Topological Coding of Single Fingerprints.

Malcolm K. Sparrow, M.A.(Cantab.)

June 1986

ABSTRACT.

The motivation for seeking topological descriptions of single fingerprints is provided by the elasticity of the human skin; successive 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.

This thesis is divided into three parts: part I outlines the traditional use of fingerprints as a basis for personal identification and gives detailed explanation of the arguments in favour of topological coding. Methods for the extraction of topology based digital codes are suggested and the 'placing of lines' is introduced as an effective means of ordering topological information.

In part II specific 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.

Part III examines the special problems relating to fragmentary 'scenes-of-crime' marks. It describes methods of coding fingerprint patterns by a variety of 'topological coordinate schemes', with fingerprint comparison being performed on the basis of localised topological information which is extracted from the recorded coordinate sets. Furthermore, a method for pictorial reconstruction of a complete fingerprint, from its coordinate representation, is demonstrated.

Comparison of fingerprints on the basis of digital topological descriptions is shown to offer a substantial improvement in performance over existing (spatial) techniques.

ACKNOWLEDGEMENTS.

The author is indebted to the Institute of Computer Science and Technology at the National Bureau of Standards for its hospitality and for the use of its excellent facilities. More particularly he is most grateful to Ray Moore (NBS) for sharing the benefits of his vast experience in this field; to Mike McCabe (NBS) for his invaluable advice on computational problems and his expert assistance in negotiating the foibles of Fortran; to Bob Stock (FBI fingerprint automation and research) and his staff for their advice, their cooperation in providing the many hundreds of fingerprints required, and for allowing access to, and use of, their automatic scanning facilities; and to Dr. Gordon Makinson (University of Kent at Canterbury) for his support, encouragement and guidance.

Most of all he is indebted to his wife Penny who acted as research assistant throughout the experimental stages — doing all of the laborious manual preparation of databases for these experiments. Without her patience, diligence and competence this work would have taken very much more than twice as long.

It is almost certain that this research would never have proceeded beyond Part I had it not been for the provision of a Harkness Fellowship from the Commonwealth Fund of New York; it was that award which afforded the opportunity to continue this project in the U.S.A.

Note — Patents and Publications

All of the methods for coding and comparing fingerprints on the basis of extracted topological data which are described or mentioned in this thesis are solely the inventions of the author. Each specific algorithm mentioned (with the exception of the M82 matching algorithm — which is a traditional matcher using spatial information) was written in Fortran '77 by the author.

The coding and comparison methods associated with the algorithms MATCH4 and LM6, together with an image retrieval system based on the pictorial reconstruction algorithm PLOT1, are the subject of a U.S. patent application which has been filed with the U.S. Patent and Trademark Office, U.S. Department of Commerce, Washington D.C..

The following publications consist of material contained wholly or mostly in the stated chapters of this thesis :—

1. SPARROW, Malcolm K. **"Digital Coding of Single Fingerprints: A New Approach for the Computer Age."** *Journal of Police Science and Administration.* June 1982, pages 206 -217.
   — chapters 2 and 3

2. SPARROW, Malcolm K. and SPARROW, Penelope J. **"Topological Coding of Single Fingerprints for Automated Comparison."** Carnahan Conference on Security Technology, University of Kentucky, Lexington, Kentucky, May 1985.
   — chapters 7 - 11 (condensed)

3. SPARROW, Malcolm K. and SPARROW, Penelope J. **"A Topological Approach to the Matching of Single Fingerprints: Development of Algorithms for Use on Rolled Impressions."** N.B.S. Special Publication 500 - 124, U.S. Department of Commerce, May 1985.
   — chapters 7 - 11

4. SPARROW, Malcolm K. and SPARROW, Penelope J. **"A Topological Approach to the Matching of Single Fingerprints: Development of Algorithms for Use on Latent Fingermarks."** N.B.S. Special Publication, U.S. Department of Commerce, September 1985.
   — chapters 12 - 15

TABLE OF CONTENTS

INTRODUCTION.

This project was born in December 1980 on the occasion of an informal visit to the fingerprint department at Kent Police Headquarters. The author was somewhat surprised to find a completely manual operation in existence, and enquired as to the reasons for this apparent technological backwardness in the fingerprint world. The deficiency, it was explained, was not in the available hardware but in the coding processes; there was, in fact, no known method for reducing a single fingerprint to a concise digital code containing sufficient information to identify it uniquely.

Subsequent enquiries into current automation projects showed that significant and substantial efforts had been made towards development of automatic fingerprint identification systems. Research in that field had been undertaken in Britain, the United States, West Germany and Japan dating back, in some cases, as far as the early 1960's. At the present time systems for conducting automatic fingerprint searches are manufactured by at least three industrial companies, and several other companies will probably enter the rapidly growing market within the next few years.

It could be considered too late, therefore, to be developing new fingerprint coding schemes suitable for automation. Indeed, anyone who believes that present day systems offer all that is desirable in terms of accuracy, speed and cost-effectiveness will undoubtedly not be interested in any such new schemes. However, very few in the field believe that to be the case. There has been, of late, a growing tendency to look for fresh approaches to this whole problem in the hope of achieving very much greater accuracy, speed and cost-effectiveness.

Virtually all existing automated systems have one property in common — they perform single print comparison on the basis of spatial information (i.e. the coordinates of the ridge characteristics). This remains the case despite a variety of forays, at various times and by a variety of people, into topological coding of fingerprints. Those forays were presumably born out of conviction that topological coding ought really to offer much in terms of distortion-independent data, and thereby ought to facilitate automation and render it more effective. Those past efforts at topological coding have all been shortlived and have come to little or nothing. Some studies have explicitly concluded that 'topology has nothing to offer' in this field.

It is certainly true that schemes for coding the topology of a print have been devised in the past, and that they have not worked! That does not show us that 'topology

has nothing to offer'; rather that application of some imagination is likely to be required before we find an effective solution. This project is intended to fill a gap in the history of automation research by devising topological coding methods that are neat and effective, and by demonstrating the value of such an approach.

The research has been spread over a considerable period of time — but falls into three natural parts. Part I was written in 1981 and presented to the National Police Staff College, Bramshill, Hampshire. It contains ideas, ideas, and some more ideas. As such it carried little weight, having been completed without any supporting experimentation, and having been written by a complete 'outsider' to the world of fingerprints and fingerprint automation.

Parts II and III describe the practical experiments designed to test those ideas, and give an account of the development of matching algorithms based on them. This phase of the work was conducted at the Institute of Computer Science and Technology, National Bureau of Standards (U.S. Department of Commerce) in Washington D.C. between September 1984 and August 1985. Part II describes the application to comparison of clear rolled impressions (as in 10-print card searches) and Part III describes the application to latent marks.

Some consideration was given to disbanding Part I as a separate entity and prefacing Parts II and III with the relevant ideas — but Part I was eventually left intact for the following two reasons : firstly, that it provides a convenient vehicle for explanation of the nature of fingerprints, the conventional classification systems and the motivation for seeking distortion-independent descriptions : secondly, that it traces the development of ideas prior to the need for substantiation by experiment. Part I shows which ideas evolved by processes of *reasoning*, rather than by *experiment*.

Part I is not, however, in its original form. It has been extensively edited in view of the change of audience. It was originally written to be read by Police Officers with no detailed knowledge of Mathematics, Computer Science or of the Science of Fingerprints.

Perhaps an advance apology is appropriate for the comfort of true Mathematical Topologists — the use of the word 'topology' in this whole project has almost nothing to do with the subject of mathematical topology, nor is any substantial use made here of any of the results from that subject. 'Topology' is simply a good word (if not the best word) for making clear the fact that we are dealing with an *elastic* world where distances and directions mean very little.

The author went into this research with a severe warning from an experienced British automation researcher to the effect that "ideas are all very well — but real fingerprints always behave far worse than you could possibly imagine". Having experimented with a great number of 'real' fingerprints, both rolled impressions and latent marks, the conclusion which must be drawn from this research is that, in this field, a topological approach has a *great deal* to offer.

PART I

Fingerprints and their classification

CHAPTER 1.

THE NATURE OF FINGERPRINTS AND THEIR CLASSIFICATION.

1.1 Friction, skin and perspiration.

Nature, it seems, intended us to be able to pick things up without there being any excessive danger of them sliding out of our hands. To that end, all the primary contact areas of our hands and feet are covered with ridges of elevated skin. These ridges are perpetually moistened by sweat pores positioned near the midline of the ridges. Ridged skin affords a far greater degree of friction between skin and object grasped than would completely flat skin — just as a well treaded tyre grips the road so much more effectively than a bald one. The slight moistening of those ridges by the sweat pores studded along them adds a helpful degree of 'stickiness'. In fact we are not alone in being provided with such apparatus; it is common to all primates.

A sectional cut across such a ridge would reveal a structure as illustrated in figure 1.

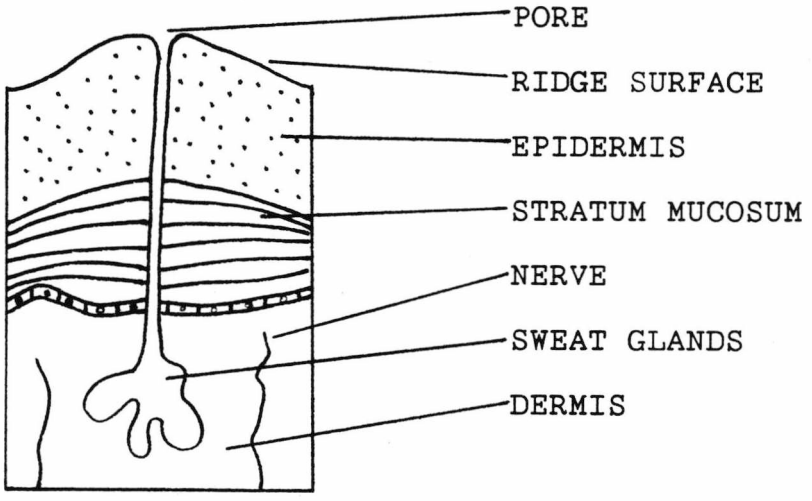


Figure 1. Sectional cut through the skin.

1.2 Ridge pattern and identification.

The usefulness of these ridges as a means of identification is a simple corollary of a few basic properties:—

Firstly: the pattern formed by the ridges is permanent and invariable. It will remain the same throughout the entire life of the individual, from its formation during the third and fourth months of pre-natal development until its decomposition after death.[1] The pattern of these ridges has been found to survive the process of body decomposition long after every other identifiable feature of the body (with the possible exception of the teeth) has been eroded. The only occasion on which the pattern will change is as a result of deep-seated injury to the area — deep enough to disturb not only the epidermis (surface layers) but the dermis (sensitive tissue).[2] In this case scar tissue will form and be visible within the pattern. Normal superficial injuries will only temporarily alter the ridge formation, as the skin will always grow back into its original shape as determined by the positions of the sweat glands in the dermis.

Secondly: these ridge patterns are unique. So far, the only part of these patterns to be recorded extensively have been those parts falling at the fingertips (i.e. on the distal part of the finger). Uniqueness of those patterns has not yet been, nor ever can be, proved; but it is *assumed*, in the light of the experience that nobody has ever yet found two the same.[3]

Thirdly: the presence of the sweat pores along the ridges keeps the ridges slightly moist. It is that moisture which remains on any object touched — leaving therewith an impression of the ridge pattern from the contact area of skin. Methods of recovering that 'latent' impression and restoring it to a visible form are many and varied. New techniques have now been pioneered whereby latent prints can be 'lifted' (developed) from surfaces such as the flesh of another human being, tissue paper and cloth.[4] Those methods are not the subject of this study and are consequently not examined here. It would be true to say, though, that advances in these development techniques make advances in coding and searching techniques all the more important.

1.3 The two bases for differentiation of prints.

Prints have a variety of features by means of which they may appear similar or dissimilar to others. Firstly there is *pattern type*. This can perhaps best be described as the general appearance of the pattern — or the nature of the *flow* of the ridges. Secondly there are the *ridge minutiae* — the intricate detail of the pattern. One particular ridge may end abruptly, or divide into two; the point at which it does so is called a *ridge characteristic*. There are, on average, roughly one hundred such characteristics visible on each clear rolled print.

It is as well to be reminded at this stage that *pattern type* has been the traditional basis for classifying prints — and that it is only *ridge characteristics* that can identify a print uniquely.[5] This paper is concerned with the existence, or otherwise, of methods for actually coding the ridge characteristics, as opposed to pattern type.

1.4 Pattern type.

Various people, in the past, have described a variety of divisions into pattern *classes*. The most useful for our purposes are the classes devised and employed by Sir William Henry as the basis of his ten finger system, as these are the ones currently most widely used. Some of the boundaries between these eight types have been adjusted from time to time – according to different sets of rules relating to borderline cases.

The eight basic types are:—

1. ARCHES (plain). (see figure 2.) Here the ridges run from side to side in an arch-like fashion, without turning back on themselves. These are of particular interest, and will raise their own particular problems in chapter 3.

Figure 2. Arch.

Figure 3. Tented arch.

2. TENTED ARCHES. (see figure 3.) These are similar to arches, but the ridges near the centre tend to have a vertical direction on either side of an axis towards which adjoining edges converge. They therefore give the impression of a tent supported by a pole.

3. LOOPS (Radial or ulnar). (see figure 4.) Here the ridges near the centre of the pattern double back on themselves and form a type of hairpin structure. These are subdivided into Radial and Ulnar loops, depending on whether the hairpin slopes away from, or towards, the little finger of the same hand.

4. WHORLS. (see figure 5.) The central ridges form a circular pattern, giving the appearance of a vortex.

Figure 4. Loop.

Figure 5. Whorl.

5. 'DOUBLE' or 'TWINNED' LOOP. (see figure 6.) Where two hairpin-like loops appear wrapped around each other, each originating from opposite sides of the print.

6. LATERAL POCKETS. (see figure 7.) Where two loops appear on the same print, but they originate from the same side of the print.

Figure 6. Twinned loop.

Figure 7. Lateral pocket.

7. COMPOSITES, COMPOUNDS and ACCIDENTALS. Prints comprising more complicated combinations of the above types. These are relatively uncommon.

1.5. Ridge Characteristics.

The 'ridge details', 'ridge minutiae', 'Galton details', or 'points' as they are variously referred to, are the tiny irregularities in the ridge pattern by which a print can be identified. There are eight types which are generally recognised; they are shown in figure 8.

1. Ridge ending.
2. Bifurcation or fork.
3. Enclosure or lake.
4. Dots.
5. Isolated or short ridge.
6. Ridge break.
7. Ridge crossing.
8. Ridge trifurcation.

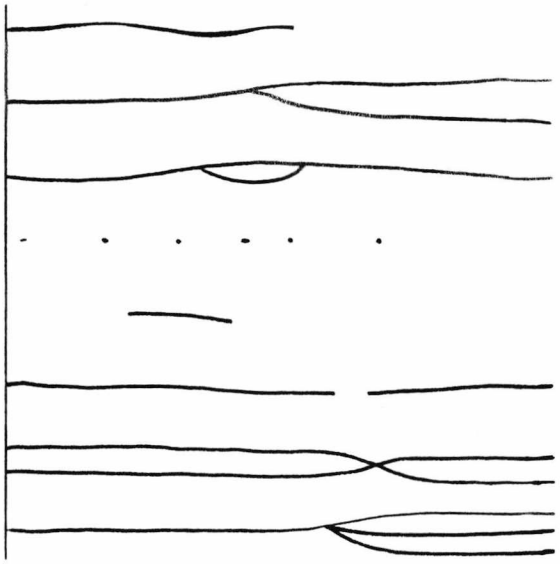


Figure 8. Ridge minutia.

There are a number of less common and more dubious features that some experts recognise as being *characteristics* in their own right;[6] they include those shown in figure 9.

9. Spur.
10. Double fork.
11. Bridge.
12. Triradius.
13. Crossover.

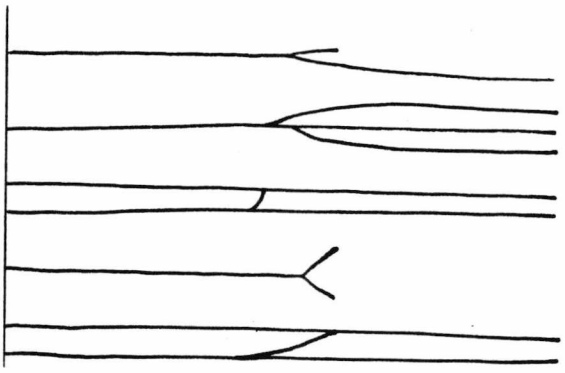


Figure 9. Additional ridge minutia.

Many are of the opinion that these should not be regarded as basic features, as they are compounds of other features: for example, a spur is a bifurcation with a ridge ending, and a double fork could be seen as two bifurcations.

These *characteristics* are not connected with the pattern type at all; they may appear practically anywhere on any one of the eight types of print.

1.6 Deltas.

The triradius ((12) above) is of interest in that it usually occurs at a *delta. Deltas* can be seen on figures 4,5,6 and 7. They are points where the general flow of the ridges diverges drastically — giving a triangular appearance. It is interesting (and will be very useful to us) to note that every loop pattern has one delta at varying distances from the *core* or *heart* of the loop; every whorl, twinned loop, lateral pocket, composite or accidental has at least two deltas appearing [7] — and it is only plain arches that have no deltas at all.

Exact rules for determining the precise positions of deltas are already in existence as they are widely used for classification purposes. [8]

1.7 Number of characteristics required for identification.

The 'Standardisation Committee of the International Association for Identification' reported in 1973 (after a three year study of the subject) that there was "no valid scientific basis for requiring a minimum number of ridge characteristics to be present in two fingerprints in order to establish positive identification". In Britain 16 points of similarity are necessary for proof of identification in court — or 10 points on each of two different fingers. Most European countries require 12. The origin of the standard of 12 is unclear; one theory suggests that 12 points were necessary if the use of fingerprints was to be a better means of identification than the Bertillon system of body measurements, which recorded 11 features. [9]

The chance of two distinct individuals having a print with 12 points in similar relative positions has been estimated to be in the order of one in ten million million. [10]

1.8 Number of characteristics required for searching.

It may be necessary to find 12 or 16 points of similarity for proof of identification in court — however the fingerprint expert will be convinced that he has found the right print if he finds 5 or 6 in the correct place, and he'll be fairly sure with only 3 or 4. [11] This fact will be referred to many times in the following chapters.

1.9 Finer division of the eight pattern types.

Wherever substantial collections of prints have been kept it has always been necessary to break down the eight basic pattern types further. This has traditionally been

done by closer examination of some particular features of the print. The three principal methods used are :—

(a) Ridge counting. (Used for loops and whorls.)

(b) Classification of Core type. (Mostly for loops.)

(c) Ridge tracing. (For whorls and composites.)

1.10. Ridge counting.

Loops account for 60 to 65% of all prints,[12] and so the sub-division of this, the largest class, is vital. As mentioned in para 1.6 every loop has a core and a delta. Ridge counting is simple determination of the number of ridges that cross an imaginary straight line drawn from core to delta. There are numerous rules for determining the precise position of the core and delta.[13] The expert uses a magnifying glass with a line printed on it to help him. Ridge counts may vary from 1 to 50, or occasionally even more. The count in figure 10 is 20.

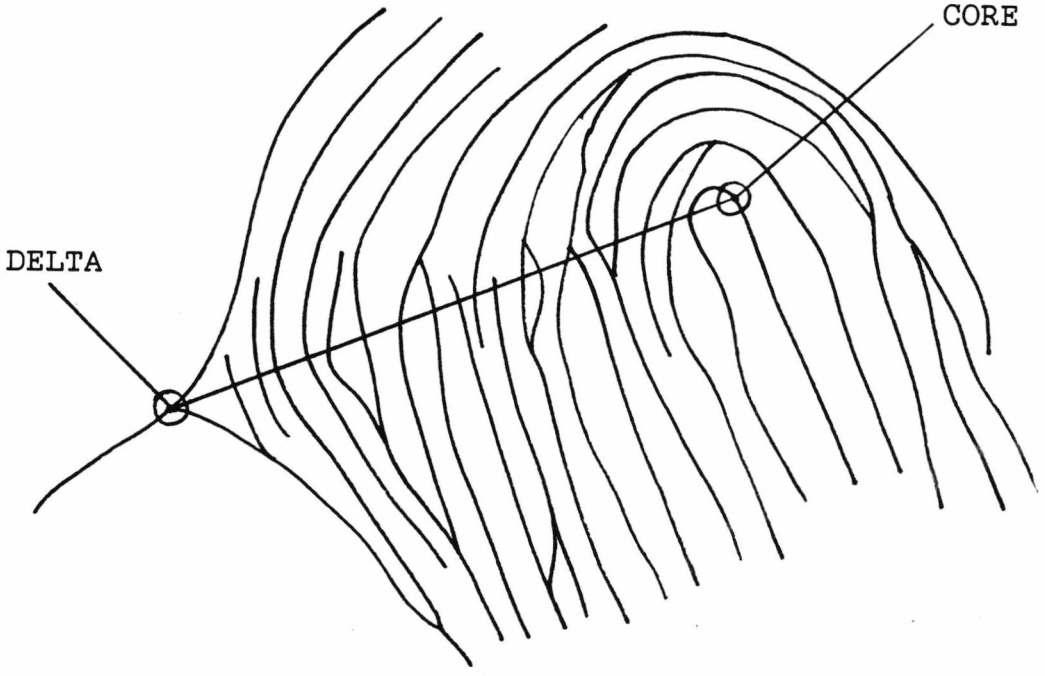


Figure 10. Loop with a ridge counting line superimposed.

1.11. Core type.

The 'heart' of a loop takes a large number of distinct forms and these have been

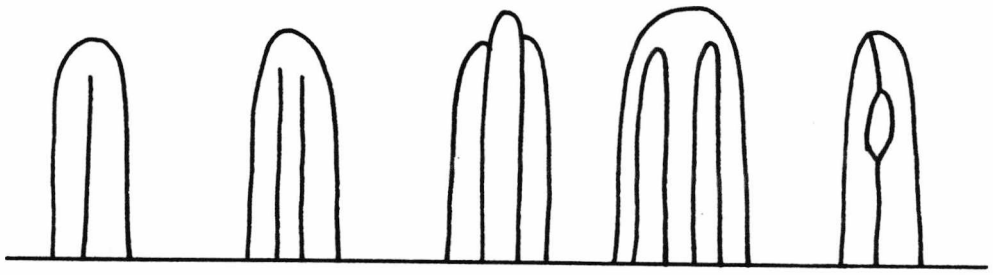


Figure 11. Examples of core type.

divided into groups for subclassification purposes.[14] Figure 11 shows some common examples.

1.12. Ridge tracing.

The second largest group are the whorls, accounting for 30 to 35% of all fingertip patterns. As mentioned in para 1.6 these always have two deltas, one on either side of the core and both below it. 'Ridge Tracing' divides whorls into three groups [15] according to the following instructions :—

Find the lower ridge of the left hand delta and follow its course from left towards the right. If it stops (ridge ending) drop to the ridge immediately below and continue. Should the ridge fork, take the right hand limb and carry on. That course will either take us above, straight to, or below the right hand delta. See figure 12.

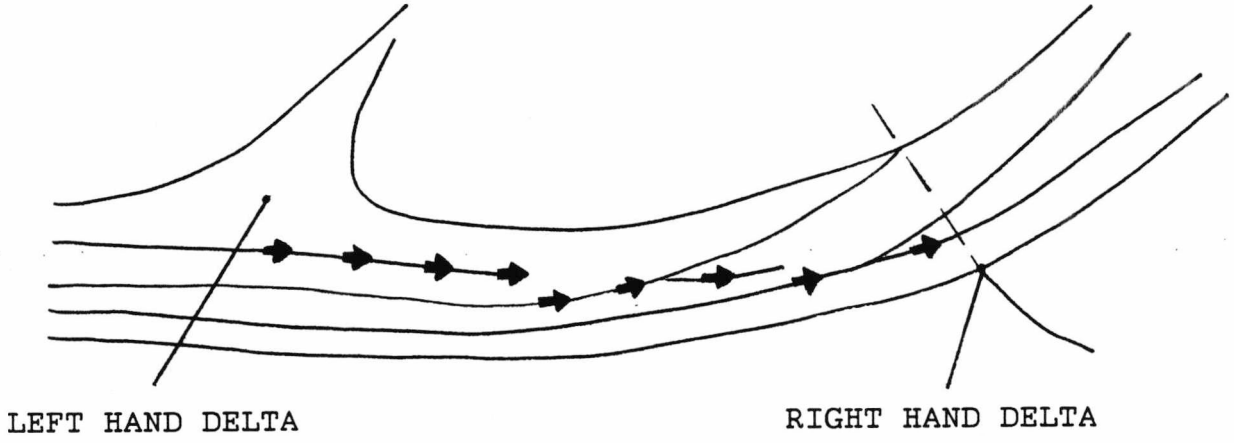


Figure 12. Ridge tracing in a whorl.

Whorls are then classified as 'I', if the ridge trace passes three or more ridges 'inside' (above) the right delta, 'M' if within three either side, and 'O' if it passes three or more 'outside' (below) it. That divides whorls into three classes of comparable size.

In the case of composites, ridge tracing can also be used — but attention would be limited to the two outermost (lowest) deltas.

1.13 The Henry system.

The 'Henry' system — together with all other ten and five finger systems — is a means of combining and recording the pattern types (together with the subclassifications mentioned above) of the ten fingers, and of filing them in a manner suitable for quick retrieval. [16] The actual mechanics of the Henry system, or of any of the other combinatorial systems are of no interest to us here.

1.14. Computerization of combinatorial techniques.

In the United States of America the National Crime Information Centre (NCIC) is a computerised information handling service available to all the various law enforcement agencies. It includes a computerised ten-finger classification system whereby each of the ten fingers is categorised by the methods described above, and then the digital codes for the ten classes so represented are simply written down in order — giving one long number.[17] These numbers are stored, together with the individual's name, date of birth and criminal record number. Given an unidentified set of ten prints these numbers can be used to reduce the field of search to just those persons with all ten fingers of the correct classifications.

This must not be confused with attempts to computerise a single print index — nor to code the ridge characteristics. It is simply a kind of computerised 'Henry' system, a convenient method of recording the conventional information. The NCIC system does nothing towards breaking down the pattern types sufficiently to facilitate single fingerprint identification.

1.15 The 'Battley' single-print system.

Harry Battley's system is much used in Great Britain and abroad, and is, without doubt, the most practical of the traditional attempts to classify single prints more finely.[18] His system relies on the taking of a number of physical measurements.

The features used in his system are, in order, as follows:—

(a) Basic pattern type — as per para 1.4.

(b) Subclassification as per para 1.9.

(c) For loops and whorls: physical measurement from the core to each delta.

(d) For loops and whorls: the ridge count from the core to each delta.

(e) For whorls: physical measurement from the summit to the bottom of the innermost recurving ridge. See figure 13.

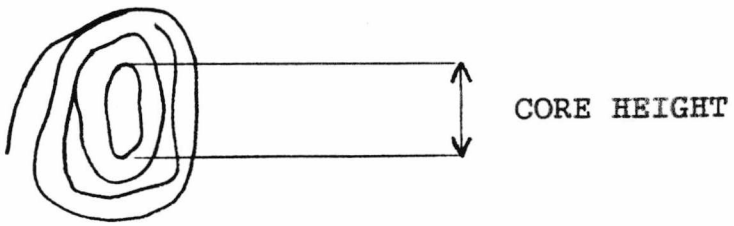


Figure 13. Core height of a whorl.

(f) For whorls: the core type. (He defined five classes of core.)

(g) For double loops and whorls: the ridge trace category, as per the Henry system. See para 1.12.

(h) For tented arches: physical measurement from the crest of the arch, to the closest point of the highest platform ridge. See figure 14.

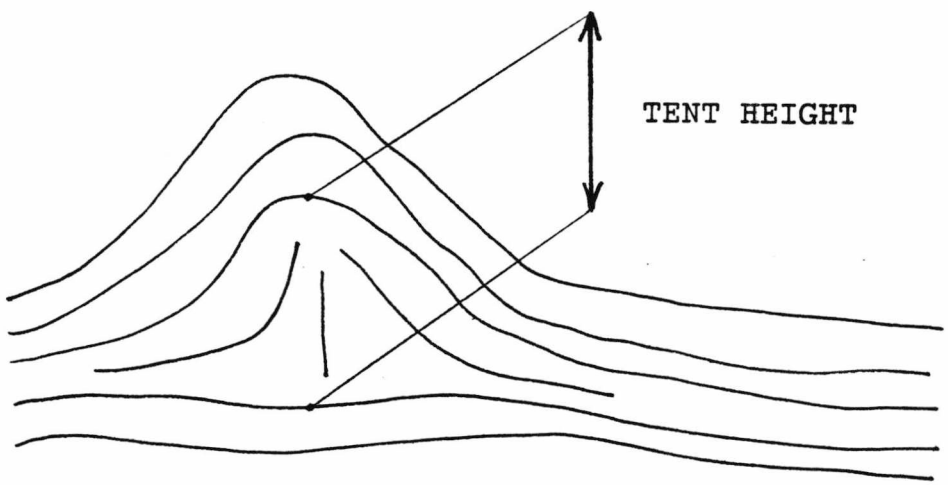


Figure 14. Tent height of a tented arch.

The Battley system does not subdivide the type 'plain arch' at all: there is nothing reliable to measure. Notice also that it makes no attempt to record any of the ridge characteristics themselves.

1.16 Conclusion.

This chapter has provided a brief outline of present methods of classification used the world over. Each system varies slightly in the precise demarcation of the classes, but all use the same basic approach.[19] It is only an appreciation of the nature of the methods used, rather than a knowledge of the details, that will be required in order to appreciate the significance of the following chapters.

CHAPTER 2.

THE NEED FOR A NEW APPROACH.

2.1 Deficiencies of 'pattern type' classification.

A stranger to the world of fingerprints will probably have become aware just how thoroughly *visual* the origins of fingerprint classification have been. Present day methods remain basically visual — and 'methods, measurements, rules and devices' are employed only to classify borderline cases, or to subdivide classes more finely. Visual, in this context, means 'as readily discerned by the human eye' — the sort of descriptions that, through our experience, spring readily to the forefront of our minds when confronted with a pattern.

Naturally enough, *pattern type* is the feature of prints that strikes us first. Human beings have a pretty good idea what a 'whorl' or a 'loop' means in the natural world, and are quick to recognize anything of that appearance in complex patterns.

The mistake we are prone to make, in approaching computers, is to assume that the most effective way of using them is to programme them to 'think' in the same way that we do — and hope that they will be able to do the same task faster. We fail, sometimes, to recognize two things :—

> Firstly: that it may be incredibly complicated to reduce the sort of conceptual discernment that we have to a tightly defined set of operations comprehensible to a machine.
>
> Secondly: that computers are eminently well suited to performing intricate detailed analysis by 'number crunching' in a manner that we would never dream of attempting manually.

Therefore, if we hope to make the best possible use of computers in fingerprint identification, we should avoid blundering into the assumption that traditional classification systems are going to lend themselves to automation.

John Fitzmaurice of Cambridge, Massachusetts was one of the pioneers into research with the use of computers and optical scanners in the early 1960's. He arrived at the conclusion that no headway towards computerisation could be made so long as the traditional methods of classification and interpretation of patterns remained the cornerstone of fingerprinting. Machines could not be effectively programmed "to make distinctions among questionable pattern types, which are based on arbitrary rules of interpretation on which technicians often disagree".[20] Fitzmaurice and others concluded that interpretation of pattern type could profitably be bypassed if a machine could be devised to catalogue,

file and compare fingerprints *by their individual characteristics alone* — precisely for the reason that identity is established by characteristics and not by pattern.

Parduman Singh actually states[21]

> "... nevertheless one thing remains for certain for ever: no change takes place in the sequence of their ridge characteristics. The whole secret of identification thus rests solely on the order in which these minutiae appear in the two prints under comparison, and it need not be emphasized that exact measurements, or exact shapes, have no place in the identification of fingerprints ... ".

We cannot possibly hope to have a computerised single print index dependent on pattern type alone when, for instance, the class 'plain arches', which represent about 5% of prints, are not subclassified at all.

2.2 Current research into coding characteristics.

The bulk of the current research being conducted into the coding of prints (by characteristics) is largely following the lines pioneered by Fitzmaurice.[22]

Once Fitzmaurice had appreciated the difficulties of trying to programme a machine to spot patterns, and had realised that it was time to start coding characteristics, he then fell into the next most natural trap! He made the assumption (as many have done since) that the way to record the characteristics was by looking to see where they were (i.e. their position in space). Why? — probably because that is the natural way for human beings to observe them. It is not, however, the only way of recording the details.

That basic assumption, it seems, has never been questioned — let alone 'thrown off'. It pervades the techniques of all current automation projects. For example, the FBI's approach to automation has been described[23] thus :—

> "The general approach adopted by the FBI's automatic fingerprint identification system is to duplicate, insofar as possible, the human technician's visual and mental processes as he performs the task of fingerprint identification."

Hence the FBI endeavour to programme machines to identify pattern type, to *align* the two prints under comparison and then to compare the spatial positions of the characteristics visible.

2.3 Deficiencies of the 'spatial' approach.

Moenssens states "the most critical test of any automatic classification system is the capability of the apparatus to allow for pressure distortion".[24] The skin of the fingers is soft and flexible. If two Police Officers take the same man's fingerprints, even under ideal conditions, they will come out slightly dissimilar. Each will apply differing distributions of pressure, and may tilt or twist the print to varying degrees. Moreover, if a *latent* print is lifted from an object at the scene of a crime, then its precise shape will depend not only on the pressure applied, but also on the shape of the object touched and the manner in which it was held. Comparison of two prints, therefore, must allow for considerable variations in the distances and angles observed within the print.[25]

Description of the position of characteristics is through the use either of Cartesian or polar coordinate systems — using some standardised rules for selection of an origin and for determination of the 'correct' orientation. Of course, one has to rely on the origin and orientation being identical each time, or *close enough* to identical.

The likelihood of incorrect re-registration of a print, combined with some spatial distortion, almost certainly ensures that most, or all, of the minutia coordinates will change from one impression of a finger to the next. Thus the automated comparison of prints by this method is an incredibly complicated process. The number of operations currently required for spatial comparison of the ridge data of two single prints, using sophisticated statistical techniques, is in the order of six million.

Proponents of such systems — Fitzmaurice included — have to allow a degree of leeway in their comparison of coordinates.[26] Unfortunately the greater the degree of leeway allowed (in order to prevent the machine from missing correct matches) the less the power of resolution (efficiency at rejecting incorrect matches) becomes.

Attempts have been, and are being, made to find ways of enabling a computer to systematically 'un-distort' two prints under comparison in order to cut down the amount of leeway required. Some promise has been shown by the employment of 'Analogue Video Reshaping Systems' — which were developed to compare biological and geological material and to remove distortion from crushed or distorted geological specimens.[27]

Such are very sophisticated comparison methods indeed — but it is my contention that comparison procedures must be made simple even if the initial coding procedure has to be made more complicated.

2.4 Sophistication of coding, rather than of comparison.

Suppose you have a database, in an automated system of 100,000 prints, recorded in a coded form. Suppose, then, that you are given an unidentified print to check against your file: you have first to code that print, and, second, to compare the code with those in your file. It is quite obvious that you have to do the coding process once, and the

comparison process up to 100,000 times. Economically speaking, therefore, it is foolish to have a simple coding process that requires a sophisticated comparison; wiser to prefer a sophisticated coding process that allows for a simple comparison.

Now the simplest comparison of all, for a computer, is straightforward checking of two numbers to see if they are the same. So our ideal aim would be to find a coding process that generates one number (albeit a long one) for each print. If we could find such a thing the computer, to compare two prints, would simply have to check the two numbers against each other — if they agree, then the prints represented by them would be the same (provided the numbers were unique to each print).

Using a 'topological' approach to the problem, this aim outlined above is virtually attainable.

Topological descriptions of fingerprints are those descriptions whose information content reveals how the characteristics are connected up, and how the various ridge segments are joined — rather than *where* the characteristics are, or *in what direction* the ridges flow.

The beauty of topological descriptions lies in their complete immunity to variation through the distortions of stretching and twisting. Moreover (in relation to computers) topological descriptions are of a purely digital, rather than analogue, nature.

2.5 Topological information in fingerprints.

The question which immediately springs to mind is how to record a concise topological description of a pattern so complex as a print. That will be dealt with in the next chapter. For the time being let us recognise that there is a vast amount of topological information within each print just begging to be extracted.

Suppose figure 15 is a small portion of a print — near the core of a loop. A, B, C, D, E, F, G and H are just some of the ridge characteristics present.

We can write down a few of their topological relationships simply, like this :—

"A is a ridge ending and B a lake. They are directly linked by one ridge (never mind how long that ridge is or which way it slopes). B is also linked to C — a bifurcation. C is linked, further, to F (ridge ending) by the right fork, and to G by the left fork".

We could continue for some time writing down the wealth of topological detail contained just in this small segment.

Notice that every one of the topologically descriptive comments made above will not change at all, however much figure 15 was rotated, twisted or stretched. This will

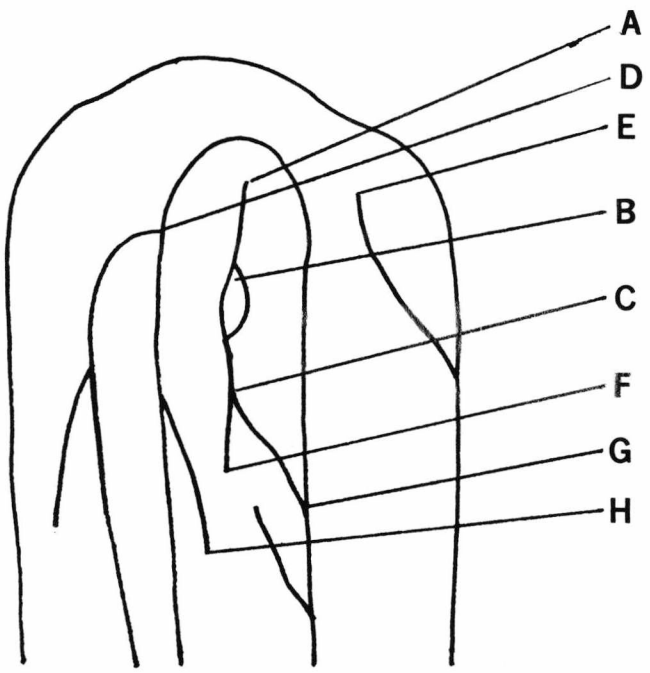


Figure 15. Ridge characteristics of a loop.

always be so provided we never mention *how long* a link is, nor *at what angle* lines meet or slope. Such measurements are distance dependent, and therefore subject to change.

2.6 Topological information already in use.

The use of ridge-tracing as a means for subclassifying the class of 'whorls' is essentially a *topological* description of the relationship between two deltas. The rules for ridge-tracing (see para 1.12) say, for example, "Follow this ridge, and should it fork, turn to the right and carry on". There is no mention of how far you may have to go before the ridge ends or forks. It is a purely topological set of rules — hence ridge-traced subclassifications are invariant under the types of distortion mentioned. The idea of 'following a ridge and seeing what happens along it' will be used extensively in later chapters.

It is worth pondering the amount of topological information that is discarded every time ridge-tracing is done. If the ridge-trace, obeying the rules, drops twice onto a lower ridge, then has to fork right twice at bifurcations before passing outside the right hand delta — why not classify the whorl in question as a type 'DDFF - O' rather than just an 'O'? ('D' for a drop, 'F' for a fork, written in the order in which they occur). Preserving and recording that information would divide whorls into about 100 classes, rather than just three.

2.7 The problem of ordering.

Remembering that, ideally, we would like to devise a system which would generate a simple digital numeric code for each print, we must try to find a way of getting the vast amount of topological information contained in a print into some semblance of an *order*. Then, when we have coded the topological relationships between characteristics we will know in what order they must appear in the code.

Finding such orderings is a substantial problem, and has been, almost certainly, the major stumbling block for previous attempts to devise topology based coding systems. The following chapter gives some ideas for solution of the ordering problem.

CHAPTER 3.

THE ORDERING OF TOPOLOGICAL INFORMATION.

3.1 Avoiding 'spatial' orderings.

Having gone to such pains to eliminate spatial considerations from the actual information to be recorded, it would seem a shame to have to use them to 'order' the information. We may feel drawn towards a *spatial* ordering simply because it is, once again, the natural approach. It is clear that you could put all the characteristics in order by, for instance, starting at the left hand side of the print and working over towards the right — or from top to bottom; maybe one should start at the core and work outwards?

It is best to avoid *all* such spatial orderings, and the reason for this should, by now, be fairly clear: all such orderings are distance dependent and therefore subject to change. It is no good being able to record the right information if it turns up in the wrong order. We should therefore reject spatial considerations as the basis for any efficient ordering system.

3.2 Topological progression from a fixed point.

Having rejected spatial orderings we should next consider using *topological progression* from a fixed point. The fixed point could be determined by rules already in existence under the Henry system[28] for pinpointing the core of any pattern other than an arch (which has no core).

Having established the 'starting point' by such a set of rules, we would code the nature of the core itself, first of all. Suppose it were the core of a loop — we could code that 'L'. That point would lie on a ridge — so we can 'look along the ridge' to see what is the first characteristic that we would arrive at by 'walking along' it, in either direction. Suppose that, one way, it came to a fork, and the other way to a crossover. We could then record our findings so far thus:

$$L \begin{cases} F \\ C \end{cases}$$

Then, from the fork we could follow the other two ridges away from it (a fork can be regarded as the meeting of three ridge segments) and see what happens: we'll assume that one goes 'out of sight' (i.e. off the print without passing through any characteristic) and the other comes to another fork. Also assume that the other three ways out of the crossover come, respectively, to a fork, a ridge ending and a bridge. If we continue using

the capital letters as the codes we now have:

$$L \begin{cases} F \begin{cases} O \\ F \end{cases} \\ C \begin{cases} F \\ R \\ B \end{cases} \end{cases}$$

This could be stored as L/FC/(OF)(FRB)/ ... and progressive exploration could be continued for as long as necessary.

The computer, to compare two prints, then merely has to compare one string of characters against the other, and if they agree for more than the first five or six letters then there are 5 or 6 characteristics in similar topological relationship to each other and we can then be fairly sure that a 'correct match' has been found. (See para 1.8 re number of characteristics required for searching.)

3.3 Reasons for rejecting 'coding by progressive characteristic association'.

In retrospect, however, this method has to be rejected in favour of more effective ones. The method described has four principal drawbacks :—

(a) The need for a starting point excludes arches from the system.

(b) Exploration by progressively following ridges sends us all over the print, collecting fairly sparse topological information. ('Sparse' in the sense that it will often record information a long way from the core before getting to characteristics which are much closer to it.) That is less efficient than a *denser* system because it would require us to have available a comparatively large physical area of print from the scene of a crime before we could use it for identification purposes.

(c) The frequency with which ridges run 'out of sight' without passing through characteristics is disconcertingly high — and often an exploration as described will 'run out' in the sense that all ridges currently under investigation run 'out of sight'. We then would have to devise rules for *jumping* from one ridge to another in order to be able to explore further. It gets a little complicated at this stage!

(d) Misinterpretations of characteristics can and do arise — and will certainly do so in automated systems. The effect of one single error in a progressive system is to throw all the subsequent entries into disarray — hence if we have one error in the coding we will not get a 'close' match at all — it will change a large proportion of the character sequence.

The problems posed by possible topological 'mutations' of characteristics are more fully dealt with in the next section.

3.4 Problems of topological mutations.

We have said a great deal so far about the variations that distortion will cause in any spatial information — but nothing, until now, about topological variation. Fortunately topological variation is less frequent — but, nevertheless, any usable system will have to allow for it. Let us explain what is meant by 'topological variation'.

It is a fact that a change in pressure applied can alter a ridge ending to a bifurcation, or vice versa.[29] In figure 16 a decrease in pressure applied could lead to the line segment AB disappearing, consequently changing a bifurcation into a ridge ending.

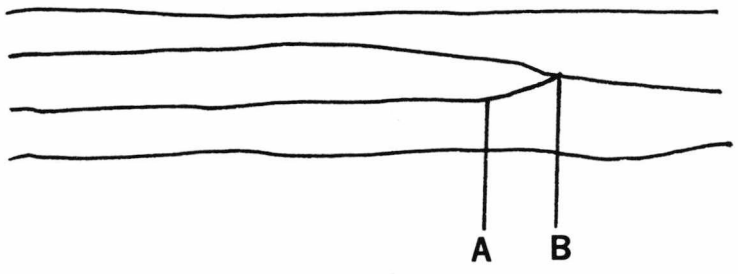


Figure 16. Bifurcation / ridge-ending mutation position.

Also, subsidiary ridges may appear, with increased pressure, that were not there before. These are thinner, lighter lines that sometimes appear between the principal ridges.

Such changes are relatively uncommon — but we should devise sufficient *leeway* into any digital system to allow for several such changes in each print.

This new factor effectively banishes all hope of finding an ideal, unchangeable, unique, digital code — with its simple one-move comparison. It means that we have to be able to identify *close* matches (in terms of topological information stored) as well as *perfect* matches.

3.5 Topological exploration ordered by lines — or the 'fishbone' method.

The origin of the idea of using lines lies in the special problems presented by the 'arch' pattern. A plain arch has no identifiable core, nor any deltas,[30] but a central line is fairly easily placeable. (See figure 17.) This line has been *placed* on the arch simply by joining the summits of the upcurving ridges.

We have available to us a number of other ways of placing lines on various types of pattern. For instance, the ridge-counting system (described in para 1.12) places a line

Figure 17. Line-placing by ridge summits on an arch.

from the delta to the core of a loop. We can define rules for the placing of lines (either straight ones or flexible ones) on all the types of pattern.

Provided we find a way of placing the lines across the flow of the ridges, that line will determine for us a series of points of intersection with ridges; i.e. an *ordered* set of points on adjacent ridges. In fact, as will become evident later, the precise positioning of the line need not be critical provided it runs roughly orthogonal to the flow of the ridges.

Such an ordered set of points can now act as a basis for the generation of ordered digital codes. Figure 18 represents the ridge-counting line drawn on a loop pattern. The points of intersection with ridges are labelled A,B ... N as we go away from the core. From each of the points we can *look left* and *look right* along the ridge to 'see what happens'. If we code the possible events (i.e. what characteristic does exploration along that ridge come to first) this will yield an ordered string of characters or digits where each is *not* dependent on the correct interpretation of the previous characteristic.

Let me illustrate this method more fully by coding the loop shown in figure 18. We will use the set of digital codes shown in figure 19 for possible *events* as we *look along the ridges* :—

So, in figure 18, starting at point A — first look left (downwards) and see that the ridge runs out of sight (coded 0), looking right (and following the ridge round) we come to a ridge ending (coded 3). Then point B; left — out of sight, right — 'see' a new ridge starting on the left (coded 6). So far, then we have 03 06. Continuing in like manner as far as point J we can generate the 20 digit code :—

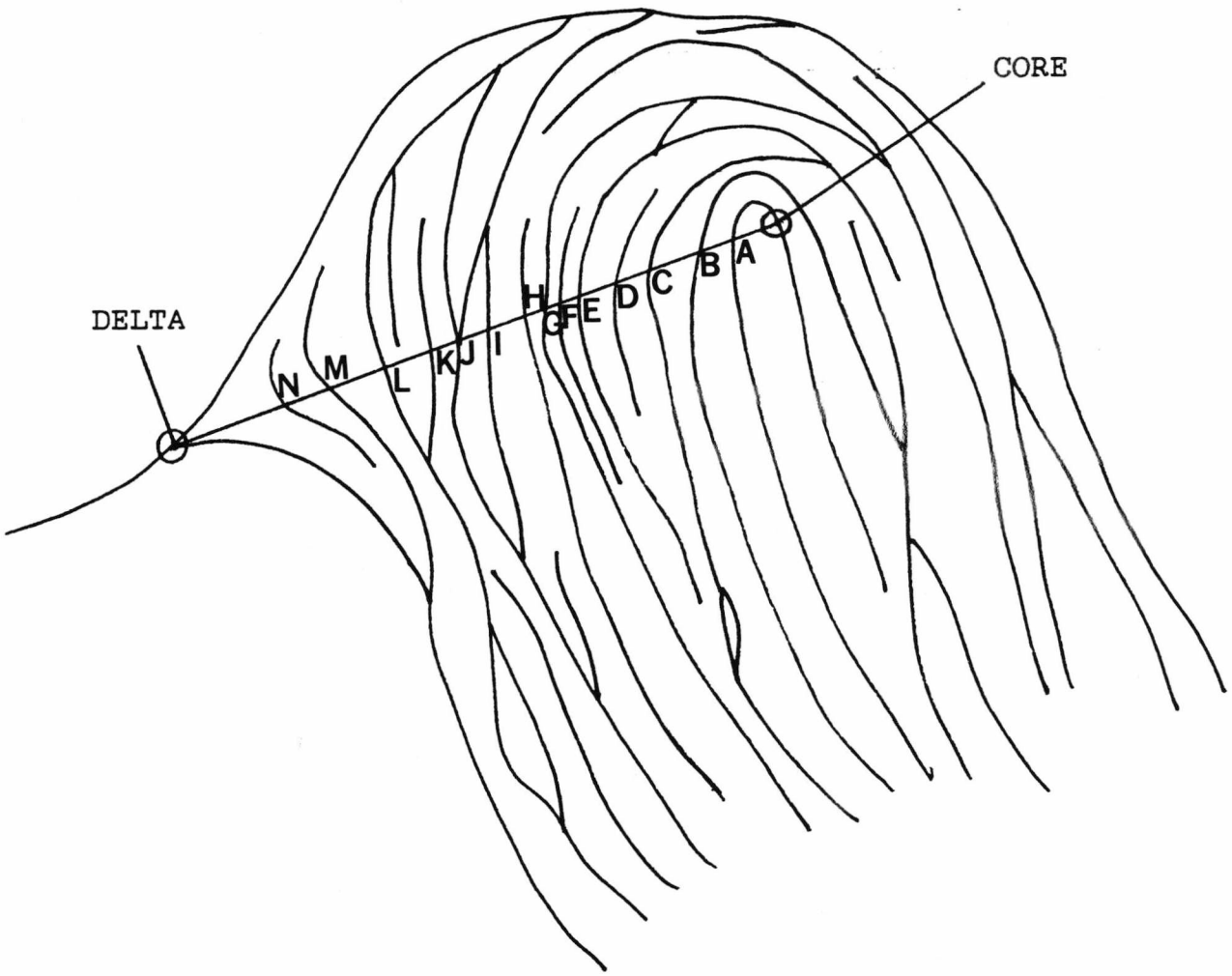


Figure 18. Ridge-counting line showing intersection points.

03 06 72 33 37 33 06 83 42 27

We have thereby systematically recorded 20 pieces of interrelated topological information about characteristics, which is ample to identify the print uniquely!

3.6 Comparison of digitally coded prints.

To compare two prints so coded, a computer simply needs to compare the two strings of digits. However, bearing in mind the presence of possible topological mutations we could expect two or three of the digits to have been altered. Moreover, if the print was distorted the line drawn might cross some of the ridges in significantly different positions to its previous intersections. These changes may, in effect, introduce new points to the set of points of intersection. That would shift part of the sequence along by two places (remembering that there are two digits for every ridge crossed).

Code Description.

0. The ridge goes out of sight without meeting any characteristic.
1. Not allocated.
2. Ridge meets a bifurcation as if from left fork.
3. Ridge ends.
4. Ridge meets a bifurcation as if from right fork.
5. Ridge returns to its 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.
9. Not allocated.

A. Ridge encounters scarred tissue.

B. Ridge encounters blurred or unclear print.

C. Ridge meets a compound (e.g. a cross-over).

D. Not allocated.

E. Not allocated.

F. Used for vector padding.

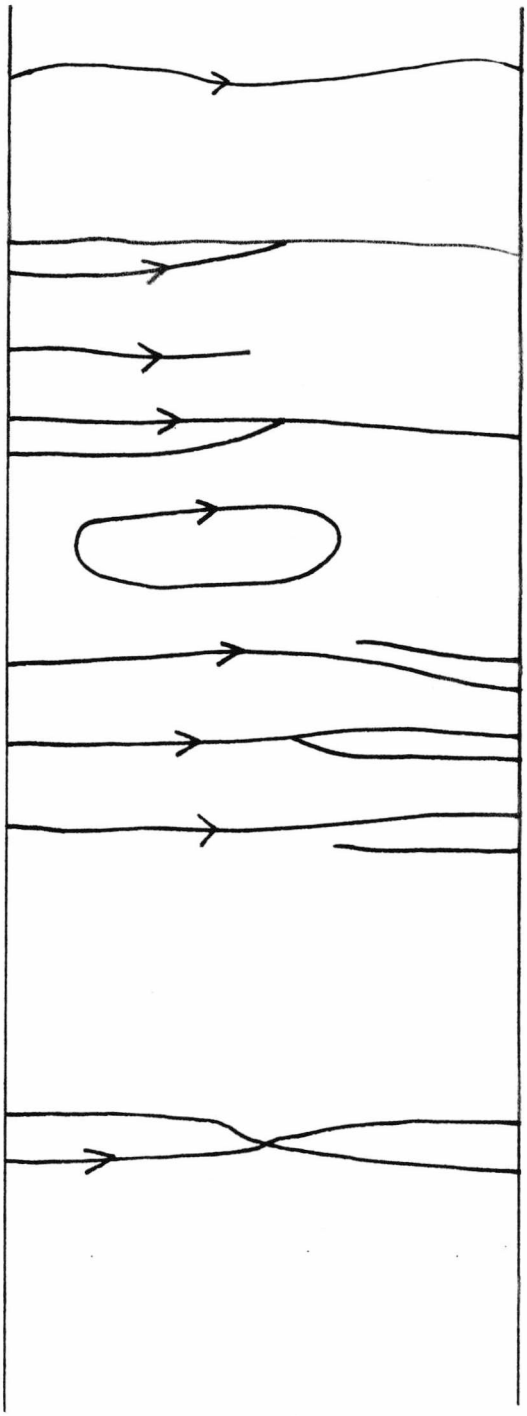


Figure 19. Table of ridge exploration event codes.

So, given a fairly severe set of mutations, a print that had been coded :

03 03 07 47 26 46 48 03 33 05 might, if badly distorted, appear:
03 03 33 07 47 26 06 26 03 05.

We would need the computer, therefore, to be able to recognise such facts as that these two codes have 7 pairs of digits in common (03, 03, 07, 47, 26, 03 and 05) — and 3 blocks of four digits in common (0303, 0747 and 4726) — or 1 block of six (or 6-block, as it will be known) in common, namely 074726.

Any one of these observations above is statistically most unlikely to arise in the case of incorrect matches. One block of six alone represents six similarly placed characteristics — enough for us to know that we have found the right print. The chance of a block of six recurring elsewhere by accident is extremely small.

The number of matching pairs, 4-blocks and 6-blocks can be determined in well under a hundred simple digital comparisons as follows :—

To allow for 'shifts' in the positions of the 2-blocks (due to appearance or disappearance of ridge intersections with the line) each pair of digits should be compared to the pair in the corresponding position, the pair one to the left, and one to the right. That can be done in 28 simple digital comparisons for the 10 pairs, 25 for the 9 4-blocks and 22 for the 8 6-blocks. (The ten pairs are each compared with *three* other pairs, except the two end ones which are only compared with two. Hence 28 comparisons. The 25 and 22 are similarly derived.)

This compares very favourably indeed with the spatial approach — where the number of operations currently required to compare two prints is of the order of six million. That could make the topological method in the order of 60,000 times as fast!

The precise number of matching pairs, 4-blocks or 6-blocks needed to indicate a match remains to be determined by empirical experiment. Suffice it to say here that the method's power of resolution (ability to distinguish matches from non-matches) should be considerable.

3.7 The placing of lines.

The position of the line used in para 3.6 is by no means the only possible position for it. We could, for example, have extended it beyond the core of the loop on the opposite side to the delta — and recorded some information about that *far* side of the loop as well.

Alternatively we could forget about the delta altogether for now, and draw the line through the core at 90° to the *slope* or *direction* of the core. (See figure 20.)

Let us use this line, and take, firstly, five points of intersection on the left hand side of the

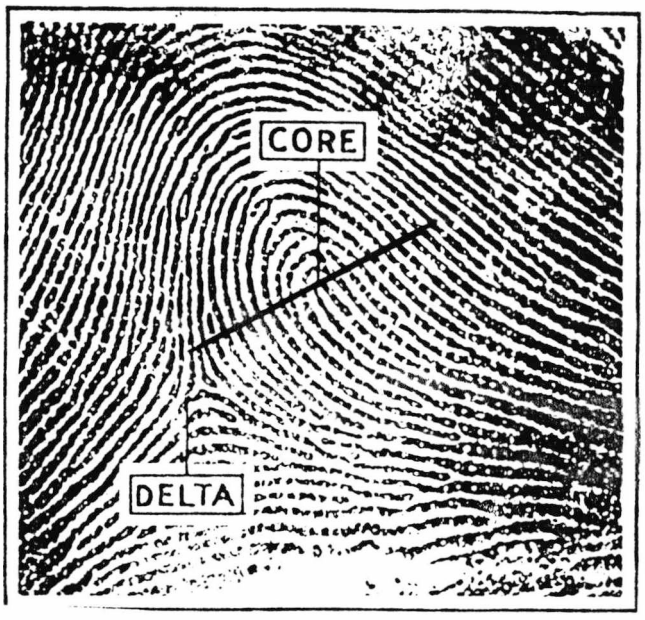


Figure 20. Line-placing orthogonal to core slope in a loop.

core (yielding ten digits) and then 5 points of intersection on the right hand side of the core (yielding a further ten). We would always use the *left* branch of a ridge before the *right* one, and *left* and *right* are determined as if you were looking away from the core. We always work outwards from the core.

Thus we can produce a 20 digit code of the core itself without reference to the rest of the print.

For figure 20, using the same digital codes as before — the core would give 42 46 22 27 36 (from the five ridges to the left of the core) and 70 32 74 20 83 (from the five ridges to the right).

By this method we can digitally code any core — whatever pattern it appears in — sufficiently to identify it uniquely.

3.8 Accuracy in placing lines.

The angling of the line is not critical, as it does not usually pass through areas of high characteristic density. The irregularities of the pattern tend to be concentrated more along the axis of the core. For this reason, movement of the line by even as much as 20 or 30 degrees will only change 2 or 3 digits of the code, and should preserve at least one or two *6-blocks* intact.

3.9 Coding deltas.

Likewise we can digitally code any delta, the only difference being that we draw three lines rather than one, and thereby generate 30 digits (assuming that we still proceed over five ridges in each direction). First we have to define a central point in the delta; in fact this has already been done for the purposes of ridge-counting. (See para 1.10.) Then draw three lines radially outwards from that central point, each roughly perpendicular to the flow of the ridges in that sector — as per figure 21.

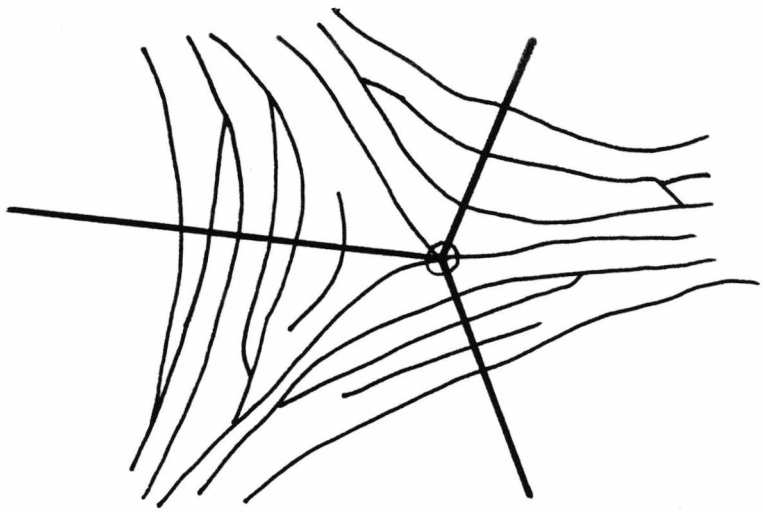


Figure 21. Line-placing on a delta.

Thus 30 digit codes for deltas can be generated — again, enough information being stored to make each quite unique.

3.10 Coding the various types of pattern.

With these techniques for coding cores and deltas lengthy digital codes could be prepared for all the various types of print that have either cores or deltas — that is, *all* prints with the exception of plain arches. They, as previously pointed out, have neither core nor delta.[31]

3.11 Coding plain arches.

The problem presented by 'arches' can be solved by using a central line as illustrated in figure 17. There is, however, no well defined starting place (as we would have with a core or delta). However let us start at the bottom of the line drawn on figure 17

(where the ridges begin to curve upwards) and work our way upwards; we can generate one long continuous string of digits by a similar method to those described above. (Simply look *left* and *right* along each ridge crossed etc.)

The line in figure 17 crosses 28 ridges, and therefore yields the series of 56 digits :—

04 43 22 34 03 03 42 22 22 42 24 00 33 33 00 30 43 26 28 02 37 83 62 04 72 34 33 36

Each one of these 56 digits represents one piece of topological information — each being the coded reply to the now familiar question "What is the first characteristic that you would find if you *walked along* each of the 28 ridges crossed — in the specified direction?".

Had the arch been coded using a slightly different starting point (and there seems to be no practical way of assuring that it would be the same each time), then the order of the majority of these digits will not be changed — but they would be shifted a few places to the left or right by reason of the addition or omission of some digits at the beginning of the series. For comparison purposes the computer will again have to check for matching pairs, 4-blocks and 6-blocks. [Rather than try *all* the very many possibilities it would be a short cut for the computer to align the two sets of digits for comparison by reference to the positions of appearances of the rarer codes — i.e. 'C's (compounds) if there are any.] By this method it is quite possible to identify plain arches automatically — and thus the particular problems posed by this class of prints are solved.

3.12 Scenes of crime marks.

The 'fishbone' method described above clearly makes possible the rapid automated identification of single fingerprints, provided we have available on each print to be identified certain localised areas of information; for instance, to identify a loop, we will need a fairly clear impression of areas either close to the core or close to the delta.

The application of this method, therefore, to scenes of crime marks is restricted to those marks where one of these particular areas appears, and is clear enough to be digitally coded for comparison. The following two chapters examine ways of dealing with scenes of crime marks that contain no such focal points.

3.13 Advantages of the 'fishbone' method.

The fishbone method (so called because of the similarity of the ordering lines to the principal backbone of a fish, with the ridges to be explored fanning out from the points of intersection) overcomes all four of the difficulties stated earlier in connection with 'progressive association of characteristics from a fixed point'. It can be applied to arches, it records more localised information, it never 'runs out', and it does allow for topological variation.

As such it represents a highly effective method of coding and recording topological information. The coding can be done fairly quickly by hand, but it was always intended that it should be done by automatic scanners. All the technology is currently available, and in use, whereby such data could be gathered automatically. No new equipment is needed — just a few new algorithms.

CHAPTER 4.

NON-ORDERED CODING SYSTEMS.

4.1 Introduction.

The aim of this research has always been to find coding methods suitable for automation that could do everything that manual fingerprint systems can do at the present time.

Chapter 3 manifestly falls short of this goal; none of the methods described there can cope with the matching of partial scenes of crime marks that contain neither cores, deltas nor the central strip of an arch pattern.

Such prints are currently identified by manual systems. In fact roughly 5% of successful identifications made by the Kent Fingerprint Department are of this type. This 5% is a significant enough proportion to be catered for, even if it involves a substantial amount of further research.

By way of example, figure 22 shows one fingermark from the scene of crime which has been identified (in Kent) with a portion of the left thumb impression of a known criminal. The SOC mark is on the left and the portion from records on the right.

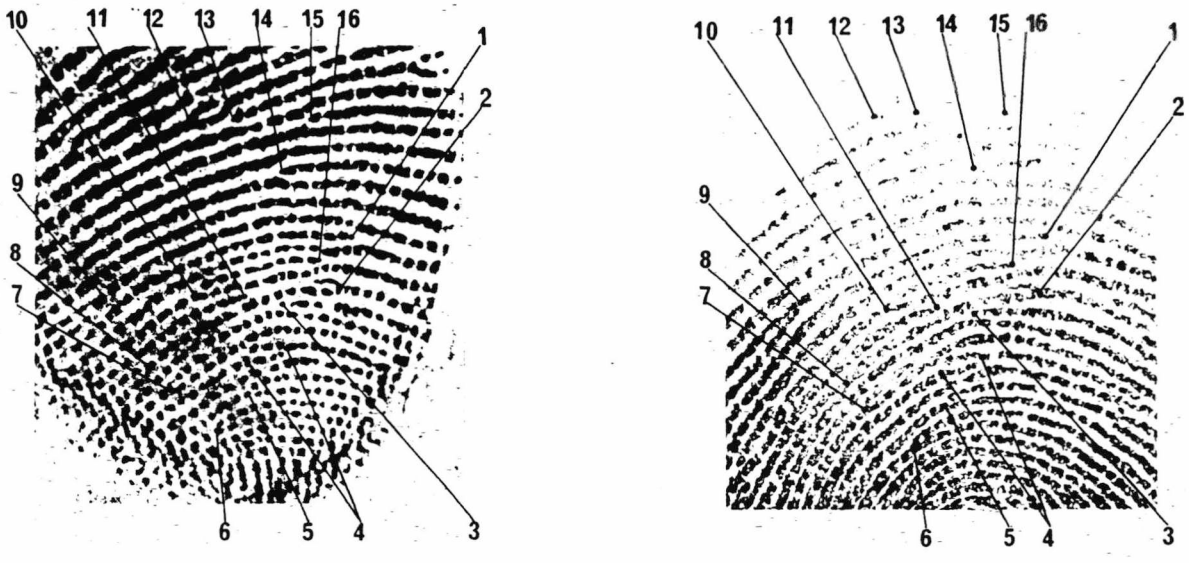


Figure 22. A latent mark and its matching file print.

The mark in figure 22 contains a wealth of topological information but no central reference points such as we have been using. We clearly need something more sophisticated than the ordering methods described so far if we are to be able to deal with those occasions when we have no core or delta to work from.

4.2 The need for non-ordered systems.

The purpose of chapter 3 was to find methods of ordering topological information so that, when coded, it appeared as ordered digital codes. Such orderings were determined by reference to the print as a whole, or by reference to some particular parts of it. They relied on cores or deltas to provide a framework within which to work — the only exception being the coding of an arch, where the ordering required us to be able to see the pattern or shape of the print, without which we would not have known where to place the line.

Now the special problem posed by the scene of crime mark that contains none of our fixed reference points (cores or deltas) is that even if the characteristics appearing on it had been coded, we might not be able to glean enough information from the mark about the rest of the print to find out *in what order* those particular pieces of information had been coded.

Here is a very simple example for demonstration purposes: suppose a scene of crime mark appeared as represented by figure 23.

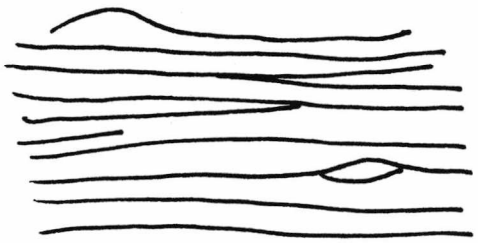


Figure 23. Tracing of fragmentary latent mark.

This contains four characteristics — 2 bifurcations, a ridge ending and one lake. They have certain topological relationships to each other, for example :—

"The bifurcations face in opposite directions and are on adjacent ridges. Also you can travel from the lake to the ridge ending by moving along the ridge to the left and then jumping over one ridge".

All that is good topological information, invariant under distortion. However if it had been coded as part of the coding of a whole print, the codes would probably have

been ordered by reference to the position of these characteristics in relation to the core, or delta. The only information we have available in figure 23 is their topological relationships *to each other* — and not to any other feature of the print. Hence any digital coding methods ordered by reference to fixed points or pattern types are useless in this context.

What alternatives do we have then? How else can we record the information? Broadly speaking, there are just two possibilities :—

Firstly: we could use a completely *unordered* system — which, as will be explained, involves the use of matrices and fairly complicated matrix searching techniques.

Secondly: we can use *unordered* collections of *locally ordered* information.

Both of these possibilities have been investigated, and it has become clear that the former leads us to searching methods that are neither quick, nor simple. In retrospect it must be rejected (at least for the time being) in favour of the second, which appears to be more workable.

Consequently, the former will be described in detail in the remainder of this chapter, purely because it contains some interesting ideas that would not appear elsewhere.

Chapter 5 contains a detailed explanation of the latter (and more hopeful) approach.

4.3 The two dimensional nature of completely unordered systems.

The benefit of ordering *by line* was that codes appear as strings of digits; these can be regarded as one-dimensional arrays of numbers.

Now, if we choose to record each characteristic's relationship to every other characteristic we cannot avoid storing that information as a two-dimensional array of digits, rather than a one-dimensional array.

Non-ordered systems, therefore, concern us with the use of matrices.

4.4 The need for physically denser information recording.

A scene of crime print may be comparatively small in area, and the smaller it is the less likely we are to find direct 'links along ridges' (of the type used in chapter 3) between two visible characteristics. (Figure 23 has *no* such links.)

However if we widen our scope slightly we notice that we can often find two characteristics separated by only one or two ridges (as in figures 22 and 23). To extract the

maximum possible topological information from such a mark we must therefore be prepared to code *these* pieces of information as they tend to be physically denser within small areas. If we ignored them, the fact that figure 23 'has two bifurcations, a ridge ending and a lake with no direct topological links along ridges' would be nowhere near enough to identify the print from which it came.

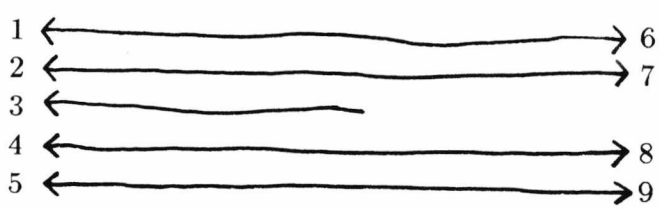
4.5 Relationship tables.

Let us now find a way to code the relationships of characteristics to one another. For this purpose, and for the sake of simplicity, we will only recognise bifurcations, lakes and ridge endings as being the 'basic' types of characteristics — and will view the others as compounds. The task we have is to devise a set of rules for digitally coding the type of topological relationships discussed in para 4.2. That is not difficult — but it has to be done separately for each of the three types of characteristic under consideration.

The ridge-ending. If there is another characteristic near to a ridge ending we could ask ourselves the question "by which, if any, of the following routes can we travel from the ridge ending to that other point?" :—

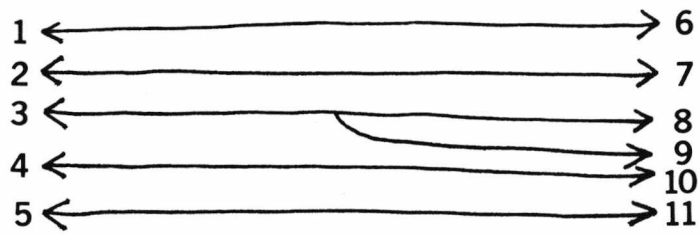
1. Jump two ridges to the left and then go backwards along that ridge.
2. Jump one ridge to the left and then go backwards along that ridge.
3. Just go backwards away from the ridge ending without jumping.
4. Jump one ridge to the right and then go backwards.
5. Jump two ridges to the right and go backwards.
6. Similarly, two ridges left, but forwards.
7. Similarly, one ridge left and forwards.
8. Similarly, one ridge right and forwards.
9. Similarly, two ridges right and forwards.

These rules can be shown thus :—

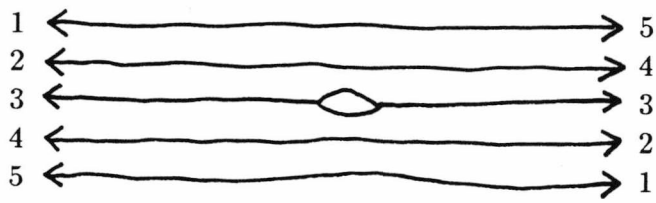


This shows the nine routes that we will consider leading away from a ridge ending. If none of the above routes lead us to the other characteristic, then a zero coding would be used.

The bifurcation. Similarly a set of routes *away from a bifurcation* can be constructed, as represented by the following diagram :—



The lake. Likewise — the codes being as follows :—



Notice the rotational symmetry in this last case — as a lake does not determine 'direction' for us, and we may not be able to tell from the scene of crime mark which way the print was orientated. Also notice that it may be possible to arrive at a particular bifurcation by more than one of the routes specified — in which case the lowest appropriate code will be chosen.

Now let us have another look at figure 23.

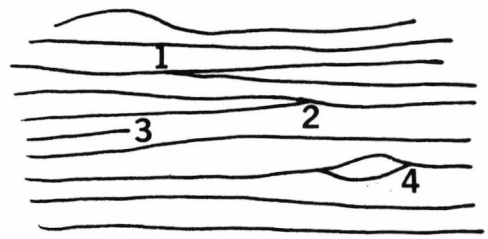


Figure 24. Characteristics on a latent mark.

The points 1, 2, 3 and 4 are respectively the bifurcations, the ridge ending and the lake, which we will code 'B', 'B', 'E' and 'L'. We can then draw up a table of the

topological relationships between these 4, using the route-codes described above :—

	To			
	B	B	E	L
B(1)	/	10	0	0
B(2)	10	/	0	1
From E(3)	0	6	/	9
L(4)	0	1	0	/

This tells us, for example, that to get from point 3 (ridge ending) to the lake, one follows the route no.9 specified on the 'ridge ending' diagram. All the other entries convey similar information. The digits underlined involved the choice of 'lowest code' from two possible routes (e.g. one can get from point 2 to point 1 by using either route 10 or route 11 on the 'bifurcation' diagram.)

The leading diagonal is free, as it would be meaningless to speak of a characteristic's topological relationship with itself. So we can slot the characteristic type code into that gap and obtain the simple four by four matrix :—

$$\begin{pmatrix} B & 10 & 0 & 0 \\ 10 & B & 0 & 1 \\ 0 & 6 & E & 9 \\ 0 & 1 & 0 & L \end{pmatrix}$$

If we had seen ten characteristics on the scene of crime mark we could have constructed a 10 by 10 matrix in the same way.

Remember that this is an *unordered* system, and we might have taken the four characteristics in a different order. Had we done so, the coding process might have yielded the table :—

$$\begin{pmatrix} L & 0 & 1 & 0 \\ 0 & B & 10 & 0 \\ 1 & 10 & B & 0 \\ 9 & 0 & 6 & E \end{pmatrix}$$

This is the same table — except that the order of the rows and columns has been changed.

4.6 The searching problem.

Suppose an entire single print collection had been similarly coded. On average a single print reveals in the order of 100 characteristics, and so their relationships to each other would be stored in coded form as a series of large matrices, roughly 100 square. (These are effectively 'sparse matrices' as the vast majority of entries would be zero.) The problem of searching for a match with a mark from the scene of crime becomes analogous to the problem of searching large matrices to see whether a given smaller matrix lies within it.

Suppose, for simplicity's sake, that figure 23 is actually part of a loop pattern which only showed up 20 characteristics when coded, giving the following 20 by 20 matrix:—

$$\begin{pmatrix}
L & 2 & 5 & 1 & 7 & 0 & 6 & 0 & 0 & 0 & 8 & 9 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & \mathbf{L} & 0 & 5 & \mathbf{0} & 2 & 1 & \mathbf{1} & 8 & 0 & 0 & \mathbf{0} & 4 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
3 & 0 & L & 1 & 2 & 4 & 0 & 0 & 0 & 0 & 8 & 9 & 0 & 0 & 0 & 6 & 0 & 0 & 5 & 0 \\
4 & 2 & 0 & E & 1 & 0 & 5 & 0 & 0 & 0 & 0 & 0 & 9 & 0 & 0 & 7 & 0 & 0 & 0 & 3 \\
4 & \mathbf{0} & 0 & 0 & \mathbf{B} & 1 & 0 & \mathbf{10} & 0 & 0 & 0 & \mathbf{0} & 0 & 0 & 6 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & B & 0 & 0 & 0 & 6 & 0 & 0 & 0 & 7 & 0 & 0 & 0 & 4 & 0 & 0 \\
1 & 3 & 4 & 0 & 0 & 0 & B & 0 & 0 & 7 & 0 & 0 & 0 & 0 & 8 & 0 & 0 & 0 & 6 & 0 \\
0 & \mathbf{1} & 0 & 0 & \mathbf{10} & 0 & 0 & \mathbf{B} & 0 & 0 & 0 & \mathbf{0} & 3 & 2 & 4 & 0 & 0 & 0 & 0 & 7 \\
3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & E & 0 & 0 & 7 & 0 & 0 & 5 & 0 & 0 & 0 & 2 & 0 \\
0 & 0 & 0 & 4 & 0 & 0 & 0 & 0 & 0 & E & 0 & 0 & 0 & 0 & 7 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 6 & 9 & 8 & 0 & 0 & 0 & B & 3 & 0 & 0 & 0 & 5 & 0 & 0 & 0 & 0 \\
0 & \mathbf{9} & 0 & 0 & \mathbf{0} & 0 & 0 & \mathbf{6} & 0 & 0 & 0 & \mathbf{E} & 8 & 0 & 0 & 0 & 5 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 6 & 5 & 9 & 0 & 0 & 0 & 0 & 0 & E & 4 & 0 & 0 & 0 & 7 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & B & 0 & 7 & 9 & 5 & 0 & 0 \\
0 & 0 & 6 & 4 & 2 & 0 & 8 & 9 & 7 & 0 & 0 & 0 & 0 & 0 & B & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & B & 5 & 3 & 6 & 7 \\
3 & 1 & 4 & 6 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 7 & 9 & 0 & 0 & 0 & B & 0 & 0 & 0 \\
4 & 0 & 0 & 0 & 0 & 7 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 8 & 0 & 0 & 0 & B & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 8 & 9 & 7 & 5 & 6 & 0 & 0 & 0 & E & 0 \\
0 & 0 & 0 & 0 & 0 & 7 & 8 & 5 & 0 & 0 & 0 & 3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & B
\end{pmatrix}$$

The figures in bold type reveal the presence of our 4 by 4 matrix for figure 23.

Finding economical techniques (algorithms) for identifying these sub-matrices is a formidable task and the need to be able to identify *close matches* as well as exact ones (because of the possibility of topological variation) greatly compounds the difficulty.

The mammoth searching problems thrown up by such a coding method may not be insoluble — but there is no obligation to solve them here as the alternative approach is far more appealing.

CHAPTER 5.

PARTIALLY ORDERED SYSTEMS.

5.1 Introduction.

Chapter 4 has shown us just how lengthy, and therefore costly, a procedure the use of a completely unordered system might be. The system described there recorded the maximum possible amount of topological information — in recording every characteristic's relationship with every other one.

Chapter 4 leads us to realize that this is simply *too much* information. Some of it has to be discarded. The question we have to answer now is, 'Which information can we discard? Which of the topological relationships are of least value to us in identifying scenes of crime marks?'.

It would seem sensible to discard the information about each characteristic's relationship with all those other characteristics that are a *long way* away (physically) from it, and to record only localised relationships. Why so? The reason is that the further apart (physically) two characteristics are, the more likely their mutual relationship is to suffer topological change (i.e. intervening ridges appearing or disappearing and so on) and, more importantly, the less likely both are to appear on the same scene of crime mark. Consequently if two characteristics are far apart their relationship is of little or no use, and would appear usually in the matrices of chapter 4 as yet another zero code.

It is these factors that lead towards consideration of localised orderings.

5.2 Local orderings.

Any non-symmetrical characteristics (i.e. bifurcations and ridge endings — but not lakes) could be used as focal points for a local ordering in much the same way as deltas and cores were used as focal points for general orderings.

For the time being let us use bifurcations, and bifurcations alone — as the basis for forming collections of local orderings. (The choice is arbitrary. It could have been ridge endings, but it would seem superfluous to use every visible characteristic.)

Wherever a bifurcation appears, other than within 5 ridges of a core or delta, let us draw a line through it, at 90° to the ridge flow (in much the same way as we did for

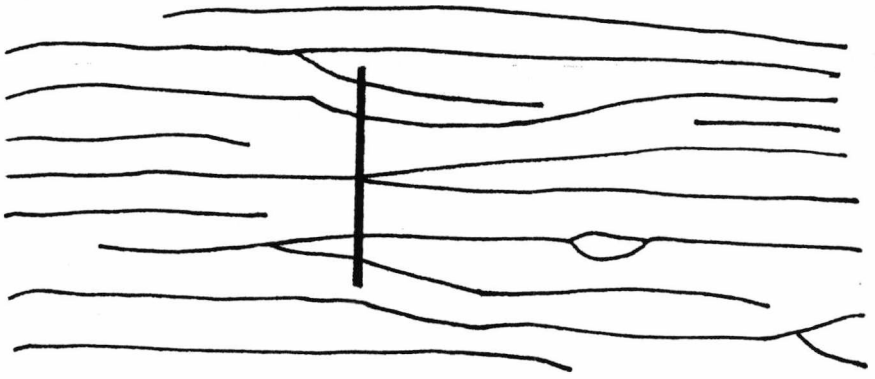
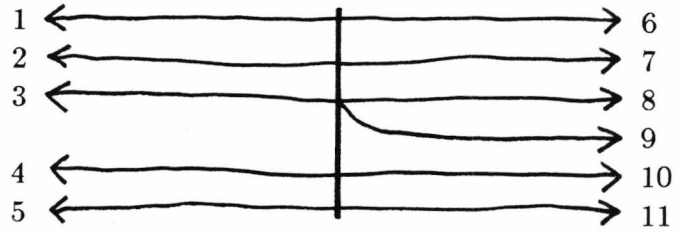


Figure 25. Line placing for local ordering around a bifurcation.

the core in para 3.7) and let it extend just two ridges either side of the bifurcation, as illustrated in figure 25.

Now, by looking left and right along the *five* ridges crossed we can, once again, code the replies to the question 'what characteristic do we come to first?', and thereby generate an eleven digit code for that bifurcation. (There are eleven ridges to look along altogether, rather than ten, due to the extra 'prong' of the bifurcation itself.)

If we use the same digital codes as we specified in para 3.5 for the possible 'events', and take the eleven ridges in the order represented below :—



then the bifurcation used as focal point in figure 25 gives the code 46624386073.

That eleven digit code represents the ordered topological information most local to that bifurcation. It is a coding of the bifurcation itself in relation to its own immediate neighbourhood.

5.3 The use of unordered collections of local orderings.

We now have a method of ordering topological information locally without any reference to fixed central points of the whole print pattern.

Suppose, when coding a print, that we picked out every bifurcation which appeared on the rolled impressions (except those very close to cores or deltas) and coded their

relationships to their immediate neighbourhoods; then we would have a collection of 11-digit codes, one for each bifurcation spotted.

The problem of finding matches for scenes of crime marks that contain no cores nor deltas is therefore solved by picking out the visible bifurcations on the scene of crime mark, coding them in relation to their immediate (visible) locality, and then comparing those codes with the 11-digit 'bifurcation-codes' in our database.

We would use the rule that a B, A or 0 in the scene of crime mark code is equivalent to *any* digit in the corresponding position — as characteristics may be *unclear* or *out of sight* on the mark whilst they were quite distinct on the rolled impression. The finger may also have been scarred since the rolled impression was taken.

On the scene of crime mark shown in figure 22 there are roughly 10 visible bifurcations. We'll choose one roughly central in the mark, as such will be the most useful for searching purposes (their neighbourhoods will not be 'out of sight' if they are central enough).

The bifurcation labelled no.10 in figure 22 gives the code 00007303838 on the scene of crime mark and gives *exactly the same* code on the portion of the rolled impression. That is sufficient information for positive identification of the mark.

The code 00007303838 would be one of, perhaps, 25 or 30 such codes for 'outlying' bifurcations spotted on the complete rolled thumbprint. The search for a match would be by simple, straightforward comparison of the scene of crime mark code with each bifurcation code in the database, in turn.

If no matches were found (perhaps due to topological variation) another search could be done very easily by coding up one of the other bifurcations visible on the scene of crime mark. Also, searching by a second 11-digit code could be used if the first search threw up several possible matches and distinction between them was required.

5.4 Summary.

This method of locally ordering information should never be used in isolation : the first choice is always to use cores or deltas as fixed starting points. Those are, after all, few in number (at most two cores and three deltas to a print) and therefore provide the fastest possible search.

A complete record in an automated system based on these ideas would comprise the following elements :—

(a) Descriptive (demographic) data relating to the individual — to be used for limiting the field of search.

(b) Traditional pattern type classification, if known.

(c) (i) In the case of an arch — one long sequence of digits as per paragraph 3.11.

(ii) Otherwise — for each core a 20-digit code as per paragraph 3.7 and, for each delta, a 30-digit code as per paragraph 3.9.

(d) An 11-digit code for each outlying bifurcation as per paragraph 5.2.

Such a system would be capable of performing all those tasks currently accomplished manually. It could effectively identify any scene of crime mark whether or not it was possible to tell from which part of the print, or from what type of pattern, the mark came.

It was mentioned previously (paragraph 4.1) that currently 5% of successful identifications do not depend on the presence of cores or deltas. The system described above — with the use, if need be, of the localised bifurcation codes — not only caters for that 5%, but might vastly increase the proportion of times when such marks are matched. There is no doubt that many more such marks *could* be matched, if only the present searching methods did not take so long.

CHAPTER 6.

OTHER RIDGED AREAS OF THE BODY.

6.1 General extension to other ridged areas.

The purpose of this brief chapter is to point out that the use of the methods developed in chapters 3 to 5 is by no means restricted to the coding of the ridge pattern on the fingertips. They can be applied to any part of the body where ridge patterns are found — for example, on the second and third joints of the fingers, on the palms of the hands, the heels and the toes.[32]

In order to use the 'fishbone' method of chapter 3 on any area one needs to either :

(a) isolate cores or deltas and code the locality of these, or

(b) determine the placing of a physical line across the ridge flow by any means whatsoever, whether spatial, topological or anatomical (e.g. by fixing the line's position in relation to flexion creases), and then code from that line as we did for a plain arch.

To be able to use the locally ordered methods of chapter 5 we need only specify a restricted physical area for coding and to code all the bifurcations visible therein.

This opens up many possibilities for the keeping of toe-print indices, heel-print indices and so on – in each case with the intention that scenes of crime marks from those areas of the body should be rapidly checkable against sizeable computerised collections.

6.2 Palmprints.

Palmprints merit particular attention here on the grounds that they are already extensively recorded and stored, but efforts to classify them have been, on the whole, inefficient and unusable. There are very few palmprint collections that are used for any other purpose than the checking of scenes of crime marks against those of a known suspect. Little actual *searching* of them is done.

The attempts to classify them have embodied the same approach as the conventional attack on fingerprints by 'pattern type'. The vast range and lack of regularity in patterns that appear on palmprints makes that task extremely difficult.

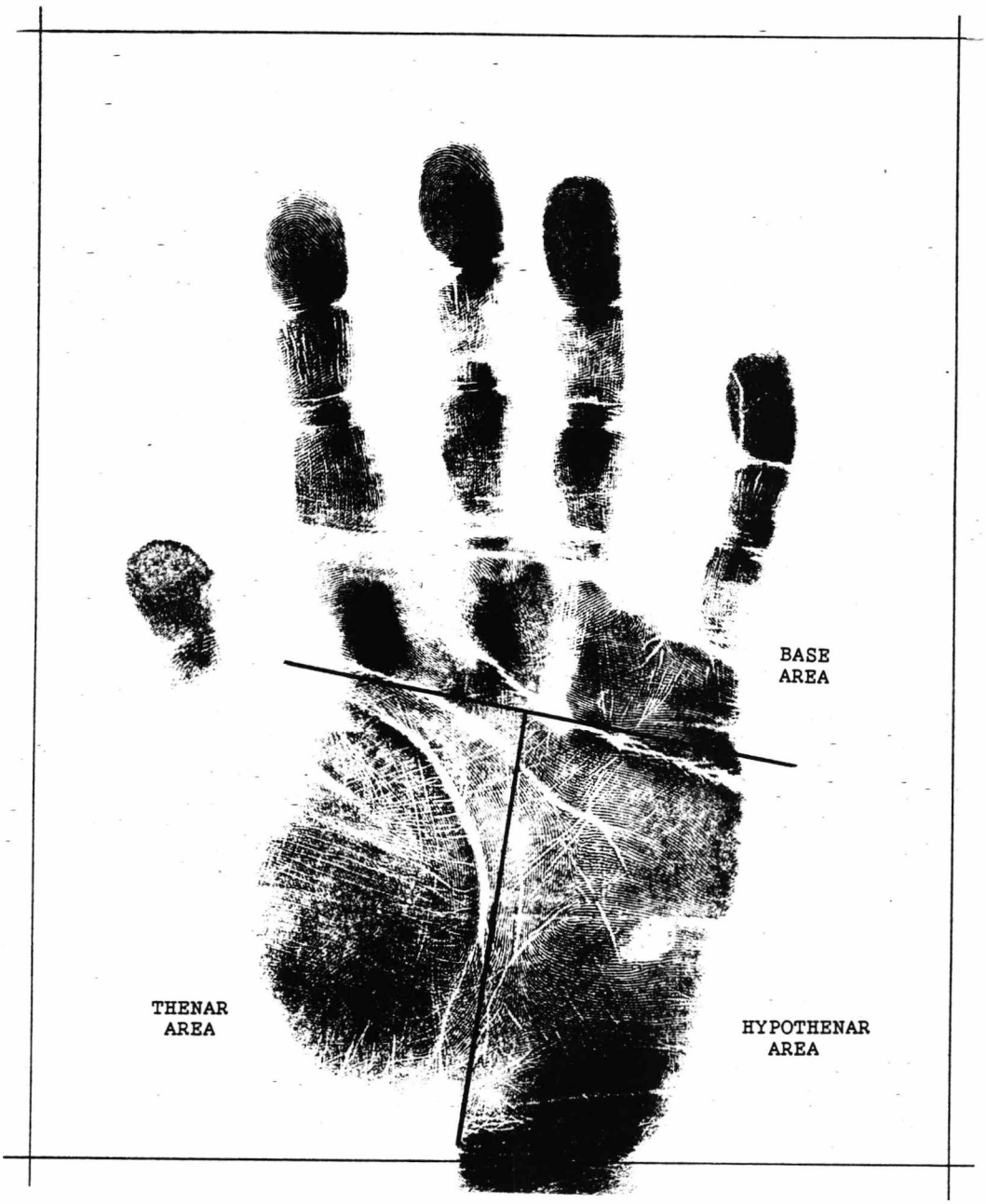


Figure 26. Thenar, hypothenar and base areas of a palmprint.

The palmprint itself can be divided into three distinct areas (these are the ones employed by the 'Brogger-Moller' pattern-type classification system)[33] : see figure 26.

The Brogger-Moller system, in common with other such systems, looks at the three areas in turn and tries to classify the types of pattern that occur therein. The three types of pattern represented on the three respective areas of the print are then formed into a combinatorial code, in a similar way to the combining of fingertip pattern types in the 10-finger Henry system. The categories of palmprint so formed are then broken down further by the recording of a variety of physical measurements, in a manner very similar

to the subclassification system of the Battley single print index.

Complete palmprints are hardly ever left at the scene of a crime; it tends to be part of the hypothenar area pattern that is usually left.[34]

Therefore an automated palmprint collection might well concentrate on the hypothenar area, and the following are simple suggestions as to methods of coding the topological information thereon :—

(a) Where there appears a delta (nearly always) code that as per paragraph 3.9.

(b) Where a loop or whorl pattern appears — code the core as per paragraph 3.7.

(c) Where neither (a) nor (b) above can be applied, draw a straight line across the hypothenar area from the flexion crease at the wrist to the 'outside' of the base of the 5th finger — and apply the method used for arches as per 3.11.

(d) Use the locally ordered method of chapter 5 on the whole or specified part of the hypothenar area.

Moving straight to topological coding of palmprints, heelprints, or toeprints completely avoids the need to classify the multitude of pattern types that do appear.

PART II

Topological comparison of rolled impressions

CHAPTER 7.

BACKGROUND, AIMS AND ANTICIPATED PROBLEMS.

7.1 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 recognise, 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 Part I.

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 and comparison techniques based on the more traditional (spatial) approach.

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

7.2 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 faced by 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 256 most likely candidates. Only then does computerised 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 authorisation verification) may well benefit from a quick and effective single-print comparison technique.

7.3 Selection of raw data rather than enhanced images.

The current processes 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 differentiate between ridge-endings and bifurcations. They simply identify the presence of a ridge-flow irregularity 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 of 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' ability to make the correct topological decisions.

7.4 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 65% 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 whorls and arches at a later stage.

7.5 Selection of line based system.

The stated aims of the work on rolled impressions (para 7.2) make it plain that a quick, easy coding method is sought. It should provide information sufficient to identify each single print uniquely, and should be easily reproduceable. The coding method selected was the 'ordering of topological information by lines' as described in chapter 3. 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-centre, using the conventional rules for precise location of the core-point.[35]

Figure 27 shows a typical ulnar loop pattern with horizontal line superimposed according to these rules.

These line placement rules are by no means the only possible for loops. There are innumerable possibilities; some centred 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.

7.6 Selection of digital codes.

Figure 28 shows the digital codes selected to correspond to possible ridge-exploration events (It is the same as figure 19, repeated here for convenience). 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.

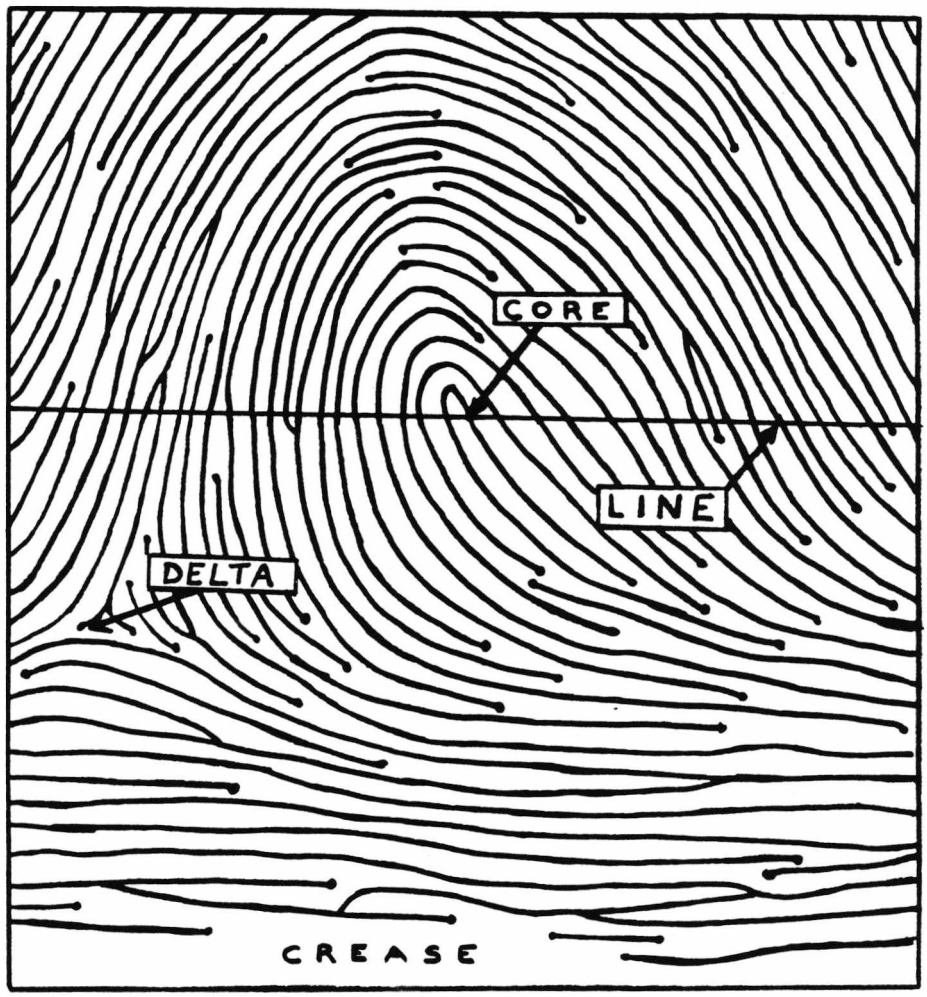


Figure 27. Horizontal line placed on ulnar loop.

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 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 standardised 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

Code Description.

0. The ridge goes out of sight without meeting any characteristic.
1. Not allocated.
2. Ridge meets a bifurcation as if from left fork.
3. Ridge ends.
4. Ridge meets a bifurcation as if from right fork.
5. Ridge returns to its 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.
9. Not allocated.

A. Ridge encounters scarred tissue.

B. Ridge encounters blurred or unclear print.

C. Ridge meets a compound (e.g. a cross-over).

D. Not allocated.

E. Not allocated.

F. Used for vector padding.

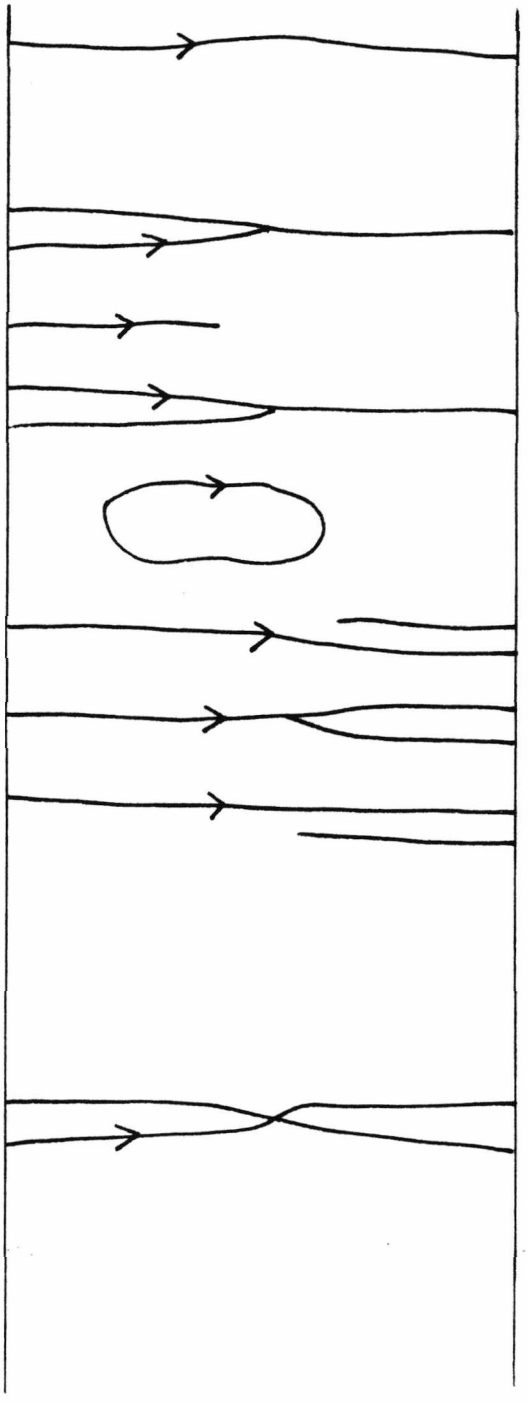


Figure 28. Ridge exploration event codes.

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 (see para. 3.4). 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.

7.7 Method and 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[36]). This projects an enlarged image (10× 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 centres 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 29 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

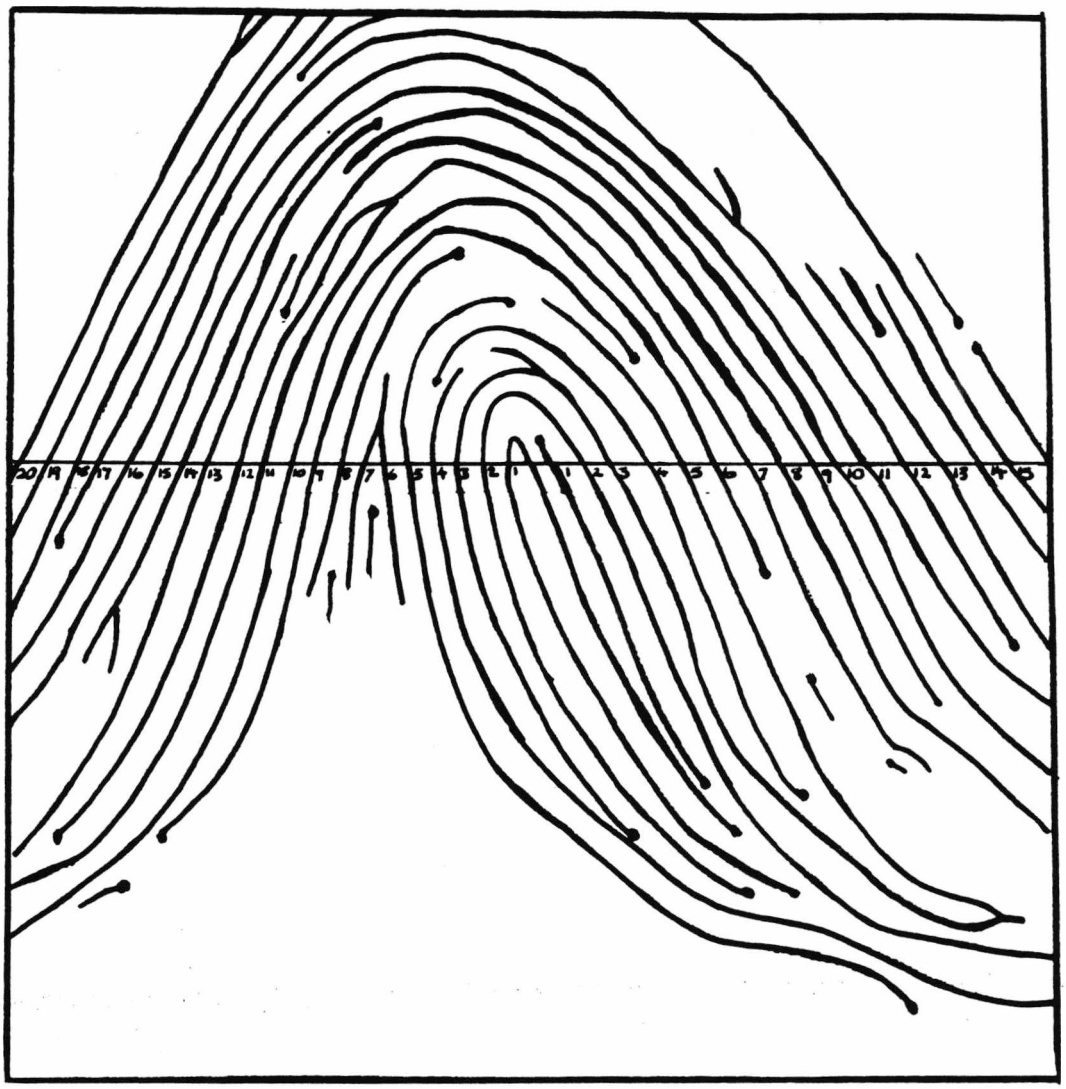


Figure 29. Loop tracing generated by coding process.

intersected on the right of the core. Whenever less than twenty ridges were intersected on the left or the right hand 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 29) is shown in figure 30. To facilitate interpretation the intersection point numbers (from figure 29) are shown also, 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.

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	80	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 30. Vector generated from figure 29.

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

7.8 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.

7.9.1 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.

7.9.2 Frequency analysis : results.

152 prints were coded according to the scheme described above (para 7.7). 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 intersected by the generating line within the confines of the central rectangle (0.78″ by 0.71″).

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

Figure 32 shows those frequencies divided into two classes — 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: appendix A deals with the ridges on the left of the core, looking downwards; appendix B with those same ridges but looking upwards. Appendix C deals with ridges on the right of

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 (on left)	1235	12.0
7	Ridge bifurcates ahead	1280	12.4
8	Faces ridge-ending (on 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 31. Global code frequencies.

	Frequency			*Percentage*	
Code.	*Upwards*	*Downwards*	*Code.*	*Upwards*	*Downwards*
0	419	1996	0	8.1	38.8
1	0	0	1	0.0	0.0
2	707	371	2	13.7	7.2
3	906	770	3	17.6	15.0
4	719	392	4	14.0	7.6
5	0	0	5	0.0	0.0
6	703	532	6	13.7	10.3
7	816	464	7	15.9	9.0
8	725	516	8	14.1	10.0
9	0	0	9	0.0	0.0
A	47	46	A	0.9	0.9
B	74	49	B	1.4	1.0
C	29	9	C	0.6	0.2
D	0	0	D	0.0	0.0
E	0	0	E	0.0	0.0
F	0	0	F	0.0	0.0
Total	5145	5145	*Total*	100.0	100.0

Figure 32. Upwards and downwards code frequencies.

the core, looking upwards and appendix D with the same ridges, looking downwards. In each case the ridges were divided up 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.

7.9.3 Frequency analysis : conclusions.

Detailed scrutiny of these tables can be interesting. For instance one observes, in appendix A, 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% (appendix D, ridge-band 16–20) to just 1.2% (appendix C, ridge-band 1–5).

There is only *one* general and important conclusion to be drawn from these results and it is this: 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.

7.10 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) Topological mutation : the changing of characteristic types from bifurcation to ridge-ending and vice versa (as explained in para. 7.6). 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.

(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 centre).

(d) Line placement errors: it would be foolish to expect orientation of the horizontal line to be exactly repeatable as it involved some subjective judgements. 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°. (In fact, when substantial numbers of mated prints had been coded using this scheme it was found that line orientation differed by less than 5° in 73% of pairs, by 5°–10° in 21% of pairs, by 10°–15° in 5% of pairs, and by over 20° 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 algorithm must be capable of recognising similarity between two vectors whilst 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.

7.11 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 — and 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 had 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 (nos. 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.*

CHAPTER 8.

DESCRIPTION OF BASIC MATCHING ALGORITHM.

8.1 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 act as a framework within which to understand all subsequent developments.

The algorithm described in para 8.2 is, in some particulars, comparatively crude: it is, nevertheless, surprisingly effective. It is called 'MATCH1', and will be referred to as such.

8.2 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.

8.2.1 Preliminary stage 1 — fileset analysis.

Suppose that the statistical analysis of para 7.9 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 re-evaluate 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 up into ridge bands.

The ridge-band width for this analysis is to be a parameter of the programme. (It was '5' when the tables in appendices A-D 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 analysed 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 (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 $\sum_j P(j,k,l) = 1.0$, for any fixed pair (k,l).

8.2.2 Preliminary stage 2 — setting up the 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 $i,j \in \{0,2,3,4,6,7,8,C\}$ then

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

where BOUND is another parameter — it is an imposed upper bound on the values taken by elements of the matrix S.

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), except for the case $i = j = 12$, 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 para 8.2.5.

(c) Paragraph 7.5 described the phenomenon of *topological 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 topological mutations.

Consequently if the comparison algorithm is to recognise *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 below. (Here the close match scores have been set to 2 and an upper bound of 15 applied. Also the exact match scores have been rounded to the nearest integer for ease of presentation.)

i \ j	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	1	1	1	0	0	0
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	0	0	8	2	0	0	0	0	0	0	1	1	1	0	0	0
3	0	0	2	9	2	0	0	0	0	0	1	1	1	0	0	0
4	0	0	0	2	7	0	0	0	0	0	1	1	1	0	0	0
5	0	0	0	0	0	15	0	0	0	0	1	1	1	0	0	0
6	0	0	0	0	0	0	10	2	0	0	1	1	1	0	0	0
7	0	0	0	0	0	0	2	11	2	0	1	1	1	0	0	0
8	0	0	0	0	0	0	0	2	9	0	1	1	1	0	0	0
9	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
A	1	0	1	1	1	1	1	1	1	0	1	1	1	0	0	0
B	1	0	1	1	1	1	1	1	1	0	1	1	1	0	0	0
C	1	0	1	1	1	1	1	1	1	0	1	1	1	0	0	0
D	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
F	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 and close match scores 2.

8.2.3 Comparison stage 1 — formation of file and search matrices.

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 para 7.10 subparas (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 as follows :

(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×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 row ...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 33.

(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 33 also shows such a *file matrix.*

8.2.4 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 value of r and 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 :

$$E(r,s) = S(i,j,k,l) \qquad \text{where} \quad \begin{array}{l} i = C(r,s) \\ j = D(r,s) \\ (k,l) \text{ are determined by } s. \end{array}$$

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'.

8.2.5 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) 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

```
          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 B0 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 B0 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 B0 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 B0 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 B0 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 B0 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 B0 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 B0 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 B0 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 B0 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 B0 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 B0 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 00 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 81 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 33. A search matrix, C, file matrix, D and initial score matrix, E.

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 33 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 $\big(S(r,s)\big)$ for each exact match (when $C(r,s) = D(r,s)$) were measures of the *unlikelihood* of such coincidence occurring 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 para 8.2.2) 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.

8.2.6 Comparison stage 3 — filtering for dependent pairs.

As explained in para 7.8 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 $[E(r,s) \times E(r,s+2)]$ is more weighty than is appropriate in view of that 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 wherever $E(r,s)$ and $E(r,s+2)$ are exact-match scores derived from a dependent pair then :

$$F(r,s) = \min(E(r,s), E(r,s+2))$$
$$F(r,s+2) = 2.0$$

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 para 7.9.2 (b). The step typically reduces about 2 entries per row of the matrix E.

8.2.7 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, whilst 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 :

$$G(r,s) = \begin{cases} 0 & \text{if } F(r,2s-1) \text{ and } F(r,2s) \text{ are } \textit{both zero}; \\ \text{Maximum}(F(r,2s-1), F(r,2s)) & \text{if } \textit{one}, \text{ and } \textit{only one}, \text{ is } \textit{non-zero}; \\ F(r,2s-1) \times F(r,2s) & \text{if } \textit{both are non-zero}. \end{cases}$$

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 34 (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.

```
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 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 00
71 01 01 01 00 00 00 00 00 00 00 40 00 00 00 00 00 0F 00 00 00 50 00 00 00 00 00 06 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 A0 00 00 00 00 00 00 00 00 00 06 00 02 00 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 00 00 A5 00 00 00 00 00 00 00 00 00 06 06 02 02 00 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 20 50 00 00 00 06 00 A2 91 00 00 00 00 00 00 00 00 00 00
00 00 01 01 00 00 00 00 00 00 00 00 F0 00 00 50 00 00 00 00 0F 00 00 50 05 00 00 00 00 00 01 00 00 00 00 00 00 00 00 00 00
00 00 00 01 00 00 00 00 00 00 40 00 00 00 F0 50 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 00 00 40 40 00 F0 00 40 00 00 00 20 50 00 00 00 00 60 00 00 06 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 90 00 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 00 02 05 00 00 00 A0 90 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  0 7150 10  0  0  0  2 15  0  6  0  0  6  0 30  2  0  0  0  0  0  0  0  0
  1  5  1  1  0  0  0 11  0  0 16  4  0  0  0  0  0    5  0 10  0  0  0  0  0  5  0  0  0  6  0  2  1  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  0  0 15  0  0  0  0  5  0  0  0  0  0 10  1  0  0  0  0  0  0  0
  1  1  1  1  0  0  0  0  0  0  0  2  4  0  0  0  0    0  0  0 10  0  0  0  0  0  0  0  0  0  6  0  2  0  0  0  0  0  0  0  0
  1  5  1  1  0  0  0  0  0  0  0  4  0  0  4  0  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  1  1  1  0  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  0  6  0 20  9  0  0  0  0  0  0
  0  0  1  1  0  0  0  0  0  0  0  0  0  0 15  0  0    5  0  0  0 15  0  0  0  0  5  5  0  0  0  0  0  0  0  1  0  0  0  0  0  0
  0  0  0  1  0  0  0  0  0  0  0  4  0  0  0  0 15    5  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  0  0  0  0  4  4 15  0  4  0    0  0  0  0  2  5  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  0  0  0  4  4  0 15  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  4  4  0  0  0 15    5 10  0  0  0  0  0  0 15  0  2  5  0  0  0  0 10  9  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 34. A filtered score matrix, F, condensed matrix, G and final score evaluation.

8.2.8 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 is also shown in figure 34.

8.3.1 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 enquiry against a file collection on a computerised 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 labour saving device.

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

BOUND = 50.0	(Upper bound on exact match scores)
MAXSHIFT = 5	(Maximum number of ridge-shifts anticipated)
CLOSE MATCH SCORES = 1	
BANDWIDTH = 5	(Ridge band width for frequency analysis)

the results were : MR1 = 90% (No. of mates ranked in first place.)
: LMR = 25 (lowest mate rank.)

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^{43}. The lowest mate score was 3.9×10^5. 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.

8.3.2 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.
BANDWIDTH = 2	
CLOSE MATCH SCORES = 0	

A complete table of parameters/performance for MATCH1 is given in appendix E.

8.3.3 Conclusions.

The fact that the parameters given in para 8.3.2 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.

(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 in appendix F. (This is part of the programme 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 topological 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 11 for comparison with a matching algorithm based on the traditional 'spatial' approach.)

8.4 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.

CHAPTER 9.

ALGORITHM DEVELOPMENTS: MATCH2 and MATCH3.

9.1 Need for new performance measures.

Testing MATCH1 with different parameter sets produced a variation in performance, as described in para 8.3. 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 programme 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.

9.2.1 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 relatively few data points (hopefully).

The proper way out of this apparent dilemma is to base performance measures on deductions about the behaviour 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.

9.2.2 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 logarithms (base 10) of the match and mismatch scores from MATCH1 are shown in figure 35.

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 36. Sadly the right hand tail (which is the crucial area) is the part of the distribution most badly fitted. Figure 36 shows an enlarged section of the right hand tail.

Fortunately one of the earliest amendments to MATCH1 (the *score-normalisation* procedures described in para 9.4.3) 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, behaviour

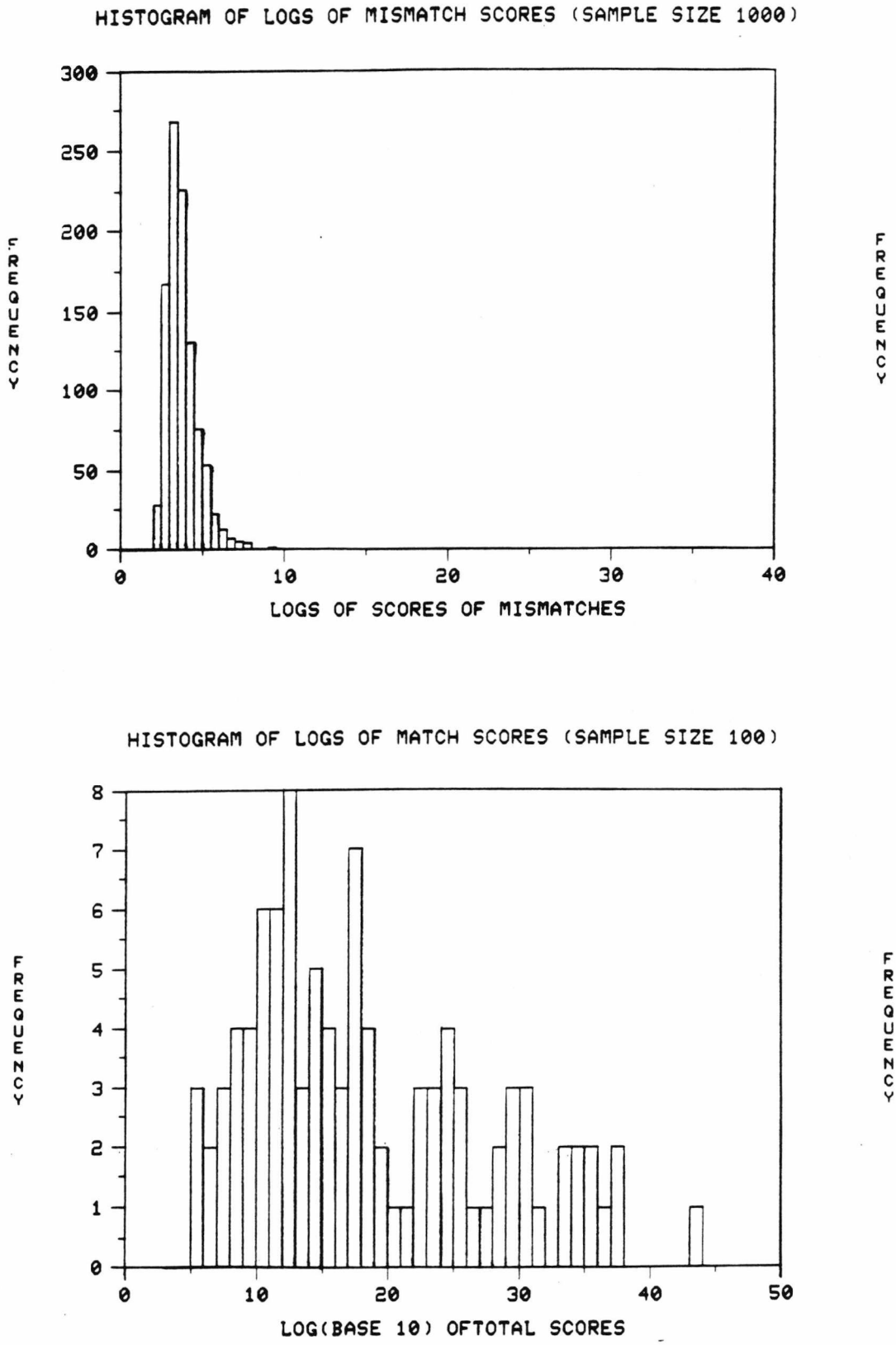


Figure 35. Histograms of $\log_{10}$ of match and mismatch scores from MATCH1

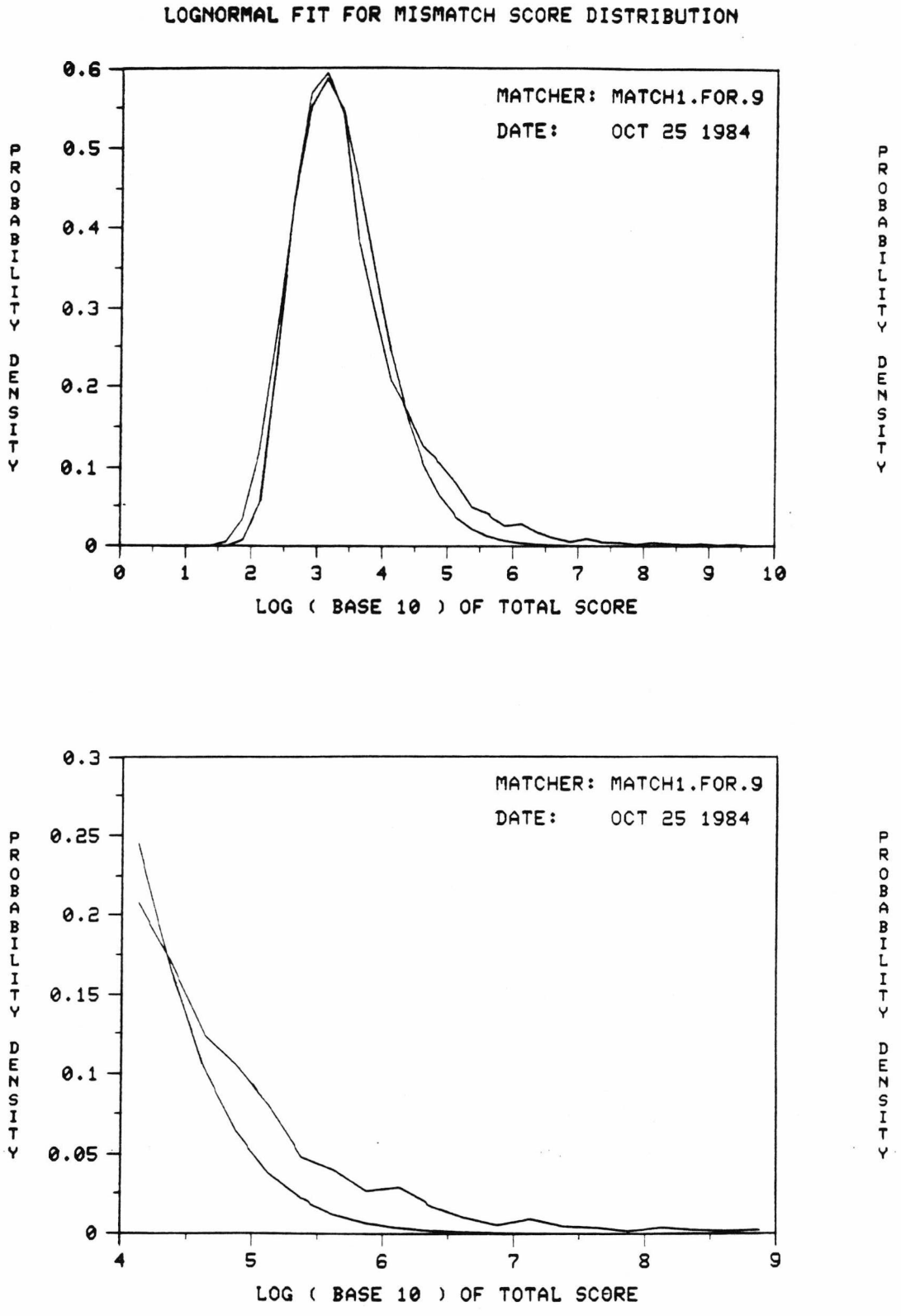


Figure 36. Lognormal fitting for mismatch score distribution with blown up portion showing the badly fitting right hand tail.

in the tails of score distributions could be estimated by use of non-parametric density estimation techniques[37] ; a non-parametric density function can be derived from the observations by summing a series of small *kernel* distributions centred on each observed value. The sum of the kernel functions approximates the underlying distribution. Practical use of this technique would be laborious — and its value in drawing inferences from just 100 observations would depend somewhat on a fairly arbitrary choice of kernel shape. Use of this technique might have been essential nevertheless had not the mismatch scores behaved 'nicely' in turning out to be lognormally distributed.

9.3.1 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 para 9.3.2

(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 para 9.3.3.

(d) The percentage of observed match scores exceeding the 99.9th percentile of the fitted lognormal mismatch distribution (P999) — see para 9.3.3.

9.3.2 Minimum total error (MTE).

Operational computerised 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 'substitution' and 'rejection' errors (known variously as type 1 and 2 errors, or as 'false drops' and 'misses' 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 37 for a pictorial representation of this.

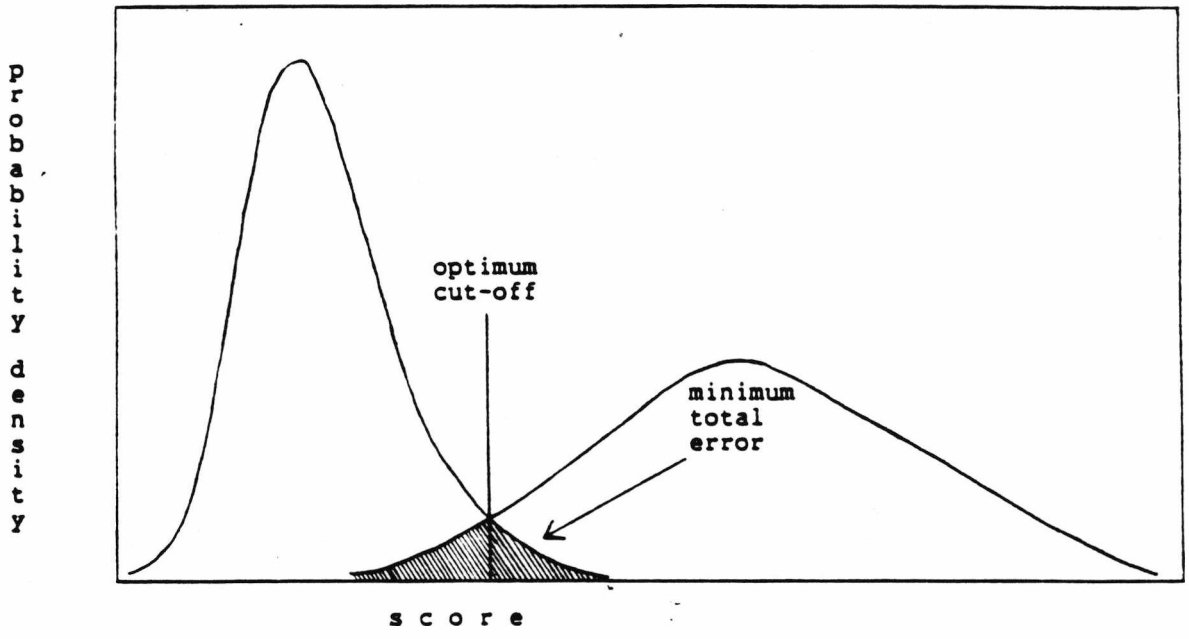


Figure 37. Theoretical match and mismatch curves showing optimum cut-off and minimum total error.

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.

9.3.3 P99 and P999.

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 enquiry (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 as least as high as the 99.9th percentile of the mismatch score distribution. Then the point of concern becomes the proportion of matches that 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.

9.4 Description of MATCH2 improvements.

MATCH2 used the same basic techniques as MATCH1, but several important modifications were made. (They are described in paras 9.4.1 to 9.4.4.)

9.4.1 Array operations made integer addition.

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 realised 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.

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 :

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

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.

(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 recognised 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.

9.4.2 Final score evaluation.

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 analysed as already described.

9.4.3 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-normalisation 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 normalised 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 normalisation) was an integer.

This normalisation 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 38 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.

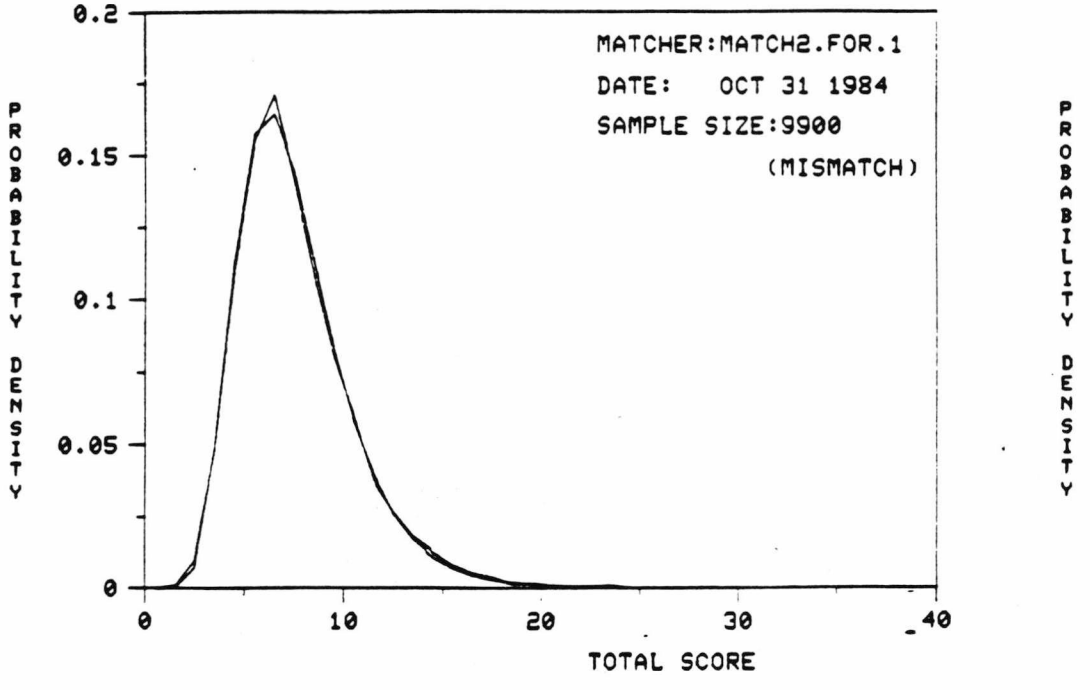


Figure 38. Lognormal fit for the mismatch scores from MATCH2.

9.4.4 '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 topological mutations (one on either side of the generating line) that just 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 to the row below (as shown in figure 39).

...	−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 39. Part of a condensed matrix showing a suitable 'hopping' place.

An intelligent algorithm would recognise this phenomenon, and would be able to put these broken strings back 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.

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.

9.5.1 MATCH2 performance on loops.

MATCH2 was run on TESTSET1 with a variety of different parameter sets. A table of results is shown in appendix G. (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 and 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.]

9.5.2 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 40.

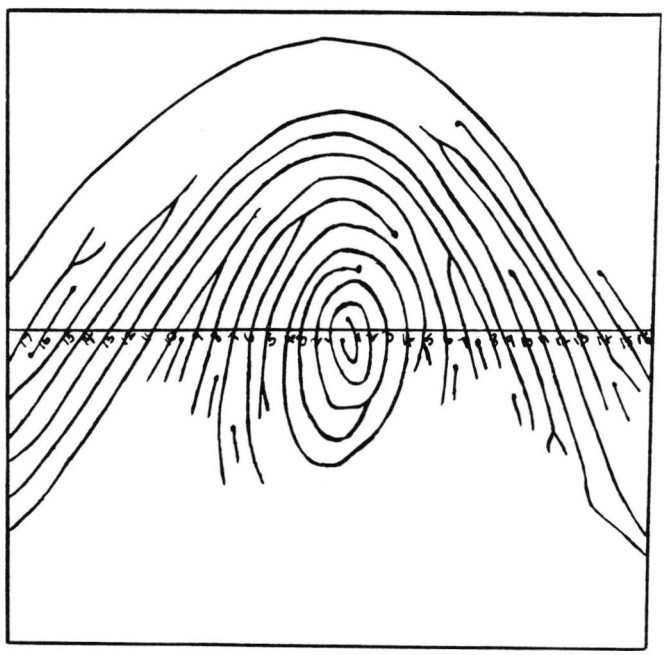


Figure 40. Sample tracing of whorl generated during coding process.

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 appendix G.)

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%).

9.5.3 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 41. 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 41.

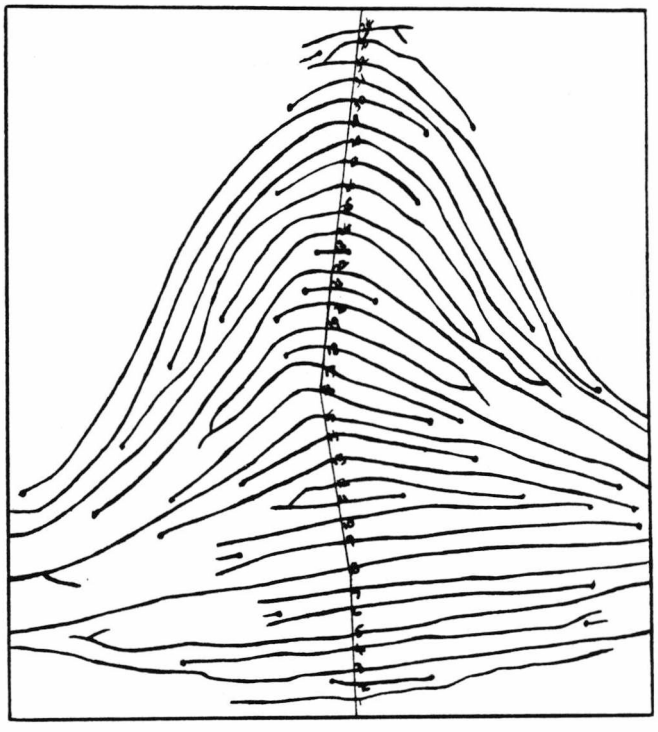


Figure 41. Sample tracing of plain arch generated during coding process.

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 appendix G (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.

9.6 Description of MATCH3 improvements.

The score normalisation procedure described in para 9.4.3 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 $B_1 \ldots B_n$. 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 $\overline{R}$. 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 para 9.4.3 gave a final score of:

$$\frac{T_{ij} \times 100}{Q_j}$$

which is a percentage.

Two different ways of incorporating R_i into this normalisation formula were tried. MATCH3 version 1 used the formula :

$$\frac{T_{ij} \times 100 \times \overline{R}}{Q_j \times R_i}$$

(where the ratio of R_j to the mean file perfect score ($\overline{R}$) is used as the second correction factor.)

MATCH3 version 2 used the formula :

$$\frac{T_{ij} \times 100}{\sqrt{Q_j \times R_i}}$$

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.

9.7 Performance of MATCH3 — versions 1 and 2.

The normalisation procedure used in MATCH3 version 1 seemed to overcompensate 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 appendix H.

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, been elusive.

CHAPTER 10.

THE INTRODUCTION OF DISTANCE MEASURES.

10.1 Motivation.

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 entirely 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

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.

10.2.1 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× 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 $0, 1, 2, \ldots 15$.

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.

10.2.2 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).

10.3 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.

10.3.1 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 becomes a parameter of the programme and is called the *absolute distance tolerance* (ADT).

10.3.2 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. the 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.

10.3.3 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).

10.4.1 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 8 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 $|C'(r,s) - D'(r,s)| > \text{ADT}$ 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 $|(C'(r,s) - C'(r,s+2)) - (D'(r,s) - D'(r,s+2))| > \text{DDT}$ then change one of $E(r,s)$ and $E(r,s+2)$ to -1. (Which of the two is reduced depends on other neighbouring elements within E.)

(c) *Summed distance test*: whenever $E(r,2s)$ and $E(r,2s-1)$ are both positive elements within E, then

If $|(C'(r,2s) + C'(r,2s-1)) - (D'(r,2s) + D'(r,2s-1))| > \text{SDT}$ 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.)

10.4.2 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.

10.5 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 appendix H) — producing performance measures:—

MTE	=	3.48%
P99	=	97.0%
P999	=	91.0%
MR1	=	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 appendix I). 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%

Appendix I 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 para 10.2.1). 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 para 8.3.3(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. 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 topological 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 42.) 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.

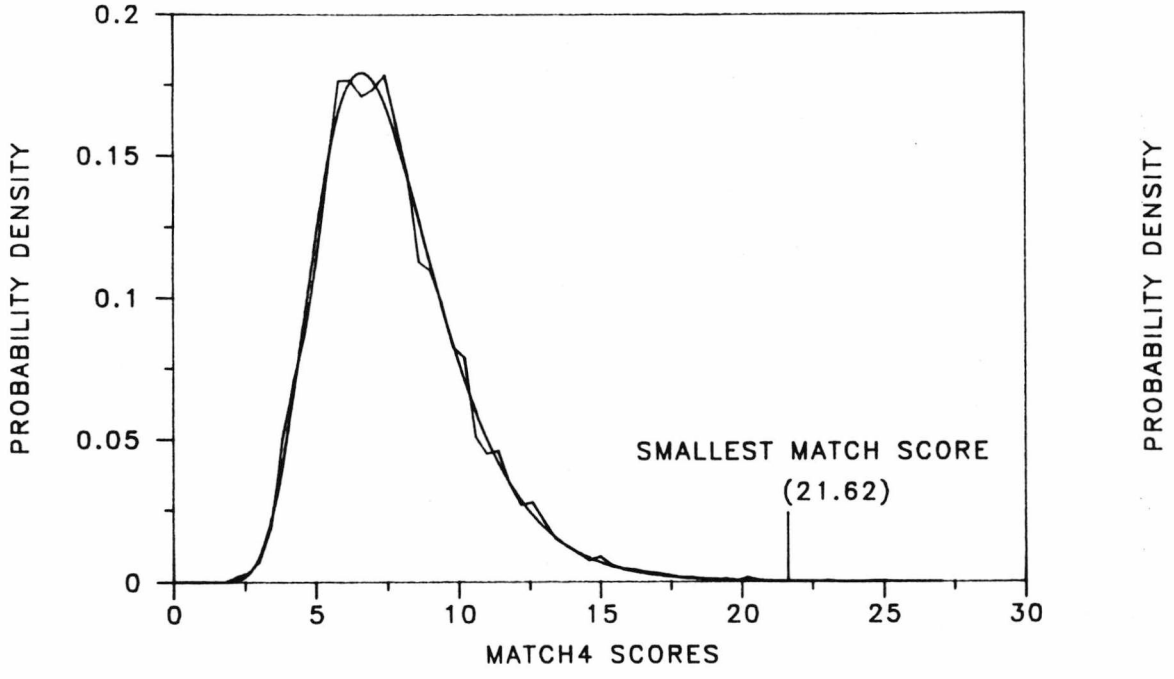


Figure 42. Lognormal fit for MATCH4 with lowest match score shown.

10.6 MATCH4 performance on whorls and arches.

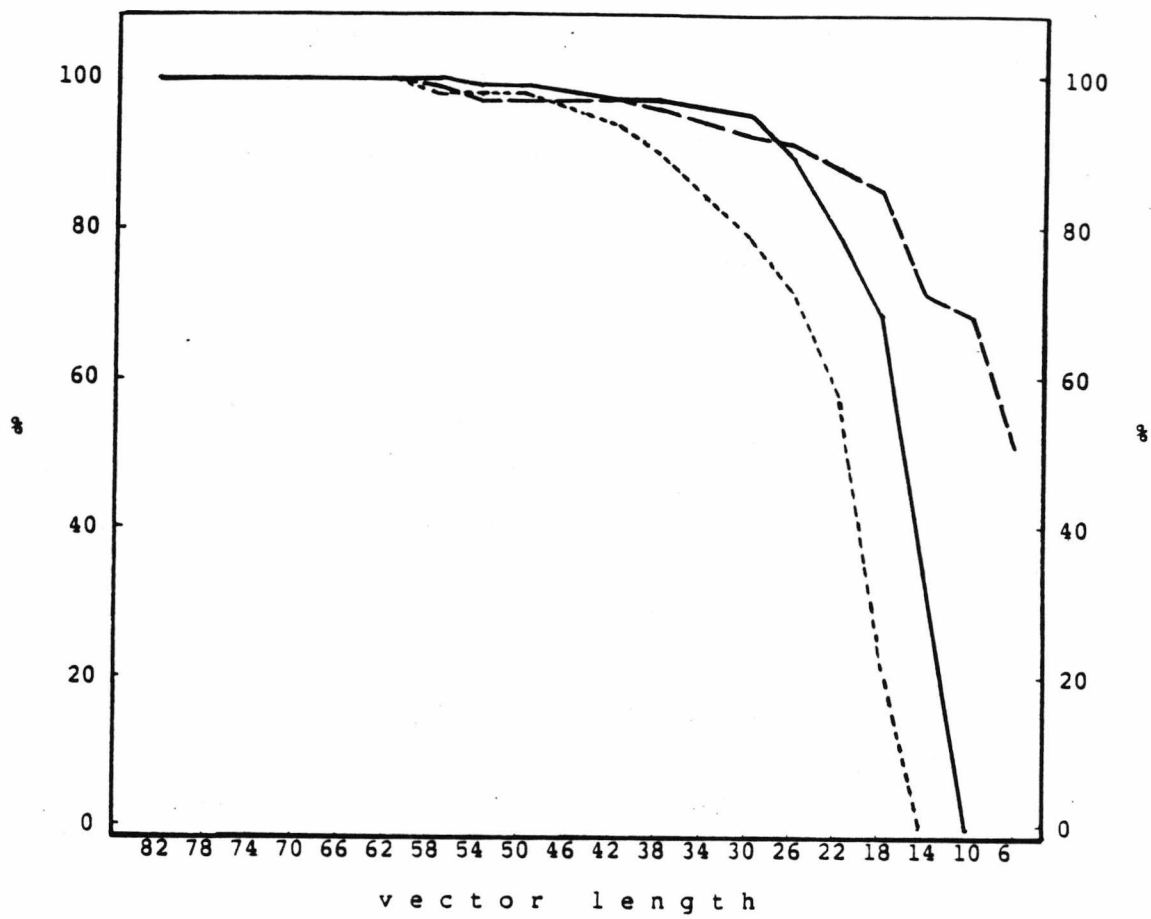
(a) WHORLS: MATCH4 was tested against TESTSET5 (100 pairs of whorls, with distance measures included) and the summary of results is given in appendix J, tests 56 — 58. Once again all 100 mates were ranked 1st, and P99 and P999 values of 100% were obtained in tests 57 and 58. The lowest value for MTE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

10.7 Use of shortened vectors — results on loops.

Tests 17 to 33 (see appendix K) used progressively less and less information from the database TESTSET1: 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.

The standard size of a code array in TESTSET1 was 82×2. 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 43 shows how the performance measures MR1, P99 and P999 vary with array length. Figure 44 shows how the performance measure MTE varies with array length.



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

Figure 43.

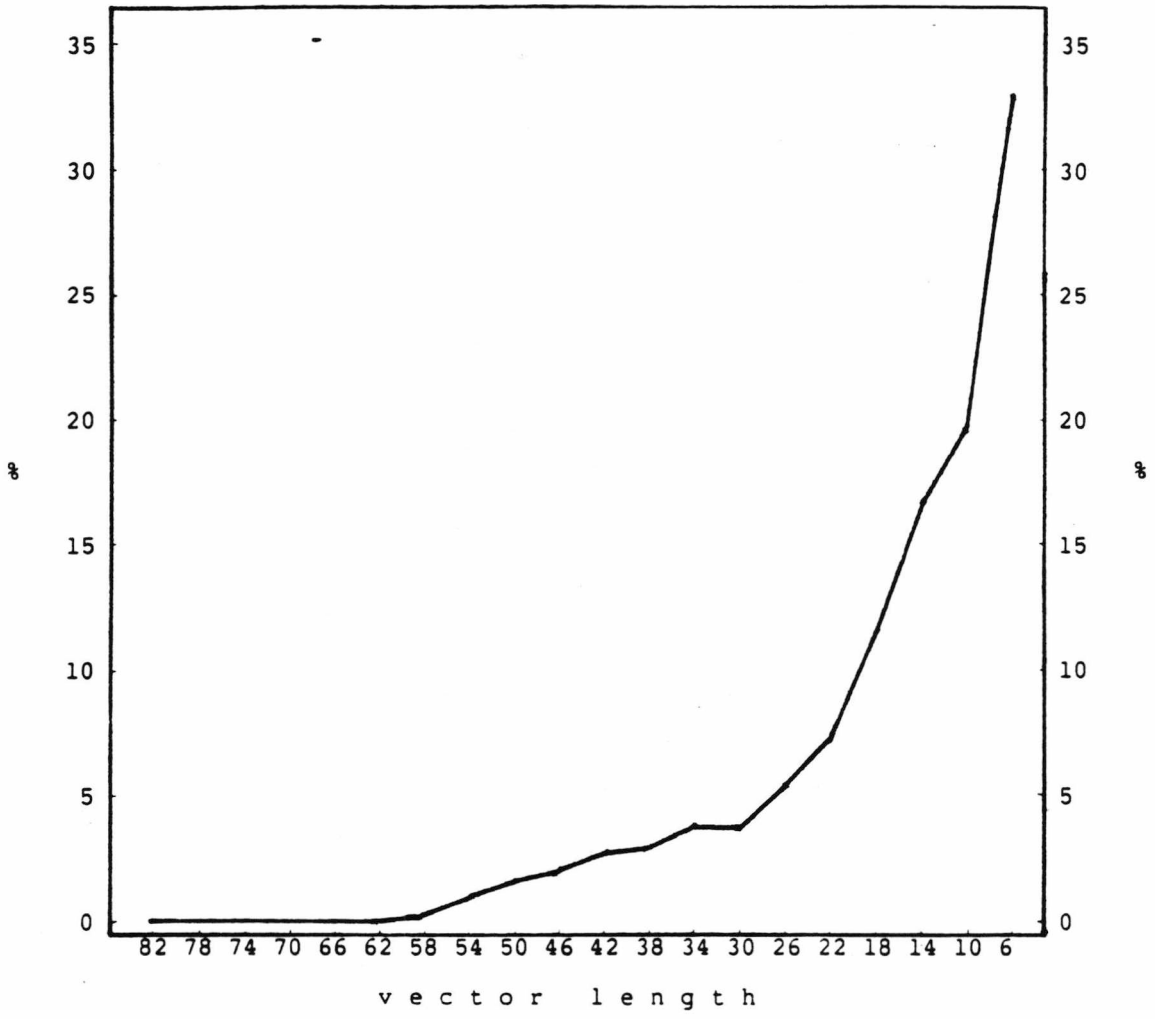


Figure 44. Graph showing how MTE varies with vector length reduction.

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.

CHAPTER 11.

COMPARISON OF TOPOLOGICAL AND SPATIAL APPROACHES.

11.1 Aims and method of the comparison.

A direct comparison of performance between an algorithm using the conventional (spatial) techniques and the topology-based algorithm, MATCH4, 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 the reference.[38] 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, θ, 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.

11.2 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, and no other known *pdf* could be found to fit it adequately.

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 21.23 seconds. M82 – 1 hour and 35 minutes.

These times are for the whole test. i.e. 10,000 comparisons plus some administrative calculations. In the case of MATCH4 only, the time given includes detailed statistical analysis of the scores — which was always done routinely during the test. 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 θ) — and up to 100 characteristics are recorded per print. The maximum storage requirement for the minutiae information is therefore 300 bytes per print. The 82×2 arrays used by MATCH4 each require exactly 82 bytes per print. They can also be shortened to 62 bytes (see para 10.7) with no appreciable drop in reliability.

11.3 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.

PART III

Coding and searching of fragmentary latent prints

CHAPTER 12.

INTRODUCTION TO THE CODING OF LATENT MARKS.

12.1 Introduction.

At the commencement of the work on rolled impressions it was stated (para 7.1) that such work could be regarded as preparatory for tackling the problems of latent marks, and that it could be expected to provide general education as to the behaviour of topological codes under conditions not much worse than 'ideal'. We should consider, therefore, which of the major lessons learnt we can expect to apply to any topological coding scheme for latent searching. In fact there are just two such major lessons worth recalling at this stage :—

Firstly: that the 'placing of lines' is a neat and efficient basis for the ordering of topological information provided, of course, that sufficient global information is available to determine the 'correct' placing.

Secondly: that the greatest power of discrimination between mates and non-mates will be realised by algorithms that use a combination of topological and spatial information.

12.2 Problems of interpretation and system design assumptions.

It would also be prudent to remind ourselves of the special problems posed by latent marks. Some of those problems stem directly from the physical nature of the marks themselves — usually being chemically developed (and subsequently photographed) versions of a perspiration deposit on some object that has been handled. These are :—

(a) that the image will usually lack the clarity of an inked impression.

(b) spatial distortion will be exaggerated and unpredictable as it will be dependent on the shape of the object handled and on the direction and magnitude of pressure exerted upon it.

(c) the surface of the object itself may well give an interference pattern superimposed (rather 'sub-imposed') on the ridge detail and which needs to be filtered out from it.

(d) the fingerprint image may well be smudged if (as is usually the case) there was a degree of lateral movement at the moment the impression was left.

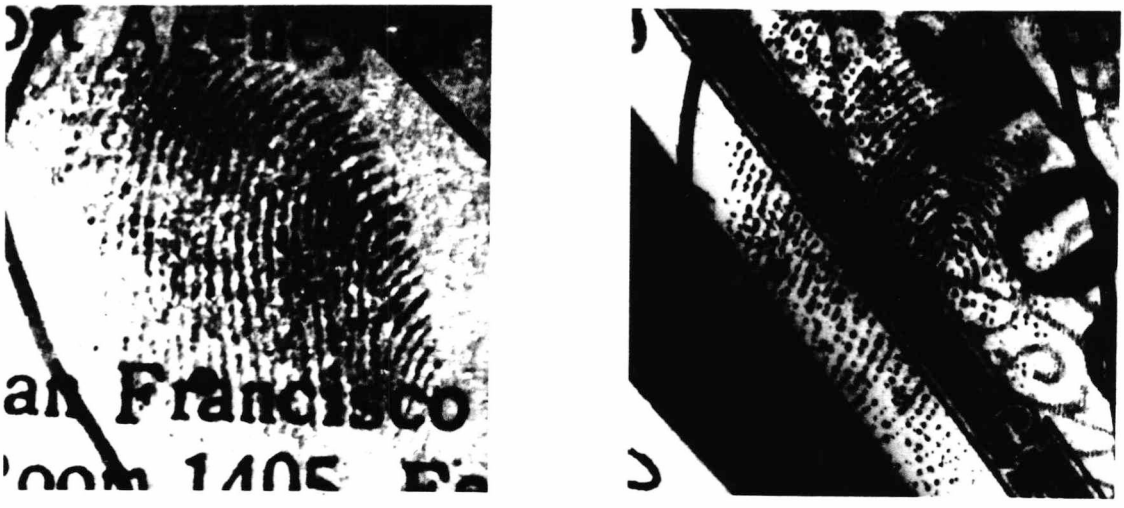


Figure 45. Sample latent marks. (Approx. 5×)

The sample latent marks shown in figure 45 illustrate these problems very well.

These problems sound as if they ought to be the very meat of some image enhancement process. One could expect that two-dimensional Fourier transforms would be used to remove the effects of lateral movement and to utilise the periodicity of the ridge pattern in order to separate it from the background interference.

Perhaps, at some time in the future, image-enhancement techniques may be so much improved as to render them capable of doing a reasonably good job of interpreting latent prints; for the time being, however, they are nowhere near effective enough for this application. Current research methods are, just now, bringing such processes close to the point where we can rely on them to make a fairly accurate interpretation of clearly inked rolled prints from a scanned image, and to automatically extract the positions of characteristics from that interpretation. However the degree of success with which even the most sophisticated systems can handle rolled impressions of poor quality is highly questionable — and nobody seriously expects machines to be able to read *latent* prints effectively without a great deal of human (interactive) assistance. (Some systems provide for technicians to make a tracing of the latent — the tracing then being read by automatic scanners. In this case the interpretative stage is completed in the process of making the tracing.)

Indeed the reading of latent marks requires the very highest level of interpretative filtering that the human brain can provide. The job of reading and searching latents is the most difficult task asked of the fingerprint expert and is, in many organisations, the preserve of only those technicians with the greatest amount of experience and expertise.

It is currently the case, therefore, that when minutiae data representing latent marks are fed into an automated system (for searching against a large file collection) the

data are already the outcome of a human (and usually manual) interpretative process. This 'state-of-affairs' is, in fact, perfectly reasonable. A latent mark is usually found by a painstaking and thorough search of the scene of a crime by highly trained personnel. It is then developed by a variety of means (the use of LASER being the most publicised recent development) but always with great care — for the information content within the mark is both scant and fragile. One could expect a similar degree of care to be exercised in entering that information into an automated system lest any of it be lost. The whole of the information-gathering process is a 'once only' process, as opposed to the comparison against file-prints, which is a repetitive process. There is therefore very little to be gained, and much that could be lost, by automating the latent entry process.

For these reasons the *fact* of manual human encoding of latent marks is an underlying assumption of this project. We should endeavour to ensure that any method devised for coding a latent mark by topological means can be carried out manually by a human technician both easily and quickly, and without requiring any detailed mathematical knowledge. This requirement is met by all the coding schemes described hereafter.

Quite a different assumption pertains to file collections — namely, that automatic file conversion (by scanners linked to processors) is a prerequisite for establishing major computerised systems. The data requirements for topological coding schemes *for the file-prints* are therefore limited to those which demand little or no advance on existing automatic-reading techniques.

12.3 Referencing and incompleteness problems.

Once the latent has been traced, or otherwise interpreted, to the best of the technician's ability some special problems remain which make significant demands on any searching algorithm :—

(a) It may not be possible to determine the 'pattern type' classification of the finger which made the mark.

(b) 'Referencing' or 'registration' of the mark to some standard orientation may not be possible as referencing features (such as cores/deltas/creases) may not be visible.

(c) Ordering of information within a latent mark according to any standardised global scheme will not be possible. Frequently one cannot tell precisely from which part of the finger the latent comes, nor can one always accurately determine its orientation.

It is clear that problems (b) and (c) above render the topological schemes used on rolled impressions wholly inappropriate and that either an *unordered* or a *locally ordered* information system is required as a basis for topological comparisons involving latent marks.

12.4 Early approaches and their drawbacks.

Two possible methods of coding prints for latent searching arise out of the preceding chapters. Neither of them are really satisfactory as self-contained schemes (as will be explained), but they were both important stages in the evolution of the eminently satisfactory solution to be described in chapter 14. It was the bridge built between these two ideas that pointed the way firmly towards development of a 'topological coordinate system'. The two foundation ideas are described in turn.

12.4.1 Local characteristic codes.

The idea was expressed in chapter 5 that a fingerprint could be coded topologically by recording an *unordered* selection of *local* topological codes. Each topological code would be a vector generated by systematic exploration from short straight lines drawn through a characteristic, and orthogonal to the local ridge flow direction. Searching a latent mark against a collection so coded would then be by extraction, from the latent, of a similar vector (or vectors), followed by vector comparison of the kind well established in Part II. Chapter 5 suggested the use of bifurcations alone as bases for local vector extraction — and derived eleven digit codes by allowing the line to span two ridges either side of the selected bifurcation. Such an information gathering process could be represented pictorially as in figure 46.

There are a number of adaptations to this basic idea which would help to bring it into line with the work of Part II, and to make it compatible with those vector comparison algorithms already developed. They are :—

(a) that lines placed should be imagined to be *offset* by an infinitesimally small distance, so they pass *right by* the bifurcation rather than *through* it. The reason for this is that it gives an even number of topological exploration paths (rather than an odd number) yielding an even number of digital codes.

(b) that the order of topological exploration shall be changed to the convention *work outwards from the core, and always look left before you look right.* *

(c) that each topological event code shall have a distance measure associated with it.

Such an updated version of local bifurcation coding would provide digital arrays compatible with the array comparison techniques incorporated into the algorithm

---

* The core itself may not be visible. There are, however, very very few latent marks where the ridge curvature does not give away a very rough location for the core (or, in the case of an arch, an idea of the print orientation). To order these vectors correctly in the absence of a visible core one needs only to be able to determine which is the *inside* of the mark and which is the *outside*. On those few occasions when this is not possible a double-entry facility would be needed to cover both possible interpretations.

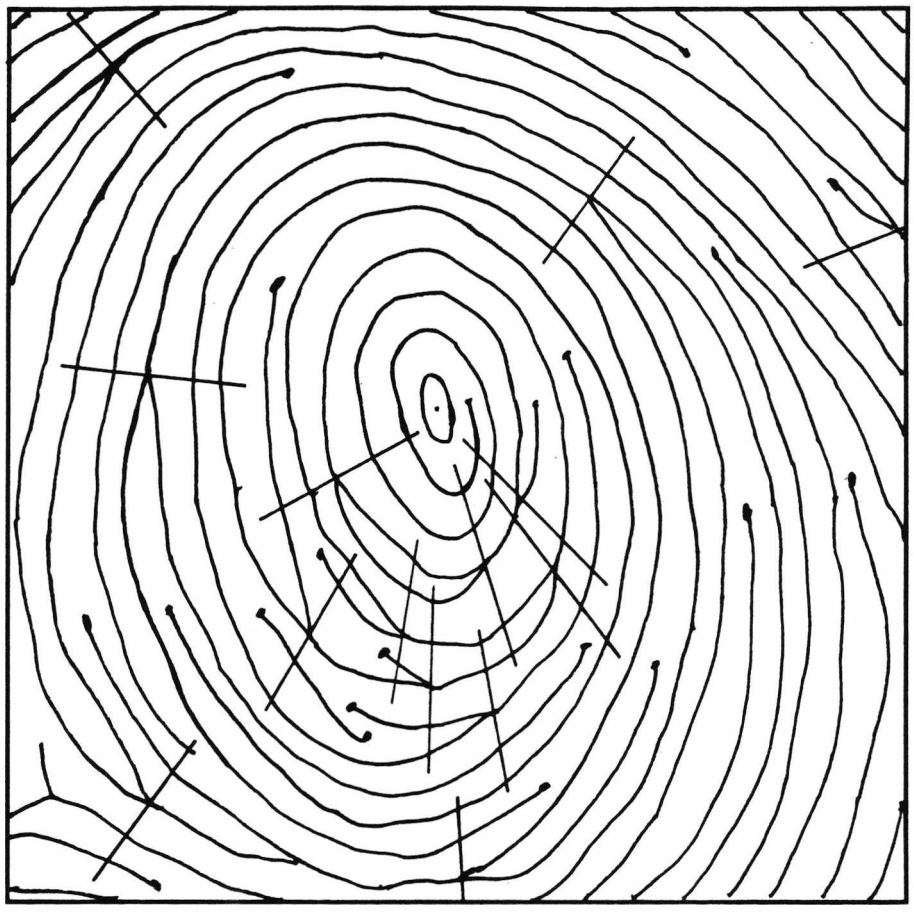


Figure 46. Local bifurcation-based vector coding (schematic).

MATCH4. The new order for the ridge exploration event codes would be as shown in figure 47. Note the slightly offset line, and the fact that the 5th exploration runs immediately (i.e. at *zero* distance) into the central bifurcation where it would give digital code 7 (for 'bifurcation ahead').

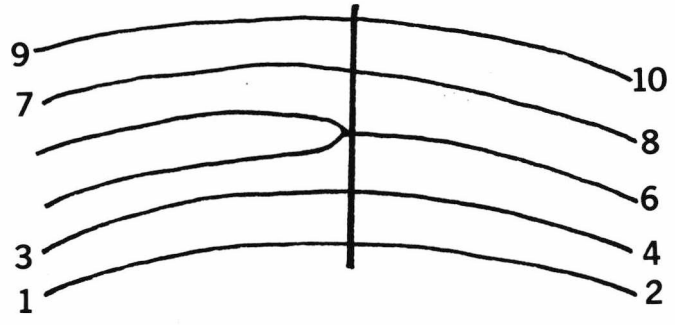


Figure 47. 'Offsetting' of generating lines.

If the bifurcation had faced in the opposite direction then we would choose to offset the line to the right, as before, rather than to the left. We thus add a further convention regarding the placing of characteristic-centred lines namely that *lines based on minutiae should be marginally offset in a clockwise direction (clockwise with respect to the assumed position of the core) for the purpose of ordering topological information, but by a negligible physical distance so as to make the distance from the characteristic to its line effectively zero.*

Furthermore, in the light of previous experience with topological code vectors, the following generalisations ought to be made to this scheme :—

(a) All true characteristics should have their topological neighbourhoods coded rather than bifurcations alone. The inclusion of ridge-endings is essential in view of the increased frequency of bifurcation/ridge-ending mutations observed when dealing with latent marks whose interpretation is so difficult.

(b) Vectors should not be limited in length by the span of the generating line being set at just two ridges: rather the span should be a parameter of any comparison algorithm.**

The principal drawback of this coding scheme is its data storage requirement. Having accepted the desirability of using longer vectors, let us suppose a standard span of 10 ridges was chosen: there are then 20 ridge intersection points (ten each side of the characteristic) yielding 40 topological event codes, and forty associated hexadecimal distance measures. The storage requirement for file collection prints is therefore 40 bytes *per characteristic*, which is quite unreasonable. It is particularly unreasonable when account is taken of the very high degree of redundancy that there would be in such a set of data. The relationship of one characteristic to a near neighbour would be recorded many times over.

Shortening the vectors stored (by reducing the parameter SPAN) would certainly reduce the data storage requirement but would be expected to worsen performance. Facing such trade-offs between data storage requirements and performance is a situation that we can, and should, avoid.

12.4.2 Series of radial lines.

The second fundamental approach to file-print coding for latent searches is a simple extension of the line-based coding system used in Part I for rolled impressions. One single line superimposed on the rolled print was used to generate 82-digit vectors, and the lines

** We already know that discrimination between mates and non-mates improves substantially with vector length up to size 30×2 (i.e. 15 ridge intersection points) as can be seen from figure 43, para 10.7. The assumption that vector comparison algorithms would be implemented on array processors removes any concern that there might be over increases in processing time that could result from the use of longer vectors.

were placed (except in the case of arches) by reference to the central core. Topological information was thereby recorded mainly from those parts of the print close to that line, and not from the entire print.

It is essential, in any latent scheme, that information from *every* part of each fileprint be recorded in order that information from a latent mark will have some representation in the matching file-print data irrespective of which part of the finger made the latent impression.

If a whole series of lines were drawn radially from the core, as shown in figure 48, and vectors derived from each of them, then topological information would be recorded from all over the print. In figure 48 the spacing of the lines has been set at 30°. Given a latent mark one could then draw a line centrally across it at such orientation as was deemed most likely to pass through the core (assuming the core is not visible). Then one topological event code vector can be generated from that line according to established conventions (i.e. working outwards, and looking left before right).

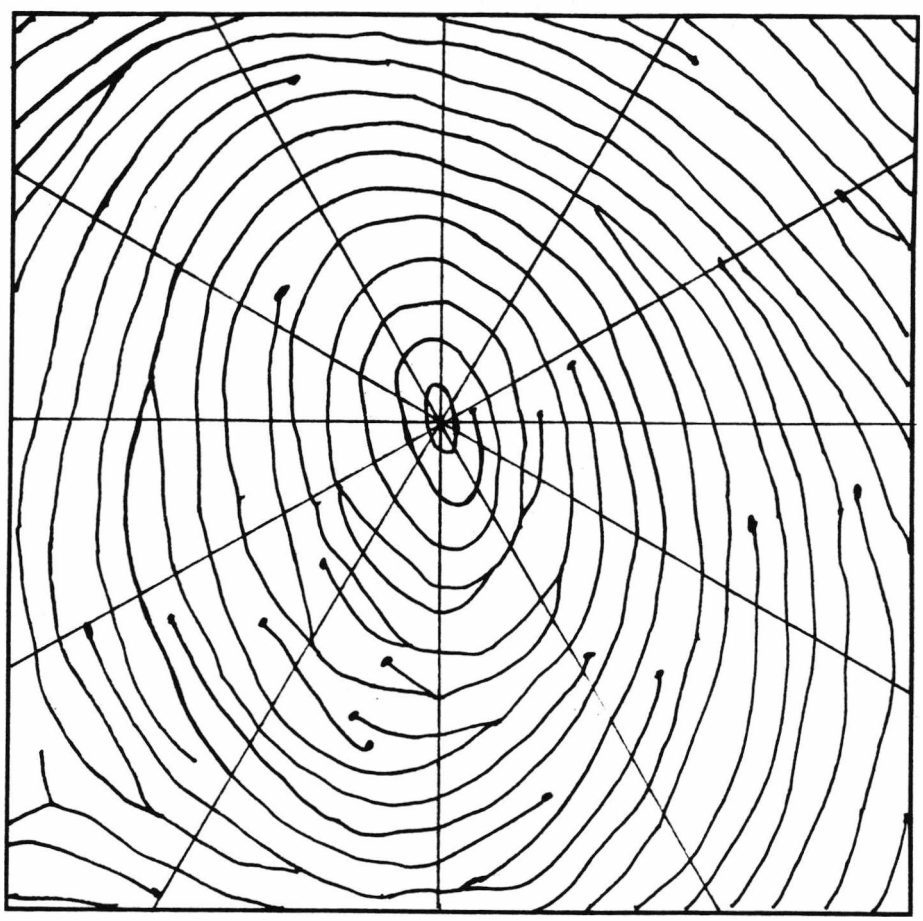


Figure 48. Radial line coding scheme.

Provided the radial lines on the matching file-print were sufficiently close together one could expect some portion of one of those file vectors to be very similar to some portion of the latent vector. The degree of similarity would depend, to a certain extent, on how lucky one was in choosing the position for the line on the latent. If it corresponded within, say, 5° of the position of one of the radial lines on the mate file-print then a very good vector comparison score would result. If the latent line fell half way between the corresponding positions of two of the file-print radial lines, and in an area of high characteristic density, then vector comparison scores would be very poor.

Use of a greater number of radial lines (e.g. with 10° spacing) would raise latent mate scores but would, once again, increase data requirements for file-prints to unacceptable levels. Moreover, use of line-placement, on a latent, that is not tied either to a core or to any visible characteristic effectively rules out the use of distance measures as a means of enhancing the performance of topological vector comparison (except, perhaps, careful use of *summed* and *differential* distance tests. These tests measure distance between two characteristics not directly associated with the line placement - see para 10.3).

12.5 Ultimate objectives for file collection data storage.

The two methods described above appear cumbersome; there is neither speed nor reliability to be obtained through their use. No substantial experiments were conducted on either of them as the data requirements (and therefore the time taken in a manual encoding process) were prohibitive — especially if any attempt was to be made to obtain the maximum reliability. Consideration of their use did, however, help to formulate an objective for the design of a workable topology-based latent scheme, namely that we should find :—

a method for recording a complete topological description of a print (so that the topology of any part of it can be inferred) subject to the constraint that each characteristic be recorded once, and once only.

12.6 Sweeping-line systems.

The key to attaining the objective stated above lay in the realisation that characteristics could be seen as *small changes* in the otherwise laminar flow of the ridge pattern. That realisation leads onto the idea that *the whole topology of a print is merely the summation of a series of small changes in an otherwise smooth ridge flow pattern.*

For the sake of a more practical understanding of this statement suppose that a topological code vector (of the type with which we are now familiar) had been generated by a line placed in some particular position on a print. Now suppose that the line was displaced by a small translation in the direction of the ridge flow so that it now passed the other side of one characteristic (in other words — the line *passed over* one characteristic),

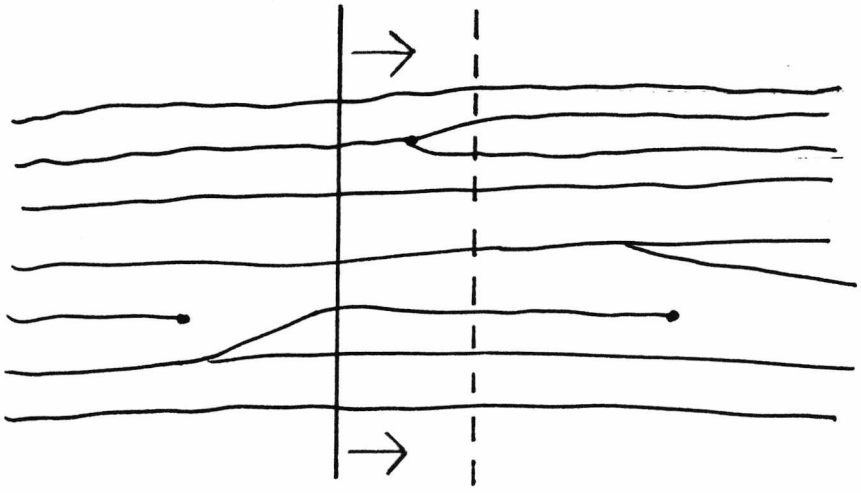


Figure 49. 'Sweeping line' system.

and a new code vector generated to represent the new line position. (See figure 49). How would the two vectors differ? Certainly they would be very similar, and the differences (which would all be local to the characteristic *passed over*) could all be deduced from certain knowledge about that one characteristic. In order to detail those changes you would need to know:—

(a) what type of characteristic was it, and which way was it facing?

(b) which ridge, or ridges, was it on?

(c) what can we now see (looking right along ridges) that we could not see before by virtue of the presence of that characteristic? (i.e. we now have new ridges to explore — two new ones in the case shown in figure 49.)

A set of rules can be built which would detail all the vector changes that are caused by each particular type of characteristic when they are *passed over* by a sweeping line.

In figure 49 the new (displaced) line vector can be seen as the original line vector 'plus' the changes caused by passing over that characteristic. Further displacement of the line (i.e. a continued *sweep*) will add further changes to the vector as other characteristics are reached and *passed over*. This is a very general introduction to the basis of what could be called 'sweeping line systems'.

12.7 Radial scanning.

The 'radial scanning' scheme is one particular case from the broader class of sweeping line systems. It provides a method for recording the *whole topology* of any sector of a fingerprint. It has two principal determining features:—

(1) that a *central observation point* on the fingerprint is selected.

(2) that the sweeping line used is a straight one, and it scans radially as if it were pivoted from the observation point.

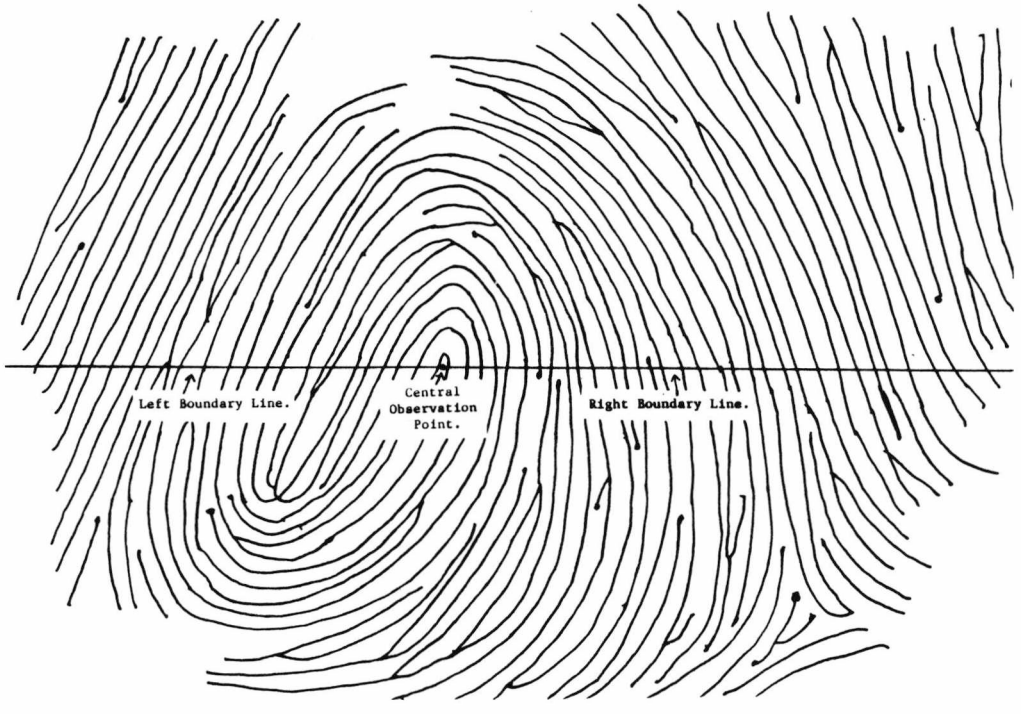


Figure 50. Sample sector for radial scanning.

The similarity of such an idea with the appearance of a radar screen is quite obvious, and may well be a helpful aid to understanding the application. To demonstrate the use of radial scanning let us consider the 180° sector of the fingerprint shown in figure 50. (In effect this means the half of the print above the horizontal line; that part of the print should be regarded, however, as a sector enclosed by two radial lines.) The topology of the whole sector can be described by recording the following information:

(a) 2 boundary vectors: these are the topological code vectors generated from the boundary radial lines.

(b) a complete listing of all of the characteristics, together with any other irregularities in the otherwise laminar flow, that occur within the sector. Each irregularity must be listed in a manner which shows the *nature* of the irregularity, the *order* of their appearance, and *on which ridge* each one occurs.

The form of data contained in the boundary vectors can be assumed (for the purpose of this section) to be our standard format for line-generated vectors with their

associated distance measures. The listing of flow irregularities, however, is quite new — and takes the form of a *coordinate set*. The *coordinates* for each irregularity consist of

(i) a hexadecimal digital code (T) representing the type of the irregularity.

(ii) the angular coordinate (θ) of the irregularity. This is sufficient to specify the order in which they are *passed over* by the sweeping radial line. We will use angular measures that increase clockwise, with 0° being coincident with the left boundary line. Thus θ will range from 0° to 180°, in the case of figure 50.

(iii) the ridge-count (R) between the irregularity and the central observation point. This is sufficient to specify *on which ridge* it occurs.

A most valuable observation can now be made, namely that

a list of coordinate sets of the form (T, θ, R) *specifies the topology of a sector uniquely.*

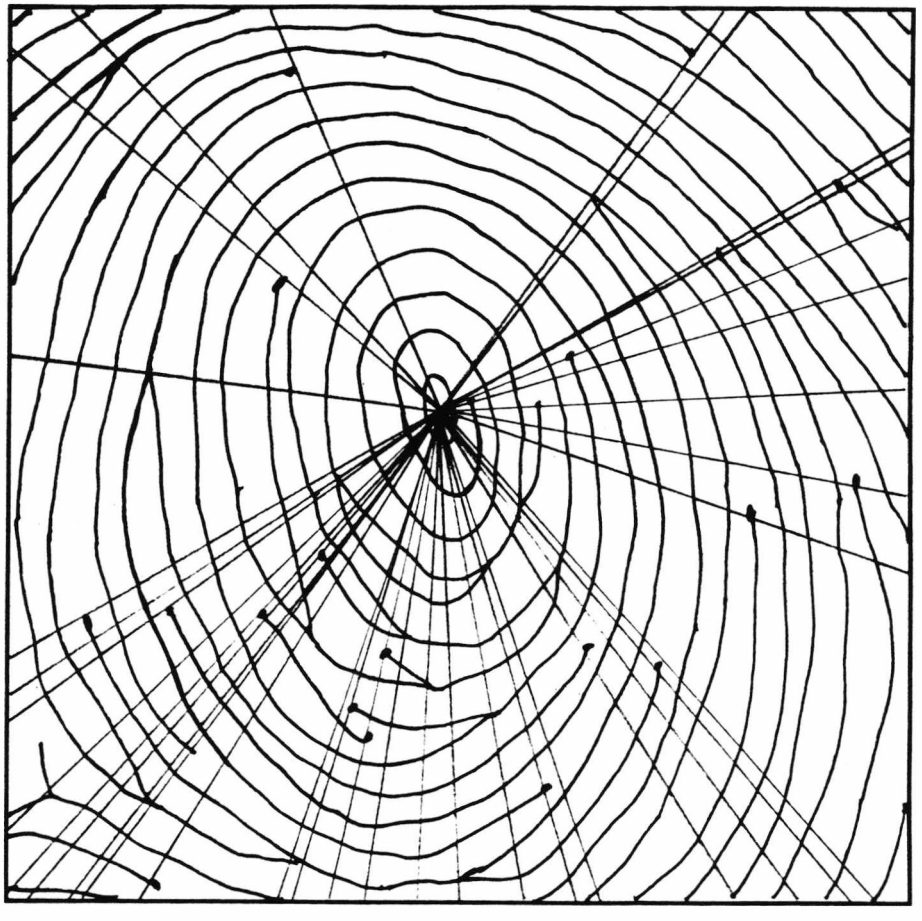


Figure 51. Characteristic-based radial lines.

That statement could be presented as a theorem, requiring proof — but it is hardly necessary. The best proof of the assertion that the whole ridge structure can be reconstructed unambiguously from such a set of data, is to describe the method for doing just that. In chapter 14 appears a detailed explanation of the mechanism for *topological reconstruction* from such a *topological coordinate set.* Such detailed description is not included here as the purpose of this chapter is purely to recount the evolution of ideas which led to development of *topological coordinate systems.*

In order to show just how closely related this coordinate system is to the two foundation ideas described earlier (para 12.4) let us adapt figure 48 slightly. Figure 51 shows the same print with a radial line drawn marginally offset from every visible ridge flow irregularity. The lines span the whole visible ridge structure (rather than being limited to just a few ridges), and their spacing is determined by the angular position of the irregularity (rather than by a fixed, regular interval). A set of coordinates of the form (T, θ, R) can then be seen as the most economical method of recording the sequence of changes in topological code vectors that occur between one radial line and the next. The diagram (figure 51) bears an interesting resemblance both to figure 46, and also to figure 48, and could be taken to be a hybrid of the two.

CHAPTER 13

EARLY LATENT SEARCHING ALGORITHMS.

There are two different ways of describing a chicken. The first is to describe an egg in detail and then to trace all the changes that take place as it develops into a chicken. The second is to describe the fully grown chicken in detail and, perhaps, make a few brief comments about the egg just to put things in context.

In describing latent matching algorithms we shall follow the second of these two paths. Chapter 14 is the detailed account of the fully-fledged solution, and these next few paragraphs are intended merely as an overview of the early stages of development. Consequently the intricacies of these algorithms are not explained here, and there may be nagging questions in the mind of the reader as to some of the finer points of *topological reconstruction.* All those questions will be answered in due course.

13.1 Latent entry by vectors.

All of the algorithms to be mentioned in this chapter have certain basic features in common. They are: —

(a) That the entry of data from the latent mark is by way of characteristic-centred vectors which are manually encoded from a traced image of the latent.

(b) That file-print data is entered and stored in the form outlined in paragraph 12.6 (i.e.by two boundary vectors plus topological coordinates (T, θ, R) for all intervening characteristics and other ridge flow irregularities).

In order to perform a comparison each algorithm first topologically reconstructs the file-print from its coordinate set, and then automatically extracts characteristic-centred topological code vectors from its reconstruction. Vectors centred on all 'suitable looking' characteristics (i.e. characteristics of the right *type* that lie within an area of the print which is specified at the time of latent data entry) are then compared with the latent vector and a score is obtained in each case. The highest score obtained by an extracted file-print vector is taken as the score for that file-print. It is assumed to be the score from the characteristic (on the file-print) whose *topological neighbourhood* most closely resembles that of the characteristic on the latent mark upon which the latent vector's generating line was centred.

The vector comparison itself is practically identical to that used on rolled impressions (i.e. as per the algorithm MATCH4).

13.2 Details of the latent enquiry.

Figure 52 shows the tracing of a latent mark (at 7× magnification) with a generating line placed on it. The placing of the line requires some subjective judgement on the part of the operator. Firstly a characteristic should be chosen which is fairly central on the mark. Secondly a line should then be drawn across the ridge flow, oriented so that it points at the assumed position of the core (or actually *through* the core if it is visible on the mark), and spanning as many ridges as are considered *useful* in gathering information from the latent. The line is to be marginally offset from the central characteristic, as discussed in para 12.4.1. The topological code vector generated by this line is entered as the *latent enquiry vector*, complete with its associated distance measures (which are manually measured by the operator.)

Figure 52. Selected line placement on latent mark.

Also certain information about the selected central characteristic (hereafter referred to as the *central feature*) is entered as part of the latent enquiry. Its *type* code is required, as are angular and ridge count bounds within which it is deemed to lie with

respect to the assumed core position. (These bounds are solely for the purpose of limiting the number of vector comparisons to be performed. If they cannot reasonably be specified then they are 'defaulted' so that the whole file-print sector is searched for suitable matching characteristics.) A complete latent enquiry is shown at appendix L, where the data appears on a form prepared for the purpose.

13.3 Details of the file-print coding.

The sector chosen for early experiments was a 180° sector that covered the upper half of each file-print. (This is the part of the finger that most often appears on latent marks.) Limiting the data recorded to a 180° sector was for convenience alone, due to the time consuming nature of the manual coding operation.

The *observation point* was selected to be adjacent to the core in the case of loops and whorls, and at the base of the upcurve (the point at which a 'summit line' can begin to be seen) on arches. Figure 50 shows a typical position for the *observation point* and *boundary lines* on a print with a central core, and Figure 53 shows a suitable placing for these when used on a plain arch. Notice that the observation point is always placed in a valley rather than on a ridge: this is so as to give unambiguous ridge counts in every direction.

All of the irregularities in the sector *between* (in this case *above*) the boundary lines are then recorded by sets of topological coordinates of the form (T, θ, R). The *type* of irregularity is shown by a single hexadecimal digit — and the allocation of digits is closely related to the allocation already in use for ridge-exploration events. The list of possible irregularities, with their hexadecimal codes is given here. The descriptions can best be understood clearly if you think of these irregularities as being *passed over* by a pivoted radial line which is sweeping in a clockwise direction.

Code 0 — ridge runs out of sight.
Code 1 — ridge comes into sight.
Code 2 — bifurcation facing anticlockwise.
Code 3 — ridge ending.
Code 4 — ridge recurves with the effect of *losing* two ridges.
Code 5 — ridge recurves with the effect of *gaining* two ridges.
Code 6 — facing ridge ending (i.e. facing in the opposite direction to a '3'.)
Code 7 — bifurcation ahead (i.e. a '2' reversed).
Code A — ridge runs into scarred tissue.
Code B — ridge runs into an unclear area.
Code C — compound characteristic (2 ridges in, and 2 ridges out).
Code D — ridge emerges from scarred tissue ('A' reversed).
Code E — ridge emerges from unclear area. ('B' reversed).

Figure 54 shows a completely artificial fingerprint pattern which just happens to have one of each type of irregularity shown on it, spaced at 25° intervals. Radial lines are

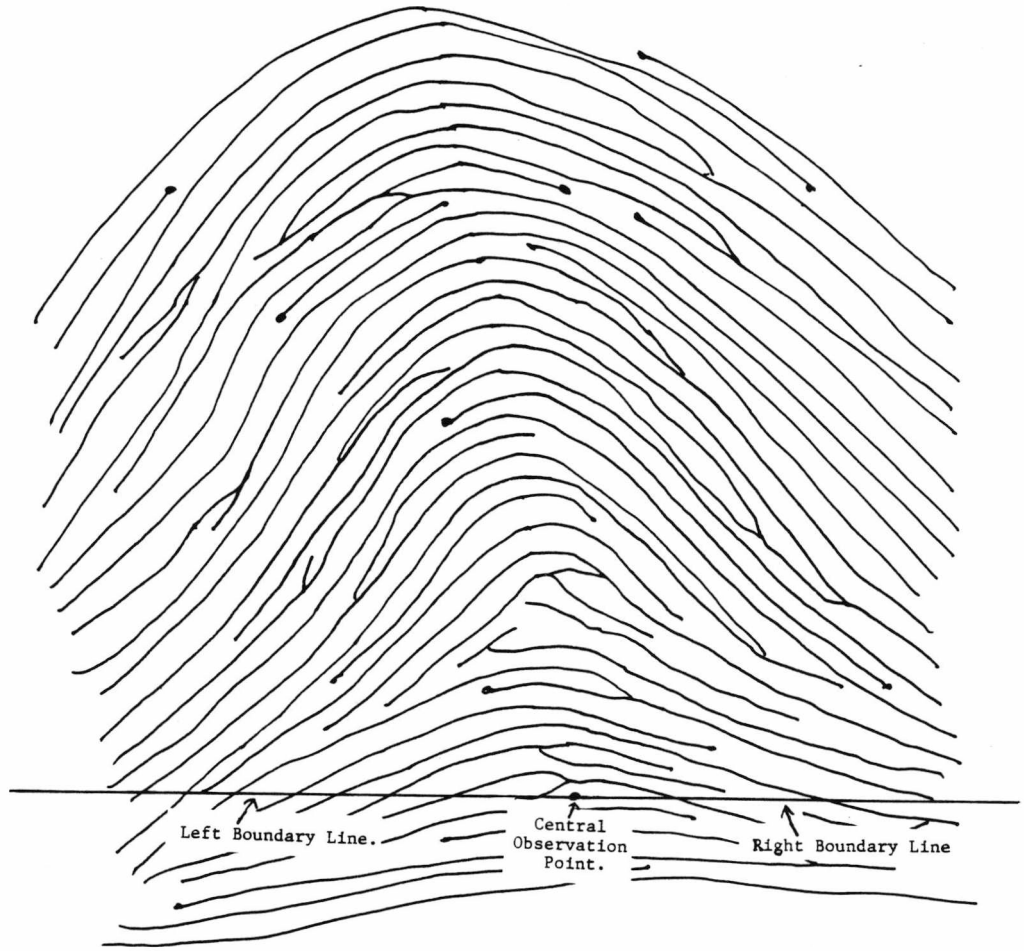


Figure 53. Boundary lines and observation point on a plain arch.

used to identify each of the irregularities with its hexadecimal code. It gives an adequate illustration of each different type.

On the print shown in figure 50 there were a total of 77 such irregularities between the boundary lines. The complete data representation of that file-print is shown in Appendix M — there you will notice the inclusion of some numbers referred to as *distance conversion measures.* These give an approximate ridge spacing wavelength at four sample orientations $(0°, 60°, 120°, 180°)$ which enable the comparison algorithms to convert angular information into an estimate of *ridge-traced distances* for the purposes of vector comparison. You may also observe, in Appendix M, that the boundary vectors are one-sided (as opposed to the more normal double-sided form). This is because it is only necessary to provide the reconstruction algorithm with the parts of the boundary vectors that represent information from *outside* the coordinate sector. The algorithms are quite capable of working out for themselves what happens when ridges are traced *into* the sector — as this can be deduced from the coordinate information.

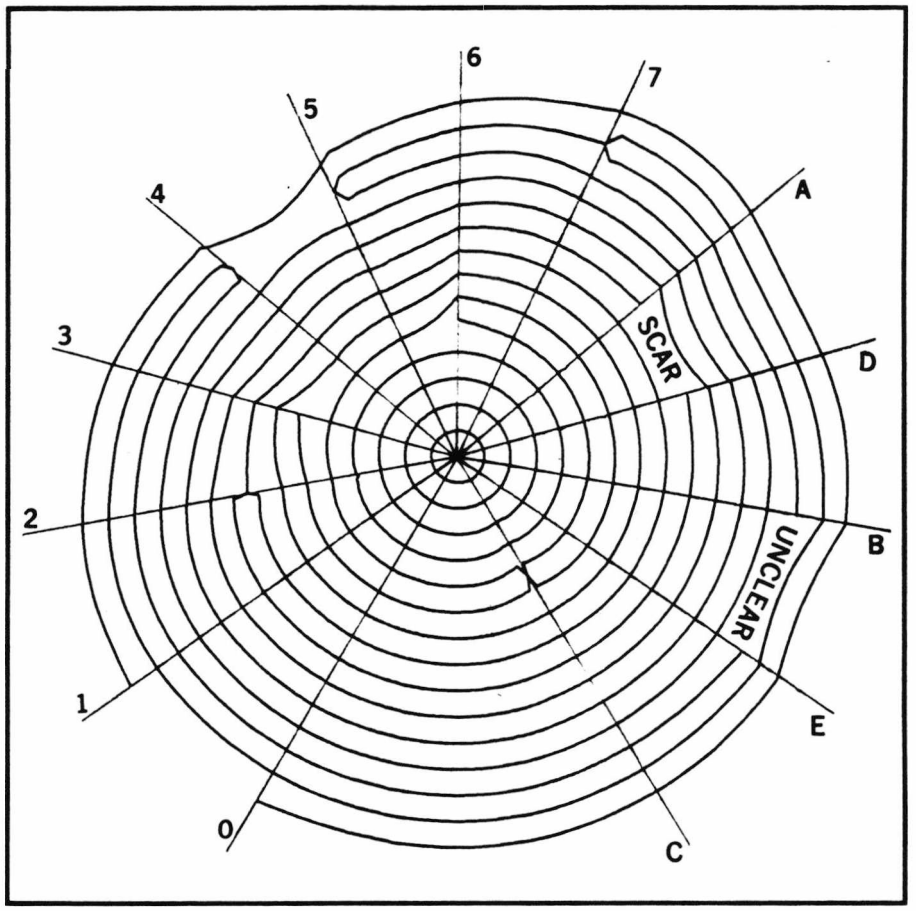


Figure 54. Irregularity types, and their codes.

13.4 The algorithm "LATENT-MATCHER 1" (or "LM1").

The first algorithm tested was an interactive one, in the sense that one vector enquiry was entered at a time and immediately searched against a prepared file collection database. It enabled experiments to be done quickly and easily to find suitable values for the many programme parameters and to give an idea of what sort of *latent enquiry vectors* worked, and which ones did not.

Several valuable lessons were learnt from its use :—

(a) It rapidly became clear that entry of a *single* latent enquiry vector was a most unsatisfactory way of doing latent enquiries. Frequently the central feature upon which the vector was centred was spurious (i.e. it did not exist on the file-print, and had appeared on the latent tracing as a product of misinterpretation of the latent mark) and so no characteristic-centred vector even remotely similar could be extracted from the mate file-print data. It was found to be much more reliable to enter two or three latent vectors per latent mark, each centred on a different characteristic, and to combine their individual scores in formulating an overall

score for the latent mark's comparison with each file-print.

(b) Inferred distance measures (see para 13.3) were unreliable, and demanded that distance tolerances in the vector comparison stages be set much wider than was desirable. Their use helped very little in aiding discrimination between mates and non-mates.

(c) A 180° span for the file-prints (i.e. coding the upper half only) was inadequate. There were several cases where the information available from the latent fell largely *outside* that sector, and the latent could not be identified by the fragment of information that lay *within* the sector. (Nevertheless, in the vast majority of latent marks all, or most, of the useful information lay within the sector, and usually towards the tip of the finger.)

13.5 Improved latent-matching algorithms.("LM2", "LM3" and "LM4")

In the light of these difficulties the following alterations were made to the algorithm LM1.

(a) To cover those cases where the latent mark was comparatively low on the finger, it was made permissible to enter an *approximated boundary vector*, rather than a *characteristic-centred vector*, as a latent enquiry vector. An *approximated boundary vector* was generated from a line placed at what appeared to be a horizontal orientation on the latent, and which did not need to be centred on any visible characteristic. The comparison algorithm would then recognise this vector as such, and compare it to the file-print boundary vectors rather than comparing it with any extracted characteristic-centred vectors.

(b) Facility was built into algorithms LM2, LM3 and LM4 for several latent enquiry vectors to be entered per latent mark. Each vector would then be first treated in isolation, and the best matching vector score from the file-print obtained. LM2 then simply added up the individual scores to give a combined score for the latent mark. LM3 and LM4 added the slight sophistication of combining the individual latent vector scores if, and only if, their relative angular orientation was matched (within specified angular tolerance) by the relative angular orientation of the file-print characteristics upon which the high scoring extracted vectors were centred. That procedure tended to prevent the combination of 'fluke' scores from non-mates.

(c) Distance tolerances were treated *linearly* (i.e. greater tolerance was allowed for greater distances) rather than *absolutely.*

(d) LM4 allowed a different set of distance tolerances to be used in vector comparisons involving *boundary vectors* than those used in comparing *characteristic-centred vectors.* The boundary vectors always required greater distance tolerances due to the uncertainty in the positioning of their generating lines.

Each of these modifications appeared to improve performance somewhat — and it was time to get some idea of the overall discriminatory power of the algorithm.

13.6 Testing algorithm performance.

A collection of 56 latent marks (of varying quality) was provided by the FBI. All of these were interpreted and traced using the 'Graphic Pen' (see para 7.7). Latent enquiry vectors were extracted from each tracing using a degree of subjective judgement as to selection of *central features*, and the latent enquiries formed together into a single database. The mate file-prints (rolled impressions taken from standard FBI ten-print cards) of the 56 marks, together with 44 other randomly selected prints were all traced and coded according to the scheme already described (para 13.3) to give a database of 100 file-prints.

Batch tests were then run, in which each latent search enquiry was compared with each of the 100 file-prints, and a score obtained in every case. For each latent enquiry the file-prints were then ranked according to score, and the position of the mate in the list was noted (the *mate rank*). Performance was then measured by the percentage of mates that were ranked in first place (which is the performance measure 'MR1' described in para 8.3.1). Attention was also paid to the number of mates that were ranked in the top three places ('MR3') and in the top ten places ('MR10').

As performance for latent marks is clearly very much worse than it was for clear rolled impressions, it is unnecessary to use the kind of sophisticated performance measures developed in Part II. The indicators MR1, MR3 and MR10 provide an adequate picture of comparative performance — and will continue to do so until such time as MR1 exceeds 90% .

In order to get some feeling for what levels of performance are desirable, the same set of latent marks and the same set of 100 file-prints were encoded in the traditional coordinate form for use with spatial matching algorithms. Once again the Graphic pen was used, and the data entered from the same interpretative tracings as were used for extraction of the topological information. Thus the performance of spatial matching algorithms could be measured on *precisely* the same dataset. * The best performance by the M82 matcher (see para 11.2) gave the following rankings :—

MR1 — 26.8%
MR3 — 37.5%
MR10 — 48.2%

* Latent marks vary so greatly in quality that it is not possible to quote meaningful performance statistics without reference to a *specific* set of latent marks. In this case, not only is the same set of prints used, but the same *interpretation* of those prints was used for the testing of both the topological and spatial matching algorithms.

A series of tests was conducted, both with LM3 and with LM4, to try to tune the various algorithm parameters. Complete tables of the test results are given in appendices N and O. The best results achieved (by LM4 in test number 39) gave the rankings :—

MR1 — 58.93%
MR3 — 67.86%
MR10 — 83.93%

This clearly represents a fairly substantial improvement on the level of performance given by the spatial approach. Special significance can be given to the raising of MR1 from 26.8% to 58.93% as it is the mates ranked in first place that tend to have scores way clear of the field and they are the only ones which would be likely to be correctly identified irrespective of the size of the file collection. Those mates that do not come in top place in a collection of size one hundred are most unlikely to come even in the top *fifty* places if the file collection were of size one million.

13.7 Latent enquiry by vector: shortcomings.

Despite its fairly impressive performance there remained something inherently objectionable about the method of latent enquiry by manual extraction of vectors. The process of selecting central features on which to base the enquiry vectors was too *subjective*: success or failure of any particular vector enquiry depended very heavily on the reliability of its central feature — and vectors based on spurious latent characteristics (those that arose from false interpretation of the mark) invariably scored abysmally against the mate file-print.

An analysis of the 23 latents (out of 56) that had their mates ranked in a position other than first (in test no.39 on LM4) revealed the following :—

(a) in three cases — the central feature selected was spurious.

(b) in two cases — the central feature was in an unclear portion of the file print and so *apparently* did not exist.

(c) in two cases — an unclear area of the file print lay close to the central feature chosen, thus reducing vector comparison scores dramatically.

(d) in three cases — the central feature selected on the latent corresponded to a feature below the boundary lines on the mate file-print, and thus could not be correctly matched.

In at least 10 cases out of 23, therefore, the failure was directly attributable to *unlucky* (or *unwise*) central feature selections. In all of these ten cases there were other characteristics visible on the latent which would have served much better as centres for topological coding.

The sensible deduction from such observations is that it is unwise to base a complete latent search on a small number of extracted vectors. Presumably the greater the number of vectors entered, the greater the chances are of limiting the effects of unlucky central feature selection. The *ideal* policy might well be to enter *every possible* characteristic-centred vector that can be obtained from the mark; that means one vector per visible characteristic. The obvious difficulty with that proposal is the resulting complexity and tedium of the manual data extraction process.

The next step forward now becomes very clear: we must enter latent enquiry data in the highly economical *topological coordinate* form, and allow the comparison algorithm to do all the work involved in extracting the required vectors. The treatment of the latent mark data will then be virtually identical to the treatment already being given to the file-print data. Topological reconstruction of both prints (latent and file-print) becomes the essential preliminary for comparison based on characteristic-centred vectors.

CHAPTER 14.

LATENT SEARCHING: TOPOLOGICAL COORDINATE SYSTEMS.

The problems caused by unfortunate choice of central feature have shown the need for latent enquiry data to be less *selective* and less *subjective*. The most desirable latent data form is therefore a *complete* and *objective* description of the latent tracing. The tracing process itself still is, and always will be, substantially subjective — but it ought to be the last stage requiring subjective judgement. A set of topological coordinates of the form (T, θ, R), (showing type, angular orientation and ridge-count) provides a *complete topological* description, and it therefore becomes the basis for latent data entry. The latent mark data can then be presented in much the same form as the file print data.

The manual latent data preparation process is fairly simple: first the mark is traced (enlarged to $10\times$ magnification). Then the position of the central observation point is *guessed* by the fingerprint expert, and its position marked on the tracing. The guessed core point position may be some way away from the 'visible' part of the latent. Then the correct orientation of the mark is estimated by the expert, and the coordinates of the characteristics, and other irregularities can then be written down.*

There are a number of very major changes in the use of topological coordinates that have to be made in order to enhance their versatility and usefulness. These changes are described in the following three sections.

14.1.1 The 4th coordinate.

Bearing in mind the unreliable nature of inferred distance measures (see para 13.4.b), and bearing in mind also that the topological coordinate scheme already records angular orientation of each characteristic, it would seem to be a very sound investment to include a 4th coordinate — namely a radial distance (D) measured from the central

* An extremely useful tool, for this operation, is a large board with a pin hole at its centre. Around the circumference of a 7 or 8 inch circle the angular divisions are marked (i.e. much like an oversized 360° protractor). A transparent ruler is then pivoted at the pinhole in the centre. When the tracing has been made it is placed over the board, the pivot pin pressed through the guessed central observation point. The tracing falls entirely *inside* the protractor markings, and the ruler is long enough to reach those markings. Radial movement of the transparent ruler (which has one central line on it) over the tracing makes it very easy both to count the ridge-counts for each irregularity, to measure radial distances (these are marked on the ruler in the appropriate units), and to read off the angular orientations from the circumference of the inscribed circle.

observation point. The combination of angular position and radial distance (θ, D) for each characteristic gives a complete *spatial* description of the positions of the characteristics in space. A set of coordinates of the form (T, θ, R, D) therefore gives a *complete topological and spatial* description of a print. It records everything that a comparison algorithm might need to know about the positions of the characteristics and their topological relationships to each other. The data storage requirement for such a description is a mere 4 bytes per irregularity.

We shall record radial distances in units of 0.5mm (or 0.5cm on the 10× enlargement) and round to the nearest integer. No greater accuracy is either required or useful. These distances then appear as integers in the range 0 to 50.**

14.1.2 Dispensing with boundary vectors.

Whatever the sector chosen for description by coordinates the *boundary vectors* can be made *null* by pretending that all the ridges inside the sector go 'out of sight' just before they reach the boundary lines. Thus the boundary lines cross *no ridges* and are therefore empty. The imaginary appearance of each ridge just inside the sector can then be recorded by coordinates. The resulting data is now pleasantly uniform and easier to handle. Boundary vectors, in the earlier algorithms, had been something of a nuisance.

14.1.3 'Wrap around' 360° sector.

The sector to be recorded can be enlarged at will by moving the radial boundary lines, until such time as the internal angle reaches 360°. At that stage the two boundary lines coincide and where they coincide will be called the *cut*. Provided our topological reconstruction algorithm can cope with the fact that, at the *cut*, some ridges effectively leave one end of the sector and reappear at the opposite end, then we can forget about boundary lines and boundary vectors altogether.

The reconstruction algorithm will need to be told *how many* ridges need to be connected up in this way — and that number (which is the number of ridges that cross the *cut*) will be recorded as a part of the fingerprint data. It is convenient to specify that the *cut* will be vertically below the central observation point, and that the ridges which cross it be called *moles* (as they pass *underneath* the observation point).

The coordinate system can now be used to describe the complete topology of a *whole fingerprint.*

** The type code (T) is a hexadecimal integer, the angular orientation (θ) an integer in the range 0 – 360, and the ridge count (R) an integer in the range 0 to 50. The total storage space required for all four coordinates is, in fact, closer to 3 bytes; to be precise, it is 25 bits.

14.2 Topological reconstruction from coordinate sets.

It is time to reveal the mysteries of *topological reconstruction* from a set of coordinates of the form (T, θ, R, D). The method to be described here is certainly not the only way it could be done — but this one does work very well, is probably as fast as any could be, and leads *directly* to the point at which no further work is required to be done in order to extract characteristic-centred vectors from the reconstruction. In fact all the characteristic-centred code vectors can be simply lifted out of the array formed by this method.

It will be noticed that the fourth coordinate (D) is ignored throughout this section as it plays no part in the reconstruction process. It is used in the comparison algorithms only after the topology has been restored.

Let us suppose that the print to be reconstructed has m *moles* and n topological irregularities, whose coordinates are the set $\{(T_i, \theta_i, R_i, D_i) : i = 1, \ldots n\}$.

14.2.1 The 'continuity' array.

This reconstruction method involves the systematic development of a large 3-dimensional array, which will be called the 'continuity' array (C) comprising elements $c(i, j, k)$. To understand the function of this array it is necessary, first, to examine figure 55: it shows a (simplified) fingerprint pattern with selected central observation point and the radial *cut* vertically downwards. A radial line from the central observation point is drawn marginally to the clockwise side of every topological irregularity in the picture (whether it be a true characteristic or not). If there are n irregularities (which we will call $\{I_1, \ldots I_n\}$, then there are $n + 1$ radial lines in total (this includes the *cut*). Calling the *cut* line l_0, and numbering the lines consecutively in a clockwise direction gives the set of lines $\{l_0, l_1, \ldots l_n\}$.

Now re-order the topological coordinate set by reference to the second coordinate (θ) — so that the coordinate set satisfies the condition :—

$$\theta_i \leq \theta_{i+1} \text{ for all } i \in \{1, 2, \ldots n-1\}$$

There are then simple 1–1 mappings between the lines $\{l_1, \ldots l_n\}$, the irregularities $\{I_1, \ldots I_n\}$ and their coordinates $\{(T_i, \theta_i, R_i, D_i) : i = 1 \ldots n\}$.

Each of the lines $\{l_0, \ldots l_n\}$ intersect a certain number of ridges, giving an ordered sequence of *ridge intersection points*. Let the number of ridges crossed by line l_i be called r_i. Further, let the ridge intersection points on the line l_i be called points $\{p(i, j) : j = 1, \ldots r_i\}$ — point $p(i, 1)$ being the closest to the central observation point and $p(i, r_i)$ being the closest to the edge of the visible print.

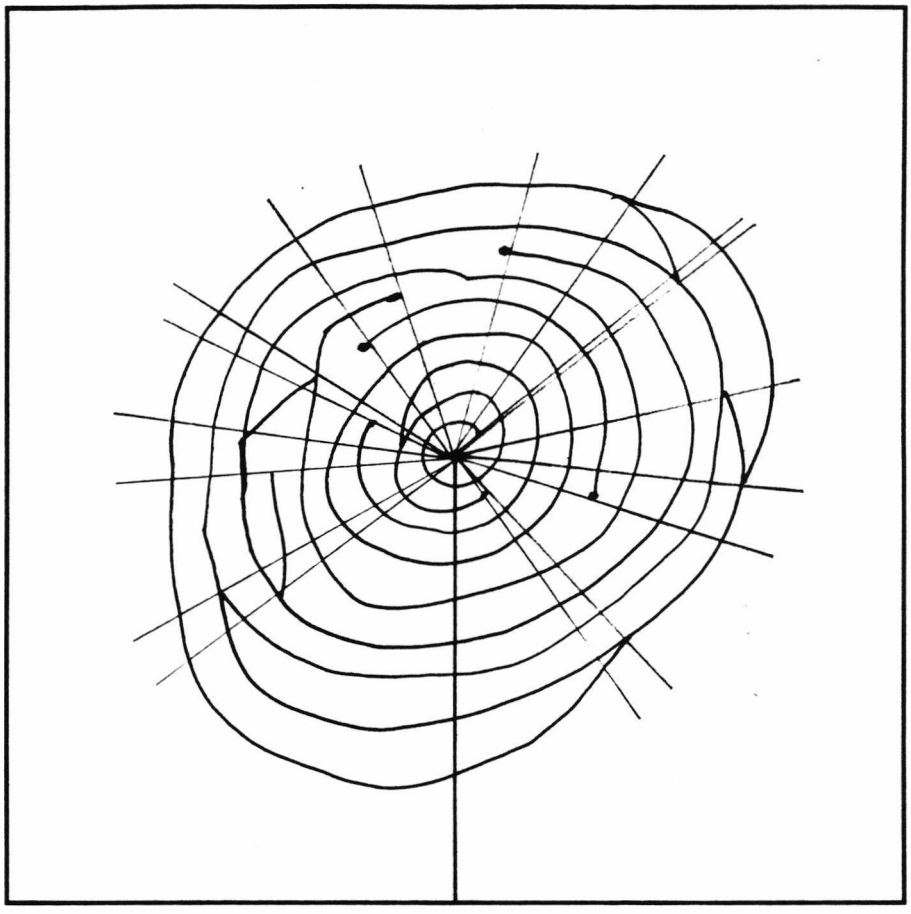


Figure 55. Radial irregularity-centred lines, with the 'cut' vertically below observation point.

The continuity array C is then set up with a direct correspondence between the ridge intersection points $p(i,j)$ and the elements of C, namely $c(i,j,k)$. k takes the values 1 to 4, and thus there is a 4 to 1 mapping of the elements

$$\{c(i,j,k) : i = 0,\ldots n : j = 1,\ldots r_i : k = 1,2,3,4\}$$

onto the set of ridge intersection points

$$\{p(i,j) : i = 0,\ldots n : j = 1,\ldots r_i\}$$

The array C can therefore be used to record four separate pieces of information about each of the ridge intersection points.** The meanings assigned to each element of C are as follows :—

$c(i,j,1)$ — "what is the first *event* that topological exploration from the point $p(i,j)$ in an *anticlockwise* direction will discover?"

** The part of the matrix C which will be used for any one print is therefore irregular in its 2nd (j) dimension.

$c(i,j,2)$ — "*which* of the irregularities $I_1, \ldots I_n$ is it that such anticlockwise exploration will discover first?"

$c(i,j,3)$ — "what is the first *event* that topological exploration from the point $p(i,j)$ in a *clockwise* direction will discover?"

$c(i,j,4)$ — "*which* of the irregularities $I_1, \ldots I_n$ is it that such clockwise exploration will discover first?"

$c(i,j,1)$ and $c(i,j,3)$ should, therefore, be ridge-tracing event codes in the normal hexadecimal integer format (not to be confused with the different set of hexadecimal codes currently being used for the irregularity type (T_i)).

$c(i,j,2)$ and $c(i,j,4)$ are integers in the range 1—n which serve as *pointers* to one of the coordinate sets. They are a kind of substitute for distance measures (being associated with $c(i,j,1)$ and $c(i,j,3)$ respectively) but they act by referring to the coordinates of the irregularity found, rather than by giving an actual distance. They will be called *irregularity indicators* in the following few sections.

14.2.2 Opening the continuity array.

To begin with, the whole of the continuity array is empty (and, in practice, all the elements are set to –1). It will be filled out successively starting from the left hand edge $(i = 0)$ and working across to the right hand edge $(i = n)$.

Starting with $i = 0$ (at the *cut*) we know only that $r_0 = m$ (the number of ridges crossing the *cut* is the number of *moles* recorded in the data.) Nothing is known (yet) about any of these ridges. The first set of entries in the continuity array is made by assigning a *dummy number* to every possible ridge exploration from the line l_0.

The *dummy numbers* are integers in a range which cannot be confused with real *event-codes.** Each dummy number assigned is different, and the reconstruction algorithm views them thus :

"I do not yet know what happens along this ridge — I will find out later — meanwhile I need to be able to follow the path of this ridge segment, even before I find out where it ends."

This first step in filling in the continuity matrix is therefore to assign *dummy numbers* to each of the elements $\{c(0,j,k) : j = 1, \ldots r_0 : k = 1 \text{ or } 3\}$.

The elements $\{c(0,j,k) : j = 1, \ldots r_0 : k = 2 \text{ or } 4\}$ are left untouched for now.

---

* In practice dummy numbers start at 100 and, whenever another one is needed, the next free integer above 100 is used. Obviously a record is kept of how many different dummy numbers have been assigned.

14.2.3 Associations, entries, and discoveries in the continuity array.

The next stage is to consider each of the coordinate sets $(T_i, \theta_i, R_i, D_i)$ in turn starting with $i = 1$. We know that the irregularity I_1 is the *only* change in the laminar flow between lines l_0 and l_1. We also know its *type* (T_1) and its *ridge-count* (R_1). Depending on the type T_1 there are various *associations, entries* and *discoveries* that can be made in the continuity array.

Suppose, for example, that $T_1 = 3$ (i.e. a ridge ends — according to the table of irregularity types, para 13.3). We can deduce that

$$r_1 = r_0 - 1$$

(i.e. line l_1 crosses one less ridge than line l_0), and we can make the following *associations* in the second column $(i = 1)$ of the continuity array. (*Associations* occur when one element of the array is set equal to another.)

$$\begin{aligned} c(1,j,1) &= c(0,j,1) \quad \text{for all } 1 \leq j \leq R_1 - 1, \\ c(1,j,3) &= c(0,j,3) \quad \text{for all } 1 \leq j \leq R_1. \end{aligned}$$

(i.e. ridges below the irregularity pass on unchanged) also

$$\begin{aligned} c(1,j,1) &= c(0,j+1,1) \quad \text{for all } R_1 + 2 \leq j \leq r_1, \\ c(1,j,3) &= c(0,j+1,3) \quad \text{for all } R_1 + 1 \leq j \leq r_1. \end{aligned}$$

(i.e. ridges above the irregularity pass on unchanged, but are displaced downwards by one ridge, due to the $R_1 + 1$'th ridge coming to an end.)

Thus many of the *dummy numbers* from the $(i = 0)$ column are copied into the $(i = 1)$ column — and their successive positions show *which ridge intersection points lie on the same ridges.*

Further information is gained from the immediate vicinity of the irregularity and this allows us to make *entries* in the array. (*Entries* result directly from the coordinate set being processed, rather than by copying from another part of the array).

$$\begin{aligned} c(1,R_1,1) &= 8, \\ c(1,R_1,2) &= 1, \\ c(1,R_1+1,1) &= 6, \\ c(1,R_1+1,2) &= 1. \end{aligned}$$

(i.e. the line l_1 is drawn marginally past the ridge-ending I_1, and so that ridge-ending appears as a facing ridge ending in anticlockwise exploration from ridge intersection points $p(1,R_1)$ and $p(1,R_1+1)$. The event seen, in each case, is I_1 itself.)

We also have *discovered* what happened to the ridge that passed through the point $p(0,R_1+1)$: it ended (code 3) at irregularity I_1. That *discovery* enables us to note the fact

that the ridge exploration clockwise through point $p(0, R_1 + 1)$ ended here. The existing entry in $c(0, R_1 + 1, 3)$ is a *dummy number*, and the new found meaning for that number is recorded in the *dummy number index.* Suppose the dummy entry had been the number 107: then we store its meaning thus:

$$\text{index}(107) = (3,1)$$

Eventually all the appearances of the number 107 in the array will be replaced by '3', and, at the same time, all the associated *irregularity indicators* will be set to '1'.

Knowledge of T_1 and R_1 has therefore enabled us to make a particular set of *associations, entries* and *discoveries* — from which it has been possible to place something (either entries or dummy numbers) in all of the elements of the set

$$\{c(1, j, k) : j = 1, 2, \ldots r_1 : k = 1 \text{ or } 3\}$$

The process now begins again, with examination of irregularity I_2, followed by $I_3 \ldots I_n$. Each different possible *type* code T_i generates its own individual set of associations, entries and discoveries. Each set allows the next column of C to be filled in. ** It should be pointed out that whenever *association* is made of event codes (as distinct from dummy numbers) then association is also made of their respective *irregularity identifiers.*

After all the n coordinate sets have been processed (and entries thereby made in the whole of the continuity array) a few last associations need to be made in order to account for the fact that ridges cross the *cut.* These associations are that :—

$$\begin{aligned} c(0, j, 1) &\quad \text{is equivalent to } c(n, j, 1) \text{ for all } 1 \leq j \leq r_0, \\ \text{and } c(n, j, 3) &\quad \text{is equivalent to } c(0, j, 3) \text{ for all } 1 \leq j \leq r_0. \\ &\quad (\text{Of course } r_0 = r_n = m) \end{aligned}$$

which effectively 'wrap around' the ends of the continuity array by sewing up the *cut.* As each of these elements of C already has some sort of entry in it, the mechanics of making these *associations* are more akin to the normal mechanics of *discovery*, in that they involve making entries in the *dummy number index.* They may, in fact, enter *dummy numbers* in the *dummy number index* thus indicating that two different dummy numbers are equivalent (i.e. they represent the same ridge exploration).

14.2.4 Properties of the completed continuity array.

Once this process is complete the continuity array will have acquired some very important properties:

** Some of the *entries* may well be new (unassigned) dummy numbers. This occurs wherever new ridge segments start at the irregularity. It did not happen in the case of the ridge ending.

(a) all the elements $\{c(i,j,k) : 0 \leq i \leq n : 1 \leq j \leq r_i : k = 1 \text{ or } 3\}$ contain either ridge exploration event codes (hexadecimal) or *dummy numbers* (integers over 100).

(b) wherever $c(i,j,1)$ or $c(i,j,3)$ is an event code, then the corresponding entries, $c(i,j,2)$ or $c(i,j,4)$ respectively, will contain an *irregularity identifying* number that shows *where* that ridge event occurs.

(c) all the different appearances of a particular dummy number in the continuity array reveal all the intersection points through which one continuous ridge exploration has passed. (Hence the name for the array.)

(d) a *discovery* has been made in respect of *every* dummy number that has been allocated, and there is, in the dummy number index, an equivalent event code and associated irregularity identifier waiting to be substituted for all the appearances of that dummy number. The dummy number index is therefore *complete*. This simply *must* be the case as a *discovery* has been recorded every time that a ridge ran into an irregularity. There can be no ridge explorations that do not end at one, or other, of the n irregularities — consequently there can be no outstanding 'unsolved' ridge explorations by the time all n irregularities have been dealt with.

14.2.5 Final stage of topological reconstruction.

The final stage of the reconstruction process is to sweep right through the continuity matrix replacing all the dummy numbers with their corresponding event codes from the index. The related *irregularity identifiers* are filled in at the same time, also from information held in the index. This second (and final) sweep through the elements of the continuity array leaves every element in the set

$$\{c(i,j,k) : i = 1 \ldots n : j = 1 \ldots r_i : k = 1 \text{ or } 3\}$$

as an event code, and every element of the set

$$\{c(i,j,k) : i = 1 \ldots n : j = 1 \ldots r_i : k = 2 \text{ or } 4\}$$

as an irregularity identifier.

For any particular line l_i the entries of C in the ith column correspond exactly to the elements of a topological code vector generated by that line. The only difference in appearance is that we have *irregularity identifiers* rather than *distance measures* to go with each exploration event code. The later vector comparison stages of the matching algorithm are adapted with that slight change in mind.

This completes a somewhat simplified account of a rather complex process. There are other complications which have not been explained in full — such as how the algorithm deals with sequences of dummy numbers that are all found to be equivalent, and the special

treatment that ridge recurves have to receive, and how the algorithm copes with multiple irregularities showing the same angular orientation. Nevertheless this explanation serves well to demonstrate the methodical and progressive nature of this particular reconstruction process. It also makes clear that only two sweeps through the matrix are required — which is surprisingly economical considering the complexity of the operation.

14.3 The matching algorithm LM5.

The algorithm LM5 was the first to accept latent data in coordinate form, rather than by prepared vectors. Topological reconstruction was performed both on the latent mark (once only per search) and on each file print to be compared with it. The continuity matrix generated from the latent coordinate set will be called the *search continuity array*, and the continuity array generated from the file set will be the *file continuity array.*

There are two distinct phases of print comparison which take place after these topological reconstructions are complete. Firstly, the appropriate vector comparisons are performed and their scores recorded — secondly, the resulting scores are combined to give an overall total comparison score.

It is most important to realise that the *observation points* selected on the two prints under comparison do not need to have been in the same positions. The reconstructed topology will be the same *no matter where it was viewed from.* Just as two photographs of a house, from different places, look quite different — nevertheless the house is the same. The final comparison scores will be hardly affected by misplacement of the central observation point *provided they lie in roughly the right region of the print.* The reason for *approximately* correct placement being necessary is that the orientation of the imaginary radial lines, which effectively generate the vectors after reconstruction, will depend on the position of the central observation point. The effect of misplacing that point (in a comparison of mates) is to rotate each generating line about the characteristic on which it is based. Such rotation is not important (as we learnt in part II) provided it does not exceed 20 or 30 degrees. Slight misplacement of the observation point is not going to materially affect the orientation of these imaginary generating lines, except those based on characteristics which are very close to it. Specifying that the central observation point should be adjacent to the core (in the case of whorls or loops) and at the base of the 'upcurve' (in the case of plain arches) is a sufficiently accurate placement rule.

14.4 The vector comparison stage.

From the *search continuity array* a vector is extracted for each true characteristic on the latent mark. Vectors are *not* extracted for the other irregularities ('ridges going out of sight', 'ridge recurves', etc.) If the latent mark shows 13 characteristics we then have 13 vectors, each vector based on an imaginary line drawn from the central observation point to one of those 13 characteristics, and passing marginally to the clockwise side of

it. Let us now forget about all the other topological irregularities in the coordinate list and number the characteristics $1, 2, 3, \ldots k$. If the number of coordinate sets, in total, was n then certainly $k \leq n$. The extracted search vectors can now be called $S_1 \ldots S_k$. In a similar fashion the extracted file vectors, each based on true characteristics, can be called $F_1 \ldots F_m$.

For each search vector a subset of the file vectors is chosen for comparison. The selection is made on these bases :—

(a) that the characteristic on which the file vector is based must be of similar *type* (either an 'exact' match or a 'close' match) to the one on which the search vector is based.

(b) that the angular coordinates of the characteristic on which it is based must be within a permissible angular tolerance of the angular coordinate of the characteristic on which the search vector is based. The permissible angular tolerance is a parameter of the algorithm.

This selection essentially looks for file print characteristics that are potential *mates* for the search print characteristics. The vector comparison that follows serves to compare their neighbourhoods. It is quite obvious that allowing a wide angular tolerance significantly increases the number of vector comparisons that have to be performed. If a small angular tolerance is permitted then a badly misoriented latent mark may not have the mated vectors compared at all.

The vector comparison itself is much the same as used hitherto — except that the vectors contain *irregularity identifiers* rather than *distance measures.* At the appropriate stages of the vector comparison subroutine the *actual linear distance* ('as the crow flies') from the central characteristic to the ridge-event is calculated by reference to the appropriate coordinate sets. Thus ordinary *spatial* distances can be used rather than *inferred ridge-traced* distances, and a much greater degree of reliability can therefore be attached to them.

For each search vector S_i, and candidate file vector F_j, a vector comparison score q_{ij} is obtained. For each search vector S_i a list of candidate file vectors, with their scores, can be recorded in the form of a list of pairs (j, q_{ij}). There are typically between 5 and 15 such candidates for each search vector when the angular tolerance is set at 30°. These lists of candidates can then be collected together to form a table, which will be called the *candidate minutia table.* An example of such is shown overpage.

Each column is a list of candidates for the search vector labelled at the head of the column. In each case the first of a pair of numbers in parentheses shows *which* file vector was a candidate, and the second number is the score obtained by its vector comparison.

S_1	S_2	S_3	...	...	...	S_k
(5,89)	(6,45)	(25,41)	...	...	...	(15,138)
(14,29)	(10,40)	(34,12)	...	...	...	(23,12)
(15,0)	(16,35)	(37,19)	...	...	...	(28,65)
(52,19)	(21,92)	(41,84)	...	...	...	(36,71)
(55,81)	(35,5)	(48,91)	...	...	...	(37,103)
(61,34)	(36,0)	(53,101)	...	...	...	(47,82)
(79,0)	(41,3)	(65,180)	...	...	...	(56,41)
.	.	.	...	...	...	.
.	.	.	...	...	...	.
.	.	.	...	...	...	.
.	.	.	...	...	...	.
(0,0)	(46,85)	(0,0)	...	...	...	(0,0)

14.5 Final score formulation.

We are now left with the problem of intelligently combining these individual candidate scores to give one overall score for the print. If the file print and latent mark are mates it would be nice to think that the highest candidate score in each column of the candidate minutia table indicated the correct matching characteristic on the file print. If that were the case then simply picking out the highest in each column, and adding them together, might serve well as a method of formulating an overall score. However that is *not* the case. Roughly 50% of true mated characteristics manage to come top (in score) of their column — the others usually come somewhere in the top five places.

14.5.1 The notion of 'compatibility'.

We learnt from earlier experiments with latent entry by vectors that combination of scores was best done subject to *conditions* — and, in that case, the condition was correct relative angular orientation (see para 13.5(b)). It will make sense, therefore, to combine the individual candidate scores when, and only when, they are *compatible.*

If (j, q_{1j}) is a candidate in the S_1 column, and (i, q_{2i}) is a candidate in the S_2 column — then there are various reasonable conditions that can be set in respect of these two candidates before we accept that they could *both* be correct. We will say that these two candidates are *compatible* if, and only if, these three conditions hold true :—

(a) i is not equal to j . (Obviously *one* file print characteristic cannot simultaneously be correctly matched to two different search print characteristics.)

(b) The distance (linear) between file print characteristics numbered i and j should be the same, within certain tolerance, as the distance between the two search

print characteristics that they purport to match. That tolerance is an important programme parameter.

(c) The relative angular orientation of the file print characteristics should be roughly the same as the relative angular orientation of the two search print minutiae that they purport to match. The tolerance allowed, in this instance, is the same angular tolerance that was used earlier to limit the initial field of candidate minutiae.

14.5.2 Score combination based on compatibility.

The application of the notion of compatibility in formulating a *total* score was originally planned as follows :—

Step 1: Reorder the candidates in each column by reference to their scores, putting the highest score in each column in top place.

Step 2: In each column, discard all the candidates that do not come in the top five places.

Step 3: For each remaining candidate check to see which candidates *in the other columns* are compatible with it.

Step 4: Taking at most one candidate from each column, pick out the *highest scoring mutually compatible set* that can be found. A *mutually compatible set* is a set of candidates each pair of which are compatible.

Thus a set of file print characteristics is found, each of which has similar topological neighbourhood to one of the latent mark characteristics (as shown by their high vector comparison scores) and whose *spatial distribution* is very similar to that of the latent mark characteristics (as shown by their *compatibility*). Spatial considerations are therefore being used in the combination of topological scores — as is already the case at a lower level, when distance measures are used in the vector comparison process.

The algorithm LM5 was originally written to perform the steps described above. Unfortunately it ground to a halt completely when it tried to do the comparison of a *very good latent* with its mate! The reason for this is that the algorithm will examine every possible mutually compatible set in turn. Certainly non-mates have very few mutually compatible sets of any size. However, if a good quality latent gives a largest compatible set of size N (i.e. N characteristics match up well with the file print) then there are $2^N - 1$ subsets of that largest set, each of which will be a mutually compatible set. The total number of such sets is therefore *at least* 2^N, and probably much greater. In some cases N exceeded 25 and, consequently, the computer did not finish the job!

Acceptable shortcuts, or approximations, to this method had to be found.

14.5.3 Candidate promotion schemes.

The following method accomplishes much the same sort of candidate selection, but very much faster, and without requiring complete mutual compatibility in the selected set. The first three steps are the same as before :—

1. Reorder the candidates in each column, by their scores.

2. Discard all candidates not ranked in the top 5 places in their column.

3. Check the compatibility of all remaining candidates with the remaining candidates in each other column.

The fourth step is calculation of what will be called a *compatible score* for each of the remaining candidates. Here are two possible alternative methods for doing this :—

(a) For each individual candidate add together all the scores of top-ranked candidates in *other columns* with which that candidate is compatible. Finally add the candidate's *own* score to the total.

(b) For each individual candidate find, in each other column, the highest scoring compatible candidate. Add together those scores (one from each column), and then add the target candidate's *own* score to the total.

On the basis of these *compatible scores*, rather than on the original vector comparison scores, reorder the remaining candidates in each column.

This 4th step can be regarded as a promotion system based on compatibility with other high-ranking candidates. The difference between options (a) and (b) is this: in rule (a) promotion depends on a candidate's compatibility with those *already in top place* (and could be called a 'bureaucratic' promotion system). With rule (b) a whole group of candidates in different columns, *none of whom are in top place* can all be promoted to the top at once by virtue of their strong compatibility with *each other* (a 'revolutionary' promotion system). Both were tried and the 'revolutionary' system was found to be the most effective.

The *promotion stage* could be repeated several times if it was considered desirable (to give the top set time to 'settle') — in practice it was found that one application was sufficient. Mate scores improved very little, if at all, when second and third stages of promotion were introduced.

After the promotion stage is complete all but the top ranked candidates in each column are discarded, and the *compatible score* for the remaining candidate in each column is then recalculated on the basis of only the other remaining candidates.

The final score is then evaluated by adding together all of these new *compatible*

scores that exceed a given threshold. That threshold is a programme parameter, and is expressed as a percentage of the 'perfect' latent self-mated score.

The use of these *compatible scores*, rather than the original vector comparison scores, in evaluating the final score has the effect of multiplying each original vector score by the number of other selected (i.e. now top-ranked) candidates with which it is compatible. The more *dense* the compatabilities of the final candidate selection, the higher the score will be.

14.6 Performance of LM5.

The latent mark data file was converted to the form of coordinate sets, and the fourth coordinate (distance) was added into the file print collection data set. A series of tests was then performed using the algorithm LM5 — and the results and parameters used are shown in full in appendix P.

The best test results obtained gave the following rankings :—

MR1	80.36%
MR3	82.14%
MR10	85.71%

These indicate a *vast* improvement over the performance of the traditional spatial methods (recall that the M82 algorithm gave test results with an MR1 value of 26.8%).

It is worth saying a few words about some of the parameter values that gave the above results :—

(a) Exact match scores were set to be 5, with close match scores (CMS) set to be 3. Thus close match scores were given a higher relative weighting than previously used in the comparison of rolled impressions (where the optimum ratio had been 5:1 — see para 10.5(b)). The higher weighting can be attributed to a higher incidence of *topological mutation* in the interpretation of latent marks.

(b) The distance tolerances were set at 10% (of the distance being checked) with a minimum of 1. (PDT, in appendix P, stands for 'percentage distance tolerance', and MDT for 'minimum distance tolerance'.) The same distance tolerances were used in the vector comparison stage of the algorithm and in the score combination stages (where correct relative distance was one of the three conditions that needed to be satisfied for two file print minutiae to be *compatible*.)

(c) The ridge span used in vector comparison was 10 ridges — this means that vectors of a standard length of 40 digits, with 40 associated irregularity indicators, were used whenever vector comparisons were performed. The results were no worse with longer vectors, but the smaller value for SPAN gave faster comparison times on a serial machine.

(d) The minimum angular tolerance (MAT) was 20°. This is almost inconsequential as the true angular misorientation limits were set individually for each latent mark (by subjective judgement) and written as a part of the latent search data.

(e) The *candidate minutia selection depth* ('DEPTH') was 5 throughout. This means that, for each search minutia, only the top 5 candidate file print minutia would be considered. This parameter was set to 5 as a result of observation, rather than experiment (see para 14.5)

(f) The *compatible score cutoff point* ('CUTOFF') is the percentage of the latent mark's perfect self-mated score that must be attained by the final compatible score of a candidate file print minutia before it will be allowed to contribute to the final total score (see para 14.5) The best value for this parameter was found to be 15%, which is surprisingly high. The effect of this setting was to ensure that the vast majority of file print minutiae that were not true mates for search minutia contributed nothing to the score; the net effect of this was to make *most* of the mismatch comparison scores zero. In fact, for 28.6% of the latents used, the true mate was the *only* file print to score at all — the other 99 file prints all scoring zero. Of course such a stringent setting also made things tough for the mates, as shown by the fact that 7% of the mate scores were zero also. However, these 7% were mates that had not made the top ten places in any of the tests, and were therefore most unlikely to be identified anyway. It is also worth pointing out that on each occasion when one file print alone scored more than zero (i.e. exactly 99 out of the 100 in the file collection scored zero) that one was the true mate. (These are the 28.6% mentioned above.) This represents a surprisingly high level of what might reasonably be termed 'doubt-free identifications'.

14.7 Computation times.

The foregoing description of the algorithm LM5 will have made it quite clear that this is not, in its present form, a particularly fast comparison algorithm. The CPU time taken on a VAX 11/780 for the above test (5600 comparisons) was 12 hours and 11 minutes. [Hence the absence of any extensive parameter tuning.] That means an average CPU time per comparison of 7.8 seconds — which is a somewhat disconcerting figure when the acceptable matching speeds for large collections are in the order of 500 comparisons per second.

However 7.8 seconds per comparison is not quite so alarming when one considers the extensive and multi-layered parallelism of the algorithm. At the lowest level, the vector comparisons themselves are sequences of array operations. At the next level, many vector comparisons are done per print comparison. In the score combination stages calculations of compatibility and compatible scores are all simple operations repeated many many times. There is, in this algorithm, *enormous* scope for beneficial employment of modern parallel processing techniques. It is hardly appropriate to take too much notice of the CPU time

in any serial computer — where each operation is done element by element.

Moreover, in the area of latent searching, the primary area of concern for law enforcement agencies is shifting from the issue of *speed* onto the issue of *accuracy*. The FBI, for example, is certainly prepared to obtain the necessary speed through 'hardwiring' (with its associated cost) for the sake of matching algorithms that will actually make a substantial number of identifications from latent marks.

14.8 File storage space — defaulting the 'edge topology'.

It is noticeable that the need to include *all topological irregularities*, rather than just the true characteristics, significantly enlarges the volume of the file print data. In the 100 file cards in the experimental database the average number of irregularities recorded per print was 101.35. The majority of irregularities that were not true characteristics fell at the *edge* of the print; they recorded all those places where ridges 'came into sight' or 'went out of sight'. Thus a significant proportion of the file data storage requirement is spent in describing the edge of the file print.

In practice the edge of the file print is not very important — as the latent mark invariably shows an area completely *within* the area of the rolled file print. The edge consequently plays little or no part in the print comparison process, and the edge description serves only to help the topological reconstruction process make sense of the ridge pattern.

For the sake of economy in file size, therefore, the algorithm LM6 was prepared by adapting the reconstruction stage of LM5 slightly. It is adapted in such a way that the reconstruction will *invent its own edge topology* in the absence of an edge description. The *default topology* selected is not important; it is only important that the algorithm does *something* to tie up all the loose ridges around the edge.

The file collection was then pruned substantially by elimination of all of the edge descriptions, and this reduced the average number of coordinate sets per print from 101.35 to 71.35. * The test reported above was then rerun using the algorithm LM6 and the condensed file set. The rankings obtained were *exactly the same* as before (see para 14.6) — so a saving of 30% in file data storage was achieved with absolutely no loss of resolution.

---

* The pruning operation was not performed on the latent mark data file for two reasons. Firstly, latent mark databases (where these are kept) are tiny in comparison to rolled file print collections, and so storage requirements are not a major concern. Secondly, the edge of a latent mark does play an important part in the comparison process.

CHAPTER 15.

ASSOCIATED APPLICATIONS AND CONCLUSIONS.

15.1 Derivation of vectors for rolled print comparison.

The ability to perform topological reconstruction from a set of coordinates has some rather interesting 'by-products'. The first of these relates to the fast comparison of rolled prints on the basis of a single vector (as per part II.)

As the data format for a latent mark and a rolled impression is now identical, it would be possible to use the latent matching algorithm (LM6) to compare one rolled print with another. (One of the rolled prints would be acting as a very high quality latent.) However, to use LM6 in this way on rolled prints would be 'taking a sledge hammer to crack a nut'. We know, from Part II, that one single vector comparison deals with comparison of two rolled prints perfectly adequately — so it would be madness to use this latent matching algorithm, with its hundreds of vector comparisons, in this application.

Nevertheless there is a significant benefit to be gained from the topological reconstruction section of the latent matching algorithm. The data-gathering requirements from Part II included the need to *track along ridges, in order to find the first event that happened.* Although that, in itself, is not a particularly demanding programming task — the ability to reconstruct topologies from coordinates renders it unnecessary. A topological code vector representing a *horizontal* line passing through the core of a loop can be lifted out of the continuity matrix after reconstruction. The left half of it (i.e. the part that falls to the left of the core) and the right half will be extracted separately. Each half is extracted by selecting the column of the continuity matrix that corresponds with an imaginary line *just to the anticlockwise side* of horizontal. (i.e. just *below* for the left side, and just *above* for the right side). Amalgamating these two halves, reversing the 'up' and 'down' pairs from the right half, gives a single long vector of the required format.

There will be two minor differences between these extracted vectors and the design originals :—

(a) the core point, which was to be on a ridge, is replaced by the *central observation point* which is in a valley. The central observation point will, however, be only fractionally removed from the core in the case of loops and whorls.

(b) the vector has *irregularity identifiers* rather than *ridge-traced distance measures.* Consequently the vector comparison algorithm has to be adapted to refer to the appropriate coordinate sets when the time comes to apply the various distance tests.

In the case of arches the extracted vector will have to be a *vertical, straight* line as opposed to the original flexible one which followed successive ridge summits. *

In an operational system the maximum speed would be obtained by performing topological reconstruction, and vector extraction, *at the time each print is introduced to the collection.* The extracted 'long' vectors could be stored in a separate file so that they could be used for fast vector comparison without the need to perform topological reconstruction each time. That would obviously increase the data storage requirement per print by the 60 bytes required for such 'long' vectors (see chapter 10). The coordinate sets, and topological reconstruction would then only be used when a *latent search* was being conducted.

If the derived long vectors were to be made completely independent of the coordinate sets, it would be necessary to replace the *irregularity identifiers* with calculated *linear distances* at the time of vector extraction.

15.2 Image-retrieval systems.

The second by-product of the development of the latent matching algorithms is an application in image-retrieval systems. There is a significant demand for automated identification systems to be linked with an image-retrieval facility for all the prints in the file collection. The system operator obtains a list of the highest scoring candidates each time an automated search is conducted — these candidates have then to be checked visually by the fingerprint expert to determine which of them, if any, is the true mate. This visual checking can be done much more easily if the fingerprints can be displayed on a screen, rather than having to be fetched from a cupboard. Much research is currently underway with the aim of finding *economical* methods for storing the two dimensional pictures (fingerprints) in computer memory so that they can be called up and displayed on the terminal screen.

There are two distinct paths for such research. The first aims to record the original grey-scale data which is output from automatic scanners, with no interpretative algorithms ever being applied to the print (although data compaction techniques will, of course, be used). The second uses interpretative algorithms to identify the ridges and valleys within the grey-scale image, to resolve the picture into a binary (black and white) image, and then finally to reduce the thickness of each ridge to one pixel by a variety of ridge-thinning techniques. What is then stored is sufficient data to enable each thinned ridge segment to be redrawn (i.e. start position, end position, curvature etc.).

---

* The performance of MATCH4 on such derived vectors has not been tested. This is because of the incredibly time consuming nature of manual encoding according to the latent scheme (up to 1 hour per print for clear rolled impressions). The time for such tests will be after the development of automatic data extraction techniques, when large numbers of prints can be encoded automatically according to the latent scheme, and then have derived vectors extracted after topological reconstruction.

The data requirements per print are in the order of 2,000 to 4,000 bytes for compressed grey-scale images, and between 1,000 and 2,000 bytes for a thinned image.

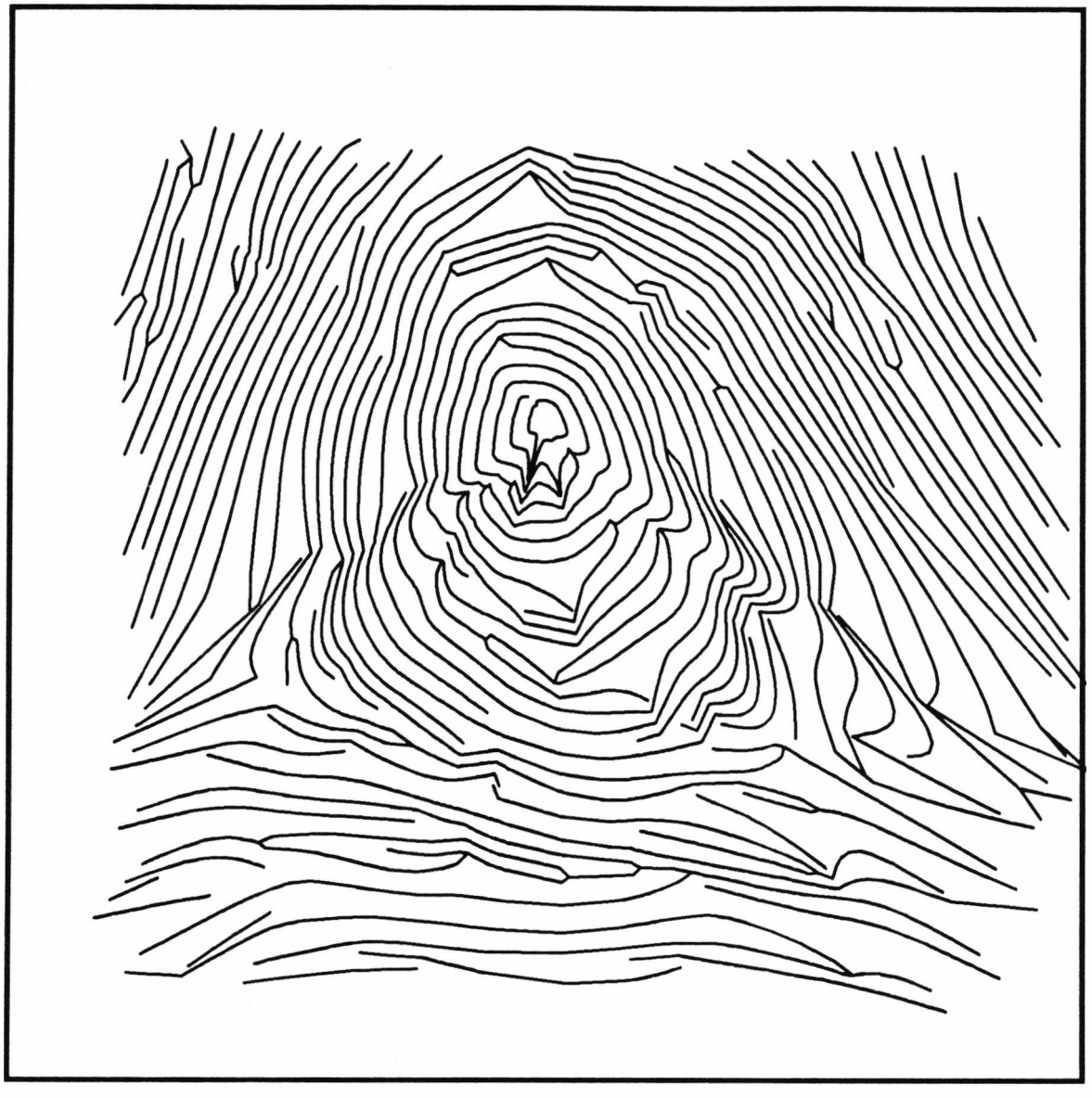
We know that the 4-coordinate system used in the latent scheme records, in between 300 and 400 bytes, a *complete topological and spatial* description of the characteristics. It should therefore be possible to *redraw* the fingerprint, in the style of a thinned image, from that data. Firstly topological reconstruction has to be performed, and then the *elastic* (topological) image has to be 'pinned down' at each characteristic, by reference to their polar coordinate positions contained in the coordinate sets.

The substantial problem in such a process is the business of generating a *smooth* ridge pattern that accommodates all the pinned points. The problems raised are not completely dissimilar to those in cartography — when a smooth contour map has to be drawn from a finite grid of discrete height (or depth) samplings.[39] [40] Certainly if a satisfactory redrawing process could be devised, the 4-coordinate system would, almost certainly, be the most economical method of image storage available.

Development of adequate smoothing algorithms was not adopted as a part of this research; it is a fairly major research problem in itself. However one fairly crude reconstruction algorithm was written, simply because generation of a *picture* from topological coordinate sets provides a most satisfying demonstration of the sufficiency of such coordinate descriptions.

The algorithm PLOT1 was written as a Fortran programme: its input was the set of coordinates representing a specified print, and its output was a file of 'QMS-QUIC' instructions for the graphics display facility of a QMS LASERGRAFIX 1200 printer. The algorithm first performed topological reconstruction in the normal manner, and then assigned polar coordinates to every ridge intersection point in such a manner that all the topological irregularities were assigned their own (real) polar coordinates. A series of simple linear *smoothing* operations are applied, coupled with *untangling* and *gap-filling* procedures that make successive small adjustments to the radial distances of all the intersection points that are *not* irregularities. These processes continue until a certain standard of smoothness is attained. Finally the picture is output as a collection of straight line segments between connected ridge intersection points.

A sample reconstructed fingerprint image is shown in figure 56, together with its descriptive data. The picture is made up of 4,404 straight line segments, and it almost looks like a fingerprint! Certainly the topology is correct, and each irregularity is properly located: it is just the intervening ridge paths that have suffered some unfortunate spatial distortions. For the sake of comparison, the original print tracing from which the coordinate sets were derived is shown in figure 57 (it has been reduced from 10× to 5× magnification). Detailed comparison of figures 56 and 57 will reveal a few places where the topology appears to have been altered. In fact it has not been altered — but, at this magnification, some ridges appear to have touched when they should not. This tends to occur where the ridge flow direction is close to radial. In such places the *untangling* sub-



FINGERPRINT RECONSTRUCTION DATA:

Card number 6. Finger number 8
Window size: 6"
Magnification : 5.00
Downward displacement of origin : -700
Number of line segments drawn: 4404
Fingerprint data size : 526 bytes.

Figure 56. Fingerprint reconstruction.

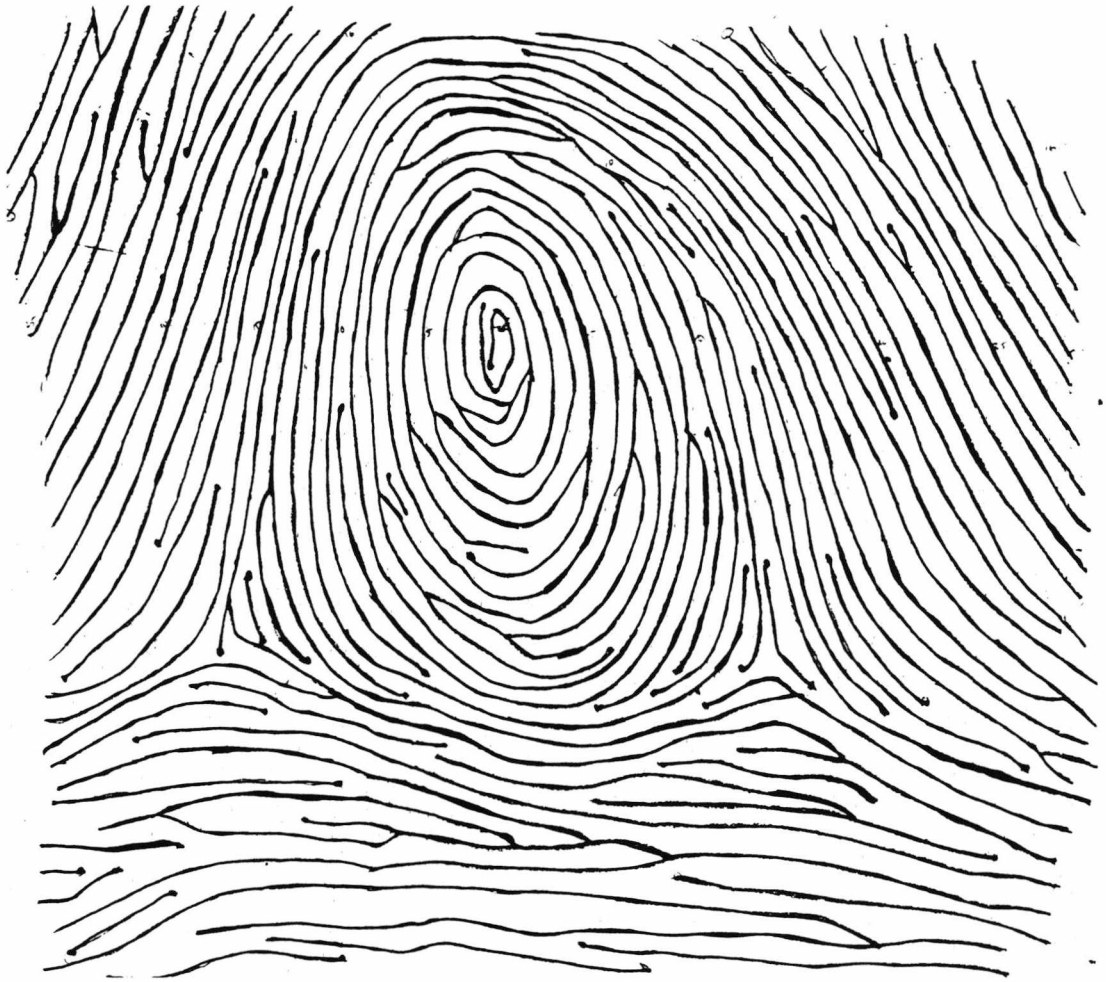


Figure 57. Copy of fingerprint tracing.

routine, which moves ridges apart when they get too close together, has not been forceful enough in separating them.

Figure 58 shows the tracing of a latent mark, together with its reconstructed picture. In this case the latent data comprised 32 coordinate sets (filling approximately 100 bytes), of which 21 make up the edge-description. There are ten genuine characteristics shown, and the remaining topological irregularity is the ridge recurve close to the core. The reconstructed image is made up from 780 straight line segments.

The facility for reconstruction also affords the opportunity to actually *see* a 'default edge-topology'. Figure 59 shows two further reconstructed images of the print in figure 56. The upper picture is the same as figure 56, except for a reduction in magnification (to 2.5×). The lower picture is a reconstruction from the condensed data set for the same print, after all the coordinate sets relating to ridges going 'out of sight' have been deleted. All the loose ends have been tied up by the reconstruction algorithm in a fairly arbitrary, but interesting, way. The lower picture does, of course, show some *false* ridge structure in areas that were 'out of sight'. However the data storage requirement for the corresponding

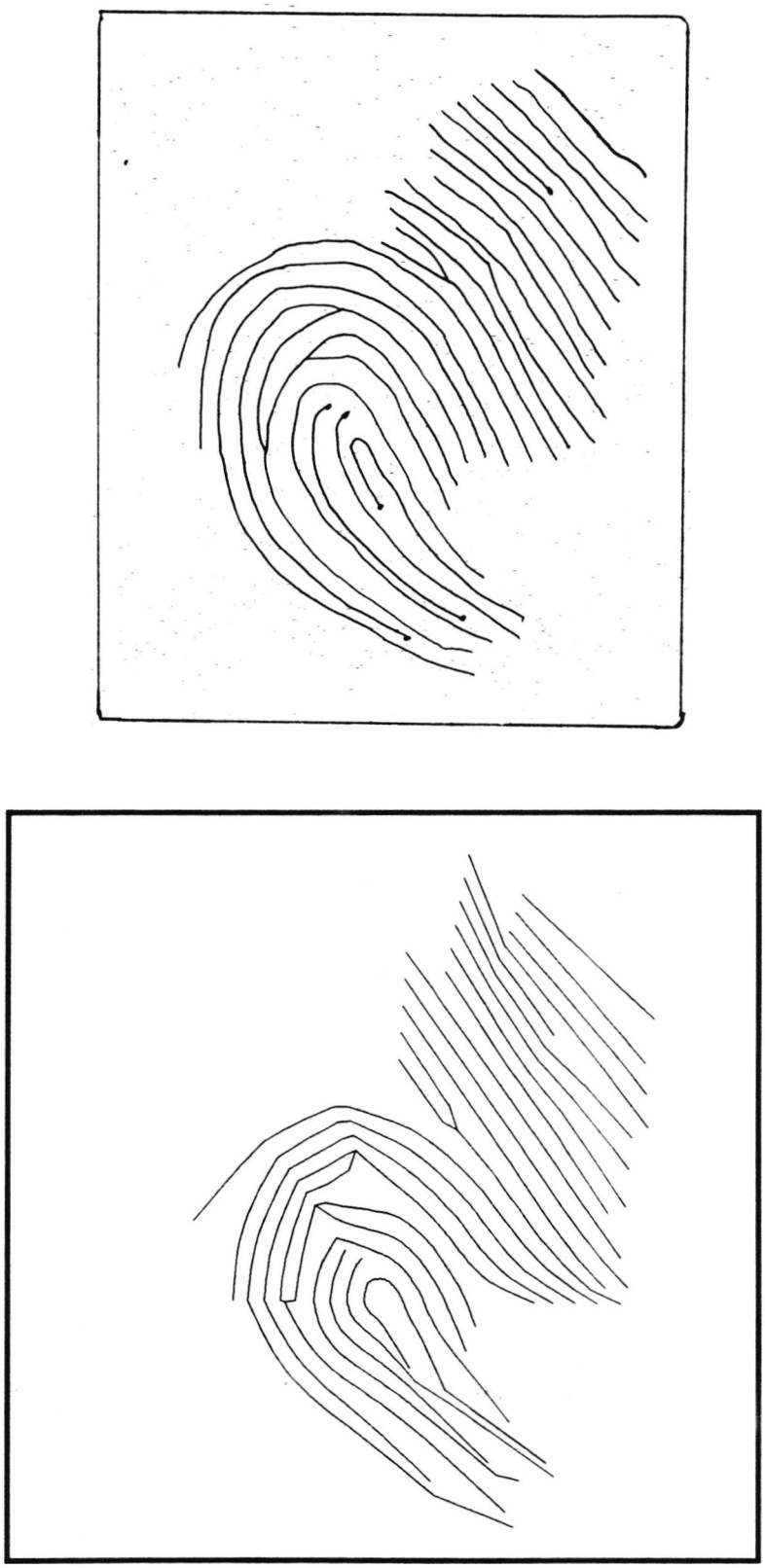


Figure 58. Latent tracing, and its reconstruction.

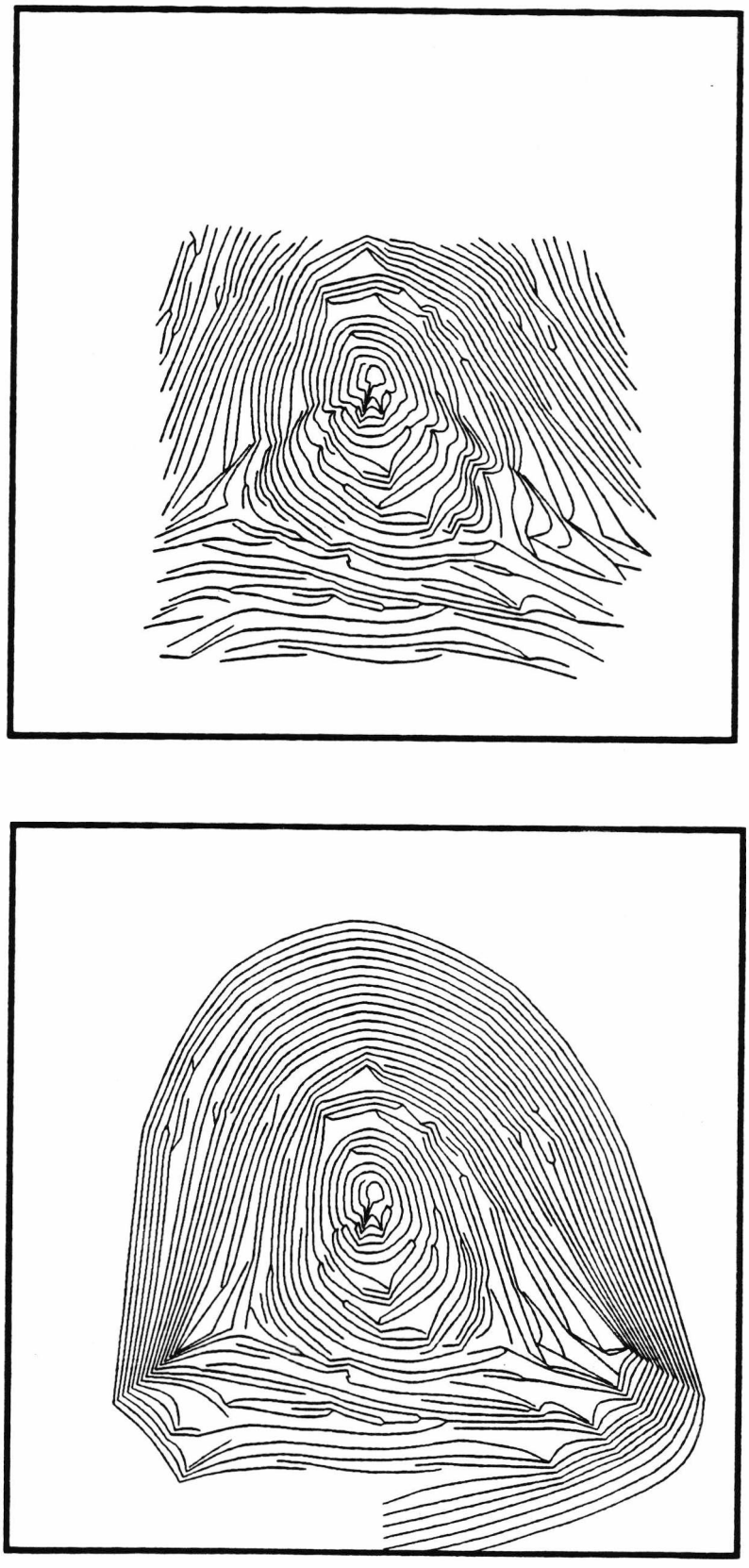


Figure 59. Reconstructions with, and without, defaulted edge-topology.

coordinate sets was only 354 bytes for the edge-free description, as opposed to 526 bytes for the original description.

From these pictures it is fairly clear that more sophisticated smoothing techniques will need to be applied before really reliable images can be retrieved. These pictures are quite sufficient nevertheless to demonstrate the potential for such a scheme. They are also a fine demonstration of the effectiveness and accuracy of the topological reconstruction algorithms. *

15.3 Outline of further work to be done.

The work outlined in this paper has lead to development of systems which could be implemented now — but which would require a *manual* file-print encoding process. It was, of course, the intention that such datafile conversion should be an automatic process; consequently development of such necessary data extraction algorithms would be desirable. A list of possible areas for further research is given here :—

(a) Automatic data gathering algorithms should be designed which are capable of extracting the required forms of data from the grey-scale output from automatic fingerprint scanners. For the reasons given in paragraph 15.2 the ability to track along ridges is *not* required. However the ability to locate every interruption of the otherwise smooth ridge flow in the print *is* needed. Moreover each interruption has to be *typed* according to the table of possibilities laid out in paragraph 13.3. 'Unclear' areas, rather than simply being rejected, must be *fenced off* — and all the places where ridges run into the *fenced area*, or emerge from it, must be recorded. This is a substantial departure from current practice; normally unclear areas would simply be rejected.

(b) Once such data-gathering algorithms have been written, and sizeable experimental databases built up — then the various parameters of the matching algorithms must be tuned finely by extensive experiments. Optimum parameter values for use on automatically read data are unlikely to be identical to their optimum values for manually prepared databases.

(c) Some investigation should be conducted in order to determine if there is any value in including a fifth coordinate, namely 'ridge direction', for each characteristic. No use of ridge direction data has been made in any of these topological schemes, even though it is the standard third coordinate for all the existing spatial methods (where (X, Y, θ) is the coordinate format for each characteristic, and θ is the ridge

* remember that the *path* of the ridges plays no part in the comparison algorithms LM5 and LM6; only the topology, and the positions of the characteristics are used. The defects in these pictures are not, therefore, a reflection of defects in the latent searching algorithms.

flow direction local to each particular characteristic.) There are a number of places within the various topological matching algorithms where tests on ridge direction could be applied in conjunction with consideration of angular misorientation. It is felt, however, that sufficient spatial information is already in use, and that the dividends would be too small to justify the 25% increase in data storage requirement that such a change would inevitably produce.

(d) An appropriate parallel architecture for the algorithms MATCH4 and LM6 has to be developed in conjunction with selection of the most suitable of the available parallel processors.

15.4 Conclusion.

The results obtained in these experiments show, beyond any reasonable doubt, that a topological approach to fingerprint coding offers a great deal in terms of improved accuracy and cost-effectiveness. It is also clear that topology based matching algorithms are greatly improved by utilising some spatial information. The power of resolution between mates and non-mates given by the *combination* of topological and spatial information is vastly superior to that which can be obtained by use of spatial information alone.

The greatest benefit that has been obtained is *accuracy*. With rolled impressions there is also a clear increase in *speed* and a massive reduction in storage requirement. With the latent searching scheme the question of *speed* has to be left open until the benefits of LM6's extensive parallelism have been realised.

It is certainly not 'too late' in the development of automated systems for use of these techniques to be made. In fact the time is probably ripe. Only in very recent years have national law-enforcement agencies begun to contemplate *international* fingerprint searching facilities; new initiatives to standardise the formats of fingerprint data on an international basis are now being taken.* This initiative results from an expectation that, with available computing power increasing so rapidly, it will not be very long at all before international fingerprint searches will be a practical proposition.

Whilst such 'dreams' are seen as only a few years off, it still remains the case that the vast majority of law-enforcement agencies worldwide do not have any automated searching equipment at all. Britain only has a small pilot scheme (in parts of London), and does not have plans for establishing a national scheme until 1990. France has just opened a pilot scheme (in Versailles). Germany is currently considering buying its first automated system — and so is Australia. Sweden recently purchased one. A few of the

* The National Bureau of Standards is accredited by the American National Standards Institute for the development of an American National Standard for fingerprint data. Participants in the development process include virtually all major users and manufacturers of automated systems worldwide.

provincial American Police forces have operational systems, but most are fairly small. The Federal Bureau of Investigation has an *enormous* automated system — but is currently very uncertain as to the quality both of its data and of its matching techniques. At the time of writing this paragraph the total number of latent marks *ever identified by computer* at the FBI does not exceed 10.

The difference between what *is now*, and what will be in a few years is quite startling. It suggests that the next few years are going to be years of rapid growth in fingerprint automation. The anticipation of massive national and international systems throws the research emphasis fairly and squarely back onto two essential system ingredients — accuracy, and speed.

The development of more and more powerful computers (at steadily diminishing cost) may well provide all the speed that such systems require. However it is practically certain that conventional spatial comparison methods cannot provide the corresponding increase in accuracy.

That is why this project is *not* too late.

REFERENCES.

[1] Andre A. MOENSSENS, **"Fingerprint techniques"**, (Chiltern Book Company 1971), page 33.

[2] MOENSSENS, page 52.

[3] Robert D. OLSEN, **"Scott's Fingerprint Mechanics"**, (Charles C. Thomas 1978), page 11.

[4] OLSEN, pages 405 - 426.

[5] OLSEN, page 26.

[6] OLSEN, page 9.

[7] D. VENKAIAH, **"Laws of Prints and Impressions"**, (Law Book Co. Allahabad 1979), page 84.

[8] FBI, **"The Science of Fingerprints"**, (U.S. Government Printing Office 1979), pages 8 - 13.

[9] MOENSSENS, page 16.

[10] OLSEN, page 28.

[11] Frederick R. CHERRILL, **"The Fingerprint system at Scotland Yard"**, (Her Majesty's Stationery Office 1954), page 25.

[12] MOENSSENS, page 64.

[13] FBI, pages 8 - 17.

[14] CHERRILL, pages 83 - 84.

[15] CHERRILL, page 22.

[16] CHERRILL, pages 33 - 78.

[17] FBI, pages 113 - 115. Also, Anthony L. CALIFANA, **"Simplified version of the NCIC technique for coding fingerprints"**, Parts I, II, III, *Law and Order*, March, April, May 1974.

[18] MOENSSENS, page 192 ff.

[19] MOENSSENS, chapter 8.

[20] MOENSSENS, page 243.

[21] Parduman SINGH, **"Pressure distortions in fingerprinting"**, *Police Journal*, February 1963, page 82.

[22] Conrad S. BANNER, **"The state of development of the FBI's automatic fingerprint identification system"**, *FBI Law Enforcement Bulletin*, June/July 1973.

[23] Conrad BANNER & Robert STOCK, **"The FBI's approach to automatic fingerprint identification"**, *FBI Law Enforcement Bulletin*, January 1975, pages 2 - 9.

[24] MOENSSENS, page 244.

[25] SINGH.

[26] MOENSSENS, page 244.

[27] Dr.R.M.APPLEBY, **"AVR: a new approach to some problems of fingerprint comparison"**, *Police Journal*, January 1979, pages 57 - 60.

[28] CHERRILL, pages 83 - 84.

[29] VENKAIAH, pages 98 - 99.

[30] VENKAIAH, page 91.

[31] VENKAIAH, page 67.

[32] VENKAIAH, pages 252 - 253.

[33] MOENSSENS, page 199 ff.

[34] OLSEN, page 36.

[35] FBI, pages 8 - 17.

[36] R.T.MOORE and J.R.PARK, **"The graphic pen, an economical semi-automatic fingerprint reader"**, *1977 Carnahan Conference of Crime Countermeasures, Lexington, Kentucky*, pages 59 - 62.

[37] Emanuel PARZEN, **"On estimation of a probability density function and mode"**, *Annals of mathematical statistics*, volume 33, pages 1065 - 1076, Institute of Mathematical Statistics, 1962.

[38] Joseph H. WEGSTEIN, **"An automated fingerprint identification system"**, *N.B.S. Special Publication 500 - 89*, February 1982.

[39] Thomas M. DAVIS and Angelo L. KONTIS, **"Spline interpolation algorithms for track-type survey data with application to the computation of mean gravity anomalies"**, (U.S. Naval Oceanographic Office 1970).

[40] R.J.VANWYCKHOUSE,**"Synthetic bathymetric profiling system (Synbaps)"**, (U.S. Naval Oceanographic Office 1973).

BIBLIOGRAPHY.

APPLEBY, Dr.R.M. **"A.V.R: a new approach to some problems of fingerprint comparison."** *Police Journal*, January 1979, pages 57 - 60.

BANNER, Conrad S. **"The FBI approach to automatic fingerprint identification."** Conrad S.Banner & Robert M.Stock. *FBI Law Enforcement Bulletin.* January 1975, pages 2 - 9 & 24.

BANNER, Conrad S. **"The state of development of the FBI's automatic fingerprint identification system."** *FBI Law Enforcement Bulletin.* June and July 1973.

CALIFANA, Anthony L. **"Simplified version of the NCIC technique for coding fingerprints."** Parts I, II, III. *Law and Order.* March, April and May 1974.

CHERNOFF, Herman. **"Some applications of a method of identifying an element of a large multidimensional population."** *Multivariate Analysis — IV, edited by P. R. Krishnaiah.* North-Holland Publishing Company, 1977.

CHERRILL, Frederick R. **"The fingerprint system at Scotland Yard."** Her Majesty's Stationary Office, London, 1954.

DAVIS, Thomas M. and KONTIS, Angelo L. **"Spline interpolation algorithms for track-type survey data with application to the computation of mean gravity anomalies."** Technical Report 226, U.S. Naval Oceanographic Office, Washington, D.C., 1970.

ERIKSSON, Sven Arne. **"The application of computerised information retrieval to fingerprint identification."** *International Criminal Police Review.* August 1976, pages 194 - 201.

FBI. **"The science of fingerprints."** U.S. Government printing office, 1979.

FLETCHER, T.A. **"Fingerprinting techniques."** *Police review.* 19/8/1977, pages 1104 - 1106.

GODSELL, J.W. **"A method for converting regional main fingerprint collections."** Home Office, Police Research and Planning Department, 1967.

GRANT, Douglas. **"Why the unknown dead?."** *Police Journal.* July 1977, pages 248 - 251.

HANKLEY, W.J. and TOU, J.T. **"Automatic fingerprint interpretation and classification via contextual analysis and topological coding: Pictorial pattern recognition."** Thompson Book Company, Washington, D.C., 1968.

KAWASHIMA, Misao and HOSHINO, Yukio. **"Automatic reading and matching method for fingerprint identification."** International Conference on Fingerprints, London. November 1984.

MACLEAN, George. **"Searching single fingerprints."** Identification Bureau, City of Glasgow Police, 1970.

MENZEL, E.R. **"Fingerprint detection with lasers."** Marcel-Dekker Inc., New York, 1980.

MILLARD, K. **"Developments on automatic fingerprint recognition."** International Carnahan Conference on Security Technology, Zurich, Switzerland, October 1983.

MOENSSENS, Andre A. **"Fingerprint techniques."** Chiltern Book Company, 1971.

MOORE, R.T. and MCCABE, R.M. **"An overview of algorithms for matching fingerprint minutiae in latent searches using relative positional information of the minutiae."** International Conference on Fingerprints, London, November 1984.

MOORE, R.T. and PARK, J.R. **"The graphic pen, an economical semiautomatic fingerprint reader."** Carnahan Conference on Crime Countermeasures, University of Kentucky, Lexington, Kentucky, 1977, pages 59 - 62.

OLSEN, Robert D. (Snr). **"Scott's fingerprint mechanics."** Charles C. Thomas, 1978.

PARZEN, Emanuel. **"On estimation of a probability density function and mode."** *Annals of Mathematical Statistics,* volume 33, Institute of Mathematical Statistics, 1962, pages 1065 - 1076.

SCHENDEL, U. **"Introduction to numerical methods for parallel computers."** Translated by B.W.CONOLLY. *Ellis Horwood Ltd., Chichester.* 1984.

SINGH, Parduman. **"Pressure distortions in fingerprinting."** *Police Journal.* February 1963, pages 79 - 82.

SNYDER, Richard E. **"Automated fingerprint identification."** *Police Chief.* October 1977, pages 58 - 61.

SPARROW, Malcolm K. **"Digital Coding of Single Fingerprints: A New Approach for the Computer Age."** *Journal of Police Science and Administration.* June 1982, pages 206 - 217.

SPARROW, Malcolm K. and SPARROW Penelope J. **"Topological Coding of Single Fingerprints for Automated Comparison."** Carnahan Conference on Security Technology, University of Kentucky, Lexington, Kentucky, 1985.

SPARROW, Malcolm K. and SPARROW Penelope J. **"A Topological Approach to the Matching of Single Fingerprints: Development of Algorithms for Use on Rolled Impressions."** N.B.S. Special Publication 500 - 124, U.S. Department of Commerce, May 1985.

SPARROW, Malcolm K. and SPARROW Penelope J. **"A Topological Approach to the Matching of Single Fingerprints: Development of Algorithms for Use on Latent Fingermarks."** N.B.S. Special Publication, U.S. Department of Commerce, September 1985.

SPIVA, Wayne E. **"Microfilmed fingerprints — an automated system for latent fingerprint files."** *Police Chief.* February 1971.

STANWIX, W.D. **"Fingerprint classification and identification."** *Police Surgeon.* April 1975, pages 46 - 60.

STOCK, Robert M. **"Research and development program for continued automation of the FBI's identification division."** International Conference on Fingerprints, London, November 1984.

VANWYCKHOUSE, R.J. **"Synthetic bathymetric profiling system (Synbaps)."** Technical Report, U.S. Naval Oceanographic Office, Washington, D.C., 1973

VENKAIAH, D. **"Laws of prints and impressions."** Law Book Company, Allahabad, 1979.

WEGSTEIN, Joseph H. **"An automated fingerprint identification system."** N.B.S. Special Publication 500 - 89, U.S. Department of Commerce, February 1982.

WEGSTEIN, Joseph H. **"A semi-automated single fingerprint identification system."** N.B.S. Technical Note 481, U.S. Government Printing Office, Washington D.C., 1969.

WEGSTEIN, Joseph H. **"The M40 fingerprint matcher."** N.B.S. Technical Note 878, U.S. Government Printing Office, Washington D.C., 1978.

Appendices.

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

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

APPENDIX A.

APPENDIX B.

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
0	34	47	79	80	81	159	240
1	0	0	0	0	0	0	0
2	166	163	109	76	329	185	514
3	142	214	192	92	356	284	640
4	174	154	119	81	328	200	528
5	0	0	0	0	0	0	0
6	54	61	89	67	115	156	271
7	101	47	66	68	148	134	282
8	71	61	83	75	132	158	290
9	0	0	0	0	0	0	0
A	0	9	5	4	9	9	18
B	2	3	16	20	5	36	41
C	16	1	0	0	17	0	17
D	0	0	0	0	0	0	0
E	0	0	0	0	0	0	0
F	0	0	0	0	0	0	0
TOTAL	760	760	758	563	1520	1321	2841

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

FREQUENCY ANALYSIS FOR CODES FOUND ON THE RIGHT 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
0	9	36	89	45	45	134	179
1	0	0	0	0	0	0	0
2	66	61	57	9	127	66	193
3	72	85	89	20	157	109	266
4	67	56	57	11	123	68	191
5	0	0	0	0	0	0	0
6	133	176	103	20	309	123	432
7	266	145	100	23	411	123	534
8	134	177	104	20	311	124	435
9	0	0	0	0	0	0	0
A	2	17	10	0	19	10	29
B	1	7	20	5	8	25	33
C	10	0	0	2	10	2	12
D	0	0	0	0	0	0	0
E	0	0	0	0	0	0	0
F	0	0	0	0	0	0	0
TOTAL	760	760	629	155	1520	784	2304

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
0	1.2	4.7	14.1	29.0	3.0	17.1	7.8
1	0.0	0.0	0.0	0.0	0.0	0.0	0.0
2	8.7	8.0	9.1	5.8	8.4	8.4	8.4
3	9.5	11.2	14.1	12.9	10.3	13.9	11.5
4	8.8	7.4	9.1	7.1	8.1	8.7	8.3
5	0.0	0.0	0.0	0.0	0.0	0.0	0.0
6	17.5	23.2	16.4	12.9	20.3	15.7	18.8
7	35.0	19.1	15.9	14.8	27.0	15.7	23.2
8	17.6	23.3	16.5	12.9	20.5	15.8	18.9
9	0.0	0.0	0.0	0.0	0.0	0.0	0.0
A	0.3	2.2	1.6	0.0	1.3	1.3	1.3
B	0.1	0.9	3.2	3.2	0.5	3.2	1.4
C	1.3	0.0	0.0	1.3	0.7	0.3	0.5
D	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
F	0.0	0.0	0.0	0.0	0.0	0.0	0.0

APPENDIX C.

FREQUENCY ANALYSIS FOR CODES FOUND ON THE RIGHT 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
0	173	342	493	133	515	626	1141
1	0	0	0	0	0	0	0
2	109	74	15	3	183	18	201
3	193	140	53	3	333	56	389
4	105	100	21	5	205	26	231
5	0	0	0	0	0	0	0
6	45	36	17	2	81	19	100
7	64	10	5	1	74	6	80
8	61	44	16	8	105	24	129
9	0	0	0	0	0	0	0
A	2	4	4	0	6	4	10
B	7	4	5	0	11	5	16
C	1	6	0	0	7	0	7
D	0	0	0	0	0	0	0
E	0	0	0	0	0	0	0
F	0	0	0	0	0	0	0
TOTAL	760	760	629	155	1520	784	2304

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
0	22.8	45.0	78.4	85.8	33.9	79.8	49.5
1	0.0	0.0	0.0	0.0	0.0	0.0	0.0
2	14.3	9.7	2.4	1.9	12.0	2.3	8.7
3	25.4	18.4	8.4	1.9	21.9	7.1	16.9
4	13.8	13.2	3.3	3.2	13.5	3.3	10.0
5	0.0	0.0	0.0	0.0	0.0	0.0	0.0
6	5.9	4.7	2.7	1.3	5.3	2.4	4.3
7	8.4	1.3	0.8	0.6	4.9	0.8	3.5
8	8.0	5.8	2.5	5.2	6.9	3.1	5.6
9	0.0	0.0	0.0	0.0	0.0	0.0	0.0
A	0.3	0.5	0.6	0.0	0.4	0.5	0.4
B	0.9	0.5	0.8	0.0	0.7	0.6	0.7
C	0.1	0.8	0.0	0.0	0.5	0.0	0.3
D	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
F	0.0	0.0	0.0	0.0	0.0	0.0	0.0

APPENDIX D.

	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	14,13,9,3,2,2,2,2,2.
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.

APPENDIX F.

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

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.)

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

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

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

APPENDIX K.

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	0	0
33	6	51	96	34.17	32.59	0	0

APPENDIX L.

FORM FOR LATENT INFORMATION.

LATENT REF.NO: ..82..... PATTERN TYPE: ..... FINGER NO: .....

NO. OF EXTRACTED VECTORS: ...1.....

CENTRAL FEATURE CODE: 6.....

ANGULAR LOWER BOUND: .0°....

ANGULAR UPPER BOUND: .120°.

CENTRAL FEATURE RIDGE-COUNT LOWER BOUND: ..13....

CENTRAL FEATURE RIDGE-COUNT UPPER BOUND: ..17....

NO.OF RIDGES CROSSED BY GENERATING LINE: .21.....

NO.OF FIRST CENTRAL FEATURE RIDGE: ...12..

EVENT CODES (LEFT), 10 AT A TIME, UNIT = 0.5 cm

CODES.	B	B	B	4	2	B	3	B	B	B	B	3	B	8	6	B	B	B	7	B
DISTANCES.	6	7	8	1	1	9	7	9	10	10	10	0	10	1	1	10	11	11	2	11

CODES.	B																			
DISTANCES.	10																			

EVENT CODES (RIGHT), 10 AT A TIME, UNIT = 0.5 cm

CODES.	3	B	6	8	4	6	8	B	B	B	B	3	6	8	B	6	8	B	2	4
DISTANCES.	1	8	2	2	2	4	4	9	9	8	9	2	2	2	13	10	10	10	7	7

CODES.	7																			
DISTANCES.	1																			

APPENDIX M.

PROFORMA FOR FILE PRINT INFORMATION IN LATENT SCHEME.

CARD SET: 3333. CARD NUMBER: 82 FINGER NO: 9 . PATTERN TYPE: whorl

BOUNDARY ARRAY LENGTHS: LEFT... 24... RIGHT.. 33....

BOUNDARY ARRAY(LEFT) : (NOTE - DISTANCE UNIT IS 0.5 cms)

CODES.	2	3	8	3	C	C	4	2	2	C	8	6	6	7	8
DISTANCES.	16	16	24	7	9	8	2	2	7	8	7	7	8	3	10
CODES.	6	8	6	0	0	0	0	0	0						
DISTANCES.	10	8	8	10	7	5	2	1	0						
CODES.															
DISTANCES.															

BOUNDARY ARRAY(RIGHT) : (NOTE - DISTANCE UNIT IS 0.5 cms)

CODES.	4	3	7	4	6	6	3	6	8	4	7	4	3	2	6
DISTANCES.	18	22	8	34	9	6	1	1	1	24	6	20	5	20	8
CODES.	3	8	8	6	8	6	8	6	8	0	0	7	3	0	0
DISTANCES.	2	8	14	7	7	10	10	4	4	11	9	5	4	7	8
CODES.	0	0	0												
DISTANCES.	4	3	2												

DISTANCE CONVERSION MEASURES: (NOTE - DISTANCE UNIT IS 0.5 cms)

DEGREES FROM LEFT BOUNDARY:	0	60	120	180
DISTANCES MEASURED:	21.4	19.8	21.2	27.6
RIDGE COUNT COVERED:	24	20	24	33

EVENT CODES OVERLEAF.

EVENT CODES. (TOPOLOGICAL COORDINATES.)

NO.	CODE.	THETA.	RC.	NO.	CODE.	THETA.	RC.	NO.	CODE.	THETA.	RC.
1	2	1	15	26	B	83	12	51	1	134	28
2	1	2	23	27	6	84	8	52	7	136	22
3	1	5	24	28	E	87	13	53	1	138	30
4	1	8	25	29	E	90	13	54	7	140	30
5	7	11	14	30	E	90	14	55	1	140	32
6	3	18	24	31	E	92	14	56	3	143	28
7	7	19	25	32	7	95	14	57	1	144	32
8	6	22	10	33	E	97	14	58	1	146	33
9	6	40	27	34	2	98	5	59	1	147	34
10	0	44	28	35	6	103	6	60	1	147	34
11	0	47	27	36	E	104	12	61	1	149	36
12	0	48	26	37	1	106	20	62	1	150	37
13	0	49	25	38	1	109	21	63	6	150	34
14	0	52	24	39	2	113	7	64	1	151	39
15	B	53	16	40	1	114	21	65	3	152	32
16	0	55	22	41	1	117	22	66	6	154	38
17	0	56	21	42	1	119	23	67	2	156	35
18	B	59	16	43	1	124	24	68	2	163	37
19	0	62	19	44	2	125	15	69	0	165	37
20	0	64	18	45	7	129	10	70	7	169	33
21	0	67	16	46	1	129	25	71	0	170	37
22	0	68	16	47	1	130	26	72	3	171	36
23	B	73	13	48	1	133	27	73	3	172	31
24	B	79	13	49	3	134	21	74	0	176	34
25	B	80	12	50	7	134	7	75	0	178	33
								76	0	179	32
								77	6	180	14

APPENDIX N.

Table of results of tests performed using LM3.

No.	Parameters							Performance		
	BOUND	CMS	HOPS	MAXSHIFT	ADT	DDT	SDT	MR1	MR3	MR10
1	15	1	1	1	5	3	3	44.64%	62.50%	82.14%
2	5	1	1	1	4	2	2	46.43%	60.71%	82.14%
3	5	2	1	1	4	2	2	44.64%	64.29%	82.14%
4	5	3	1	1	4	2	2	46.43%	66.07%	83.93%
5	5	-1	1	1	4	2	2	42.86%	60.71%	75.00%
6	5	1	1	1	7	5	5	46.43%	53.57%	78.57%
7	5	1	1	1	10	5	5	42.86%	51.79%	75.00%
8	5	1	0	0	4	2	2	50.00%	69.64%	78.57%
9	5	1	2	2	4	2	2	44.64%	55.36%	83.93%
10	5	-1	0	0	2	2	2	42.86%	60.71%	71.43%
11	5	-1	0	0	2	1	1	44.64%	64.29%	73.21%
12	5	5	1	1	4	2	2	46.43%	60.71%	85.71%
13	5	5	1	1	7	5	5	42.86%	51.79%	80.36%
14	10	1	1	1	4	2	2	46.43%	58.93%	82.14%
15	15	0	1	1	4	2	2	41.07%	60.71%	80.36%
16	5	0	1	1	4	2	2	41.07%	60.71%	80.36%
17	5	0	0	0	4	2	2	42.86%	64.29%	76.79%
18	5	0	0	0	5	5	5	44.64%	58.93%	76.79%
19	5	1	2	2	4	2	2	44.64%	55.36%	83.93%
20	5	3	0	0	4	2	2	53.57%	67.86%	82.14%
21	5	4	0	0	4	2	2	53.57%	66.07%	80.36%
22	5	5	0	0	4	2	2	48.21%	67.86%	82.14%
23	5	2	0	0	4	2	2	53.57%	66.07%	78.57%
24	5	3	0	0	2	2	2	50.00%	67.86%	80.36%
25	5	3	0	0	3	2	2	53.57%	73.21%	82.14%
26	5	3	0	0	6	2	2	50.00%	64.29%	78.57%
27	5	3	0	0	10	2	2	42.86%	53.57%	76.79%
28	5	3	0	0	99	2	2	33.93%	53.57%	80.36%
29	5	3	0	0	3	1	1	51.79%	73.21%	80.36%
30	5	3	0	0	3	3	3	55.36%	73.21%	83.93%

Appendix N continued.

No.	Parameters							Performance		
	BOUND	CMS	HOPS	MAXSHIFT	ADT	DDT	SDT	MR1	MR3	MR10
31	5	3	0	0	3	0	0	50.00%	66.07%	83.93%
32	5	3	0	0	3	2	1	53.57%	73.21%	82.14%
33	5	3	0	0	3	1	2	51.79%	73.21%	80.36%
34	5	3	0	0	3	4	4	57.14%	71.43%	82.14%
35	5	3	0	0	3	3	2	55.36%	73.21%	83.93%
36	5	3	0	0	3	2	3	53.57%	73.21%	83.93%
37	5	3	0	0	3	1	0	48.21%	67.86%	82.14%
38	5	3	0	0	3	0	1	53.57%	71.43%	82.14%
39	5	3	0	0	3	2	0	50.00%	67.86%	82.14%
40	5	3	0	0	3	0	2	57.14%	71.43%	82.14%
41	5	3	0	0	3	5	5	57.14%	69.64%	82.14%
42	5	3	0	0	3	6	6	53.57%	69.64%	82.14%
43	5	3	0	0	3	7	7	53.57%	69.64%	82.14%
44	5	3	0	0	3	2	4	53.57%	73.21%	80.36%
45	5	3	0	0	3	3	5	57.14%	71.43%	82.14%
46	5	3	0	0	3	2	6	53.57%	73.21%	80.36%
47	5	3	0	0	3	3	6	57.14%	71.43%	82.14%
48	5	3	0	0	3	0	4	58.93%	71.43%	78.57%
49	5	3	0	0	3	1	4	51.79%	73.21%	78.57%
50	5	3	0	0	3	4	2	53.57%	71.43%	83.93%

APPENDIX O.

Table of results of tests performed using LM4.

In tests 1-24 the following parameters were fixed: BOUND=5, MAXSHIFT=0.

The following parameters were fixed for the non-boundary vectors only: CMS=3, HOPS=0, ADT=3, DDT=3, SDT=5.

Tests 1-23 were performed only on the subset of 25 latents that included at least one boundary vector. Tests 24, 25, 30-42 were performed on the whole latent set. Tests 26-29 used the subset of latents that contained no boundary vectors.

Tests 1-23 used the original 59 file prints and tests 24-42 used the expanded set of 100 file prints.

No.	Parameters					Performance		
	CMS	HOPS	ADT	DDT	SDT	MR1	MR3	MR10
1	1	1	6	4	8	44.00%	72.00%	84.00%
2	3	1	6	4	8	52.00%	72.00%	88.00%
3	3	0	6	4	8	56.00%	68.00%	80.00%
4	2	0	6	4	8	56.00%	68.00%	80.00%
5	1	0	6	4	8	52.00%	68.00%	76.00%
6	0	0	6	4	8	48.00%	60.00%	80.00%
7	-1	0	6	4	8	52.00%	60.00%	76.00%
8	1	0	4	4	8	52.00%	60.00%	68.00%
9	1	0	8	4	8	48.00%	64.00%	80.00%
10	1	0	10	4	8	48.00%	60.00%	80.00%
11	1	0	6	3	5	52.00%	68.00%	76.00%
12	1	0	3	3	5	52.00%	60.00%	84.00%
13	3	1	8	4	8	44.00%	68.00%	80.00%
14	3	1	6	3	5	52.00%	68.00%	88.00%
15	3	1	3	3	5	60.00%	68.00%	80.00%
16	3	1	6	4	6	52.00%	72.00%	88.00%
17	3	1	4	4	4	52.00%	68.00%	84.00%
18	3	1	5	3	5	48.00%	64.00%	80.00%
19	3	1	3	3	3	56.00%	68.00%	80.00%
20	3	1	2	2	2	52.00%	68.00%	84.00%
21	3	1	2	2	4	56.00%	68.00%	84.00%
22	3	1	2	3	4	52.00%	68.00%	84.00%
23	3	1	3	2	3	56.00%	68.00%	80.00%
24	3	1	3	3	5	48.21%	67.86%	80.36%

Appendix O continued.

In tests 25-42 the following parameter was fixed: BOUND=5.

The following parameters were fixed for the boundary vectors only: CMS=3, HOPS=1, ADT=3, DDT=3, SDT=5.

Tests 25, 30-42 were on the complete set of 56 latents and the 100 file prints. Tests 26-29 were on the subset of latents that contained no boundary vectors and the 100 file prints.

No.	Parameters						Performance		
	CMS	HOPS	MAXSHIFT	ADT	DDT	SDT	MR1	MR3	MR10
25	3	0	0	2	1	4	44.64%	71.43%	80.36%
26	3	0	0	3	3	5	50.00%	76.67%	83.33%
27	1	0	0	3	1	2	54.84%	74.19%	80.65%
28	3	0	0	2	1	2	54.84%	77.42%	80.65%
29	0	0	0	2	1	2	38.71%	54.84%	67.74%
30	3	0	0	2	1	2	51.79%	71.43%	80.36%
31	2	0	0	2	1	2	55.36%	71.43%	80.36%
32	4	0	0	2	1	2	51.79%	73.21%	82.14%
33	3	0	0	2	2	2	50.00%	71.43%	82.14%
34	3	0	0	1	1	1	48.21%	62.50%	82.14%
35	3	0	0	2	1	1	51.79%	73.21%	80.36%
36	3	0	0	3	1	3	51.79%	67.86%	80.36%
37	3	0	0	2	1	3	51.79%	69.64%	80.36%
38	3	1	1	2	1	2	58.93%	67.86%	83.93%
39	3	1	1	4	2	2	53.57%	66.07%	80.36%
40	3	1	1	2	1	4	51.79%	69.64%	80.36%
41	3	1	1	2	1	2	53.57%	67.86%	83.95%
42	2	1	1	2	1	1	58.93%	67.86%	85.71%

APPENDIX P.

Table of results of tests performed using LM5.

The following parameters were fixed in these tests: BOUND=5, HOPS=0, MAXSHIFT=0, MDT=1, PDT=10, DEPTH=5.

No.	Parameters				Performance		
	CMS	MAT	CUTOFF	SPAN	MR1	MR3	MR10
1	3	20	20	30	71.43%	78.57%	83.93%
2	3	20	5	30	75.00%	76.79%	80.36%
3	3	20	15	30	80.36%	82.14%	85.71%
4	3	20	13	10	78.57%	80.36%	85.71%
5	1	90	15	10	69.64%	80.36%	82.14%
6	3	20	15	10	80.36%	82.14%	85.71%