by an array (size 82 x 2) rather than by a vector. (The 'ends' were padded with 'F's in the same way as the vector had been.)

TESTSET4 corresponds to TESTSET1 (Ulnar Loops), but with distance measures inserted.

TESTSET5 corresponds to TESTSET2 (Whorls), but with distance measures inserted. TESTSET5 was also expanded from 53 pairs to 100 pairs of whorls. It was found that the coding of a single fingerprint, manually, to include the required distance measures took approximately 12 minutes. (It had been 7 minutes without distance measures).

TESTSET6 corresponds, in the same way, to TESTSET3 (Plain arches).

4.3 The Three Tests to be Applied

During a print comparison (which is now an array comparison rather than a vector comparison) the distance measures will be used in the application of three different tests. All three tests are applied to the initial score matrix in such a way as to reduce (to -1) any positive initial scores that the distance measure tests indicate ought to be so reduced. This will occur if the distance measure tests show that the matched event codes (which gave that positive value) are from 'events' that are not roughly in the same area (spatially) of their respective prints.

These three tests are described in the next three paragraphs.

4.3.1 Absolute Distance Test

Before the matching algorithm accepts an event code in a file print array as possibly being correctly 'matched' with an event code in the search print array--it now has to ask not only 'are the event codes the same?' but also a number of questions relating to their distance measures. The first is called the ABSOLUTE DISTANCE TEST:

'Is the distance between the generating line and the ridge-event adequately preserved (i.e., is it preserved within a given tolerance)?'

The tolerance allowed become a parameter of the programme and is called the ABSOLUTE DISTANCE TOLERANCE (ADT).

4.3.2 Differential Distance Test

If two 'events' from adjacent ridges on the file print seem to match two events on adjacent ridges on the search print (where, in each case, both events lie on the same side of the generating line) then we should ask the question:

'Is the DIFFERENCE in their distance measures adequately preserved?'

The tolerance allowed in this test is another parameter, called the DIFFERENTIAL DISTANCE TOLERANCE (DDT).

The difference between distance measures on adjacent ridges, looking in the same direction (i.e., same side of the generating line) is a measure of the distance between the two events seen on those ridges--and is independent (except for rounding errors) of the exact position of the generating line. If this 'differential distance' is not preserved then one, or other, of the two events cannot be correctly matched; they cannot BOTH be right.

4.3.3 Summed Distance Test

If two 'events' on the same ridge (i.e., both halves of a digit pair) seem to be matched from search to file print, then the SUM of their distance measures should be preserved (within certain tolerance). That SUM represents the total distance, along the relevant ridge, from one event to the other. The measures are added because the events are appearing on opposite sides of the generating line. Again, if this sum is not preserved then one event,or the other, is not correctly matched; they cannot both be.

The tolerance allowed in this case is called the SUMMED DISTANCE TOLERANCE (SDT).

4.4.1 Building These Tests Into The Algorithm--MATCH4

MATCH4 incorporates these three tests into the comparison algorithm. It operates on datasets having distance measures included. The bulk of the algorithm is completely unaffected--operating on the topological event codes only, and ignoring the distance measures.

The distance tests are applied as the first filtration step for the INITIAL SCORE MATRIX E--before the filtering for dependent pairs. (See Chapter 2 for the sequence of phases in comparison.) The manner of their application (briefly) is as follows:

(a) ABSOLUTE DISTANCE TEST: every positive element, E(r,s) of the initial score matrix E is derived by comparison of C(r,s) and D(r,s)--elements of the search and file matrices. Each element of C now has a corresponding distance measure, as C is composed of several staggered repetitions of the search vector A. Likewise each element of D has a related distance measure, being derived from the file vector.

We call these related distance measures C'(r,s) and D'(r,s) respectively.

The rule for the absolute distance test is:

If MOD [C'(r,s)-D'(r,s)] exceeds ADT, then change E(r,s) to -1.

(b) DIFFERENTIAL DISTANCE TEST: whenever E(r,s) and E(r,s+2) are positive elements within E then

If MOD [(C'(r,s)-C'(r,s+2)) - (D'(r,s)-D'(r,s+2))] exceeds DDT then change one of E(r,s) and E(r,s+2) to -1. (Which of the two is reduced depends on other neighboring elements within E.)

(c) SUMMED DISTANCE TEST: whenever E(r,2s) and E(r,2s-1) are both positive elements within E, then

If MOD [(C'(r,2s)+C'(r,2s-1) - (D'(r,2s)+D'(r,2s-1)] exceeds SDT, then one of E(r,2s) and E(r,2s-1) is reduced to -1. (In this case the largest of the two is reduced.)

4.4.2 Omission of Distance Tests

The algorithm was prepared so that any or all of the three distance tests could be omitted by entering the appropriate parameter value as '99.' This was an essential provision if the effect of each test was to be evaluated. Consequently where '99' appears in the tables of results it shows that a test has not been applied.

4.5 Performance of MATCH4 on Ulnar Loops (TESTSET4)

The inclusion of these simple digital distance measures, and the related distance tests had the most startling effect on the performance of the matching algorithm. The previous 'best performance' on TESTSET1 had been test no. 3 with MATCH3 (see figure 25)--producing performance measures:

MTE (Minimum total error)	= 3.48%
P99	= 97.0%
P999	= 91.0%
MR1 (% mates ranked 1st)	= 95.0%

The best performance with MATCH4 on the same set of ulnar loops (now TESTSET4, with the distance measures) was that given in test no. 34 (see figure 26). This time the performance measures indicated 'close to perfect' discrimination between matches and mismatches. They were:

MTE	= 0.05%
P99	= 100%
P999	= 100%
MR1	= 100%

Figure 26 gives a result summary for MATCH4 on TESTSET4. There are a number of particular observations that should be made from this table:

(a) From tests 1 to 5 it can be seen that all three distance tests helped performance, and that the optimum value for all three parameters (ADT, DDT and SDT) was 1. One would expect these parameters (tolerances) to be AT LEAST 1 just because of the effect of rounding the distance measures down to integers (see paragraph 4.2.1). The fact that they can be set as low as 1 without detrimental effect on mate scores suggests that the distance measures (measuring along ridges) are surprisingly robust.

(b) For all previous algorithms (MATCH1 to MATCH3) it had been better policy not to recognise 'close matches'--consequently close match scores had been set at -1 (or zero, in the case of MATCH1). The predominant effect of scoring positively for possible topological mutations had been to boost mismatch scores, worsening the discriminatory performance of the algorithm. (See paragraph 2.3.3(c)). However, once the distance tests are applied, results are IMPROVED by positively scoring close matches--the optimum value for close match scores being +1.

Parameters fixed throughout these tests are: Vector length=82 Band=2

Test	Hops	Close Match	ADT	DDT	SDT	Bound	MR1	LMR	Optimum Cut-off	MTE %	P99 %	P999 %
Tests 1-5 examine the effect of the distance measures												
1	0	-1	2	99	99	15	98	3	13.47	0.70	100	96
2	0	-1	2	1	99	15	98	3	13.47	0.48	100	97
3	0	-1	2	1	1	15	97	3	13.47	0.41	100	97
4	0	-1	1	1	1	15	99	2	13.47	0.22	100	98
5	0	-1	1	0	0	15	98	5	10.87	1.70	99	93
Tests 6-16 examine the effects of various HOPS and Close Match Scores												
6	1	-1	1	1	1	15	100	1	15.60	0.39	100	98
7	2	-1	1	1	1	15	97	2	18.05	0.31	100	96
8	1	0	1	1	1	15	99	2	18.66	0.09	100	100
9	1	1	1	1	1	15	100	1	19.44	0.08	100	100
10	1	2	1	1	1	15	100	1	20.02	0.09	100	100
11	2	0	1	1	1	15	98	2	20.77	0.13	100	99
12	2	1	1	1	1	15	98	2	21.48	0.13	100	99
13	2	2	1	1	1	15	99	2	22.18	0.13	100	99
14	0	0	1	1	1	15	100	1	13.72	0.20	100	98
15	0	1	1	1	1	15	100	1	14.44	0.15	100	99
16	0	2	1	1	1	15	100	1	15.40	0.09	100	100
Tests 34-36 examine the effects of BOUND												
34	1	1	1	1	1	5	100	1	21.67	0.05	100	100
35	1	1	1	1	1	3	100	1	21.77	0.07	100	100
36	1	1	1	1	1	8	100	1	19.96	0.07	100	100

Figure 26. Table of MATCH4 performance (Ulnar loops).

(This is a positive, but not very significant, weighting for close matches.) A reasonable inference to draw from this observation would be that any high scoring series inadvertently formed in the score matrices of mismatches are adequately broken up by the distance tests. The predominant effect of recognising, and positively scoring, possible toplogical mutations now becomes that of boosting match scores--as had been originally intended.

It is important to note that the mismatch score distribution produced by MATCH4 is still LOGNORMAL (see figure 27).. It is also interesting to see just how far down the right hand tail is the appearance of the lowest of the observed match scores (21.62). In fact with 9900 mismatch scores, and 100 match scores output from the test, a MTE value of 0.05% means that just 5 of the 9900 mismatch scores exceeded the lowest of the match scores.

4.6 MATCH4 Performance on Whorls and Arches

(a) WHORLS: MATCH4 was tested against TESTSET5 (100 pairs of whorls, with distance measures included) and the summary of results is given in figure 28, tests 56-58. Once again all 100 mates were ranked 1st, and P99 and P999 values of 100% were obtained in tests 57 and 58. The lowest value for MTE was 0.1% (test 57).

(b) PLAIN ARCHES: the algorithm was also applied to TESTSET6 (23 pairs of mated plain arches) and the results summary is given in the upper portion of figure 28. All four performance measures registered 'perfect' performance on this occasion--but 23 print pairs could be said to form a significantly smaller database than one hundred pairs. It was heartening, nevertheless, to see that (in test 54) the highest mismatch score (of 506 observations) was 20.81 while the lowest match score (23 observations) was 27.11.

4.7 Use of Shortened Vectors--Results (Loops)

Tests 17 to 33 (see figure 29) used progressively less and less information from the database TESTSET 1: the purpose of the experiment was to see how rapidly performance dropped off as the print codes were pruned more and more severely, and thereby to determine just how much information was actually needed (from each single print) to form a reliable basis for identification.

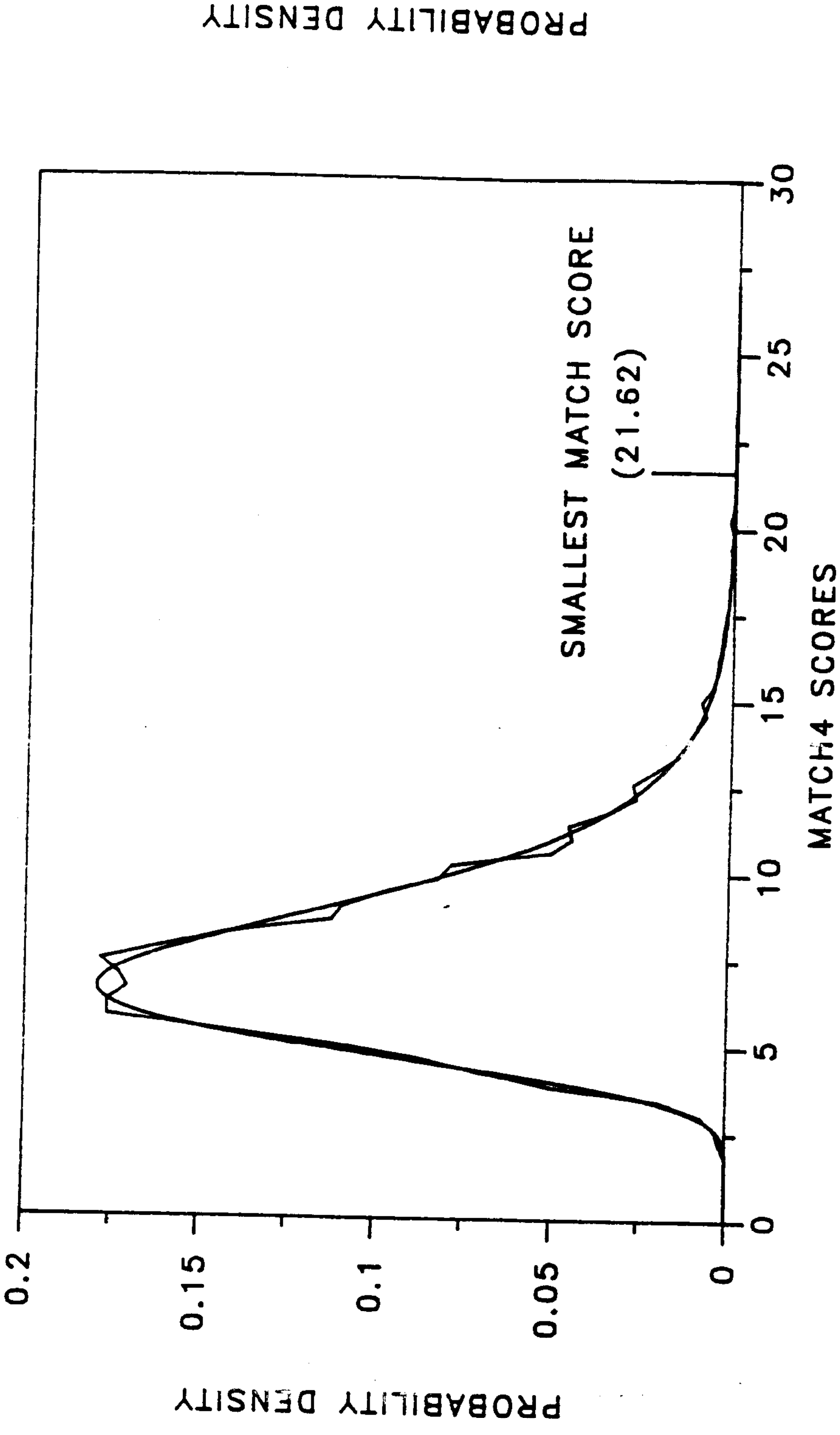


Figure 27. Lognormal fit for MATCH4.dat;9 with lowest match score shown.

Parameters fixed throughout these tests are: Close match score=1
Band=2 Vector length=82
Absolute distance tolerance=1
Differential distance tolerance=1
Summed distance tolerance=1

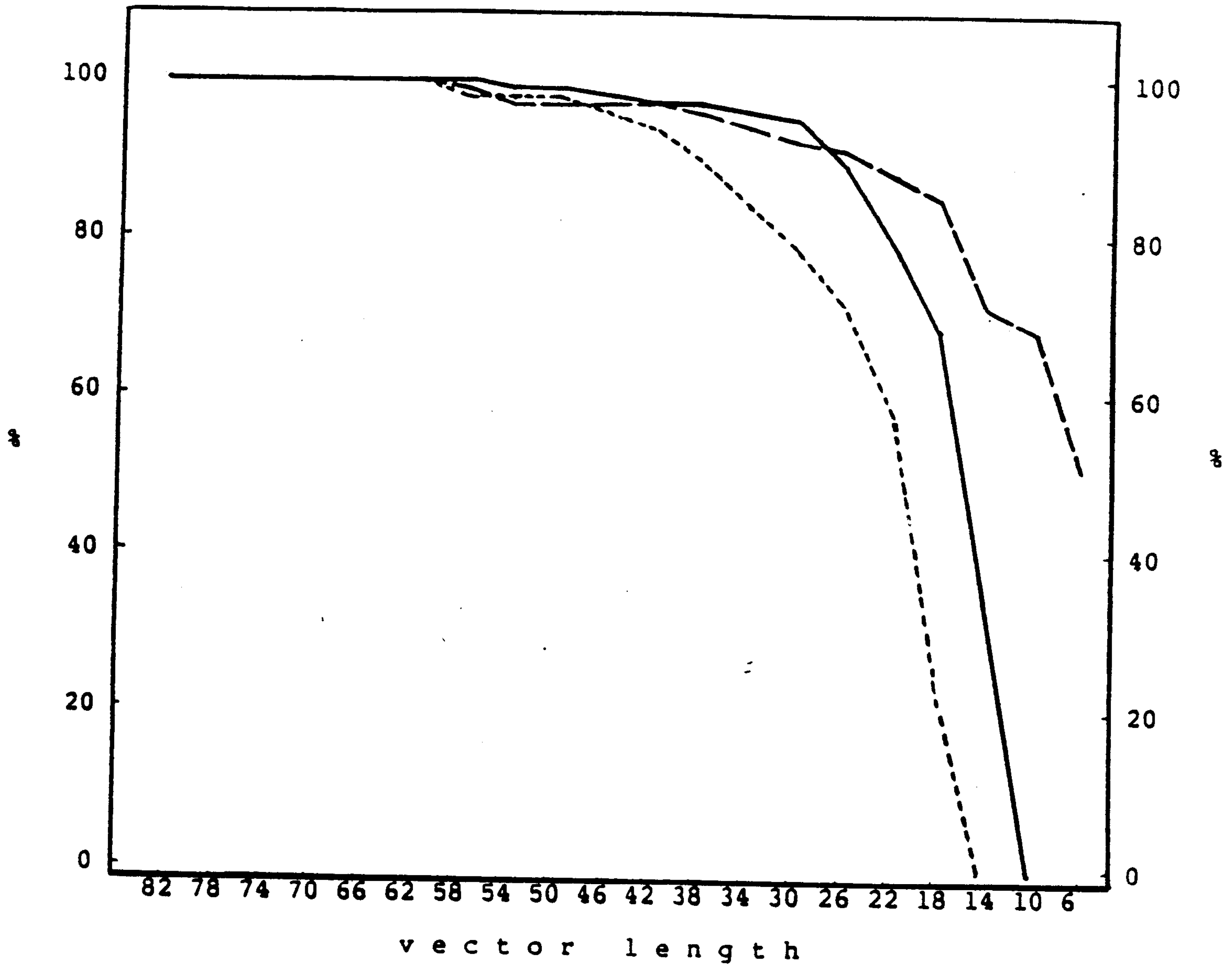
Test	Pattern Type	HOPS	Maxshift	Bound	MR1	LMR	Optimum Cut-off	MTE %	P99 %	P999 %
53	Arches	1	5	5	23	1	26.68	0.00	100	100
54	Arches	2	5	5	23	1	27.10	0.00	100	100
55	Arches	0	5	5	23	1	19.17	0.00	100	100
56	Whorls	1	2	5	100	1	18.15	0.11	100	99
57	Whorls	1	2	20	100	1	17.80	0.10	100	100
58	Whorls	1	2	15	100	1	17.80	0.11	100	100

Figure 28. Table of MATCH4 performance on Whorls and Arches.

Parameters fixed throughout these tests are: Bound=15 Close match score=1
Band=2 Hops=1
Absolute distance tolerance=1
Differential distance tolerance=1
Summed distance tolerance=1

Test	Vector Length	MR1	LMR	Optimum Cut-off	MTE %	P99 %	P999 %
9	82	100	1	19.44	0.08	100	100
17	70	100	1	19.06	0.09	100	100
18	66	100	1	19.75	0.09	100	100
19	62	100	1	20.03	0.09	100	100
20	58	99	2	19.21	0.20	100	98
21	54	97	3	19.54	1.24	99	98
22	50	97	4	18.24	1.74	99	98
23	46	97	10	23.80	2.12	98	96
24	42	97	18	20.76	2.80	97	94
25	38	96	29	27.27	3.15	97	90
26	34	94	26	23.94	3.95	96	84
27	30	92	30	26.08	3.99	95	79
28	26	91	61	28.01	5.69	89	72
29	22	88	77	28.17	7.78	79	58
30	18	85	78	32.50	11.99	68	25
31	14	71	67	35.90	17.22	35	0
32	10	68	74	41.02	19.81	[illegible]	0
33	6	51	96	34.17	32.59	0	0

Figure 29. Table of MATCH4 tests on TESTSET1 with shortening of the vectors used.



Graph to show how MR1(——),P99(——) and P999(----) vary with vector length.

Figure 30. Graph showing how MR1, P99 and P999 vary with vector length reductions.

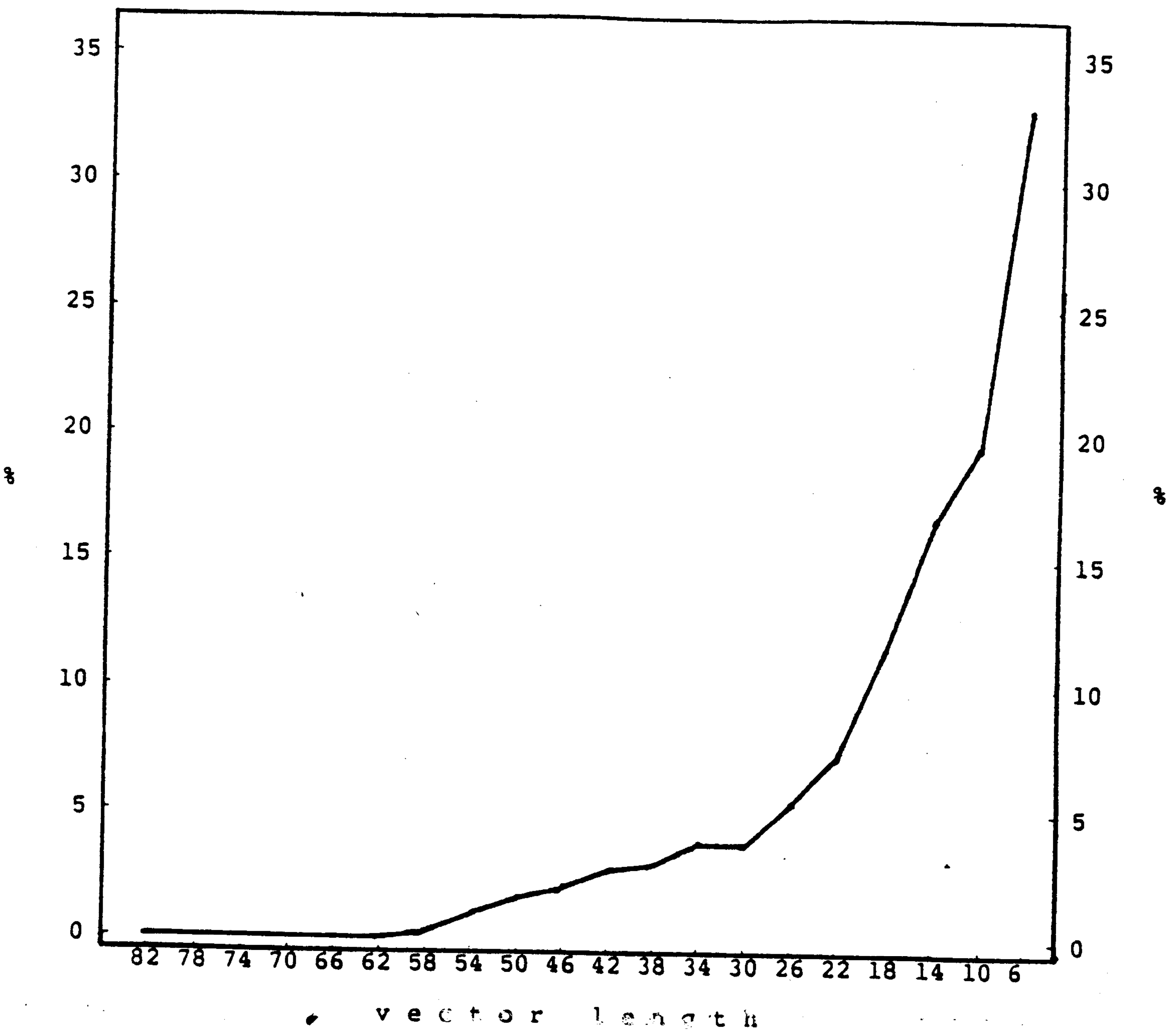


Figure 31. Graph showing how MTE varies with vector length reductions.

The standard size of a code array in TESTSET1 is 82 x 2. The length (82) was progressively shortened by symmetrical pruning (i.e., off both ends)--leaving a shorter and shorter, but still core-centred, array. Figure 30 shows how the performance measures MR1, P99 and P999 vary with array length. Figure 31 shows how the performance measure MTE varies with array length.

It is worthwhile to note from these results that:

(a) the array length can be reduced from 82 to 62 with virtually no worsening of the results at all.

(b) P99 and P999 only dip below 90% at lengths 26 and 34 respectively.

(c) the percentage of mates ranked in first place (MR1) still exceeds 90% when the length of array used is 26.

(d) the percentage of mates ranked first exceeds 50% even when the shortest arrays (of length 6) are used.

5. COMPARISON OF TOPOLOGICAL AND SPATIAL APPROACHES

5.1 Aims and Method of the Comparison

A direct comparison of performance between the topology-based algorithm, MATCH4, and an algorithm using the conventional (spatial) techniques was sought. The algorithm M82 was selected to represent the spatial approach, and both algorithms were run on the same set of prints. (These were the 100 pairs of mated ulnar loops that had been used for TESTSET4.)

The M82 algorithm is one of the most reliable spatial matching algorithms that has been developed. It recognises, and corrects for, translational errors--and it is sophisticated enough to apply tensor corrections for 'stretching'. It was developed at the National Bureau of Standards and is used by the FBI. A full description of it is given in reference 5. The version of the algorithm used for the test was written in FORTRAN and run on a VAX-11/780. (MATCH4 was also written in FORTRAN and run on exactly the same machine.)

The particular fingerprints comprising the selected TESTSET were read by the FBI's automatic scanning system--and the cartesian coordinates (X,Y) of each detected minutia, together with an angle (theta) for the ridge flow direction at each minutia, were extracted. That data was fed into the VAX-11/780 as the representation of the 100 pairs of mated ulnar loops--in the form required by the M82 algorithm.

5.2 M82 and MATCH4 Performance

The M82 output scores were analysed in exactly the same way as the MATCH4 scores had been--and the performance measures used were MR1, LMR and MTE ('Mates ranked 1st', 'Lowest mate rank', and 'Minimum total error'). P99 and P999 could not be used as a chi-square test indicated that the M82 mismatch score distribution was, most definitely, not lognormal.

The performance measures were:

	MATCH4	M82
MR1	100.0%	91.0%
LMR	1	40
MTE	0.05%	6.34%

The CPU times taken (on the VAX 11/780) were respectively:

MATCH4 - 16 minutes 21.33 seconds
M82 - 1 hour and 35 minutes

These times are for the whole test, i.e., 10,000 comparisons plus some administrative calculations. However, neither program was optimized for speed. Moreover it should be borne in mind that none of the advantages of the 'array' nature of the MATCH4 algorithm have been realised here; the array operations were all conducted element by element in the VAX 11/780.

Another interesting comparison is the storage space required PER PRINT for the two different methods. The spatial descriptions (required by the M82) fill 3 bytes per characteristic (X,Y and theta)--and up to 100 characteristics are recorded per print. The maximum storage requirement for the minutiae information is therefore 300 bytes per print. The 82 x 2 arrays used by MATCH4 each require exactly 82 bytes per print. They can also be shortened to 62 bytes (see paragraph 4.7) with no appreciable drop in reliability.

5.3 CONCLUSIONS

It should be borne in mind that the comparative test described gave the M82 the initial disadvantage of working from machine-read data.

It would be fair, nevertheless, to conclude from these results that a topological basis for fingerprint coding can provide a fast, economical and extremely reliable basis for computerised single-print comparison. Providing scanning and pattern recognition techniques can be developed to extract this type of topological data automatically (or even semi-automatically) then the schemes described here can provide a sound basis for relatively inexpensive and highly efficient ten-print systems.

Investigation of techniques for use on clear rolled impressions has also led us to a clear understanding of the behaviour of topological codes, and a good idea of which approaches are likely to be most successful when attention is turned to latents.

REFERENCES

1. "Digital Coding of Single Fingerprints--A New Approach for the Computer Age," M. K. Sparrow, Journal of Police Science and Administration, Vol X, No. 2 , June 1982, International Association of Chiefs of Police, Gaithersburg, Maryland.

2. "The Science of Fingerprints," Federal Bureau of Investigation, United States Department of Justice, U. S. Government Printing Office, 1963.

3. "The graphic pen, an economical semiautomatic fingerprint reader," Moore, R. T. and Park, J. R., Proceedings of the 1977 Carnahan Conference of Crime Countermeasures, pp. 59-62, University of Kentucky, Lexington, KY, 1977.

4. "On Estimation of a Probability Density Function and Mode," Emanuel Parzen, Stanford Univesity, Annals of Mathematical Statistics, Vol. 33, 1962, pp. 1065-1076, Institute of Mathematical Stastics.

5. "An Automated Fingerprint Identification System," Joseph H. Wegstein, NBS Special Publication 500-89, U. S. Department of Commerce, February 1982.

*U.S. GOVERNMENT PRINTING OFFICE: 1985-461-105:20113

NBS-114A (REV. 2-8C)

U.S. DEPT. OF COMM. **BIBLIOGRAPHIC DATA SHEET** (See instructions)	1. PUBLICATION OR REPORT NO. NBS/SP-500/124	2. Performing Organ. Report No.	3. Publication Date May 1985

4. TITLE AND SUBTITLE

Computer Science and Technology
A Topological Approach to the Matching of Single Fingerprints:
Development of Algorithms for Use on Rolled Impressions

5. AUTHOR(S)

Malcolm K. Sparrow and Penelope J. Sparrow

6. PERFORMING ORGANIZATION (If joint or other than NBS, see instructions)

National Bureau of Standards
Department of Commerce
Gaithersburg, Maryland 20899

7. Contract/Grant No.

8. Type of Report & Period Covered

Final

9. SPONSORING ORGANIZATION NAME AND COMPLETE ADDRESS (Street, City, State, ZIP)

Same as item 6.

10. SUPPLEMENTARY NOTES

Library of Congress Catalog Card Number: 85-600541

[ ] Document describes a computer program; SF-185, FIPS Software Summary, is attached.

11. ABSTRACT

The motivation for seeking topological descriptions of single fingerprints is provided by the elasticity of the human skin; successive rolled impressions from the same finger will invariably have suffered a degree of relative distortion (translation, rotation and stretching). Topology based systems should be free from the detrimental effects of plastic distortion.

Systems are described for the extraction of simple topological codes from rolled impressions of the pattern types 'loops,' 'whorls' and 'arches.' The generated codes take the form of vectors or simple digital arrays.

The nature and frequency of changes that may occur in such codes is investigated and fingerprint comparison algorithms, based on these topological codes, are developed. The objective of such algorithms is to draw a score derived from the degree of 'nearness' of the topological codes in such a manner that it intelligently reflects similarity or dissimilarity in the two prints under comparison.

Detailed analysis of the performance of such algorithms is given, making extensive use of the results of investigation into the 'match' and 'mismatch' score distributions produced by each one. A final test is described in which the most effective 'topology-based' algorithm was directly tested against one of the best existing 'spatial' algorithms. Topology-based coding, with the inclusion of a crude 'distance measures,' is found to be an extremely accurate and efficient basis for the comparison of rolled impressions.

12. KEY WORDS (Six to twelve entries; alphabetical order; capitalize only proper names; and separate key words by semicolons)

Automated comparison; distoration independence; fingerprints; minutiae; ridge-tracing; topology.

13. AVAILABILITY

[✓] Unlimited
[ ] For Official Distribution. Do Not Release to NTIS
[✓] Order From Superintendent of Documents, U.S. Government Printing Office, Washington, D.C. 20402.
[ ] Order From National Technical Information Service (NTIS), Springfield, VA. 22161

14. NO. OF PRINTED PAGES

75

15. Price

ANNOUNCEMENT OF NEW PUBLICATIONS ON COMPUTER SCIENCE & TECHNOLOGY

Superintendent of Documents,
Government Printing Office,
Washington, DC 20402

Dear Sir:

Please add my name to the announcement list of new publications to be issued in the series: National Bureau of Standards Special Publication 500-.

Name ____

Company ____

Address ____

City ____________________ State ____________ Zip Code ________

(Notification key N-503)