AUTOMATIC SEGMENTATION AND RECOGNITION SYSTEM FOR HANDWRITTEN DATES ON CHEQUES

Qizhi Xu

A thesis
in
The Department
of
Computer Science

Presented in Partial Fulfillment of the Requirements
For the Degree of Doctor of Philosophy
Concordia University
Montréal, Québec, Canada

December 2002

Abstract

Automatic Segmentation and Recognition System for Handwritten Dates on Cheques

Qizhi Xu, Ph.D.
Concordia University, 2003

The ability to recognize handwritten dates on bank cheques is very important in application environments where cheques cannot be processed prior to the dates shown on the cheques. This thesis presents the first automatic date processing system developed on a Canadian real-life standard cheque database. This system can process unconstrained handwritten dates written in English or in French, and it can also be applied to the recognition of any handwritten dates with similar format on many other kinds of documents.

Our date processing system aims at achieving high performances in terms of reliability and efficiency. To increase the efficiency of the system, a segmentation based strategy is adopted. The entire date zone is first divided into *Day*, *Month* and *Year* fields, so that different recognizers can be applied to recognize cursive words and numeral strings. In addition, the segmentation can be accomplished at several different levels depending on the quality of the handwriting, so that most of the simple cases are resolved quickly. To increase the reliability of the system, a knowledge-based module has been proposed for the date segmentation and a new cursive month word recognition system has also been implemented based on a combination of classifiers. The interaction between the segmentation and recognition stages has been properly established by using a multi-hypotheses generation and evaluation module. In addition, a verification module with two levels is designed in the postprocessing stage to correct some errors and reject invalid results, which further improves the reliability of the system. As a result, very promising performances with different reliability rates have been obtained in our date recognition system.

The segmentation of the date zone can be implemented in the knowledge-based segmentation module, the multi-hypotheses generation and evaluation module, or the verification module. These modules are invoked under different situations, and simple segmentation cases can be efficiently handled using only the knowledge-based segmentation module. In this knowledge-based module, knowledge about the writing style of the date is extracted, which is very useful for correct segmentation. An effective neural network ensemble system is proposed in this knowledge extraction stage to differentiate handwritten alphabetic words from numeric strings (A/N). In this approach, we investigate the use of effective features extensively, and propose several new methods in the design of neural networks, creation of neural network ensembles, and combination methods for the ensembles created. These new methods can also be applied to solve general pattern recognition problems.

For date recognition, the new cursive month word recognizer is implemented by combining a Hidden Markov Model classifier (HMM) with two Multi-Layer Perceptron (MLP) classifiers. An effective conditional combination topology and a newly modified Product combination rule have been proposed and implemented in this combination system.

Acknowledgments

First and foremost I would like to express my sincere gratitude to my supervisors Dr. Ching Y. Suen and Dr. Louisa Lam for their guidance, encouragement and assistance at so many levels throughout my study at Concordia. I also thank them for suggesting this topic and for carefully reviewing this thesis. It is their supervision and support that have made this work possible.

My great appreciation goes to those people whose work has directly contributed to this thesis. I would like to express my appreciation to Rong Fan, Ke Liu and Nick Strathy who have not only provided me with some source codes and libraries related to the project but also shared with me their knowledge and research experience in the field. I am also grateful to Christine Nadal and Guiling Guo for their help in building the database.

Special thanks go to Dr. Mary Ye, Dr. Xuejing Wu, Jianxiong Dong and Dr. Yuntao Qian for their fruitful discussions with me and their friendship.

I would like to thank all the friends at CENPARMI for their help and friendship in the past several years: Beverley, Jie, Xiaoping, Jun, Danny, Liqiang, Hao, Ping, Rita, Andrea, Yousef, Nicola, Boulos, Karim, Javad, and Michelle, etc. I also shared nice memory with visitors to CENPARMI, in particular with Dr. K. Kim, Dr. J. Kim, Dr. Zheng Lou and Dr. Li Feng.

Finally, I am indebted to my parents, my husband Hua Han, and my sister who give me endless love and support. Their care, understanding, and encouragement enabled me to complete this thesis.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Optical character recognition (OCR) is the process of converting scanned images of machine printed or handwritten text (numerals, letters and symbols), into a computer readable code (such as ASCII code). Research on OCR began in the 1950s, and it is one of the oldest research areas in the field of pattern recognition. Nowadays, many commercial systems are available for reliably processing cleanly machine-printed text documents with simple layouts, and some successful systems have also been developed to recognize handwritten texts, particularly isolated handprinted characters and words. However, the analysis of documents with complex layouts, recognition of degraded machine-printed texts, and the recognition of unconstrained handwritten texts continue to require improvements through research. This thesis aims at developing an automatic recognition system for unconstrained handwritten dates on bank cheques, which is a very challenging topic in OCR. The system proposed in this thesis can also be applied to the recognition of any handwritten dates with similar format on many other kinds of documents, and methodologies presented can also be applied to OCR systems of wider scope.

1.1 The Motivation

Written documents had been developed a long time ago as a means to extend human memory and to facilitate communication. Today, as computer and related information technologies grow rapidly, speculation arises on whether paper documents will eventually disappear [114]. However, paper is still and will continue to be one of the most commonly used media for saving and transporting people's ideas. Paper-based documents have advantages as they are considered to be secure, confidential, highly portable and the most convenient means for numerous day-to-day situations. Therefore, it is fair to conclude that paper-based documents will continue to play an important role in people's daily lives.

As long as paper documents are used there will be a need for machine recognition of text on paper. The objective is to use machines to process written (either handwritten or machine-printed) materials so that human beings can be relieved from the tedious manual work. Many applications can be found in this area, e.g. financial document (cheques, credit card slips, etc.) processing, postal address recognition for mail sorting, form (tax forms, census forms, etc.) processing, and establishment of digital libraries and other web-based OCR services, etc.

OCR addresses the above need and has been investigated for many years [135, 6]. Nowadays, OCR systems for the recognition of machine-printed text or isolated units of handwriting (such as a character or a word) have been extensively developed and have achieved over 90% recognition rates for restricted applications [8, 36, 80]. However, recognition of handwritten phrases has not received the same attention. This is the case even though OCR systems often have to read handwritten texts consisting of phrases for various applications. One such application is the automatic processing of date zones on cheques, which is the focus of our research.

A strong demand exists for cheque processing in North America and elsewhere. According to a survey, 83% of Americans still favor the cheque as the most convenient form of monthly bill payment [15] and over 55 billion cheques are processed annually in North America at a cost of 25 billion dollars [139]. Therefore, as an important commercial application and a challenging problem in OCR applications, automatic

bank cheque information processing has become a popular topic of research in recent years. Many advanced approaches have been developed in the areas of bank cheque information extraction, courtesy amount and legal amount recognition, and signature verification [60]. However, few research reports have been published on the processing of date information on cheques [57, 33, 105]. This is the case even though the ability to recognize the date information handwritten on bank cheques is very important in application environments where cheques cannot be processed prior to the dates shown, e.g. in Canada, where it is illegal to process post-dated cheques. At the same time, date information also appears on many other kinds of forms. So there is a great demand to develop reliable automatic date processing systems.

1.2 The Challenge

The processing of date zones on cheques is extremely challenging due to the high degree of variability and uncertainty of the data. As shown in Figure 1, the date fields can contain either only numerals or a mixture of alphabetic letters (for *Month*) and numerals (for *Day* and *Year*), and different recognizers are needed to process the information. *Month* can be written either before or after *Day*. Punctuations (period ('.'), comma (','), slash ('/') and hyphen ('-')) can be used to identify the end of one field. In addition, suffixes, such as "th", "st", "nd" and "rd" are written after *Day* by some users, and article "Le" sometimes appears at the beginning of date zone when cheques are written in French (the dates can be written in French or in English in the province of Quebec in Canada).

The challenge also comes from the immense variety in styles of handwriting. Figure 2 shows that many handwriting styles can be used to indicate a date even when the same format is used. Since such variety is unpredictable, it is very difficult to analyze and interpret human handwriting using exact mathematical methods. Furthermore, sometimes "poor" handwriting containing touching and overlapping characters and words, may appear.

Finally, as used in the everyday world, cheques are often composed of complicated

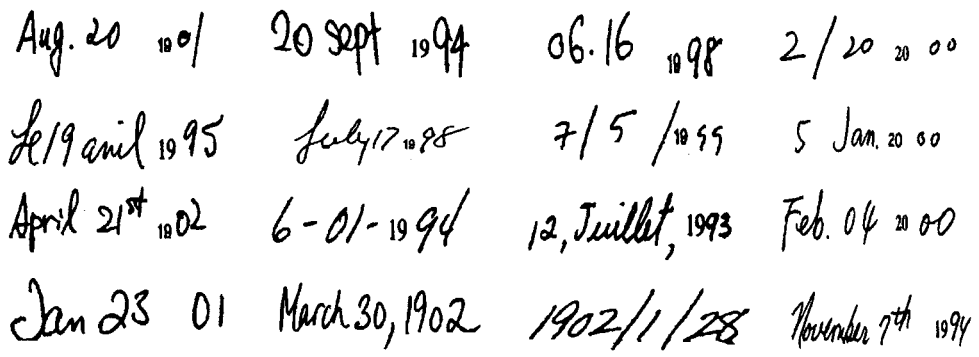

Figure 1: Sample dates handwritten on standard Canadian bank cheques

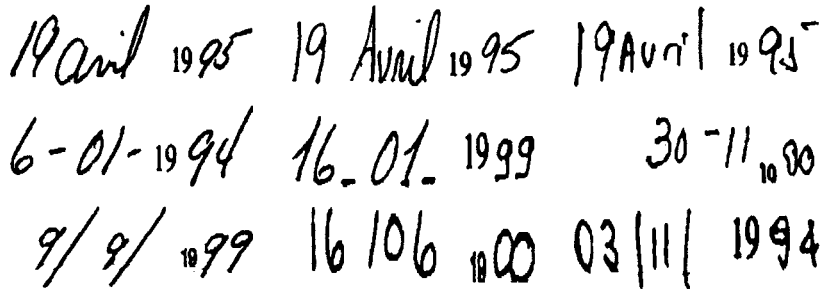

Figure 2: Sample dates with various handwriting styles

backgrounds with a variety of colorful scene images, which cause additional problems in image preprocessing. The noise introduced by improper binarization has a great impact on the date image analyses to follow; e.g. it is difficult to distinguish actual small punctuation marks from noise. In addition, meeting the demanding industrial requirements such as high processing speed, robustness and extremely low error rates, poses a big challenge in the development of automatic date processing systems.

1.3 The Proposal

The primary work in this thesis aims at developing an effective automatic processing system for processing date zones on bank cheques. The system analyzes and interprets the date images, and outputs the equivalent ASCII characters including *Day*, *Month* and *Year*.

Driven by the inherent goal of using this date processing module in real applications, the focus of our work is to achieve high performances in terms of reliability and efficiency. To increase the efficiency of the system, a segmentation based strategy is adopted. The entire date zone is first divided into *Day*, *Month* and *Year* parts, so that different recognizers can be applied to recognize cursive words and numeral strings, respectively. The segmentation can be accomplished at several different levels depending on the quality of the handwriting, so that most simple cases can be solved quickly. To tackle the problem of reliability, a knowledge-based segmentation module and a multi-hypotheses generation and evaluation approach are presented to improve the performance of the date segmentation. An effective recognition module for cursive month word recognition is also proposed, which combines a Hidden Markov Model classifier (HMM) with two Multi-Layer Perceptron classifiers. In addition, a verification module with two levels is designed in the post-processing stage to correct some errors and reject invalid results, which further improves the reliability of the system.

Although the objective of this thesis is to develop an application system for the automatic recognition of handwritten dates on bank cheques, some proposed methodologies and algorithms can be applied to OCR systems of wider scope and even general pattern recognition problems. This is another objective of this thesis.

1.4 Outline of the Thesis

The first chapter outlined the motivations, challenges and goals of the thesis. A review of the state of the art for off-line handwriting recognition, automatic cheque processing and date recognition is given in Chapter 2.

In Chapter 3, we present the basic architecture of our date processing system.

Some analyses about the writing styles of the date zones on bank cheque will also be given in this chapter because our system is designed based on the knowledge about the writing styles.

Chapter 4 presents two approaches for differentiating between alphabetic data and numeric data (A/N differentiation), together with the experiments conducted and the results obtained. This differentiation module is applied extensively in our date processing system.

Two main processing modules of our system, a date segmentation and a date recognition module, will be addressed in Chapters 5 and 6, respectively. The implementation of the entire system, including the preprocessing and verification modules, will be presented in Chapter 7, together with experimental results for the entire system. Finally, the thesis concludes with Chapter 8 which summarizes the main contributions of this thesis and directions for future work.

Chapter 2

State of the Art

In the field of handwriting recognition, handwritten data is first converted to digital form either by a scanner or by a special pen on an electronic surface which also records the timing information. These two input approaches distinguish the corresponding recognition systems as off-line and on-line systems, respectively.

Automatic processing of handwritten dates on bank cheques represents a typical application in the field of off-line handwriting recognition. In general, there are five major stages in off-line handwriting recognition systems: preprocessing, segmentation, feature extraction, training and recognition, and postprocessing. In our processing system, handwritten dates on bank cheques are processed through all of these stages. The approaches utilized in segmentation, recognition, and postprocessing stages (the main stages of our date processing system) will be briefly reviewed in the first section. This includes some knowledge-based systems for processing off-line handwriting, as we have used knowledge in our system to improve its performance. At the end of this section, a survey of the current systems for numeral string recognition and cursive word recognition is presented.

Naturally, a bank cheque processing system has its own characteristics compared to general handwriting recognition systems. A bank cheque processing system must be able to process handwritten data of many styles and poor quality with very high reliability and efficiency. In this challenging field of handwriting recognition, contextual

information and limited vocabularies are usually used to achieve acceptable recognition performance. So the second section will give a survey of automatic bank cheque processing, and the last section will focus on research on date image processing.

2.1 Off-line Handwriting Recognition (HR)

2.1.1 Methodologies

This subsection reviews the methodologies commonly implemented in the segmentation, recognition and postprocessing stages of general handwriting recognition systems.

A. Segmentation

In the segmentation stage, the document is segmented into its subcomponents. In general, there are three types of segmentation, that is, line segmentation, external sentence-to-word segmentation and internal word-to-character segmentation. Segmentation is an important stage of HR because separation of lines, words, or characters directly affects the recognition rate of the script. Segmentation is also a focus of our research in developing a date processing system, which is based on separating a line of handwritten text (*Date*) into words (*Day*, *Month* and *Year*). Therefore, approaches implemented in separating a line of handwritten text into words, i.e. external sentence-to-word segmentation is reviewed below.

External Sentence-To-Word Segmentation

This is a procedure that separates a text line into words. Only a few approaches in the literature have dealt with this word segmentation issue:

- The first segmentation strategy is based on spatial distance clues [125, 98], that is, inter-word gaps and inter-character gaps are determined according to spatial distances between adjacent connected components. Therefore, the central subproblem in this method is the estimation of gaps between adjacent components.

The first and most straightforward estimation method computes the horizontal distance between the bounding boxes of adjacent components [125]. The second method uses run-lengths and Euclidean distances between connected components and heuristics [125], while the third method approximates the gaps between adjacent components by the distance between their convex hulls [98]. An experiment has been carried out in [98] to compare the performance of the three gap metrics. The results show the convex hull based metric [98] performs better than the best metric in [125].

This segmentation strategy consists of the following steps: (i) determine the connected components in the given line; (ii) compute the distance (or gap) between pairs of adjacent components; (iii) sort the gaps in descending order of magnitude; and (iv) classify the gaps into inter-word gaps and inter-character gaps according to appropriate thresholds. When some punctuation marks are used as word separators, punctuation detection is also incorporated into this word segmentation strategy. (In [125], a set of fuzzy features is developed to detect punctuation, where each feature describes a shape or location aspect of a connected component).

- The second segmentation strategy is knowledge-based. This kind of algorithm usually uses multiple contextual clues and human knowledge to enhance the performance of segmentation.

  Kim *et al.* [69, 72] proposed a neural network based approach. This method employs some of the clues that humans use and additional information such as author's writing style in terms of spacing. The style is captured by characterizing the variation of spacing between adjacent characters as a function of the corresponding characters themselves. Some human knowledge is also encoded into the segmentation scheme. The approach begins with locating the bounding boxes of character segments. Intervals and heights of the center lines of the

boxes are treated as key parameters and input to a neural network designed for locating word breaks. This word segmentation method is based on character segmentation and it will be particularly effective in situations where character segmentation is a necessary step for word recognition.

Hussein *et al.* [59] provided a pattern-directed feedback mechanism to improve the segmentor of the handwritten courtesy amount, and the same methodology can be used in word segmentation. This approach first splits the input binary image of the courtesy amount into a set of binary images that are probable primitives, and then a segmentation analysis module is invoked to verify the segmentation process using a knowledge base of courtesy amount syntax as well as information regarding the number of digits from the preliminary recognition of the legal amount.

- The third segmentation strategy focuses on hypotheses generation. The word segmentation algorithm puts forth many hypotheses, where each hypothesis consists of a possible segmentation of the line into distinct words. The individual hypotheses are then interpreted by higher-level modules that employ additional recognition and contextual information. Mahadevan and Srihari [99] present three techniques for generating multiple segmentation hypotheses. In [79], the output of the legal amount segmentation is a set of potential word cuts with probabilities. Here the probabilities are computed using Bayes' rule and two measures for the distance between adjacent graphemes.

Among the three word segmentation strategies, the first strategy is most straightforward. However, word segmentation methods based on spatial clues alone usually have limited success for two reasons. First, handwriting does not always adhere to the rule of placing larger gaps between words than between characters. Secondly, the perceived space between components cannot be easily estimated by a 1-dimensional metric. To overcome the drawbacks of the first method, knowledge-based segmentation algorithms are developed, which try to exploit the context within which they operate. Since errors made in the segmentation stage become compounded by the

time recognition is performed, the third segmentation strategy, hypotheses generation, is often used together with the first or the second strategy to recover from errors in word separation.

B. Recognition Methods

Over the years, many systems have been developed for handwriting recognition. These systems use the methodologies of pattern recognition to assign an unknown sample to a predefined class. In the following discussion, general recognition techniques and their applications to word recognition will be reported first. The recognition of word phrases will also be briefly reviewed since date processing belongs to this domain.

(1) General Recognition Techniques

Numerous techniques for handwriting recognition have been investigated based on four general approaches of pattern recognition, as suggested in [61]: (i) template matching; (ii) statistical techniques; (iii) structural techniques; and (iv) neural networks. A new technique, support vector machine, which has been used in handwriting recognition in the past few years, will also be reviewed below.

- Template Matching: Generally speaking, matching operations determine the degree of similarity between two vectors (groups of pixels, shapes, curvatures, etc.) in the feature space. Matching techniques can be grouped into three classes: (i) direct matching [14, 39]; (ii) deformable templates and elastic matching [62, 6]; and (iii) relaxation matching [150].

- Statistical Techniques: Statistic decision theory is concerned with statistical decision functions and a set of optimality criteria, which determine the probability of the observed pattern belonging to a certain class. Several popular handwriting recognition approaches belong to this domain:

    1. The k-nearest-neighbor (k-NN) rule is a popular non-parametric recognition method, in which the *a-posteriori* probability is estimated from the

frequency of nearest neighbors of the unknown pattern. As a special case of k-NN rule, the nearest-neighbor rule classifies the unknown pattern to the class of the nearest sample. Good recognition results have been reported in [42] by using this approach for handwriting recognition. In addition, a comparison of fast nearest neighbor classifiers for handwriting recognition is given in [102].

2. The Bayesian classifier assigns a pattern to a class with the maximum *a-posteriori* probability. Class prototypes are used in the training stage to estimate the class-conditional probability density function for a feature vector [28].

3. The polynomial discriminant classifier [124] assigns a pattern to a class with the maximum discriminant value which is computed by a polynomial on the components of a feature vector. The class models are implicitly represented by the coefficients in the polynomial.

4. Hidden Markov Model (HMM) is one of the most widely and successfully used techniques for handwriting recognition problems. It is defined as a stochastic process generated by two interrelated mechanisms: a Markov Chain having a finite number of states and a set of random functions, each of which is associated with a state [116]. There are two basic approaches for handwriting recognition systems using HMM [12]. One approach builds one or more HMMs for each class of pattern. Given an observed pattern, we calculate the path probability against each model, and assign it to the class whose model leads to the maximum path probability. This classical approach is called model discriminant HMM (MD-HMM) [45]. Another approach is the Path-Discriminant HMM (PD-HMM) method, in which a single HMM is constructed for the whole language or context, and recognition consists of estimation of the optimal path for each class using a Viterbi algorithm based on dynamic programming [17].

5. Fuzzy set reasoning is a technique which employs fuzzy set elements to

describe the similarities between the features of the characters. An off-line handwriting recognition system is proposed in [1] using a fuzzy graph theoretic approach, where each character is described by a fuzzy graph. A fuzzy graph-matching algorithm is then used for recognition. In [38], a fuzzy rule-based system is defined for locating street number fields using uncertain information provided by image processing. In addition, a linguistic fuzzy recognizer of handwritten characters is proposed in [89].

- Structural Techniques: In structural handwriting recognition, the characters are represented as unions of structural primitives. It is assumed that the character primitives extracted from handwriting are quantifiable, and one can find the relations among them. There are two general structural methods: (i) the grammatical method [128]; and (ii) the graphical method [73].

- Artificial Neural Networks (NN): A NN is defined as a computing structure consisting of a massively parallel interconnection of adaptive "neural" processors. The primary advantages of NNs are (i) their ability to be trained automatically from examples, (ii) good performance with noisy data and (iii) possible parallel implementation. NNs have been widely used in handwriting recognition problems and have achieved promising results. NNs that are usually used in handwriting recognition include multi-layer perceptron networks (MLP), Kohonen's Self-Organizing Maps (SOM) classifiers, and convolutional networks.

    1. Multilayer Perceptrons: MLP was proposed by Rosenblatt [9] and elaborated by Minsky and Papert [103]. The perceptron, as the building block of MLP network, forms a weighted sum of n components of the input vector and adds a bias value. The results are then passed through a nonlinear activation function. MLP networks trained by back propagation [122] are among the most popular and versatile forms of neural network classifiers and are also among the most frequently used traditional classifiers for handwriting recognition [83, 92]. A recent study in [115] incorporates the

properties of MLPs and morphological/rank neural networks for handwriting recognition.

2. Self-Organizing Maps: Kohonen's SOM [82] integrates the feature extraction and recognition steps in a large training set of characters. It can be shown that it is analogous to k-means clustering algorithm. Many recent developments on handwriting recognition research are concentrated on SOMs. An example is the study in [121], where a neural network which is a combination of modified self-organizing map and learning vector quantization is proposed for handwritten numeral recognition. Another system which realized the adaptive-subspace SOM to build a modular classification system for handwritten digit recognition is presented in [156].

3. Convolutional Networks: Convolutional networks combine three architectural ideas to ensure some degree of shift, scale and distortion invariance. These three ideas are local receptive fields, weight replications, and spatial or temporal sub-sampling. A typical convolutional network for handwriting recognition is LeNet [23], which is one of the best classifiers for handwritten digit recognition.

In addition, some other types of neural networks have been proposed for handwriting recognition including radial basis function (RBF) networks [18] and Quantum neural network [158].

- Support Vector Machine (SVM): In the past few years, SVMs [13] have received increasing attention in the community of machine learning due to its excellent generalization performance. SVM is based on the statistical learning theory [144] and quadratic programming optimization. An SVM is basically a binary classifier and multiple SVMs can be combined to form a classification system for multi-class classifiction. Some SVM classification systems have been developed for handwritten digit recognition in recent years, and some promising results have been reported [93, 27].

(2) Combinations

In the past decade, many researchers have employed various methodologies to combine decisions of multiple classifiers in order to improve recognition results. Some good recent surveys on this topic are [117, 140, 84, 77]. Many combination techniques can be grouped and analyzed in different ways. On a theoretical level, one can consider that there are two classifier combination scenarios [84]: all classifiers use the same representation of the input pattern, or different representations. In the former case, each classifier can be considered to produce an estimation of the same *a-posteriori* class probability. In terms of implementation, combination approaches can be classified according to the architectures used. These are: (i) conditional, (ii) serial (hierarchical), (iii) parallel (multiple), and (iv) hybrid, architectures [84, 117].

- Conditional Architecture: In this architecture, a primary classifier is first used. When it rejects a pattern due to inability to give a classification or the classification cannot satisfy a certain confidence level, a secondary classifier, which can use more complex procedures, is adopted. Some examples of using this architecture can be found in [18, 87]. The main advantage of this architecture is computational efficiency because usually the primary classifier is a fast one and it processes most of the easily recognizable patterns.

- Serial Architecture: Using this architecture, classifiers are applied in succession. The goal of each classifier is to reduce the number of classes to which the unknown pattern may belong, so that the individual classifiers can become increasingly focused. Examples of this approach are given in [142, 119].

- Parallel Architecture: In this architecture, the classifiers are first applied concurrently and independently. The decisions of the classifiers are then combined to yield a final decision. This architecture allows for development and introduction of new classifiers without requiring major modifications to the fusion process, and by combining individual classifier outputs, this architecture produces a more accurate system as a whole. However, this architecture incurs more computational costs since several classifiers have to be operated first before the

final fusion.

The parallel combinations can be implemented using different strategies, and the combination methods can be analyzed and categorized in different ways:

1. Depending on the type of individual classifier outputs, combination methods can be implemented on different output levels: abstract level or single class output level, ranked list of classes, and measurement level [140].
2. Based on the implicit decision emphasis, the combination approaches can be divided into three broad categories [120]. The first category consists of methods implementing computationally intensive decision frameworks incorporating *a-priori* information about the target task domain and the reliability of the participating classifiers. The second category encompasses approaches implementing group consensus without assigning any importance to the reliability of the classifiers and ignoring other contextual information. The third category is a hybrid decision fusion method of the above two approaches.
3. Combination methodologies can also be simple operators, such as *Max*, *Min*, *Sum*, *Product*, and *Majority vote* [77], Bayesian [63], Dempster-Shafer (D-S) [151], Behavior-knowledge space [58], and pooled ranking figure of merit [40]. In addition, measurement outputs of classifiers can be combined through the use of neural networks [81].

- Hybrid Architecture: The main idea of this architecture is to combine the power of previous architectures. In [118], a hybrid approach combining the serial and parallel architectures was proposed.

More discussions about combination methods will be given in Chapters 4 and 6.

(3) Cursive Script Word Recognition

Only a few systems have been implemented for off-line recognition of unconstrained, cursive and handwritten text [30]. However, many more have been developed

to recognize handprinting or for handwriting recognition in restricted domains such as mail address recognition [20] and legal amount recognition [45] where the limited lexicon facilitates the task.

In the field of word recognition, three different types of approaches have been considered: analytic (segmentation-based), holistic (segmentation-free), and perception-oriented [132].

- Analytic approaches typically apply some techniques to segment the word into basic components and try to recognize the whole word on the basis of the recognition of these components. Most successful analytic approaches are segmentation-recognition methods in which words are first segmented into characters or parts of characters, and then the recognition process is based on an attempt to find the best complete match between blocks of primitive segments and the letters of a word. The best match can be implemented in different ways. Three main best match methods are the following:

  1. The best match is determined by a straightforward dynamic programming algorithm [21]. Dynamic programming is executed for each word separately and the word associated with the overall best matching score is selected. Some good examples of using this approach for cursive script word recognition are given in [69, 129].
  2. The best match can also be realized by finding the shortest path in a special graph in which each node represents a pair of possible blocks of primitive segments and letter interpretations, associated with their matching score as a weight. The Viterbi algorithm can be used to compute the best path required. Two recent examples using this shortest path method are [34, 16].
  3. Another alternative method is utilizing segmentation-based HMMs in which the pair of segmentation and interpretation is associated with the underlying states of the optimal path. The Viterbi algorithm can be used to find the optimal path. An example of this method is given in [31].

- The global approach recognizes a word without using a single component analysis. There is no attempt in this approach to split the word image into segments that relate to characters. Still, it is possible that the image would be split into pieces in order to produce a sequence of observations. Instead of a letter-by-letter recognition, this approach processes a word as a whole by searching for a word with the complete description most similar to the one obtained from the whole input image. Many methods have been proposed for comparisons between a pair of observation sequences, and most methods use either a minimum edit-distance calculation based on dynamic-programming [145, 111], or a resemblance estimation provided by segmentation-free HMMs [43, 104].

  Based on the discussions above, some conclusions can be drawn. Analytic approaches are generally sensitive to writing style and quality and they are strongly dependent on the effectiveness of the segmentation procedure. However, they have the advantage of being generalizable to large or limited but dynamic vocabulary with limited training. Global approaches are considered to be closer to the human way of reading and more effective for poor handwriting, but they are limited in their discrimination capability and usually seem to be suitable only for limited and static vocabulary applications. Actually some methods try to combine the advantages of both approaches [52].

- Unlike general sequential recognition methods which attempt to match an ordered list of observations/primitive segments with a word pattern from left to right, perception-oriented methods operate in a bottom-up manner to identify letters anywhere in the observation sequence derived from a word image. Next, a decision procedure is run, resulting in an interpretation which consists of the best non-overlapping set of letters, and the most likely interpretation of possible gaps left between them. This method seems to resemble the human reading scheme. Some examples using this kind of method are given in [30, 22].

(4) Phrase Recognition

While recognition of isolated units of writing, such as a character or a word, has been extensively studied in the literature, research on phrase recognition has been limited. This is the case even though in many applications, such as postal address recognition and bank cheque processing, phrase recognition plays a very important role. In the following, we summarize research on this topic and categorize it into two types: segmentation-based and segmentation-free methods.

- Segmentation-Based Method: In this method, there is an attempt to segment the phrase into its constituent words by applying various sentence-to-word segmentation followed by recognition of the segments. This method can also be implemented by examining a phrase and breaking it into a number of hypotheses which represent likely word segmentations of the sentence [99]. Recognition is then done on each candidate word of each sentence hypothesis. The results of recognition and linguistic analysis can be used to rank each sentence hypothesis for correctness. In [69], the word break point detector is first implemented by a neural network, and then an exhaustive combination of word segments is submitted to a word recognizer with a given lexicon and statistical characteristics of character segmentation are used to limit the number of combinations. Further improvement of this method was given in [112].

- Segmentation-Free Method: This method tries to recognize the entire phrase as a single unit. In [71], a street name recognition system is proposed which ignores spaces between words in the input street name images. A single word recognizer is modified to handle this phrase recognition problem. The modifications include labeling components of lexicon entries during expansion, tracing the implicit segmentation points while the match is being performed and computing the matching confidence of each component, and making decisions by different rules based on matching confidences. Another phrase recognition method without word segmentation is given in [123], in which a system for recognizing legal amounts on Italian cheques was developed. The author of this paper showed that when the *a-priori* knowledge was described by a regular grammar, the phrase recognition can be performed by a simple extension to the dynamic

programming algorithms usually adopted for isolated word recognition. In a recent research [101], a system for reading unconstrained handwritten text is presented. The core of the recognition procedure is a HMM. It receives a sequence of feature vectors as input and outputs a sequence of words, applying constraints from the language model. The feature vectors input to the HMM are extracted from a complete line of text. Thus segmentation of a line of text into individual words is not required; instead, the segmentation is obtained during the recognition process.

C. Postprocessing

A postprocessing stage is usually used in word level or phrase (or sentence) level recognition. This process relates to lexicon lookup, string correction, and re-evaluation of a word or phrase probability with respect to syntax and context, etc. Some examples can be found in string matching processes that use statistical information derived from the training data and the syntactic knowledge, e.g. systems using N-grams [67, 46], a lexicon driven recognition system that represents a word image by a segmentation graph and matches each entry in the lexicon against this graph [68], and a phrase recognition system that uses linguistic constraints to intelligently parse and recognize text [72].

Besides the approaches mentioned above, other attempts have also been made in the postprocessing stage to enhance the performance of recognition systems. One new technique is to apply a verification module after the general purpose recognizer [159, 160]. The main goal of the verification module is to improve the reliability of the system, which is a very important requirement in many applications such as automatic bank cheque processing systems.

2.1.2 Knowledge-Based Processing System

Handwritten information processing is a rather easy task for humans, but still remains very difficult using automatic algorithmic methods. One way to bridge this gap is to incorporate knowledge derived from human experts, contextual and specific

application requirements (syntactic and semantic constraints, etc.). Such knowledge can be acquired in the training phase and then applied to process unknown samples.

In order to build a knowledge-based system, the knowledge should be first acquired, then encoded properly, and finally built into the system. Several knowledge-based handwriting recognition systems have been developed to improve machine recognition ability. In the previous reviews, we have given two examples of knowledge-based sentence-to-word segmentation. More knowledge-based systems will be discussed below:

- A knowledge-based approach is proposed in [100] to recognize unconstrained handwritten numerals. In the training stage, data on features observed in a training set are stored in two knowledge bases of features and pattern classes. In the recognition stage, an inference engine compares the features of an unknown sample with the knowledge base of features and infers the most probable hypothesis on the identity of the sample. An alternative method uses a structural classifier which matches the results of feature extraction against the knowledge base of pattern classes. The final recognition decision in both methods is obtained on the basis of Bayes' Rule.

- An expert system for the analysis and recognition of general symbols is introduced in [3]. The system uses a structural pattern recognition technique that models symbols by a set of straight lines referred to as segments. The system is capable of learning new symbols by simply adding their models to the system knowledge base.

- A system for exploiting context in automatic textual information recognition (particularly in a form) is presented in [96]. In order to improve the recognition process, the system has been designed to make use of application-dependent constraints. The crux of the proposal is a Document Specification Language (DSL). DSL programs specify the sequences of tokens that form an acceptable solution and the constraints that have to be satisfied by the recognized information. At execution time, the system parses and interprets the input according

to specifications and systematically tries various alternatives until it finds the best (or near-best) solution.

- The objective of [106] is to apply human knowledge to improve machine recognition of confusing handwritten numerals. To help improve machine performance on confusing samples, comments collected from an experiment conducted with a group of human experts specialized in unconstrained handwritten numeral recognition are analyzed. Based on this knowledge, a tool was built which managed the information and gave results of analyses to help extract knowledge on confusing styles of writing and regions of ambiguity. These analyses will facilitate the design of new specialized modules aimed at differentiating numerals belonging to the same confusing pair.

- In [148], the authors attempt to inject human knowledge into the rejection criterion of a neural network classifier for recognizing isolated handwritten numerals. Knowledge from five human experts is gathered and analyzed. Based on the knowledge, a new way to construct databases and represent the required output values in the output layer of MLP's training process is given to improve the reliability of a single neural network classifier on confusing numerals.

- A knowledge-based cursive script segmentation method is proposed in [149]. Knowledge concerning the structures of some English letters, as well as the structural characteristics between background regions and characters is investigated as a novel approach to cursive script segmentation. A set of rules are derived based on the knowledge to guide the separation for more accurate segmentation results.

2.1.3 Survey of Current Recognition Systems

In this section, the best published performances of current systems in the field of off-line handwriting recognition are briefly summarized. We will focus on numeral string recognition and cursive script word recognition since these topics are closely

related to our date processing system.

A. Numeral String Recognition

Many systems have been developed to achieve accurate recognition of handwritten numerals and many experiments have been conducted on the CENPARMI, CEDAR and NIST databases [160]. Nowadays systems of recognition of isolated handwritten numerals have achieved more than 99% recognition rates [47, 24, 133] on CEDAR *goodbs* database. In comparison, the literature indicates that the recognition of handwritten numeral strings is less studied and more difficult due to complicated cases of touching characters and more diversified writing styles. However, this problem has received increasing attention from researchers because of its important role in practical applications. Table 1 shows a comparison of recognition performances for numeral strings on test sets from CEDAR and NIST databases.

Table 1: Performance of numeral string recognition based on CEDAR and NIST databases

Authors	Database	Samples	Error	Correct	Date
Shi & Govindaraju [127]	CEDAR BU (numeral pair)	1089	19.2%	80.8%	1997
Wang et al. [147]	CEDAR BHA (numeral pair)	628	16.5%	83.5%	2000
Ha et al. [48]	CEDAR test/binzips/bs	495	16.4%	83.6%	1998
Ha et al. [48]	CEDAR test/binzips/bs (5-digit)	436	15.1%	84.9%	1998
Liu et al. [94]	CEDAR test/binzips/bs (5-digit)	436	9.6%	90.4%	2002
Ha et al. [48]	NIST SD3	4925	7.3%	92.7%	1998
Lu et al. [97]	NIST SD3 (numeral pair)	3355	7.1%	88.2%	1999
Britto Jr et al. [11]	NIST SD19 (numeral pair)	2370	14.7%	85.3%	2000
Zhou et al. [159]	NIST SD19 (numeral pair)	4395	3.5%	85.7%	2000
Suen et al. [136]	NIST SD19 (numeral pair)	3500	7.5%	92.5%	2000
Oliveira et al. [109]	NIST SD19 (numeral pair)	–	2.6%	97.4%	2001
Yu & Yan [155]	NIST (numeral pair)	3287	16.0%	84.0%	2001

Since results are obtained from different databases or different portions of the same database, different numbers of training and testing samples have been used,

and also different segmentation methods and recognition algorithms are adopted, any direct comparison of results in Table 1 would be rather difficult. In addition, the studies on the CEDAR database reported recognition rates of below or around 90% while those using the NIST database have produced better recognition rates because the CEDAR database contains more unconstrained data than NIST where the writers were requested to write numeral strings in preprinted boxes.

In real applications, the reliability needs to be very high, especially for cheque or other financial document processing systems. Table 2 shows some practical up-to-date system performances of numeral string recognition where error rates have been controlled to be lower than 1.0%. It should be noted that the data used in practical systems usually contain more noise, which also reduces the recognition rate.

Table 2: Performance of application systems in numeral string recognition

Systems	Application Domain	Error	Correct	Date
Mitek [57]	Cheques	1.0%	55.0%	1996
ParaScript [35]	5-digit zip codes	0.8%	64.0%	1998
CENPARMI [138]	Cheques and financial documents	1.0%	62.0%	1998
CENPARMI [157]	Cheques	1.0%	69.75%	2001

B. Cursive Word Recognition

The recognition of cursive words has been studied for several decades, and many interesting results have been reported [31, 70, 80]. However, we cannot provide a direct performance comparison because no standard database is available for cursive word recognition. Table 3 gives a performance comparison of legal word recognition based on several real cheque databases. Although the application domains are the same, it is still difficult to compare the results shown in the table because the experiments were conducted on different databases, different numbers of classes, and different numbers of training and testing samples, etc.

Table 3: Performance of legal word recognizers on real cheque databases

Classifiers	Database	Class	Error	Correct	Date
Han & Sethi [49]	English	-	15.1%	84.9%	1997
Knerr & Augustin [80]	French	30	7.1%	92.9%	1998
Guillevic & Suen [45]	French	30	13.3%	86.7%	1998
Côté et al. [22]	English	32	26.4%	73.6%	1998
Kaufmann & Bunke [64]	German	-	28.1%	71.9%	1998
Soan [130]	French	26	17.5%	82.5%	1999
Suen et al. [136]	English	32	7.3%	92.7%	2000

2.2 Automatic Cheque Processing

2.2.1 Overview of Cheque Processing Systems

In order to produce an effective cheque reading system, many problems have to be solved, e.g., background and noise removal; recognition of the immense variety of handwriting styles; touching and overlapping characters and words, and the presence of errors in the recognition process. Up to now, many solutions have been proposed by research groups in many countries (e.g. Canada [141], France [79, 53, 91], U.S.A [29, 68, 2], Italy [26], Switzerland[65], Brazil [90], China [134], Japan [108]). The main research areas in cheque processing systems include item extraction, courtesy amount recognition, legal amount recognition, date recognition and signature verification.

2.2.2 Survey of Available Systems

In this section, we review several successful cheque processing systems.

A. A2iA INTERCHEQUE System

This system [79, 80] is designed for the recognition of French, omni-bank, omni-scriptor, handwritten bank cheques, and it recognizes both the legal and the courtesy amounts on bank cheques. Two important ideas, the segmentation-recognition principle and the regularity-singularity principle, are used at several stages throughout the

system. The recognition process is divided into 5 steps: (i) extraction of the image parts containing the amount; (ii) preprocessing of the extracted amount images; (iii) segmentation of the courtesy amount into character candidates, and segmentation of the legal amount into character candidates and word candidates; (iv) recognition of characters, words and amounts; and (v) fusion and final decision based on both the legal and courtesy amount recognition. The system produces between 50% and 65% recognition rates for less than 0.1% substitution, depending on the importance of the centimes part on the cheques to be processed. This system is one of the most successful cheque processing systems up to now.

B. CEDAR Bankcheck Reading System

CEDAR designed a bank cheque reading system using cross validation of both the legal and the courtesy amounts [70]. This approach to cheque amount recognition differs from most traditional methods in two significant ways. First, emphasis on both the legal amount and the courtesy amount is placed evenly, while most previous approaches have largely focused on the courtesy amount. Secondly, the combination strategy for the cross validation of both the legal and the courtesy amounts is serial rather than the commonly used parallel methods. The recognition engine first processes the legal amount and outputs a list of hypotheses which are submitted to the courtesy amount recognizer as a lexicon. The courtesy amount is treated as a numeric string and the same word recognition scheme used for the legal amount is applied. Experimental results of this system show that 43.8% of cheque images are correctly read with an error rate of 0%.

C. Cheque Processing System with Error Detection and Correction Module

G. Kaufmann and H. Bunke describes a complete system for the automated reading of German amounts extracted from bank and postal cheques in [64, 65, 66]. This system introduces a new method for the comparison of legal and courtesy amounts. Based on the method of syntax-directed translation (SDT), the result of legal amount recognition is translated into a digit string that can be directly compared with the

result of courtesy amount recognition. SDT is a systematic, simple and flexible technique, which is especially useful for languages where the translation cannot be done in a straightforward way and many different word representations correspond to the same digit amount. The information provided by SDT in this system is used to infer the relation between each digit and the corresponding part of the word amount. Thus, inconsistencies between the legal and the courtesy amounts can be detected and precisely located at the level of individual subwords in the legal amount and digits in the courtesy amount. In this way, some recognition errors can be corrected. Experimental results of this system show that 68.8% of cheque images are correctly read with an error rate of 0.4%.

D. Engineered Automatic Bank Cheque Processing System

The paper [26] gives an example of a complete system for Italian bank cheque processing. The use of an advanced software engineering tool, Khoros software tool, is proposed for the system design and development. It allows the re-engineering of the system, in order to re-use the software and easily integrate new solutions. In this system, a full exploitation of the contextual knowledge, together with a multi-expert approach, have been used both to analyze the complex shape of handwritten text and to design the system. The main processing modules in this system include data acquisition, preprocessing, machine-printed numeral recognition, layout analysis, courtesy amount recognition, legal amount recognition, amount validation, and signature verification. No overall experiment results are reported in this paper.

E. Cheque Processing Systems Developed by ParaScript, Mitek and Orbograph

Since bank cheque processing is becoming one of the most promising commercial applications of handwriting recognition, some industrial companies have been involved in developing cheque processing systems. Three successful products for recognizing amounts on American personal cheques are reviewed here, and they are developed by ParaScript, Orbograph and Mitek, respectively.

The cheque amount recognition system developed in ParaScript is based on the

cross validation of courtesy and legal amounts [29]. The system has been tested on a test set of 5000 real cheques. At a 1% error rate level, the recognition rates of courtesy amount, legal amount and final amount have reached 47%, 18% and 67%, respectively. For Orbograph, the basic product for cheque processing is OrboCar (http://www.orbograph.com), which is a courtesy amount recognition system with high speed and high performance. OrboCar has achieved a read rate of over 75% for 0.1% error rate, and with the speed of up to 150,000 items per hour. In addition, the newest product of Orbograph, OrboCAR APEX, has been presented in 2001, and it has achieved much better results in cheque amount recognition compared with previous products. Lastly, in Mitek (http://www.miteksys.com), a product named CheckScript has been developed to automatically read legal amounts and courtesy amounts and cross-validate the information from these two sources to ensure accurate processing of bank cheques. Although the overall performance of this product is not mentioned on the website, a 55% recognition rate with 1.0% error rate for courtesy amount recognition has been reported for this product in [57].

F. CENPARMI Bank Cheque Processing System

A prototype reading system has been developed in CENPARMI to automatically process handwritten date, courtesy amount and legal amount on Canadian bank cheques. This system is an integration of the results of several researchers [141, 85, 95, 154, 76, 45, 133, 157, 161, 33, 153]. The overall system architecture can be described as shown in Figure 3. Some discussions about the item extraction, courtesy amount recognition and legal amount recognition will be given below. Since date recognition is the topic of this thesis, more detailed reviews of this aspect will be presented in the next section.

The first step of this system is to extract the handwritten items of information to be processed, i.e. to locate the items, to remove the baselines and bounding boxes, and to separate the user-entered items from the background and noise. Two methods [95, 154] have been developed for this step, and both of them are based on gray images and use a layout directed scheme. However, in [95], the method of extracting and eliminating baselines is based on edge images, and Hough Transform and

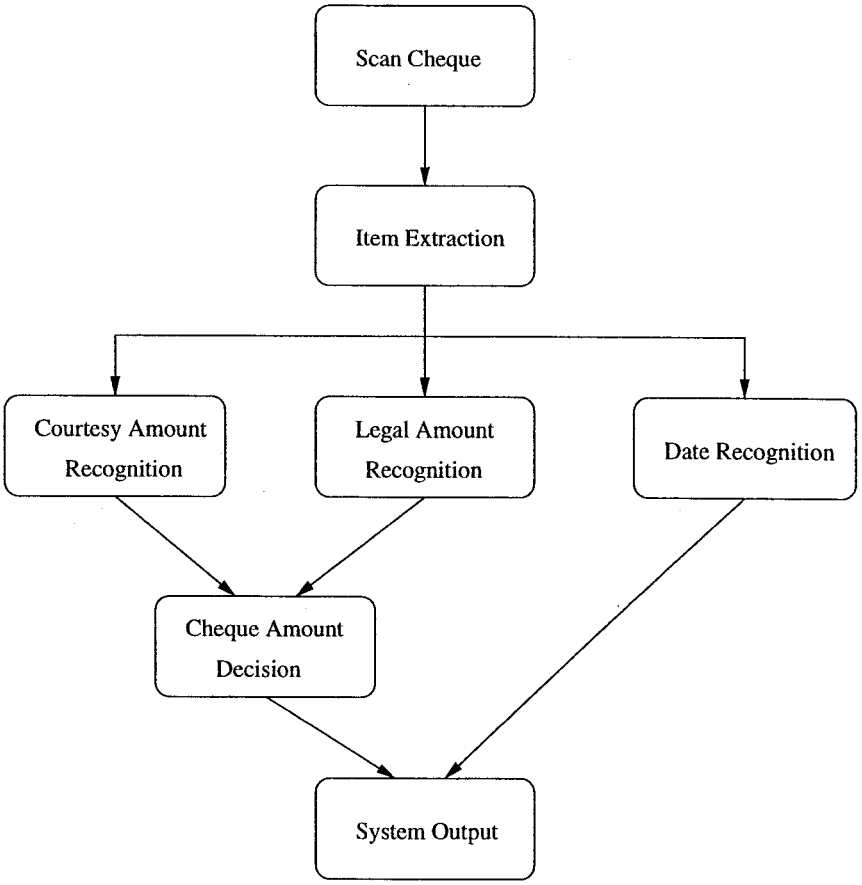

Figure 3: Bank cheque processing system developed by CENPARMI

Least Square Fitting algorithms are used. The method in [154] uses gray-level mathematical morphology to accomplish the same task. In addition, different thresholding techniques are used in these two methods to separate the user-entered items from the background. Experimental results of [154] show the method presented in [154] is superior to that of [95].

For both courtesy amount recognition and legal amount recognition in the system, segmentation-based methods are used, i.e. the whole item is first segmented into basic components which are recognized, after which the recognition of the whole

item is determined from the recognition of the components. Segmenting the courtesy amount into characters and segmenting the legal amount into words are realized by recognition-based approaches [141, 157, 161]. In the recognition stage, a classifier has been designed to recognize segmented numerals for courtesy amount recognition by combining three MLP neural networks [141, 133, 157], while three classifiers have been developed to recognize the legal words. For the latter, the three methods used are HMM-KNN combination method [44], HMM-MLP hybrid method [76] and MLPs combination method [161]. For the performances, 69.75% recognition rate (with less than 0.1% substitution rate) has been obtained for courtesy amount recognition [157], while 61.5% recognition rate (without rejection) has been obtained for legal amount recognition [161]. In addition, the results of courtesy and legal amount recognition can be combined in this system for the final amount recognition. Since the performance of the courtesy amount recognizer is better than that of the legal amount recognizer, the result of the legal amount recognizer can be used as a verifier for the courtesy amount recognizer. This combination module is being developed.

G. Summary

From the above system review, we can summarize several key aspects of current bank cheque processing systems:

- The basic function of cheque processing systems is amount recognition, and in order to obtain more reliable systems to meet the requirements of commercial applications, the most popular technique adopted in the systems is cross-validating the results from the legal and the courtesy amount recognition. In the cross-validation stage, the information from the two fields can be applied serially or in parallel.

- Many advanced methodologies have been used in developing automatic cheque processing systems, e.g. the use of advanced software engineering tools and the incorporation of HMM-MLP hybrid architecture, etc..

- The performances reported cannot be compared directly due to the different

databases used. However, practical systems have been developed and used in different banks in different countries, and acceptable results have been achieved.

2.3 Date Recognition

Even though the ability to automatically process handwritten dates is important in application environments where cheques cannot be cashed prior to the dates shown, very few papers have been published on this subject.

The date processing model presented in [57] was the first reported work on processing the date image of machine-printed cheques. The algorithm first identifies the bigrams '95', '96', '97', which are common to all date patterns (e.g. Jan 25, 1996, JANUARY 25 1996, 01/25/96, 01-25-96, etc.). It next examines the left neighboring characters to determine whether they are the completion of year (i.e. 1996), or a delimiter for the day or month. A tool shell has been designed which supports both word spotting through inexact matching, and the use of wild card characters to search for formatted patterns such as $**/**/**$, $**\_**\_**$, etc. A sample of 500 valid date fields were segmented by a human operator and tested. The complete recognition performance with the automated field segmentation is estimated at 44% to 49% by using four commercial recognition devices. This reference also considers date processing to be the greatest challenge in cheque processing, given that it has the worst segmentation and recognition performance.

In [105], a system is being developed for handwritten date recognition on Brazilian bank cheques. Instead of segmenting the date information into sub-fields using a set of rules based on structural features, an HMM-based approach was developed to perform segmentation in combination with the recognition process. The date sentence is represented by global Markov models formed by the concatenation of appropriate sub-field models according to the lexicon of this application. After the sub-fields have been identified, the system tries to recognize the different data types such as words and digits using different recognizers for processing. A neural network approach is used to recognize digits, while HMMs are used to recognize words. The final recognition

rate for date images on a small test set (400 samples) is 76.8% without rejection.

The previous work of the CENPARMI date processing system was reported in [33]. In this system, a solution is proposed, which segments the date image into *Day*, *Month* and *Year* fields first so that the problem can be reduced to the separate processing of cursive words and numerals. Figure 4 illustrates the block diagram of this recognition system.

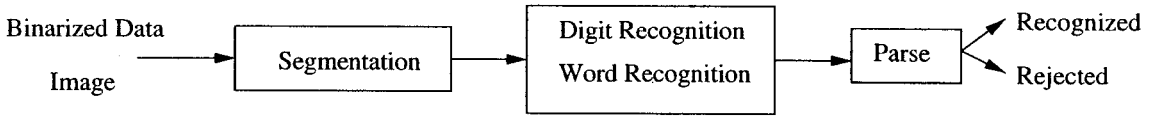

Figure 4: Previous recognition system for date zones on bank cheques

Although the results obtained have shown the effectiveness of this approach, the poor performance of the entire system (only 21.98% of 4564 date images from CENPARMI Cheque database were correctly recognized) suggests further studies would be needed. The main weaknesses of this system can be summarized as:

- The segmentation module is based on a bottom-up method, with no contextual analysis.
- Interaction between the segmentation and the recognition stages has not been properly established. No information in the recognition stage is used in the segmentation module.
- The digit and the word recognizers used are not specifically designed for date processing (the digit recognizer was developed for recognizing courtesy amounts, while the word recognizer was for legal amount recognition).

These weaknesses indicate the areas where further research would be needed, and they also provide the motivation for the work done in this thesis.

Chapter 3

System Overview

3.1 Writing Style Analysis

3.1.1 Variations of Handwritten Dates

In North America, all bank cheques have a similar layout (as shown in Figure 5). The date zone is always positioned at the upper right corner of each cheque. '1' and '9' or '2' and '0' are often printed as isolated numerals in the date zone indicating the century (the 20^{th} century or 21^{th} century). The part to the right of the machine-printed "19" or "20" is intended for entering the last two digits of *Year*. The part on the left is intended for *Month* and *Day*, and it is completely blank, which implies there is no pre-defined position for *Month* and *Day*.

As mentioned in Chapter 1, the users usually write the cheques freely and there is no restriction on the writing style for the date field. To represent a certain date, for example "February 20, 2002", the following 8 variations are mostly used when the machine-printed "20" appears in the date field to indicate the current century:

February(,) $20^{(th)}$(,) 2002
Feb(.) $20^{(th)}$(,) 2002
$20^{(th)}$ February 2002
$20^{(th)}$ Feb(.) 2002

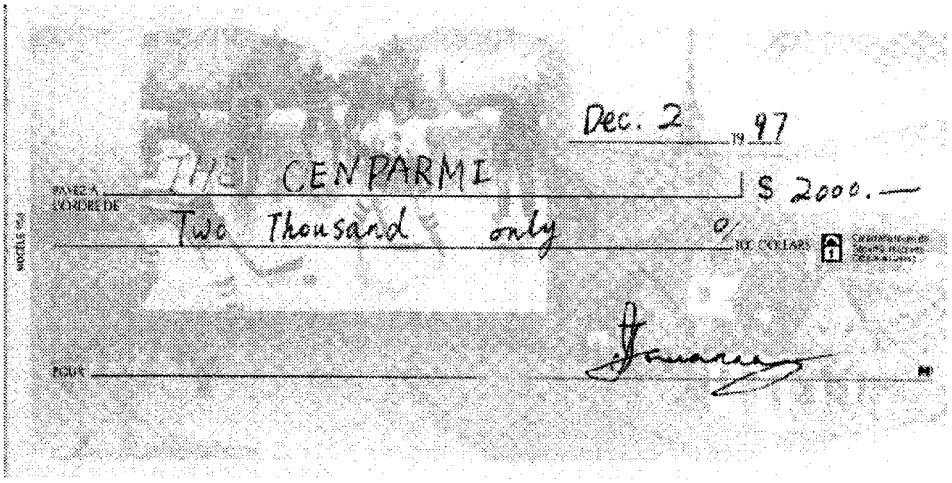

Figure 5: Example of a standard bank cheque

(0)2/20(/) 2002
(0)2-20(-) 2002
20/(0)2(/) 2002
20-(0)2(-) 2002

In the list above, items shown within parentheses denote entities which may or may not be present. Suffixes such as 'st', 'nd', 'rd', or 'th' can be written either as superscripts or at the same horizontal level as the rest of the date field information.

Besides these popular formats to represent a date on bank cheques, even more variations exist in some parts of Canada. First, different punctuations can be used to identify the end of one field, e.g. *February, 20. 2002.* Second, when *Month* is written in letters, slash ('/') or hyphen ('-') can also be used, e.g. *February/20 2002.* Third, sometimes the article "Le" appears at the beginning of the date zone when cheques are written in French. In addition, the machine-printed "19" or "20" may not appear on some bank cheques, and different writing styles are used in this situation (as shown in Figure 6).

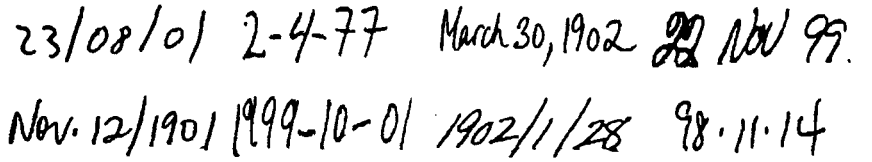

Figure 6: Sample dates without machine-printed century symbols

3.1.2 Basic Assumptions on Writing Styles

The writing styles of date fields extracted from Canadian bank cheques can be grouped into two categories, which are defined as standard format and free format in this thesis.

The standard format can be expressed in the following 4 patterns, with possible suffixes and article "Le":

(a) MM S dd (S) Cent yy

(b) dd S MM (S) Cent yy

(c) mm S dd (S) Cent yy

(d) dd S mm (S) Cent yy

Here Cent represents the century symbol ("19" or "20") printed on the cheques. "MM" refers to *Month* field in letters (in full or abbreviated), either English or French; "mm" refers to *Month* field in numerals; "dd" refers to *Day* field in numerals; "yy" refers to *Year* field in numerals and 'S' refers to separators. Both punctuations and the gap between two fields can be considered as separator 'S'. The suffixes and article "Le" are not shown, which can be written after and before *Day* field, respectively.

When the machine-printed "19" or "20" does not appear on a bank cheque, the free format is adopted, which can be simply expressed as:

field1 S field2 S field3

Here *Year* can only be field1 or field3 and can include 2 numerals or 4 numerals. According to common sense and database analysis, when *Year* is located at field3, the other two fields are *Month* and *Day* with similar variations to those in the standard format. However, when *Year* is located at field1, *Month* and *Day* can only be field2 and field3, respectively. So there are more variations in the free format. More discussions about the free format will be given in Chapter 5.

3.1.3 Statistical Results about Writing Styles

Cheque Databases

In the design and development phase of an automatic cheque processing system, it is necessary to have access to a large quantity of bank cheques in order to gain insight into the various ways in which cheques can be written. However, due to security and confidentiality considerations, it is very difficult to have access to real cheques from banks or utility companies even for research purposes. For this reason, we decided to create our own databases.

Two databases have been established in CENPARMI for the research on an automatic bank cheque processing system. One is CENPARMI Cheque database, and the other is CENPARMI_IRIS Cheque database.

The CENPARMI Cheque database contains cheques having similar size and layout as regular bank cheques. The sample cheque was carefully designed with a white background, and all the lines are printed in special drop-out ink which is invisible to the scanner. This facilitates the extraction of written information. The cheques were filled in by students of Concordia university and Ecole Polytechnique de Montréal.

Unlike the CENPARMI Cheque database, the CENPARMI_IRIS Cheque database consists of real-life standard Canadian personal cheques. The cheques are designed and used by different banks in Canada, which means the backgrounds, layout and geometric measurements may be different. Usually the cheques have complicated backgrounds with a variety of colorful scene images. The cheques were filled in by employees/students of Concordia university and staff of Bell Canada.

The cheques in both of the databases are first scanned with a resolution of 300 dots

per inch, after which the courtesy amount, legal amount and date fields are extracted and binarized [95], ready for further processing for each item of written information.

For the date processing system, 4564 date images are obtained from the CENPARMI Cheque database, and 10865 date images are obtained from CENPARMI_IRIS Cheque database. Database truthing for the date zone images in these two databases has been performed by using a powerful truthing tool [32]. The date images were tagged by attaching a message string to each date image, which contains the position information about all the three fields, *Day*, *Month* and *Year*, as well as punctuations and the printed numerals "19", etc. Based on the message string tagged to each image, cursive month words have been extracted for training and testing the month word recognizer, and the entire date processing system can be tested automatically by comparing results with the truth values.

We have established two training sets from the above databases to train our date processing system, and the system is tested on another subset of the CENPARMI_IRIS Cheque database in order to test the capability of the system in a practical environment. The first training set, Training Set 1, consists of all samples (4564 samples) from the CENPARMI Cheque database and about half of the samples (5470 samples) from the CENPARMI_IRIS Cheque database. The second training set, Training Set 2, consists of 1996 samples from the CENPARMI_IRIS Cheque database. The test set consists of the remaining 3399 samples from the CENPARMI_IRIS Cheque database. Training Set 1 is mainly used to train the month word recognizer and A/N differentiation modules used in date processing, while Training Set 2 is used for developing the rules for date segmentation, etc. The training sets will be indicated in the following chapters when they are used.

Statistics on CENPARMI_IRIS Cheque Database

As discussed in previous sections, when dates are written in an unconstrained environment, such as on bank cheques, many writing styles can be used to represent a date. Since it is very difficult and inefficient to develop an individual algorithm for each writing style, the best strategy is to analyze all the variations in detail, and to

categorize them if possible. Therefore, writing styles are further studied here, and the information obtained will be used for making decisions in automatic date zone processing.

Given that a study of the CENPARMI Cheque database has already been made in [32], we present here some statistics on the CENPARMI_IRIS Cheque database. The samples from the CENPARMI_IRIS Cheque database in Training Set 1 form the Statistical Set (5470 samples). Some statistics on this set are given below.

Table 4: General information on Statistical Set

	Number of Images	Percentage (%)
Total no. (English)	2411	100
Month_in_letter	2168	89.92
Month_in_numeral	243	10.08
Total no. (French)	3059	100
Month_in_letter	2258	73.81
Month_in_numeral	801	26.19

Table 5: Layout information on Statistical Set

	Number of Images	Percentage (%)
Total no. (English)	2411	100
With machine-printed century symbol	2296	95.23
Without machine-printed century symbol	115	4.77
Total no. (French)	3059	100
With machine-printed century symbol	2009	65.68
Without machine-printed century symbol	1050	34.32

Table 4 and Table 5 provide some general information about the samples analyzed. From these results, we can infer that most people tend to use *Month_in_letter* to represent a month, and that the machine-printed century symbol often appears on date zones of Canadian bank cheques.

In addition, in the English set, 29.66% of *Month_in_letter* date images have punctuations and 98.77% of *Month_in_numeral* date images have punctuations. In the French set, 12.36% of *Month_in_letter* date images have punctuations and 98.25% *Month_in_numeral* date images have punctuations. Detailed statistics about punctuation usage is given in Table 6 and Table 7.

Table 6: Use of punctuation when *Month* is in letters

	Number of Images	Percentage (%)
Total no. (English)	643	100
with '.' or ',' only	582	90.51
with '/' only	20	3.11
with '-' only	24	3.73
with two of the above types	17	2.64
Total no. (French)	279	100
with '.' or ',' only	240	86.02
with '/' only	9	3.23
with '-' only	28	10.04
with two of the above types	2	0.71

From the results above, we can conclude that when *Month* is written in letters, people prefer to use gaps to separate *Day*, *Month* and *Year* fields (74.34% for the English set and 87.64% for the French set); if punctuation is used in this case, the punctuation '.' or ',' is mostly used (90.51% for the English set and 86.02% for the French set). However, when *Month* is written in numeral(s), people usually use punctuation to separate *Day*, *Month* and *Year* fields (98.77% for the English set and 98.25% for the French set); if punctuation is used in this case, the punctuation '/' or '-' is mostly used (86.25% for the English set and 94.41% for the French set). All the information obtained here will be used to guide us in the design of our date processing system.

From analyses of Statistical Set, we also find that 4.31% of English date images have suffixes while 0.95% of French date images have suffixes, and 4.12% of French date images have "Le" at the beginning of the date zones.

Table 7: Use of punctuation when *Month* is in numeral(s)

	Number of Images	Percentage (%)
Total no. (English)	240	100
with '.' or ',' only	23	9.58
with '/' only	145	60.42
with '-' only	62	25.83
with two of the above types	10	4.17
Total no. (French)	787	100
with '.' or ',' only	26	3.30
with '/' only	434	55.15
with '-' only	309	39.26
with two of the above types	18	2.29

3.2 System Architecture

Since the date image contains fields which may belong to different categories (cursive script or numeric), it is difficult to process the entire date image at the same time efficiently. Therefore, the basic principle used in our previous system [33] will be adopted here; that is, a segmentation-based strategy is employed in our new system, and many modifications have been made to improve the performances.

The main procedures in the system consist of segmenting the date image into fields through the detection of the separator or transition between the fields, identifying the nature of each field, and applying an appropriate recognizer for each.

In order to improve the performance and efficiency of the system, segmentation is divided into two stages. In the first stage, a knowledge-based segmentation module is used to solve most segmentation cases. Ambiguous cases are handled by a multi-hypotheses generation module at this stage, for a final decision to be made when more contextual information and syntactic and semantic knowledge are available, i.e., multi-hypotheses evaluation is made in the recognition stage.

Figure 7 illustrates the basic modules in our date processing system. In addition to the two main modules related to segmentation and recognition, a preprocessing

module has been designed to deal with simple noisy images, and to detect and process possible appearances of "Le". In the postprocessing stage, a two-level verification module is implemented to further improve the reliability and performance of the system. The two-level structure means the verification module has the ability not only to reject or accept results, but also to correct some errors.

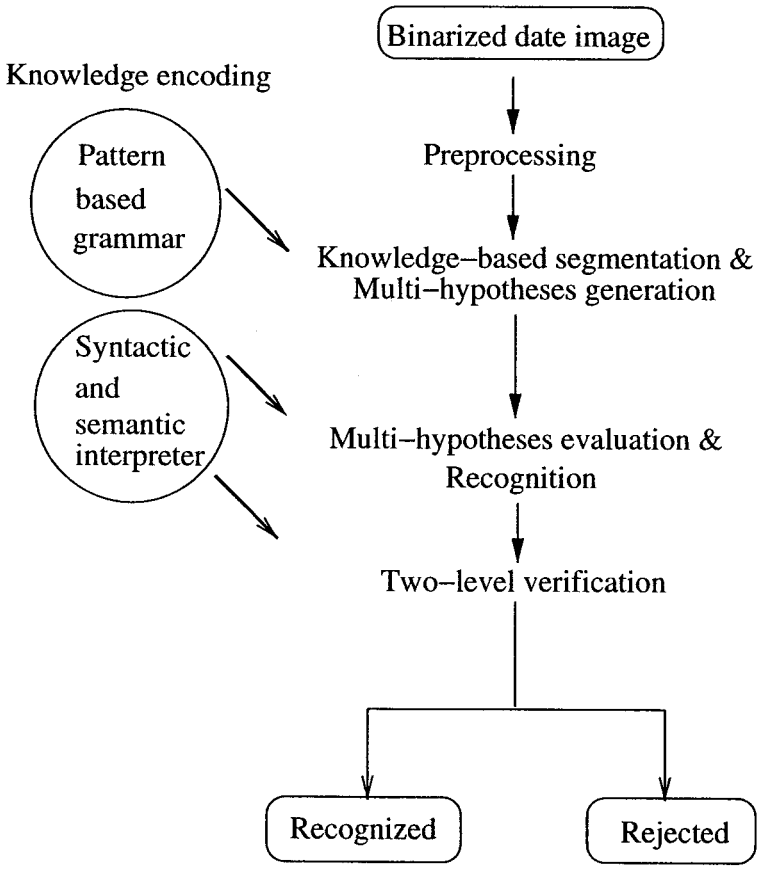

Figure 7: Diagram of date processing system

Chapter 4

Differentiation between Alphabetic and Numeric Data (A/N Differentiation)

4.1 A/N Differentiation and its Applications to Date Segmentation

In the handwritten date fields on bank cheques, both cursive words and numeric strings can be used in various writing styles to represent dates. Since our approach of processing the date image is a segmentation-based method, the differentiation between alphabetic words and numeric strings (A/N differentiation) is very important in the segmentation stage. It is obvious that this differentiation can be used to identify the nature of each field so that the numeric or alphabetic format of the month field could be determined, and a proper recognizer can be applied. At the same time, this differentiation module can be used to detect the segmentation point between *Day* and *Month* fields, as motivated by human performance, i.e., human beings usually use the following cues to separate *Day* from *Month*:

(a) Significant gap or punctuation ('-', '/', '.', and ',') between two parts of *Day&Month* field (the subimage which contains both *Day* and *Month* fields is the *Day&Month* field)

(b) Transition between numerals and alphabetic letters, and

(c) Actual recognition prior to segmentation

Among these, (a) is always used first, (b) becomes an important cue when month is written in alphabetic format, since the separator usually appears at this transition point, and (c) can be used at any stage.

In order to make use of (b), the A/N differentiation module is applied to determine the subimages in *Day&Month* field that are written in either numerals or in letters, with the result that the separator at the transition between numerals and alphabetic letters can also be found. In fact, this differentiation module has more applications in the segmentation stage, which will be discussed in Chapter 5.

Two methods will be presented in this chapter for the A/N differentiation. The first method adopts a *distance_to_numeral* measure to represent the likelihood of a subimage in *Day&Month* field being numeric, and it is based on feedbacks of a digit recognizer and structural features. A system of combining multiple Multi-layer Perceptron (MLP) networks is the second method to realize this A/N differentiation task. More details about these two methods are given below.

4.2 Distance_to_numeral Measure for A/N Differentiation

The *distance_to_numeral* measure represents the likelihood of a subimage in *Day&Month* field being numeric, and it is a combination of (a) a confidence value from a connected digit recognizer; (b) recognition results from the connected digit recognizer; and (c) structural features.

(a) Confidence value from a connected digit recognizer

A connected digit recognizer designed for the courtesy amount recognition in a bank cheque processing system [133] is used here. We define the confidence value returned from the recognizer as $Confid_{\mathrm{DR}}$, where $Confid_{\mathrm{DR}}$ represents an average confidence value for the numerals recognized.

(b) Recognition results from the connected digit recognizer

The results include the string value returned from the digit recognizer, $String_{\mathrm{value}}$, and the length of the string, $String_{\mathrm{length}}$. If the digit recognizer produces correct results, $String_{\mathrm{value}}$ and $String_{\mathrm{length}}$ should vary within certain ranges for any numeric subimage in *Day&Month* field, that is, $String_{\mathrm{length}} < 3$ (representing 1 or 2 numerals) and $String_{\mathrm{value}} \leq 31$.

(c) Structural features

These features consist of the maximum number of horizontal runs, the sum of the numbers of peaks and valleys, and the number of inner loops in the subimage. These feature values should be below certain thresholds for numeric subimages since they are usually simpler than alphabetic ones. The thresholds are determined empirically in the training stage. The training set (named as A/N Training Set) is composed of 2103 month words and 2200 numeric samples extracted from Training Set 1 (described in Chapter 3).

- Maximum number of runs

  The Maximum number of runs, *max_run*, can be found for the entire subimage (record the number of runs for each row and return the maximum number over all rows). Usually, a subimage with only one numeral has a low value for the *max_run*, while a alphabetic subimage can have a much higher *max_run*, and this can sometimes be used to differentiate between numeric and alphabetic subimages. For example, part of a word 'December' (see Figure 8) is recognized by the digit recognizer as '6' and the confidence value is 0.58.

Figure 8: Sample image extracted from a word 'December'

Although the confidence value is not low, the *max_run* is 6, which is larger than the threshold of the *max_run* for a normal numeral. In this case, the subimage would not be considered as a numeral.

- Sum of the numbers of peaks and valleys
  For this feature, we calculate the sum of the numbers of peaks and valleys for each connected component in the subimage and return the maximum value as the feature value. The feature is represented by *num_peak*.

- The number of inner loops
  The number of inner loops, *num_loop*, is calculated for the entire subimage. A subimage containing only one or two numerals should have a low threshold for the *num_loop*. Since different numerals have different shape features, different threshold values for *num_loop* are assigned to different numerals. The threshold value for '0', '2' and '8' is 2, for '1' is 0 and for other numerals is 1. The threshold, *threshold_loop*, for the entire subimage is the sum of the threshold values of the numerals recognized by the digit recognizer. In addition, if only one numeral is returned from the digit recognizer, *threshold_loop* is incremented by 1.

When $Confid_{\mathrm{DR}}$ and $String_{\mathrm{length}}$ have been obtained from the digit recognizer, and *max_run*, *num_peak*, and *num_loop* have been calculated for a subimage, a confidence value $Confid_{\mathrm{numeric}}$ is estimated to provide the *distance_to_numeral* measure for the subimage. If $String_{\mathrm{length}}$, *max_run*, *num_peak*, and *num_loop* values are below certain thresholds for the subimage, $Confid_{\mathrm{numeric}}$ value is equal to $Confid_{\mathrm{DR}}$. Otherwise,

$Confid_{\text{numeric}}$ is equal to a reduced $Confid_{\text{DR}}$ value, and the reduction is based on $String_{\text{length}}$, max_run, num_peak, and num_loop values. In the following, the procedure for calculating $Confid_{\text{numeric}}$ is given. $Component_{\text{number}}$ is the number of main connected components in the subimage (Main components have heights exceeding a threshold established through experimentation).

if $String_{\text{length}} > 3 \cup (String_{\text{length}} = 3 \cap Component_{\text{number}} = 3)$
  then $Confid_{\text{numeric}} = 0$
  else
    if $String_{\text{length}} = 1 \cap max\_run > 5$
      then $Confid_{\text{numeric}} = Confid_{\text{DR}} * 5/(max\_run + 2)$
      else
        if $String_{\text{length}} \geq 2 \cap max\_run > 6$
          then $Confid_{\text{numeric}} = Confid_{\text{DR}} * 6/(max\_run + 2)$
          else $Confid_{\text{numeric}} = Confid_{\text{DR}}$
        end
    end
    if $num\_peak > 5$
      then $Confid_{\text{numeric}} = Confid_{\text{numeric}} * 5/(num\_peak + 2)$
    end
    if $num\_loop > threshold\_loop$
      then $Confid_{\text{numeric}} = Confid_{\text{numeric}} * threshold\_loop/(num\_loop + 2)$
    end
end

Usually $String_{\text{length}} < 3$ for a numeric subimage, but sometimes the digit recognizer splits one component and recognizes it as two digits. In addition, the values for the structure features (the maximum number of runs, the sum of the numbers of peaks and valleys and the number of inner loops) are expected to be less than some thresholds

for a subimage with only one or two numerals. These thresholds are obtained in the training stage from A/N Training Set mentioned above.

We have tested the $Confid_{\text{numeric}}$ value on a test set consisting of 2102 month word samples and 2000 numeric strings (each containing one or two numerals) extracted from Test Set (described in Chapter 3). This test set is named A/N Test Set 1. The statistical results on A/N Test Set 1 are presented in Figure 9.

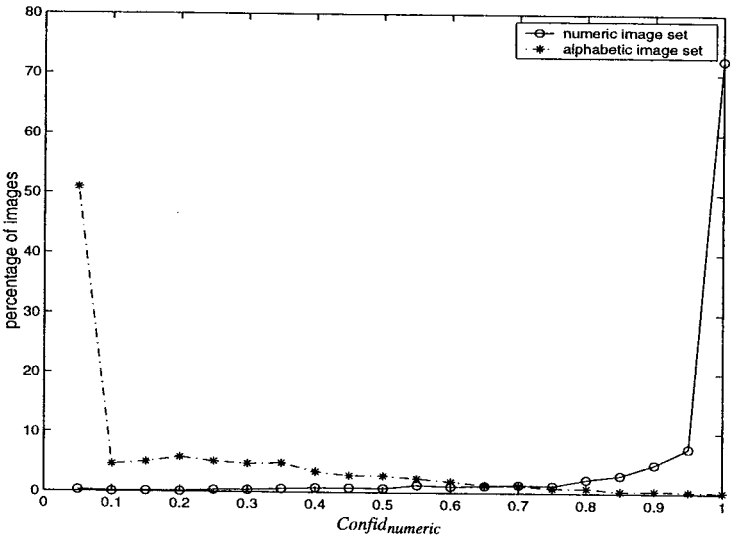

Figure 9: Relationship of $Confid_{\text{numeric}}$ to numeric and word images

The effectiveness of the measure $Confid_{\text{numeric}}$ in differentiating between alphabetic and numeric images can be seen from Figure 9 since there is a strong correlation between very high (low) values of $Confid_{\text{numeric}}$ and numeric (alphabetic or word) samples. However, this method depends heavily on the results from the digit recognizer, which was developed for recognizing the courtesy amount [133] and cannot be fully controlled. In addition, when the $Confid_{\text{numeric}}$ value is not at the two extremes, this value is not effective in making a reliable decision. Therefore, an independent module for differentiating between numeric and alphabetic data is developed by combining

multiple multi-layer perceptron networks. In the following section, a multi-layer perceptron model for A/N differentiation is first presented, to be followed by a discussion of the combination system.

4.3 A Multi-layer Perceptron Model for A/N Differentiation

As discussed in Chapter 2, neural network classifiers are among the most frequently used classifiers for handwriting recognition. In this section a neural network classifier is presented to differentiate handwritten alphabetic words from numeric strings, which is a novel research topic in handwriting recognition.

To implement the A/N differentiation, a MLP classifier is developed. The architecture of the typical MLP has been adapted for the A/N differentiation, and the features are also selected and designed specifically for this new handwriting recognition task.

4.3.1 The Architecture and Learning Algorithm of the MLP Network

Network Architecture

It has been shown that multi-layer perceptron networks with a single hidden layer and a nonlinear activation function are universal classifiers [51, 55]. That is, such networks can approximate decision boundaries of arbitrary complexity. A diagram of a standard MLP with one hidden layer is shown in Figure 10.

The input vector representing the pattern to be recognized is incident to the input layer and distributed to the subsequent hidden layer and finally to the output layer via weighted connections. Each neuron in the network operates by taking the sum of its weighted inputs and passing the result through a nonlinear activation function.

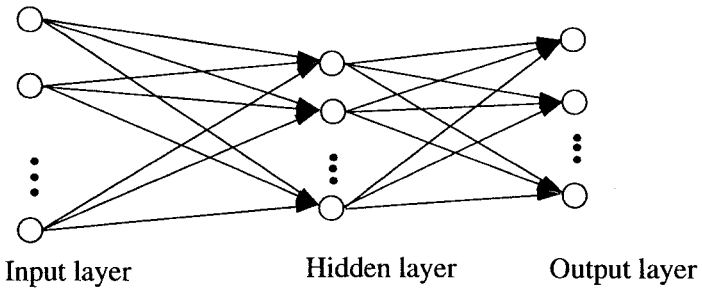

Figure 10: Diagram of MLP with one hidden layer

The most common activation function used is a Sigmoid function, defined as:

$$sig(x) = (1 + e^{-\beta x})^{-1} \tag{1}$$

To implement the A/N differentiation, this typical architecture has been modified. Since alphabetic month words can be written in full or abbreviated form, multi-subclass output nodes (e.g., 2 nodes) for the alphabetic class are set to counter the high variation in the lengths of month words. On the other hand, since the numeric strings from the date images include only one or two numerals, one output node is assigned to the numeric class. Figure 11 shows the new architecture of the MLP. Performances have been improved by using this architecture, as shown in the section on experiments.

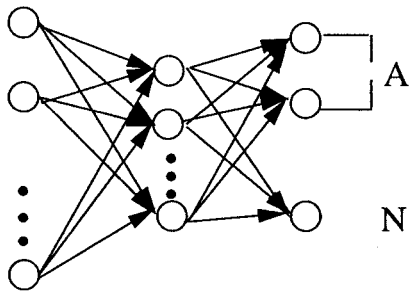

Figure 11: Architecture of MLP for A/N differentiation

Learning Algorithm

The error back-propagation (BP) algorithm is used as the learning algorithm as is commonly used. Since we are dealing with supervised training, the desired value is known for the training set. Typically for the back-propagation [122] training algorithm, the mean square error is used to measure how close the network has come to an established desired value. The aim is to minimize the network total error by adjusting the weights, which can be done using a gradient search technique. With the notation as shown in Table 8, together with the architecture shown in Figure 10, the learning algorithm can be summarized as follows:

Table 8: Notations used in BP training algorithm

$w_{ij}^{(l)}$	Connecting weight between ith node in $l+1$ layer and jth node in l layer
$d_{i,k}$	Desired output of ith output node for input sample x_k
$u_{i,k}$	Real output of ith output node for input sample x_k
$o_{j,k}$	Output of jth hidden layer node for input sample x_k
$x_{n,k}$	nth component for input sample x_k
p	Number of training patterns
n_h	Number of hidden layer nodes
n_i	Number of input layer nodes
n_o	Number of output layer nodes

$$u_{i,k} = sig(\sum_{j=1}^{n_h} w_{ij}^{(2)} o_{j,k}) \tag{2}$$

$$= sig(\sum_{j=1}^{n_h} w_{ij}^{(2)} sig(\sum_{n=1}^{n_i} w_{jn}^{(1)} x_{n,k})) \tag{3}$$

The mean square error E is defined as:

$$E_k = \frac{1}{2}\sum_{i=1}^{n_0}(d_{i,k} - u_{i,k})^2 \tag{4}$$

$$E = \sum_{k=1}^{p} E_k \tag{5}$$

During the training process, the connecting weights are updated by gradient descent, where the derivatives with respect to the weights are computed as:

$$\frac{\partial E_k}{\partial w_{ij}^{(2)}} = \delta_{i,k}^{(2)} o_{j,k} \tag{6}$$

$$\delta_{i,k}^{(2)} = (d_{i,k} - u_{i,k}) u_{i,k} (1 - u_{i,k}) \tag{7}$$

$$\frac{\partial E_k}{\partial w_{jn}^{(1)}} = \delta_{j,k}^{(1)} x_{n,k} \tag{8}$$

$$\delta_{j,k}^{(1)} = o_{j,k}(1 - o_{j,k}) \sum_{i=1}^{n_0} w_{ij}^{(2)} \delta_{i,k}^{(2)} \tag{9}$$

So the weights are adjusted by:

$$\Delta w_{ij}^{(2)} = \eta \delta_{i,k}^{(2)} o_{j,k} \tag{10}$$

$$\Delta w_{jn}^{(1)} = \eta \delta_{j,k}^{(1)} x_{n,k} \tag{11}$$

where η refers to the learning rate, and $\delta_{m,k}^{(l)}$ refers to the error signal at node m in l layer for input sample x_k.

Corresponding to the modified architecture given in Figure 11, the above learning algorithm has been adjusted. Only the maximum output value of the multi-subclass output nodes within one class is used to adjust the weights. Let the ith node in the output layer be the mth subclass of nth class:

$$u_{i,k} = u_{n_m,k} \qquad 1 \le n \le S \qquad 1 \le m \le N_n \tag{12}$$

where S is the total number of classes, and N_n is the number of output nodes for the nth class. Then Equation 7 is modified as:

$$\delta_{i,k}^{(2)} = \begin{cases} (d_{i,k} - u_{i,k}) u_{i,k} (1 - u_{i,k}) & \text{if } u_{i,k} = \max_{m=1}^{N_n} u_{n_m,k} \\ 0 & \text{otherwise} \end{cases} \tag{13}$$

This new $\delta_{i,k}^{(2)}$ is used in Equation 10 to adjust the weights.

4.3.2 Feature Extraction

Features play an important role in OCR systems, and they can be crucial for obtaining high recognition rates [137]. For A/N differentiation, three statistical features, gradient, mesh and projection features, are first adopted. These features are among the most widely used features by neural network classifiers [107, 19]. In addition, two distance features (horizontal and vertical) having good discriminative powers for A/N differentiation have been designed.

(1) Gradient feature

The gradient measures the magnitude and direction of the greatest change in intensity in a small neighborhood of each pixel. Since the object contour and structure are encoded in the gradient direction and magnitude at each pixel of the image, features extracted from the gradient representation are very powerful for pattern recognition [131].

In our gradient feature extraction module, the gradients are first computed by convolving two Sobel operators on the binary image [131]. The Sobel operators used to compute the horizontal (x) and vertical (y) components S_x and S_y of the gradient are shown in Figure 12.

Given an input image $I()$ of size $D_1 \times D_2$, S_x and S_y are determined as:

$$\begin{aligned} S_x(i,j) &= I(i-1,j+1) + 2I(i,j+1) + I(i+1,j+1) \\ &\quad -I(i-1,j-1) - 2I(i,j-1) - I(i+1,j-1) \end{aligned} \tag{14}$$

$$\begin{aligned} S_y(i,j) &= I(i-1,j-1) + 2I(i-1,j) + I(i-1,j+1) \\ &\quad -I(i+1,j-1) - 2I(i+1,j) - I(i+1,j+1) \end{aligned} \tag{15}$$

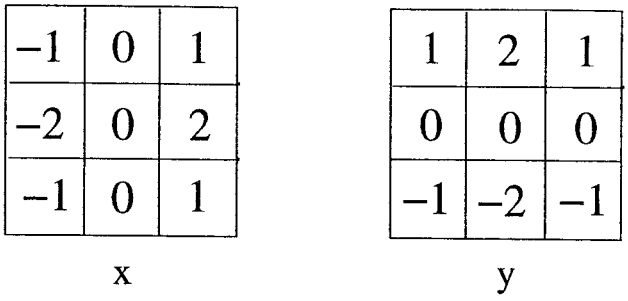

Figure 12: Sobel operator templates used for convolution

Here, i and j range over D_1 and D_2, respectively. The gradient magnitude is then calculated as:

$$r(i,j) = \sqrt{S_x^2(i,j) + S_y^2(i,j)} \tag{16}$$

The gradient direction is calculated as:

$$\theta(i,j) = \tan^{-1}\frac{S_y(i,j)}{S_x(i,j)} \tag{17}$$

The complete gradient map is rich in information; however, some processing of the gradient map is necessary to highlight the important information for the purpose of A/N differentiation. An adaptive gradient thresholding method [131] is used to filter out spurious gradients in the image. Thresholding is performed to nullify pixels whose gradient magnitude value lies below a threshold. Here the average gradient magnitude over the whole image gradient map is used as the threshold.

After the thresholding step, only the gradient direction at each pixel with nonzero gradient magnitude value is used in the computation of our feature vector. In order to generate a fixed number of features, the gradient directions are quantized into a small number of ranges, and images are divided into local regions as shown in Figure 13(a). Finally, the number of pixels having the same gradient angle range in each of the sub-divided local regions of the binary image is calculated as one component of

the gradient feature. These values are normalized by dividing by the maximum value.

(2) Mesh feature

The mesh feature is obtained by computing the number of black pixels in each of the sub-divided local regions and then dividing the values by the maximum number of black pixels among all the regions (as shown in Figure 13(a)).

(3) Projection feature

The ratio of the number of black pixels in each equally spaced horizontal line to the width of the binary image (as shown in Figure 13(b)) is calculated to form the projection feature.

(4) Distance features

Two kinds of distance features are extracted: horizontal distance feature and vertical distance feature. (i) The horizontal distance feature is the horizontal distance from the right edge of the minimum bounding rectangle of the word to the contour divided by the width of the image. Only the upper part of the image is considered (see Figure 13(c)). (ii) The vertical distance feature is the vertical distance from the minimum bounding rectangle of the word to the contour divided by the height of the image (see Figure 13(d)). The vertical distance feature is further divided into two types according to the distance positions: top and bottom distance features.

The vertical distance features are designed based on the knowledge that usually the differences in heights of letters within an alphabetic word are greater than those of digits within a numeric string. The horizontal distance feature is adopted because usually capital letters are used in the beginning of alphabetic words, which makes them different from numeric strings, and the horizontal distance feature is an attempt to exploit this difference.

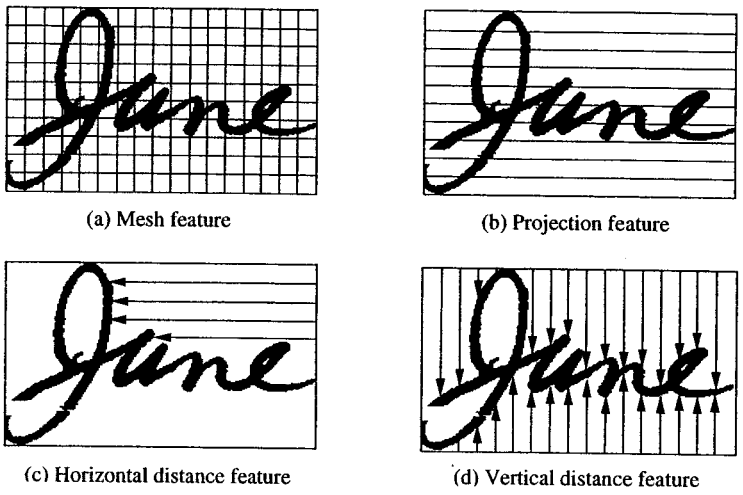

Figure 13: Feature extraction

4.3.3 Experiments

In this subsection, several experiments have been designed to test the effectiveness of the MLP architecture and feature extraction methods described in previous subsections. Eight MLP networks based on the same architecture but different feature sets are first presented. Then the architecture of the MLP networks is discussed. Finally some analyses about the features and some concluding remarks will be given.

Design of MLP Networks

The MLP networks have the architecture shown in Figure 11. A Sigmoid function in the range from 0 to 1 ($\beta = 1$) is used as an activation function. During the training phase by BP algorithm, all the connecting weights are initialized to random small values, and the learning rate η is set to 0.05. The training set used here is A/N Training Set (described in the last section). The 4303 training samples are repeatedly fed into the network and the weights are updated according to Equations 10 and 11. The training process stops if one of the following conditions is satisfied: (a) the number of training epochs exceeds a threshold of 5000; and (b) the recognition rate

on the training set is higher than 99.5%. In the test stage, since the architecture of MLP networks has been modified as shown in Figure 11, only the output from the maximum subclass output node within a class is considered.

Eight MLP networks are implemented by using different feature types:

- Feature set 1 consists of gradient features (120D), where the gradient direction is quantized into four ranges of 90° each, and the number of local regions is 30 (3 vertical and 10 horizontal divisions).
- Feature set 2 consists of mesh features (120D), where the number of local regions is 256 (16x16) and the dimension of the feature vector is reduced from 256 to 120 by using principal component analysis.
- Feature set 3 consists of projection features (33D).
- Feature set 4 consists of 33D top and 33D bottom vertical distance features (66D).
- Feature set 5 consists of 16D horizontal distance feature and 33D projection feature (49D)
- Feature set 6 consists of 33D top vertical distance feature and 33D projection feature (66D)
- Feature set 7 consists of 33D bottom vertical distance feature and 33D projection feature (66D)
- Feature set 8 consists of 66D top and bottom vertical distance features and 33D projection feature (99D)

For testing the performances of these individual MLP networks, A/N Test Set 2 is used. A/N Test Set 2 is composed of 2102 month words and 2000 numeric samples extracted from Training Set 1 (described in Chapter 3) that is separate from A/N

Table 9: Performances of individual MLP networks on A/N Test Set 2

	Correct (%)	Error (%)
MLP1 (gradient feature) (121-60-3)	97.22	2.78
MLP2 (mesh feature) (121-60-3)	95.49	4.51
MLP3 (projection feature) (34-15-3)	93.86	6.14
MLP4 (top & bottom vertical distance feature) (67-30-3)	92.42	7.58
MLP5 (horizontal distance & projection feature) (50-25-3)	94.64	5.36
MLP6 (top vertical distance & projection feature) (67-30-3)	95.73	4.27
MLP7 (bottom vertical distance & projection feature) (67-30-3)	94.15	5.85
MLP8 (top & bottom vertical distance & projection feature) (100-50-3)	95.20	4.80

Training Set. The performances of these individual MLP networks on A/N Test Set 2 are shown in Table 9.

From Table 9, we can see that both the MLP model proposed and the features extracted are effective. In the following, more analyses and experiments are given to support this conclusion.

Architecture of the MLP

In previous discussions, the reasons for using multi-subclass output nodes for one class have been given, and these are also supported by experimental results. Since MLP1 and MLP6 have produced the best performances as shown in Table 9, we performed experiments with these MLPs on A/N Test Set 2 using only one output node for each class, and Table 10 shows that results are better when two output nodes are used for the alphabetic class.

Table 10 also shows that different networks can gain different benefits from the architecture used. Further experimentation has been conducted to determine the

Table 10: The usage of multi-subclass output nodes for one class

	Correct (%)	Error (%)
MLP1 (Gradient feature): two output nodes for alphabetic class and one node for numeric class (121-60-3)	97.22	2.78
Modified MLP1: one node for each class (not using the multi-subclasses for one class) (121-60-2)	97.12	2.88
MLP6 (Top distance & projection feature): two output nodes for alphabetic data and one for numeric data (67-30-3)	95.73	4.27
Modified MLP6: one output node for each class (not using the multi-subclasses for one class) (67-30-2)	95.00	5.00

number of multi-subclasses that should be selected for one class, and Table 11 gives some results related to MLP1. In this experiment, MLP1 gives the best results when two and one output nodes are used for the alphabetic and numeric classes, respectively. Currently all MLP networks adopt fixed numbers of multi-subclasses for the two classes: two for alphabetic class and one for numeric class. It is also consistent with the discussion in section 4.3.1.

With regard to the architecture of the MLP, another experiment is conducted to compare performances from using different numbers of hidden layer nodes. Usually the number of hidden layer nodes should correspond to the numbers of input layer and output layer nodes. The results obtained from MLP1 are shown in Table 12.

Since no significant difference is observed in the three cases in Table 12, we can infer that the number of hidden layer nodes is not a key factor in the performance of MLP1.

Table 11: Selection of multi-subclass output nodes for one class

	Correct (%)	Error (%)
MLP1: two multi-subclass output nodes for alphabetic class and one output node for numeric class (121-60-3)	97.22	2.78
Modified MLP1: three multi-subclass output nodes for alphabetic class and one output node for numeric class (121-60-4)	97.10	2.90
Modified MLP1: two multi-subclass output nodes for alphabetic class and two multi-subclass output node for numeric class (121-60-4)	97.10	2.90

Table 12: Performance from using different numbers of hidden layer nodes

	Correct (%)	Error (%)
MLP1: the number of hidden layer nodes is 60 (121-60-3)	97.22	2.78
Modified MLP1: the number of hidden layer nodes is 30: (121-30-3)	97.20	2.80
Modified MLP1: the number of hidden layer nodes is 80: (121-80-3)	97.25	2.75

Feature Analyses

In this subsection, we will focus on the gradient feature because MLP1 produces much better results than other networks. Firstly, some experimental results concerning the division of local regions and the quantization of the gradient directions in relation to performance will be given. Secondly, the effectiveness of the gradient feature will be demonstrated by considering several aspects. Finally, principal component analysis will be discussed, as this method has been used to reduce the dimension of the mesh feature and will be applied to the gradient feature to improve the performance.

(1) Parameter selection in gradient feature extraction

The parameters of the gradient feature used in MLP1 are: partition of 360° gradient direction into 4 directions, 10 horizontal and 3 vertical divisions (3x10 local regions). The first experiment is designed to test the effect of changing the horizontal and vertical divisions, and the results are shown in Table 13.

Table 13: Performance comparison by using different region divisions

	Correct (%)	Error (%)
Gradient feature used in MLP1: 4 directions, 3 vertical and 10 horizontal divisions (121-60-3)	97.22	2.78
Gradient feature: 4 directions, 3 vertical and 5 horizontal divisions (61-30-3)	97.05	2.95
Gradient feature: 4 directions, 2 vertical and 10 horizontal divisions (81-40-3)	96.64	3.36
Gradient feature: 4 directions, 4 vertical and 10 horizontal divisions (161-60-3)	97.32	2.68

From Table 13, it can be seen that dividing the region into different subregions does affect the result: the number of vertical divisions should be at least 3, while both 5 and 10 horizontal divisions are acceptable.

In the second experiment, the quantization level for the gradient directions is tested, and the results are given in Table 14. From these results, we can conclude that using 4 quantization levels for the gradient feature is the best choice, as the results actually deteriorate when more levels are used.

From the experiments above, we conclude that the best parameters for the gradient feature are 4 partitions of 360° gradient direction, 10 horizontal and 4 vertical divisions (4x10 local regions). The MLP based on this gradient feature has 97.32% correct rate for the A/N differentiation on A/N Test Set 2, which is much better than the other MLP networks presented in Table 9. Since all MLP networks have the same architecture, we can conclude that the gradient feature is more powerful for the

Table 14: Performance comparison by using different quantization levels for the gradient directions

	Correct (%)	Error (%)
Gradient feature used in MLP1: 4 directions, 3 vertical and 10 horizontal divisions (121-60-3)	97.22	2.78
Gradient feature: 6 directions, 3 vertical and 10 horizontal divisions (181-60-3)	96.88	3.12
Gradient feature: 8 directions, 3 vertical and 10 horizontal divisions (241-90-3)	96.39	3.61

A/N differentiation task than the other features. In the following subsection, some analyses will be presented to show the effectiveness of the gradient feature.

(2) Effectiveness of the gradient feature

Good features must minimize the within-class pattern variability and maximize the between-class pattern variability to provide for sufficient discrimination between different classes. To consider these variabilities of the gradient feature, the mean value and standard deviation of each gradient feature component (160 dimension: 4 direction, 4 vertical and 10 horizontal divisions) are determined for both alphabetic and numeric data in A/N Training Set. These results are shown in Figure 14 and Figure 15. The discrimination between the two classes can be observed in the mean value as shown in Figure 14, especially from dimensions 40 to 160 of the feature.

In order to obtain a clearer representation, we try to obtain some information from different regions of the images. The images are equally divided into 4 vertical strips and 5 horizontal strips. Figure 16 shows the mean feature values in 4 gradient directions for 4 vertical strips, while Figure 17 gives the mean feature values in 4 gradient directions for 5 horizontal strips. Except for the first vertical strip in Figure 16, the figures show that alphabetic data have larger mean values in all 4 gradient directions than numeric data. This phenomenon means the alphabetic images contain more complicated strokes so that more pixels have gradient feature values. (Since an adaptive threshold has been applied to filter out small gradient values in the images,

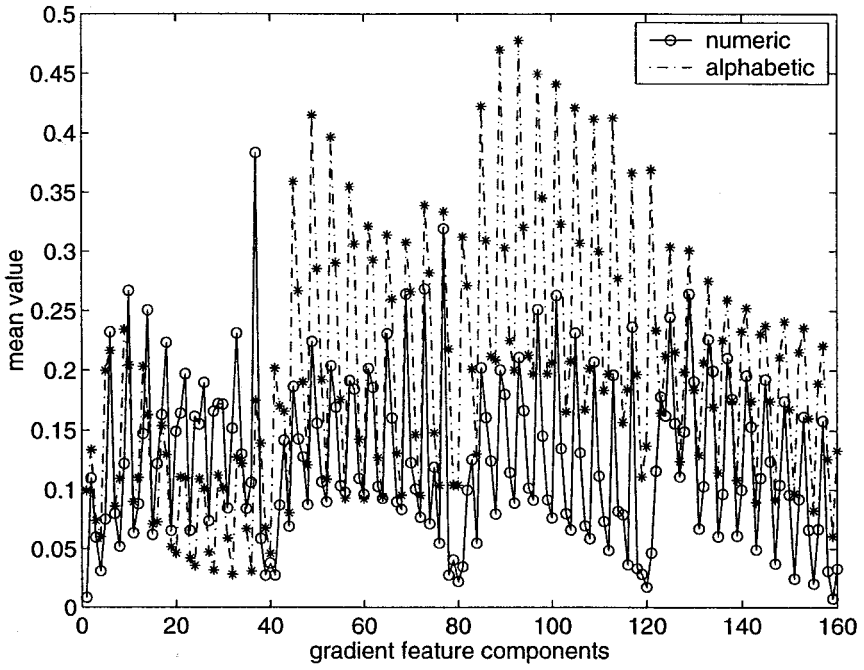

Figure 14: Mean values of gradient feature components

only the information on the contour is used in the feature extraction). The feature information shown in the first vertical strip is different from that of other strips in the images. The reason is that usually capital letters are used in the beginning of month words and capital letters are close to digits in terms of the gradient feature, which means that the gradient feature information in the first vertical strip of the images does not have strong discriminating power for A/N differentiation.

Figures 16 and 17 show that these two classes have different characteristics for different gradient directions in different local regions. Taking another point of view, gradient directions are calculated in different regions, and similar conclusions are obtained. We divide the data images into 20 local regions (4 vertical and 5 horizontal strips) and consider 4 gradient directions. Each gradient direction represents angles

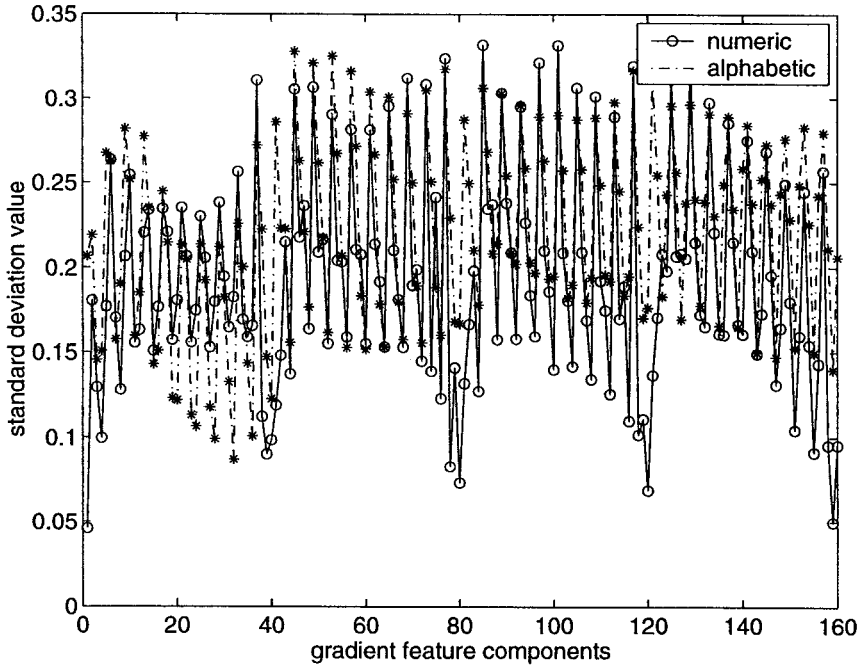

Figure 15: Standard deviations of gradient feature components

in a 90° range, with gradient direction 1 covering the range $(-45°, 45°]$, and so on. For each gradient direction, the mean gradient values for the numeric and alphabetic data in each local region are shown in Figure 18. The results indicate that for each gradient direction, the mean values are higher for alphabetic data except for local regions in the first vertical strip (local regions 1-5).

From these figures, it is clear that the gradient feature is very effective for the A/N differentiation task.

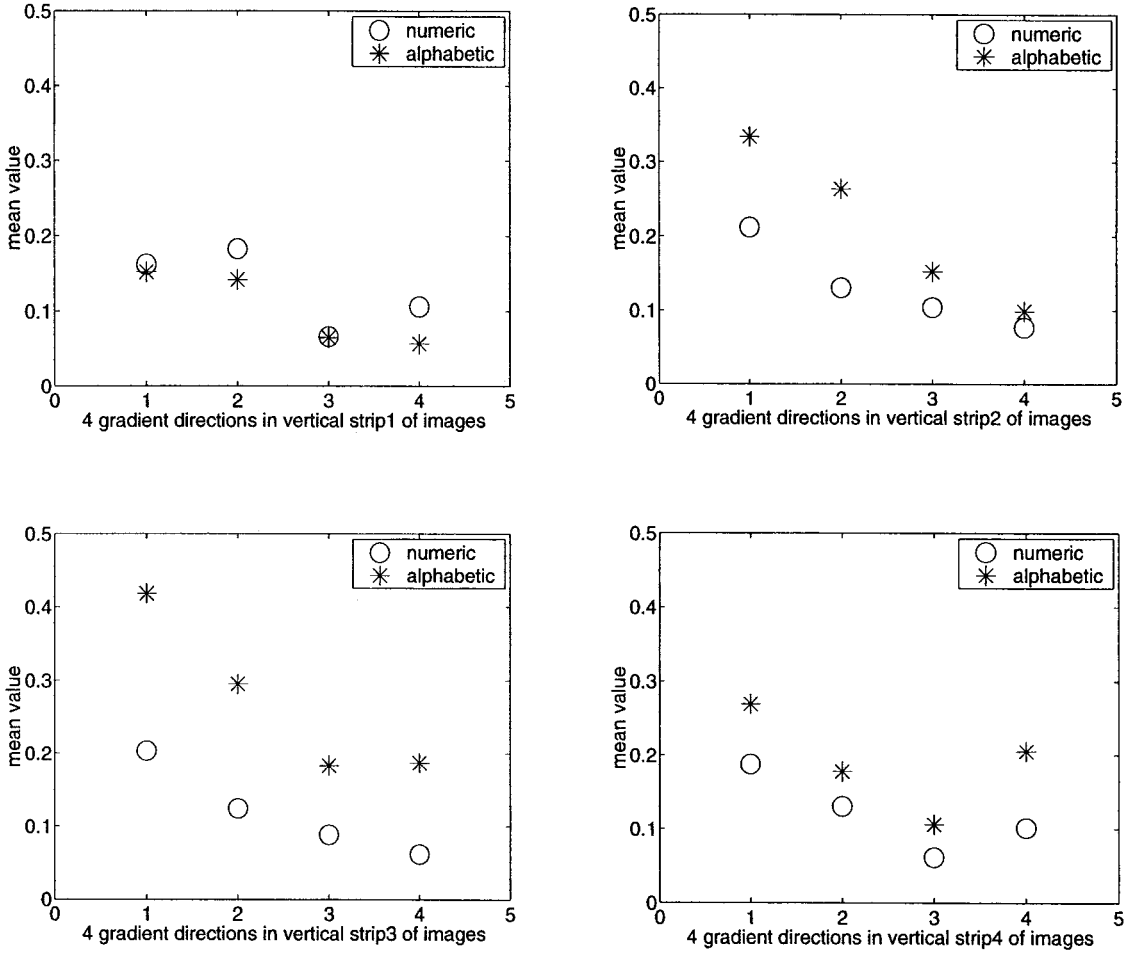

Figure 16: Feature analysis for vertical strips

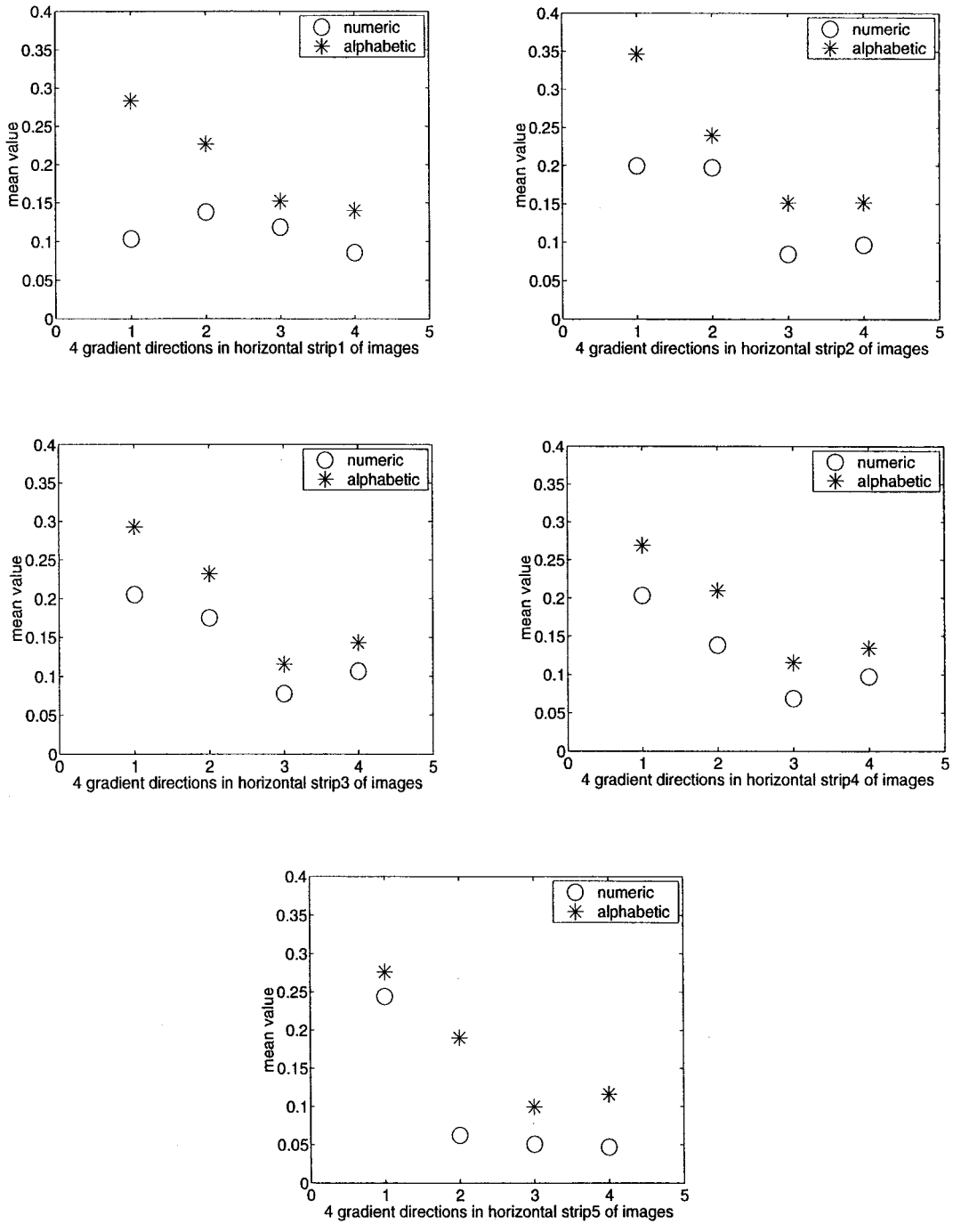

Figure 17: Feature analysis for horizontal strips

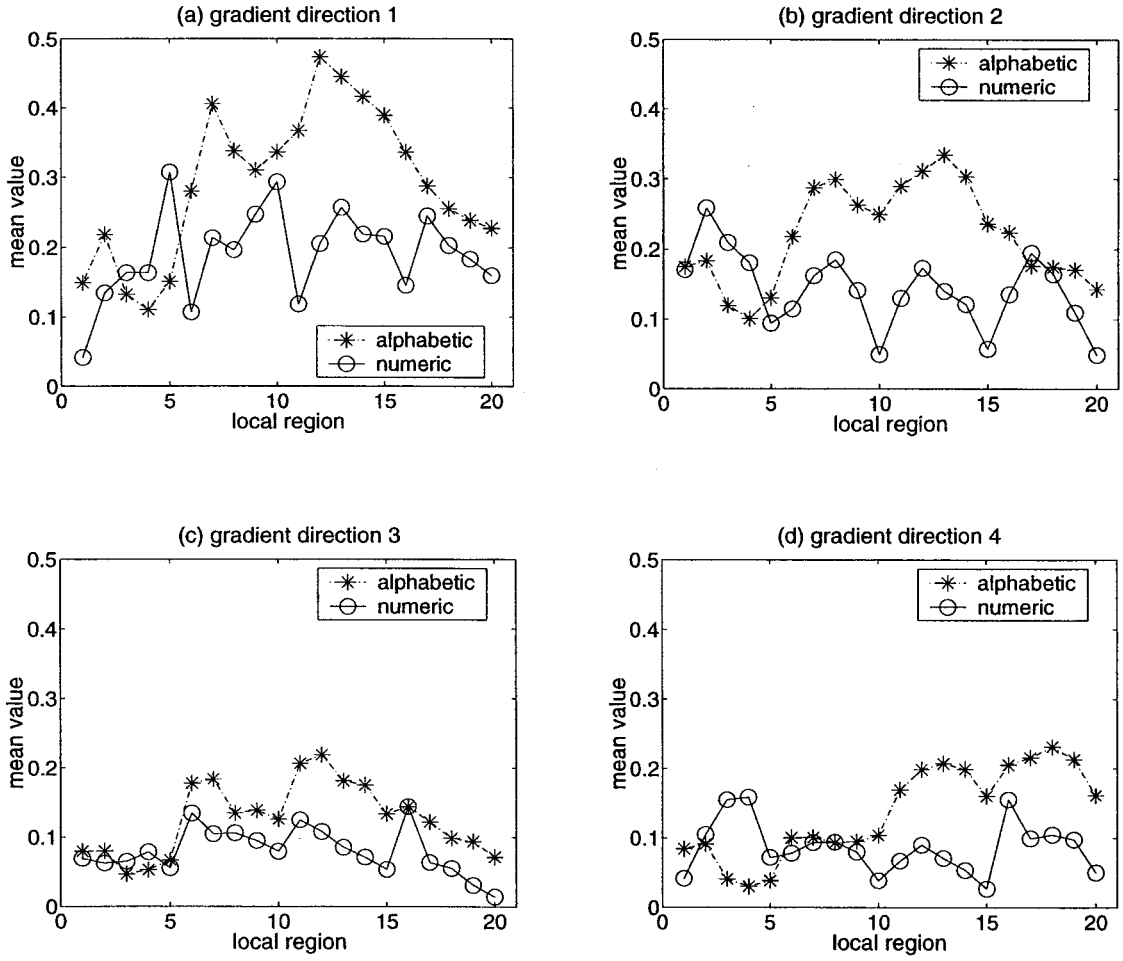

Figure 18: Feature analysis for 20 local regions

(3) Principal component analysis

For a given network architecture with inputs of a given dimensionality, a certain minimum number of training samples would be required to successfully train the network weights [78]. In addition, the feature used should not contain redundant information since this leads to a complex distribution in the feature space.

A principal component analysis (PCA) method using the Karhunen-Loéve expansion [37] has been implemented to reduce the dimensions of gradient and mesh features. This method of dimension reduction is optimal in terms of guaranteeing decorrelation of the feature space and therefore minimizing the squared error of the data approximation [37]. This can be accomplished using a transformation that rotates the coordinate axes so that the resulting covariance matrix of the data becomes a diagonal matrix.

For sample vector $\mathbf{x_i} \in R^d$, $\mathbf{y_i} = \mathbf{x_i} - \frac{1}{n}\sum_{i=1}^{n} \mathbf{x_i}, i = 1, 2, ..., n$. Let $\mathbf{A}$ be $n \times d$ matrix consisting of n row vector $\mathbf{y_i^T}$, with $d \times d$ covariance matrix $\mathbf{C}$. There exists a transformation matrix $\mathbf{U_C}$, that transforms the sample data:

$$B = AU_C^T \tag{18}$$

such that $\mathbf{U_C}$ is an orthonormal basis:

$$U_C^T U_C = \mathbf{I} \tag{19}$$

where $\mathbf{I}$ is the $d \times d$ identity matrix, and the covariance matrix of the transformed data is diagonal:

$$\frac{1}{n} B^T B = U_C C U_C^T = \Lambda_C \tag{20}$$

where $\Lambda_C = diag(\lambda_1, ..., \lambda_d)$. The parameters, $\lambda_i, 1 \leq i \leq d$ are the eigenvalues, and the vectors that make up the rows of $\mathbf{U_C}$, the eigenvectors of the covariance matrix for $\mathbf{A}$.

The eigenvalues measure the scatter of the data in the transformed space, and therefore represent the information that is provided by each dimension of the transformed vectors. These eigenvalues correspond to the amount of total information contained in the original features that is encoded in each dimension of the new features. A reduction in feature dimensionality can be achieved by eliminating those feature dimensions which contribute little information. This reduces the data to m-dimensional vectors, $1 \leq m \leq d$, such that

$$\frac{\sum_{i=1}^{m} \lambda_i}{\sum_{j=1}^{d} \lambda_j}$$

measures the fraction of total information in the original data representation that remains in the reduced representation.

Table 15 shows the performance improvement achieved on A/N Test Set 2 by using the principal component analysis to reduce the dimension of the gradient feature. The 120D is selected to ensure that

$$\frac{\sum_{i=1}^{m} \lambda_i}{\sum_{j=1}^{d} \lambda_j}$$

exceeds a threshold (e.g., 0.95).

Table 15: Performance improvement by using Principal Component Analysis

	Correct (%)	Error (%)
MLP based on gradient feature without PCA (160D: 4 direction, 4 vertical and 10 horizontal divisions) (161-60-3)	97.32	2.68
MLP based on the same gradient feature with PCA (120D) (121-60-3)	97.95	2.05

The MLP with PCA (121-60-3) will replace MLP1 in the following discussions, as it produces the best results so far.

4.3.4 Conclusions

We have discussed the architecture of MLP and the different feature extraction methods which have been used in the A/N differentiation. Theoretical analyses and experimental results show that both the MLP model proposed and the features extracted are effective.

We also find that the gradient feature provides the best performance. Actually the gradient feature has produced very high recognition performances when used by neural network classifiers for handwritten character recognition [107, 19]. A gradient operator emphasizes the object contours, and since much information about object shapes is concentrated on the contours, the features based on gradient information describe the character patterns well. This analysis also holds for the A/N differentiation.

Since the A/N differentiation model is very important in our date processing system, the performance, and especially the reliability, should be very high [153]. The A/N classifier based on the gradient feature can give the following results on A/N Test Set 2, which are the best results obtained so far (see Table 16):

Table 16: Performance of MLP1

	Without Rejection		With Rejection			
	Correct	Error	Correct	Error	Rejection	Reliability
MLP1 (121-60-3)	97.95	2.05	94.81	0.95	4.24	99.01

Rejections are made by comparing the confidence value of the first choice with a threshold. Although this result is good and acceptable, it would be preferable for the reliability to be higher. To further improve the performance, some efforts can be made, e.g. complementary features can be used together to build a MLP network. However, this implies a dimensionality increase of the feature space, which means the

number of input nodes of the neural network would be increased, and the neural network generalization problem has to be considered, as discussed before. Consequently, an alternative approach of combining multiple classifiers is considered.

During the past decade, multiple classifier systems (MCSs) based on combining the outputs of a set of different classifiers have been proposed as a method to develop high performance classification systems [151, 84]. In the neural network field, several methods for creating ensembles of neural networks and methods of combining ensemble members have been investigated [50, 126, 25, 113, 41, 110]. In the following section, the methods for combining neural networks for the A/N classification will be discussed, and it can be seen that better performances have been obtained.

4.4 Combination of MLP Networks for A/N Differentiation

An ensemble of classifiers is a set of classifiers whose individual decisions are combined in some way to classify new samples. A necessary and sufficient condition for an ensemble of classifiers to be more accurate than any of its individual members is that the classifiers are diverse [50], which means they make different errors on data points. Therefore, the key problem of classifier combination is to find methods for creating "error-independent" ensemble members.

4.4.1 Methods for Constructing Ensembles

Many methods for constructing ensembles have been investigated [126, 25]. The main methods in [126] can be included in the following categories: varying the initial random weights, varying the network architecture, varying the network type, and varying the training data. Besides these main methods, Bayesian voting (enumerating the hypotheses) and manipulating the output target methods are reviewed in [25].

Both [126] and [25] indicate that varying the training data is the most frequently used method, while [113] suggests that the order with respect to diversity-generating potential is: net type > training set structure > training set elements > number of hidden units $\approx$ weight initialization.

It has been noted that neural network ensembles could be created using a combination of two or more of the above methods. Therefore, the engineering design of neural network ensembles has been investigated [113, 41, 110]. Two strategies have been proposed: the "direct" strategy [110] and the "overproduce and choose" strategy [113, 41]. The first design strategy aims to generate an ensemble of error-independent nets directly, such as by genetic algorithms. The second strategy is based on the creation of an initial large set of nets and the subsequent choice of the subset of the most error-independent nets. This latter strategy is more practical in the current state of development of MCSs [113, 41], and several methods have been proposed for the selection phase, e.g. clustering [41], "choose the best", and "heuristic picking" methods [113].

Various methods of constructing ensembles have been attempted to produce an optimal combination system for our A/N differentiation task. As mentioned above, the most frequently used method of ensemble construction is to vary the training data, which can be implemented by varying the training samples and varying the features used [151, 54, 18, 146]. Since eight different sets of features have been extracted to build eight MLP networks in Section 4.3, we will first consider the ensembles of these classifiers. The "overproduce and choose" strategy is used here to select the most effective combination of the classifiers. In addition, a bagging (or bootstrapping) method, which is based on varying the training samples, will also be described in this section.

Method of Varying Features

Based on the features described in Section 4.3, eight MLP networks have been built. In order to produce a better performance, we should select a subset of these networks, which will form the most effective combination. The "overproduce and choose" strategy can be used here. The candidate set in the overproduction phase is composed of these eight MLP networks. In the selection phase, two selection techniques will be considered: "choose the best" and "heuristic picking" [113].

For the method of "choose the best", the selection of the "best" individual networks is an obvious strategy, given that the overall system performance is the ultimate goal. However, since the key problem of classifier combination is to have "error-independent" ensemble members, a "heuristic picking" method has been proposed [113] to select a subset with high within-set diversity. This method can be expressed as:

```
Create a group containing the best network
while this group is not big enough do
    for each network n remaining in the candidate set
        score ← 0
        for each pattern p in the test set
            if more than half the networks in the group fail on pattern p and
            network n is correct on pattern p then score ← score−1
            if more than half the networks in the group fail on pattern p and
            network n fails on pattern p then score ← score+1
        end for
        remove network with lowest score from the space and add it to the group
    end for
end while
```

"Heuristic picking" has been shown to be superior to the "choose the best" in [113]. In our experiments, this method also gives good performances especially in terms of

reliability. However, a further refinement can be made. In the **while** loop above, sometimes two networks have identical or very close lowest scores, and the performance of the network with the second lowest score may be much better than that of the network with the lowest score. Since the accuracy of individual classifiers is also useful in the combination, the "heuristic picking" method has been modified as following:

Create a group containing the best network
while this group is not big enough do
    for each network n remaining in the candidate set
        score $\leftarrow$ 0
        for each pattern p in the test set
            if at least half the networks in the group fail on pattern p and network n is correct on pattern p **then** score $\leftarrow$ score-1
            if at least half the networks in the group fail on pattern p and network n fails on pattern p **then** score $\leftarrow$ score$+1$
        end for
        compare the two networks with the lowest scores. Assume N_1 is the network with lowest score $Score_1$ and its recognition rate is R_1, while N_2 is the network with second lowest score $Score_2$ and its recognition rate is R_2.
        if $R_2 - R_1 > threshold_1$ **and** $|Score_2 - Score_1| < threshold_2$,
        then remove network N_2 from the space and add it to the group
        else remove network N_1 from the space and add it to the group
    end for
end while

The experimental results presented in the following section will show that this modified "heuristic picking" method is more effective than the original one.

Bagging

The Bagging algorithm (Bootstrap aggregating) [10] combines classifiers generated by different bootstrap samples (replicates). In this algorithm, a training set containing N cases is obtained by sampling with replacement (bootstrap) N times from the entire training set. The perturbed data set may contain repeats. This procedure can be repeated several times to create a number of different, although overlapping, data sets. Such statistical resampling techniques are particularly useful when there is a shortage of data because classifiers built on small training sets are usually biased or unstable. Bagging can reduce variations caused by limited training data. Another advantage of Bagging is that it enables a systematic construction of ensembles. This training data sampling method is also applied to construct ensembles for our combination system. Experimental results will be given below.

4.4.2 Methods for Combining Classifiers

Many methods have been designed and implemented by various researchers to combine classifiers [140, 77]. For example, the Sum, Product, Maximum, Minimum, Median and Voting rules have been studied extensively [77, 4, 143, 56]. The main advantages of these rules are their simplicity and the fact that they do not require any training. Three of these widely used combination rules have been applied to combine the outputs of individual classifiers in our A/N differentiation module. These are Majority Vote, Ave (Sum) and Product. A Modified Product rule is also used to reduce the veto effect of very low outputs in applying the Product rule.

Although detailed descriptions of these rules can be found in the literature [77, 5], this subsection will present the particular usages of these rules in our application under the assumption of combining three networks N_1, N_2 and N_3. It is necessary to first define the notations used. Let x be an input sample, and $D_k(x)$, where k=1, 2, 3, be the decision on x by network N_k. $C_{k,w_j}(x)$, where k=1, 2, 3 and j=1,2 represents the confidence assigned to class w_j by network N_k, where the confidences used are neural network output node activation levels. $D(x)$ and $C_{w_j}(x)$ are the

final combined result and confidence respectively. In addition, we consider combined results both with and without rejections.

(a) Majority Voting Rule

Case 1: No rejections

If $D_1(x) = D_2(x) = D_3(x) = w_j$,

$D(x) = w_j$, $C_{w_j}(x) = 1.0$ and $C_{w_i}(x) = 0.0$, $i \neq j$

If two of $D_k(x)$, k=1,2,3 are class w_j,

$D(x) = w_j$, $C_{w_j}(x) = \frac{1}{3}\sum_{k=1}^{3} C_{k,w_j}(x)$, $j = 1, 2$

Case 2: With rejections

$C_{w_j}(x)$, j=1, 2 as in Case1.

If $|C_{w_1}(x) - C_{w_2}(x)| <$ threshold,

rejection is made

Else $D(x)$ is determined as in Case1.

(b) Ave (Sum) Rule

$C_{w_j}(x) = \frac{1}{3}\sum_{k=1}^{3} C_{k,w_j}(x)$, $j = 1, 2$

Case 1: No rejections

$D(x) = w_j$, if $C_{w_j}(x) = \max_{i=1}^{2}(C_{w_i}(x))$

Case 2: With rejections

If $|C_{w_1}(x) - C_{w_2}(x)| <$ threshold,

rejection is made

Else $D(x)$ is determined as in Case1.

(c) Product Rule

$C_{w_j}(x) = \prod_{k=1}^{3} C_{k,w_j}(x)$, $j = 1, 2$

Case 1: No rejections

$D(x) = w_j$, if $C_{w_j}(x) = \max_{i=1}^{2}(C_{w_i}(x))$

Case 2: With rejections

If $|C_{w_1}(x) - C_{w_2}(x)| <$ threshold,

rejection is made

Else $D(x)$ is determined as in Case1.

(d) Modified Product Rule

The veto effect [5] in the Product rule is caused by small classifier measurement output values dominating the product, i.e. by giving close to zero values. This effect can be significantly reduced by modifying the output of a classifier if it falls below a specified threshold. A Modified Product (Mproduct) has been presented in [5]. This method has been further refined here and it is described as follows:

If two of $C_{k,w_j}(x)$, k=1, 2, 3 on class w_j are larger than a high threshold (thr1) **and** the other $C_{m,w_j}(x)$ is less than a low threshold (thr2)

$C_{m,w_j}(x) =$ thr2.

else

$C_{k,w_j}(x)$, where k=1, 2, 3 and j=1, 2 remain unaltered.

After the above recomputation of $C_{k,w_j}(x)$, the same steps as shown in the Product method are adopted.

Here the modification of using two different thresholds has been made, and this has resulted in better performances.

4.4.3 Experiments

In this section, both the methods for constructing ensembles and the methods for combining ensembles discussed above will be reported.

Ensembles Construction and Combination

Eight networks (MLP1, MLP2, ...MLP8) have been established in Section 4.3.3 and they serve as the ensemble candidates in the construction stage. Since Majority Vote

is considered to be one of the combination methods, the number of "the best networks" which will be used in the combination stage is chosen to be 3, 5 or 7 because combining an odd number of classifiers by majority vote will generally produce higher recognition rates [86]. The "best" networks are selected by "heuristic picking", modified "heuristic picking" and "choose the best" methods.

(a) Three networks are selected to form the combination group

Based on the performances of the eight individual networks obtained on A/N Test Set 2 in Section 4.3.3, MLP1, MLP2 and MLP8 are selected by both "heuristic picking" and modified "heuristic picking" methods, and by using "choose the best" method, MLP1, MLP2 and MLP6 are selected. Since A/N Test Set 2 has been used to evaluate the performance of individual networks in the previous section, A/N Test Set 2 is being used as a training set for constructing the ensembles. Therefore, A/N Test Set 1 will be the test set for evaluating combination performances (A/N Test Set 1 has also been used to test the effectiveness of the measure $Confid_{\text{numeric}}$ in differentiating between alphabetic and numeric images in Section 4.2). Table 17 shows the performances of the eight individual networks on A/N Test Set 1.

Table 18 and Table 19 show combination performances based on different network selection methods. From these results, we can infer the following:

- Comparing the results from the best individual classifier (MLP1) and those from combination systems, we note that performance improvement of combination systems can be observed specifically in obtaining highly reliable systems (Reliability = Correct/(Correct+Error)). In Table 18, some slight degradations are observed when rejection is not considered. The reason for this phenomenon is that the performance of MLP1 is much better than those of the other two MLP networks combined. However, with regard to reliability all combination systems outperform the best individual classifier.

- Comparing the ensemble construction methods, it can be observed that although "choose the best" can produce better results without rejection, "heuristic picking" is better in building reliable systems. It is also consistent with the

Table 17: Performances of individual networks on A/N Test Set 1

	Correct (%)	Error (%)
MLP1 (gradient feature) (121-60-3)	97.29	2.71
MLP2 (mesh feature) (121-60-3)	95.27	4.73
MLP3 (projection feature) (34-15-3)	92.25	7.75
MLP4 (top & bottom vertical distance feature) (67-30-3)	92.37	7.63
MLP5 (horizontal distance & projection feature) (50-25-3)	93.76	6.24
MLP6 (top vertical distance & projection feature) (67-30-3)	94.64	5.36
MLP7 (bottom vertical distance & projection feature) (67-30-3)	93.59	6.41
MLP8 (top & bottom vertical distance & projection feature) (100-50-3)	94.00	6.00

concept of "heuristic picking" which focuses on the diversity of the ensemble.

- In comparing the combination methods, it is noted that Ave (Sum) is the most stable method whether rejections are considered or not. The Product rule can give better results than Ave as shown in Table 19, but it can be the worst among the four methods as shown in Table 18 when rejections are considered. The veto effect can be the reason for this case. Modified Product can produce the best results when the two thresholds are carefully set; however, this requires training and may introduce generalization problems. In fact, other researchers have arrived at similar conclusions about these combination rules [77, 56, 5].

- Since the reliability of the A/N differentiation model is the most important aspect in our date processing system, the modified "heuristic picking" method and Ave combination rule are selected for our combination system.

Similar results can also be observed when the same experiments are carried out on A/N Test Set 2 which has been used as the training set for constructing the ensembles

Table 18: Performance of combining MLP1, MLP2 and MLP8 (selected by "heuristic picking" and modified "heuristic picking" method) on A/N Test Set 1

	Without Rejection		With Rejection			
	Correct	Error	Correct	Error	Rejection	Reliability
Majority Vote	97.20	2.80	93.91	0.78	5.31	99.18
Ave (Sum)	97.25	2.75	93.91	0.78	5.31	99.18
Product	97.61	2.39	93.64	0.85	5.51	99.10
Modified Product	97.64	2.36	94.08	0.80	5.12	99.15
MLP1	97.29	2.71	93.93	1.05	5.02	98.90

Table 19: Performance of combining MLP1, MLP2 and MLP6 (selected by "choose the best" method) on A/N Test Set 1

	Without Rejection		With Rejection			
	Correct	Error	Correct	Error	Rejection	Reliability
Majority Vote	97.32	2.68	93.91	0.97	5.12	98.97
Ave (Sum)	97.49	2.51	93.91	0.97	5.12	98.97
Product	98.03	1.97	94.08	0.95	4.97	99.00
Modified Product	97.73	2.27	94.08	0.92	5.00	99.02
MLP1	97.29	2.71	93.93	1.05	5.02	98.90

(as shown in Tables 20 and 21).

(b) Five networks are selected to form the combination group

When five networks are considered, MLP1, MLP2, MLP3, MLP5 and MLP8 are selected by "heuristic picking" method, while MLP1, MLP2, MLP5, MLP6 and MLP8 are selected by modified "heuristic picking" and "choose the best". A comparison of their performances is shown in Table 22, where only the Ave combination rule is applied.

Some information can be obtained from the results presented in Table 22:

- Combination systems of 5 "best" networks are less effective than the systems

Table 20: Performance of combining MLP1, MLP2 and MLP8 (selected by "heuristic picking" and modified "heuristic picking" methods) on A/N Test Set 2

	Without Rejection		With Rejection			
	Correct	Error	Correct	Error	Rejection	Reliability
Majority Vote	97.93	2.07	94.76	0.80	4.44	99.16
Ave (Sum)	97.98	2.02	94.76	0.80	4.44	99.16
Product	98.24	1.76	94.59	0.80	4.61	99.16
Modified Product	98.12	1.88	95.12	0.91	3.97	99.06
MLP1	97.95	2.05	94.81	0.95	4.24	99.01

Table 21: Performance of combining MLP1, MLP2 and MLP6 (selected by "choose the best" method) on A/N Test Set 2

	Without Rejection		With Rejection			
	Correct	Error	Correct	Error	Rejection	Reliability
Majority Vote	97.73	2.27	94.98	0.83	4.19	99.13
Ave (Sum)	97.76	2.24	94.98	0.83	4.19	99.13
Product	98.22	1.78	94.71	0.78	4.51	99.18
Modified Product	97.98	2.02	95.17	0.81	4.02	99.16
MLP1	97.95	2.05	94.81	0.95	4.24	99.01

obtained by combining 3 "best" networks (as shown in Tables 18 and 19), especially in terms of reliability. The reason is that much lower performing networks are included when five networks are selected.

- Modified "heuristic picking" performs better than original "heuristic picking". In fact, when two networks can contribute similar increase in diversity, the performances of the two networks should be considered. This is the reason that Modified "heuristic picking" is superior to the original "heuristic picking".

- Since combining five networks does not improve the performance obtained from combining three networks, we made no further attempt to combine more networks.

Table 22: Performance of combining 5 classifiers selected by "heuristic picking", and by modified "heuristic picking" and "choose the best" on A/N Test Set 1

Combination	Without Rejection		With Rejection			
	Correct	Error	Correct	Error	Rejection	Reliability
MLP1, MLP2, MLP3, MLP5 and MLP8	96.93	3.07	93.71	1.44	4.85	98.49
MLP1, MLP2, MLP5, MLP6 and MLP8	97.29	2.71	94.15	1.19	4.66	98.75

Bagging

Three training sets are first obtained by sampling with replacement to generate multiple MLP networks, and four combination methods are used to obtain the final results. Two experiments have been conducted (as shown in Table 23 and Table 24), one of which is based on the gradient feature used in MLP1 and the other is based on the features used in MLP6. (MLP1 and MLP6 are the networks with the best performances).

Table 23: Bagging performance (gradient feature 120D) on A/N Test Set 1

	Without Rejection		With Rejection			
	Correct	Error	Correct	Error	Rejection	Reliability
Majority Vote	96.25	3.75	94.00	2.39	3.61	97.52
Ave (Sum)	96.32	3.68	94.00	2.39	3.61	97.52
Product	96.15	3.85	94.08	2.39	3.53	97.52
Modified Product	96.25	3.75	94.44	2.51	3.05	97.41
MLP1	97.29	2.71	93.93	1.05	5.02	98.90

Some analyses of the results are presented here:

- Table 23 shows the performance of Bagging is worse than the performance of

Table 24: Bagging performance (top vertical distance and projection features 66D) on A/N Test Set 1

	Without Rejection		With Rejection			
	Correct	Error	Correct	Error	Rejection	Reliability
Majority Vote	94.95	5.05	91.98	3.07	4.95	96.77
Avr (Sum)	95.00	5.00	91.91	2.97	5.12	96.87
Product	95.08	4.92	92.03	3.02	4.95	96.82
Modified Product	95.05	4.95	92.03	3.02	4.95	96.82
MLP6	94.64	5.36	91.76	3.27	4.97	96.20

the individual classifier MLP1 using the same feature set. The reason is that the performances of classifiers built from the bootstrap samples are much worse than that of MLP1 (as shown in Table 25).

Table 25: Performances of classifiers built from the three bootstrap samples

	Without Rejection		With Rejection			
	Correct	Error	Correct	Error	Rejection	Reliability
Classifier 1	95.32	4.68	93.13	3.21	3.66	96.66
Classifier 2	95.83	4.17	93.42	2.73	3.85	97.12
Classifier 3	95.59	4.41	93.20	2.92	3.88	96.96
MLP1	97.29	2.71	93.93	1.05	5.02	98.90

This performance degradation can arise because each bootstrap sample contains only about 63.2% unique instances from the training set [7], so that effectively smaller training sets are used for training each classifier while the feature dimension is 120D, which is high compared with the size of training set.

- Results in Table 24 indicate that improvements can be made by using Bagging on MLP6. However, currently the combination of classifiers built on complementary features can produce better results in our A/N differentiation module, especially in terms of reliability.

- Results in Tables 23 and 24 also show the Ave is the most stable rule among the four combination schemes.

So far, based on extensive experiments conducted, the best combination system for A/N differentiation has been obtained when the modified "heuristic picking" method and Ave combination rule are selected, and a high reliability of 99.18% has been achieved with a recognition rate of 93.91%. Compared with the method based on the *distance_to_numeral* measure, the combination system gives a more reliable performance, and can be fully controlled. However, the method based on the *distance_to_numeral* measure has its advantages. In particular, the use of a digit recognizer in this method does bring more information such as recognition results, and this information can be very useful in some applications, which will be elaborated in the following chapter.

As discussed in Section 4.1, the A/N differentiation module is very useful in the date segmentation stage. In the following chapter, we will present our date segmentation module and the applications of these two A/N differentiation methods in the segmentation stage.

Chapter 5

Date Image Segmentation

5.1 Segmentation Strategy

5.1.1 Overview

The purpose of date image segmentation is to divide the entire image into three subimages corresponding to *Day*, *Month* and *Year* respectively, and also make a decision on how *Month* is written so that it can be processed using an appropriate recognizer. Since there is no predefined position for each field, and no uniform or even obvious spacing between the fields, it is difficult to implement the segmentation process with a high success rate by using simple structural features.

Figure 19 shows an overview of the date segmentation module. The first step is to separate *Year* from *Day&Month*. Based on the database analysis given in Chapter 3, about 80% of date zones have machine-printed "19" (or "20") to indicate the *Year* field, and even if no machine-printed "19" (or "20") appears, *Year* is located at one of the two ends of the date zone. In addition, we can assume *Year* should belong to one of the two patterns: 19** (or 20**) and **, where * is a numeral. Therefore, compared with *Day* or *Month* detection, *Year* detection is an easier task. A set of rules has been developed to detect *Year* field based on structural features and the characteristics of the *Year* field.

In the *Day&Month* segmentation stage, a knowledge-based segmentation module

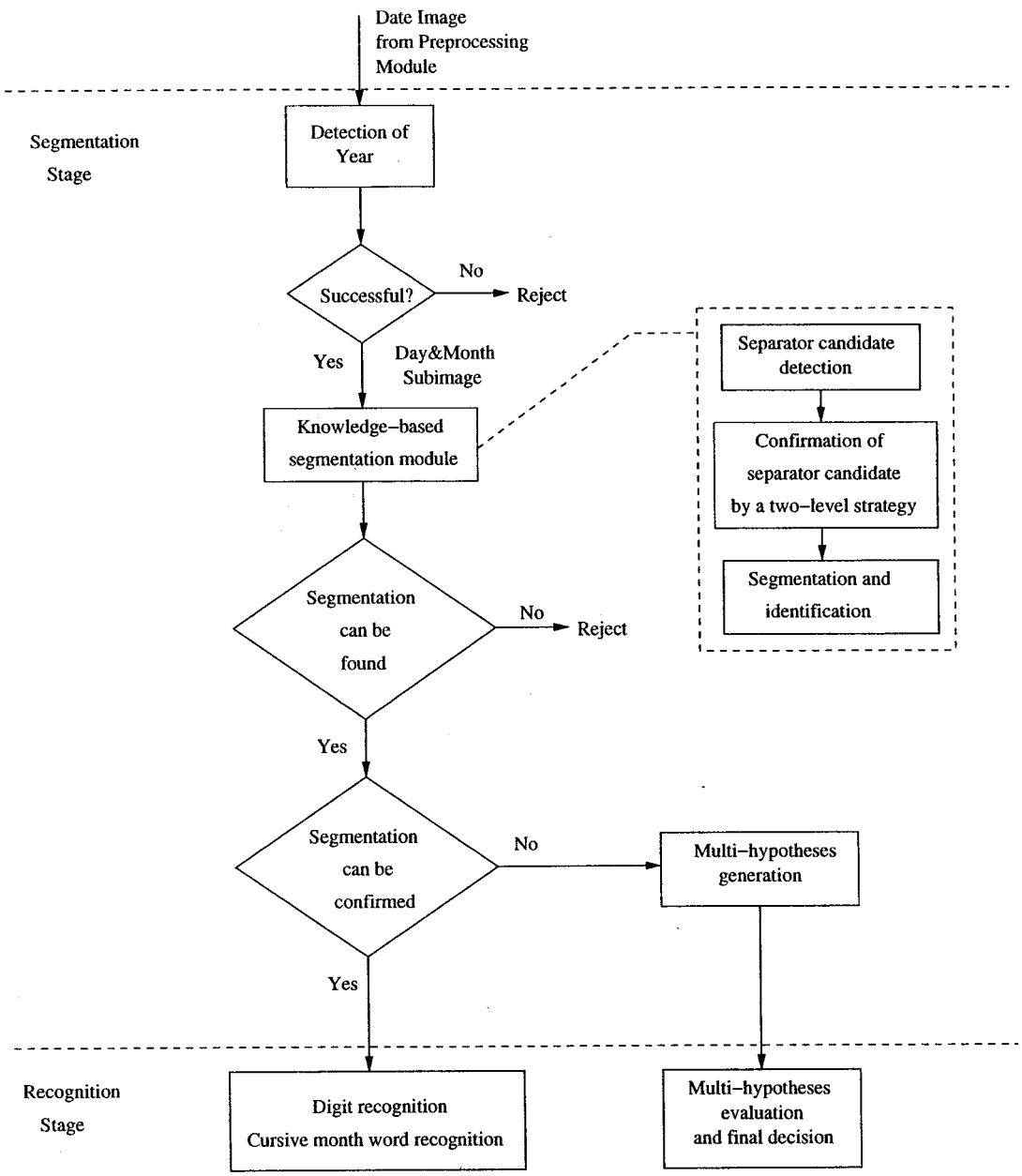

Figure 19: Diagram of date segmentation module

is used first to tackle most segmentation cases. Ambiguous cases are handled by a multi-hypotheses generation module at this stage, which puts multiple hypotheses into a list. Each hypothesis in the list consists of a possible segmentation of *Day&Month* field, and these hypotheses will be considered in the recognition stage by a multi-hypotheses evaluation module.

The task of the knowledge-based segmentation module includes (i) detection of the separator between *Day* and *Month*, and (ii) segmentation and identification. The detection of the separator is realized by two steps: the detection of candidates and the confirmation of candidates. In the segmentation and identification stage, based on the separators detected, a set of rules has been developed to segment the *Day&Month* field into *Day* and *Month* fields and to determine whether the month field is written in numeric or alphabetic form. In this knowledge-based segmentation module, if it is difficult to make a decision with high confidence in the separator detection stage or in the segmentation and identification stage, the multi-hypotheses generation procedure is invoked. In addition, some contextual knowledge about *Day&Month* fields are extracted and used in this knowledge-based segmentation module to improve its performance. More explanations about the extraction and use of this contextual knowledge are given below.

5.1.2 Knowledge-Based System

In our *Day&Month* segmentation module, the detection of the separator is accomplished by a candidate_then_confirmation strategy. Separator candidates are first detected by structural features. While some of the candidates with high confidence values of the features can be confirmed immediately at the first confirmation level, others should be evaluated by considering more information, i.e. be confirmed at the second confirmation level. In the second confirmation level, some contextual knowledge about the writing styles is used.

Based on the database analyses in Chapter 3, some relationship between the type of separator and the writing style has been found, e.g. approximately 98% of *Month_in_digit* date images have punctuations. So some separator candidates are

easily confirmed or rejected using a set of rules if the knowledge about the writing style can be obtained. Furthermore, these rules can be designed in the training stage based on human knowledge and syntactic constraints, and in general they can be encoded in a pattern based grammar. Here the pattern is image pattern, and usually a *Day&Month* field can appear in any one of the three patterns: NSN, NSA and ASN, where S denotes the separator (slash, hyphen, period, comma or gap), N denotes the numeric string (*Day* or *Month* field) and A denotes the alphabetic string (*Month* field). The pattern based grammar used to detect the separator can be expressed as:

Given image Pattern:
If $<$ *condition* $>$ then $<$ *action* $>$
$<$ *condition* $>\Rightarrow$ the separator candidate (S) in the image Pattern is P, $P \in \Sigma$
$<$ *action* $>\Rightarrow$ Confidence Adjustment|
Add New Feature|
Adjust Segmentation Method|...

Here Σ represents the set of separator types. Some explanations to the pattern based grammar are given in Table 26. When we try to confirm a separator candidate and the "Condition" is that the separator candidate is a "/" or "–" or..., we can take the corresponding "Action".

Table 26: Examples of condition-action rules

Pattern	Condition	Action
NSN	–	"–" candidate is confirmed
NSA	–	Straight_enough feature is checked
ASA	NULL	Separator candidate is not confirmed
...	...	...

For this knowledge-based method, two questions should be asked: (1) how can the writing style be determined? and (2) how can the knowledge about the relationship

between the type of separator and the writing style be used to determine the correct separator?

As discussed in Chapter 4, two methods have been developed to determine the writing styles. One method is based on a *distance_to_numeral* measure, while the other is an A/N differentiation module that combines results from multiple neural networks. The effectiveness of the two methods have been proven in the experiments. However, their results may be inconclusive for some ambiguous segmentation cases, where the multi-hypotheses generation and evaluation are introduced.

In the following, according to the processing order of our date segmentation system, the detection of *Year* field is presented first in Section 5.2, and the segmentation of *Day&Month* field is presented in Sections of 5.3, 5.4 and 5.5.

5.2 Detection of *Year* Field

As discussed in Chapter 3, the writing styles of date fields extracted from Canadian bank cheques can be grouped into two categories: standard format and free format. For the standard format, the century symbol ("19" or "20") is printed on the date zone, and for the free format, no machine-printed century symbol appears on the date zone. In this section, the method for detecting *Year* field will be presented focussing on the approach of *Year* field detection for the free format.

The basic steps in the *Year* detection module are shown in Figure 20. In this module, *Year* with the standard format is first detected. If the detection is not successful, *Year* with the free format is detected. The method of *Year* detection for the standard format can be designed to detect the machine-printed "19" or "20". However, when "19" or "20" is not printed on a date zone, that is, the free format is adopted, the task of *Year* field detection becomes difficult.

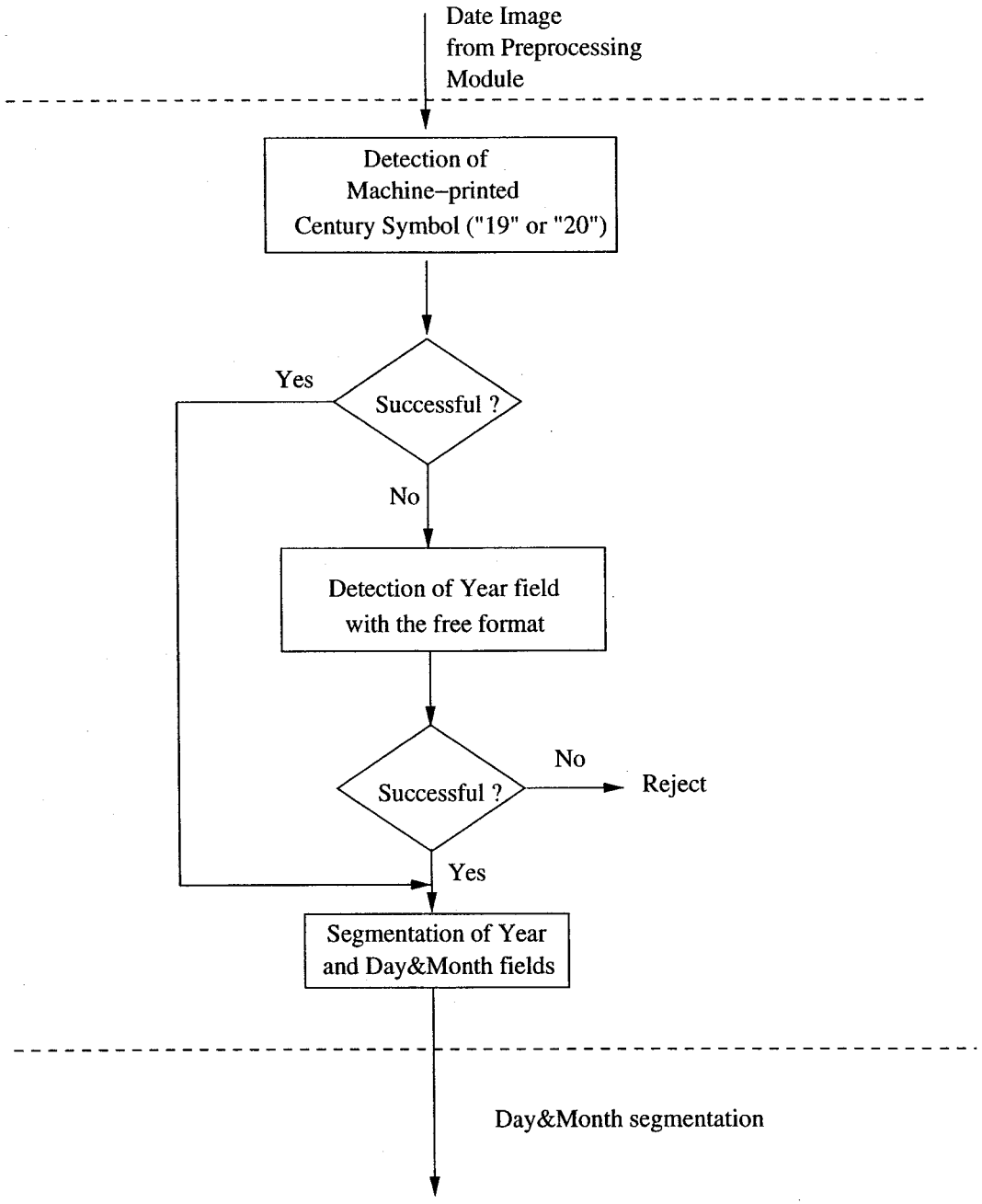

Figure 20: Diagram of *Year* field detection

5.2.1 Detection of Machine-Printed Century Symbol ("19" or "20")

It is observed that on the date zone with the standard format, numerals '1' and '9' (or '2' and '0') are printed as separate characters, and each numeral consists of one connected component. In addition, the two adjacent numerals '1' and '9' (or '2' and '0') have the same height. All these features can be used to detect the machine-printed century symbol. However, in practice, real bank cheques can be printed with various background images and colors, and many different kinds of noise may be introduced to the binary date images extracted from the cheques. Consequently, the printed '1' and '9' (or '2' and '0') may touch each other, or either of them can touch its neighbor, and a numeral may be broken into more than one connected component, etc.. All these problems make the detection of machine-printed century symbol difficult.

However, based on our database analyses, we find that most of the date zones with the standard format contain two separated horizontal line segments (to underline the handwritten *Day&Month* and *Year* fields), and the printed century symbol "19" (or "20") is positioned in the gap between the two line segments (refer to Figure 5). Therefore, if the coordinates of this gap are obtained in the binarization and extraction procedure of a date zone, the machine-printed century symbol is assumed in the gap, and *Day&Month* and *Year* subimages are obtained by simply cutting the date image using these coordinates. Based on Statistical Set used in Section 3.1.3, more than 95% of date zones with the standard format have these two horizontal lines, so most of the *Year* fields with the standard format are detected by this method in our date processing system.

If there is only one horizontal line or no line printed in the date zone, another procedure has to be used to detect the printed century symbol. In our current *Year* detection module, this procedure is simple, and it can only deal with the cases in which the printed century symbol consists of two isolated numerals, and each numeral consists of one and only one connected component. A method for detecting printed

"19" has been presented in our previous date processing system [32]. For the detection of printed "20", a method is presented here. Starting from the right end of a date zone image, all the connected components are examined one by one until the middle point of the date zone is reached. In this search procedure, if a component contains one and only one inner loop, and its left neighbor has approximately the same height, these two adjacent connected components are taken as a candidate of a printed "20", and is processed by a digit recognizer [133]. If the recognition result for the candidate is "20", the right part of the "20" candidate on the date zone (which should be the *Year* field) is recognized by the digit recognizer. If this part can be recognized as two numerals with high confidence, the candidate for the printed "20" is confirmed as the century symbol. This procedure has been used for handwritten dates on real cheques, and it has been found to be effective.

5.2.2 Detection of *Year* Field with the Free Format

In this section, before presenting the method for detecting *Year* fields with the free format, some analyses of *Year* fields are given. The information obtained from the analyses has been used to develop our *Year* field detection module.

Analysis of the *Year* Field

In Chapter 3, the general expression of the free format was given as:

field1 S field2 S field3

In order to develop an effective *Year* detection method based on data analyses, the free format can be further grouped and expressed as:

(a) *Year - Month - Day*

(b) *Day* (or *Month*) - *Month* (or *Day*) - *Year*

(c) *Year / Month / Day*

(d) *Day* (or *Month*) / *Month*(or *Day*) / *Year*

(e) *Day&Month*, (or .) *Year*

(f) *Day&Month/* (or -) *Year*

(g) *Year*, (or .) *Day&Month*

(h) *Day&Month Year*

(i) *Year Day&Month*

Here *Year* can be 4 numerals (19** or 20**), or 2 numerals (**). When we built our database we asked the writer to fill in a desired year with any format (if no century symbol was printed on the date zone), and the numeral * can be any numeral from 0 to 9. In addition, we find that if *Year* is written at the left end of a date zone, *Month* is always written adjacent to the *Year* field (as in case (a)). All the above writing styles are found to exist in our database. More information about the *Year* field is obtained from analyses of Statistical Set (see Chapter 3) and is shown in the following tables:

Table 27: Usage of the Separator between *Year* and *Day&Month*

	Number of Images	Percentage (%)
Total no. (Date image with the free format)	1165	100
Gap	561	48.15
Punctuation	604	51.85

Table 27 and Table 28 show some general information about the *Year* field and the separator between *Year* and *Day&Month* fields. Table 29 gives some information about the usage of punctuations. The "others" in the table means ':' is used, which is not a normal case. In addition, we find that when two '/' or two '-' are used as the separators between the fields, 99.01% of the date images have *Month_in_numeral.*

Table 28: Position of *Year* field

	Number of Images	Percentage (%)
Total no.	1165	100
Right end	932	80.00
Left end	233	20.00

Table 29: Usage of punctuation

	Number of Images	Percentage (%)
	604	100
Two '/' (between *Day* and *Month*, and between *Day&Month* and *Year*)	289	47.85
Two '-' (between *Day* and *Month*, and between *Day&Month* and *Year*)	216	35.76
One '/' or '-' (between *Day&Month* and *Year*)	22	3.64
',' or '.' (between *Day&Month* and *Year*)	74	12.25
others	3	0.50

Overview of *Year* Detection Method

Although we know that *Year* is located at one end of the date zone, and that it should belong to one of the two patterns: 19** (or 20**) and **, it is difficult to detect it by just using a digit recognizer because we have to answer the question of how many numerals (four or two numerals) are included in the *Year*, and at which end of the date zone the *Year* is located. Therefore, our *Year* candidate detection method is based on separator detection (the separator between *Year* and *Day&Month* fields), and it is based on the following assumptions:

- *Year* is located at one end of the date zone.
- *Year* contains either 4 numerals starting with "19" (or "20") or 2 numerals.

- Separator (big gap or puctuation) is used by writers to separate *Year* field from *Day&Month* field.

The strategy used to detect *Year* field is a candidate_then_confirmation method. Based on the assumptions given above, *Year* candidates are first detected from the two ends of the date zone, and then the confirmation is implemented by using the assumption that a *Year* field contains either 4 numerals starting with "19" (or "20") or 2 numerals. Here *Year* candidates are obtained by detecting the separator (a punctuation or a big gap) between *Year* and *Day&Month* fields.

According to the analyses about date images with the free format, we find that when punctuations are used to separate *Year* fields from *Day&Month* fields, about 85% of date images contain two slashes or two hyphens, and *Day*, *Month* and *Year* fields are separated by the two punctuations. Therefore, if two slashes or two hyphens can be found in a date zone, the *Year* candidates can be easily detected from the two ends of the date zone. In addition, if *Year* candidate detection is based on finding two punctuations of the same type (two slashes or two hyphens), the result obtained from this procedure is considered reliable. Consequently, the first step of our detection module for the *Year* field with the free format is based on the detection of two punctuations. If a *Year* is not found by this step, we try to detect the *Year* from the right end of the date zone (based on our database analyses, 80% of the date images have *Year* at the right end), and then at the left end of the date zone.

The basic steps for detecting *Year* fields with the free format are shown in Figure 21. The idea of this *Year* detection module is to detect *Year* candidates from the two ends of a date image, and then to determine the *Year* from the candidates in the confirmation stage. The *Year* candidate detection is based on the identification of a separator (between *Year* and *Day&Month* fields), and the confirmation is implemented by using the characteristics of *Year* fields. In the following, the separator (punctuation or gap) detection is first addressed, and then the types of *Year* field candidates are discussed, which will be used in the confirmation stage. Finally, the detailed steps for detecting the *Year* field with the free format will be given.

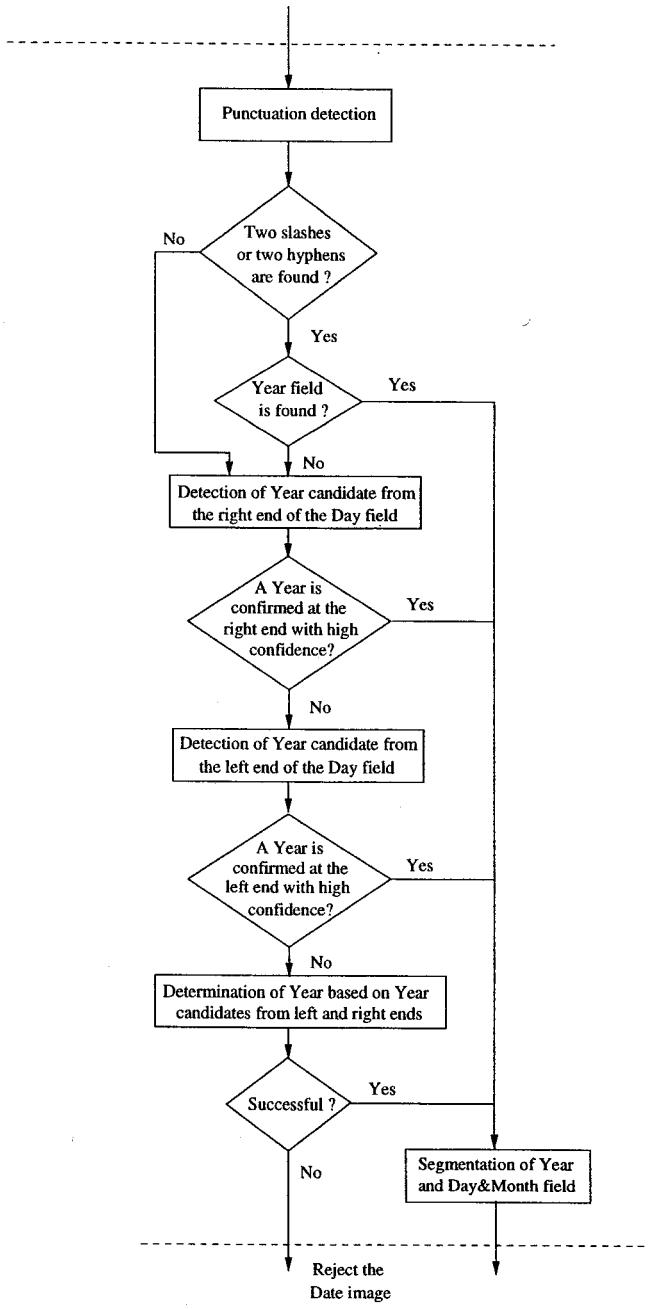

Figure 21: Diagram for detecting the *Year* field with the free format

Separator Detection

In our *Year* detection module, we assume a separator is used by the writer to separate a *Year* field from a *Day&Month* field. The separator can be a punctuation ('/', '-', '.', or ',') or a gap. Therefore, the separator detection is the basis of the *Year* detection.

1. Puctionation detection

The features used to describe punctuations can be divided into shape and spatial features [33, 125]. In [33], some shape features and spatial features were discussed, and different feature combinations were given to detect the punctuations:

- *narrow and exceed_neighbor for slash*
- *high_density, flat, at_middlezone, and mid_to_neighbor for hyphen*
- *high_density, small and below_lowerhalf for period*
- *narrow, small, and below_lowerhalf for comma*

Similar to the procedure for punctuation detection in [32], the procedure for punctuation detection in our *Year* detection module can be described as follows:

1. *No_innerloop* and *simple_curve* features are first applied to exclude unlikely candidates.
2. Only those components with a height greater than the height of the image middle zone are considered for slashes. Only those having the low_to_left feature are tested for periods and commas. Other candidates are checked for hyphens only.
3. Estimate the likelihood of a connected component being a certain punctuation. For slash, period and comma, we use the average confidence value obtained from each feature set. Different weights, 1 for *high_density* and $flat$; 0.5 for

at_middlezone and *mid_to_neighbor*, are used to obtain the average confidence value for hyphen.

4. A punctuation is detected when its confidence value as a punctuation is higher than a threshold, which is established in the training stage.

2. Gap detection

When no punctuation is used to separate a *Year* field from a *Day&Month* field, it has been assumed that a big gap is used by the writer to separate the two fields. This gap is called a word gap here because it separates two words corresponding to *Year* and *Day&Month* fields respectively. It is possible that two word gaps are used in a date image to separate the three fields of *Day*, *Month* and *Year*. For detecting the word gaps, a central subproblem is the estimation of inter_component distances. Here we choose two gap metrics, one based on horizontal bounding box distance [125] and the other based on the method in [32], which uses the number of maximum distance occurrences (when scanning all lines within the middle zone of a subimage) to locate the gap between a *Day* field and a *Month* field. For convenience, we abbreviate this measure by NMD and bounding box distance by BBD.

Locating the word gap between a *Year* field and a *Day&Month* field is a difficult task because the users may write the cheques in such free styles that the gap does not always appear as the observed wide word gap in the date zone. Therefore, in our *Year* detection module, we choose four gap candidates, that is, gap_{BBD_1}, gap_{BBD_2}, gap_{NMD_1} and gap_{NMD_2}, where BBD_1, BBD_2, NMD_1 and NMD_2 are the two maximum distances obtained from each of the two gap estimation methods. If BBD_1 (BBD_2) is less than a threshold, gap_{BBD_1} (gap_{BBD_2}) is not considered as a gap candidate, and the number of gap candidates will be less than four. Based on these gap candidates, *Year* candidates located at the both ends of date images can be obtained, and the *Year* field will be detected from the candidates in the confirmation stage.

Types of *Year* Field Candidates

As discussed before, the strategy used to detect *Year* fields with the free format is a candidate_then_confirmation method. When *Year* candidates are obtained from the

two ends of a date zone, the *Year* field is detected by selecting a candidate with the highest likelihood to be a reasonable *Year* field. In this subsection, we give some discussions about the meaning of a reasonable *Year*, and the likelihood of an end field of a date zone to be a *Year*. We define 4 types of end fields:

I A numeric string with 19** (or 20**) pattern returned from a digit recognizer [133] with high confidence value, and the horizontal bounding box distances between the components in this field are less than a threshold. This distance threshold is used to distinguish a *Year* containing 4 numerals starting with "19" or "20" from a combination of a *Day* with value "19" or "20" and a *Year* (or a *Month*) consisting of two numerals.

II A numeric string with length of 2 returned from the digit recognizer with high confidence value and the value of the string is larger than 31 or equal to 00.

III A numeric string with length of 2 returned from the digit recognizer with high confidence value and the value of the string is at most 31.

IV Other types of numeric strings returned from the digit recognizer.

From this definition, obviously the order of the confidence values for the 4 types being *Year* is from I to IV (from the highest value to the lowest value). In fact, for type III, we have to check the other fields to decide if it is a *Year* field because it may be a *Day* (or *Month*), and type IV usually means the candidate is not a reasonable *Year* field based on the recognition results from the digit recognizer.

Procedures for *Year* Detection

The basic steps for detecting *Year* fields with the free format have been given in Figure 21. In this subsection, the detailed steps for detecting the *Year* fields are presented. Based on Figure 21, the main procedure for the *Year* detection is as follows:

1. Call *GetYearFrom2Punc* procedure, which searches the entire date image and detects hyphen and slash puctuations. If two slashes or two hyphens are found,

and reasonable *Year* can be obtained from one end of the date image, return the *Year* detected. Otherwise, go to the next step.

2. Call *GetYearFromRight* procedure. If *Year* can be detected with high confidence value from the right end of the date image, return the *Year* detected. Otherwise, return a *Year* candidate from the right end or FALSE (no candidate can be found), and go to the next step.

3. Call *GetYearFromLeft* procedure. If *Year* can be detected with high confidence value from the left end of the date image, return the *Year* detected. Otherwise, return a *Year* candidate from the left end or FALSE (no candidate can be found), and go to the next step.

4. Compare the two *Year* candidates from the right and left ends of the date image. If a reasonable *Year* can be obtained, return the *Year* detected. Otherwise, return FALSE (*Year* can not be detected) and reject the date image.

More discussions about the procedures used are given below:

- *GetYearFrom2Punc*

  This procedure is used to detect *Year* fields when two slashes or two hyphens are used to separate the three fields of *Day*, *Month* and *Year*. Based on the method of separator detection given above, if two slashes or two hyphens are detected in a date zone, two *Year* candidates from the two ends of the date zone can be obtained. According to the types of the two end fields, a field is selected from the two candidates to be the *Year* field if it has higher likelihood to be a reasonable *Year*. If no reasonable *Year* can be detected based on the two candidates, this procedure return FALSE, and go to the next step of calling *GetYearFromRight* procedure.

  1. The first step in *GetYearFrom2Punc* procedure is hyphen and slash candidate detection for a date image.

2. If two hyphen or two slash candidates can be found, the three fields separated by the two candidates are processed by the digit recognizer. Here we assume that these three fields are written in numerals (99.01% of date images using two such punctuations have *Month_in_numeral*). Therefore, the recognition results returned from the digit recognizer for the three fields are checked. If a confidence value is less than a threshold, or the string length of the middle field is larger than 2, or the string length of one of the end fields is not equal to 1 or 2 or 4 (only one of the end fields can have string length of 4), the punctuation candidates can not be confirmed.
3. In order to decide the location of *Year*, we use the following procedure to search the *Year* from the two candidates at the two ends of the date image:
   (a) Step 1: Check the right end. If the field is of type I, it is returned as the *Year*; otherwise go to the next step.
   (b) Step 2: Check the left end. If the field is of type I, it is returned as the *Year*; otherwise go to the next step.
   (c) Step 3: Check the right end. If the field is of type II, it is returned as the *Year*; otherwise go to the next step.
   (d) Step 4: Check the left end. If the field is of type II, it is returned as the *Year*; otherwise go to the next step.
   (e) Step 5: Check the right end. If the field is of type III, check the left end. If the left end field is of type IV, the right field is returned as the *Year*; otherwise go to the next step.
   (f) Step 6: Check the left end. If the field is of type III, check the right end. If the right end field is of type IV, the left field is returned as the *Year*; otherwise go to the next step.
   (g) Step 7: No reasonable *Year* can be found, return FALSE to indicate no *Year* candidate can be found from this procedure.

Based on our database analyses, 80% of the date images have *Year* fields at the right ends; for this reason, right ends of date images are always checked first

in the confirmation stage, and if reasonable *Year* fields can be detected with high confidence values from the right ends, they are returned as the *Year* fields detected.

- *GetYearFromRight* This procedure is used to detect *Year* field or *Year* field candidate from the right end of a date image. The detection method is based on separator detection. Punctuations are first detected, and if a *Year* field or a *Year* field candidate can be found based on the punctuation detected, return the *Year* or the *Year* candidate; otherwise the gap candidates are obtained based on our gap detection method. The gap candidates based on BBD are first checked. If a *Year* field with high confidence value can be found or a *Year* field candidate with high confidence value can be found based on the gap candidates, return the *Year* or the *Year* candidate; otherwise the gap candidates based on NMD are checked using the same method. If no reasonable *Year* candidate can be detected based on all the separator candidates, FALSE is returned by the procedure to indicate no *Year* candidate can be found from the right end.

  1. The first step of this procedure is to detect the separator candidate between *Year* and *Day&Month* fields from the right side of a date image. If a punctuation can be found with high confidence within the five connected components at the right half part of the date image, check the field on the right side of the punctuation, if it is of type I, this field is returned as the *Year* detected; otherwise if the right field is of type II or type III, it is returned as the *Year* candidate from the right side. For other cases, go to the next step.
  2. In this step, the separator is detected from the four maximum gap candidates, gap_{BBD_1}, gap_{BBD_2}, gap_{NMD_1} and gap_{NMD_2}, and only the gap candidates on the right half part of the date image are considered.
     (a) gap_{BBD_1} and gap_{BBD_2} are first evaluated when they exist (if only gap_{BBD_1} exists, just evaluate gap_{BBD_1}). Check the fields on the right sides of the gap candidates, and if a *Year* candidate of type I can be

detected based on one of these two gaps, this field is returned as the *Year* detected; otherwise if a *Year* candidate of type II can be detected, this field is returned as the *Year* candidate from the right side; otherwise go to the next step.

(b) gap_{NMD_1} and gap_{NMD_2} are evaluated using the same method as in (a). If no *Year* candidate can be obtained in this step, go to next step.

(c) If a *Year* candidate of type III can be detected based on these four gap candidates (gap candidates based on BBD are checked first), this candidate is returned as the *Year* candidate from the right side; otherwise go to the next step.

3. Return FALSE to indicate no *Year* candidate can be found from the right end.

- *GetYearFromLeft*
  This is analogous to *GetYearFromRight* to return a *Year* candidate from the left end of the date image.

The final step of our *Year* detection module is to compare the two *Year* candidates from the right and left ends if they exist. The comparison uses a method similar to the procedure used in *GetYearFrom2Punc*, which can be expressed as:

1. Step 1: Check the right end. If the field is of type II, it is returned as the *Year*; otherwise go to the next step.

2. Step 2: Check the left end. If the field is of type II, it is returned as the *Year*; otherwise go to the next step.

3. Step 3: Check the right end. If the field is of type III, check the left end. If no *Year* candidate is found from this end, the right field is returned as the *Year*; otherwise go to the next step.

4. Step 4: Check the left end. If the field is of type III, check the right end. If no *Year* candidate is found from this end, the left field is returned as the *Year*; otherwise go to the next step.

5. Step 5: No reasonable *Year* can be found, return FALSE and reject the date image.

To summarize, in our *Year* detection method for the free format, the *Year* candidates are first detected based on the separator candidate detection, and then the confirmation is implemented by using the characteristics of the *Year* field.

5.3 Detection of the Separator between *Day* and *Month*

From this section onwards, the segmentation system of *Day&Month* field will be presented. In this section, the method for detecting the separator between *Day* and *Month* fields is addressed. Based on the separators detected, *Day&Month* segmentation and identification problem will be discussed in the following section. Since ambiguous cases in separator detection, segmentation and identification stages are handled by a multi-hypotheses generation module and a evaluation module, the final section of this chapter presents the multi-hypotheses generation and evaluation modules

If *Day&Month* is separable, two types of separators can serve as the separator between *Day* and *Month*: punctuation and gap. In our segmentation module, we first find the separator by detecting the punctuation. If no punctuation can be found or confirmed, the gap between *Day* and *Month*, denoted by gap_{DM}, is detected.

5.3.1 Detection of Punctuations

The most frequently used punctuations in separating *Day* from *Month* are slash '/', hyphen '-', comma ',' and period '.'. Shape and spatial features, and the knowledge about writing styles are used to detect them by a candidate_then_confirmation strategy.

Candidate Detection

A punctuation detection method has been given in the *Year* detection module. Here a similar method is used to detect punctuation candidates between *Day* and *Month* fields. Since only one separator is expected to separate the *Day* field from the *Month* field, we introduce a new spatial feature, *in_max_gap*, which means a *punctuation gap* (the gap between the right neighbor and the left neighbor of the punctuation) should be gap_{BBD_1}. This feature is used for the detection of hyphen, period and comma. Therefore, the procedure for punctuation candidate detection can be described as follows:

1. *No_innerloop* and *simple_curve* features are first applied to exclude unlikely candidates.

2. Only those components with a height greater than the height of the image middle zone are considered for slashes. Only those having the *low_to_left* and *in_max_gap* (except those located at the end of the images) features are tested for periods and commas. Other candidates having *in_max_gap* feature are checked for hyphens only.

3. Estimate the likelihood of a connected component being a certain punctuation candidate. For slash, period and comma, we use the average confidence value obtained from each feature set. Different weights, 1 for *high_density* and *flat*; 0.5 for *at_middlezone* and *mid_to_neighbor*, are used to obtain the average confidence value for hyphen.

4. A punctuation candidate is detected when its confidence value as a punctuation is higher than a threshold, which is established in the training stage.

Candidate Confirmation

Candidate confirmation is accomplished by an efficient two-level strategy. If the shape and spatial features of a candidate are significant, the candidate is verified as a punctuation. Otherwise, confirmation is accomplished at the second level by considering

knowledge about the writing style. Although different punctuations require different procedures for candidate confirmation, the basic ideas are the same.

1. Slash candidate confirmation

Compared with hyphen, comma and period, slash candidates are more difficult to be confirmed because the numeral '1' or letter '*l*' is easily misinterpreted as slash '/'. Some slash candidates are shown in Figure 22.

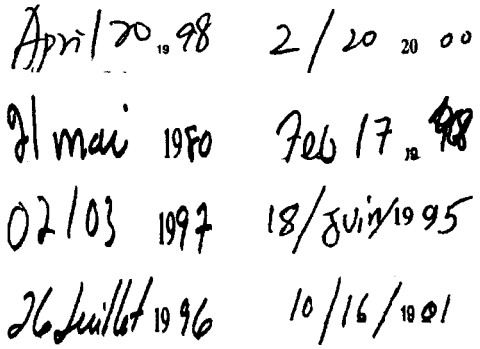

Figure 22: Sample dates including slash candidates

Based on observations, "long" is the critical feature for slash. So we confirm a slash candidate at the first level if the ratio of the height of a candidate to that of the other longest component in the *Day&Month* field exceeds a threshold. If there are two slash candidates in the *Day&Month* field and the second one is located at the end, we ignore the second one when we make the comparison to confirm the first one.

If a slash candidate cannot be confirmed as above, we make a decision at the second confirmation level by (a) analyzing the writing style of *Day&Month* using the A/N differentiation module of combining multiple MLPs (as discussed in Chapter 4) and (b) making use of the knowledge that slashes usually appear when both *Day* and *Month* are written in numerals and at least one side of a slash should be in numerals (from the database analyses in Chapter 3, more than 95% of slashes belong to the first case).

The basic pattern based grammar for the confirmation of a slash candidate in the second level is the following:

Table 30: Basic condition-action rules for the confirmation of a slash candidate

Pattern	Condition	Action
NSN	/	'/' candidate is confirmed if an additional distance feature can be satisfied
NSA	/	New "long" features are checked
ASN	/	New "long" features are checked
ASA	/	Separator candidate is not confirmed

Some explanations about Table 30 are the following:

- If the contextual knowledge shows high likelihood that a *Day&Month* subimage is composed of only numerals (except for slash candidates), the slash candidate should be confirmed as a slash because slashes usually appear when both *Day* and *Month* are written in numerals, so the feature set for slash candidate detection is good enough for confirmation in this case. However, in the case that a gap is used to separate *Day* from *Month*, and *Month* is written in numerals (1.25% of the English date images and 1.75% of the French date images are of this case), a numeral '1' in *Day* or *Month* field may be detected as a slash candidate, and this numeral '1' is easily misinterpreted as slash if no other conditions are checked. Therefore, a distance feature is also used for the confirmation. The *distance_to_left* (*distance_to_right*) is the minimum run-length distance between the candidate and its left (right) nearest neighbor [125]. The difference between these two distances should be less than a threshold for the candidate to be confirmed as a slash.

- If the contextual knowledge shows a high likelihood that both neighbors of the slash candidate are written in alphabetic form, the slash candidate is retained (but not considered as a slash) for the next stage of processing.

- In all other cases, more stringent conditions than in the candidate detection

stage (related to "long" feature) are used according to the contextual knowledge obtained.

Before presenting the detailed steps used in the second confirmation level, outputs from the A/N differentiation module of combining multiple NNs are first given, which will be used in the confirmation stage. Two outputs can be obtained from the A/N differentiation module for a subimage. One is the writing style W, and the other is the rejection label *Rej*. The writing style W can be 'A' or 'N', which represents that the subimage is written in alphabetic format or numeric format, respectively. The rejection label *Rej* represents the confidence value for deciding the writing style of the subimage. The rejection label is equal to '0' or '1' based on the confidence value returned by the NN combination system in the A/N differentiation module. If the confidence value is less than a threshold, the rejection label is set to '1', which means the decision on the corresponding writing style is not reliable. If the rejection label is equal to '0', the corresponding decision on the writing style is reliable.

The detailed steps used in the second confirmation level can be described as:

1. By using the A/N differentiation module of combining multiple NNs, the writing styles W, and rejection labels *Rej* for both left and right neighbors of the candidate are obtained. Let W_{Left} and W_{Right} be the writing styles for the left and right sides of the candidate, respectively, and Rej_{Left} and Rej_{Right} be the corresponding rejection labels.

2. If $W_{Left} = N$ and $W_{Right} = N$ and $Rej_{Left} = 0$ and $Rej_{Right} = 0$, and if the difference of *distance_to_left* and *distance_to_right* is less than a threshold th_d, the candidate is confirmed.

3. If $W_{Left} = A$ and $W_{Right} = A$ and $Rej_{Left} = 0$ and $Rej_{Right} = 0$, the candidate is ignored since at least one side of a slash should be in numerals.

4. If $W_{Left} = A$, more stringent conditions than in the candidate detection stage are used to avoid confusing a letter '*l*' or a numeral '1' with a slash '/' because *A'/'N* is not a common style (based on the database analyses in Chapter 3).

Therefore, in addition to just using the *exceed_neighbor* which only compares the "long" feature of the slash candidate with its left and right nearest neighbors, a new feature *long_to_left* is checked to avoid a letter '*l*' being misinterpreted. This new feature compares the height of the candidate with that of the longest left neighbor (not necessarily the nearest) due to greater differences in heights of different letters within a word compared with numerals (refer to Figure 23).

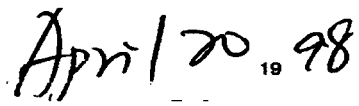

Figure 23: Sample date including a letter '*l*' as a slash candidate

We also check whether the candidate is longer than its right neighbor by a value so as to avoid a numeral '1' being misinterpreted (refer to Figure 24).

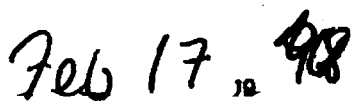

Figure 24: Sample date including a numeral '1' as a slash candidate

If the candidate is not long enough for the comparisons above, it is retained (but not considered as a slash) for the next stage of processing; otherwise it should be confirmed as a slash. However, in order to obtain more reliable results, instead of confirming the candidate under this situation (*A'/'N* is not a common style), the multi-hypotheses generation module is activated to put the candidate into a multi-hypotheses list, which will be evaluated later in the recognition stage. More discussions about multi-hypotheses generation and evaluation will be given in the final section of this chapter.

5. The same method of step 4 is applied to the right side of the candidate.

6. Under other conditions, the candidate is confirmed by *exceed_neighbor* feature with a higher threshold, and the multi-hypotheses generation module is also activated when "common style" cannot be detected.

The above steps are suitable for candidates inside *Day&Month* fields. For candidates at the ends of the subimages, the candidates are confirmed by *exceed_neighbor* feature with a higher threshold because just structural features will be used in this case. After the confirmation, the slash detected is assigned the confidence value obtained in the candidate detection stage.

2. Hyphen candidate confirmation

Since it is difficult to distinguish small punctuation marks (hyphen, period and comma) from noise (such as pieces of broken strokes resulting from improper preprocessing) or some small letters by using structural features, all hyphen candidates are confirmed at the second level. The basic pattern based grammar for the confirmation is as follows:

Table 31: Basic condition-action rules for the confirmation of a hyphen candidate

Pattern	Condition	Action
NSN	-	'-' candidate is confirmed
NSA	-	New *straight_enough* feature is checked
ASN	-	New *straight_enough* feature is checked
ASA	-	Separator candidate is not confirmed

The steps of the confirmation are as follows:

1. Calculate W_{Left}, W_{Right}, Rej_{Left} and Rej_{Right}.

2. If $W_{Left} = A$ and $W_{Right} = A$ and $Rej_{Left} = 0$ and $Rej_{Right} = 0$, the candidate is ignored since at least one side of a hyphen should be in numerals.

3. If $W_{Left} = A$ or $W_{Right} = A$, check whether the candidate is straight enough since a flat "v" or "c" can be confused easily with a hyphen when some letters

appear in the image. The *straight_enough* feature limits the number of occurrences of more than one run in one row or one column of the candidate image. If the candidate is not straight enough, it is retained (but not considered as a hyphen) for the next stage of processing; otherwise it should be confirmed as a hyphen. However, as in the case of slash detection, *A'-'N* or *N'-'A* is not a "common style", and therefore the multi-hypotheses generation module is activated in this situation.

4. If steps 2 and 3 do not apply, the candidate is considered to be a hyphen.

After confirmation, the hyphen is assigned the confidence value obtained in the candidate detection stage.

3. Period or comma candidate confirmation

All period or comma candidates are confirmed at the second level. The basic pattern based grammar for the confirmation is as follows:

Table 32: Basic condition-action rules for the confirmation of a period or a comma candidate

Pattern	Condition	Action
ASN	.(,)	Separator candidate is confirmed
ASA	.(,)	Separator candidate is not confirmed
NSN	.(,)	Multi-hypotheses generation module is activated
NSA	.(,)	Multi-hypotheses generation module is activated

The steps of the confirmation are as follows:

1. Calculate W_{Left}, W_{Right}, Rej_{Left} and Rej_{Right}.

2. If $W_{Left} = A$ and $W_{Right} = A$ and $Rej_{Left} = 0$ and $Rej_{Right} = 0$, the candidate is ignored since at least one side of a period or a comma should be in numerals.

3. If $W_{Left} = A$ and $W_{Right} = N$ and $Rej_{Left} = 0$ and $Rej_{Right} = 0$, and the confidence value of the candidate is larger than a threshold (higher than that

used in the candidate detection stage), the candidate is confirmed because a period or a comma usually appears in this case.

4. If steps 2 and 3 do not apply, the multi-hypotheses generation module is activated.

Due to the high degree of variability among periods and commas, the structural features used in the candidate detection stage are not very effective. Therefore, a low threshold is used in the candidate detection stage, and the period or comma candidates with low confidence values are put into the multi-hypotheses list in step 4.

After confirmation, the period or the comma is assigned the confidence value obtained in the candidate detection stage.

5.3.2 Detection of gap_{DM}

If no punctuation is detected inside the *Day&Month* subimage, or some punctuation candidates have been put into multi-hypotheses list, the gap between *Day* and *Month*, gap_{DM}, is located for the purpose of segmentation. Locating gap_{DM} is a difficult task because users may write the cheques in such free styles that gap_{DM} does not always appear as the observed widest gap in *Day&Month* field. The candidate_then_confirmation strategy is also used to detect gap_{DM}.

Detection of Candidates

The method for gap candidate detection in the *Year* detection module is used here to detect the candidates of gap_{DM}. We also choose four gap candidates, that is, gap_{BBD_1}, gap_{BBD_2}, gap_{NMD_1} and gap_{NMD_2} in *Day&Month* field.

Confirmation of Candidates

Similar to the confirmation of punctuation candidates, a two-level strategy is adopted to select gap_{DM} from its candidates. Before giving the detailed description about

the two-level confirmation strategy, basic ideas used in the confirmation stage are addressed below.

Based on observation and experiments, only a wide gap_{BBD_1} can be reliably confirmed as gap_{DM} at the first confirmation level (see example in Figure 25). Here a wide gap_{BBD_1} means the gap is wider than a threshold (or it is equal to gap_{NMD_1}) and much wider than gap_{BBD_2}.

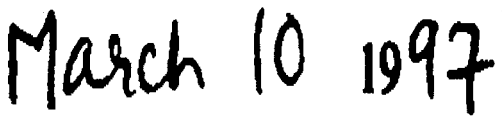

Figure 25: Sample date for gap_{DM} confirmation at the first level

If gap_{DM} cannot be confirmed at the first confirmation stage, the second confirmation level is invoked. In order to detect gap_{DM} from the four gap candidates at the second confirmation level, a step-by-step method is used. We first select one gap candidate gap_{BBD} from gap_{BBD_1} and gap_{BBD_2}, and one gap candidate gap_{NMD} from gap_{NMD_1} and gap_{NMD_2}. Then gap_{DM} is detected from the two gap candidates gap_{BBD} and gap_{NMD}. Therefore, a procedure which selects a gap from two gap candidates is called three times in the above confirmation stage.

For selecting a "good" gap (the gap is likely to be gap_{DM}) from two gap candidates, we assume that there is only one word gap (gap_{DM}) and that it appears at the only transition point from numerals to letters, or from letters to numerals in the *Day&Month* field. This assumption is reasonable because according to the statistical results of the database analyses in Chapter 3, more than 98% of date images have *Month_in_letter* when gaps are used to separate *Day* and *Month* fields. When this assumption is not true, our method will be expected to return FALSE which indicates that the gap detection cannot be done here, and the multi-hypotheses generation module will be invoked. (More explanations can be found in the detailed description of the method given later.) Based on the above assumption, the basic idea used in this method is to try to group the three subimages separated by two gap candidates

into two parts, one consisting of numerals, and the other consisting of letters (possible suffixes in these subimages should be removed first). In Figure 26, the gap candidates are gap_1 and gap_2 which result in three subimages, Im1, Im2 and Im3. After applying our method, Im2 and Im3 are clustered together, and gap_1 is therefore detected as gap_{DM}.

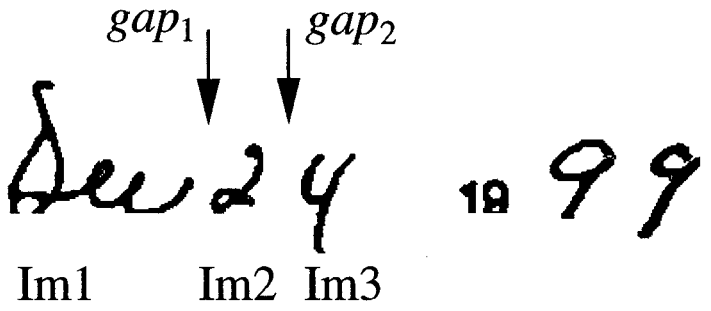

Figure 26: Sample date for the detection of gap_{DM}

In order to cluster Im1, Im2 and Im3 of Figure 26 into two parts, one consisting of numerals and the other consisting of letters, the writing styles of subimage Im1, Im2 and Im3 have to be analyzed. The two methods presented in Chapter 4 can be used to implement this task. Since Im2 may consist of only letter '*l*' of "April", or letter 'O' of "Oct.", etc., which are very similar to numeral '1' and '0', the A/N differentiation module of combining multiple MLPs cannot provide reliable results under this situation. The method based on the *distance_to_numeral* measure can be used here because it provides recognition results from the digit recognizer. Therefore, if Im2 is recognized as numeral '1' or '0' with high confidence, we know that Im2 may consist of letter '*l*', or letter 'O'. Under this situation, it is difficult to determine the writing style of Im2, and so the clustering method is not reliable. The multi-hypotheses generation module would be invoked in this case, as it would be for some other ambiguous cases.

The above method of selecting a "good" gap from two gap candidates has been implemented in a procedure Gap_{select} in our date segmentation system, which can be

described as follows.

The description of Gap_{select}:
The input to Gap_{select} are two gap candidates and the output is TRUE with a gap selected or FALSE otherwise. Here TRUE means one "good" candidate has been found, and FALSE means no candidate can be confirmed by this procedure. Assume that the two gap candidates divide the *Day&Month* image into three subimages, Im1, Im2 and Im3 from left to right (see example in Figure 26). After removing possible suffixes from the three subimages, we try to cluster the three subimages into two parts, one consisting of numerals, and the other consisting of letters, based on: (i) the *distance_to_numeral* measure ($Confid_{\text{numeric}}$) of the three subimages; and (ii) the assumptions that the numerals (*Day*) is located at one end of the *Day&Month* field and *Month* is written in letters. (When *Month* is written in numerals and the separator between *Day* and *Month* is a gap, Gap_{select} should return FALSE, which can be inferred from the description below).

Suppose $Confid_{\text{numeric}}$ values of the three subimages, Im1, Im2 and Im3, are $Confid_1$, $Confid_2$ and $Confid_3$, and the corresponding gaps are gap_1 and gap_2 from left to right. The procedure considers the following two values:

$$Dist_1 = |Confid_1 - (Confid_2 + Confid_3)/2| \text{ and}$$
$$Dist_2 = |(Confid_1 + Confid_2)/2 - Confid_3|$$

Since Im1, Im2 and Im3 are to be clustered into two parts, two possible results can be obtained: Im1 and Im2 are clustered into one part, or Im2 and Im3 are clustered into one part. Therefore these *Dist* values represent a distance (based on $Confid_{\text{numeric}}$ measure) between two parts obtained from the clustering. Since one part should consist of numerals, while the other should consist of letters, a larger value of $Dist_1$ ($Dist_2$) would imply that Im2 and Im3 (Im1 and Im2) should be clustered together. In addition, the part with high *Confid* value is likely to be numeric. The procedure works as follows:

1. If $Dist_1 > Dist_2 + \alpha$ and $Confid_1 > (Confid_2 + Confid_3)/2$, check the subimage Im1. If it is a reasonable *Day*, that is, its $String_{length} < 3$ and not wider than

the components on its right (combination of Im2 and Im3), then gap_1 is selected as the output and return is TRUE. Otherwise return FALSE. Here α is a small value, which means $Dist_1$ needs to be larger than $Dist_2$ by only a small value.

If $Dist_1 > Dist_2+\alpha$ and $Confid_1 < (Confid_2+Confid_3)/2$, check the combination of subimages Im2 and Im3. If it is a reasonable *Day* (its $String_{length} < 3$, $String_{value} \leq 31$ and not wider than Im1), check subimage Im2. If $String_{value} \neq 1$ for Im2, gap_1 is chosen as the output and return is TRUE. Otherwise return is FALSE. Here the condition of $String_{value} \neq 1$ for Im2 is used to avoid mistakes caused by the last letter '*l*' of "April", etc.

2. If $Dist_2 > Dist_1 + \alpha$, use an analogous method as in step 1 to check the combination of subimages Im1 and Im2, or to check subimage Im3, and make a decision. The difference is that the condition of $String_{value} \neq 1$ used in step 2 is replaced by the condition of $String_{value} \neq 0$. Here the condition of $String_{value} \neq 0$ is checked for Im2 when the combination of subimages Im1 and Im2 are considered to be *Day*. In this case, it is possible that Im2 consists of the first letter 'O' of "Oct" etc., which is easily misinterpreted as numeral '0'. Therefore this condition is checked, and if it does not apply, return is FALSE as in step 2.

3. If both conditions 1 and 2 do not hold, that is, $Dist_2$ and $Dist_1$ have similar values, return FALSE.

Based on the procedure Gap_{select} given above, the detailed description about the two-level confirmation strategy for gap_{DM} detection is as follows:

1. The first confirmation level.
   If the following are true, the confirmation is accomplished.

   (a) If $BBD_1 > th_g$ and $\frac{BBD_2}{BBD_1} < th_{d_1}$,
   gap_{BBD_1} is confirmed as gap_{DM}, where th_g is a gap distance threshold, while

th_{d_1} is a threshold for the ratio of the two distances. These thresholds are obtained from the training set.

(b) If $gap_{BBD_1} = gap_{NMD_1}$ and $\frac{BBD_2}{BBD_1} < th_{d_1}$ and $\frac{NMD_2}{NMD_1} < th_{d_2}$, $gap_{BBD_1}(gap_{NMD_1})$ is confirmed as gap_{DM}.

If gap_{DM} cannot be confirmed above, the second confirmation level is invoked.

2. The second confirmation level

(a) Select one candidate gap_{BBD} from gap_{BBD_1} and gap_{BBD_2} if gap_{BBD_1} exists. If gap_{BBD_2} exists and $\frac{BBD_2}{BBD_1} \geq th_{d_1}$, call Gap_{select} procedure to detect the candidate gap_{BBD}. Otherwise gap_{BBD_1} is considered as gap_{BBD}. Here Gap_{select} may return FALSE, which means the selection cannot be implemeted in this procedure.

(b) Select one candidate gap_{NMD} from gap_{NMD_1} and gap_{NMD_2}. If gap_{NMD_2} exists and $\frac{NMD_2}{NMD_1} \geq th_{d_2}$, call Gap_{select} to detect gap_{NMD}. Otherwise gap_{NMD_1} is selected as gap_{NMD}.

(c) If Gap_{select} returns FALSE in (a) or (b), the corresponding gap selection cannot be determined, and therefore gap_{DM} cannot be confirmed at the confirmation stage. In this case, the multi-hypotheses generation module is invoked. The candidates with their types and confidence values are recorded in the multi-hypotheses list (e.g. if Gap_{select} returns FALSE in (a) and Gap_{select} returns TRUE in (b), gap_{BBD_1}, gap_{BBD_2} and gap_{NMD} are put into the list). Here the types input to the list is "gap", and the confidence values are calculated as: (i) For gap_{BBD_1} and gap_{NMD_1}, the value is equal to 1.0; and (ii) For gap_{BBD_2} and gap_{NMD_2}, the values are $\frac{BBD_2}{BBD_1}$ and $\frac{NMD_2}{NMD_1}$, respectively. If Gap_{select} returns TRUE in both (a) and (b), goto the next step.

(d) Detect gap_{DM} from gap_{BBD} and gap_{NMD}. If gap_{BBD} does not exist (due to no BBD existing) gap_{DM} is located at gap_{NMD}. Otherwise call Gap_{select} procedure to choose gap_{DM} from gap_{BBD}

and gap_{NMD}. Similar to step (c), the multi-hypotheses generation module is invoked when Gap_{select} returns FALSE. If Gap_{select} returns TRUE, gap_{DM} is found.

In summary, for the detection of gap_{DM}, we first choose four gap candidates based on two gap distance measures and then try to confirm one candidate by the spatial features or in the contextual analysis. If the confirmation fails, the multi-hypotheses generation module is invoked.

5.4 *Day&Month* Segmentation and Identification

After the separator detection step, *Day&Month* segmentation and identification can be conducted. However, in the separator detection stage, some separator candidates have been put into the mult-hypotheses list due to uncertainty. In this case, *Day&Month* segmentation and identification should be implemented in the multi-hypotheses evaluation module. Therefore, if the multi-hypotheses list is not empty, the segmentation is not conducted here; instead, the separators detected with their types and confidence values are put into the multi-hypotheses list. For other cases (the multi-hypotheses list is empty), according to the number of punctuations detected within the *Day&Month* image, their types and their positions, a set of rules have been developed to segment the *Day&Month* field into *Day* and *Month* fields and to identify whether the *Month* field is written in letters or numerals.

1. If no punctuation is detected, *Day&Month* field is segmented according to gap_{DM} (In the case that no gap_{DM} can be found in a *Day&Month* field, the corresponding date image is rejected by our date processing system).

   (a) By using the A/N differentiation module of combining multiple MLPs, we can obtain the writing styles (A or N), together with the rejection labels for both sides of gap_{DM}

   (b) If $W_{Left} = A$ and $W_{Right} = N$ and $Rej_{Left} = 0$ and $Rej_{Right} = 0$, the left segment is assigned to the *Month* field, which is supposed to be written

in letters, while the right segment is assigned to the *Day* field written in numerals. Similarly, if $W_{Left} = N$ and $W_{Right} = A$ and $Rej_{Left} = 0$ and $Rej_{Right} = 0$, the right segment is assigned to the *Month* field written in letters, while the left segment is assigned to the *Day* field. For all the other cases, the A/N differentiation module cannot provide reliable information to indicate a "common style" is adopted in the *Day&Month* field (when gap_{DM} is used as the separator). Here the "common style" should be ASN or NSA based on our database analyses in Chapter 3. Therefore the multi-hypotheses generation module is activated for these cases, and the gap candidate with its type and confidence value are put into the multi-hypotheses list.

2. Suppose one punctuation is detected.

   (a) When it is located at the right end of *Day&Month* field, remove it if it is either a period or a comma and retain it if it is either a hyphen or a slash. The reason is that people may write a period or comma after *Day* and *Month*, but they would not put a single slash or hyphen at the end of *Day&Month* field. Therefore a slash or hyphen here would not be a true punctuation but should be processed as part of the image. In this case, since no punctuation is detected inside *Day&Month* field, *Day&Month* field is segmented according to gap_{DM}.

   (b) When a slash or a hyphen is detected inside *Day&Month* field, the image is segmented at the position of that punctuation. Since a slash or a hyphen can only be confirmed in NSN pattern in the separator confirmation stage (see Section 5.3), both the left and right segments are supposed to be written in numerals. (An exception is that the presence of very "long" slashes, which can be confirmed at the first confirmation level based on structural features, may indicate any image format such as ASN, NSN, etc.. In this case, A/N differentiation module is used to determine the writing style of each field). The assignment of the two segments to the

Day and *Month* fields will be implemented by a parser in the recognition stage.

(c) When a period or a comma is detected inside *Day&Month* field, segmentation occurs at the position of punctuation. Since a period or a comma can only be confirmed in ASN pattern in the separator confirmation stage (see Section 5.3), the left segment is assigned to the *Month* field written in letters, while the right segment is assigned to the *Day* field written in numerals.

3. Suppose two punctuations are detected. If they are of the same type and one of them is located at the end of the subimage, the one located at the end is usually removed. Otherwise the one with a higher confidence value is used, and the other is retained. If two punctuations are of different types, we usually select the more likely candidate between the two, using their confidence values and corresponding weights [32]. However, some special cases should be taken into consideration:

   (a) One punctuation is a period, and the other is a hyphen. If they have similar sizes and similar $below\_lowerhalf$ feature values, and one of them occurs at the end of the subimage (as in Figure 27), they probably come from the same class. Therefore, we remove the one at the end of the subimage, and use the other as a separator and go to step 2.

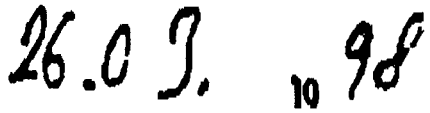

Figure 27: Sample date with two punctuations ('-' and ',') detected

   (b) If a slash is detected after or before a hyphen which has a high confidence value, this slash is usually confused with numeral "1"(as shown in Figure 28). We ignore the slash and select the hyphen as a separator and go

to step 2.

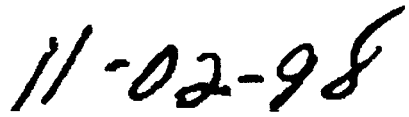

Figure 28: Sample date with two punctuations ('/' and '-') detected

(c) If a slash is detected after a period or comma which has a high enough confidence value, this slash is usually confused with the numeral "1". Therefore, we ignore the slash and proceed with segmentation and hypotheses as in step 2.

When more than two punctuations are detected in *Day&Month*, two of these punctuations are selected by comparing weighted confidence values of all punctuations [32] and step 3 is carried out.

5.5 Multi-Hypotheses Generation and Evaluation

5.5.1 Multi-hypotheses Generation

As discussed above, in the module of *Day&Month* segmentation, only the separators with high confidence values and the writing styles with high confidence values to indicate "common styles" would be confirmed. Here "common styles" are determined based on database analyses, e.g. the gap separator usually occurs at the transition between numeric and alphabetic fields and the subimages on both sides of slash or hyphen are often numeric, etc. Otherwise the multi-hypotheses generation module is activated. This module puts multiple hypotheses into a list, where each hypothesis consists of a possible segmentation of *Day&Month* field. These hypotheses should be interpreted by higher-level modules that employ additional knowledge. In our current system this module passes the unconfirmed separator candidates with their

types and confidence values to the recognition stage to make the final decision based on the recognition results and semantic and syntactic constraints.

In summary, multi-hypotheses are currently generated for the following cases:

(a) One side of a slash or a hyphen indicates a high likelihood of being an alphabetic field.

(b) The confidence value of a period or a comma is less than a threshold, or the corresponding writing style is not *ASN* pattern.

(c) gap_{DM} cannot be confirmed with high confidence from the four gap candidates, or the writing style based on gap_{DM} detected is not *ASN* or *NSA* pattern.

(d) In the segmentation and identification stage, if the multi-hypotheses list is not empty, the final segmentation is not conducted; instead, the corresponding separator with its type and confidence value is put into the list.

In addition, for each hypothesis in the list, we try to remove the possible suffixes from the left and right sides of the candidate. The method used to detect the suffixes is based on *Above_Upperhalf* feature and *Not_Cap* feature. For the *Above_Upperhalf* feature, if part of the connected component falls above the UPPERHALF reference line [32], and the component is small enough so that it does not reach the LOWERHALF reference line, TRUE is returned; otherwise FALSE is returned. For the *Not_Cap* feature, if the *simple_curve* feature is not satisfied or the *straight_enough* feature is not satisfied, return is TRUE; otherwise return is FALSE. If a component has both *Above_Upperhalf* and *Not_Cap* features, it should be considered as a suffix. However, the positions of suffixes vary greatly, which make it difficult to select the thresholds used in *Above_Upperhalf* feature. Therefore, two thresholds are adopted here. If a component can satisfy the high threshold of *Above_Upperhalf*, it is considered as a suffix and it is removed. Otherwise, if it satisfies the low threshold of *Above_Upperhalf*, it is considered as a suffix candidate, and multi-hypotheses generation is activated. Both the segment with this suffix candidate and the segment without this candidate (candidate is removed) are put into the multiple-hypotheses list.

5.5.2 Multi-hypotheses Evaluation

If the segmentation has been confirmed in the previous stage, then appropriate recognizers can be used directly in the recognition stage. Otherwise, each separator and the corresponding writing style are selected from the multi-hypotheses list (obtained in the multi-hypotheses generation module) and considered with results from the digit and month word recognizers.

Together with a separator candidate and a writing style (NSN, NSA or ASN), the segments on both sides of the separator candidate are considered and the confidence value for the three styles (types) are calculated as the weighted sums:

$$\begin{aligned}
Confidence_{Type1} &= w_1 * DigitConfidence_{Left} + \\
&\quad w_1 * DigitConfidence_{Right} + \\
&\quad w_3 * SeparatorConfidence \\
Confidence_{Type2} &= w_1 * DigitConfidence_{Left} + \\
&\quad w_2 * WordConfidence_{Right} + \\
&\quad w_3 * SeparatorConfidence \\
Confidence_{Type3} &= w_2 * WordConfidence_{Left} + \\
&\quad w_1 * DigitConfidence_{Right} + \\
&\quad w_3 * SeparatorConfidence
\end{aligned}$$

$DigitConfidence_{Left}$ and $DigitConfidence_{Right}$ are modified confidence values from the digit recognizer, i.e. $Confid_{\text{numeric}}$ values (as described in Section 4.2) for the left and right sides of the separator candidate respectively. $WordConfidence_{Left}$ and $WordConfidence_{Right}$ are confidence values from the month word recognizer (This recognizer will be discussed in the next chapter). $SeparatorConfidence$ is the confidence value of the separator candidate, which has been derived from the segmentation stage.

The weights w_1, w_2 and w_3, in the equations are determined in the training stage so that $w_1 = 0.8$, and $w_2 = 1$ because the distribution of confidence values from the digit recognizer is different from that of the word recognizer, and these weights are

set to make the confidence values of the digit and word recognizers comparable. In general cases, w_3 is set to 1. However, since different calculations are made to obtain the confidence values for punctuation and gap, the confidence value of the maximum gaps based on the two distance measures (BBD and NMD) are always equal to 1.0, while the confidence values of punctuations are usually less than 1.0. Therefore, if a punctuation candidate adjacent to a gap candidate in the list, $w_3 = 1.2$ for the punctuation candidate because the punctuation candidate is likely to be the true separator in this case. (The value of 1.2 is determined in the training stage.)

For each hypothesis in the list, usually the Type with the maximum $Confidence_{Type}$ value is recorded to be compared with the corresponding information of other hypotheses in the list. More explanations about this basic multi-hypotheses evaluation procedure are given by a simple example described in Figure 29 and Table 33.

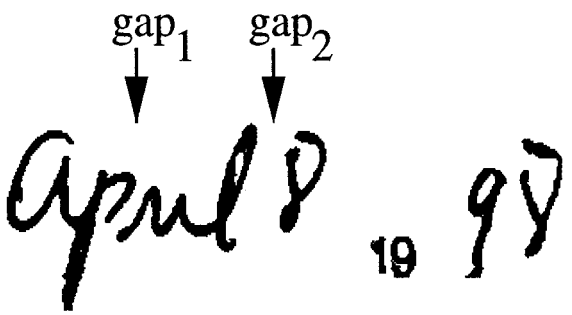

Figure 29: Sample date with two separator candidates in the multi-hypotheses list

Table 33: Example of the multi-hypotheses evaluation

	$DConf_L$	$DConf_R$	$WConf_L$	$WConf_R$	$SConf$	$Conf_{T1}$	$Conf_{T2}$	$Conf_{T3}$
gap_1	0.47	0.85	0.80	0.34	0.79	1.84	1.51	**2.27**
gap_2	0.00	0.96	0.95	0.44	1.00	1.77	1.44	**2.72**

In this example, two gap_{DM} candidates gap_1, gap_2 with their confidence values have been put into the multi-hypotheses list in the multi-hypotheses generation module.

For these two candidates, $Confidence_{Type}$ values are calculated and given in Table 33 (In this table, $DigitConfidence_{Left}$, $DigitConfidence_{Right}$, $WordConfidence_{Left}$, $WordConfidence_{Right}$, $SeparatorConfidence$, and $Confidence_{Type}$ are represented by $DConf_L$, $DConf_R$, $WConf_L$, $WConf_R$, $SConf$, and $Conf_T$, respectively).

For each separator candidate in Figure 29, three corresponding $Confidence_{Type}$ values are compared, and Type3 is found with the maximum $Confidence_{Type}$ value for each of the two candidates. By comparing these two maximum $Confidence_{Type}$ values for the two separator candidates, we can finally conclude that gap2 with the writing style *ASN* (corresponding to Type3) is the segmentation result for the *Day&Month* field of this sample. At the same time, the corresponding recognition results for the *Day* and *Month* fields can be obtained when $DigitConfidence_{Left}$, $DigitConfidence_{Right}$, $WordConfidence_{Left}$, and $WordConfidence_{Right}$ are returned from the recognizers. (Both segmentation and recognition results are correct for this sample.)

From the above example, we can see that the multi-hypotheses evaluation method is effective. In application, some semantic and syntactic constraints can be used to improve the performance of this multi-hypotheses evaluation module.

First, based on semantic constraints, if the recognition result from a Type is not a valid date, the corresponding $Confidence_{Type}$ would be reduced by a small value (α) before $Confidence_{Type}$ values are compared in Type selection. (For the above example, $Conf_{T1}$, $Conf_{T2}$, and $Conf_{T3}$ values are modified to 1.34, 1.51, and 1.77 for gap_1, and 1.27, 0.94, and 2.72 for gap_2 by using this rule and setting α to 0.5. Therefore, same correct results can be obtained for this sample with higher confidence.)

In addition, as we discussed above, "common styles" have been determined based on database analyses. These "common styles" can be used as syntactic constraints to modify the Type selection procedure, that is, if the interpretation of the first choice is not a "common style" and the difference between the confidence values of the top two choices is very small, the second choice which has the second largest $Confidence_{Type}$ value should be the final selection if the interpretation of this choice is a "common style" and is a valid date. These cases are the following:

(a) Suppose a slash or a hyphen is the separator candidate. If $Confidence_{Type1}$ does not have the largest value, but the difference between $Confidence_{Type1}$ and the largest value is less than a (small) threshold, and a valid date can be obtained from Type1, then Type1 with its confidence value $Confidence_{Type1}$ is selected for this hypothesis. This is because slashes and hyphens are most commonly used to separate the *Day* and *Month* fields when they are both numeric.

(b) When a period or a comma is the separator candidate, the same method as in (a) is applied except that Type3 is the common style now.

(c) Suppose a gap is the separator candidate. If $Confidence_{Type1}$ has the largest value, but the difference between $Confidence_{Type1}$ and the second largest $Confidence_{Type}$ value is less than a (small) threshold, the Type with the second largest $Confidence_{Type}$ is selected for this hypothesis because a gap is most commonly used to separate a numeric field from an alphabetic field.

Now the date images have been segmented into three subimages corresponding to *Day*, *Month* and *Year* fields and writing styles of *Month* fields have been determined. The output of the segmentation stage can be one segmentation hypothesis or multiple segmentation hypotheses, which will be processed in the recognition stage. If only one segmentation hypothesis is produced in the segmentation stage, an appropriate recognizer will be applied to each of *Day*, *Month* and *Year* fields. If multiple segmentation hypotheses are generated, the multi-hypotheses evaluation module will be invoked in the recognition stage. The next chapter will discuss recognizers used in this stage focusing on cursive month word recognizers.

Chapter 6

Date Image Recognition

6.1 Outline of Date Recognition

As discussed in Chapter 3, segmentation and recognition are two main stages of our date processing system. If one segmentation hypothesis can be confirmed in the segmentation stage, an appropriate recognizer (digit recognizer or cursive month word recognizer) is invoked for each of *Day*, *Month* and *Year* fields in the recognition stage. Otherwise, if multiple segmentation hypotheses are generated in the segmentation stage, the final recognition result is obtained based on the multi-hypotheses evaluation module (presented in Chapter 5) which makes use of the results from the digit and word recognizers.

This chapter focuses on the recognition of cursive month words. A combination method with an effective conditional topology has been implemented to recognize month words, and the Vote, Sum and Product combination rules are used. The modified Product rule proposed in Chapter 4 is also used here with some modifications for month word recognition.

6.2 Cursive Month Word Recognition

The recognition of month words on bank cheques poses some new problem in addition to the challenges caused by the high degree of variability in unconstrained handwriting. First, improper binarization and preprocessing can have a great impact on the quality of month word images. Secondly, although a limited lexicon is involved, many similarities exist among the month word classes, and this can give rise to problems in feature extraction and classification. Furthermore, month words on bank cheques can be written in French or in English in Canada, which increases the complexity of the problem. In addition, most month words have very short abbreviated forms, which makes it more difficult to differentiate between them.

According to our literature review, only two systems have been reported for month word recognition. In [105], a system is being developed for handwritten month word recognition on Brazilian bank cheques by using an explicit segmentation-based HMM classifier. Another system mentioned in [74] is the previous work already reported in this thesis, which deals with English month word recognition by combining an MLP classifier and an HMM classifier. In our new recognition system some improvements and modifications based on the previous work have been made to recognize both French and English month words. An effective conditional combination topology is presented to combine two MLP and one HMM classifiers, and the proposed modified Product fusion rule gives the best recognition rate so far.

6.2.1 Writing Style Analyses for Cursive Month Word Recognition

In this section, the writing styles of month words are analyzed based on Training Set 1 (see Chapter 3). Some samples extracted from the set are shown in Figure 30, including both English and French month words.

Altogether 33 English/French month word classes have been observed in the databases including full and abbreviated forms. Based on analyses of the database, many similarities have been found among the word classes:

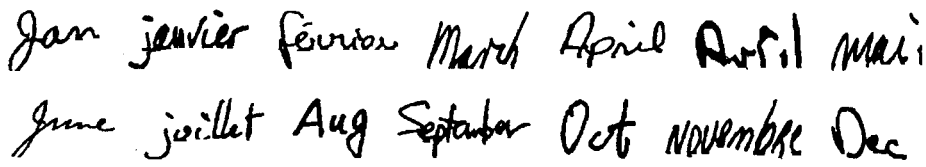

Figure 30: Month word samples from CENPARMI_IRIS bank cheque database

- The most similar classes are: "September" and "Septembre", "October" and "Octobre", "November" and "Novembre", and "December" and "Décember". Since each pair of classes are very similar in shape and they represent the same month, they are assigned to the same class. So only 29 classes will be considered.

- Some other similarities among word are due to shapes, e.g. "Jan", "June" and "Juin", and "Mar", "Mai" and "Mars". They can also contain similar subimages, e.g. "September", "November" and "December". These similarities can affect the performance of the recognition systems, and they have been considered in the feature extraction and individual classifier design.

From analyses of the database, we also find that about one-third of the month word classes consist of only 3 letters, mostly from abbreviations. In addition, French month words have been found to have freer writing styles, e.g. both capital and small letters can be used at the beginning of words, and mixtures of capital and lower-case letters appear more frequently.

These writing style analyses show some of the challenges for month word recognition. Some solutions to this problem will be given in the following sections using three individual classifiers and a combination system.

6.2.2 Individual Classifiers

MLP Classifiers

Two MLP classifiers have been developed in CENPARMI for month word recognition. MLPA and MLPB were originally designed for the recognition of the legal amount [75] and the courtesy amount [133, 157] respectively, and both of them have been modified for month word recognition. The two MLPs use a holistic approach. Their architectures and features can be summarized below. More details can be found from the references cited.

- MLPA is a combination of two networks using different features. The combination adopts the "fusion" scheme as shown in (b) of Figure 31, that is, MLPA is implemented by combining the two networks at an architectural level, and uses the outputs of the neurons in two hidden layers as new input features [75]. As in (b) of Figure 31, the input feature set 1 consists of mesh features, direction features, projection features and distance features, while the input feature set 2 consists of gradient features [75].

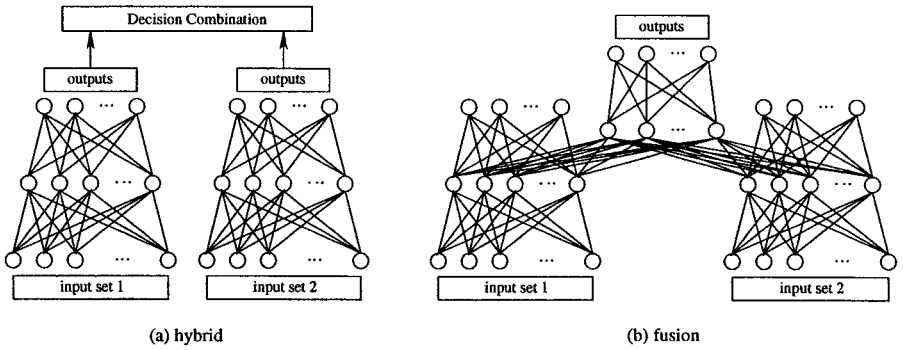

Figure 31: Combination schemes of two MLPs for two input feature sets

- MLPB is a combination of three MLPs. The combination scheme is a "hybrid" strategy which combines the output values of the MLPs (a similar architecture is

shown in (a) of Figure 31). The MLPs are implemented by using three different sets of input features. Feature set 1 consists of pixel distance features (PDF), while feature sets 2 and 3 consist of size-normalized image pixels from different preprocessed images [133, 157].

HMM Classifier

A segmentation based grapheme level HMM has been developed using an analytical feature extraction scheme [74] as shown in Figure 32 for month word recognition. This model has the potential to solve the over- or under-segmentation problem in cursive script recognition. The features used to obtain pseudo temporal sequences for the HMM come from shape feature, direction code distribution feature, curvature feature and moment feature.

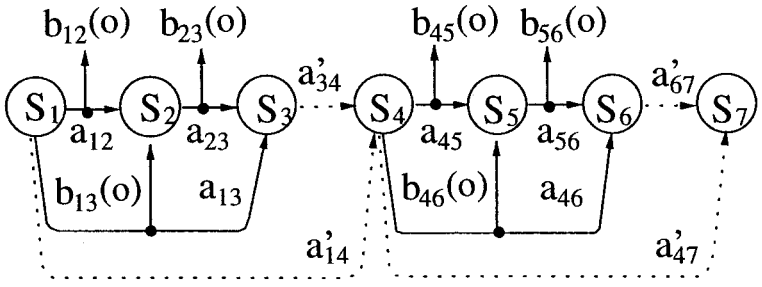

Figure 32: Segmentation based grapheme level hidden Markov model

Performances

The training set for implementing the individual month word classifiers consists of 6201 month word samples extracted from Training Set 1, and the test set consists of 2063 month word samples extracted from Test Set (see Chapter 3). The performances of the two MLP classifiers and the HMM classifier on the test set are shown in

Table 34. In the table, the 12 outputs correspond to the 12 months. These results indicate that the two MLP classifiers produce comparative recognition rates, while the HMM classifier produces the worst results. In order to enhance the recognition performance, combinations of these classifiers are considered in the following section.

Table 34: Performances of individual classifiers

Classifier	Recognition rate(%)	
	29 outputs	12 outputs
MLPA	76.44	78.87
MLPB	75.42	77.27
HMM	66.89	69.70

6.2.3 Combinations

Combination systems are expected to produce better recognition results. Before discussing the combination topology and combination rule for month word classification, further analyses of the performances of the three individual classifiers are presented while focusing on the correlation of the recognition results among these classifiers. Some useful information has been obtained from these analyses.

Correlation of Recognition Results among Individual Classifiers

The correlation of recognition results among the three individual classifiers is shown in Table 35. It was obtained from an additional training set (2063 samples extracted from Training Set 1) separate from the training set for training the individual classifiers. This additional training set is also used in the experiments for developing the combination topology and combination rules to be discussed later.

In Table 35, "Correct" means the classifiers are correct on the sample, "Error" means all the classifiers are incorrect, and C/E means both correct and wrong results are produced by the classifiers. The errors can be further grouped into two classes ErrorI and ErrorII, which mean the samples are assigned different and the same

Table 35: Correlation of recognition results among classifiers

Classifiers	Correct	Error	C/E	ErrorI	ErrorII
MLPA and MLPB	1297	326	440	213	113
MLPA and HMM	1162	282	619	217	65
MLPB and HMM	1167	295	601	218	77
MLPA, MLPB and HMM	1032	199	832	173	26

incorrect classes by the classifiers, respectively. From these results, we can infer the following:

- Even though combining these three classifiers may improve the recognition rate considerably due to the independence in errors, a combination would have a lower bound of 1.26% (26/2063) for the error rate without rejections, because all the three classifiers produce the same 26 errors.
- If two classifiers are to be combined, the combination of MLPA and MLPB make same errors more frequently than the other combinations, which can be observed from the ErrorIIs produced. Although the performances of both MLPA and MLPB are much better than that of the HMM classifier, they are homogeneous classifiers and tend to produce the same errors, which is not useful in combination. The idea that more distinct classifiers can better complement each other has been shown here, and it is very important in designing combination systems. In addition, three classifiers would be less likely to make the same errors than two of them, as observed from the ErrorIIs.

Combination Topology

Combination topologies can be broadly classified as parallel, serial, hybrid and conditional. Based on considerations of both speed and accuracy, as well as experimental findings, a conditional combination topology is proposed here to combine the three classifiers to improve the performance of month word recognition as shown in Figure 33.

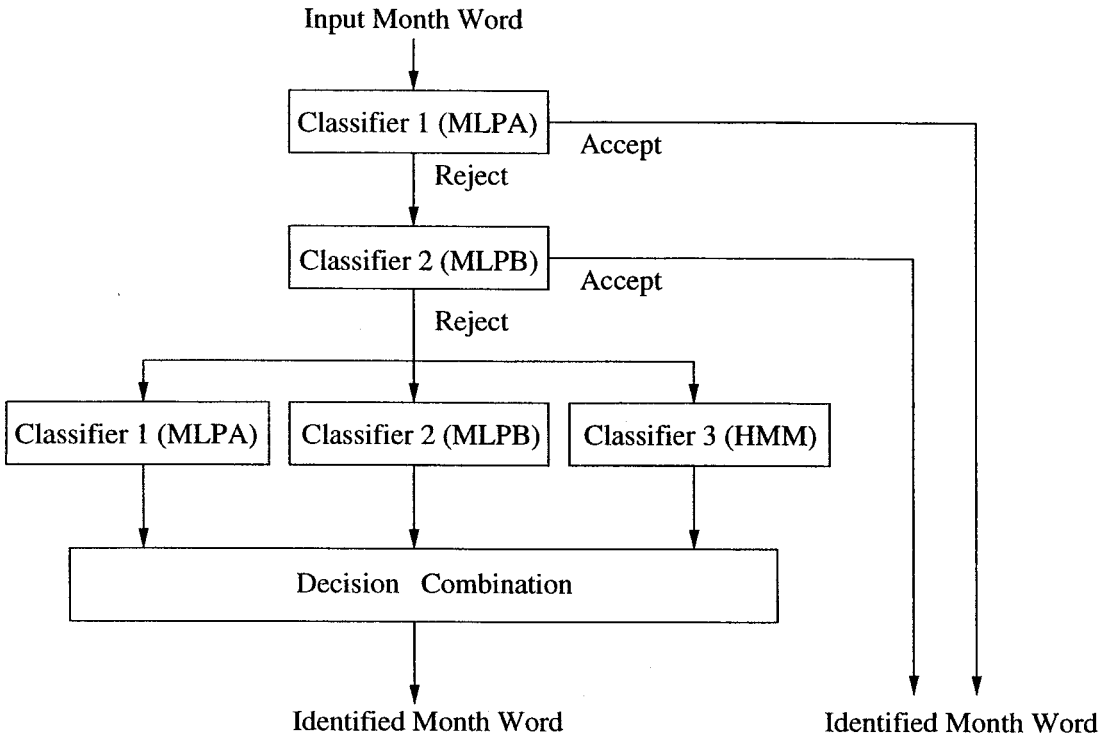

Figure 33: Conditional combination topology for month word recognition

In this architecture, MLPA and MLPB are first applied using serial strategy in the first stage; for samples rejected by both MLPA and MLPB, the decisions of all three classifiers are combined in the second stage. The rejection conditions of both MLPA and MLPB have been set very strictly (total error rate introduced in the first stage is 0.6% on the test set). A rejection is made in MLPA when the top confidence value is less than a threshold (0.99), and a rejection is made in MLPB when the difference between the top two confidence values is less than a threshold (0.4). This stage proved to be very efficient and effective based on the results of our experiments. About 33% of the samples can be recognized in the first stage. More difficult samples are rejected and sent to the three classifiers in parallel, and decisions from these classifiers are combined to get better results. Since the samples have already been

processed by MLPA and MLPB by this stage, additional processing is required only from the HMM classifier. In the following subsections, combination rules used in the second stage will be discussed.

Combination Rule

As discussed in Chapter 4, the Sum (Ave), Product, Maximum, Minimum, Median and Voting rules have been studied extensively [77] to combine parallel classifiers with measurement outputs. In our month word recognition module, the Majority Vote on decisions, Sum and Product rules have been applied.

Since detailed descriptions of these rules have been discussed in Chapter 4 and can be found in the literature [77], here we describe only the weighted Product rule used in our system. Let x be an input sample, and $D_k(x)$, where $k = 1, 2, 3$, be the decision on x by the three classifiers MLPA, MLPB and HMM. $C_{k,w_j}(x)$, where $k = 1, 2, 3$ and $j = 1, 2, ..., 29$ represent the confidence values assigned to class w_j by classifier k. $D(x)$ and $C_{w_j}(x)$ are the final combined result and confidence respectively. The weighted Product rule can be described as follows:

$$C_{w_j}(x) = \prod_{k=1}^{3}(C_{k,w_j}(x))^{Weight_k},\ j = 1, 2, ..., 29$$
$$D(x) = w_j, \text{ if } C_{w_j}(x) = \max_{i=1}^{29}(C_{w_i}(x))$$

Here the weighting factors $Weight_k$s are determined in the training stage, and they are chosen to make the confidence values of the three classifiers comparable because the distributions of the confidence values for the three classifiers are different.

Experiments have been conducted to compare these combination rules based on the conditional topology implemented. The results are shown in Table 36 without rejections.

In Chapter 4, a problem of the Product rule has been discussed regarding the effect of very low output (confidence) values on the product (veto effect). A modified Product rule was therefore proposed in this thesis to reduce this effect. This modified Product rule is also used in the combination system for month word recognition, and some changes are made here to accommodate the 29-class recognition problem (as opposed to the two-class problem of A/N differentiation).

Table 36: Performance of different combination rules on the test set

Combination	Recognition rate(%)	
	29 outputs	12 outputs
Majority Vote	76.59	79.01
Sum	79.69	81.39
Weighted Product	84.97	86.67

Modified Product Rule

The modified Product rule used in our combination system for the month word recognition can be described as follows:

If two of $C_{k,w_j}(x)$, $k = 1, 2, 3$ on class w_j are larger than a high threshold (thr1)
and the other $C_{m,w_j}(x)$ is less than a low threshold (thr2), $j = 1, 2, ..., 29$
$C_{m,w_j}(x) =$ thr2.
else
$C_{k,w_j}(x)$, where $k = 1, 2, 3$ and $j = 1, 2, ..., 29$ remain unaltered.
After the above recomputation of $C_{k,w_j}(x)$, the same steps as shown in the previous weighted Product rule are adopted.

In the A/N differentiation of Chapter 4, two different thresholds are also used. However, a different method of choosing these two thresholds is used here, and it is described as follows:

- thr1 is larger than thr2, but it is not a real "high" threshold. We set thr1 to a value such that when a classifier measurement output is larger than this value, this output is very often among the top three choices of all the classifiers in the system.

- thr2 is not a fixed value. The measurement output value of the fifth choice of the classifier considered is used as the threshold in our system. (If the value is zero, then thr2 is set to the small value of 0.05 instead).

In this modified Product rule, the thresholds are related to the measurement outputs in the system. It is a reasonable process that provides an automatic setting of the thresholds. Based on this new modified Product rule, better results have been obtained for our month word recognition as shown in Table 37.

Table 37: Combination results of modified Product rule on the test set

Combination	Recognition rate(%)	
	29 outputs	12 outputs
Modified Product	85.36	87.06

This is the best result produced by our month word recognition system. It is difficult to compare this result with the other systems [105, 74] due to the use of different databases. In [105], a 91% recognition rate for 12 Brazilian month word classes has been reported on a small test set (402 samples). In another system [74], 21 English classes are recognized, and an 87.3% recognition rate was obtained on a test set of 2152 samples. Since both French and English month words are processed in our system, a total of 29 classes are involved, giving rise to greater complexities. The recognition rate of 85.36% for 29 classes in our system is promising.

Comparing the combination results obtained here with those of the individual classifiers, significant performance improvements have been observed. In addition, some experiments have been conducted to compare the effect of combining three or two classifiers. Combining the MLPA and HMM classifiers using the weighted Product rule produces the best results among the combinations of two classifiers, with recognition rates of 83.18% and 85.26% for 29 and 12 outputs, respectively, which is not as good as those of combining the three classifiers.

In Table 38, the performance of our word recognizer for each word class is given when the three classifiers are combined using the modified Product rule. From these results, we can observe the following:

- The recognition rates of different month word classes vary in a wide range from 60.00% to 97.03% (29 outputs).

- The overall recognition rates of short words are worse than those of long words (29 outputs). Month words can be devided into three groups according to their lengths. Words in group 1 contain three letters ("Apr", "Aug", "Dec", "Feb", "Jan", "Mai", "Mar", "May", "Nov" and "Oct"), words in group 2 contain more than six letters ("December", "February", "Février", "January", "Janvier", "Juillet", "November", "October" and "September"), and words in group 3 contain four to six letters. We can see that 50% of the short word classes in group 1 have recognition rates below 80%, while only about 22% of the long word classes in group 2 have such low recognition rates.

- The overall recognition rate of French words is worse than that of English words (29 outputs). After removing some abbreviated forms of month words which are the same for both French and English words, the French word group consists of "Août", "Avril", "Février", "Janvier", "Juillet", "Juin", "Mai" and "Mars", while the English word group consists of "April", "August", "February", "January", "July", "June", "March" and "May". We notice that about 62% of the word classes in the French word group have recognition rates below 80%, while none of the word classes in the English word group have such low recognition rates.

- The recognition rates of "Mar" and "Mai" word classes are the worst among those of all the classes (29 outputs). Based on the error analyses, we find that most errors here are caused by the similar shapes of several word classes. For "Mar" word class, 75% of the errors are produced by misrecognizing "Mar" samples as "Mars", "Mai" or "March" class, and for "Mai" class, 66.67% of the errors are produced by misrecognizing "Mai" samples as "Mars", "Mar" or "March" class.

- Compared with other classes, "Juin" and "Mar" classes show more performance improvements when the number of output classes are changed from 29 to 12. This means that "Juin" tends to be recognized as "June", and "Mar" tends to be recognized as "Mars" or "March" when 29 classes are considered.

The above observations agree with the analyses presented in Section 6.2.1. These observations also indicate the areas where further research would be needed to improve the performance of our month word recognizer in the future.

In Table 38, we can also see that the numbers of samples for the classes are quite different. For example, only 18 samples in the test set belong to "Apr", while 106 samples belong to "Nov". This is because "Apr" is the abbreviation for "April", which is not a long word, and people tend to use the full word of "April". However, for "November" (a long word), its abbreviation "Nov" is used more frequently.

To summarize, the new month word recognizer presented here is effective for the recognition of both French and English month words on Canadian bank cheques. The conditional combination topology with the new modified Product rule has given very promising recognition performance.

Table 38: Combination results of modified Product rule on the test set

Class	Samples	Recognition rate(%)	
		29 outputs	12 outputs
Août	88	82.95	84.09
Apr	18	72.22	77.78
April	101	97.03	98.02
Aug	55	81.82	81.82
August	33	81.82	87.88
Avril	103	85.44	86.41
Dec	97	96.91	96.91
December	64	84.38	85.94
Feb	79	96.20	96.20
February	20	95.00	95.00
Février	58	75.86	75.86
Jan	87	78.16	79.31
January	25	92.00	96.00
Janvier	57	75.44	77.19
Juillet	67	94.03	94.03
Juin	78	78.21	89.74
July	87	86.21	86.21
June	89	89.89	93.26
Mai	91	67.03	69.23
Mar	20	60.00	85.00
March	75	86.67	88.00
Mars	82	79.27	84.15
May	99	83.84	83.84
Nov	106	78.30	78.30
November	55	90.91	90.91
Oct	85	84.71	85.88
October	85	95.29	95.29
Sept	101	91.09	91.09
September	58	91.38	91.38

Chapter 7

Date Image Processing System

An outline of the entire date processing system has already been given in Chapter 3, and the two main modules of segmentation and recognition have been presented in previous chapters. In this chapter, the preprocessing module and the verification module in the postprocessing stage will be addressed. The performance of the complete system will also be given with emphasis on the performance of the segmentation module.

7.1 Preprocessing

The preprocessing module has two main tasks: (i) detection and removal of noise (or rejection of very noisy images); and (ii) detection and removal of "Le".

Since binarized date images provide the input for our date processing system, it is difficult to remove the noise introduced by improper binarization because limited information is available. For this reason, only simple procedures are used to detect noise in our current system, and to remove small noise from the binarized images and to reject images containing strong noise such as a big blob of ink or too many broken pieces. The detection and removal of "Le" will be discussed in detail below.

7.1.1 Detection and Removal of "Le"

Strategy

According to our database analyses, 4.12% of French date images have "Le" at the beginning of the date zones. Since Canadian cheques can be written in French or English, the appearance of "Le" is rather random. If the detection and removal module of "Le" is invoked for each date image, it is not efficient, and "Le" detection may introduce errors. Therefore the strategy used to process "Le" is to first obtain recognition results without considering "Le". If the recognition results are acceptable with high confidence, output the results; otherwise, the detection of "Le" is conducted. If a "Le" candidate can be found in the detection, compare the recognition results before and after removing the "Le" candidate, and finally output the result with the higher confidence value. The outline of this approach is shown in Figure 34. In this figure, the "recognition of date" includes all steps in our system after the preprocessing, and the output can be a rejection. It can be observed that the main problem in "Le" processing is the detection of the "Le" candidate.

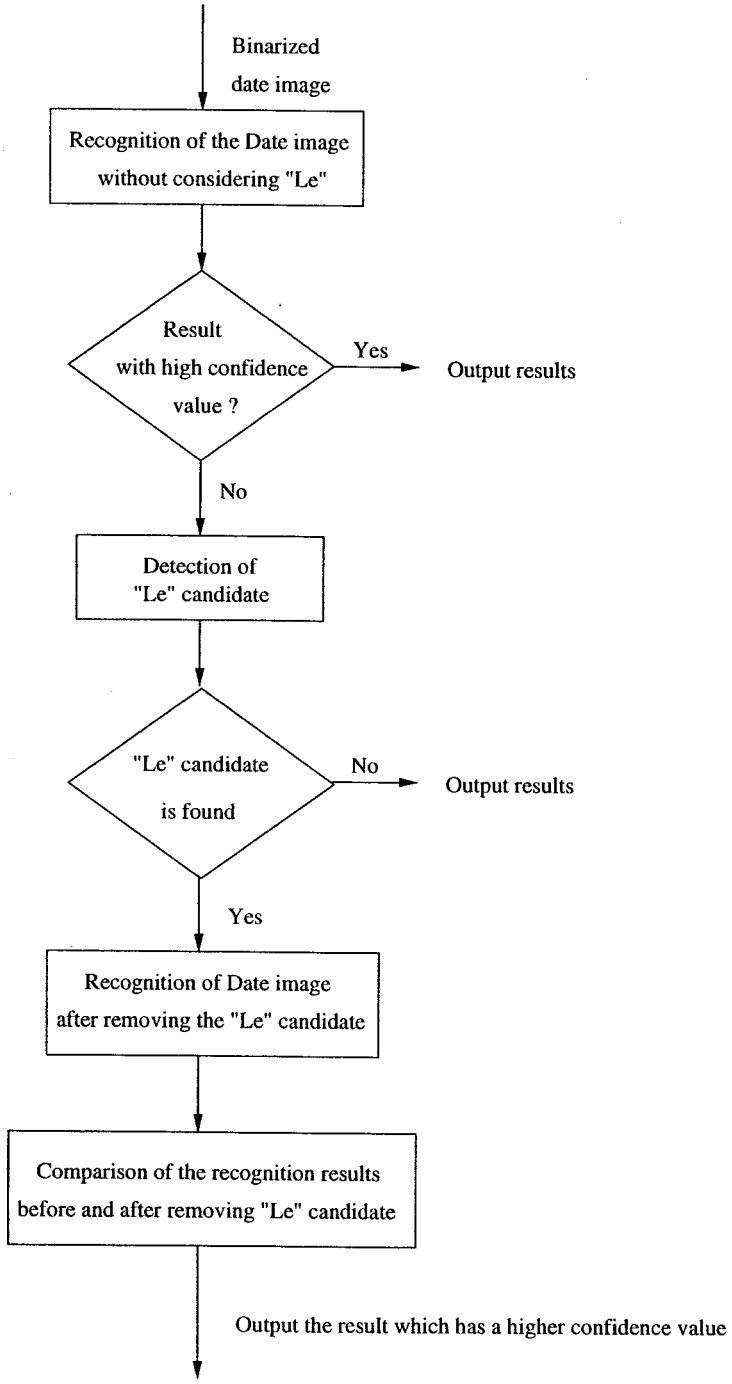

Figure 34: Diagram for detection and removal of "Le"

Detection of "Le" Candidate

The presence of "Le" can be characterised by the following:

- If "Le" is used, it always appears at the extreme left end.
- "Le" is always followed by *Day* field.

In addition, some people write "Le" as two isolated letters 'L' and 'e', while others write it in a cursive manner as one connected component.

Since no significant structural features can be found for the handwritten "Le", a cursive word recognizer is used to detect "Le". MLPB (see Chapter 6), which has been trained for month word recognition, considers "Le" as one class. However, the shapes of some numerals or numeral pairs are similar to that of "Le" (i.e. '6', '12' and '14', etc.), and sometimes these numeric data are recognized by the word recognizer as "Le" with high confidence values. In addition, some beginning letters of month words (such as 'F' and 'J') are also easily recognized as "Le" with high confidence values. Therefore, it is not reliable to detect "Le" by the word recognizer alone. Our "Le" detection procedure also makes use of recognition results from a digit recognizer [133], and the knowledge that "Le" is always followed by *Day* field which is numeric.

The main steps of the "Le" detection module are as follows:

1. Search from the leftmost end of the date image and select the first candidate component whose size is larger than a threshold and whose position is not above the UPPERHALF reference line. If the number of connected components to the right of this candidate within the date image is less than 3, return FALSE; otherwise go to next step.

2. Obtain the recognition results of the candidate component from both the word recognizer and the digit recognizer. If the candidate is recognized as "Le" with high confidence by the word recognizer, and the confidence value returned from the digit recognizer is below a threshold, then this component is likely to be

"Le" and the first main component (whose height is larger than a threshold) to its right is considered. If the recognition result of this component from the digit recognizer show a high confidence value, a "Le" candidate is obtained, and its confidence value is assigned (the confidence value from the word recognizer); otherwise the confidence value of the candidate is set to 0.

3. Search from the leftmost end of the date image and select two main components as the candidate, and use the same method as in step 2 to obtain the confidence value of this candidate. If the number of the connected components following this candidate is less than 3, the confidence value is set to 0.

4. If at least one candidate with non-zero confidence value can be found from the steps above, select the "Le" candidate with higher confidence value, and return TRUE together with the position of the "Le" candidate; otherwise return FALSE.

When a "Le" is detected, according to the strategy shown in Figure 34, the date recognition results before and after removing the "Le" candidate are compared and the result with higher confidence value is the output. Here the confidence value of the recognition result is the average confidence value of the three fields (*Day*, *Month* and *Year* recognition), and if the recognition result is not a valid date, the corresponding confidence value is reduced.

To summarize, the processing procedure for "Le" is based on the characteristics of "Le", and it makes use of the recognition results from the word and digit recognizers. The method has been proven effective by experiments (results will be given in the section on experimental results below). However, since this procedure heavily depends on the recognition results from the word and digit recognizers, which are not always reliable, two main types of errors may occur: (i) a "Le" exists, but is not detected by the procedure; and (ii) a part of *Day* or *Month* field is misrecognized as "Le" when it does not exist.

7.2 Verification Module

For a commercial application such as a cheque processing system, reliability is a very important consideration. In our date processing system, a two-level verification module is implemented in the postprocessing stage to further improve the reliability and performance of the system.

Generally speaking, a verification module aims to confirm, reject or negate the recognition results. Negation means the verifier concludes that "the recognition results are wrong", which may lead to using another recognition module. Rejection means the recognition results cannot be accepted by the verifier due to insufficient knowledge to give a precise verification, which may lead to a rejection of the input pattern. The verification module used in our date processing system is shown in Figure 35.

The segmentation-specific verifier in the first level is a negation scheme achieved by detecting possible errors made in segmentation. The recognition-specific verifier in the second level is designed to confirm valid and reliable recognition results, and to reject others.

More details about the two verifiers and Segmentation module #2 are given below.

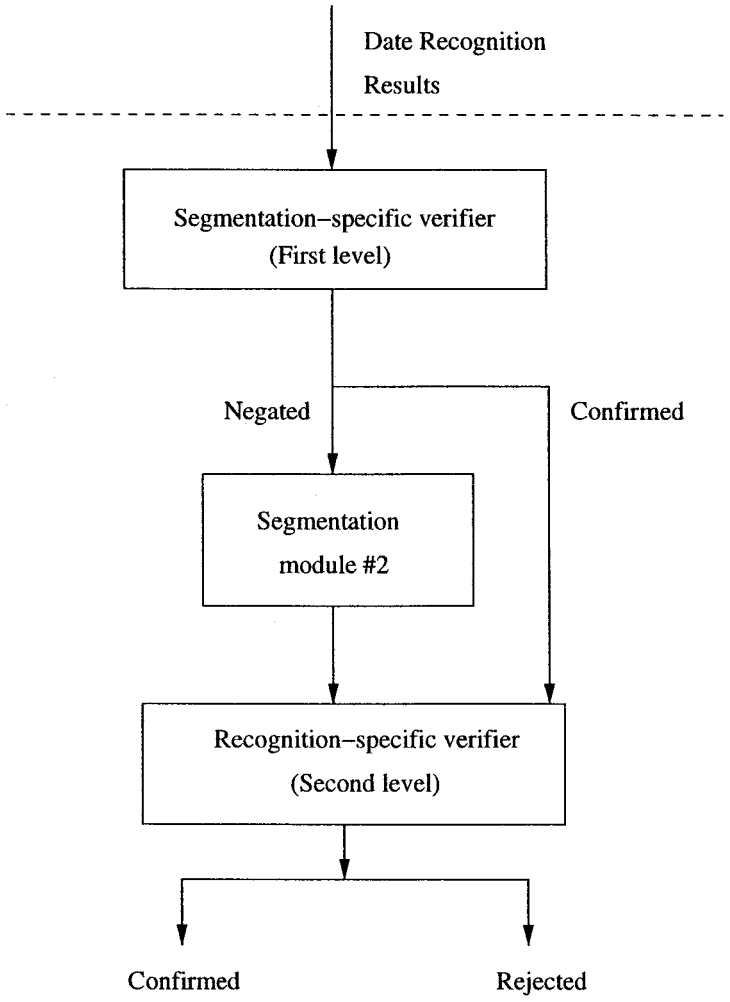

Figure 35: Diagram of verification module

7.2.1 Segmentation-specific Verifier

In our experiments, we find that some segmentation errors are unavoidable by the methods given above, especially in *Day&Month* segmentation when gap_{DM} is used as the separator. It is possible that gap_{DM} does not exist among the four gap candidates based on the two distance measures, and the real gap_{DM} cannot be found from our segmentation module. (This situation is less serious for punctuation separators, for which more candidates can be obtained by adjusting the thresholds for candidate detection, after which the candidates can be verified by our segmentation module). In addition, some other segmentation problems may appear when gap_{DM} is used as the separator, e.g. sometimes a wide gap_{BBD_1} is easily misconfirmed as gap_{DM} in the first confirmation level (see Section 5.3.2) bacause users write the cheques in free styles. For these segmentation problems, one solution is to find a method to reject these errors at the postprocessing stage. Here instead of just rejecting the errors, we try to find the errors and correct some of them. The segmentation-specific verifier is designed to negate some segmentation results which may contain errors so that another segmentation module is used to try to find the real gap_{DM}. (The current segmentation-specific Verifier only deals with segmentation problems when gap_{DM} is detected as the separator between *Day* and *Month* fields.)

Some segmentation errors are shown in Figure 36. In the first example, the gap after 'v' is determined as gap_{DM} by our previous method, and the left part of the gap_{DM} is assigned to *Day* (which is recognized as a invalid *Day* "2550"). The reason for this error is that the real gap_{DM} is not among the four gap candidates. In the second example, the gap after '2' is detected as the gap_{DM} by our previous method, and in the third example, the part of "november" at the right end is separated from the word and assigned to *Day*. The reason for these two errors are that wrong gap_{DM}s are confirmed at the first confirmation level.

A method for detecting these kinds of segmentation errors is proposed here, and the main idea of this method is derived from detailed analyses of examples similar to

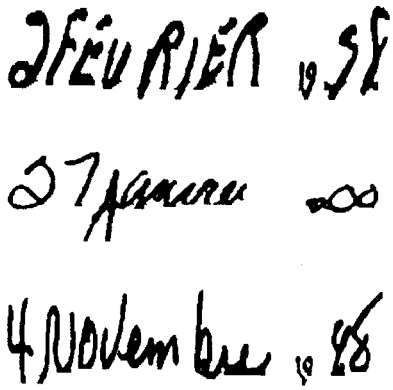

Figure 36: Examples of date images with segmentation errors

those given in Figure 36. If a gap_{DM} is detected and the corresponding recognition results indicate that *Month* is written in letters, and the confidence value of *Month* recognition is low, then the *Month* is possibly wrong. The *Month* may be just part of the real *Month* (see the first example in Figure 36), or it may contain part of the real *Day* (see the second and third examples in Figure 36). Therefore, in order to detect these segmentation errors, two cases of wrong *Month*s are considered. In the first case, the *Day* would be checked to see if it would contain part of the real *Month* (because the *Month* would be just part of the real *Month*). In the second case, connected component(s) at the end(s) of the *Month* would be considered to see if it would from a valid *Day* together with a possible recognized adjacent *Day* segment (refer to the second and third examples in Figure 36). This method can be described in detail as follows when the separator detected in *Day&Month* segmentation is a gap:

- If this gap is obtained from horizontal bounding box distance, and the distance exceeds a high threshold, it will be accepted as gap_{DM} because it is of high confidence to be a true separator. If this is not the case, and *Month* is written in letters, and the confidence value from the month word recognizer is less than a threshold, check the following aspects.

- Check the recognition result of the *Day* field. If the recognition result indicates

that the *Day* is invalid (it may contain part of the real *Month*), return TRUE, which means a possible error is found; otherwise continue with the next step.

- If *Day* consists of only one numeral and there are at least two connected components in the *Month* field, check right neighbor (when *Day* is at the left end) or left neighbor (when *Day* is at the right end) of the *Day*. If the first main component can be recognized by the digit recognizer with high confidence value, and the recognition result of the *Day* with this main component can be a valid *Day*, return TRUE; otherwise continue with the next step.

- Check the rightmost end of the *Day&Month* field (when *Day* is at the left end) or the leftmost end of the *Day&Month* field (when *Day* is at the right end). If the first main component with a certain height in this end can be recognized by the digit recognizer with high confidence value, return TRUE; otherwise return FALSE.

7.2.2 Segmentation Module Based on *Day* Searching

When a date image is negated by the segmentation-specific verifier, we know that an error may have occurred in the *Day&Month* segmentation by the gap_{DM} detection. Since it is possible that gap_{DM} does not exist among the four gap candidates based on the two distance measures, we try to find other gap candidates to see if the real gap_{DM} can be found among them. Since *Day* must be located at one end of the *Day&Month* field, we use the following method to obtain new gap candidates.

1. Search from the leftmost end of *Day&Month* field and select two gap candidates, one after the first main component, and the other after the second main component. (The main component is the one whose height exceed a threshold).

2. Similarly, search from the rightmost end of *Day&Month* field and select two gap candidates.

3. Use multi-hypotheses generation and evaluation method to obtain gap_{DM} from the four gap candidates.

After obtaining the new gap_{DM}, the recognition result based on this separator will be compared with the old recognition result. As in the processing of "Le", if the new result based on the new gap_{DM} has a higher confidence value, it is accepted; otherwise the old output is sent to the recognition-specific verifier, which is likely to reject it.

The recognition-specific verifier is used to make the final decision for our date processing system, and it will be discussed in the following section when experimental results are given.

7.3 Experimental Results

Several experiments have been designed to test the performance of our date processing system. In Chapters 4 and 6, experimental results about writing style analysis and cursive month word recognition have been presented. In this section the performance of both the date segmentation module and the entire system will be given, together with an error analysis.

7.3.1 Date Image Segmentation

The training set for developing the date segmentation module (described in Chapter 5) is Training Set 2 (see Chapter 3). In the following, some experimental results on Test Set (see Chapter 3) will be given. Test Set is derived from CENPARMI_IRIS Cheque database, and it contains 3399 date images, of which 1219 samples are written in English, and the other 2180 samples are written in French. The segmentation results based on different rejection thresholds are given in the following two tables (Table 39 and Table 40) for English and French samples respectively.

Some comments on these results are given below:

- The rejection

  Different rejection criteria are used to obtain different correct rates. The low rejection rate is obtained by trying to recognize every date sample, and a rejection is made only when no separator can be found by our segmentation method.

Table 39: Performances of date segmentation system for the English set

	Correct (%)	Rejection (%)	Error (%)	Reliability (%)
low rejection	90.40	1.81	7.79	92.06
medium rejection	74.57	21.16	4.27	94.59
high rejection	48.65	50.20	1.15	97.69

Table 40: Performances of date segmentation system for the French set

	Correct (%)	Rejection (%)	Error (%)	Reliability (%)
low rejection	82.94	4.22	12.84	86.59
medium rejection	65.69	26.97	7.34	89.95
high rejection	40.00	58.58	1.42	96.57

To achieve this low rejection rate the *Year* detection module has been modified. Since the *Year* detection module depends on the recognition results from a digit recognizer and some touching numeral strings cannot be separated and recognized correctly by the recognizer, many rejections are normally made in the *Year* detection module. To achieve the low rejection rate, we still return a possible *Year* even if it is not recognized as a valid *Year* with either 19 (or 20) ** or ** patterns. We first try to find two possible *Year* fields from the two ends of the date field by using the methods discussed in *Year* detection. In the event that no *Year* can be returned by the original *Year* detection module, but one of the two possible *Year* fields can be recognized by the digit recognizer with a certain confidence value, we return it as the *Year* field. (If both of the two possible *Year* fields can be recognized by the digit recognizer with a certain confidence value, we return the field at the right end of the date zone as the *Year* field.)

For the low rejection rate, a rejection is made when at least one of the three fields corresponding to *Year*, *Month* and *Day* cannot be found, e.g., *Day&Month* is written or binarized as one component or the *Year* field cannot be found. If

strong noise such as a big blob of ink (due to improper binarization) or too many components have been detected in the preprocessing stage, a rejection is also made.

For medium and high rejection rates, the recognition result should be a valid date and the average confidence value of the three fields should exceed a threshold; otherwise a rejection is made. Of course this includes some cases in which it is impossible to decide which field is the *Day*, and which field is the *Month*. This occurs when two fields are written in numerals and both of them are less than "12". Under these circumstances, we output the result in the order in which it is written together with an indication. For other cases when *Month* is written in numerals, the field which has a recognition value greater than 12 is assigned to the *Day* field. If the recognition results for both fields are greater than 12, the date image is rejected by the parser.

- The performance

  The performance for the English set is better than that for the French set. Several reasons can account for this difference. First, based on the database analysis in Chapter 3, it was found that more French cheques have the free format, that is, neither "19" nor "20" is printed on the date zone. With the free format, more variations exist and detecting the handwritten *Year* is more difficult than detecting the machine-printed "19" (or "20"). Therefore more errors and rejections occur in the date images with the free format. In addition, the article 'Le' sometimes used at the beginning of French date zones, together with freer writing styles on French cheques, also increase the difficulty of segmentation.

  In order to compare the performances from the current system with those from the previous system [32], we also test the previous system on the same English test set. Since the previous system cannot process date images with free format, these samples are removed from the English set to produce a new set of 1135 samples. The comparison is given in Table 41.

Table 41: Performance comparison of the current system and the previous system

	Correct (%)	Rejection (%)	Error (%)
Previous System	71.12	8.04	20.84
Proposed System	90.75	1.76	7.49

The results for the proposed system are shown with the low rejection rate. Since the previous system is only trained on the CENPARMI Cheque datebase, a direct comparison is not entirely valid. However since the date images from the two databases have very similar layout after removing the samples with the free format, we can conclude that a very significant improvement in performance has been achieved by the proposed system.

Some date images with successful segmentation by using the proposed system are shown in Figure 37.

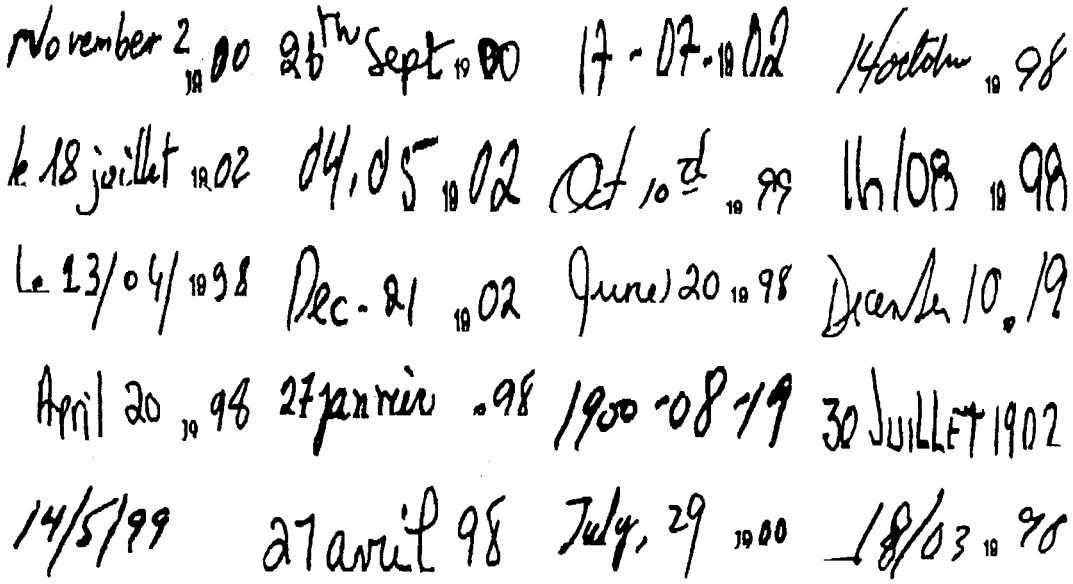

Figure 37: Examples of date images with successful segmentation

- Error analysis

  Based on our experiments on Training Set 2 (see Chapter 3), the errors made can be categorized into two classes. The first class contains date images of very poor quality, and it includes: (i) date images having touching fields; (ii) strong noise introduced by improper binarization; and (iii) incomplete dates with one of the three fields missing. Some examples of such cases are shown in Figure 38.

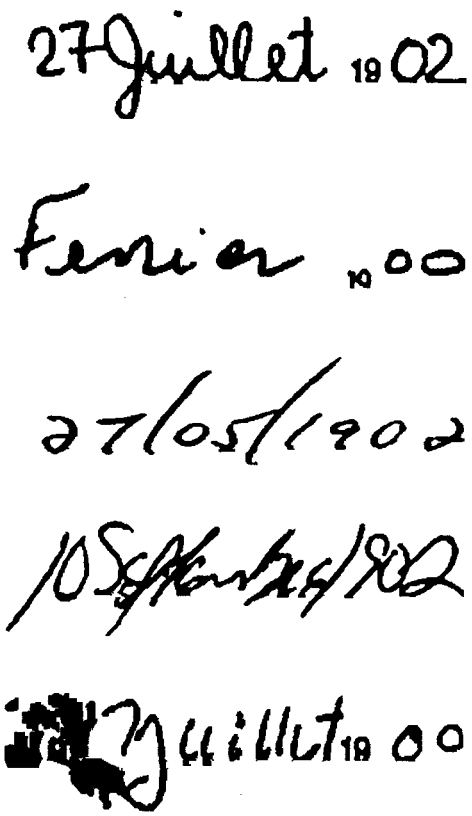

Figure 38: Examples of date images with segmentation errors of the first type

Our current segmentation module cannot process this type of date image, so a rejection is usually made when the high rejection rate is imposed. Based on the analysis on the training set, about 40% of the errors belong to this type.

The second type of error consists of errors generated by all segmentation modules in the system. Figure 39 shows several samples with this type of error.

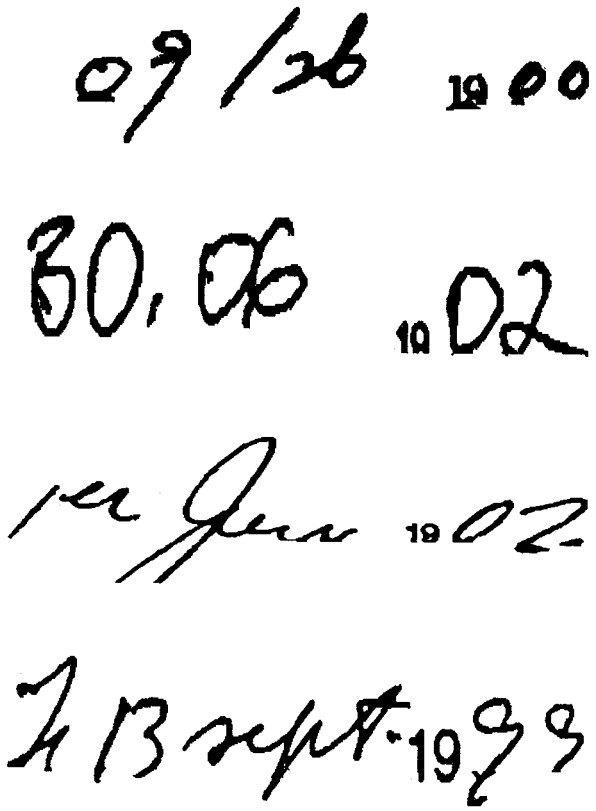

Figure 39: Examples of date images with segmentation errors of the second type

In the first sample, the slash '/' is not detected as a separator, and the '/' together with the following numeral string is assigned to the *Month* field and it is recognized as "Feb". In the second sample, the first field is assigned to the *Month* field and it is determined to be written in letters ("Oct"). The other two samples have errors caused by the "Le" and suffix processing modules, where both the suffix and the "Le" are not found. (Only about 72% of "Le"s can be found by the current "Le" processing module.) In addition, only one processing

system is used for all cheques (English and French samples). The "Le" detection module sometimes introduces the error of removing a part of *Day* or *Month* field and recognizing it as "Le" even for English cheques.

The aim of this thesis is to reduce this type of error. It is found that separator detection introduces most of the segmentation errors of this type, and when the free format is used, this type of error occurs more often.

- The efficiency

  As presented in previous chapters, the date segmentation is divided into two stages in order to improve the performance and efficiency of the system. A knowledge-based segmentation module is used in the first stage to solve most segmentation cases. Ambiguous cases are handled by multi-hypotheses generation module and evaluation module at a later stage. Tables 42 and 43 show the percentage of the date images handled by the different modules at the different stages.

Table 42: Segmentation at different stages for English set

	No. of images	Percentage (%)
Total no.	1197	100
Stage 1	888	74.19
Stage 2	309	25.81

Table 43: Segmentation at different stages for French set

	No. of images	Percentage (%)
Total no.	2088	100
Stage 1	1491	71.41
Stage 2	597	28.59

In the above tables, the total number of images represents the number of images which can be segmented by our system as in the case with the low rejection rate.

The corresponding correct rates in the two stages are given in Tables 44 and 45.

Table 44: Correct rates of segmentation at different stages for English set

Stage	Correct (%)	Error (%)
Stage 1	94.03	5.97
Stage 2	78.97	21.03

Table 45: Correct rates of segmentation at different stages for French set

Stage	Correct (%)	Error (%)
Stage 1	89.81	10.19
Stage 2	71.36	28.64

From Tables 44 and 45, we can see that the correct rates in the first stages are much higher than those in the second stages. However, since difficult cases of *Day&Month* segmentation are handled in the second stage, *Day&Month* images of very poor quality (e.g. *Day&Month* images having touching fields, containing strong noise, or wrong *Day&Month* images produced by error *Year* detection) are usually processed in the second stage. Therefore, the performance of segmentation in the second stage is not comparable with that in the first stage. In addition, the final segmentation performance improves about 2.0% in correct rate on the test set by using the segmentation-specific verifier.

7.3.2 Overall Performances

The overall performances are given in Tables 46 and 47 for the English and French subsets respectively. In Tables 46 and 47, the rejections are made under the same conditions as in Tables 39 and 40. Some discussions on these results are given below:

- The errors

  The errors of the date processing system mainly come from segmentation, month

Table 46: Performances of date processing system for the English set

	Correct (%)	Rejection (%)	Error (%)
low rejection	62.34	1.81	35.85
medium rejection	61.69	21.16	17.15
high rejection	44.55	50.20	5.25

Table 47: Performances of date processing system for the French set

	Correct (%)	Rejection (%)	Error (%)
low rejection	57.75	4.22	38.03
medium rejection	53.67	26.97	19.36
high rejection	36.42	58.58	5.00

word misrecognition and/or numeral misrecognition. The performance of the date segmentation has been discussed in the previous section, and some sample errors have been shown in Figure 38 and Figure 39.

For cursive month word recognition, the recognition rate of 85.36% has been given in Chapter 6 already (29 classes are considered here for both English and French month words). Some sample errors returned by this cursive month word recognizer are shown in Figure 40. In the first sample, "May" is misrecognized as "Aug". In the second sample, "AUG" is misrecognized as "Mars". Here we notice that when all letters in a month word are written in capital format, the performance of the month word recognizer is not good. For the third and fourth samples, "Janvier" and "Août" are misrecognized as "Juin" and "August", respectively.

For the numeral recognition, the recognizer used was originally developed for processing the courtesy amount written on bank cheques [133]. A 74% recognition rate (without rejection) has been reported for processing the courtesy amount written on bank cheques [32, 133]. Unlike *Day*, or *Month* (when written in numerals), each of which contains at most two numerals, and *Year* which

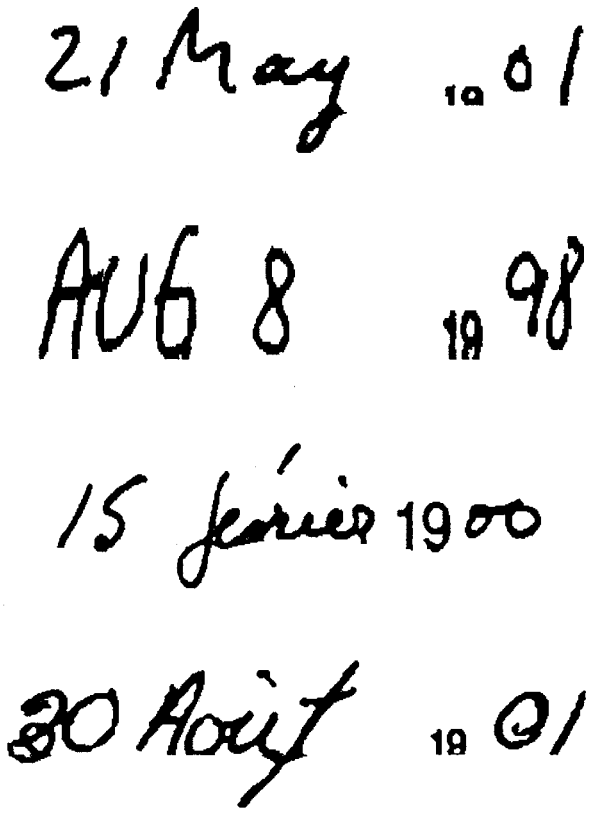

Figure 40: Examples of errors from the cursive month word recognizer

contains at most four numerals, the courtesy amount can be any number. Therefore, the digit recognizer does not impose restrictions on the number of output numerals or the range of the output numbers (such as *Day* should never exceed 31, and *Month* should never exceed 12). Some sample errors returned by this digit recognizer in our date processing system are shown in Figure 41.

Most of the errors returned from the digit recognizer are caused by touching numeral pairs or numeral strings, as shown in the first four samples. In the first sample, touching "01" is misrecognized as "0". In the second sample, "22" is misrecognized as "2", and "01" is misrecognized as "4". In the third sample, "1900" is misrecognized as "190". In the fourth sample, "1998" is misrecognized

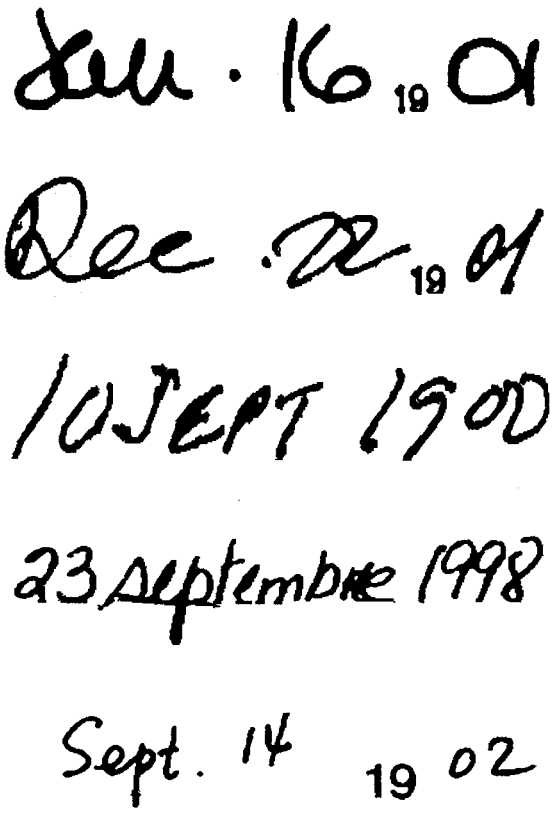

Figure 41: Examples of errors from the digit recognizer

as "198". In addition, since the digit recognizer was originally developed for processing courtesy amounts, which can contain punctuations (period, comma, etc.), some errors are caused by misinterpreting a numeral '1' as a comma or misinterpreting a small numeral '0' as a period. For the last sample, the small numeral '1' is recognized by the digit recognizer as a period so that the whole *Day* field is recognized as '4'.

- The rejection

  As discussed above, different rejection criteria are used in our date recognition system to obtain different recognition results. For example, if the low rejection

rate is considered, all the samples shown in Figures 38, 39, 40, and 41 are misrecognized by the date processing system. However, if the high rejection rate is considered, most of the samples shown in these figures are rejected by the date processing system, i.e. except the first sample in Figure 39 ("09/26 1900"), the first two samples in Figure 40 ("21 May 1901" and "AUG 8 1998"), and the last sample in Figure 41 ("Sept. 14 1902"), all the other samples shown in these figures are rejected by the system. Therefore, a high reliability can be obtained by using a high rejection rate.

Chapter 8

Conclusion

8.1 The Contribution

In Canada, the ability to process handwritten dates automatically is very important; nevertheless, the system developed in this thesis (which is an extension of a previous work [33]) is the only publication on processing date zones on Canadian bank cheques. Perhaps this is an indication of the complexity of the task involved. In order to enable the system to become a useful component for processing cheques in a real application environment, the performance of the previous system had to be improved. At the same time, the research of this thesis is conducted on a new database consisting of real-life standard cheques. As compared with the previous work which used an "artificial" database of cheques carefully designed to have similar size and layout but without background, many new problems have to be solved when real-life cheques are processed. These problems include the noise introduced by improper preprocessing, freer formats for writing a date when there is no century symbol on the date zone, etc. For this reason, this thesis is making an original contribution. Besides this, the contributions can be summarized as follows:

1. Date processing system

   A knowledge-based module has been proposed for the date segmentation [139, 85, 153] and a new cursive month word recognizer has also been presented

based on a combination system [152]. The interaction between the segmentation and the recognition stages has been properly established by using the multi-hypotheses generation and evaluation strategy. In addition, a verification module with two levels is designed in the postprocessing stage to correct some errors and reject invalid results, which further improves the performance of the system. As a result, very promising performances with different reliability rates have been obtained, which are much better than those produced by the previous system.

2. Date segmentation

   In order to be more effective and efficient, the date segmentation is implemented at different levels using knowledge obtained from different sources. For the ambiguous cases, only soft decisions are taken at the lower level and the final decision is based on a criterion of coherence of all the available information so that more reliable results can be obtained. Our date segmentation system includes a knowledge-based segmentation module, a multi-hypotheses generation and evaluation module, and a segmentation module within a segmentation-specific verifier. These modules are invoked under different situations, and simple segmentation cases can be efficiently solved by simply using the knowledge-based segmentation module. In the knowledge-based module, two approaches have been used to estimate the writing style of the date. One approach is based on a *distance_to_numeral* measure, and the other is an effective NN ensemble system. These approaches are very useful for correct segmentation.

3. Date recognition

   A new cursive month word recognizer is presented based on a combination of classifiers. An effective conditional combination topology and a newly modified Product combination rule are proposed in this combination system.

4. A/N differentiation

   In the knowledge-based segmentation module, in order to estimate the writing

style of the date, an effective NN ensemble system is proposed [88] to differentiate handwritten alphabetic words from numeric strings (A/N). In this method, multi-layer perceptron networks are first considered, and the combination of neural networks is then implemented. We extensively investigated the use of effective features, design of neural networks, creation of neural network ensembles, and combination methods for the ensembles created. Several new methods have been proposed in these aspects, which can also be applied to solve general pattern recognition problems. In addition, this A/N differentiation module can be applied to process other types of documents.

5. Database establishment
   Much effort has been made to establish the CENPARMI_IRIS Cheque database, especially for building the database of date images.

8.2 Future Work

This research has provided a solid basis for the real-life application of a date recognition module in bank cheque processing systems. However, the work is far from finished, and future work may include the following aspects:

- Improve the performance of the date segmentation system
  The performance needs to be further improved especially when the free date format is used. Currently the *Year* detection module for the free date format adopts a procedure which simply uses structural features for *Year* candidate detection, and uses recognition results of a digit recognizer for final *Year* confirmation. Other strategies need to be considered. For example, the interaction between *Year* detection module and *Day&Month* segmentation module can be established, and a multi-hypotheses generation and evaluation module which considers all the *Day*, *Month* and *Year* fields can be used for solving difficult cases. In addition, solutions to some complex segmentation problems, e.g. separating two connected fields, etc. should be found.

- Develop a digit recognizer for date recognition

  The digit recognizer used in our date processing system had been designed for courtesy amount recognition. For date recognition, some restrictions on the number of output numerals or the range of output number (e.g. *Day* and *Month* should never exceed 31 and 12 respectively) can be used to develop a better date-specific digit recognizer, and overall performance improvements can be expected based on the new digit recognizer.

- Study other strategies for date processing

  Other strategies, e.g. the HMM-based approach used in [105] to process date images on Brazilian bank cheques, should be investigated and compared with our current method. Some experiments can be designed to process date images on Canadian bank cheques by using these new methods, and better performances can be expected.

- Industrial application

  The long term goal of this date processing module is its application in a real cheque processing system. For this purpose, the system has to be further trained and tested on larger real-life databases. In order to obtain more reliable results, a more effective verification module is needed. Since the final recognition-specific verifier used in the current system is simply based on the recognition results, its reliability can be improved by incorporating other knowledge. In addition, some application-specific knowledge can be used, e.g. information about the current year, current month, and current day, etc., and this new knowledge can be used in different ways to improve the performance of the date processing system.

Bibliography

[1] I. S. I. Abuhaiba and P. Ahmed. A fuzzy graph theoretic approach to recognize the totally unconstrained handwritten numerals. *Pattern Recognition*, 26(9):1335–1350, 1993.

[2] A. Agarwal, L. Granowetter, K. Hussein, and A. Gupta. Detection of courtesy amount block on bank checks. In *Proc. of 3rd Int. Conf. on Document Analysis and Recognition*, pages 748–751, Montreal, Canada, 1995.

[3] M. Ahmed and R. K. Ward. An expert system for general symbol recognition. *Pattern Recognition*, 33:1975–1988, 2000.

[4] F. M. Alkoot and J. Kittler. Experimental evaluation of expert fusion strategies. *Pattern Recognition Letters*, 20(11-13):1361–1369, 1999.

[5] F. M. Alkoot and J. Kittler. Improving the performance of the product fusion strategy. In *Proc. of 15th Int. Conf. on Pattern Recognition, vol.2*, pages 164–167, Barcelona, Spain, 2000.

[6] N. Arica and F. Yarman-Vural. An overview of character recognition focused on off-line handwriting. *IEEE Trans. on Systems, Man and Cybernetics – Part C: Applications and Reviews*, 31(2), May 2001.

[7] E. Bauer and R. Kohavi. An empirical comparison of voting classification algorithms: bagging, boosting, and variants. *Machine Learning*, 36(1/2):105–139, July/August 1999.

[8] I. Bazzi, R. Schwartz, and J. Makhoul. An omnifont open vocabulary OCR system for English and Arabic. *IEEE Trans. on Pattern Anal. and Mach. Intell.*, 21:495–504, June 1999.

[9] H. D. Block, B. W. Knight, and F. Rosenblatt. Analysis of a four layer serious coupled perceptron. *Rev. Mod. Phys.*, 34:135–152, 1962.

[10] L. Breiman. Bagging predictors. *Machine Learning*, 24:123–140, 1996.

[11] A. S. Britto, R. Sabourin, E. Lethelier, F. Bortolozzi, and C. Y. Suen. Improvement in handwritten numeral string recognition by slant normalization and contextual information. In *Proc. of the 7th Int. Workshop on Frontiers in Handwriting Recognition*, pages 323–332, Amsterdam, The Netherlands, September 2000.

[12] H. Bunke and P. S. P. Wang. *Handbook of Character Recognition and Document Image Analysis.* World Scientific, 1997.

[13] C. J. C. Burges. A tutorial on support vector machines for pattern recognition. *Knowledge Discovery and Data Mining*, 2(2):121–167, 1998.

[14] R. G. Casey and G. Nagy. Decision tree design using a probabilistic model. *IEEE Trans. on Information theory*, 30:93–99, January 1984.

[15] S. Casey. Checks still America's favorite payment method. *Item Processing Report*, page 7, 1994.

[16] D. Y. Chen, J. Mao, and K. Mohiuddin. An efficient algorithm for matching a lexicon with a segmentation graph. In *Proc. of 5th Int. Conf. on Document Analysis and Recognition*, pages 543–546, Bangalore, India, 1999.

[17] M. Y. Chen, A. Kundu, and S. N. Srihari. Variable duration hidden Markov model and morphological segmentation for handwritten word recognition. *IEEE Trans. on Image Processing*, 4:1675–1688, 1995.

[18] Y. C. Chim, A. A. Kassim, and Y. Ibrahim. Dual classifier system for handprinted alphanumeric character recognition. *Pattern Analysis and Applications*, 1:155–162, 1998.

[19] K. Chung, Y. Shin, and J. Yoon. An application of feature selection based on neural network pruning to handwritten character recognition. In *Proc. of 5th Int. Workshop on Frontiers in Handwriting Recognition*, pages 263–266, Essex, UK, 1996.

[20] E. Cohen, J. J. Hull, and S. N. Srihari. Control structure for interpreting handwritten addresses. *IEEE Trans. on Pattern Anal. and Mach. Intell.*, 16(10):1049–1055, Oct. 1994.

[21] L. Cooper and M. W. Cooper. *Introduction to Dynamic Programming.* Pergamon Press, 1981.

[22] M. Côté, E. Lecolinet, M. Cheriet, and C. Y. Suen. Automatic reading of cursive scripts using a reading model and perceptual concepts. *Int. J. on Document Analysis and Recognition*, 1:3–17, 1998.

[23] Y. Le Cun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, November 1998.

[24] V. Delevski and S. Stankovoc. Recognition on handwritten digits based on their topological and morphological properties. *Advances in Pattern Recognition*, pages 516–523, eds. A. Amin, D. Dori, P. Pudil and H. Freeman Springer, 1998.

[25] T. G. Dietterich. Ensemble methods in machine learning. In *Proc. of 1st Int. Workshop on Multiple Classifier Systems*, pages 1–15, Cagliari, Italy, 2000.

[26] G. Dimauro, S. Impedovo, and A. Salzo. Automatic bankcheck processing: A new engineered system. *Int. J. of Pattern Recognition and Artificial Intelligence*, 11(4):467–504, 1997.

[27] J. Dong, A. Krzyzak, and C.Y. Suen. A fast SVM training algorithm. In *Proc. of International Workshop on Pattern Recognition with Support Vector Machines*, pages 53–67, Niagara Falls, Canada, August 2002.

[28] R. O. Duda and P. E. Hart. *Pattern Classification and Scene Analysis.* John Wiley & Sons, New York, 1973.

[29] G. Dzuba, A. Filatov, D. Gershuny, I. Kil, and V. Nikitin. Check amount recognition based on the cross validation of courtesy and legal amount fields. *Int. J. of Pattern Recognition and Artificial Intelligence*, 11(4):639–656, 1997.

[30] S. Edelman, T. Flash, and S. Ullman. Reading cursive handwriting by alignment of letter prototypes. *Int. J. of Computer Vision*, 5(3):303–331, 1990.

[31] A. El-Yacoubi, M. Gilloux, R. Sabourin, and C. Y. Suen. An HMM-based approach for off-line unconstrained handwritten word modeling and recognition. *IEEE Trans. on Pattern Anal. and Mach. Intell.*, 21(8):752–760, 1999.

[32] R. Fan. *Recognition of dates handwritten on cheques.* Master's thesis, Concordia University, Montréal, Québec, Canada, March 1998.

[33] R. Fan, L. Lam, and C. Y. Suen. Processing of date information on cheques. *Progress in Handwriting Recognition*, pages 473–479, eds. A. C. Downton and C. Impedovo, World Scientific, 1997.

[34] J. T. Favata. General word recognition using approximate segment-string matching. In *Proc. of 4th Int. Conf. on Document Analysis and Recognition*, pages 92–96, Ulm, Germany, 1997.

[35] A. Filatov, V. Nikitin, A. Volgunin, and P. Zelinsky. The AddressScriptTM recognition system for handwritten envelopes. In *Proc. of the 3rd Int. Association for Pattern Recognition Workshop on Document Analysis Systems*, pages 222–236, Nagano, Japan, November 1998.

[36] J. Franke, L. Lam, R. Legault, C. Nadal, and C. Y. Suen. Experiments with the CENPARMI database combining different classification approaches. In *Proc. of 3rd Int. Workshop on Frontiers in Handwriting Recognition*, pages 305–311, Buffalo, New York, May 1993.

[37] K. Fukunaga. *Introduction to Statistical Pattern Recognition.* Academic Press, Inc., 1990.

[38] P. Gader, J. M. Keller, and J. Cai. A fuzzy logic system for the detection and recognition of handwritten street numbers. *IEEE Trans. on Fuzzy Systems*, 3(1):83–95, 1995.

[39] P. D. Gader, B. Forester, M. Ganzberger, A. Billies, B. Mitchell, M. Whalen, and T. Youcum. Recognition of handwritten digits using template and model matching. *Pattern Recognition*, 24(5):421–431, 1991.

[40] K. A. Ghoneim and B. V. K. V. Kumar. Unified decision combination framework. *Pattern Recognition*, 31(12):2077–2089, 1998.

[41] G. Giacinto, F. Roli, and G. Fumera. Design of effective multiple classifier systems by clustering of classifiers. In *Proc. of 15th Int. Conference on Pattern recognition, vol.2*, pages 160–163, Barcelona, Spain, 2000.

[42] D. Guillevic and C.Y. Suen. Cursive script recognition applied to the processing of bank cheques. In *Proc. of 3rd Int. Conf. on Document Analysis and Recognition*, pages 11–14, Montréal, Canada, August 1995.

[43] D. Guillevic and C.Y. Suen. HMM word recognition engine. In *Proc. of 4th Int. Conf. on Document Analysis and Recognition*, pages 544–547, Ulm, Germany, August 1997.

[44] D. Guillevic and C.Y. Suen. HMM-KNN word recognition engine for bank cheque processing. In *Proc. of 14th Int. Conf. on Pattern Recognition*, pages 1526–1529, Brisbane, Australia, August 1998.

[45] D. Guillevic and C.Y. Suen. Recognition of legal amounts on bank cheques. *Pattern Analysis and Applications*, 1:28–41, 1998.

[46] I. Guyon and Pereira. Design of a linguistic postprocessor using varible memory length Markov models. In *Proc. of 3rd Int. Conf. on Document Analysis and Recognition*, pages 454–457, Montreal, Canada, 1995.

[47] T. M. Ha and H. Bunke. Off-line handwritten numeral recognition by perturbation method. *IEEE Trans. on Pattern Anal. and Mach. Intell.*, 19(5):535–539, May 1997.

[48] T. M. Ha, M. Zimmermann, and H. Bunke. Off-line handwritten numeral string recognition by combining segmentation-based and segmentation-free methods. *Pattern Recognition*, 31:257–272, 1998.

[49] K. Han and I. Sethi. An off-line cursive handwritten word recognition system and its application to legal amount interpretation. *Automatic Bankcheck Processing*, pages 295–308, eds. S. Impedovo, P. S. P. Wang and H. Bunke, World Scientific, Singapore, 1997.

[50] L. Hansen and P. Salamon. Neural network ensembles. *IEEE Trans on Pattern Anal. and Mach. Intell.*, 12(10):993–1001, October 1990.

[51] K. Hartman, J. D. Keeler, and J. M. Kowalsky. Layered neural networks with Gaussian hidden units as universal approximations. *Neural Computation*, 2:210–215, 1990.

[52] A. Hennig and N. Sherkat. Cursive script recognition using wildcards and multiple experts. *Pattern Analysis and Applications*, 4:51–60, 2001.

[53] L. Heutte, P. B. Pereira, O. Bougeois, J. V. Moreau, B. Plesis, P. Courtellemont, and Y. Lecourtier. Multi-bank check recognition system: Consideration on the numeral amount recognition module. *Int. J. of Pattern Recognition and Artificial Intelligence*, 11(4):595–618, 1997.

[54] T.K. Ho. The random subspace method for constructing decision forests. *IEEE Trans. on Pattern Anal. and Mach. Intell.*, 20(8):832–844, August 1998.

[55] K. Hornik, M. Stinchcombe, and H. White. Multilayer feedforward networks are universal approximators. *Neural Networks*, 2:359–366, 1989.

[56] G. F. Houle, D. B. Aragon, and R. W. Smith. A multi-layered corroboration-based check reader. *Document Analysis Systems II*, pages 495–546, J.J. Hull and S.L. Taylor (Eds.), World Scientific, 1998.

[57] G. F. Houle, D. B. Aragon, R. W. Smith, M. Shridhar, and D. Kimura. A multi-layered corroboration-based check reader. In *Proc. of the Int. Association for Pattern Recognition Workshop on Document Analysis Systems*, pages 495–546, Malvern, Pennsylvania USA, Oct. 1996.

[58] Y.S. Huang and C.Y. Suen. A method of combining multiple experts for the recognition of unconstrained handwritten numerals. *IEEE Trans. on Pattern Anal. and Machine Intell.*, 17(1):90–94, 1995.

[59] K. M. Hussein, A. Agarwal, A. Gupta, and P. S. P. Wang. A knowledge-based segmentation algorithm for enhanced recognition of handwritten courtesy amounts. *Pattern Recognition*, 32(2):305–316, 1999.

[60] S. Impedovo, P. S. P. Wang, and H. Bunke. *Automatic Bankcheck Processing.* World Scientific, Singapore, 1997.

[61] A. K. Jain, R. P. W. Duin, and J. Mao. Statistical pattern recognition: A review. *IEEE Trans. on Pattern Anal. and Mach. Intell.*, 22:4–38, January 2000.

[62] A. K. Jain and D. Zongker. Representation and recognition of handwritten digits using deformable templates. *IEEE Trans. on Pattern Anal. and Mach. Intell.*, 19:1386–1391, December 1997.

[63] H. J. Kang and S. W. Lee. Combining classifiers based on minimization of a Bayes error rate. In *Proc. of 5th Int. Conf. on Document Analysis and Recognition*, pages 398–401, Bangalore, India, 1999.

[64] G. Kaufmann and H. Bunke. A system for the automated reading of check amounts – some key ideas. In *Proc. of 3rd Int. Association for Pattern Recognition Workshop on Document Analysis Systems*, pages 302–315, Nagano, Japan, November 1998.

[65] G. Kaufmann and H. Bunke. Automated reading of cheque amounts. *Pattern Analysis and Application*, 3:132–141, 2000.

[66] G. Kaufmann and H. Bunke. Detection and correction of recognition errors in check reading. *Int. J. on Document Analysis and Recognition*, 2:211–221, 2000.

[67] F. G. Keenan, L. J. Evett, and R. J. Whitrow. A large vocabulary stochastic syntax analyzer for handwriting recognition. In *Proc. of 1st Int. Conf. on Document Analysis and Recognition*, pages 794–802, Saint Malo, France, 1991.

[68] G. Kim and V. Govindaraju. A lexicon driven approach to handwritten word recognition for real-time applications. *IEEE Trans. on Pattern Anal. and Mach. Intell.*, 19:366–379, 1997.

[69] G. Kim and V. Govindaraju. Handwritten phrase recognition as applied to street name images. *Pattern Recognition*, 31(1):41–51, 1998.

[70] G. Kim and V. Govindaraju. Bankcheck recognition using cross validation between legal and courtesy amounts. *Automatic Bankcheck Processing*, pages 195–233, eds. S. Impedovo, P. S. P. Wang and H. Bunke, World Scientific, Singapore, 1997.

[71] G. Kim, V. Govindaraju, and S. N. Srihari. Extension of handwritten word recognition method to street name images. In *Proc. of 5th Int. Workshop on Frontiers in Handwriting Recognition*, pages 221–226, Essex, England, 1996.

[72] G. Kim, V. Govindaraju, and S. N. Srihari. An architecture for handwritten text recognition systems. *Int. J. on Document Analysis and Recognition*, 2:37–44, 1999.

[73] H. Y. Kim and J. H. Kim. Handwritten Korean character recognition based on hierarchical random graph modeling. In *Proc. of 6th Int. Workshop on Frontiers in Handwriting Recognition*, pages 577–586, Taejon, Korea, 1998.

[74] J. Kim, K. Kim, C. P. Nadal, and C. Y. Suen. A methodology of combining HMM and MLP classifiers for cursive word recognition. In *Proc. of 15th Int. Conf. on Pattern Recognition, vol.2*, pages 319–322, Barcelona, Spain, September 2000.

[75] J. Kim, K. Kim, and C. Y. Suen. Hybrid schemes of homogeneous and heterogeneous classifiers for cursive word recognition. In *Proc. of 7th Int. Workshop on Frontiers in Handwriting Recognition*, pages 433–442, Amsterdam, the Netherlands, September 2000.

[76] J. H. Kim, K. K. Kim, and C. Y. Suen. An HMM-MLP hybrid model for cursive script recognition. *Pattern Analysis and Applications*, 3(4):314–324, 2000.

[77] J. Kittler. On combining classifiers. *IEEE Trans. on Pattern Anal. and Mach. Intell.*, 20(3):226–239, March 1998.

[78] S. Kneer, L. Personnaz, and G. Dreyfus. Handwritten digit recognition by neural networks with single-layer training. *IEEE Trans. on Neural Networks*, 3(6):962–968, 1992.

[79] S. Knerr, V. Anisimov, O. Baret, N. Gorski, D. Price, and J. C. Simon. The A2iA intercheque system: courtesy amount and legal amount recognition for French checks. *Automatic Bankcheck Processing*, pages 43–86, eds. S. Impedovo, P. S. P. Wang and H. Bunke, World Scientific, Singapore, 1997.

[80] S. Knerr and E. Augustin. A Neural Network - Hidden Markov model hybrid for cursive word recognition. In *Proc. of 14th Int. Conf. Pattern Recognition*, pages 1518–1520, Brisbane, August 1998.

[81] S. Knerr, O. Baret, D. Price, and J.C. Simon. The A2iA recognition system for handwritten checks. In *Proc. of the Int. Association for Pattern Recognition Workshop on Document Analysis Systems*, pages 431–494, Malvern, Pennsylvania USA, October 1996.

[82] T. Kohonen. The self-organizing map. *Proc. IEEE*, 78(9):1464–1480, 1990.

[83] A. Krzyzak, W. Dai, and C. Y. Suen. Unconstrained handwritten character classification using modified back propagation model. In *Proc. of 1st Int. Workshop on Frontiers in Handwriting Recognition*, pages 155–166, Montreal, Canada, 1990.

[84] L. Lam. Classifier combinations: Implementations and theoretical issues. In *Proc. of 1st Int. Workshop on Multiple Classifier Systems*, pages 77–86, Cagliari, Italy, June 2000.

[85] L. Lam, R. Fan, Q. Xu, and C.Y. Suen. Segmentation of date field on bank cheques. *Advances in Pattern Recognition*, pages 163–172, eds. F. J. Ferri, J. M. Inesta, A. Amin and P. Pudil, Springer, 2000.

[86] L. Lam and C. Y. Suen. Application of majority voting to pattern recognition: An analysis of its behavior and performance. *IEEE Trans. on Systems, Man, and Cybernetics – Part A: Systems and Humans*, 27(5):553–568, September 1997.

[87] L. Lam and C.Y. Suen. Structural classification and relaxation matching of totally unconstrained handwritten zip-code numbers. *Pattern Recognition*, 21(1):19–31, 1988.

[88] L. Lam, Q. Xu, and C. Y. Suen. Differentiation between alphabetic and numeric data using ensembles of neural networks. In *Proc. of 16th Int. Conf. on Pattern Recognition, vol.IV*, pages 40–43, Quebec City, Canada, August 2002.

[89] B. Lazzerini and F. Marcelloni. A linguistic fuzzy recognizer of off-line handwritten characters. *Pattern Recognition Letters*, 21:319–327, 2000.

[90] L. L. Lee, M. G. Lizarraga, N. R. Gomes, and A. L. Koerich. A prototype for Brazilian bankcheck recognition. *Int. J. of Pattern Recognition and Artificial Intelligence*, 11(4):549–570, 1997.

[91] M. Leroux, E. Lethelier, M. Gilloux, and B. Lemarie. Automatic reading of handwriten amounts on French checks. *Int. J. of Pattern Recognition and Artificial Intelligence*, 11(4):619–638, 1997.

[92] R. P. Lippmann and M. D. Richard. Neural network classifiers estimate bayesian a posterior probabilities. *Neural Computation*, 3:461–483, 1991.

[93] C. Liu, K. Nakashima, H. Sako, and H. Fujisawa. Handwritten digit recognition using state-of-the-art techniques. In *Proc. of 8th Int. Workshop on Frontiers in Handwriting Recognition*, pages 320–325, Niagara-on-the-Lake, Canada, August 2002.

[94] C. Liu, H. Sako, and H. Fujisawa. Integrated segmentation and recognition of handwritten numerals comparison of classification algorithms. In *Proc. of 8th Int. Workshop on Frontiers in Handwriting Recognition*, pages 303–308, Niagara-on-the-Lake, Canada, August 2002.

[95] K. Liu, C. Y. Suen, M. Cheriet, J. N. Said, C. Nadal, and Y. Y. Tang. Automatic extraction of baselines and data from check images. *Int. J. of Pattern Recognition and Artificial Intelligence*, 11(4):675–697, 1997.

[96] R. A. Lorie. A system for exploiting context in automatic recognition. *Document Analysis Systems*, pages 93–109, eds. A. Lawrence and A. Dengal, World Scientific, Singapore, 1994.

[97] Z. Lu, Z. Chi, W. C. Siu, and P. Shi. A background thinning-based approach for separating and recognizing connected handwritten digit strings. *Pattern Recognition*, 32:921–933, 1999.

[98] U. Mahadevan and R. C. Nagabushnam. Gap metrics for word separation in handwritten lines. In *Proc. of 3rd Int. Conf. on Document Analysis and Recognition*, pages 124–127, Montréal, Canada, 1995.

[99] U. Mahadevan and S. N. Srihari. Hypotheses generation for word separation in handwritten lines. In *Proc. of 5th Int. Workshop on Frontiers in Handwriting Recognition*, pages 453–456, Essex, England, 1996.

[100] T. A. Mai. *A knowledge-based approach to recognize unconstrained handwritten numerals.* Master's thesis, Concordia University, Montréal, Québec, Canada, April 1989.

[101] U. V. Marti and H. Bunke. Using a statistical language model to improve the performance of an HMM-based cursive handwriting recognition system. *Int. Journal of Pattern Recognition and Artificial Intelligence*, 15(1):65–90, 2001.

[102] L. Mico and J. Oncina. Comparison of fast nearest neighbour classifiers for handwritten character recognition. *Pattern Recognition Letters*, 19:351–356, 1998.

[103] M. L. Minsky and S. Papert. *Perceptrons: An Introduction to Computational Geometry.* Cambridge, MA:MIT Press, 1969.

[104] M. Mohamed and P. Gader. Handwritten word recognition using segmentation-free hidden Markov modeling and segmentation-based dynamic programming techniques. *IEEE Trans. on Pattern Anal. and Machine Intell.*, 18(5):548–554, 1996.

[105] M. Morita, A. El Yacoubi, R. Sabourin, F. Bortolozzi, and C. Y. Suen. Handwritten month word recognition on Brazilian bank cheques. In *Proc. of 6th Int.*

Conf. on Document Analysis and Recognition, pages 972–976, Seattle, USA, September 2001.

[106] C. Nadal and C. Y. Suen. Applying human knowledge to improve machine recognition of confusing handwritten numerals. *Pattern Recognition*, 26(3):381–389, 1993.

[107] Il-S. Oh and C. Y. Suen. Distance features for neural network-based recognition of handwritten characters. *Int. J. on Document Analysis and Recognition*, 1:73–88, 1998.

[108] M. Okada and M. Shridhar. Extraction of user entered components from a personal bankcheck using morphological subtraction. *Int. J. of Pattern Recognition and Artificial Intelligence*, 11(5):699–715, 1997.

[109] L. S. Oliveira, R. Sabourin, F. Bortolozzi, and C. Y. Suen. A modular system to recognize numerical amounts on Brazilian bank cheques. In *Proc. of 6th Int. Conf. on Document Analysis and Recognition*, pages 389–394, Seattle, USA, September 2001.

[110] D. W. Opitz and J. W. Shavlik. Actively searching for an effective neural network ensemble. *Connection Science*, 8(3 and 4):337–353, 1996.

[111] C. Parisse. Global word shape processing in off-line recognition of handwriting. *IEEE Trans. on Pattern Anal. and Machine Intell.*, 18(4):460–464, 1996.

[112] J. Park, V. Govindaraju, and S. N. Srihari. Efficient word segmentation driven by unconstrained handwritten phrase recognition. In *Proc. of 5th Int. Conf. on Document Analysis and Recognition*, pages 605–608, Bangalore, India, 1999.

[113] D. Partridge and W. B. Yates. Engineering multiversion neural-net systems. *Neural Computation*, 8:869–893, 1996.

[114] T. Pavlidis. Recognition of printed text under realistic conditions. *Pattern Recognition Letters*, 14:317–326, 1993.

[115] L. F. C. Pessoa and P. Maragos. Neural networks with hybrid morphological/rank/linear nodes: A unifying framework with applications to handwritten character recognition. *Pattern Recognition*, 33:945–960, 2000.

[116] L. R. Rabiner. A tutorial on Hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77:257–286, 1989.

[117] A. F. R. Rahman and M. C. Fairhurst. An evaluation of multi-expert configurations for the recognition of handwritten numerals. *Pattern Recognition*, 31(9):1255–1273, 1998.

[118] F. R. Rahman and M. C. Fairhurst. A new hybrid approach in combining multiple experts to recognize handwritten numerals. *Pattern Recognition Letters*, 18:781–190, 1997.

[119] F. R. Rahman and M. C. Fairhurst. Serial combination of multiple experts: a unified evaluation. *Pattern Analysis and Applications*, 2:292–311, 1999.

[120] F. R. Rahman and M. C. Fairhurst. Multiple expert classification: a new methodology for parallel decision fusion. *Int. J. on Document Analysis and Recognition*, 3:40–55, 2000.

[121] N. V. S. Reddy and P. Nagabhushan. A three-dimensional neural network model for unconstrained handwritten numeral recognition: a new approach. *Pattern Recognition*, 31(5):511–515, 1998.

[122] D.E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning internal representations by error propagation. *In Parallel Distributed Processing*, 1: Foundations:318–362, D.E. Rumelhart and J. L. McClelland (Eds.), MIT press, Cambridge, MA, 1986.

[123] C. Scagliola. Search algorithms for the recognition of cursive phrases without word segmentation. In *Proc. of 6th Int. Workshop on Frontiers in Handwriting Recognition*, pages 123–132, Taejon, Korea, 1998.

[124] J. Schurmann. *Pattern Classification-A Unified View of Statistical and Neural Approaches.* Wiley-Interscience, 1996.

[125] G. Seni and E. Cohen. External word segmentation of off-line handwritten text lines. *Pattern Recognition*, 27(1):41–52, 1994.

[126] A.J.C. Sharkey. On combining artificial neural nets. *Connection Science*, 8(3 and 4):299–313, 1996.

[127] Z. Shi and V. Govindaraju. Segmentation and recognition of connected handwritten numeral strings. *Pattern Recognition*, 30:1501–1504, 1997.

[128] M. Shridhar and A. Badreldin. A high-accuracy syntactic recognition algorithm for handwritten numerals. *IEEE Trans. on System, Man and Cybernetics*, 15(1):152–158, 1985.

[129] M. Shridhar, G. Houle, and F. Kimura. Handwritten word recognition using lexicon free and lexicon directed word recognition algorithms. In *Proc. of 4th Int. Conf. on Document Analysis and Recognition*, pages 861–865, Ulm, Germany, 1997.

[130] G. Soan. Cursive word recognition using a random field based hidden Markov model. *Int. J. on Document Analysis and Recognition*, 1:199–208, 1999.

[131] G. Srikantan, S. W. Lam, and S. N. Srihari. Gradient-based contour encoding for character recognition. *Pattern Recognition*, 29(7):1147–1160, 1996.

[132] T. Steinherz, E. Rivlin, and N. Intrator. Offline cursive script word recognition – a survey. *Int. Journal on Document Analysis and Recognition*, 2:90–110, 1999.

[133] N. W. Strathy. Handwriting recognition for cheque processing. In *Proc. of the 2nd Int. Conf. on Multimodal Interface*, pages 47–50, Hong Kong, January 1999.

[134] H. Su, B. Zhao, F. Ma, S. Wang, and S. Xia. A fault-tolerant Chinese check recognition system. *Int. J. of Pattern Recognition and Artificial Intelligence*, 11(4):571–594, 1997.

[135] C. Y. Suen, M. Berthod, and S. Mori. Automatic recognition of handprinted characters – the state of the art. *Proceedings of the IEEE*, 68(4):469–487, 1980.

[136] C. Y. Suen, J. Kim, K. Kim, Q. Xu, and L. Lam. Handwriting recognition – the last frontiers. In *Proc. of 15th Int. Conf. on Pattern Recognition*, pages 1–10, Barcelona, Spain, September 2000.

[137] C. Y. Suen, R. Legault, C. Nadal, M. Cheriet, and L. Lam. Building a new generation of handwriting recognition systems. *Pattern Recognition Letters*, 14:303–315, 1993.

[138] C. Y. Suen, K. Liu, and N. W. Strathy. Sorting and recognizing cheques and financial documents. In *Proc. of 3rd Int. Association for Pattern Recognition Workshop on Document Analysis Systems*, pages 1–18, 1998.

[139] C. Y. Suen, Q. Xu, and L. Lam. Automatic recognition of handwritten data on cheques – fact or fiction? *Pattern Recognition Letters*, 20(11-13):1287–1295, 1999.

[140] C.Y. Suen and L. Lam. Multiple classifier combination methodologies for different output levels. In *Proc. of 1st Int. Workshop on Multiple Classifier Systems*, pages 52–66, Cagliari, Italy, June 2000.

[141] C.Y. Suen, L. Lam, D. Guillevic, N. W. Strathy, M. Cheriet, J. N. Said, and R. Fan. Bank check processing system. *Int. J. of Image Systems and Technology*, 7:392–403, 1996.

[142] Y. Tang. Off-line recognition of Chinese handwriting by multifeature and multilevel classification. *IEEE Trans. on Pattern Anal. and Mach. Intell.*, 20:556–561, 1998.

[143] D.M.J. Tax, R.P.W. Duin, and M. van Breukelen. Comparison between product and mean classifier combination rules. In *Proc. of 1st Int. Association for Pattern Recognition Workshop on Statistical Techniques in Pattern Recognition*, pages 165–170, 1997.

[144] V. Vapnik. *The nature of statistical learning theory.* Springer-Verlag, New York, 1995.

[145] W. P. Waard. An optimised minimal edit distance for handwritten word recognition. *Pattern Recognition Letters*, 16(10):1091–1096, 1995.

[146] S. Wang, X. Zhu, and Y. Jin. Multiple experts recognition system based on neural network. In *Proc. of 13th Int. Conf. on Pattern Recognition*, pages 452–456, Vienna, Austria, August 1996.

[147] X. Wang, V. Govindaraju, and S. Srihari. Holistic recognition of handwritten character pairs. *Pattern Recognition*, 33:1967–1973, 2000.

[148] X. Wu and C. Y. Suen. Injection of human knowledge into the rejection criterion of a neural network classifier. In *Proc. of Joint 9th IFSA Word Congress and 20th NAFIPS International Conference*, pages 499–505, Vancouver, Canada, July 2001.

[149] X. Xiao and G. Leedham. Knowledge-based English cursive script segmentation. *Pattern Recognition Letters*, 21:945–954, 2000.

[150] S. L. Xie and M. Suk. On machine recognition of hand-printed Chinese characters by feature relaxation. *Pattern Recognition*, 21(1):1–7, 1988.

[151] L. Xu, A. Krzyzak, and C. Y. Suen. Methods of combining multiple classifiers and their applications to handwriting recognition. *IEEE Trans. on Systems, Man, and Cybernetics*, 22(3):418–435, 1992.

[152] Q. Xu, J. Kim, L. Lam, and C. Y. Suen. Recognition of handwritten month words on bank cheques. In *Proc. of 8th Int. Workshop on Frontiers in Handwriting Recognition*, pages 111–116, Niagara-on-the-Lake, Canada, August 2002.

[153] Q. Xu, L. Lam, and C. Y. Suen. A knowledge-based segmentation system for handwritten dates on bank cheques. In *Proc. of 6th Int. Conf. on Document Analysis and Recognition*, pages 384–388, Seattle, USA, September 2001.

[154] X. Ye, M. Cheriet, C. Y. Suen, and K. Liu. Extraction of bankcheck items by mathematical morphology. *Int. J. on Document Analysis and Recognition*, 2:53–66, 1999.

[155] D. Yu and H. Yan. Separation of touching handwritten mult-numeral strings based on morphological structural features. *Pattern Recognition*, 34:587–599, 2001.

[156] B. Zhang, M. Fu, H. Yan, and M. A. Jabri. Handwritten digit recognition by adaptive-subspace self organizing map (ASSOM). *IEEE Trans. on Neural Networks*, 10:939–945, 1999.

[157] L. Zhang. *Recognition of Courtesy Amounts on Bank Cheques Based on a Segmentation Approach.* Master's thesis, Concordia University, Montréal, Québec, Canada, August 2001.

[158] J. Zhou, Q. Gan, A. Krzyzak, and C. Y. Suen. Recognition of handwritten numerals by Quantum neural network. *Int. J. on Document Analysis and Recognition*, 2:30–36, 1999.

[159] J. Zhou, A. Krzyzak, and C. Y. Suen. Recognition and verification of touching handwritten numerals. In *Proc. of 7th Int. Workshop on Frontiers in Handwriting Recognition*, pages 179–188, Amsterdam, The Netherlands, September 2000.

[160] Jie Zhou. *Recognition and Verification of Unconstrained Handwritten Numerals.* PhD thesis, Concordia University, Montréal, Québec, Canada, Nov. 1999.

[161] Jun Zhou. *Segmentation of Legal Amounts on Bank Cheques.* Master's thesis, Concordia University, Montréal, Québec, Canada, Oct. 2001.