University of Wollongong
Research Online

Department of Computing Science Working Paper Series

Faculty of Engineering and Information Sciences

1984

On-line recognition of handprinted characters

John Collins
University of Wollongong

Recommended Citation

Collins, John, On-line recognition of handprinted characters, Department of Computing Science, University of Wollongong, Working Paper 84-11, 1984, 22p.
http://ro.uow.edu.au/compsciwp/83

Research Online is the open access institutional repository for the University of Wollongong. For further information contact the UOW Library: research-pubs@uow.edu.au

THE UNIVERSITY OF WOLLONGONG

DEPARTMENT OF COMPUTING SCIENCE

ON-LINE RECOGNITION OF HANDPRINTED CHARACTERS

John Collins

Department of Computing Science
University of Wollongong

Abstract

The project implemented an on-line handprinted character recogniser. A Bayesian decision rule was used in conjunction with selected feature transformations. Fourier transformations and transformations derived from discriminant analysis were compared. Discriminant features generally proved superior to those derived from Fourier analysis. A recognition rate of 98% was achieved with the use of the first 4 discriminant functions. The character set included numerals and arithmetic operators. The results from the current system provide a basis for constructing a practical device for data entry.

Preprint No 84-11 **July 30, 1984**

P.O. Box 1144, WOLLONGONG, N.S.W. AUSTRALIA
telephone (042)-270-859
telex AA29022

ON-LINE RECOGNITION OF HANDPRINTED CHARACTERS

John Collins

University of Wollongong

ABSTRACT

The project implemented an on-line handprinted character recogniser. A Bayesian decision rule was used in conjuction with selected feature transformations. Fourier transformations and transformations derived from discriminant analysis were compared. Discriminant features generally proved superior to those derived from Fourier analysis. A recognition rate of 98 % was achieved with the use of the first 4 discriminant functions. The character set included numerals and arithmetic operators. The results from the current system provide a basis for constructing a practical device for data entry.

TABLE OF CONTENTS

LIST OF FIGURES

1. INTRODUCTION

The aim of this project is to construct an on-line handprinted character recogniser. Handprinted characters differ from cursively written characters in that blank spaces separate successive characters. There is , then, no problem of determining when one character begins and another ends. An on-line system processes the character information while it is drawn, and the data source is something like a digitiser or data tablet.

The motivations for this project are varied. A handprinted character recogniser would be useful for some situations in which it is inconvenient to enter data by typing on a keyboard. Certainly writing is a more natural form of communication for some people. One major application of written character recognisers would be for the entry of Chinese or Japanese characters for processing. The operation of typewriters for these languages is a fairly skilled activity, and a handwritten character recogniser would greatly facilitate character entry. Another possible application is the use of characters recognisers as an aid for teaching children to write. Finally, work on handprinted character recognition serves as a test bed to evaluate procedures applicable in other problem areas of pattern recognition.

There is a wealth of available literature on character recognisers. Suen et al. (1980) list a comprehensive bibliography on the topic. The methods used range from cluster analysis, decision tree classifiers, Bayesian classification, to dictionary lookup methods. It is difficult to evaluate the relative merits of these methods because each study used its own set of constraints on the characters used for test purposes.

One criterion for the selection of a technique is that it be backed up by some well deveoped theoretical model which under certain assumptions defines an optimal procedure. The Bayesian classification method fits this requirement via a rigorous base in probability theory. Probability theory provides an established set of methods for decision problems where the available data contains errors in measurement, and redundancies between measurements. The Bayesian method also allows the option of making use of information provided by context and the like, if such information is expressable as a probability. Transitional probabilities are one source of usable information. For example, if the recogniser is processing a character in English text, and the previous letter was a 'q' then the outcome decision should generally be biased in favour of getting 'u' as a result.

The basic idea of the Bayesian method is to use the available information to estimate the probability of each member in the set of recognisable characters. These probabilities take the form of :

$$P\ (\text{Class} = c \mid \sim)$$

where :

- c is a particular character class in the set of recognisable characters. For example, it might represent the character class 'a' with the other recognisable characters being 'b', 'c', .. , 'z'.
- ~ is the information provided by the input pattern's features.

This probability is a conditional probability, and expresses the likelihood of the actual class being class 'c', when the input pattern looks the way it does. The decision rule is to pick that character class which is the most probable, i.e. that class for which P (Class = c | ~) is a maximum. These conditional probabilities are not directly available, but we can derive them by use of Bayes theorm :

$$P\ (Class = c \mid \sim) = \frac{P\ (Class = c)\ \ P(\sim \mid Class = c)}{\sum_{x} P\ (Class = x)\ \ P\ (\sim \mid Class = x)}$$

where :

c,x — are character classes within the set of recognisable characters.

P(Class = c) — is the a priori probability of class 'c'. These "prior" probabilities would be determined by the differing frequencies in usage of different characters, or by transitional probabilities.

P(~ | Class = c) — is the probability or likelihood of the features '~' under the assumption that they belong to class 'c'.

Probabilities of the form P (~ | Class = c) can be estimated if we assume some kind of probability distribution for feature data belonging to a particular class. The assumption for this project was that the features within any given category had a multivariate normal distribution. The likelihood function for this distribution takes the form of:

$$p(\sim \mid Class=c) = \frac{1}{(2\pi)^{n/2}\left|\Sigma_c\right|^{1/2}} \cdot \exp\left[-\frac{1}{2}\left(\mathbf{F}-\mathbf{M}_c\right)^T \cdot \Sigma_c^{-1} \cdot \left(\mathbf{F}-\mathbf{M}_c\right)\right]$$

$\mathbf{F}$ — vector of observed features.
$\mathbf{M}_c$ — vector of feature means for class 'c'.
Σ_c — covariance matrix for class 'c'.
n is the number of observed features.

The parameters for this density function, the means and covariances are estimated from a number of representative examples. This set of pattern examples covering all potentially recognisable characters is referred to as the 'training set', as it is from this set that the recognisable patterns are defined to the system.

2. IMPLEMENTATION

The recognition process can be broken down into 3 phases : preprocessing, where relevant features are extracted from the source data ; tranformation, where the dimensional complexity of the features is reduced ; and classification using the Bayesian decison rule. An addition step is required to generate the class means and covariances which define the reference patterns. These phases are described below.

2.1. Preprocessing

Source data from the digitiser (see Appendix 1 for details) consisted of a stream of co-ordinate positions corresponding to the various positions of the stylus on the pad. The intention throughout preprocessing was to extract features which preserved information about the basic shape of the pattern, and information about the order in which successive points were drawn. Suen et al. (1980) noted that that the sequence in which points are traversed serves as a rich source of information, over and above that provided by the shape of the character.

The input pattern was parsed into its component strokes, where a stroke is defined as the shape drawn in between putting the stylus down to write, and lifting it up off the pad. Some characters such as an 'S' can be drawn in a single stroke, and the letter
'T' is usually drawn in two strokes. Other characters such as an 'F' are typically drawn using 3 strokes.

The features extracted from the raw co-ordinate data were of 3 types :

[1] features describing the within-stroke characteristics of the pattern

Each stroke was segmented into 8 sections of equal length. The co-ordinates corresponding to the endpoints of these segments were normalised for position by subtracting from them the starting position of the stroke. They were further normalised for size, by dividing each of these offsets by the length of that stroke.

[2] features describing the relations between the component strokes

The relative positions between the strokes were expressed by the offset position of each stroke from the starting position of the first stroke. Each offset was normalised by dividing it by the pattern length, summed over all strokes. Additionally the relative sizes of the component strokes were derived, where the relative size of a stroke was the length of that stroke divided by the pattern size.

[3] global features were derived for the pattern

The pattern size was calculated by summing up the lengths of all the component strokes. The number of strokes in a character was also used as a global feature.

The resultant preprocessed character took the form of the data structure:

```
character = record
   number_of_strokes     : integer ;
   pattern_size          : real ;
   component_strokes     : array [1..number_of_strokes] of
      stroke = record
         start_point     : point ;
         relative_size   : real ;
         offsets         : array [1..MAX_N_POINTS] of real ;
   end {stroke}
 end {character}


point = record
   x              : real ;
   y              : real ;
end {point}
```

Some of these features are entirely redundant. The relative size features for all strokes adds up to the value 1, so one of the stroke sizes is linearly dependent upon the rest. Also the starting offset of the first stroke from the start of the first stroke is obviously always zero, so this value too is redundant. After the redundant variables are removed, the number of features is :

$$2*number\_of\_strokes*8 + 3*(number\_of\_strokes - 1) + 1$$

It can be noted that the number of preprocessed features grows steadily with the number of strokes in the pattern. The practical limit on the number of strokes in this system was two. This was because the matrices used to manipulate these features in later stages of recognition grew prohibitively large with a greater number of strokes.

2.2. Transformations

The performance of the current recognition system was strongly dependent on the number of features being processed. The calculation of the likelihoods in the classification stage involved pre- and post- multiplying a matrix by a vector. This operation required roughly O (N^2) multiplications. It was therefore desirable to obtain some form of reduction in the dimensionality of the preprocessed features, especially since this dimension was as high as 36 features.

Two methods of data reduction using linear transformations were applied : Fourier transforms and Discriminant Function transforms.

2.2.1. Fourier Transforms

The effect of applying Fourier tranformations is to derive the set of frequency components from a set of points which defines a connected line. Used in this way, data reduction can be effected by selecting out some small number of the lower frequency components derived from the line. The underlying assumption is that the higher frequency components represent noise in the

pattern and can be ignored. Hopefully the subset of lower-frequency components will serve to distinguish between the character classes almost as well as the entire set of components.

Fourier transforms were applied to the within-stroke offsets extracted by the preprocessor. To begin with, the x- values, and the y- values from these offsets were plotted as functions of stroke length. This roughly corresponds to plotting the x- offsets and the y- offsets against a dimension of time. The advantage of this replotting procedure is that both the x- and y- values are now single-valued functions of stroke length. In the original co-ordinate form, y- values didn't always define a single-valued function of x- values, and nor vice versa. Fourier transforms were then applied to the x- plot and y- plot in turn.

Fourier transforms were not applied to the features which did not define a connected line, such as the relative sizes of the strokes. There is no basis for supposing that the higher frequency components of a series of such points are less important than the lower frequency components. Therefore Fourier transforms are useless as a data reduction method for this set of points. The coefficients resulting from transformation were defined as :

$$a_f = \sum_{i=1} \Delta z_i \cos\left(f \cdot \pi\right)$$

where :

f : is a sampled integer frequency.

a_f : is the Fourier coefficient for frequency "f".

Δz_i : are the within-stroke x- or y- offset values after preprocessing.

These transforms were applied to the offsets within each of the strokes in the character pattern. The number of features after tranformation can be specified by the expresion :

$$s * f * 2 + (s - 1) * 3 + 1$$

where :

s : is the number of strokes in the pattern.
f : is the number of frequency components selected.

The quantity 's * f * 2' is the total number of Fourier coefficients derived from transformation of x - and y - values in each of the 's' strokes. The quantity '(s - 1) * 3 + 1' reflects the number of features which haven't been affected by transformation.

2.2.2. Discriminant Analysis Transforms

The rationale behind the use of discriminant analysis is to derive a set of transforms which maximally discriminates between the character classes. If the measure of 'discrimination' of a tranformation is the F-statistic for differences between character categories, then the transformations we seek will be defined as a subset of the eigenvectors of the matrix product (Harris, 1976):

$$SSW^{-1}SSB$$

where :

SSW : is the within-group covariance matrix.
SSB : is the between-group covariance matrix.

both *SSW* and *SSB* are estimated from the patterns in the training set.

The 'discrimination' of transformations so defined is given by the size of the eigenvalues corresponding to the eigenvectors. The eigenvectors can then be ranked in terms of their power

to differentiate categories.

A subset of tranformations based upon the eigenvectors associated with the m largest eigenvalues will differentiate between character classes more than any other set of linear transforms of size m. A neccessary qualification to this statement is that the SSW and SSB matrices must be accurate estimates of the corresponding population covariances. Further, each preprocessed feature must be normally distributed. This is a reasonable assumption in all cases except for the feature representing the number of strokes in the pattern which only took on the values of 1 or 2 in this study. This feature was treated differently to the other features in the classification process.

The matrix defined by the product $SSW^{-1}SSB$ is not generally symmetric, even though SSW and SSB individually are. The algorithm used to calculate the eigenvectors was based upon the solution method to the General Symmetric Eigenvalue problem, outlined in Gourlay and Watson (1973). This algorithm takes advantage of the fact that SSW and SSB are symmetric.

2.3. Derivation of Reference Patterns

The parameters to the multivariate normal density function constitute the reference pattern for each character class. Intuitively, the vector of means extracted from a set of patterns within a class represents a "template" for that class. In this vein, the covariance matrix specifies the degree of "slop" allowable between the template pattern and the input pattern.

The steps to generate a reference pattern involve :

[1] the input of a number of representative patterns into the training set.

[2] processing of the training set, transforming these values and deriving the means and covariances for the given character class.

[3] the inversion of the covariance matrix. This process was carried out using the Cholesky method of triangular decomposition (see Schwartz, 1973).

[4] storage of the means and the inverted covariance matrix on file.

Each reference pattern was specifically derived using a particular set of transforms of the training set data. Whenever a different set of transformations is evaluated, a new reference file has to be generated to take this into account. To ensure that the same transformations are applied to the patterns undergoing recognition, as were applied in the formation of the reference file, a copy of the linear weights of these tranformations was stored on the reference file. These weights specified the transformations taking place in the recognition phase.

The original design of the recognition system required that the reference patterns be large enough to cater for the possibility of using the entire set of preprocessed features, without transformations. Since there could be as many as 36 of these features, each reference pattern required about 5 K bytes of storage.

As it turned out, it proved impossible to invert all the covariance matrices based upon the non-transformed variables. The large size of the matrix and the high degree of correlation between the preprocessed features effectively made the matrices singular, for many of the character classes processed. The result was that all reference patterns generated were based on a small sub-set of the possible number of transformations, and little of the reference pattern record was actually used.

2.4. Classification

Features resulting from preprocessing and transformations were used to estimate the likelihood of the character classes on the reference pattern file. As outlined in the introduction, this involved calculating the class-conditional likelihood of the input pattern from the multivariate normal density functions.

The feature representing the number of strokes in the input pattern was assumed to be input without error. Only reference patterns having the same number of strokes as the input pattern were selected for calculation of the likelihoods. Reference patterns with a different number of strokes were assumed to have zero probability, and were effectively removed as candidates for

selection.

2.5. The Overall System

The overall recognition system took the form of a set of programs and files whose interrelationships are expressed in Figure 1.

The major system components are :

[1] the 'TRAINER' program, which enables entry of training patterns onto the training file. This program preprocesses the raw input pattern and allows checking of pattern details on the training set file.

[2] the transformation generators : 'FOURIER' and 'DISCRIM' generate weights to be applied. These weights are stored on transformation files.

[3] the reference pattern generator : 'GENREF' processes the training set file in conjuction with the tranform weights on the transformation files. The output of this program is the reference file.

[4] the recognition programs : 'PROB' and 'PROBTAB' derive probabilities and likelihoods. PROB is used for on-line recognition to demonstrate the use of the recogiser. This program preprocesses the digitiser input, applies transformations, and then outputs the probability estimates for each class. PROBTAB is used in conjunction with a test set of pattern, which have been stored in the same format of file as the training patterns. This program is used to evaluate the performance of the system, and prints out a table indicating which characters were confused in the recognition process with what others.

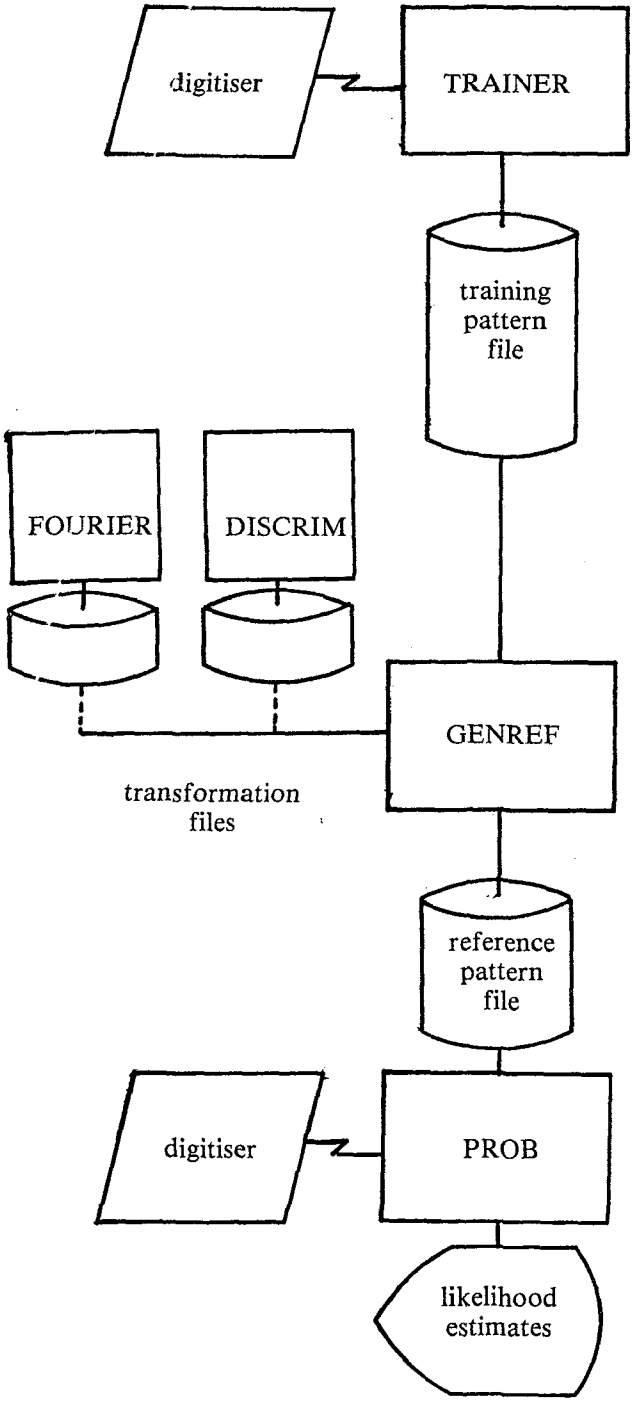


Figure 1. The recognition system.

3. EXPERIMENT

3.1. Description

The aim of the experiment was to evaluate the performance of the recognition system. The testing of the program was only preliminary because of the time constraints on the project.

In testing a recognition system the set of recognisable characters has to be specified. The alphabetic characters are an obvious choice for this test set, since they constitute the most common kind of character in text processing. Unfortunately the reference file required to store all the reference patterns for this set was too large to fit onto a floppy diskette. An alternative set, consisting of the 10 numerals and the operators '+', '-', '*', '/', and '=' was used. Such a set of characters could potentially be used on a calculator-like device with handwritten character input. In any case, this set should be capable of testing out some of the fundamental characteristics of the system.

Training patterns were entered for the above character classes. Each class had only one basic format of pattern entered into the training set. For example, the only format used for the multiplication operator was an 'x' type symbol, with both strokes being drawn with downward strokes. There were 20 sample patterns written in the same format for each character category on the traininig set.

All characters input were written to the same scale. The purpose of this was to make use of the fact that characters drawn to the same scale have differing stroke lengths. For example, the patterns for '1' and '8' when drawn to the same scale have stroke lengths, differing by a factor of about 2.5 . This is perhaps a useful feature for distingishing between categories.

A test set of patterns was used to aid in the evaluation of the system. Ten character patterns for each class were input according to the same formats and scaling as used for entry of the training patterns. These formats and relative sizes are given in Appendix 2. The training set and the test set were drawn by the same person.

A variety of Fourier and discriminant analysis transformations were applied in the recognition process, and assumed that all character classes had the same a priori probability in the Bayesian decision process.

3.2. Results

The performance of the system in recognising members of the test set was evaluated and the results are displayed in figures 2 and 3. Figure 2 shows the percentage of correct recognitions for 1-stroke chracters. The four points along the curve for Fourier transformed features correspond to the use of 1, 2, 3, and 4 frequency coefficients. Points along the graph for discriminant features correspond with the use of 1 to 4 discriminant transforms. The number of correct recognitions increases for both Fourier and discriminant transformed features as the number of transforms increases, with the respective final rates being 96 % and 98 %. It is apparent from this figure that in addition to the final recognition rate being higher in the case of discriminant features, this technique requires fewer features after transformation than do Fourier transforms.

Figure 3 shows recognition rates achieved for 2 stroke characters. The rate of successful recognition for Fourier features hovers around 98 %, with a slight drop as the number of features is increased. This decrease is an unexpected result, but it is probably due to the small sample sizes used in the training and test pattern sets. For example a small training set size could have resulted in erroneous estimation of reference pattern means for the higher frequency coefficients. Making use of these coefficients would mean less accuracy in recognition rather than more. On the other hand the recognition rate for discriminant features climbs up to 98 % when all 4 discriminant coefficients are used.

Overall then, it would appear that only about 3 or 4 discriminant features are required to achieve a reasonable level of accuracy in recognition. Additionally the results indicate that the level of accuracy obtained by discriminant features was as good as, if not better than that achieved with Fourier features, even though they involved a smaller number of features. This is decidedly an advantage if we remember that the number of multiplications required to reach a decision in the classification stage is O (N^2), where N is the number of features after transformation.

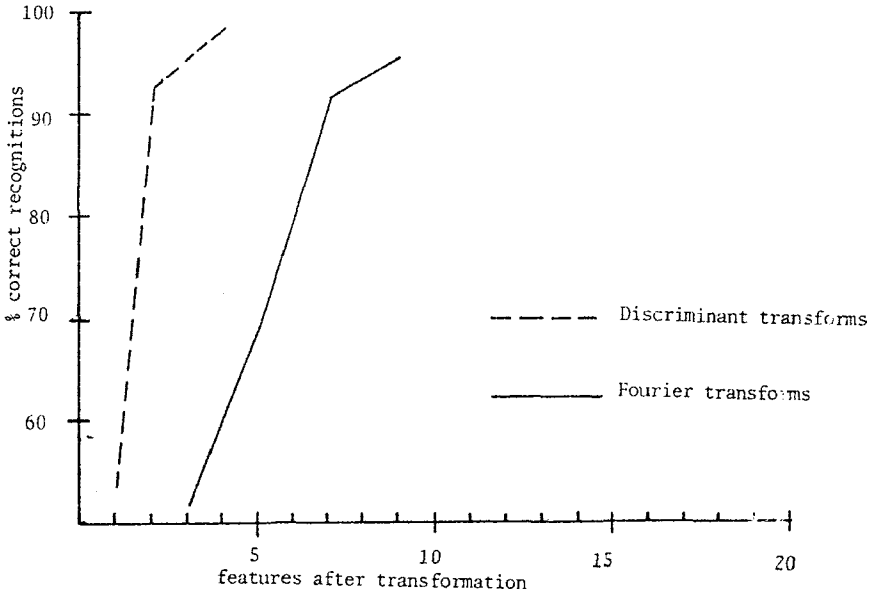


Figure 2. Recognition of 1-stroke characters.

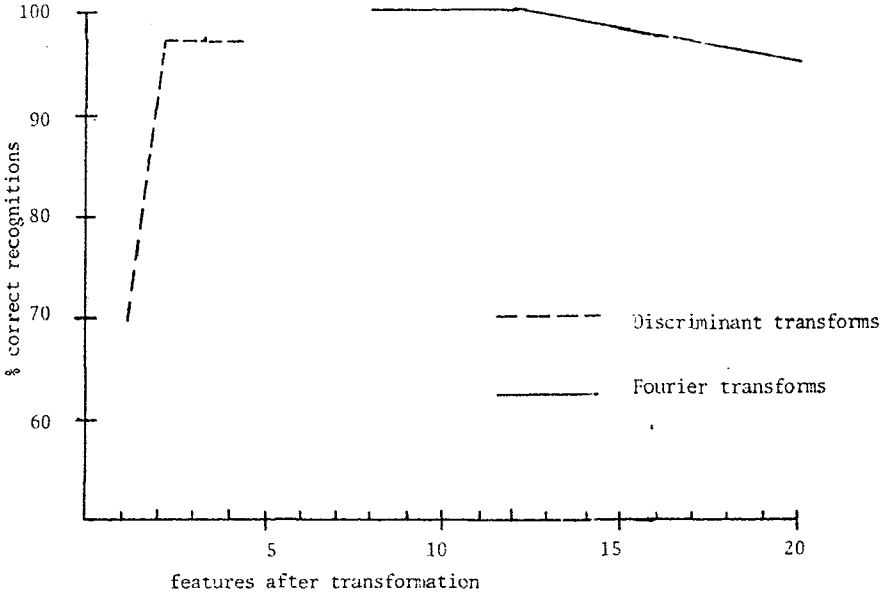


Figure 3. Recognition of 2-stroke characters.

4. SUGGESTED ENHANCEMENTS

The current system is only an exploration of on-line character recognition techniques. In its present form, it is by no means a practical method of character recognition. Particular limitations and suggested solutions are outlined below :

[1] When a character pattern was entered on the digitiser, an explicit signal had to be given to initiate preprocessing and subsequent recognition processing of the pattern. This signal took the rather awkward form of pressing a key on the host computer's keyboard and touching the tip of the digitising pen down again onto the digitising pad. This procedure was necessary to interupt the read routine which was written in Pascal. For practical purposes, the routine which reads from the digitiser must be able to infer when a character has been completely entered and to start processing without prompting. An input routine which starts preprocessing after the pen has been lifted off the pad for more than a threshold period of time would satisfy this requirement.

[2] The limit on the number of strokes for input characters was 2. The reason for this limitation was outlined in the section on patern preprocessing. The solution to this problem lies in finding a method of preprocessing which generates the same number of features regardless of the number of component strokes in a character. Arakawa (1983) implemented a preprocessing system which did not distinguish between the seperate strokes in a character, but instead interpolated straight lines between them. All patterns could then be treated for most purposes as if they were single stroke characters.

[3] The recognition process was dependent upon the absolute size of the pattern. There are two possible solutions to this limitation. The first solution involves leaving the feature corresponding to absolute character size out of the recognition process. Whether or not this leads to a significant reduction in performance has to be evaluated. The alternative is to allow the user to specify his own scaling. A suitable conversion ratio could then be worked out to re-scale the input patterns to the same size as those used in the reference file.

[4] The system was designed to allow only one format of character pattern for each pattern class. If a character is entered according to any different format, then the chances are that it will be mis-recognised. This is a notable limitation given that different writers draw their characters in differing formats. The only solution is to store several different reference patterns for a single character class, corresponding to the different drawing formats for that class. The decision technique would be to pick that character class which was most likely, as opposed to picking the most likely character format. The probability estimate for a character class which had several formats would simply be the sum of the estimates for the sub-patterns with those formats.

[5] The recognition system as it now stands is slow and costly on disk storage. It typically takes about 30 seconds to recognise a character, and the reference file for only 15 reference patterns takes up most of a floppy diskette. These problems are largely attributable to the large size of the reference patterns. It was not known initially just how many transformed features would be required to obtain adequate recognition performance, so the reference pattern records were made large enough to suit the largest possible feature vectors.

However, the current results indicate that given suitable transformations reference patterns need only be as large as those required to store the means and covariances of 3 or 4 transformed features. The consequence of this result is that all reference patterns could be stored simultaneously in memory. Thus there would be no need for repetitive disk accesses to obtain reference patterns, as there is in the current system. There is then the potential for substantial increases in processing speed and decreases in disk storage requirements.

5. CONCLUSIONS

The project demonstrated the use of an on-line character recogniser based on Bayesian classification techniques. The system was found to have a reasonably good recognition performance when it was tested with patterns input by the same person who authored the training set. Further tests are required to evaluate the performance in cases where the training set and test pattern set are penned by different people. It does appear, though, that the performance achieved in this project is roughly comparable to that obtained in similar systems (Arakawa, 1983), and by the use of other techniques (see Suen et al., 1980). An exact comparison with other results is not possible, because of the different types of patterns and constraints applying to the other studies.

The project demonstrated the usefulness of discriminant analysis as a tool for data reduction. It would appear that this technique is superior to Fourier transformations in its ability to reduce the number of features while retaining information about differences in categories. This finding is all the more important when one realises that Fourier transformation methods are more widely used.

In spite of the severe limitations of the current system, the current project demonstrated some characteristics for a viable character recogniser.

REFERENCES

[1] Arakawa H. On-line Recognition of Handwritten Characters - Alphanumerics, Hiragana, Katakana, Kanji. Pattern Recognition, 1983, 16, 9 - 20.

[2] Gourlay A., Watson G. Computational Methods for Matrix Eigenproblems. John Wiley and Sons, London, 1973.

[3] Harris R.J. A Primer of Multivariate Statistics, Academic Press, New York, 1975

[4] Schwartz, H. Numerical Analysis of Symmetric Matrices. Prentice-Hall, New Jersey, 1973.

[5] Suen C., Berthod M., Mori S. Automatic Recogniton of Handwritten Characters - the state of the art. Proceedings of the IEEE, 1980,68,469-487.

APPENDIX 1 - Hardware and Software Requirements

Hardware Requirements

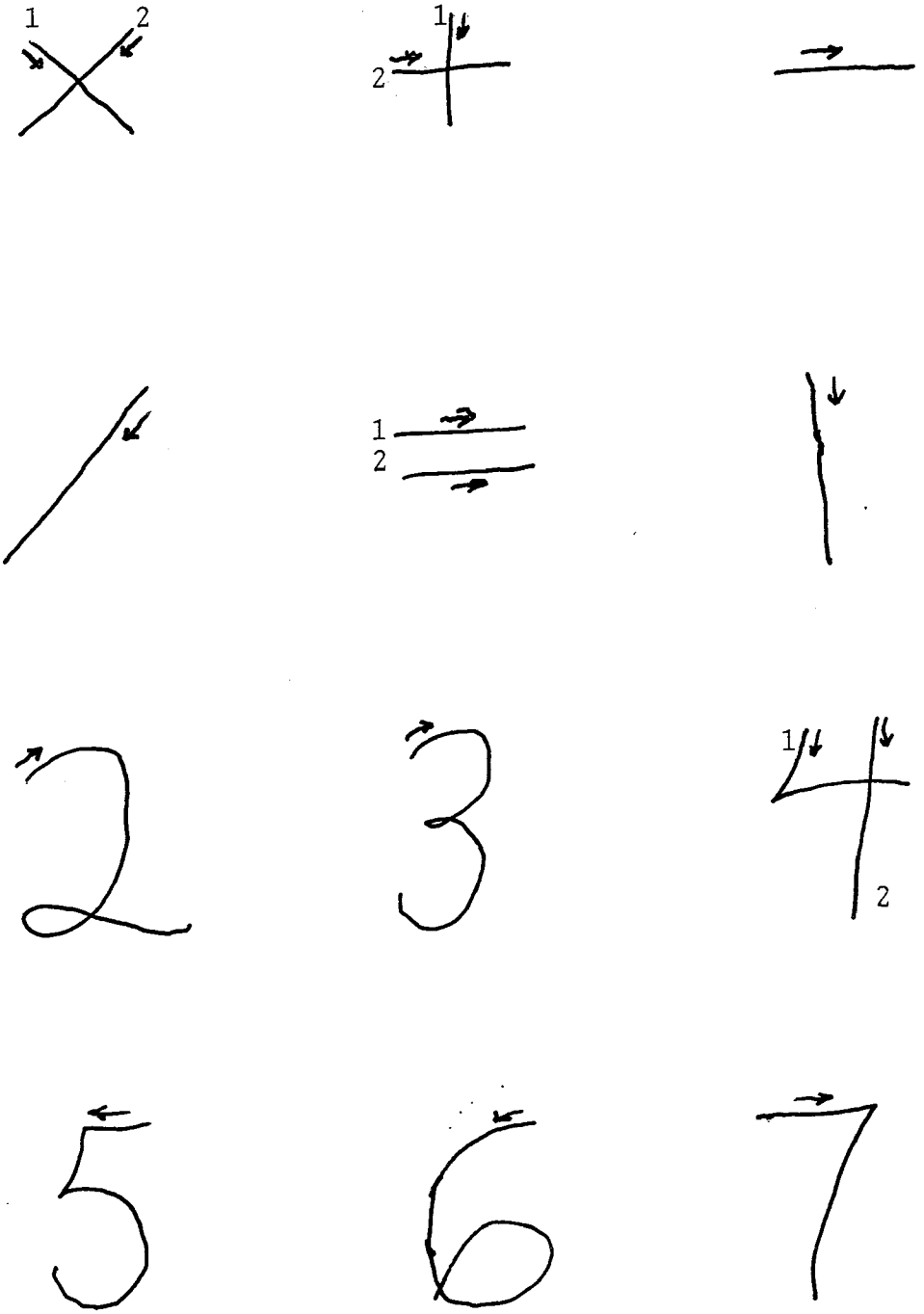
microcomputer:	an Apple IIe with 64 K bytes of RAM.
interface-card:	an RS 232 serial interface card is required in I/O slot #2 on the microcomputer.
digitiser:	HI-PAD Digitising Pad with a resolution of .015 inches. Output format was in serial ascii bcd with a data rate of 300 baud.

Software Requirements

operating-system:	Apple UCSD Pascal operating system.

APPENDIX 2 - Character Formats

The following patterns show the drawing formats and scale used in this study. There is only one type of format per character. The formats indicate the starting position and pen movement direction for each stroke. The numbers adjacent to the strokes indicate the order in which the strokes are drawn. The maximum height of the character patterns is 2.5 cms.



Drawing Formats continued

APPENDIX 3 - User's Manual

There are 4 major activities involved in using this recognition system :

[1] Input of training set patterns

[2] generation of the transformational weights

[3] generation of the reference patterns

[4] using the recogniser itself

It is generally necessary to perform these activities in the sequence indicated. The details of each of these activities is described below.

1. Input of Training Patterns

The training patterns provide the system with information about what the average member of a character class is like, and the variation that can be expected in the patterns belonging to a particular class. Therefore, the examples input on the training set file should be representative of patterns of their character class, and sufficiently numerous so as to adequately estimate the means and covariances for that character class. It is not known what is a reasonable lower limit on the number of examples required to define a character pattern, but a minimum figure of 10 seems appropriate.

The use of the program 'TRAINER' which allows the entry of training patterns is reasonably self explanatory. As with all the programs used in this project, the user is prompted for information when it is required. The program not only enables the input of training patterns, but also permits the reviewing of individual patterns, and can generate a report of character frequencies in the training set.

The basic requirements for operation of this program are :

[1] entry of the name of the training pattern file. If the file doesn't exist it will be created.

[2] if character entry is desired, then the digitiser has to be set up (see Appendix 1).

The method of entering the character pattern requires some explanation. The steps involved are detailed :

[1] Set up the TRAINER program in input mode, ensuring that the digitiser is set up properly. The prompt 'ENTER CHARACTER ON DIGITSER' indicates when the program is set up for input.

[2] Write the character in a deliberate fashion on the digitsing pad using the special pen. Lift the tip of the pen off the pad in between strokes.

[3] To signal that the character has been entered, press any key on the keyboard and then touch the tip of the pen down onto the digitising pad.

[4] Wait for the entered character to be displayed after preprocessing. In the event that nothing happens, try step [3] again. If that achieves nothing then check that the digitiser is set up correctly.

2. Generation of Transformation Weights

Weights can be generated for either Fourier or Discriminant analysis transformations.

2.1. Fourier Transforms

The program 'FOURIER' is used to generate the weights for Fourier transforms and stores the weights on file. This program does not have the capacity to create files, so if a new set of weights is being generated then the program 'INITCOV' will have to be run to create an appropriately sized empty file. INITCOV is to be run with the following inputs:

[1] The name of the file to be created.

[2] The number of 'reference' records. Just enter '0' to this query.

[3] The patterns 'to be stored'. Just ignore this by pressing enter.

The Fourier weights are generated by running FOURIER which requires :

[1] The name of the transform file created by INITCOV.

[2] The printer must be on-line so that the weights can be printed out.

2.2. Discriminant Transforms

The programs 'DISCRIM1' and 'DISCRIM2' must be run in sequence to generate these transforms. The discriminant analysis processing was broken up into 2 programs because of the lack of memory on the Apple. As for Fourier transforms, a new file is created first by using INITCOV which requires :

[1] The name of the new transformation file.

[2] The number of 'reference' records. Just enter '2' here.

[3] The names of the patterns which the discriminant transforms will be generated for.

These are to be entered all on one line without any intervening spaces.

DISCRIM2 can be run once the file has been created. This program calculates the between-class and within-class covariance matrices appropriate for the characters with 1 and 2 strokes. This program requires :

[1] The name of the training set file.

[2] The name of the new transformation file created by INITCOV.

[3] The printer must be on-line to print the output covariance matrices.

When DISCRIM1 has finished (it takes about 30 minutes to run), DISCRIM2 can be executed. This program calculates the eigenvalues and eigenvectors appropriate for 1 and 2 stroke characters. It requires :

[1] The name of the transformation file output by DISCRIM1.

[2] The tolerance values to be used in the calculation of eigenvalues. A value of about 0.1 is suggested.

[3] The printer is required to be on-line throughout.

This program takes about 3 hours to run. When the program finishes, the main menu for the UCSD Pascal system is displayed. If the program abends with a floating point error, then the tolerance value entered in step 2 was probably too small. In this case re-run DISCRIM1 and then DISCRIM2, specifying a larger tolerance value. Also rerun DISCRIM1 prior to rerunning DISCRIM2 in the case of any other abend.

3. Reference File Generation

The program 'GENREF' generates reference patterns. Like the transformation generating programs, an empty reference file has to be created before GENREF is started. When using INITCOV specify :

[1] The name of the new reference file.

[2] The number of the reference records required. This is the same as the number of potentially recognisable characters.

[3] The names of the patterns to be stored on this file. Input all these single character labels on one line without intervening spaces.

After the file has been created, GENREF can be run, and it requires :

[1] The name of the input training pattern file to be processed. This file contains representative patterns for the recognisable characters.

[2] The name of the output reference file (created by INITCOV).

This program takes about 30 minutes to execute.

4. Recognition

Two programs involve recognition-type processing : 'PROB' and 'PROBTAB'. PROB is used for on-line recognition purposes and prints out probability estimates for character classes. The requirements for this program are :

[1] The digitser must be set up as for character pattern input.

[2] The name of the reference file to be used.

PROBTAB is used to tabulate a confusion table which indicates what characters get confused with what others in the recognition process. The requirements of this program are :

[1] The name of the test pattern file. These test patterns are entered and stored in the same way as training patterns.

[2] The name of the reference file to be used.