Recognition of Hand-printed Chinese Characters using Decision Trees/Machine Learning C4.5 System[1]

Adnan Amin[1] and Sameer Singh[2]

School of Computer Science & Engineering[1]
University of New South Wales
2052 Sydney, Australia
(e-mail: amin@cse.unsw.edu.au)

School of Computing[2]
University of Plymouth
Plymouth PL4 8AA
UK
(e-mail: s1singh@plym.ac.uk)

[1] Amin, A. and Singh, S. "Recognition of Hand-printed Chinese Characters using Decision Trees/Machine Learning C4.5 System", *Pattern Analysis and Applications*, vol. 1, issue 2, pp. 130-141, 1998.

Abstract

Recognition of Chinese characters has been an area of major interest for many years, and a large number of research papers and reports have already been published in this area. There are several major problems with Chinese character recognition: Chinese characters are distinct and ideographic, the character size is very large and a lot of structurally similar characters exist in the character set. Thus, classification criteria are difficult to generate.

This paper presents a new technique for the recognition of hand-printed Chinese characters using the C4.5 machine learning system. Conventional methods have relied on hand-constructed dictionaries which are tedious to construct and difficult to make tolerant to variation in writing styles. The paper discusses Chinese character recognition using the *Hough transform* for feature extraction and C4.5 system. The system was tested with 900 characters written by different writers from poor to acceptable quality (each character has 40 samples) and the rate of recognition obtained was 84%.

Keywords

Chinese characters, Parallel thinning algorithm, Hough Transform, Feature extraction, machine learning.

Introduction

For the past three decades, there has been increasing interest among researchers in problems related to the machine simulation of the human reading process. Intensive research has been carried out in this area with a large number of technical papers and reports in the literature devoted to character recognition. This subject has attracted immense research interest not only because of the very challenging nature of the problem, but also because it provides the means for automatic processing of large volumes of data in postal code reading [1, 2], office automation [3, 4], and other business and scientific applications [5–7].

Much more difficult, and hence more interesting to researchers, is the ability to automatically recognise handwritten characters [8–9]. The complexity of the problem is greatly increased by the noise problem and by the almost infinite variability of handwriting as a result of the mood of the writer and the nature of the writing. Analysing cursive script requires the segmentation of characters within the word and the detection of individual features. This is not a problem unique to computers; even human beings, who possess the most efficient optical reading device (eyes), have difficulty recognising some cursive scripts and have an error rate of about 4% in reading tasks in the absence of context [11].

Different approaches covered under the general term ‘character recognition’ fall into either the on-line or the off-line category, each having its own hardware and recognition algorithms.

In on-line character recognition systems, the computer recognises symbols as they are drawn [12–17]. The most common writing surface is the *digitising tablet*, which typically has a resolution of 200 points per inch and a sampling rate of 100 points per second, and deals with one-dimensional data.

Off-line recognition is performed after writing or printing is completed. Optical Character Recognition, OCR [18–20], deals with the recognition of optically processed characters rather than magnetically processed ones. In a typical OCR system, input characters are read and digitised by an optical scanner. Each character is then located and segmented and the resulting matrix is fed into a preprocessor for smoothing, noise reduction, and size normalisation. Off-line recognition can be considered the most general case: no special device is required for writing and signal interpretation is independent of signal generation, as in human recognition.

Research into Chinese character recognition encounters many difficulties. First, there are a large number of Chinese characters (more than 50,000 characters, of which 6,000 are commonly used). Second, Chinese characters have a more complex structure than alphabetic characters and there are a large number of mutually similar characters. These issues become more complicated

in hand-printed Chinese character recognition where characters can appear in different fonts and sizes.

Over the past three decades, many different methods have been explored by a large number of scientists to recognise characters. A variety of approaches have been proposed and tested by researchers in different parts of the world, including statistical methods [21–23], structural [24–26] and syntactic methods [27–29] and neural networks [30–32].

Rule-based systems are also commonly used in character recognition software [33–34]. Unfortunately, many rules must be constructed through experience to achieve good accuracy. For example, 400 rules were used in [33] for recognising ten Arabic digits (0...9), with an average recognition rate of 91.4%. To increase the recognition rate further, more rules have to be added to the rule base in order to have a wider coverage of the variability of writing styles. Obviously, more rules are required for a large character set and the number of rules is not linearly proportional to the size of the character set. The problem becomes even worse in the case of Chinese character recognition.

An alternative to manual rule creation is to learn the rule by example. Symbolic machine learning algorithms are designed to accept example descriptions in the form of feature vectors that include a label that identifies the class to which an example belongs. The output of the algorithm is a set of rules that classifies unseen examples based on generalisations from the training set. This ability to generalise is the main attraction of machine learning for handwriting recognition. Samples of a character can be preprocessed into a feature vector representation for presentation to a machine learning algorithm that creates rules for recognising characters of the same class. Symbolic machine learning has several advantages over other learning methods. It is fast in training and in recognition, generalises well, is noise tolerant and the symbolic representation is east to understand.

In view of these problems, various methods have been proposed in the past. These methods can be classified into two categories. One approach is stroke analysis that extracts features such as endpoints, crossing points, stroke relative positions, etc. from the input pattern. Another approach is pattern matching which is based on the comparison between the input distribution function and the standard one.

Many researches tend to use the stroke analysis approach. For example, Augi [35] uses the concatenation-relation, cross-relation and near relation to analyse Chinese characters. Yamamoto [36] uses the stroke direction, stroke domain and stroke density to classify characters. Hsieh and Lee [37] use a one-dimensional string consisting of a stroke sequence interleaved with relationships between two consecutive strokes to represent a character.

For all of these methods, an accurate stroke extraction algorithm is essential to achieve high recognition rates. Our paper proposes the use of the Hough transform technique to extract features for classifying and recognizing Chinese characters using C4.5 system, to generate a decision tree. This decision tree can then be used to predict the class of an unseen character. Figure 1 depicts a block diagram of the system.

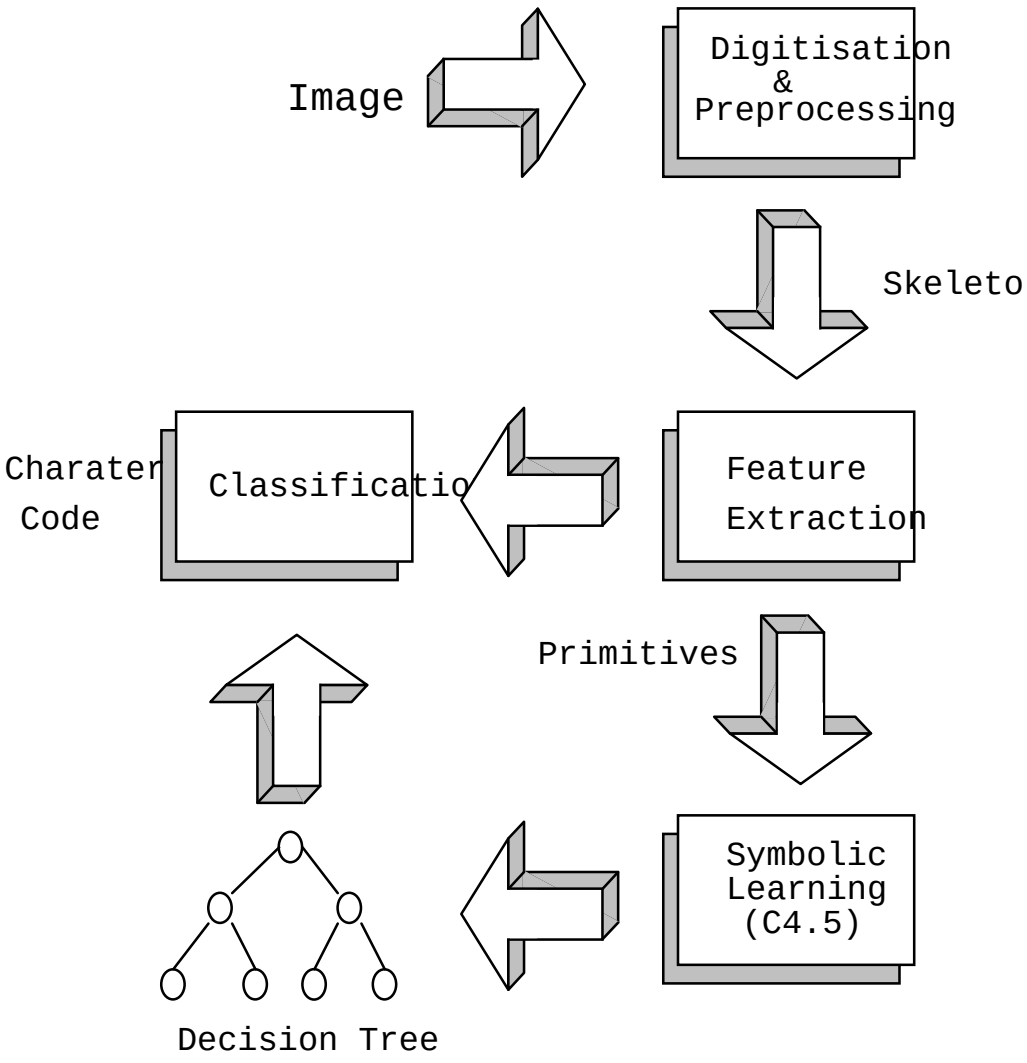

Figure 1: Block diagram of the system

1. Digitisation and Preprocessing

2. 1. Digitisation

The first phase in our character recognition process is *digitisation*. Documents to be processed are first scanned and digitised. A 300 dpi scanner is used for this and the size of the characters is approximately 30mm x 30mm. The next phase is *pre-thinning*.

2. 2. Pre-thinning

This step aims to reduce the noise that the binarization process yields. The pre-thinning algorithm used in this paper is as follows:

Algorithm 2.2.1 Pre-thinning algorithm.

Input: A digitised image *I* in the PBM format.

Output:. A pre-thinned image *I'*, also in the PBM format.

Method

1. For each pixel *P* in image *I*, let *P0*, *P1*, *P2*, *P3*, *P4*, *P5*, *P6* and *P7* be its 8 neighbors, starting from the East neighbor and counted in an anti-clockwise fashion (Fig. 2).

P3	P2	P1
P4	P	P0
P5	P6	P7

Figure 2: P and the labeling of its 8 neighbors.

2. Let $B(P) = P0 + P2 + P4 + P6$. Let *P'* be the corresponding pixel of *P* in *I'*.

3. **If** $B(P) < 2$ **then** set P' to white
   Else If $B(P) > 2$ **then** set P' to black
   Else set P' to the value of P;

2. 3. Thinning

The thinning of elongated objects is a fundamental preprocessing operation in image analysis defined as the process of reducing the width of a line-like object from several pixels to a single pixel. The resultant image is called “the skeleton”.

There is no general agreement in literature on the exact definition of thinness. A study reported in [38], however, examines the connectedness criteria to arrive at a definition. This study concludes that the thinning algorithm should be able to satisfy the following connectedness conditions:

(a) Connectedness is preserved, for both objects (the black pixels) and their complements (white pixels).
(b) Curves, arcs, and isolated points remain unchanged.
(c) Upright rectangles, whose length and width are both greater than 1, can be changed.

Iterative thinning algorithms delete border points (pixels) if their removal does not affect the connectivity of the original object. The original image cannot be recovered from the skeleton in most of these algorithms, however, some thinning algorithms permit the reconstruction of the original image from the skeleton [39-40]. These algorithms, called *reconstructible thinning* algorithms,

search for the set of centers and radii of blocks contained in the image while preserving all details of original objects within the resulting skeleton.

The iterative thinning algorithms can be implemented using either parallel or sequential strategies. In the parallel thinning strategy, the new value given to a pixel in the image during the *nth* iteration depends on its own value as well as those of its eight neighbors at the (*n*-1)th iteration. Hence, all pixels can be processed simultaneously [41-43]. Sequential thinning algorithms, on the other hand, assign a new value to the pixel in the binary image during the *nth* iteration depending on its own value and the values of the eight neighbors at the *n*th and (*n*-1)th cycles, which in turn depend on whether these pixels have already been scanned or not. This pixel processing is done in a sequential manner [44-46].

For conducting the work reported in this paper, a parallel thinning algorithm was used. The parallel thinning algorithm operates by repeatedly removing border points that satisfy certain removal conditions. This removal process is an iterative process performed in steps (or cycles). Assume that the object to be thinned is a 2 x 2 square of ones surrounded by zeros. According to the condition “c” above, the object is to be thinned; hence the removal conditions are applied to all four 1s simultaneously. Since the result of the removal step would be the disappearance of the square, the removal step is divided into smaller steps (called subcycles). If four subcycles are used, then every subcycle will remove one type of border point (North, South, East, West).

The thinning algorithm adopted in this paper is Jang and Chin's one pass parallel thinning algorithm [47] because it gives skeletons with fewer spurious branches. Figure 3 illustrates an original scanned image and the resulting skeleton after applying the thinning algorithm.

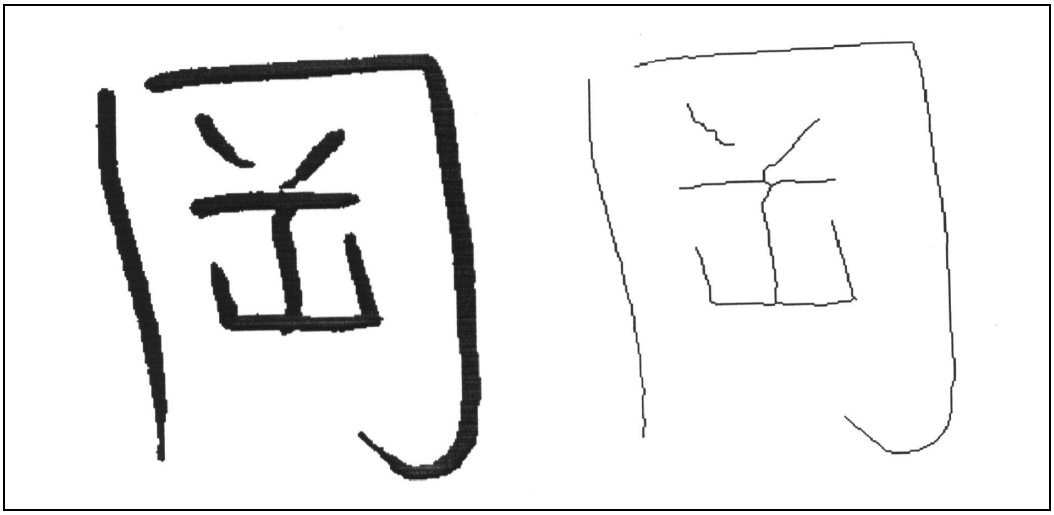

Figure 3: An example of the original image and the thinning result of the Chinese character.

2.4. Post-Thinning

Whenever a thinning algorithm is used, distortions are always brought about after thinning. The most commonly seen distortions are “hairs” in the thinned image and the splitting of fork points. To deal with these problems, a maximum circle technique is proposed to remove spurious branches in thinned images. The description of the algorithm used in this study is as follows:

Input*:* A character image *I* and its thinned skeleton *I’*.

Output*:* A thinned skeleton *I’’* with spurious branches and merged split fork points.

Method*:*

1. **for** each fork point P_i **do**
   begin

   Calculate the radius R_i of C_i, where C_i is the largest circle centered at P_i and within the original unthinned image *I*;

   Create a set S containing only the point P_i ;
   end;

2. **for** every pair of fork points P_i and P_j **do**
   begin

   if (distance between P_i and P_j) <= R_i +R_j **then**
   Merge sets S_i *and* S_j, i.e. sets containing P_i and P_j;
   end;
   end;

3. **for** each of the set S created in (2) **do**
   begin
   for each point P_i in S **do**
   begin
   Reset pixels within the circle C_i ;
   end;

   calculate the average *X* and average *Y* of all P_i in S. The point with coordinate (*X*, *Y*) is the new fork point;

   rejoin the line from the perimeters of circles to the new fork point;

   end.

2. 5. Tracing

After removing spurious branches, the next step is to extract strokes from the image. An algorithm implementing a 3x3 window is used to trace along the path of the skeleton, recording the Freeman codes (0 E, 1 NE, 2 N, 4 W, 5 SW, 6 S, 7 SE) [48] along the path. The Freeman code simply attaches a number to each of the eight major points of the compass and the orientation of a line segment is characterized by this number. A path is described as a trace between junction or end points, where an end point has a single neighbour and a junction point has two neighbours. However, if a point has three neighbors, we consider it as intersection point. Whenever a junction point is reached, a depth-first search is used to follow each path in turn.

The end points in the preprocessed image are used as starting points for tracing. As we trace out the codes, we remove the pixel from the thinned image. After all, since these end points have been visited, the thinned image should now only contain closed loops. We search this image for any remaining pixels, and use the first pixel encountered as our starting point. An example of this tracing procedure is shown in Figure 4.

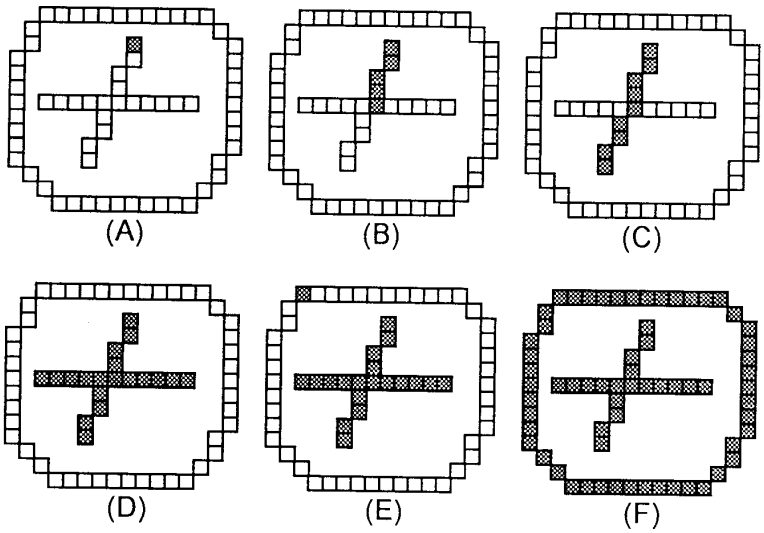

Figure 4: An example of the tracing algorithm.

(A) The starting point.
(B) A fork point is encountered,
(C) The lower branch has been chosen for tracing,
(D) All end points are consumed and the leftover is a loop,
(E) We scanned leftover pixels. The first pixel encountered will be our next starting point.
(F) All pixels are traced.

The product of the tracing algorithm is a set of line segments. We then apply a stroke segmentation technique, using inner products of line segments [49] to join line segments into strokes. The result of the stroke segmentation is the input to our feature extraction block.

An example of our preprocessing is shown in Figure 5.

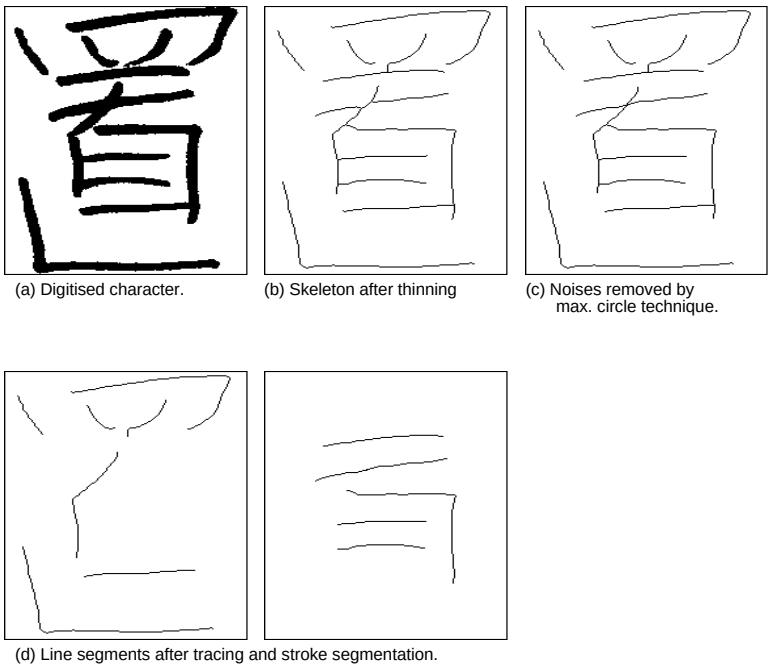

Figure 5: Preparing inputs for feature extraction.

3. Feature Extraction

3.1. Modified Hough Transform

In this project we have used a total of six primitives (Figure 6) extracted by using the *Hough transform* method [50, 51]. The Hough transform is a popular method for straight line detection. Basically, the method is to map the pixels in a picture from the x-y plane to the ρ-θ plane, where ρ is given by the equation:

$$\rho = x_i \cos\theta + y_i \sin\theta \text{ , } 0 <= \theta < 2\Pi$$

Primitive	Name
	Horizontal
	Vertical
	Backslash
	Slash
	Corner
	Dot

Figure 6: Primitive features used in this project.

However, the result of applying the Hough transform to extract the primitives was not satisfactory. The major problem is that the peaks are not obvious. For example, even for a simple pattern like the corner in Figure 7, we have as many as 45 peaks.

To handle the above problems, we modified the Hough transform algorithm to reduce the undesirable peaks. Our method was inspired by the modified Hough transform method in Cheng [52]. In [52], the authors applied a modified Hough transform to all the line segments in the picture and then matched the peaks found against a database. Our method is different in that we applied the transform to each individual line segment, not the whole picture, thus avoiding the noise due to interference by other lines in the picture.

The basic idea of the modified Hough Transform is that by inspecting the neighbours of a pixel, we can determine which directions are more likely to form a straight line, and which directions are less likely or not possible to form a straight line. We assign more weight to the directions that are more likely to form a straight line, and assign less weight to directions that are less likely and no weight to directions that are not possible to form a straight line.

For example, if we know from its neighbour that the pixel is within a straight line, then we can assign no weight to the direction perpendicular to the straight line, as the pixel is not likely to be a part of such a line.

The possible configurations of the neighbours of a pixel is shown in Figure 9, together with the possible directions of forming a straight line and the weight assigned to these directions.

```
starting point = (57 , 193)
ending point   = (105 , 267)
chain code     =
00000700000007000000000001000000000000000070000000000070000000007000007000007000070007707777676677676676766766666666666666666656666665665656655656555454454554554454444544444444444344334344334
```

ρ	θ	frequency
407.182709	0	18
440.856903	0	49
457.69397	5	31
482.949585	10	34
499.786682	15	39
516.623779	20	33
457.69397	30	11
541.879395	30	42
466.112518	35	11
482.949585	35	11
466.112518	40	11
575.553589	50	37
491.368134	55	16
583.972107	60	40
583.972107	65	40
575.553589	75	42
491.368134	80	43
567.13501	85	28
482.949585	90	84
541.879395	95	36
457.69397	100	69
516.623779	105	25
432.438354	110	43

ρ	θ	frequency
499.786682	110	24
415.601257	115	37
482.949585	115	19
390.345642	120	39
457.69397	120	16
373.508545	125	34
440.856903	125	18
348.25293	130	33
331.415833	135	30
306.160217	140	37
289.32312	145	56
339.834381	145	22
272.486053	150	28
322.997314	150	22
247.230423	155	45
221.974792	160	38
205.137711	165	33
255.648972	165	19
179.882095	170	48
154.626465	175	35
205.137711	175	18
137.789383	180	49

Figure 7:A typical output of the Standard Hough Transform. Notice that the line is a corner (type 6 primitive) but since there are too many peaks after transform we cannot find the horizontal and vertical line easily.

The algorithm is presented as follows:

Algorithm 3.3.1 Modified Hough Transform

Input: The coordinates of points $p_1, p_2, \ldots, p_n$ in a line segment L.

Output: A set of tuples (ρ, θ, m) corresponds to the peaks in the parametric space, where ρ is the normal distance of line to the origin, θ is the angle of line to the x-axis, m is the magnitude of the peak.

***Method**:*

1. The modified Hough Transform is basically the same as the standard Hough Transform, and the algorithm is shown as below. Here, $\Delta\theta$ is set to be π / 8 and *IW* is a weighting variable defined in 2. below, $H(\rho,\theta)$ is the accumulator value as in the standard Hough Transform.

```
for each p_i = (x_i, y_i) in L do
        for θ = 0 to 2π step Δθ do
                if θ is valid then
                        ρ = x_i cos θ + y_i sin θ
                        H(ρ,θ) = H(ρ,θ) + IW
                end
        end
end
```

2. The values of θ that are valid and the value of *IW* in the above algorithm is determined by 24 templates. These templates represent a point and its relationship to its neighbour. Each point in the line segment should match one of these templates. The allowable θ and *IW* of a point are those of the matching template.

The result of applying modified Hough transform to that line segment in Figure 7 is shown in Figure 8 below. It can be seen that the number of peaks has been reduced from 45 to 5. However, this is still not the 3 line segments that we wanted. We need further processing of the result.

```
starting point = (57 , 193)
ending point   = (105 , 267)
chain code =
000007000000070000000000010000000000000000700000000007000000000
070000070000700007000070077077767667767667667666666666666666666
656666665665656655656555454454554554454444544444444444344343434
```

ρ	θ	frequency	Start at	End at
0	0	158	0	91
95	90	72	96	142
103.944695	45	38	127	170
70	0	60	151	193
-30.230057	157.5	22	151	193

Figure 8: The result of applying modified Hough Transform to the line in Figure 7 above.

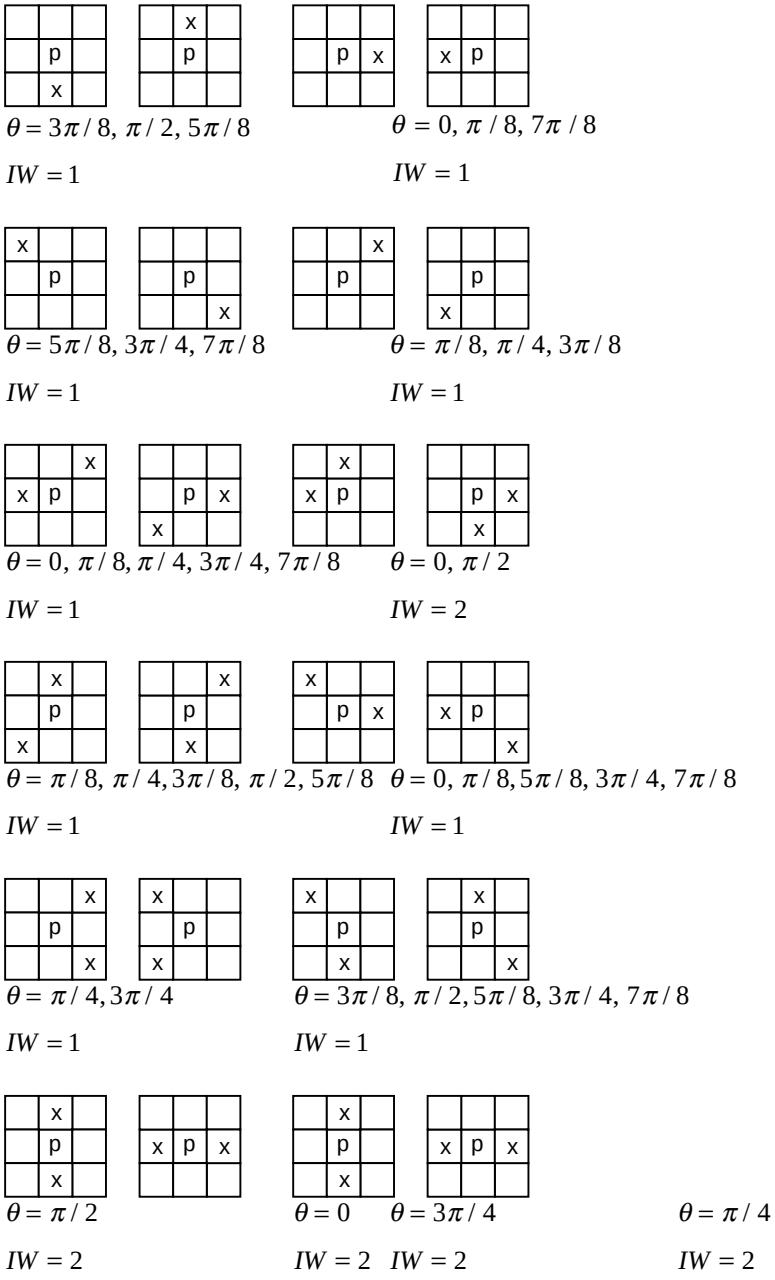

Figure 9: The 24 templates used in modified Hough Transform.

We recorded the starting and ending positions of the Freeman codes that generated the peaks in the result. These positions are shown in column 4 and column 5 of Figure 8. We observed that peaks 4 and 5 were overlapping lines. Also, peaks 2 and 3 were overlapping lines. Since in a line segment there is no reason that two strokes overlap, this result implies that one of 4 or 5 is noise and one of 2 or 3 is noise. Comparing the magnitudes of the peaks, we determined that peaks 3 and 5 are noise. So the remaining peaks 1, 2 and 4 should be the strokes that we wanted. Peak 1 is a horizontal line ($\theta = 0^o$), peak 2 is a vertical ($\theta = 90^o$) and peak 4 is a horizontal ($\theta = 0^o$). Also the sequence is peak 1, then peak 2 and then peak 4 (by ordering the starting positions and ending positions). This is exactly what we wanted to know about this line.

3.2. Stroke Probability

There are many cases where a stroke can be classified as one of the two primitive types. For example, the line in Figure 10 can be considered as either a horizontal (type 1 primitive) or slash (type 4 primitive).

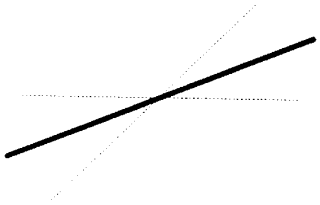

Figure 10: An example of a line which can be considered *horizontal* or a *slash*

Usually, we use 22.5° angle as a dividing line. Those lines with an inclination less than 22.5° are considered as horizontals and those larger than 22.5° are considered as slashes. In hand-writing, this is undesirable because of the variation of writing style of different writers. For example, we should not exclude the possibility of 23° inclined line a being horizontal. Therefore, we assigned an attribute P_s which was the likelihood of a line of being a primitive type. The assignment of P_s is shown in Figure 11.

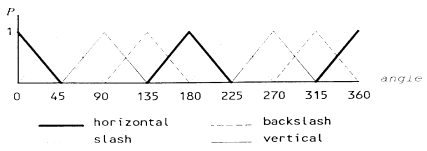

Figure 11: Functions to determine the likelihood of primitives: Horizontal, Vertical, Slash and Backslash.

In Figure 11, the angle is the inclination of the line to the horizontal axis, and *P* is the stroke probability. We assume that the likelihood of a line being a primitive type will linearly decrease with the deviation of the line from the main axes, and can be summarized with the following equations:

Primitive 1 (Horizontal)

$$P_s(\theta) = 1 - \frac{\theta}{45°} \quad \text{for } \theta \le 45°$$

$$P_s(\theta) = 1 - \frac{360° - \theta}{45°} \quad \text{for } \theta \ge 315°$$

$$P_s(\theta) = 1 - \frac{|180° - \theta|}{45°} \quad \text{for } 135° \le \theta \le 225°$$

$$P_s(\theta) = 0 \quad \text{otherwise}$$

Primitive 2 (Vertical)

$$P_s(\theta) = 1 - \frac{|90° - \theta|}{45°} \quad \text{for } 45° \le \theta \le 135°$$

$$P_s(\theta) = 1 - \frac{|270° - \theta|}{45°} \quad \text{for } 225° \le \theta \le 315°$$

$$P_s(\theta) = 0 \quad \text{otherwise}$$

Primitive 3 (Backslash)

$$P_s(\theta) = 1 - \frac{|135° - \theta|}{45°} \quad \text{for } 90° \le \theta \le 180°$$

$$P_s(\theta) = 1 - \frac{|315° - \theta|}{45°} \quad \text{for } 270° \le \theta \le 360°$$

$$P_s(\theta) = 0 \quad \text{otherwise}$$

Primitive 4 (Slash)

$$P_s(\theta) = 1 - \frac{|45° - \theta|}{45°} \quad \text{for } 0° \le \theta \le 90°$$

$$P_s(\theta) = 1 - \frac{|225° - \theta|}{45°} \quad \text{for } 180° \le \theta \le 270°$$

$$P_s(\theta) = 0 \quad \text{otherwise}$$

Primitive 5 (Corner)

Uses the same probability as that of a backslash

Primitive 6 (Dot)

$$P_s(\theta) = 1 - \frac{L}{Tp} \qquad \text{for } L \le Tp$$

$$P_s(\theta) = 0 \qquad \text{for } L > Tp$$

where L is the length of the segment, T_p is the threshold to be determined through experimentation.

4. Machine Learning C 4.5 System

There are several methods through which one may attempt to classify the Chinese character data. These include discriminant analysis, bayesian methods, neural networks and machine learning. Our past experience with the analysis of Chinese characters with discriminant analysis suggests that linear methods are only suited to small samples [53]. In order to achieve good training and testing results for hand-written Chinese characters, methods which can analyse non-linear data are required. Since we were not entirely confident about our data distribution for probabilistic analysis, we have not used bayesian analysis. In addition, long training times with neural networks for our large sample size did not attract us to such an analysis. Our aim is to recognise the given data in a manageable time framework with acceptable error.

The reason for choosing machine learning to solve the character classification problem is that the C4.5 system [54, 55] is an efficient learning algorithm that is well suited to the problem. The algorithm creates decision trees to represent classification rules. The data input to C4.5 system forms a set of examples, each labelled according to the class to which it belongs. On the basis of this data, a decision tree is constructed [56]. Building a decision tree proceeds as follows. The set of all examples form the initial population. An attribute is chosen to split the population according to the attribute's values. For each sub-population, another attribute value is chosen to split the sub-population. This continues as long as each population contains a mix of examples belonging to different classes. Hence an intermediate node in a decision tree represents a test on a particular attribute. Once a uniform population has been obtained, a leaf node is created and labelled with the name of the class of the population. A leaf node therefore represents some form of final decision. When an object reaches a leaf node, it is classified according to the class of the leaf node.

The key to the success of a decision tree learning algorithm depends on the criterion used to select the attribute to use for splitting. If attribute is a strong indicator of an example's class value, it should appear as early in the tree as possible. Most decision tree learning algorithms use a heuristic for estimating the best attribute. In the C4.5 system, Quinlan uses a modified version of the entropy measure from information theory. For our purposes, it is sufficient to say that this measure yields a number between 0 and 1, where 0 indicates a

uniform population and 1 indicates a population where there is equal likelihood of all classes being present. The splitting criterion seeks to minimise the entropy.

A further refinement is required to handle noisy data. Real data sets often contain examples that are misclassified or which have incorrect attribute values. Suppose decision tree building has constructed a node which contains 99 examples from class 1 and only one example from class 2. According to the algorithm presented above, a further split would be required to separate the 99 from the one. However, the one exception may be mislabelled, causing an unnecessary split. Decision tree learning algorithms have a variety of methods for "pruning" unwanted subtrees. C4.5 system grows a complete tree, including nodes created as a result of noise. Following initial tree building, the program proceeds to select suspect subtrees and prunes then, testing the new tree on a data set which is separate from the initial training data. Pruning continues as long as the pruned trees yield more accurate classifications on the test data. When no further advantage is gained from pruning, it is stopped.

As illustrated in Figure 1, the decision tree generated from the training data, using the C4.5 system, is used to classify handwritten Chinese characters. A total of 900 different classes of characters were chosen for recognition and for each type of character, 40 samples were gathered as already described. The sample of 900 characters is sufficient for most practical problems because we have chosen the most common and used characters of Chinese Language. For each character, a total of 12 features were extracted, the first six primitives were labelled as variables V_1 ... V_6 (see section 3.1), and the next six stroke probability variables were labelled as F_1 ... F_6 for analysis purposes (see section 3.2). The variables V_1 ... V_6 represent the presence or absence of the primitives Horizontal ... Dots. The variables F_1 ... F_6 represent the stroke probability of these primitives in sequence. The stroke probability variables, which are continuous values, are statistically normalized to have a mean of zero and standard deviation of one, so that each variable contributed to the decision making process in statistical classification in a balanced manner. The data, in all, had a total of 36,000 patterns and a total of twelve measurements in each pattern.

The C4.5 system requires two input files, the *names* and the *data* files. The *name*s file contains the names of all the attributes used to describe the training examples and their allowed values. This file also contains the names of the possible classes. The classes are the characters of the Chinese alphabet which are numbered sequentially. The C4.5 system *data* file contains the attributes values, for example objects in the format specified by the name file, where each example is accompanied by its class description.

5. Experimental Results

Quinlan [54] suggests that the drawback associated with determining the performance of decision trees on a particular problem is that when they are constructed from small training samples, their decision making ability suffers.

On way around it is to use a cross-validation approach. Cross-validation ensures that the true performance of the classifier system is measured. The available cases are divided into C equal-sized blocks and, for each block, a tree is constructed from cases in all other blocks and tested on the 'holdout' block. For moderate values of C, the assumption is made that the tree constructed from all but one block will not differ much from the tree constructed from all data. In most cases, C= 10 is suggested [54, p. 90].

In our experiments, we have used cross-validation. In order to have a more rigorous approach to our experiment, we perform the cross-validation analysis on subsets of 100, 200, 500 characters and then the complete data set of 900 characters. The subsets of 100, 200 and 500 characters are chosen randomly. We expect that as the number of classes increase, the decision making process will become difficult for the C.4.5 algorithm.

We have conducted a total of six trials. Each trial represents a random pick of 100 characters from the 900 available. Since there are 40 samples for each character, a total of 4000 samples are available for our cross-validation testing. The samples are interleaved so that training and testing are performed on all classes. For separating training and test sets, we choose a value of C = 10 as recommended. Misclassification rate in percentage for all these ten folds is averaged. This average value for the first trial of 100 characters is shown in Table 1 which is 8.63% (recognition rate = 91.37%). The average cross-validation test error for other trials (other subsets of randomly picked 100 characters) is shown in the rest of the first column. Other columns in Table 1 show the same for six trials and ten fold cross-validation for 200 and 500 characters. The misclassification rate obtained for 900 characters is 15.98% (recognition rate = 84.02%). In Table 1, CI_{error} represents the confidence interval of the error estimate on the basis of errors obtained in different cross-validation folds. As expected, the recognition increases from small to large data-set and the confidence interval becomes smaller as we have more confidence in our decision with larger samples.

Table 1. Error rates in percentage for cross-validation test trials

Trial	100 char	CI_{error}	200 chars	CI_{error}	500 chars	CI_{error}	900 chars	CI_{error}
1	8.6	±0.9	10.2	±0.7	12.5	±0.5	16.0	±0.4
2	8.7	±0.9	10.4	±0.7	12.5	±0.5		
3	9.0	±0.9	12.3	±0.7	12.9	±0.5		
4	10.1	±0.9	13.0	±0.8	13.6	±0.5		
5	11.0	±1.0	13.3	±0.7	14.1	±0.5		
6	11.9	±1.0	14.9	±0.8	14.7	±0.5		
Avg.	9.9	±0.9	12.3	±0.8	13.4	±0.5	16.0	±0.4

The results in Table 1 are very encouraging. We find that the recognition rates are significantly high and relatively consistent across different trials. Of particular importance in these experiments, was whether the attributes used to describe the examples were sufficiently informative to be able to distinguish characters of different classes. These experiments indicate that the classification accuracy does not change significantly with the number of different characters and therefore, the attributes are reasonably robust across the entire sample.

6. Conclusion

This paper presents a new technique for recognizing hand printed Chinese characters and as indicated by the experiments performed, the machine learning C4.5 algorithm resulted in a 84.02% recognition rate. This result compares well with other analysis on the same data [53].1 In this paper, we first discussed pre-processing step whose aim was to extract line segments from the "*clean up*" image. Our post-processing procedure involves of finding the starting and ending position of the Freeman codes that contribute to the peak and then determine the best combination without any overlapping stroke. Moreover, the system used modified *Hough transform* algorithm for feature extraction. However, our approach is different from [13] in that we applied the transform to individual strokes rather than the whole picture and also we used a post-preprocessing procedure after the transformation to remove unwanted peaks.

A surprising outcome from these experiments is how well such a simple set of attributes has performed. This suggests that complex representations may not be necessary to achieve high accuracy. The aim of future experiments is to determine the cause of errors in the existing classifier. This may lead to suggestions for improved attributes that can lift the classification accuracy to higher levels.

Acknowledgement

The authors would like to extend their gratitude to Mr. Seung-Gwon Kim for helping out with the experimental results using C4.5 system.

References

[1] Harmon L. D, Automatic recognition of printed and script, Proceedings of the IEEE, 1972;60(10):1165–1177.

[2] Spanjersberg A. A, Experiments with automatic input of handwritten numerical data into a large administration system, IEEE Transactions on Man Systems and Cybernetics 1978;8(4):286–288.

[3] Focht L. R and Burger A, A numeric script recognition processor for postal zip code application, International Conference of Cybernetics and Society, 1976, pp. 486–492.

[4] Schuermann J, Reading Machines, 6th International Conference on Pattern Recognition, 1982, pp. 741–745.

[5] Plamondon R and Baron R, On-line recognition of handprinted schematic pseudocode for automatic Fortran code generator, 8th International Conference on Pattern Recognition, 1986, pp. 741–745.

[6] Amin A and Al-Fedaghi S, Machine recognition of printed Arabic text utilizing a natural language morphology, International Journal of Man-Machine Studies 1991;35(6):769–788.

[7] Guillevic D and Suen C. Y, Cursive script recognition:A fast reader, 2nd International Conference on Document Analysis and Recognition, 1993, pp. 311–314.

[8] Brown, M. K, and Ganapathy, S, Preprocessing technique for cursive script word recognition, Pattern Recognition, 1983;19(1):1–12.

[9] Davis R. H. and Lyall J, Recognition of handwritten character- A review, Image and Vision Computing, 1986; 4(4):208–218.

[10] Lecolinet E. and Baret O., Cursive word recognition: Methods and strategies, Fundamentals in Handwriting Recognition, Impedovo, S. (ed.) 1994, pp.235–263.

[11] Suen C. Y., Shingal R. and Kwan C. C., Dispersion factor: A quantitative measurement of the quality of handprinted characters, International Conference of Cybernetics and Society, 1977, pp. 681–685.

[12] Burr D. J., Designing a handwritten reader, 5th International Conference on Pattern Recognition, 1980, pp. 715-722.

[12] Shoukry A and Amin A, Topological and statistical analysis of line drawing, Pattern Recognition Letter 1, 1983, pp. 365–374.

[13] Amin A, Machine recognition of handwritten Arabic word by the IRAC II system, 6th International Conference on Pattern Recognition, 1982, pp. 34–36.

[14] Kim J. and Tappert C. C, Handwriting recognition accuracy versus tablet resolution and sampling rate, 7th International Conference on Pattern Recognition, 1984, pp.917–918.

[15] Ward J. R and Kuklinski T, A model for variability effects in handprinted with implication for the design of handwritten character recognition system, IEEE Transactions on Man Systems and Cybernetics 1988;18:438–451, 1988.

[16] Nouboud F and Plamondon R, On-line recognition of handprinted characters: Survey and beta tests, Pattern Recognition 1990;25(9):1031–1044.

[17] Ulmann J. R, Advance in character recognition, Application of Pattern Recognition, Fu, K. S. (ed.) 1982, pp. 197–236.

[18] Bokser M, OmnidocumentTechnologies, Proceedings of the IEEE, 1992;80(7): 1066–1078.

[19] Fujisawa H, Nakano Y and Kurino K, Segmentation methods for character recognition: From segmentation to document structure analysis, Proceedings of the IEEE, 1992;80(7):1079–1091.

[20] Srihari S, From pixel to paragraphs: The use of models in text recognition, Second Annual Symposium on Document Analysis and Information Retrieval, 1993, pp. 47-64.

[21] Chin R. T., On image analysis by the methods of moments, IEEE Transactions on Pattern Analysis and Machine Intelligence, 1988;10(4): 496–508, 1988.

[22] Raudys S. J. and Jain A. K., Small sample size effect in statistical pattern recognition, IEEE Transactions on Pattern Analysis and Machine Intelligence, 1991;13(3):252–264.

[23] Matsunage T. and Kida H., An experimental study of learning curves for statistical pattern classifiers, 3rd International Conference on Document Analysis and Recognition, 1995, pp.1103–1106.

[24] Berthod M. and Maroy J. P., Learning in syntactic recognition of symbols drawn on a graphic tablet, Computer Graphics and Image Processing, 1979;9:166–182.

[25] Wang P. S. P., and Gupta A., An improved structural approach for automated recognition of handprinted characters, International Journal of Pattern Recognition and Artificial Intelligence, 1991;5(1,2):97–121.

[26] Amin A. and Fotti A., Recognition of multifont Latin texts, 2nd Annual Symposium on Document Analysis and Information Retrieval, 1993, pp. 243–253.

[27] Fu K. S., Syntactic pattern recognition and application, Englewood Cliffs, N. J., Prentice-Hall, 1982.

[28] Tai J. W., A syntactic-semantic approach for Chinese character recognition, 7th International Conference on Pattern Recognition, 1984, pp. 374–376.

[29] Freund R., Syntatic analysis of handwritten characters by quasi-regular programmed array grammars, Advances in Structural and Syntactic Pattern Recognition, Bunke H. (ed.), 1992, pp. 310–319.

[30] LeCun Y., Backpropagation applied to handwritten zip code, Neural Computation, 1989;1:541–551.

[31] Guyon I., Application of neural networks to character recognition, Character and Handwriting Recognition in Expanding Frontiers, Wang P. S. P. (ed.), 1991, pp. 353–382.

[32] Amin A. and Wilson W. H., Hand-printed character recognition system using artificial neural networks, 2nd International Conference on Document Analysis and Recognition, 1993, pp. 943–946.

[33] Suen C. Y. and Yu C. L., Performance Accessmant of a Character Recognition Expert System, International Conference on Expert Systems Applications EXPERSYS'90, 1990, pp. 195–300.

[34] Likfooman-Solem, L., Maitre, H. and Sirait, C, An expert and vision system for analysis of Hebrew characters and autheutication of manuscripts, Pattern Recognition, 1991;24:121-137 .

[35] Augi, T. and Nagahashi, H. A description method of hand printed Chinese characters, IEEE Transactions on Pattern Analysis and Machine Intelligence, 1979; 1.

[36] Yamamoto, E. et al., Handwritten Kanji character recognition using the features extracted from multiple standpoints, Proceedings of the IEEE Conference on Pattern Recognition and Image Processing, 1981.

[37] Hsieh, C. C. and Lee, H. J. Off-line recognition of handwritten Chinese characters by on-line model-guided matching, Pattern Recognition, 1992;25:1337-1352.

[38] Rosenfeld, A. A characterization of parallel thinning algorithms, Information and Control, 1975;29:286-291.

[39] CArcelli, C., Cordella, L. P. and Levialdi, S. From local maxima to connected skeletons, IEEE Transactions on Pattern Analysis and Machine Intelligence, 1981;3:134-143.

[40] Pavlidis, T. A flexible parallel thinning algorithm, Proceerings of the Pattern Recognition and Image Processing Conference, 1981, pp. 162-167.

[41] Guo, Z. and Hall, R. W. Parallel thinning with two subiteration algorithm, Commun. ACM-32, no. 3, pp. 359-373 (1989).

[42] Hall, R. W. Fast parallel thinning algorithms: Parallel speed and connectivity preservation, Communications of the, 1989;32(3)124-131.

[43] Jolt, C. M., Stewart, A., Clint, M. and Perrott, R. H. An improved parallel thinning algorithm, Communications of the ACM, 1987;29(3)239-242.

[44] Chu, Y. K and Suen, C. Y. An alternate smoothing and stripping algorithm for thinning digital binary patterns, Signal Processing, 1986;11:207-222.

[45] Naccache, N. J., and Shinghal, R. SPTA: A proposed algorithm for thinning binary patterns, IEEE Transactions on Systems Man and Cybernetics, 1984;14: 409-418.

[46] Xia, Y. Skeletonization via the realization of the fire front's propagation and extinction in digital binary shapes, IEEE Transactions on Pattern Analysis and Machine Intelligence, 1989;11:1076-1086.

[47] B.K. Jang, B. K. and Chin, R. T. One-pass parallel thinning: analysis, properties, and quantitative evaluation, IEEE Transactions on Pattern Analysis on Machine Intelligence, 1992;14:1129-1140.

[48] Freeman, H. On the encoding of arbitrary geometric configurations, IEEE Transactions on Electronic Computers, 1968; 10:260-268.

[49]. Cheng, F. H. and Hse, W. H. Three stroke extraction methods for recognition of handwritten Chinese characters, Proceedings of the International. Conference on Chinese Computing, Singapore, 1986, pp. 191-195.

[50] Hough P. V. C., Method and means for recognizing complex patterns, U. S. patent number 3 069 654, 1962.

[51] Kushnir M, Abe K. and Matsumoto K, Recognition of Handprinted Hebrew Characters Using Features Selected in the Hough Transform Space, Pattern Recognition, 1985;18(2):103-114.

[52] F. H. Cheng, W. H. Hsu and M. Y. Chen, Recognition of handwritten Chinese characters by modified Hough Transform techniques, *IEEE. Trans. Pattern Analysis and Machine Intelligence*, Vol. 11, No, 4, pp. 429-439, (1989).

[53] Amin, A. and Singh, S. Machine Recognition of Hand-printed Chinese Characters, Intelligent Data Analysis - An International Journal, 1997;1(2).

[54] Quinlan J. R., C4.5 : Programs for machine learning, San Mateo CA, Morgan Kauffman, 1993.

[55] Quinlan J. R., Discovering rules for a large collection of examples, Edinburgh University Press, 1979.

[56] Breiman, L., Friedman, J., Olshen, R. and Stone, C. Classification and Regression Trees, Wadsworth International Group, 1984.