ABSTRACT

Character Recognition Using String Matching

by

Xiaozhi Ye

To handle noisy and distorted pattern is the use of similarity or distance measures. A similarity or distance measure can be defined between a representation of an unknown pattern and a representation of a prototype pattern. Recognition of the unknown pattern can be carried out on the basis of the maximum-similarity or minimum distance criterion (Bunke 1990) This approach is proposed to recognize the noisy or distorted character image. In this work, a directly string representation of the pattern (prototype as well as unknown input) using the histogram method, a decision procedure for classification is the well known Levenshtein distance or weighted Levenshtein distance (Wanger 1974; Hall and Dowling 1980), the cost of a transformation is specially estimated, and typical elements of the sample set are chosen and a well known statistical decision--nearest neighbor classification (NN-classification) is applied.(P. A. Devijver and J. Kitler 1982) Experiments show that it is an efficient method and it gives satisfactory results.

CHARACTER RECOGNITION USING STRING MATCHING

by
Xiaozhi Ye

A Thesis
Submitted to the Faculty of
New Jersey Institute of Technology
in Partial Fulfillment of the Requirements for the Degree of
Master of Science
Department of Computer and Information Science
October, 1992

APPROVAL PAGE

Character Recognition Using String Matching

by

Xiaozhi Ye

Dr. David T. Wang, Thesis Adviser
Assistant Professor of Computer Science and Information Science, NJIT

BIOGRAPHICAL SKETCH

Author: Xiaoxhi Ye

Degree: Master of Science in Computer and Information Science

Date: October, 1992

Date of Birth:

Place of Birth:

Undergraduate and Graduate Education:

- Master of Science in Computer and Information Science, New Jersey Institute of Technology, Newark, NJ, 1992.

- Bachelor of Science in Computer Science, Fudan University, Shanghai, China, 1982.

Major: Computer and Information Science

ACKNOWLEDGEMENT

The author wishes to express his sincere gratitude to his supervisor, Dr. David T. Wang, Professor of Department of Computer and Information Science, for his guidance, friendship, and moral support throughout this work.

The author also appreciates the timely suggestions from the Documentation Laboratory members of the Department of Computer and Information Science, this includes Li Qiuling and Gao Ming.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Figure **Page**

CHAPTER 1

STRING MATCHING METHODS

1.1 Introduction

Two principle areas of contemporary pattern recognition are the decision theoretic or statistic approach and the syntactic/structural approach.

Decision theoretic or statistic methods are based primarily on using numerical valued features as a means for distinguishing one class of patterns from one other classes. By contrast, syntactic and structural methods are based on explicit or implicit representations of a class's structure.

Each of these methods has its strength and its limitations. In order to take advantages of these methods, different methods are combined sometimes to overcome its limitations.

1.2 Structural Representation and Matching

Structural representation used in pattern recognition most commonly are the string, trees, graphs and arrays. With respect of computational composite, strings are very efficient since similarity measures between strings can be computed very fast. However, string are limited in their representational power.At the other extreme, graphs are most powerful approach to structural pattern representation. But graph matching is conceptually rather complicated and expensive with respect to computational cost. So there is a tradeoff between representation and complexity required by matching.

1.3 String Matching Methods

1.3.1 Wanger and Fischer Algorithm

In this algorithm three elementary operations are defined; combination of them allow any string to be transformed into any other. The operation act on the terminals of the alphabet and a cost is associated with each. The operations areas follows:

substitution $a \dashrightarrow b$ cost $= c\,(a, b)$

insertion $\varepsilon \dashrightarrow b$ cost $= c\,(\varepsilon, b)$

deletion $a \dashrightarrow \varepsilon$ cost $= c\,(a, \varepsilon)$

The edit distance c(x,y) between two string x, y is defined as the cost of the transformation of minimum cost from x to y; in coding theory this is known as Levenshtein distance(Levenshtein, 1966)

The basic idea of this algorithm is to calculate the distance matrix.let D(i,j) be the element of the distance matrix, we have

$$\mathrm{D}\,(\mathrm{i}, \mathrm{j}) = \min \begin{cases} D\,(i\text{-}1, j\text{-}1) + c\,(a, b) \\ D\,(i\text{-}1, j\,) + c\,(a, \varepsilon) \\ D\,(i, j\text{-}1) + c\,(\varepsilon, b) \end{cases}$$

Suppose we have a n x m matrix, when computation reaches last element D(n,m)This will be the minimum distance between string x and string y.(Wanger 1974; Hall and Dowling 1980)

1.3.2 Similarity Method

The algorithm uses the opposite way to measure the two strings. This method is due to T. W. Sze and Y. H. Yang. we briefly describe the algorithm.

Given $x = x_1x_2 \ldots x_n$ and $y = y_1y_2 \ldots y_m$. An identity at position i is defined as if and only if $x_i = y_i$, i = 1, ... ,min(n, m). Let I the number of identities. We define

$$N = max(n, m) - I,$$

$$s(x, y) = I/N,$$

$$d'(x, y) = N/I.$$

Obviously, s(x, y) is a measure of similarity while d'(x, y) can be interpreted as a distance between x and y.

The similarity measure has an advantage of computation complexity over the above algorithm. It is order O(max(n, m)). However, its application is restricted to the two strings are properly alignment.(T.W. Sze and Y.H. Yang 1981)

1.3.3 Similarity of Common Substring

A subsequence of a string $x = x_1x_2 \ldots x_n$ is defined as an string $x_{i1}x_{i2} \ldots x_{il}$ where $1 \leq i1 \leq i2 \leq \ldots \leq il \leq n$. Let z be a common subsequence of two strings x and y if z is a subsequence of both x and y.

Let l(x,y) be the length of the longest common subsequence of x and y.Then l(x,y) can be consider as a similarity measure between x and y. The largest is the degree of similarity, or equivalently, the smaller is the distance between x and y. if we define $c\ (a, b) = 2$, $c\ (a, a) = 0$,$c\ (\varepsilon , a) = c\ (a, \varepsilon) = 1$ for any a, b $\in$ T; a $\neq$ b.

In this case, we can have

$$d(x, y) = n + m - 2l(x, y)$$

and, therefore,

$$l(x,y) = (n+m-d(x,y))/2$$

where n and m denoted the length of x and y respectively. Obviously, the computational complexity of the similarity measure l(x,y) is O(nm) with respect both time and space.(Chen, H. H, Lee, Y. C, Sun, G. Z. and Lee, H. Y. 1986)

1.3.4 Warping and Elastic Matching

In some application, it is desirable to allow the striching, or expansion, of a single symbol a into a consecutive symbols $a_1 \ldots a_k$ as well as the compression of $a_1 \ldots a_k$ into a without any costs, where $a = a_1 = a_2 = a_3 = \ldots = a_k$ $k \geq 1$ This problem is often refered as elastic matching, wrapping. (K. Abe and N. Sugita. 1982)

1.3.5 Generalized Operation Method

From a general point of view, the three edit operations - substitution, insertion and deletion can be generalized as one elementary operation -- substitution :

substitute a string of length 1 by another string of length of 1

(edit operation a --> b)

substitute a string of length 0 by another string of length of 1

(edit operation ε --> a)

substitute a string of length 1 by another string of length of 0

(edit operation a --> ε)

As an elementary edit operation, substitution u --> v, where both u and v are arbitrary strings of any length greater than or equal to zero.Let c(u--> v) denote the costs of the generalized substitution u --> v.Notice that c(u --> v) will usually be different from d(u,v).

In the following we always assume c(u --> v) = 0 if u = v for any strings u, v $\in$ T* Furthermore we assume there is a finite number of generalized substitutions u --> v. Let S be the set of these generalized substitutions.

To calculate the distance d(x, y), again use an (n + 1) x (m + 1) array D(i, j) where n and m are the lengths of x and y, respectively. We fill the elements of the array row by row from left to right. D(x, y)s equal to d ($x_1 \dots x_i$, $y_1 \dots y_j$) for i = 1, ... , n and j = 1, ... , m.

To compute D(i, j) require searching for a matrix element D(r, s) with $r \le i$, $s \le j$, $x' = x_1 \dots x_r$, $y' = y_1 \dots y_s$, $x'' = x_{r+1} \dots x_i$, $y'' = y_{s+1} \dots y_j$,such that the value of D(r,s) plus the costs for the transition from D (r,s) to D (i,j), i.e. c(x'' --> y''), is minimal. Of course, The general substitution x'' --> y'' is in the S. Formally, the computation of D(i, j) is based on the formulae:

$$D(i, j) = \min \{ d(x', y') + c(x'', y'') : x'' \dashrightarrow y'' \in S,$$

$$x_1 \ldots x_i = x'x'', y_1 \ldots y_j = y'y'' \}, \quad i = 0, \ldots, n; \; j = 0, \ldots, m.$$

$$d(x, y) = D(n, m).$$

where $x' = x_1 \ldots x_r$, $y' = y_1 \ldots y_s$, $x'' = x_{r+1} \ldots x_i$, $y'' = y_{s+1} \ldots y_j$ Finally it reaches at D(n, m). It is illustrated in Fig.1

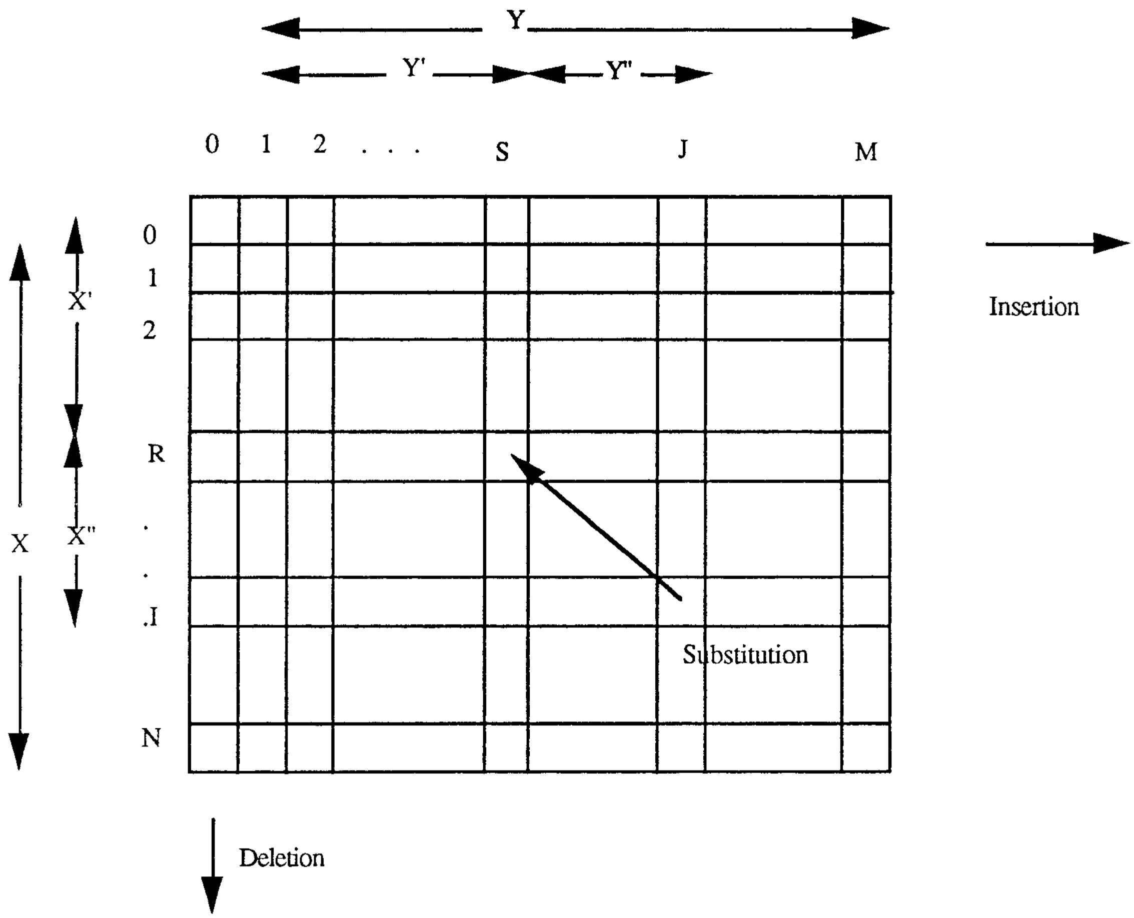

Fig. 1. The string distance computation based on the generalized substitution

The computational complexity is dependent on the different approaches. (J.B. Kruskal and D. Saukoff. 1982)

1.3.6 Other Methods

There are many other methods related the string matching and distance or cost definition. Such as one method in which the costs are based on the context, that means a symbol is operated or not depends on its context. And a method enumerates and evaluates the substring in the given string and a measure of similarity is derived from the results.(Findler Niv and Van Leeuwen J.) Of course, there is efforts to reduce the complexity.An algorithm splits the distance matrix D(x, y) into submatrices and precomputes all operations to be performed on these submatrices. The time complexity of the method is O(nm/min(m,log n). (W.J Masek and M. S. Paterson. 1980)

1.4 The Adapted Method

For the purpose to get better performance, this paper proposes a hybrid method -- structural-statistic approach by using string matching to recognize character.

The common idea of structrual matching is to compare an unknown pattern with a number of sample, or prototype, patterns using a distance or similarity measure. There is a well definition of the distance between unknown input string and prototype string, that is the Levenshtein distance, based on the cost of the edit operations. And many formal measures have been proposed. (D. F. Stanat and D. F. McAllister. 1977)

There are two phases:

1. Learning phase;

2. Recognition phase

Each phase involves two important steps

(1) Histogram string generation,

(2) Distance measurement.

These phases and steps will described in following sections

CHAPTER 2

TWO IMPORTANT STEPS

2.1 Two Phases of Recognition

Both learning and recognition phases involved the histogram string generation and distance measurement. These two steps play an important role in two phases. All information in the character image should retain in the corresponding string, but the noise is introduced in the string at the time it was transformed. The cost of three operations will affect the correct result of recognition. So it is essential.

2.2 Histogram String Generation

There are several representations used in the pattern recognition, such as string,graph, tree and array. One can choose any proper structrue to cope with particular cases.The idea of the approach is to directly represent prototype by data structure -- string. The string is transformed from the character image by using histogram method. These strings keep all information contained in the character image, whatever there are noise or not. Then it apply the algorithm of the Levenshtein distance where the cost is investigated to archeave optimize matching to get satisfied classification.

After the image of character is segmented,of course there are unvoidable noise or distortion, the number of 1's in all the row (vertical histogram) and columns (horizontal histogram) are calculated and transformed into a string for two histogram according to the transform table. In the transform table, the value of the number of 1'sis replaced by corresponding symbol. The symbol is from 1 to 9 If the value is less than nine; The symbol is from a to z if the value is greater than nine. The detail is shown in the transform table.

Symbol Transformation Table

Symbol Transformation Table							
Symbol	No. of 1	Symbol	No. of 1	Symbol	No. of 1	Symbol	No. of 1
1	1	a	10	k	20	u	30
2	2	b	11	l	21	v	31
3	3	c	12	m	22	w	32
4	4	d	13	n	23	x	33
5	5	e	14	o	24	y	34
6	6	f	15	p	25	z	35
7	7	g	16	q	26		
8	8	h	17	r	27		
9	9	i	18	s	28		
		j	19	t	29		

Table 1. Character (a - z) and number (1 - 9) convert to value of 1's

For example, the string for the character B shown in Fig. 2

```
1111111111111111111000000
0111111111111111111110000
0011111111111111111111000
0001111111110001111111100
0001111111100000111111110
0001111111100000111111110
0001111111100000111111110
0001111111100000111111110
0001111111100000111111110
0001111111100000111111100
0001111111100000111111100
0001111111100001111111000
0001111111110111111110000
0001111111111111111100000
0001111111111111111111000
0001111111110111111111100
0001111111100001111111110
0001111111100001111111111
0001111111100001111111111
0001111111100000011111111
0001111111100000011111111
0001111111100000011111111
0001111111100000011111111
0001111111100000011111111
0001111111110000111111110
0011111111110001111111100
0011111111111111111111000
0111111111111111111110000
0111111111111111111000000
```

vertical string = kllihhhhhgggiikkiiihhhhiijllj

horizontal string = 147tttttttd8aaepttrqnje7

Fig. 2. Transformation from an image into a string

After the vertical string and horizontal string are generated, then two strings are concatenated into one string. This is very simple way to get efficient.

2.3 Distance Measurement

Syntactic and structural pattern recognition is based on discrete mathematical relations (D. F. Stanat and D. F. McAllister. 1977) as the detailed descriptions of structure.The Cartesian product of set A with set B is the set of all ordered pairs

$$\{(a,b) \mid a \text{ in } A, b \text{ in } B\}$$

and is denoted A x B. A relation R from A to B is subset of A x B.A multiple relation can be similar defined on the multiple Cartesian
product A x B x C x ... x Z.

The simplest structural description of a pattern is its representation as an ordered sequence of elementary components; the presence or absence of a component, and the relative positions of the components, characterize the pattern taken as a whole. Comparison of two such descriptions provides a measure of the extent to which the corresponding patterns resemble one another.

2.3.1 Alphabets, Concatenation, Strings

Definition 1

An alphabet is a finite set of elements called terminals; the alphabet is denoted by T, the terminals by 1, 2,...,9, a, b, c,..., z.

In pattern recognition, terminals are distinct elementary forms resulting from preprocessing.

Definition 2

A string on T is an ordered sequence of elements of T and is represented by simple juxtaposition or concatenation of these elements.Example

Alphabet: T = { a, b, c }

String on T: x = bcaab

Definition 3

The length of a string x is the number of elements of which it consists. This is written |x|. In the above example |x| = 5

Definition 4

The empty string, written e , is the sequence of terminals of length zero.

Definition 5

The set of all strings on T is denoted as T^* and the set of non-empty strings T^+

Definition 6

There is an internal associative operation on T^* called concatenation, defined as follows. Let

$$x, y \in T^* \cdot \text{ where } x = x_1x_2 \dots x_n \text{ , } y = y_1y_2 \dots y_m$$

Then

$$xy = x_1x_2 \dots x_ny_1y_2 \dots y_m \in T^*$$

This operation is not in general communicative; it has e as neutral element.

Definition 7

$y \in T^*$ is called a substring of the string $x \in T^*$ if there are in T^* two strings u, v such that $x = uyv$

2.3.2 Comparison of Strings

If the objects being studied are described in terms of strings, then in order to recognize such objects a way must be found to compare them with prototypes described in the same way; this requires algorithms to be constructed that will give a measure of the resemblance between two strings written in the same alphabet.

Clearly, it is important to take account of the reasons for this operation and avoid any very direct comparison with two strings of equal length, which is unrealistic limitation.

We have defined the cost of three edit operations. Now let $s = e_1, e_2, \ldots, e_n$ be a sequence of edit operations for transforing a string x into another string y. the cost c(s) of these sequence are given by

$$c(s) = \sum_{i=1}^{n} c(e_i)$$

The distance d(x,y) between x and y can be defined, according to c(s),by

$$d(x, y) = \min \{\ s \text{ is a sequence of edit operations which transforms } x \text{ into } y\ \}$$

So the distance between x and y is obtained by summing up the costs of all elementary operations of the sequence with minimum total costs among all sequences which transform a string x into another string y.

To find the solution of the minimum of cost sequences, the Wanger and Fischer algorithm is commonly used for calculation of d(x, y) according the definition c(s).

In the proposed method, The distance between the generated string and the strings of the prototypes of classes are measured by the algorithm -- the Levenshtein distance. This is a dynamic programming procedure, i.e. a particular breadth first searching.(Chen, H. H, Lee, Y. C, Sun, G. Z. and Lee, H. Y. 1986) And it is an error-correcting string matching algorithm. The complexity of this algorithm is O(mn), with respect to both time and space, where m and n being the length of two strings. Three edit operations, namely insertion, deletion, and substitution, are introduced in the measurement of the Levenshtein distance. In this step, the cost of three operations are defined as following:

the cost of insertion: cost(a --> b) = 1;

the cost of deletion: cost(a --> ε) = 1;

the cost of insertion: cost(ε --> b) = coefficient x difference of two terminals or symbols.

CHAPTER 3

STATISTIC AND STRUCTURAL APPROACH

3.1 Learning and Recognition Phases

Statistic-structrual approach includes learning and recognition phases. In order to optimize classification performance, In the learning phase the procedure was trained to remember the "typical elements", the center of the prototypes, which are obtained by applying the statistical method. And for efficient and correct recognition some mechanisms are involved. They are reducing executing time by checking the length of a string to make sure if it belongs to some class and is worth calculating, and if the difference may caused by noise then ignore it.

3.2 Learning Phase

In this phase all character image are transformed into strings using histogram method.Because of noise or distortion the images of the same character are obviously different.It will result the different strings and bring the incorrect recognition.In order to measure the degree of typicalness of an element contained in a class. One way is to calculate the average distance of this element to all other members in its class.

Let A_i^1 be an average distance of one element in a class C_1 , A_2^1 be an average distance of one element in a class C_2,

$$A_i^1 = \frac{1}{N} \sum_{k=1}^{N} d(x_i^1, x_k^1), \ i = 1, 2, ..., N.$$

$$A_J^2 = \frac{1}{M} \sum_{l=j}^{M} d(x_j^2, x_l^2), \ j = 1, 2, ..., M.$$

The smaller A_i^1 is, the more typical is x_i^1 with respect to its class C_1. Only a certain number of elements $x_1^1, x_2^1, \ldots, x_n^1$ and $x_1^2, x_2^2, \ldots, x_n^2$ are kept.

$$A_i^1 \leq A_k^1; \ i = 1, 2, \ldots, n, \ k = n+1, \ldots, N.$$

$$A_j^1 \leq A_l^1; \ j = 1, 2, \ldots, m, \ k = n+1, \ldots, M.$$

The method described above tend to retain elements which are near to the center of a class.

3.3 Recognition Procedure

The basic idea of structural matching, i.e. recognition, is to match two string, an unknown input pattern and the prototypes, in order to find the prototype which is most similar to an unknown input pattern. The advantages of using string matching are that it is very efficient. A well-known concept from statistical decision theory, nearest-neighbor classification (NN-classification), is applied in this procedure. Using the distance measure of the Levenshtein distance to classify the unknown string x. Let D_i be the distance between x and that element in C_1 which is closest to x, i.e.

$$D_1(x) = \min.\{ d(x_i^1, x) : i = 1, \ldots, N\}$$

similar, we define

$$D_2(x) = \min.\{ d(x_i^2, x) : i = 1, \ldots, M\}$$

Now the NN decision rule is given by

$$x \in \begin{cases} C_1 & \text{if and only if } D_1(x) \leq D_2(x) ; \\ C_2 & \text{othewise.} \end{cases}$$

1) Get an unknown input image and transform into a string as a pattern, to be compared with the prototypes of classes. Similar it is a histogram string generation procedure. Of course, the length of the string is not a constant due to the noise or distortion.

2) Before executing the basic algorithm, check the length of the transformed string if it is within a certain range. If the length exceeding the range, the algorithm consider that it does not belong to a certain class of prototypes and ignore it. Then match with another one to avoid exhaustive calculations of the algorithm.Otherwise calculate the distances between unknown input string and all possible candidates. Choose the prototype which has the minimum distance with input string and identify the real character corresponding to the selected prototype.

3) In a class each prototype has a substring which is almost the "same" with others. where the "same" means the difference value of the two terminals is only one.If we ignore this little difference, there will be a longer length of the "same" substring.It surely will result in smaller distance in the measurement.During the matching, in the case of very little difference which probably caused by noise or distortion are distinguished as no difference to make the unknown input to approach to the center of the prototypes of a class as closer as possible.

3.4 Basic Algorithm

If the objects being studied are described in terms of strings, then inorder to recognize such objects a way must be found to compare them with prototypes described in the same way; this requires algorithms to be constructed that will give a measure of the resemblance between two strings written in the same alphabet.

We consider the Wanger and Fischer's algorithm is useful method. As mentioned the various methods are based on this basic idea. From this algorithm different distance measurements are produced and many applications are realized. Obviously, the method is applied in this experiment.

The algorithm below computes the elements of (n+1) x (m+1) matrix D(i,j) row by row from left to right. The first row and first column of matrix D(i, j) is separately computed in initial step. The symbols x_i and y_j in strings x and y are denoted by x(i) and y(j), respectively; i = 1, ... , n; j = 1, ... ,m. In each element D(i, j), the minimum accumulative costs are stored for transforming $x' = x_1 \ldots x_i$ into $y' = y_1 \ldots y_j$ i.e. D(i, j) = d(x' , y') . At last, the element at the lower right corner of a matrix contains the value d(x,y).

According to the algorithm, each element D(i,j) is determined by three potential predecessors, namely D(i, j-1), D(i-1, j-i) and D(i-1, j), i = 1, ... , n; j = 1, ... ,m. A graphical illustration is shown in Fig. 3.

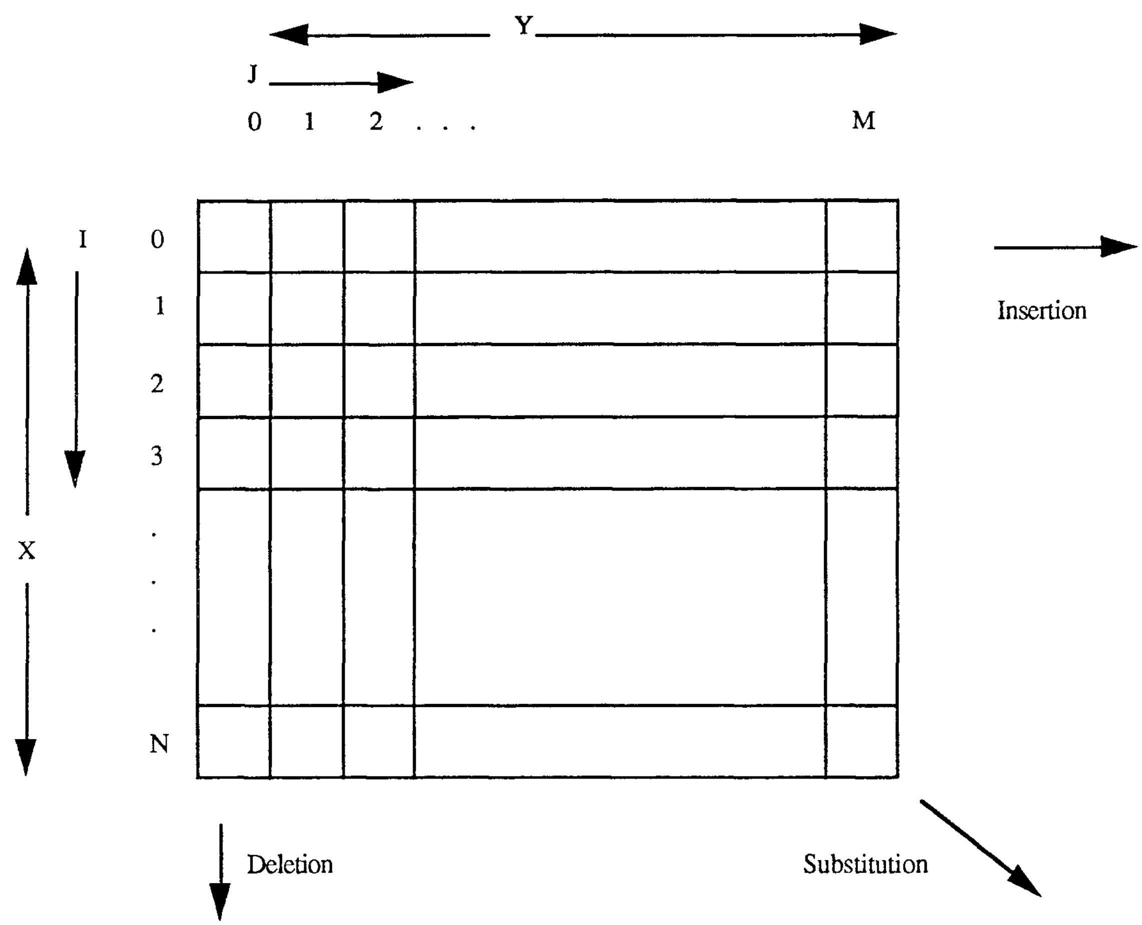

Fig. 3. Illustration of the calculation of the basic algorithm

Algorithm

Input:

$$\mathbf{x} = x_1 \ldots x_n \in \mathbf{T}^*, \mathbf{y} = y_1 \ldots y_n \in \mathbf{T}^* .$$

cost (a-->b) ; a, b $\in$ T $\cup$ { ε }

Output:

d(x, y) = **minimum distance**

Method:

```
begin
    D(0, 0) : = 0;
    for i=1 to m do D( i, 0 ) : = D(i-1,0) + c (x(i)--> ε );
    for j=1 to m do D( 0, j ) : = D(0,j-1) + c ( ε -->y(j));
        for i=1 to n do
            for j=1 to m do
                begin
                    m1: =D( i-1, j-1 ) + c( x(i)--> y(j));
                    m2: =D( i-1, j ) + c( x(i)--> ε );
                    m3: D( i, j-1 ) + c( ε --> y(j));
                    D( i, j ): = min(m1, m2, m3 );
                end
        d(x,y): = D(n, m);
        end
end
```

CHAPTER 4

EXPERIMENT RESULTS ANALYSIS

4.1 Recognition Performance Analysis

As an experiment several cases are tested in order to check the performance of the approach. In the observation of different size of K prototypes in a class, the experiment shows as K increases the performance becomes better than previous case, but the limitation still exists, after certain K performance decreases.

The reason of the performance dropping is that more prototypes are selected, there will be more chance some prototypes are far away from the center. It will results to mismatch to another prototypes, thus the the performance decrease after certain values.

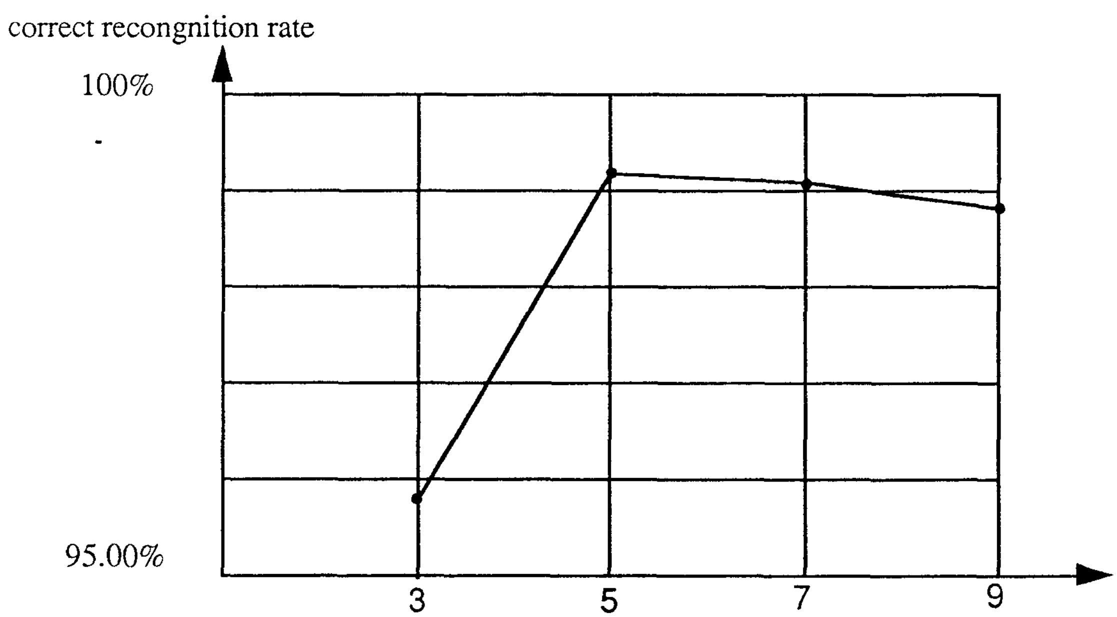

Fig. 4 The performance of recognition with different number of prototypes in one class

4.2 Input Sample

The input samples are the strings which are transformed from the input images of every testing characters For example, the character A , its strings is shown bellow:

==

Input string=> 245667899bbcdcccehjjjfdddeimm3468bdccdecfhlnnookifdb87532

==

Input string=> 34566889abccdccdehijkieeefjml3478addcdeegilnoqoljfeb97532

Because of noise even the one character can have many different strings. The big problem is the algorithm have to watch these variant samples and to recognize them correctly as the corresponding character.

4.3 Output Example

The information related to the output of result of the recognition is as followings:

==

Input string=> 245667899bbcdcccehjjjfdddeimm3468bdccdecfhlnnookifdb87532

Weight => 6.0000006

cit_cand[0]=0

candidate string[A]=> 24566889abccddcdeiijkgdedhlml3468adeddfdfhlmopoljgec97543

==

Input string=> 34566889abccdccdehijkieeefjml3478addcdeegilnoqoljfeb97532

Weight => 4.0000006

cit_cand[0]=0

candidate string[A]=> 24566889abccddcdeiijkgdedhlml3468adeddfdfhlmopoljgec97543

==

where the input string is obtained from the character image, the weight is distance which is the result of the basic algorithm, the cit_cand means the number of candidates that have same distance are contained in this array, and last one is the prototype which is matched with input string.

4.4 Experiment Results

In our experiments several sizes of the prototype have been tested.

The results are shown in a table below:

Statistic Data of the Experiments				
Size of prototype	Total No. of Character	No. of Errors	Error Rate	Correct Rate
3	2444	102	4.18	95.82
4	2392	24	1.01	98.99
5	2340	19	0.82	99.18
7	2236	22	0.88	99.02
9	2132	22	1.04	98.96

Table 2. Results of different size of prototypes

CHAPTER 5

SUMMARY

In this work, error tolerance classification for character recognition is presented. Although the noise or distortion of character image is unavoidable, the recognition procedure with some knowledge,that is roughly distinguished the difference or cost instead of exactly difference or cost, will discard the affect of the noise and by judging the length of an image string to bypass obviously unnecessary calculations thus to reduce the executing times of the basic algorithm.

The experimental result shows that as the performance becomes more and more satisfied, this method is correctly applied in the experiment. It should be noted that to improve performance of this approach, properly clustering and more efficient search (smart search)should be introduced. Also if applying scaling function could get uniform string length, clustering and classification will be more efficient.

REFERENCES

Bunke, H. 1990. " String Matching for Structural Pattern Recognition." Syntactic And Structural Pattern Recognition Theory And Applications, Series in Computer Science, World Scientific Publishing Co., Singapore. 7: 119-144.

Wagner, R. A. and Fischer, M. J. 1974. " The String-to-String Correction Problem." J. ACM. 21:168-173.

Hall, P. A. V. and Dowling, G. R. 1980. " Approximate String Matching." Computer Survey.12: 381-402

Devijver, P. A. and Kitler, J. 1982. "Pattern Recognition: A Statistical Approach." Prentice Hall, London. pp. 120-125

Stanat, D. F. and McAllister, D. F. 1977. " Discrete Mathematics in Computer Science." Prentice Hall, Englewood, Cliffs. NJ.

Chen, H. H., Lee, Y. C., Sun, G. Z. and Lee, H. Y. 1986. " High Order Correlation Model for Associative Memory." Amer. Inst. Phys. Conf. Proc. 151, Snowbird, Utah. pp. 86-99.

Sze, T. W. and Yang, Y. H. 1981. "A Simple Contour Matching Algorithm." IEEE Trans, PAMI 3: 676-678.

Abe, K. and Sugita, N. 1982. " Distances Between Strings of Symbols-Review and Remarks. " Proc. 6th ICPR, Munich. pp. 172-174.

Kruskal, J. B. and Saukoff, D. 1982. " An Anthology of Algorithms and Concepts for Sequence Comparison. " in Ref. 5: 263-310

Tanaka, E. 1987. " A String Correction Method Based on the Context - Dependent Similarity." in Syntactic and Structural Pattern Recognition, NATO. ASI Series, eds. G. Ferrate, T. Pavlidis, A. Sanfeliu and H. Bunke. (Springer, 1987). pp. 3-17

Findler, Niv and Van Leeuwen, J. " A Family of Similarity Measures Between Two Strings." IEEE Trans, on Pattern Analysis and Machine Intelligent. 1 (1): 116-118.

Masek, W. J. and Paterson, M. S. 1980. " A Faster Algorithm for Computing String-Edit Distances." J. Computer and System Science. 20: 200-205