INFORMATION TO USERS

This manuscript has been reproduced from the microfilm master. UMI films the text directly from the original or copy submitted. Thus, some thesis and dissertation copies are in typewriter face, while others may be from any type of computer printer.

The quality of this reproduction is dependent upon the quality of the copy submitted. Broken or indistinct print, colored or poor quality illustrations and photographs, print bleedthrough, substandard margins, and improper alignment can adversely affect reproduction.

In the unlikely event that the author did not send UMI a complete manuscript and there are missing pages, these will be noted. Also, if unauthorized copyright material had to be removed, a note will indicate the deletion.

Oversize materials (e.g., maps, drawings, charts) are reproduced by sectioning the original, beginning at the upper left-hand corner and continuing from left to right in equal sections with small overlaps. Each original is also photographed in one exposure and is included in reduced form at the back of the book.

Photographs included in the original manuscript have been reproduced xerographically in this copy. Higher quality 6" x 9" black and white photographic prints are available for any photographs or illustrations appearing in this copy for an additional charge. Contact UMI directly to order.

UMI®

Bell & Howell Information and Learning
300 North Zeeb Road, Ann Arbor, MI 48106-1346 USA
800-521-0600

RECOGNITION OF DATES HANDWRITTEN ON CHEQUES

Rong Fan

A thesis
in
The Department
of
Computer Science

Presented in Partial Fulfillment of the Requirements
For the Degree of Master of Computer Science
Concordia University
Montréal, Québec, Canada

April 1998

Abstract

Recognition of Dates Handwritten on Cheques

Rong Fan

In this thesis, an automatic system for processing date information handwritten on bank cheques is described. This system is composed of date image segmentation, handwritten digit recognition, and cursive word recognition. The proposed method does not impose any restriction or require any *a priori* information on the date written, and is able to handle both English and French cheques. With appropriate modification, this method can be extended to recognize date in other application environments.

Much effort has been dedicated to the design and implementation of date image segmentation. Two groups of features, *shape* features and *spatial* features, are developed and utilized to locate possible cutting positions on *date_zone* images. In addition, a set of heuristic rules are proposed to complete the segmentation of this image, identify the three fields (*Day*, *Month* and *Year*), and also determine the writing style of *Month* in order to apply the appropriate recognizer (i.e. digit or cursive word) to process the *Month* field.

Both segmentation and date recognition systems were tested on two different databases, i.e. (a) the CENPARMI database, and (b) data collected from a utility company.

The results obtained prove the efficiency and feasibility of the proposed date recognition system. Areas which require further development are also specified.

An image tagging tool has been developed for this research. In addition to its user-friendly environment and simple operation, it also provides powerful functionality in recording contextual information. This makes it possible and convenient to extract data such as words, digits, characters, punctuations, etc.. A lot of useful statistics can also be gathered from the tagged information.

Acknowledgments

I would like to express my sincere gratitude to my supervisors Dr. Ching Y. Suen and Dr. Louisa Lam for their guidance which led me from the beginning to the end of my research work. It could have been impossible to finish this work without their support and constant encouragement.

Special thanks go to Nick Strathy, Didier Guillevic and Dr. Ke Liu who have not only provided me with great help to complete my work but also shared with me their knowledge and invaluable research experience.

I would like to express my appreciation to Christine Nadal who collected and maintained the cheque database, and to Guiling Guo and Myriam Côté who processed thousands of images for me. Also, I would like to thank William Wong, Michael Assels and Michael Yu for their effort of maintaining a high-quality computing environment.

I am grateful to Dr. Z.C. Li who made all these things possible for me: starting my adventure in a completely different world and pursuing my studies in the field of Computer Science.

My great appreciation goes to a list of people who kindly helped me when I first came to Canada: Irene Mazis, Y.S. Huang and his wife, Dr. Y.Y. Tang, Yuan Chen, Daini Xie, Jianxing Yuan and Zhisong Chen; and to those who are around to share my emotions: Jie Ding, Jie Zhou, Hao Chen, Qizhi Xu, Xiangyun Ye, Flip Lunan, Zetta Lunan and Eddie Webb.

All my life, I am indebted to my parents who brought me to this world, taught me integrity, and raised me with endless love and expectations. To Jinghui, the man in my life, I am so grateful for his effort of always being there for me. T my brother who shares a similar taste in almost everything, I am thankful to have had such a buddy for more than 20 years.

Those whom I love and those who love me, thank you so much.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

This thesis aims at developing a system for automatic recognition of handwritten dates on personal cheques. This system analyzes and interprets the *date_zone* images extracted from cheques, and outputs the date written, including *Day*, *Month* and *Year*. The method proposed in this research work can be applied to the recognition of any handwritten dates with similar format.

1.1 Research Motivation

Today in the age of information technology, with the rapid growth of electronic communication and the need for fast and worldwide availability of information, the trend seems to lead us to a paperless world. However, it is too early to say that paper-based documents will become relics from a period behind us. On the contrary, as a traditional information medium, paper has had and continues to have huge advantages over alternative media. Paper is standardized. There are no interface problems between writer and reader. Paper is readily available. It does not require special hardware or software to access old documents. Paper is highly portable and its transportation is well established, though it is considerably slower than electronic document transfer. Additional advantages exist for handwritten text. Writing a note, address or filling out a form by hand needs no special preconditions beyond the human ability to write and the need for paper and a pen-like writing tool [Bar96]. Therefore, it is fair to conclude that paper-based documents will continue to play an important role in people's daily lifes.

On the other hand, we cannot neglect the advantages and superiority of computers in storing, exchanging, searching and processing text, data and information, as well as the relatively inexpensive CPU power. Growing demand has come from industries and operational companies to utilize the power of the computer to bridge the gap between the world of paper with conventional writing and the world of computers and electronic processing [Bar96].

The objective of OCR (Optical Character Recognition) is to replace human beings by electronic machines in processing written (either handwritten or machine printed) materials. In this way, it is hoped that the efficiency and correctness of such procedures can be improved, and human beings can also be relieved from the tedious manual work. This is why automatic processing of those widely and frequently used written materials, such as cheques, mailpieces, forms, etc., has become one of the hot research topics in this area.

There have been some research efforts and even products on the market for applications such as postal address processing [Sri92] [Mil96] [Sri96]. However, the complete processing of cheques, including amount and especially date, had not been seen when this project was being developed [FLS96]. In fact, automatic cheque processing is of great importance, because the favorite way of bill payment in North America is still by cheque, despite the fact that nowadays people have various options for paying their bills. According to a recent survey, 83% of Americans still prefer to make their non-grocery type of payments via cheques [Cas94], because it is considered to be secure, confidential and the most convenient. The volume of cheques in the US is growing at a rate of nearly 2 billion cheques every year and it is now reaching 69 billion cheques annually. Neither ATMs, nor debit cards, nor home banking have yet persuaded the populace to write fewer cheques [HAS+96].

On the other hand, the possibility to enhance the efficiency and quality of cheque processing is also of great concern to the utility companies. Upon receiving cheques, the important information, such as amount authorized, must be extracted for a timely deposit. However, in many environments (e.g. Canada), the written date must be observed in cashing the cheques. If there are large quantities involved, any delay in banking activity would entail significant financial losses. Therefore, date information plays an important role in optimizing the utility companies' financial activity [FLS96].

Due to the high degree of variability and uncertainty present in the dates handwritten on bank cheques, developing an efficient date recognition system has been a very challenging problem in OCR applications. Perhaps for this reason, there has been no published work on this topic until very recently, when work on the date fields of machine-printed cheques is reported [HAS+96].

1.2 The Challenge

Human handwriting is affected by a variety of factors, for instance, the person's characteristics, emotion, physical condition, education, the writing material used (e.g., pen, ink and paper), etc.. Any slight change in those factors may result in a significant difference in the handwriting. Since such changes are always unpredictable and unavoidable, it is very difficult to analyze and interpret human handwriting using mathematical methods.

The above mentioned general difficulty exists in this research work. Apart from that, the challenge also comes from the particular characteristics of handwritten dates on standard bank cheques. Unlike various forms given by different organizations where the date must be written with only digits, and *Day*, *Month* and *Year* must be filled into the designated positions, there is no such restrictions when *date_zone* is filled on cheques. People are free to write the date in any format and language of their choice (in Québec, Canada, a significant proportion of cheques are written in French), and thus the same date can be written in many styles.

Compared with the vision of human beings, the vision of the computer is extremely fuzzy and vague. When a binary *date_zone* image is sent for automatic processing, what the computer can view is just a blurred picture with black and white pixels distributed irregularly. It is very difficult for the computer to locate where *Day*, *Month* and *Year* are exactly written. It is also problematic for the computer to apply word or digit recognizer properly, as *Month* can be written in either pure digits or pure words.

Given such a complex problem, any *a priori* knowledge that can be introduced to the system under development would be a great asset. For example, if the writing style or format of the *date_zone* is known, then less computing effort is required to choose the proper recognizer (different recognizers are required for digits and words

respectively) so as to process the image or subimage. If some constraints can be placed on the date being processed, e.g., restricting the date within a known period in certain months and years, the result can be selected among a much smaller set of possible candidates, with that the parsing procedure can be simplified and the performance can be improved.

However, such information is not available for this research work. Therefore, the system must be able to detect the writing style of the date, send the *date_zone* image or subimage to the correct recognizer, compare recognition results and select candidates, validate the results, and generate an output [FLS96].

1.3 Survey

Today, automatic document processing is within reach of industrial applications because of the availability of relatively inexpensive CPU power, and the resulting possibility of implementing time consuming image processing techniques and training algorithms. Automatic bank cheque recognition for instance is such a task, benefitting from a small vocabulary and a strong syntax [KBPS96].

The volume of cheques in the US has increased steadily by two to three percent yearly for the past five years. Advances in computer technology and pattern recognition, coupled with continued customer preference for payment by cheque, make imaging-based systems a viable solution not only to save labor but also to create new services [HAS+96].

1.3.1 Corroboration-based cheque reader

Houle et al [HAS+96] designed a multi-layered corroboration-based cheque reading system which will be used in transaction-based processing. Here, *transaction* refers to the content of a payment envelope (i.e. cheque(s), stub and invoice). Cheques processed by this system (including business cheques which vary considerably in their design) are typically those used for remittance transactions, such as monthly utility bills and magazine subscriptions. Five target fields of remittance processing, i.e. signature presence, payee name, legal and courtesy amounts, and date, are to be identified and recognized. In order to control the error rate, corroborations at the

character/digit level, at the word image level, between fields such as courtesy and legal amounts, and with external sources (such as spoken amounts), are implemented.

To our special interest, processing of the *date* field is also discussed in this paper, though it is considered as the most difficult target. The matching 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 neighbouring 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 typewritten *date* fields were tested. The complete recognition performance with the automated field segmentation is estimated at 44% to 49% by using four commercial recognition devices [HAS+96].

1.3.2 A2iA cheque reading system

This section describes the A2iA [KBPS96] system developed in France. It is a cheque reading system which recognizes both legal and courtesy amount on bank cheques. The current version is designed for French, omni-bank, omni-scriptor, handwritten bank cheques. The recognition process is divided into 5 steps: *extraction* of the image parts containing the amount, *preprocessing* of the extracted amount images, *segmentation* of the courtesy amount into character candidates, and segmentation of the legal amount into character candidates and word candidates, *recognition* of characters, words and amounts, and *decision*. The system obtains between 50% and 65% recognition rates for less than 0.1% errors, depending on the importance of the centimes part on the cheques to be recognized [KBPS96]. Most of the problems of the current system are caused by the fact that it processes binary images. While the courtesy amount recognition works satisfactorily with binary images (and it could be improved by using gray scale images), the legal amount recognition suffers heavily from the problem of segmenting the handwriting from the background of the cheques.

1.3.3 Re-engineering bank cheque processing

According to Dimauro et al [DGI+96], for complex problems such as bank cheque processing, algorithm combination techniques should be applied in order to improve system performance. At present, there are algorithms available to handle the large number of bank cheque layouts, to remove image background, to identify bar code and account number, and to recognize single digit, touching digit, single character, legal amount as well as signature. Therefore, it is considered as an engineering problem to manage both in size and time, all the different components in cheque processing. A CASE (Computer Aided Software Engineering) tool, "Khoros" environment development tool, is used to design a prototype for Italian bank cheque processing. The same re-engineering action can be extended to process tax return forms, credit card slips, and different types of application forms, etc..

1.4 Proposed Research

Since no constraint is placed on the *date_zone* images processed by the system under development, there exists a large number of possible outputs [SLG+96]. In addition, invalid dates such as "February 30, 1997" must be handled by the system as well. Therefore, there is immense variety in the images to be processed, and it will be very inefficient in computation if the system is developed to process the entire *date_zone* image at the same time.

A method involving date image segmentation, digit recognition and cursive word recognition is proposed in this thesis. In this way, the problem is reduced to dividing the entire *date_zone* into *Day*, *Month* and *Year*, and then processing cursive or hand-printed words and numerals separately. The cost on computing resources can also be reduced.

Chapter 2

System Overview

2.1 Variations of Handwritten Dates

In North America, all bank cheques have a similar format, as shown in Figure 1. The *date_zone* is always positioned at the upper right corner on each cheque [LSN96]. Machine-printed digits '1' and '9' appear on the right of the *date_zone*, and the *date_zone* is thus separated into two parts. The part on the right, which is headed by printed "19", is intended for people to fill in the last two digits of *Year*. The part on the left is intended for people to fill in both *Day* and *Month*. This part is completely blank, which implies there is no pre-defined position for *Day* or *Month*.

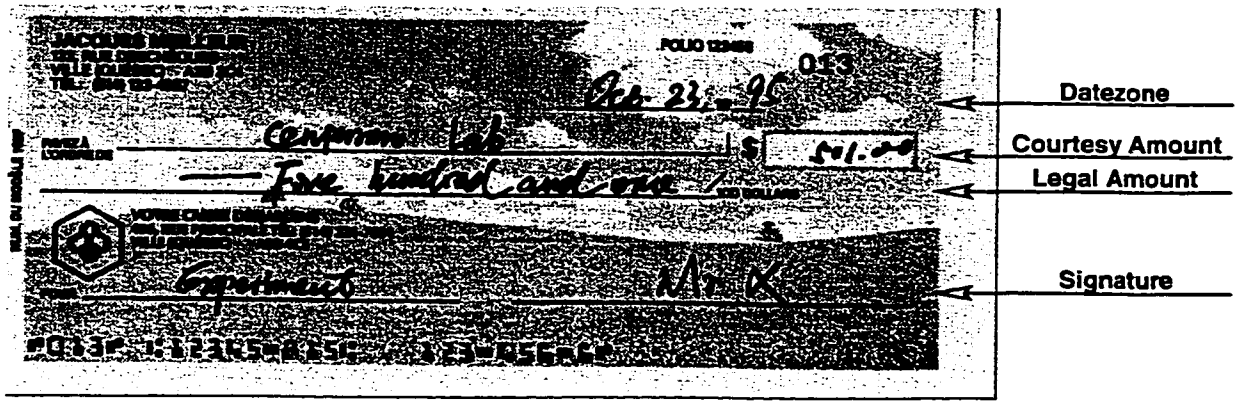


Figure 1: Example of standard bank cheques

In addition, there is no restriction on writing style either, and the appearance of *Day* and *Month* is completely dependent on each individual user's writing habit.

Month can be written in either digits or cursive script. Punctuations (period ('.'), comma (','), slash ('/') and hyphen ('-')) can be used to identify the end of one field. Even languages other than English (e.g. French) are allowed when writing *Month* in cursive script. Hence, great flexibility exists for a writer when filling the *Day* and *Month* fields on a bank cheque. This is why the same date can be represented by a large variety of writing styles (refer to Figure 2).

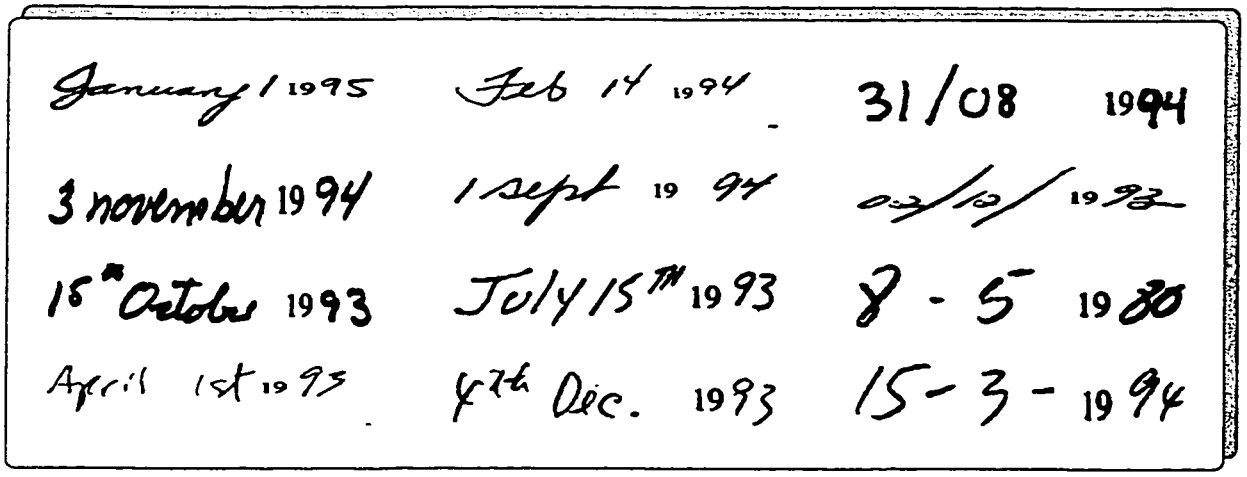


Figure 2: Examples of *date_zone* images

As shown in Figure 2, the content of *date_zone* can be either pure digits where *Month* is written in digits, or a combination of digits and cursive scripts where *Month* is written in a word. *Month* can be placed either before or after *Day*. In general, the writing styles of *date_zone* extracted from bank cheques can be expressed in the following 4 patterns, regardless of any punctuations or suffices:

dd mm yy
mm dd yy
dd MM yy
MM dd yy

where *dd* designates *Day* written in digits, *mm* designates *Month* written in digits, *MM* designates *Month* written in cursive script, and *yy* designates *Year* written in digits.

It should be noted that when *Month* is written in cursive script, a number of variations can result. In Québec, the majority are French speaking people, hence a large

proportion of cheques contain the French version of *Month* words. Apart from the language, people are free to use either the complete *Month* word or its abbreviation. *Month* words can be handprinted using pure uppercase letters, or written in lowercase letters (mostly seen on French cheques), or written in both uppercase and lowercase letters.

In addition, according to Figure 2, four types of punctuations, i.e. slash ('/'), hyphen ('-'), period ('.') and comma (','), are written in some *date_zones*. Suffixes, such as *th*, *st*, *nd* and *rd*, are also written after *Day* by some users [SLG+96].

2.2 Writing Style Analysis

Unlike the human recognition process which is very flexible and adjustable to many changes, the machine recognition process is significantly impacted when any of the above mentioned variations is present in the *date_zone* image. It will be very difficult and inefficient to develop an individual mechanism for each writing style. The best strategy in this case is to analyze all these variations in detail, and see whether they can be categorized.

Investigation was done on the 4564 *date_zone* images collected by the CENPARMI research group (further detail about this set of data is provided in Chapter 4). It showed that there did exist some characteristics which are common to all *date_zone* writing variations.

In North America, '1' and '9' are printed as isolated digits on all standard bank cheques indicating the current century (the 20th century). This restricts the position of *Year* and also simplifies its writing. For most *date_zone* images we have extracted, this rule is followed with few exceptions. Therefore, it is reasonable to conclude that *Year* is a relatively stable part of handwritten dates, since the last two digits of *Year* are written after the printed "19" [SLG+96].

When dates are written within an unconstrained environment, such as on bank cheques, most people are rather cautious that they always try to find some ways to differentiate the fields, especially *Day* and *Month*. This may not seem very obvious or may not even be realized by the writers themselves. In some cases, *Day* and *Month* are separated by a gap (obvious or not). In other cases, these two fields are separated by punctuations such as slash ('/'), hyphen ('-'), period ('.') and comma (','). To

represent a certain date, for example "February 26, 1997", all the variations can be categorized as the follows [FLS95]:

Example 1: February(,) $26^{(th)}$(,) **1997**
Example 2: Feb(.) $26^{(th)}$(,) **1997**
Example 3: $26^{(th)}$ February **1997**
Example 4: $26^{(th)}$ Feb(.) **1997**
Example 5: (0)2/26(/) **1997**
Example 6: (0)2-26(-) **1997**
Example 7: 26/(0)2(/) **1997**
Example 8: 26-(0)2(-) **1997**

In the above list, items shown within parentheses designate those entities which may or may not be present. "**19**" in bold are the printed digits which appear on all bank cheques representing the current century. Suffixes such as 'st', 'nd', 'rd', or 'th' can be written either as superscripts or at the same horizontal position as the rest of the *date_zone* information.

On handwritten date images, there is no uniform spacing present between *Day* and *Month*. To further differentiate these two fields, it is observed that punctuations are written by cheque writers, intentionally or not, and different punctuations are used in different situations. Studies on *CEMPARMI cheque* database (refer to Chapter 4 for detail) showed that period '.' and comma ',' are written mostly when *Month* is represented with cursive script, while slash '/' and hyphen '-' are mostly written when *Month* is represented with digits.

Among 4564 *CENPARMI cheques*, 84.09% have *Month* written in cursive script, and 15.91% have *Month* written in numerals. Among the 3837 images where *Month* is in word, punctuations appear on 765 images which is about 19.94%. Detailed statistics about punctuation usage in this case is shown in Table 1. Among the 726 *date_zone* images where *Month* appears in digits, only 17 of them do not have any punctuation written. Table 2 illustrates the punctuation usage when all three fields of a *date_zone* are written in numerals.

"Exceptions" in Table 1 refers to those *date_zone* images where *Month* is written in cursive script, but punctuations used are slash '/' or hyphen '-'. "Exceptions" in

Table 1: Use of punctuations when *Month* is in word

	No. of images	Percentage (%)
Total no.	765	100
Period '.' only	537	70.20
comma ',' only	154	20.13
Both *period* & *comma*	38	4.97
Exceptions	36	4.70

Table 2: Use of punctuations when *Month* is in digits

	No. of images	Percentage (%)
Total no.	709	100
Slash '/' only	423	59.67
Hyphen '-' only	234	33.00
Both *slash* & *hyphen*	2	0.28
Exceptions	50	7.05

Table 2 refers to those where *Month* is written in digits, but punctuations used are period '.' or comma ','.

In the *CENPARMI cheque* database, there are altogether 3089 images where no punctuation is used at all. Further conclusion can be made that *Month* is most probably written in word when there is no punctuation in *date_zone*, since only 0.55% of the 3089 images have *Month* written in numerals.

As discussed previously, if punctuation or suffix is not considered, *date_zone* writing styles can be classified into 4 categories. Pertinent statistics is also gathered on *CENPARMI cheque* database. As shown in Table 3, when *Month* is written in cursive script, about half of the people would like to write *Month* word prior to *Day*, and the other half prefer to place these two fields in the opposite way. However, when *Month* is written in numerals, more people tend to put it after *Day*. In addition to the above information, it should also be mentioned here that there exist 237 *date_zone* images where both *Month* and *Day* are written in numerals, but the sequence cannot be determined since both fields are less than or equal to 12.

Table 3: Statistics about *date_zone* writing style patterns

Writing style pattern	No. of images
MM dd yy	1936
dd MM yy	1901
dd mm yy	442
mm dd yy	46

2.3 System Architecture

Even if punctuations and language are not considered, there exist at least 8 writing styles to represent a date (refer to the examples of "February 26, 1997"). If the system is required to process the date within one year, then at least 2920 (i.e. 8×365) different formats would be involved. The system under development should be able to process any date without limitation. When an entire *date_zone* is processed as one image, then a huge database would be needed.

It is very difficult and almost impossible to collect exhaustively handwriting of all the dates in different writing styles. It is also very expensive to store the huge amount of data, and very inefficient in computation when a *date_zone* image is processed.

A system including *date_zone* image segmentation, digit recognition, word recognition and parser is proposed in the research work. Segmentation is used to separate the entire image into three subimages, create hypotheses to assign each subimage as *Day*, *Month* or *Year*, and detect whether *Month* is written in digits or cursive script. After segmentation, the subimages of *Day* and *Year* are sent to the digit recognizer. The *Month* subimage is sent to the word recognizer if it is considered to be written in cursive script, and to the digit recognizer otherwise. The recognition result is sent to a parser which is used to reject invalid results, and interpret an acceptable result when both *Month* and *Day* are written in digits. The system architecture is shown in Figure 3.

The method proposed here has the following advantages:

- There is no need to store a large number of entire *date_zone* images when processing, so the cost is reduced significantly.

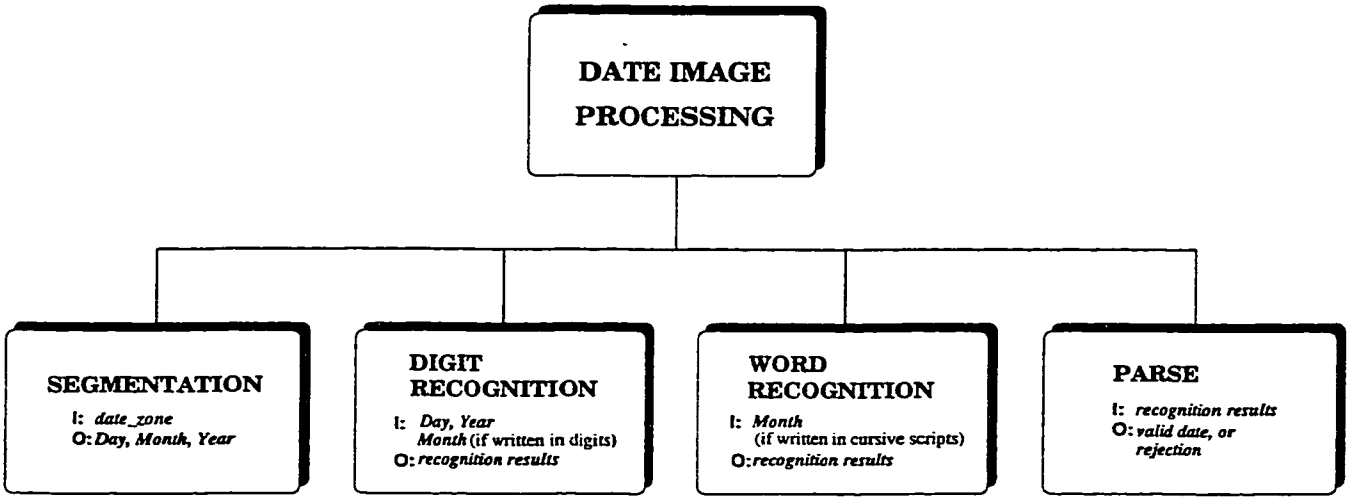


Figure 3: Processing of the *date_zone*

- The problem is reduced to processing digits and words separately. For digit recognition, the recognizer for processing the numeric amounts on cheques can be used without training. For cursive word recognition, the recognizer for processing the legal amount on cheques can be used after training on the *Month* words.

- The processing is much more efficient, since every subimage is compared with a limited set of candidates. For example, when *Month* is considered to be written in cursive script, its subimage is compared only against English and French *Month* words in both complete and abbreviated formats, which consist of only 42 candidates:

  1. January janvier Jan jan
  2. February fevrier Feb fev
  3. March mars Mar mar
  4. April avril Apr avr
  5. May mai
  6. June juin
  7. July juillet juil
  8. August aout Aug

9. September septembre Sept sept
10. October octobre Oct oct
11. November novembre Nov nov
12. December decembre Dec dec

Chapter 3

Date Image Segmentation

3.1 Segmentation Strategy

The purpose of *date_zone* image segmentation is to divide the entire image into three subimages representing *Day*, *Month* and *Year* respectively, and also generate hypothesis on how *Month* is written so that it can be processed using the proper recognizer [SLG+96]. This might sound trivial to a human being. However, our effort was to make computers, which are "good" at performing routine processes but lack adaptability, undertake exactly the same task. This is not easy bacause there is not much regulated information on a date image extracted from bank cheques. For example, there is no predefined position for each field (except for *Year* which is written after printed "19" in most cases), and no uniform or even obvious spacing among the three fields. Therefore, special characteristics of the *date_zone* image must be observed in order to implement the segmentation process.

Since machine-printed digits '1' and '9' are present on all standard bank cheques, we can assume that the information written on their right would be the last two digits of *Year*. The *Year* subimage can be obtained once these two printed digits are located. On the other hand, it is also "safer" to start segmentation by searching for "19". Compared with the rest of the *date_zone* which contains only handwritten information, this part is relatively stable. These two digits are printed clearly in regular fonts, and they are isolated from each other. Even though in terms of font and size, "19" are printed differently on different bank cheques, there are still much

less variations compared with human's handwriting. Hence, it is not only simpler and more efficient to detect printed digits "19", but it also helps to achieve a better segmentation performance.

Based on the above assumption, the information handwritten on the left of "19" should contain only *Day* and *Month*. Hence, upon locating "19", another subimage can also be obtained by segmenting the *date_zone* according to the position of printed digit '1'. This part is called *Day&Month* subimage in this work. It is the focus of segmentation process up to this stage.

As observed from *CENPARMI cheque* database, four types of punctuations, i.e. slash ('/'), hyphen ('-'), period ('.') and comma (','), are frequently seen in *date_zone* images. These punctuations are rarely located in the *Year* subimage. In fact, they are almost always written together with *Day* or *Month*. The position of a punctuation marks the end of *Day* or *Month* field. It may also identify a division between *Day* and *Month*. Based on the statistics presented in Tables 1 and 2, the type of punctuation implies how *Month* is written. The presence of slash ('/') or hyphen ('-') would imply that *Month* is written in digits, while the presence of period ('.') or comma (',') implies words. Therefore, once the *Day&Month* subimage is scanned for these four types of punctuations, a cutting position can be located. In this way, the *Day* and *Month* subimages are obtained, and the proper recognizer (either digit or word) to be used to process *Month* subimage is also determined.

Compared with the number of *date_zone* images where one or two punctuations are written, the number with no punctuation is even higher. Investigation shows in this case, *Month* is almost always written in a word. This is probably due to the fact that when writing a cheque, the majority of the population still follow the standard expression of date, i.e. *Month* written in word, and *Day* and *Year* in numerals. As digits "19" are already printed on each standard bank cheque, there is no need to write any punctuation for the purpose of distinguishing between *Day&Month* and *Year*. However, most people tend to distinguish specifically between *Day* and *Month* and a relatively obvious gap is left between these two fields, either consciously or unconsciously. This makes it possible for the segmentation system to process such *Day&Month* images by locating an interword gap. The position of the gap is considered as where *Day* and *Month* subimages should be separated. Since *Month* is

assumed to be written in word which contains at least 3 letters, but *Day* is composed of maximum 2 digits, the sizes of the two subimages are compared in order to determine which represents *Day* and which represents *Month*.

Figures 4 and 5 give a high-level view of *date_zone* image segmentation.

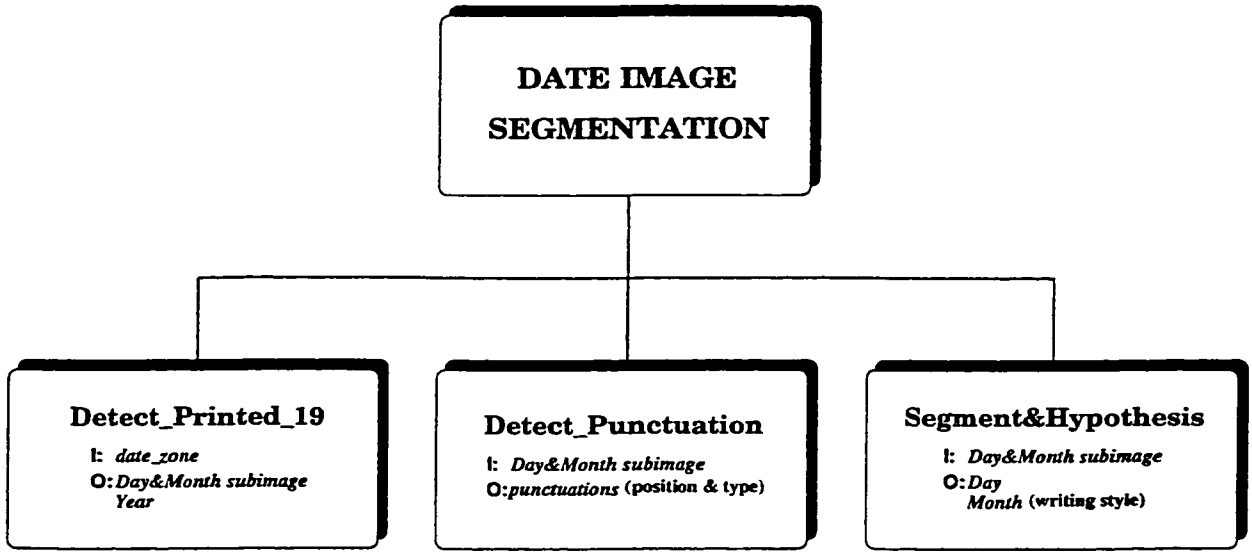


Figure 4: Segmentation of the *date_zone*

3.2 Image Description

The input of the automatic date processing system consists of binary *date_zone* images, each of which is composed of a set of "blob"s. In this work, a "blob" is used to denote a group of connected black pixels. Each "blob" might contain exactly one *date_zone* field (i.e. *Day* or *Month* or *Year*) or a part of it. It could also consist of more than one field, or even multiple segments from several fields. From the computer's vision, each date image is nothing but a number of "blob"s. In order to enable the computer to process the *date_zone*, mathematical information of each individual "blob" appearing on the binary image must be obtained. The OCR1 library, available on CENPARMI network, is utilized for this purpose. According to the implementation of this library, each "blob" is represented by a connected component associated with a set of important information [Str93]:

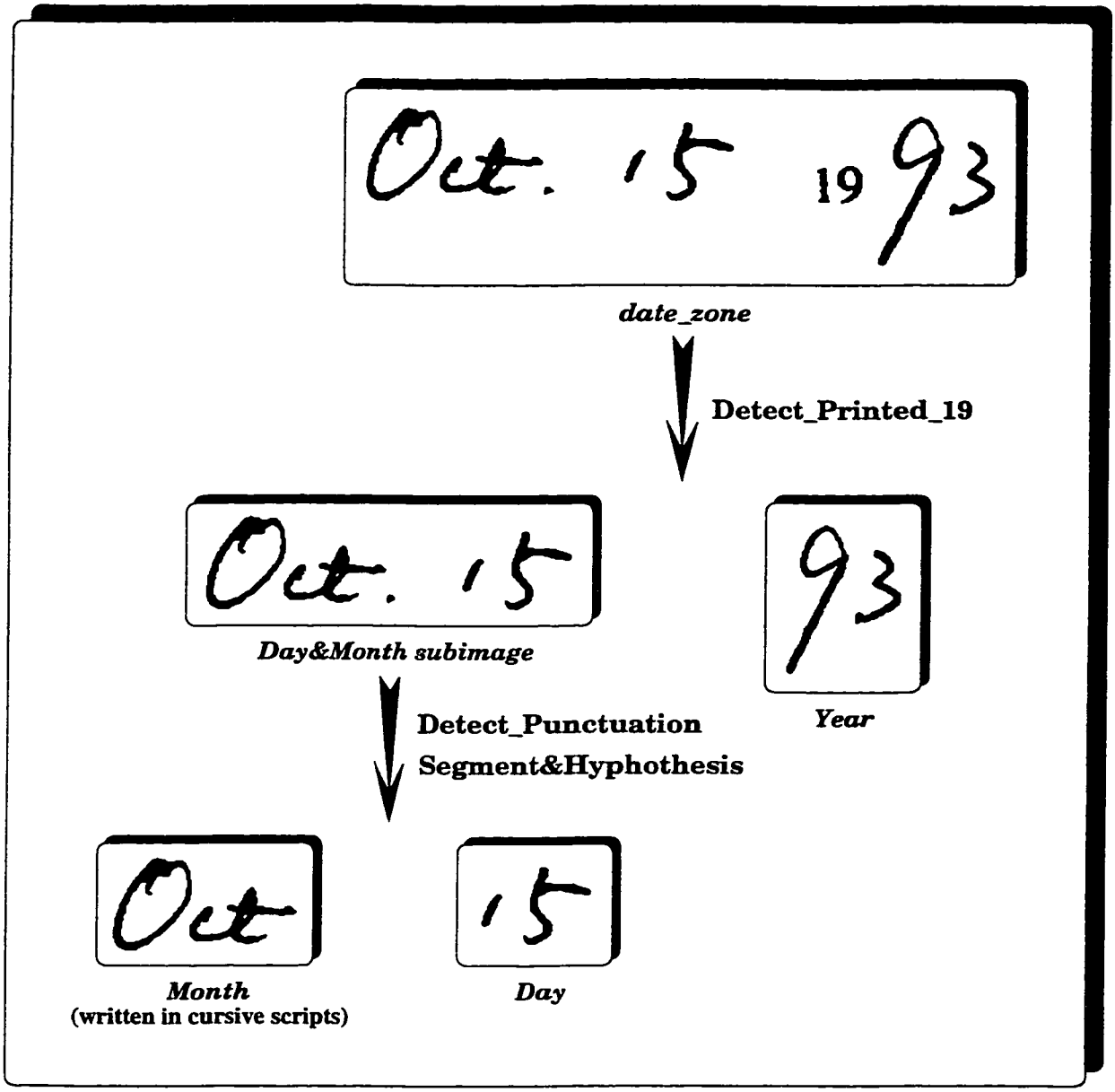


Figure 5: Illustration of the *date_zone* segmentation

- *Contours* of all the connected components stored in a double linklist, in the order of their horizontal location from left to right. This provides an overall description of the *date_zone* image, including the number of connected components existing in the image, location, etc.
- *Chain code* of each connected component, and the length of the chain. This helps to describe the shape and size of each connected component.
- *Bounding box* of each connected component gives its relative location with respect to the entire image and to the other components.
- Information about all the *inner contours* of each connected component, which is very useful in punctuation detection.
- *Runlength* information about the *date_zone* image, which can be used to describe the distribution of strokes on the entire image.

3.3 Feature Description

Based on the numeric information obtained using OCR1 library, a set of features are developed in order to analyze each connected component in the *date_zone* image for detection of the four types of punctuations as well as printed digits '1' and '9'. In general, two categories of features, shape features and spatial features [CHS94], can be considered. In this research work, shape features deal with the geometric aspect of each connected component. As described by the name, they are used to distinguish a punctuation, or printed digit '1' or '9' from the other connected components according to its own appearance and measurement. Spatial features actually deal with the context aspect of each connected component, which gives important information especially when the objective is to process a text line. They are used to describe the location of each connected component with respect to the entire *date_zone* image as well as its neighbouring components.

Each feature algorithm returns one of the two types of values, Boolean value and confidence value. Boolean value is either 1 or 0, where 1 represents the presence of a certain feature, while 0 represents its absence. However, simply returning Boolean value is in fact making a "hard" decision on feature evaluation. Experiments showed

that this could lead to errors when analyzing a connected component for punctuation or printed '1' or '9'. This is why confidence values are introduced at some feature algorithms. Confidence value here actually refers to a fuzzy value between 0.0 and 1.0, where 0.0 indicates that a certain feature is most likely to be absent, while an increasing value indicates an increasing likelihood that the feature is present.

3.3.1 Shape features

The shape features developed are *high_density*, *narrow*, *flat*, *slope*, *small*, *simple_curve* and *no_innerloop*.

high_density

The *high_density* feature compares the size of each connected component (number of foreground pixels in the component) with the size of its bounding box (height times width). This feature is introduced due to the fact that normally, punctuations such as hyphen and period occupy most of their bounding boxes.

The percentage of the component size with respect to its bounding box size is compared first:

$$size2boundingbox = (comp\_size)/(boundingbox\_size),$$

where *comp_size* is the size of the component.

Then this value is compared with a threshold, $density_H$ or $density_P$ depending on whether hyphen or period is to be detected. The return value of this feature algorithm is 1.0 if

$$size2boundingbox \geq density_H \text{ or } size2boundingbox \geq density_P$$

Otherwise, return

$$density_H/size2boundingbox$$

or

$$density_P/size2boundingbox$$

$density_H$ and $density_P$ are obtained by calculating the average *size2boundingbox* values of hyphen and period respectively in the training set.

narrow

Punctuations slash and comma, and digit '1' have a common characteristic, i.e. the ratio of component horizontal run to its height is relatively low. The *narrow* feature is introduced for this reason. However, as in some *date_zone* images, slash is written at a slant (Figure 6), so that the vertical height does not reflect the actual length of the component. Therefore, the distance between the two vertical end points (Figure 6), *vert_endpt_dist*, is used instead.

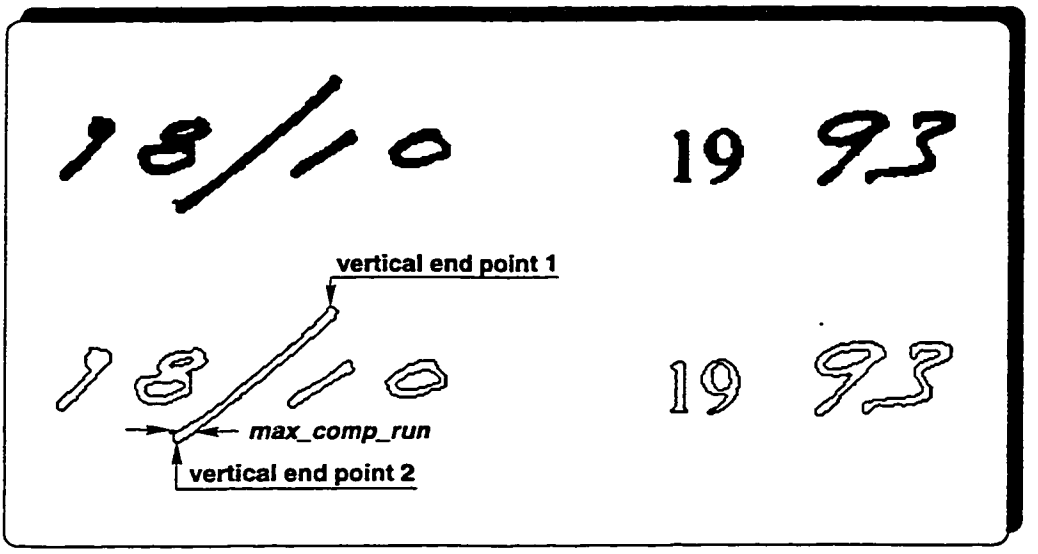


Figure 6: Illustration of the *narrow* feature

First, the ratio of the maximum horizontal run of each connected component to *vert_endpt_dist* is computed:

$$Hrun2Vdist = (max\_comp\_run)/(vert\_endpt\_dist)$$

Then, $Hrun2Vdist$ is compared with a threshold in order to obtain the confidence value of the "narrowness" of the component. Different threshold values are used in detecting slash, comma and printed '1' respectively, i.e. $narrow_S$, $narrow_C$ and $narrow_1$. The return value is assigned to 1.0, if $Hrun2Vdist$ is less than the threshold, and otherwise to

$$narrow_S/Hrun2Vdist$$

or

$$narrow_C / Hrun2Vdist$$

or

$$narrow_1 / Hrun2Vdist$$

indicating how likely the component can be considered to be "narrow". Threshold values are obtained by calculating the average $Hrun2Vdist$ values of slash, comma and printed '1' respectively in the training set.

flat

Contrary to the *narrow* feature, the *flat* feature is introduced considering the fact that punctuation hyphen has a relatively low ratio of vertical run to component width. For the same reason, the distance between two horizontal end points, *horz_endpt_dist*, is used instead of width so that a more precise description can be obtained especially when the hyphen is not written horizontally (Figure 7).

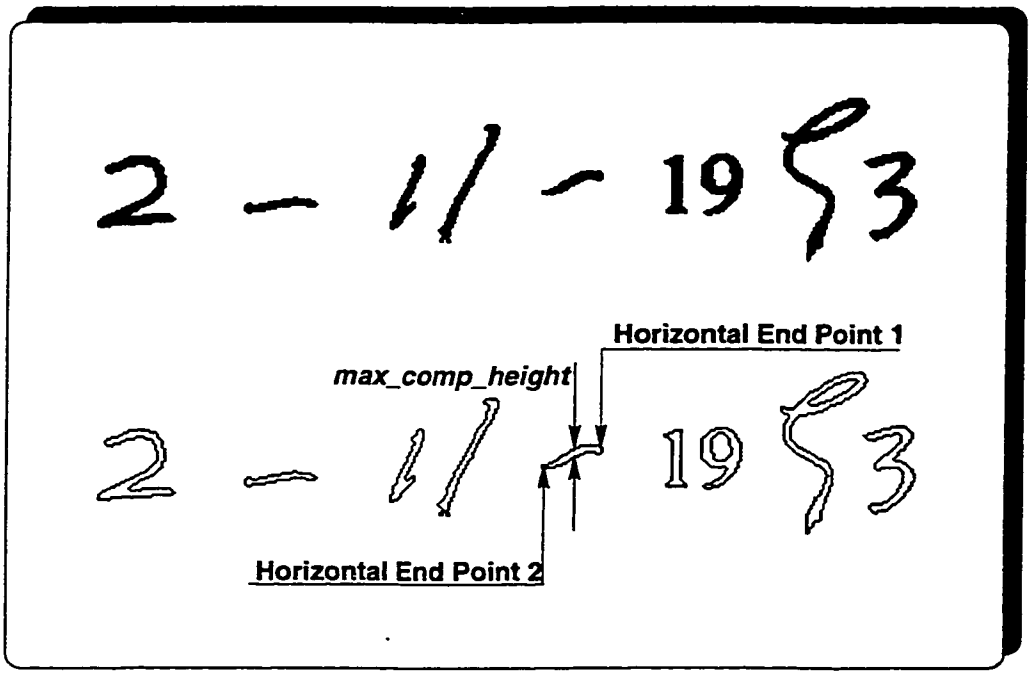


Figure 7: Illustration of the *flat* feature

It first computes the ratio of the maximum height of the connected component to *horz_endpt_dist*:

$$height2Hdist = (max\_comp\_height)/(horz\_endpt\_dist)$$

The $height2Hdist$ value is also compared with a threshold value, $flat_H$, in order to obtain the confidence values of the "flatness" of the component. $flat_H$ is the average $ratio_{v2h}$ value of all hyphens in the training set. The return value is assigned to 1.0, if $ratio$ is less than $flat_H$, and otherwise to

$$flat_H/height2Hdist$$

indicating how likely the component can be considered to be "flat".

slope

There are two straight vertical lines on the contour of printed digit '1', and slash is usually written relatively straight in most of the *date_zone* images. If we consider the curvature (only the absolute values) of such connected components, the slope over the entire contour presents approximately the same value. The *slope* feature is introduced for this purpose. It first computes the slope on each contour pixel, and then checks whether most of them fall within a certain interval centered at the component slope (*comp_slope*).

In this work, the component slope is obtained by computing the slope between the two vertical end points of the connected components (Figure 8). The slope of each contour pixel is defined as the slope between two such pixels, one of which is 5 steps before the current pixel on the contour chain code, while the other one is 5 steps after. At the same time, the number of pixels whose slope falls into the following range

$$(comp\_slope - \delta, comp\_slope + \delta)$$

is also counted. This number is then compared with the total number of contour pixels to obtain a ratio (*comp_slope_percent*). The feature algorithm returns 1.0 if *comp_slope_percent* is greater than a threshold ($slope_S$ or $slope_1$), or returns

$$comp\_slope\_percent/slope_S$$

or

$$comp\_slope\_percent/slope_1$$

otherwise. The thresholds $slope_S$ and $slope_1$ are the average *distribution* values of slash and printed digit '1' respectively in the training set.

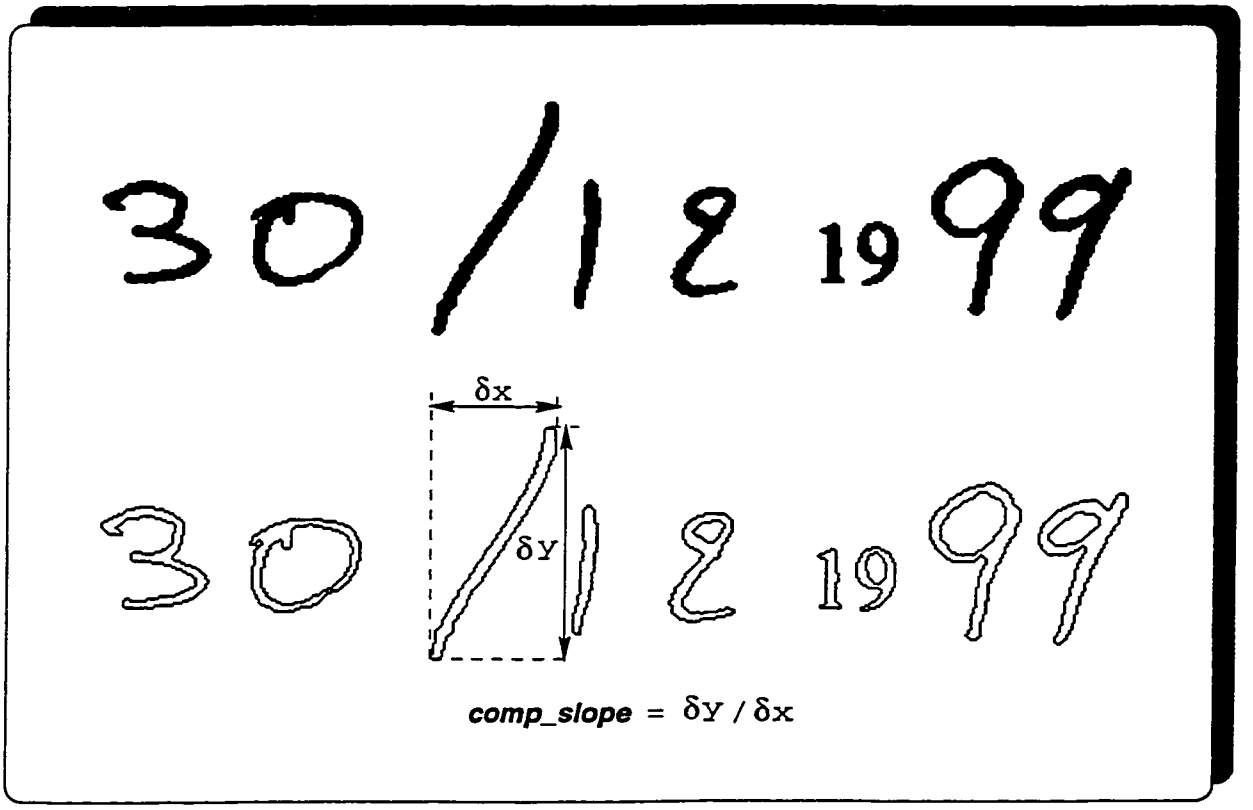


Figure 8: Illustration of the *slope* feature

small

On a properly binarized *date_zone* image (without too many broken strokes), punctuations such as period and comma are relatively small compared to the size of the other connected components. The feature *small* is therefore introduced in punctuation detection.

The average size of all the connected components on the *date_zone* image (designated as avg_comp_size) is computed first. The feature algorithm then returns different values based on a comparison between the component size ($comp\_size$) and avg_comp_size. 0.0 is returned if

$$comp\_size \geq avg\_comp\_size$$

while 1.0 is returned if $comp\_size$ is as small or smaller than a threshold, avg_size_P or avg_size_C depending on which punctuation is to be detected. Otherwise a value between 0.0 and 1.0 is returned depending on whether $comp\_size$ is closer to avg_comp_size or the threshold, respectively, i.e.

$$(comp\_size - avg\_comp\_size)/(avg\_size_P - avg\_comp\_size)$$

or

$$(comp\_size - avg\_comp\_size)/(avg\_size_C - avg\_comp\_size)$$

avg_size_P and avg_size_C are chosen respectively as the average period size and average comma size encountered in the training set.

simple_curve

All the four types of punctuations and the printed digit '1' are composed of simple strokes, i.e. there is neither inner loop nor non-uniform curvature present in the connected component. The *simple_curve* feature is introduced for this reason. Hence confusions between a narrow 'V' and a slash can be avoided.

For each connected component, the number of horizontal runs on every row and the number of vertical runs on every column are counted, so as to obtain the number of foreground segments. Ideally, there should be only one foreground segment on each row and column if the component represents a simple curve. However, the image may

not be perfectly smooth. Therefore, a certain tolerance is set, i.e. return 1 if both the numbers of rows and columns having more than one run, fall within the permitted tolerance. Otherwise return 0.

no_innerloop

For all the punctuation types considered in *date_zone* image segmentation and the printed digit '1', there exists no inner loop in the component. Hence, the *no_innerloop* feature algorithm returns 1 if there is no inner loop detected on the component, and returns 0 otherwise.

By applying this feature, connected components with one or more inner loops are excluded from the possible candidates for punctuation or printed digit '1'. The efficiency of the segmentation system is also increased in this way.

3.3.2 Spatial features

The four punctuation types considered in this application normally appear at different positions on the *date_zone* image. *Slash* is usually written long and extends from the top to the bottom of the entire image, while hyphen is often positioned around the *middle_zone* of the image. Most of the time, period and comma are seen at a relatively low position. All these indicate that the detection of the above punctuations is not context free. It is very important to obtain the information about the location of a connected component with respect to the entire date image as well as its neighbours. This is why the spatial features, *exceed_neighbour*, *at_middlezone*, *mid_to_neighbour*, *below_lowerhalf* and *low_to_left*, are developed, and they are described below.

Image reference lines

The *upperhalf* and *lowerhalf* reference lines of the date image must be defined in order to implement the spatial feature algorithms. Conventionally, the positions of these two reference lines are located according to the distribution of image histogram. However, when T-crossings exist on the image, such methods may produce erroneous results. Therefore, a method based on the number of horizontal runs instead of image histogram is used in this research work.

First, the maximum number of horizontal crossings on the entire *date_zone* image, *max_run*, is obtained. Then, the first line reached ($max\_run/2$) from the top is defined as the *upperhalf* reference line, and the first line reached ($max\_run/2$) from the bottom is defined as the *lowerhalf* reference line. A more convincing result (as shown in Figure 9) is produced in this way. The image *middle_zone* is defined as the area between the *upperhalf* and *lowerhalf* reference lines.

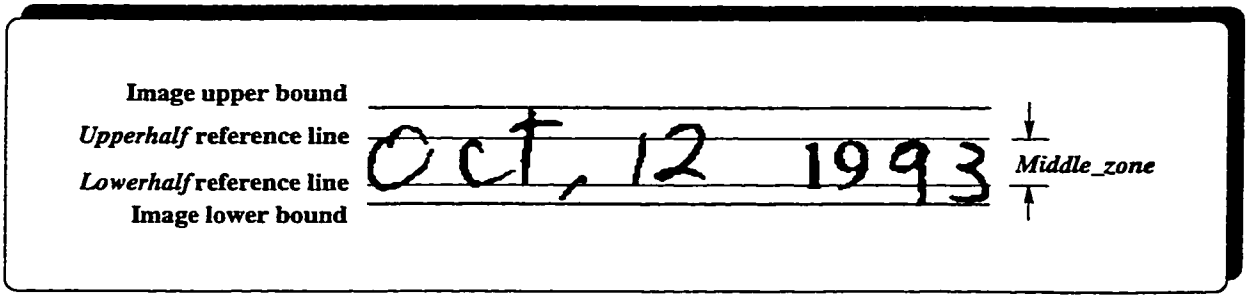


Figure 9: Illustration of image reference lines

exceed_neighbour

As a separator, slash is often written long enough to exceed its neighbours, so as to separate different fields and also avoid confusion with the digit '1'. For this reason, the feature *exceed_neighbour* is introduced in punctuation detection.

This feature algorithm compares the height of the connected component with those of its left and right neighbours. It returns 1.0, if the height of the component is as long as or longer than those of its two neighbours. Otherwise, return

$$comp\_height/max(left\_neighbour\_height, right\_neighbour\_height).$$

at_middlezone

The *at_middlezone* feature compares the location of the connected component with the position of the date image *middle_zone*. It is introduced to describe those components written close to the middle of the image, such as hyphen.

1.0 is returned, if the component falls completely within the *middle_zone*. 0.0 is returned, if the component falls completely above the *upperhalf* reference line or below the *lowerhalf* reference line. Otherwise, it returns the ratio of the height of the

part which falls within the *middle_zone* to the height of the component (Figure 10), i.e.

$$(comp\_xb - upperhalf)/comp\_height$$

if part of the component is above the *upperhalf* reference line, or

$$(lowerhalf - comp\_xt)/comp\_height$$

if part of the component is below the *lowerhalf* reference line.

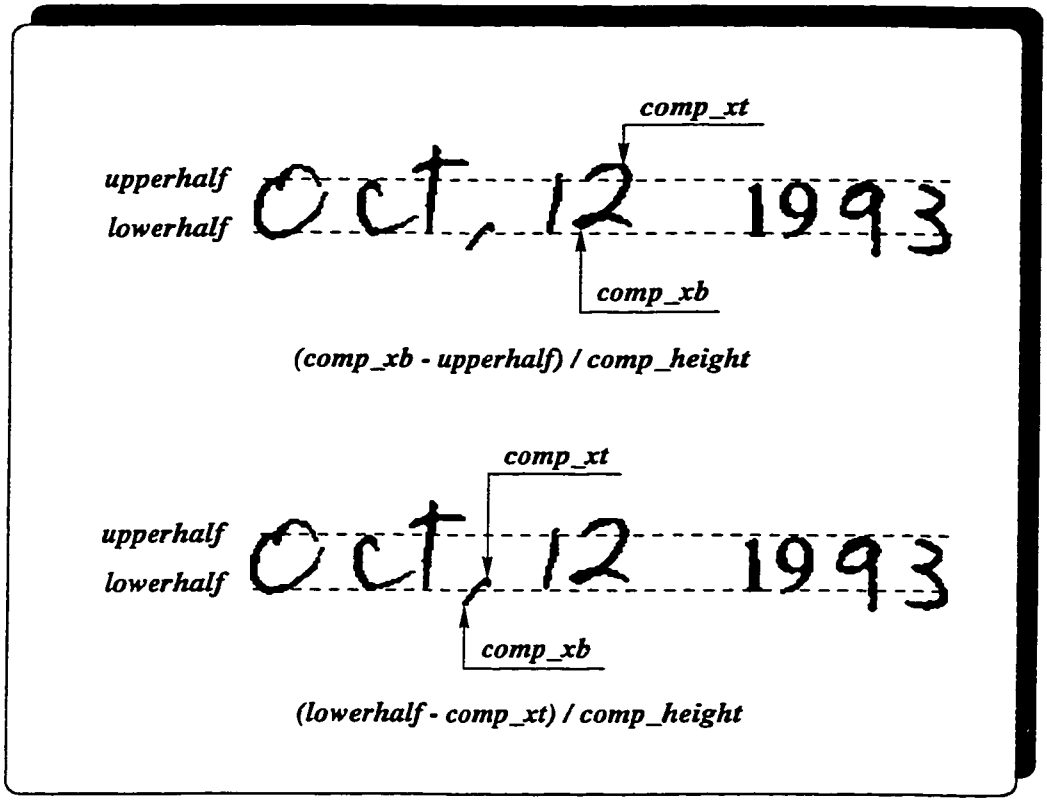


Figure 10: Illustration of the *at_middlezone* feature

mid_to_neighbour

The *mid_to_neighbour* feature compares the position of the connected component to the closer of its two neighbours. The value returned indicates how close the component is to the middle of its closer neighbour. This feature is introduced in order to avoid

misinterpreting the cap of digit '5' or letter 'T as hyphen when it is disjoint from the rest of the character.

1.0 is returned if the middle line of the closer neighbour falls between the upper and lower edges of the component, i.e.

$$comp\_xt \leq neighbour\_mid \leq comp\_xb$$

0.0 is returned if the component falls completely below or above its closer neighbour. Otherwise, it returns

$$(comp\_xt - neighbour\_xt)/(neighbour\_mid - comp\_xb)$$

if the component falls between the top edge and the middle line of its neighbour, or returns

$$(neighbour\_xb - comp\_xt)/(neighbour\_xb - neighbour\_mid)$$

if the component falls between the middle line and bottom edge of its neighbour.

below_lowerhalf

The *below_lowerhalf* feature compares the position of the connected component to the date image *lowerhalf* reference line. It is incorporated to account for the fact that punctuations such as period and comma are usually written at a relatively low position on the *date_zone* image.

It returns 1.0 if the component falls completely below the *lowerhalf* reference line, and returns 0.0 if the component falls completely above the *upperhalf* reference line. Otherwise, return

$$(comp\_xt - upperhalf)/(lowerhalf - upperhalf)$$

Therefore, if the top edge of the component is far from the *upperhalf* reference line but close to the *lowerhalf* reference line, the return value is increased.

low_to_left

Punctuations period and comma are written at a relatively low position on the *date_zone* image. Nevertheless, in most cases, it is low with respect to its left neighbour. For this reason, the feature *low_to_left* is introduced. Boolean value is returned

by this feature algorithm, where 1 indicates the component falls completely below the middle line of its left neighbour, while 0 indicates the component falls completely above the middle line.

3.4 Detection of Printed '19'

It is observed that on the *date_zone* of all standard bank cheques, digits '1' and '9' are printed completely separate from each other, unless they are made touching either by people's handwriting or by unexpected "black" noise. Each printed digit should consist of one and only one connected component, unless improper binarization or unexpected "white" noise occurs. As these two digits are always printed adjacent to each other, it is reasonable to assume that the component representing '9' can be located closely at the right side of printed '1'. In addition, these two digits are printed solidly at the same height. An inner loop exists in the component of '9'. Based on the above observation, a method for detecting printed '19' was proposed.

The printed digit '1' is to be located prior to any other tasks. Starting from the right end of the *date_zone* image, all the connected components are examined one by one until printed '1' is found or the left end of the date image is reached. Considering the fact that '1' is printed as a vertical stroke on most bank cheques and the slope over its contour is approximately the same, the *slope* feature is used. The *narrow* and *high_density* features are also used since the stroke is relatively narrow and most part of its bounding box is filled with black pixels. Once the candidate for printed digit '1' is located, its right neighbour is examined to see if it can be considered as printed digit '9'. If all the criteria are met, i.e. if the component contains one and only one inner loop, and it is of more or less the same height as its left neighbour, then these two adjacent connected components are assigned to be printed digits '19'. Two subimages, *Day&Month* and *Year* subimages are therefore obtained by simply removing the components for '19' from the *date_zone* image.

This method worked well on *CENPARMI cheques*. However, the performance decreased when it was applied on *real cheques*. Many factors existing on *real cheques* but not on *CENPARMI cheques* could account for the difference. For example, *CENPARMI cheques* are printed on white paper, but bank cheques are printed with various background images and colors. It is obvious that the binarization and handwritten

information extraction processes (to obtain separate images for *date_zone*, courtesy amount and legal amount from the cheque image) will be much more difficult to implement. Consequently, larger quantities of noise were introduced to the binary *date_zone* images extracted from *real cheques* which caused the following problems to the detection of printed '19':

- Printed '1' and '9' are touching each other, or either of them is touching its neighbour;
- The stroke of either '1' or '9' is broken;
- The connected component for '1' or '9' is distorted due to "white" noise;
- The connected component for '1' or '9' is distorted due to "black" noise.

To solve the above problems, additional information is obtained from *real cheques* by the binarization and extraction processes. On many bank cheques, there are two horizontal lines printed separately in the *date_zone*, and the printed digits '19' are positioned in the gap between the two lines. If the coordinates of this gap are available, the process of detecting '19' becomes unnecessary. *Day&Month* and *Year* subimages can be obtained by simply cutting the date image using these coordinates. Otherwise, if there is only one horizontal line or no line printed in the *date_zone*, the '19' detecting process still needs to be executed, but the parameters in each feature algorithm involved are re-adjusted to allow more flexibility and tolerance in a real application environment.

3.5 Detection of Punctuations

3.5.1 Method for punctuation detection

In this work, different feature combinations are used to detect each individual punctuation. In other words, in order to verify whether a connected component belongs to a certain class, it is examined only for the presence of the most distinguished features of that class. Since the 4 punctuations considered in date image processing are quite different in characteristics, different sets of features are chosen for the detection of each.

Furthermore, feature algorithms returning different types of value (i.e. Boolean and confidence values) are also applied in different situations for different purposes. Those returning Boolean values are used to make relatively "hard" decision, such as excluding a connected component from being considered as a possible candidate of punctuation. Those returning confidence values are used to compute the overall likelihood of a component to a certain punctuation.

As opposed to the process of locating machine-printed digits '19', the process of detecting punctuations starts from the left end of *date_zone* image to the right, one connected component per step until the right end of the *Day&Month* subimage is reached. Considering the fact that nobody would write a punctuation at the beginning of *date_zone*, the leftmost component positioned on *Day&Month* subimage is not analyzed by this process, while all the others are carefully examined one by one.

Since there is no inner loop in any of the 4 punctuations, and they all present very simple curvatures, it is not necessary to analyze those connected components with either inner loops or relatively complicated curvatures for punctuation detection. Two shape features, *no_innerloop* and *simple_curve*, are applied to exclude such components. A significantly large amount of unnecessary computation is also reduced in this way.

When a "simple" connected component is being processed, it is first pre-classified using contextual information. If the component is considered to be long, i.e. its height is longer than the image *middle_zone* height, it is matched against slash only. If the return value of feature algorithm *low_to_left* is 1, then the component is matched against period and comma only. Otherwise, the component is matched against hyphen only.

To evaluate the likelihood of a connected component to a certain punctuation, the average of all confidence values returned by the feature algorithms is computed. Each returned value identifies the presence of one of the distinguishing features of a certain punctuation, and the average values of all the features tend to compensate for each other. For example, as shown in Figure 11, the hyphen on the left appears relatively low with respect to the entire *date_zone* image as well as its neighbour. If only spatial features, *at_middlezone* and *mid_to_neighbour*, are considered, the component was rejected as a hyphen as normally a hyphen resides around the middle of the date image as well as its neighbour. But this situation was improved when two shape

features, *flat* and *high_density*, were also incorporated to compensate for the deficiency of the spatial features.

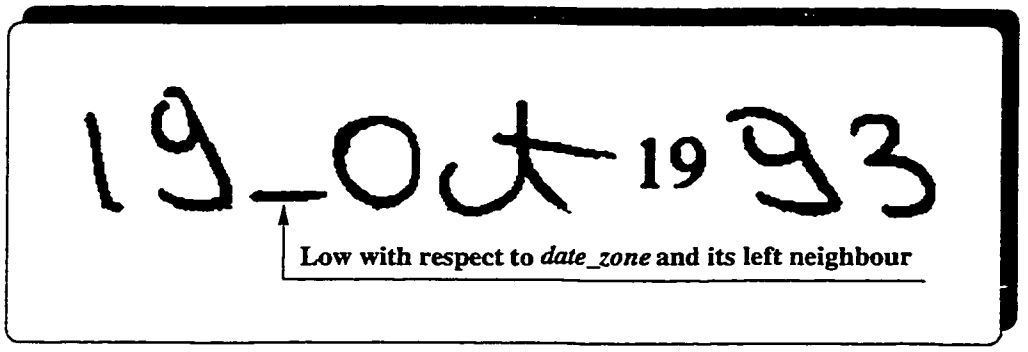


Figure 11: Illustration for feature compensation

3.5.2 Feature combinations for punctuation detection

The selection of feature algorithms for detecting each punctuation was based on a certain amount of experimentation on all the developed shape and spatial features. The following 4 sets of features were chosen in the punctuation detecting process:

- Slash: *narrow* and *exceed_neighbour*;
- Hyphen: *high_density*, *flat*, *at_middlezone* and *mid_to_neighbour*;
- Period: *high_density*, *small* and *below_lowerhalf*;
- Comma: *narrow*, *small* and *below_lowerhalf*.

3.6 Detection of Maximum Gap

Punctuations detected in the middle of the *Day&Month* subimage are used to separate *Day* and *Month* from each other. However, when no such punctuation is located, a maximum gap needs to be detected prior to segmenting the *Day&Month* subimage. Processing such handwritten information is actually equivalent to processing a free-style text line where spacing between different fields is neither uniform nor pre-defined. Nevertheless, as many people tend to write everything with a slant (due to their

posture and the way they hold their pens, etc.), connected components appear to overlap each other horizontally on a number of images.

Many algorithms exist in the literature to compute the distances between pairs of connected components [SC94]. However, most of them become much more complicated when the bounding boxes of the connected components overlap. In addition, threshold values need to be established for these algorithms. A lot of experiments have to be conducted in order to obtain proper threshold values, which are also highly dependent on the set of experiment data. Errors could easily occur if the test data differ considerably from the experiment set processed.

In this thesis, an algorithm has been developed to locate the maximum gap between connected components. It works as follows:

- Scan all the lines falling within the image *middle zone* from left to right;
- During each scan, record two connected components which present the maximum component distance on this line;
- After scanning all the lines, the maximum gap occurs where the maximum distance between the two connected components occurs the most frequently.

As described above, this method is completely independent of threshold values. It is based on image *middle_zone* where a lot of information about *date_zone* can be observed. It only looks at the distance between neighbouring connected components, and therefore no special training is needed. It is also quite effective and computationally efficient, especially when connected components are overlapping horizontally and skewed (refer to Figure 12 for details). Statistics shows that among those *CENPARMI cheques* where there is no punctuation written between *Day* and *Month*, or there is only one punctuation written at the end of *Day&Month* subimage, 80.11% of the maximum gaps are correctly detected for the purpose of segmentation.

3.7 Segmentation and Hypotheses

Once the punctuation detecting process is completed on the *Day&Month* subimage, a set of heuristic rules are applied to obtain the final result of segmentation and

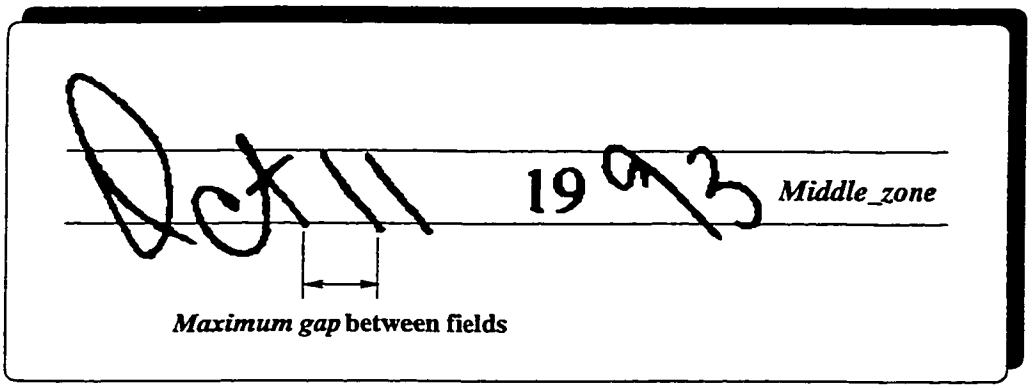


Figure 12: Illustration of maximum gap between fields

hypotheses. The segmentation process cuts the *Day&Month* subimage into two parts, while hypothesis is made to identify which part represents *Day* and which represents *Month*, and whether *Month* field is written in letters or digits.

The heuristic rules used at this step are developed based on:

- how many number of punctuations are detected;
- where they are detected;
- what types of punctuations they are.

3.7.1 No punctuation detected

If there is no punctuation detected in *Day&Month* subimage, a module named *SegbyMaxgap* will be called.

In this case, a maximum interword gap must be located in order to segment the image at the gap. As discussed before, *Month* is most probably written in word when punctuation is not written in *date_zone*. Hence, here we assume that when no punctuation is detected, *Month* is written in cursive script.

Furthermore, *Month* word, either in full or abbreviated format, is composed of at least three letters, while *Day* is composed of at most two digits. As observed from *CENPARMI cheque* database, 99.04% of the *Month* written in cursive script appears to be wider than the *Day* written on the same *date_zone*. Therefore, by simply comparing the widths of the two segments, the one with a larger width is

assigned as the *Month* field written in letters, and the one with a shorter width is assigned as the *Day* field.

3.7.2 One punctuation detected

If there is only one punctuation detected in *Day&Month* subimage, a module named as *SegbyOnepunc* will be called. It further analyzes two scenarios.

Punctuation detected at the end of the subimage

If the punctuation is detected at the end of *Day&Month* subimage, and it is interpreted as either a slash or hyphen, it is assumed to be detection error. This is due to the fact that if there is one and only one such punctuation in *Day&Month* subimage, it is usually written in the middle, i.e. between *Day* and *Month*. There is no reason why anybody would write a single slash or hyphen at the end. Therefore, in this case the detected punctuation is simply ignored, and *Day&Month* subimage is further processed by calling *SegbyMaxgap*, as if no punctuation were detected.

If the only punctuation detected at the end of *Day&Month* subimage is either a period or comma, it is simply excluded, and the rest of the subimage is processed by calling *SegbyMaxgap*. This is bacause most people would write a period or comma after writing *Day* and *Month*. It comes naturally from people's writing habit. The intention is just to identify the end of the *Day&Month* subimage. The removal of such punctuation will not affect the result of segmentation and hypotheses.

Punctuation detected in the middle of the subimage

If the punctuation is detected in the middle of *Day&Month* subimage, and it is interpreted as either a slash or hyphen, then the position of the punctuation is considered as the cutting point of the subimage. Even though slash or hyphen is written mainly when *Month* is written in digits, exceptions still exist. Hence, the widths of the two segments are compared. If one is much wider than the other, *Month* is assumed to be written in word, and the wider segment is assigned as *Month* and the narrower one as *Day*. Otherwise, if there is no significant difference between both segments, *Month* is assumed to be written in numerals. In this case, the representation of *Day* and *Month* cannot be clearly identified until the recognition results are sent to a parser.

If the only punctuation detected is either a period or comma, *Day&Month* subimage is then segmented at the position where the punctuation is located. As period or comma written in the middle of *Day&Month* usually identifies the end of *Month* word (based on the fact that 86.89% of those *CENPARMI cheques* where a period is detected in the middle of *Day&Month* subimage have *Month* word written prior to period, and 61.54% if a comma is located), the part to the left of the punctuation is assigned as *Month* field written in letters, and the part on the right is assigned as *Day* field.

Two modules are implemented for the above two scenarios. The first one is named as *SegbySH*, and the second one as *SegbyPC*.

3.7.3 Two punctuations detected

If two punctuations are detected in *Day&Month* subimage, a module named as *SegbyTwopunc* will be called. Two different scenarios are further discussed here.

Two identical punctuations detected

If the two punctuations detected are of the same type, and located in the middle and at the end of *Day&Month* subimage respectively, then the one at the end is excluded from the subimage, because the one in the middle can provide sufficient information to segmentation and hypotheses. Either of the modules *SegbySH* and *SegbyPC* will be applied to the rest of the subimage, depending on which type of punctuation is detected.

If none of the detected punctuations is located at the end of the subimage, one of them must have been detected by mistake since nobody would write two identical punctuations in the middle of *Day&Month*. In this case, a selection between the two needs to be made by the segmentation system in order to retain only one punctuation, and the one with a higher overall confidence value from all feature algorithms will be the winner. Then, either *SegbySH* or *SegbyPC* will be applied to the entire *Day&Month* subimage, depending on which type of punctuation is detected.

Two different punctuations detected

The problem is more complicated if the two punctuations detected are of different types.

If one is a period, the other is a comma, and one of them is located at the end of *Day&Month* subimage, only the one in the middle will be retained. Since both period and comma are used when people write *Month* in cursive script, there will not be any impact on date segmentation if the period or comma at the end is excluded from the subimage. To complete *date_zone* segmentation and generate hypotheses, module *SegbyPC* is called and the information provided by the punctuation located in the middle is utilized.

The same theory is applied to the scenario when a hyphen and a slash are detected. Module *SegbySH* is called in this case.

It is also possible that two punctuations are detected, and a slash is located as the immediate right neighbour of a period or comma or hyphen. However, based on the experiments on *CENPARMI cheques*, this slash is probably confused with digit '1'. Therefore, the segmentation system will keep processing as if this slash did not exist, and the module called in this case is either *SegbyPC* or *SegbySH*.

Apart from the above, there are still some cases where "incompatible" punctuations, i.e. a period and a slash, a period and a hyphen, etc., are detected at the same time. However, chances for such cases to occur in real life are slim (only 5 such images are seen in 4564 *CENPARMI cheques*). Therefore, only one punctuation is retained by the segmentation system, and either *SegbyPC* or *SegbySH* will be called. The selection here is done by comparing confidence values and corresponding weights, i.e. weighted confidence values, between the two. The *weight* for each type of punctuation is obtained from the training part of the *CENPARMI cheque* database which is used for experimenting the segmentation system.

3.7.4 More punctuations detected

If more than two punctuations are detected in *Day&Month* subimage, a module named *SegbyMorepunc* will be called.

Since it is almost impossible to have more than two punctuations written in *date_zone*, only two of the detected punctuations should be retained in this case. The

selection here is made by comparing the weighted confidence values of all punctuations. *Day&Month* subimage is then processed by calling *SegbyTwopunc* and utilizing the information provided by the two winners.

Chapter 4

Experimental Results

4.1 Database

In order to enable the CENPARMI research group to train the Cheque Processing System and conduct necessary experiments, large quantities of handwriting samples are required. For this reason, two separate sets of data, referred to as *CENPARMI cheques* and *real cheques* respectively, have been collected from different sources for different purposes.

4.1.1 *CENPARMI cheques*

During the design and development phase of an automatic cheque processing system, it was very important for the team to have access to a large quantity of bank cheques in order to get an insight into various possible ways people would write their cheques. Hence, it was highly desirable to have a dedicated cheque database available for the research. However, such data did not exist at that time. Due to security and confidentiality considerations, it was also very difficult to have access to real cheques from banks or utility companies even for R&D purposes. This made it almost impossible to build a database by collecting and scanning bank cheques. Therefore, the CENPARMI cheque processing research group decided to create their own database from scratch.

First, "The Bank of Concordia" cheque (Figure 13) was carefully designed. It has a similar size and layout as regular bank cheques. Its background is white, and all

the lines are printed in special drop-out ink which is invisible to the scanner. This facilitates the process of extracting written information, and also produces extracted images of better quality.

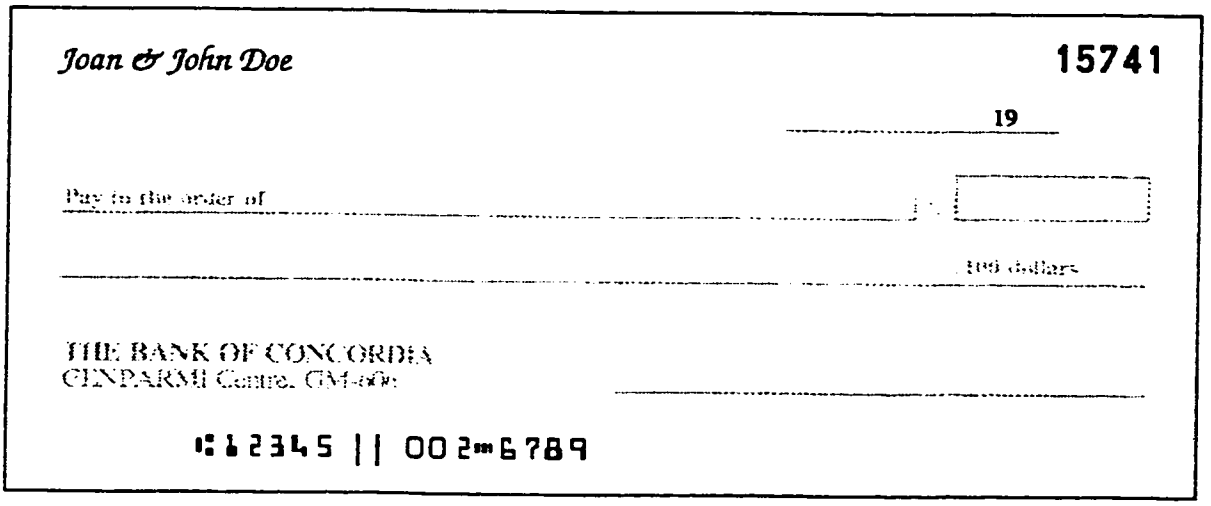


Figure 13: Sample of a *CENPARMI cheque*

Then, the CENPARMI research group visited different classes at Concordia University and Ecole Polytechnique Montréal. Each student was asked to fill in 3 cheques with a pen or ballpoint-pen. On every cheque, they were required to fill in a predefined amount, but free to fill in any information in the *date_zone* as well as the fields of "Pay to the order of" and "signature". All the filled cheques were collected, scanned and stored as binary images, ready for the extraction and further processing of each item of written information[Gui95]. A sample of filled *CENPARMI cheques* is shown in Figure 14.

4.1.2 *Real cheques*

At the end of the development cycle, a limited number of filled bank cheques were received by the CENPARMI research group from utility companies. These cheques were originally issued by different banks in the Montréal area (e.g. Bank of Montréal, Royal Bank, TD Bank, CIBC Bank, Bank of Nova Scotia, etc.). Unlike *CENPARMI cheques* which were written mainly by university students, the *real cheques* were

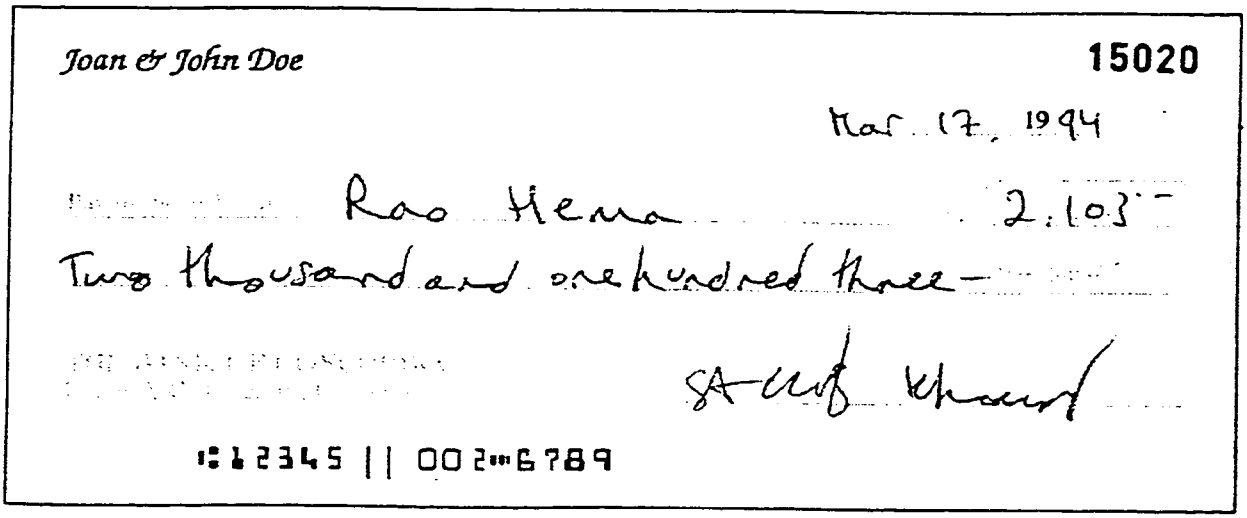


Figure 14: Example of filled *CENPARMI cheque*

written by the general public. They were scanned into a testing database in order to verify the robustness and performance of the entire cheque processing system, which includes handwritten information extraction, legal amount recognition, courtesy amount recognition and date recognition, in a real application environment.

Those cheques forming the test database were actually designed by different banks, therefore the layout and geometric measurements are not the same. In order to accommodate preferences of the general public, real bank cheques in various colors, backgrounds, fonts, lines and boxes are printed. Unfortunately, they are visible to the scanner and thus may introduce much extraneous information to the binary image of each *real cheque*. Compared with the processing of *CENPARMI cheques*, it is much more difficult and complicated to extract handwritten information from *real cheques* and produce high-quality binary images for each item. Hence further processings of legal amount, courtesy amount and date are much more challenging.

4.1.3 Database truthing

The most obvious way of *date_zone* image truthing is to manually extract each field (i.e. *Day*, *Month* and *Year*) and store them in separate files. However, this scheme requires a large amount of extra memory which can be very costly. On the other hand, the contextual information of *date_zone* images, such as the appearance of

punctuations which is the key factor in date image segmentation, would be completely lost.

A powerful truthing tool should be capable of retaining sufficient information about *date_zone* images, e.g. the date written, where it is written, whether punctuations are present, etc., without much redundancy. Therefore, we chose not to cut the original date image, but to attach a message string containing all the above information, to each image.

In order to facilitate the truthing procedure, a MOTIF [HF94] user interface (Figure 15) was developed. Within this tool, all the events are strictly mouse-driven. To tag a connected component, the user simply clicks at its top-left corner, drags the mouse to the bottom-right corner and clicks again, then a pair of coordinates uniquely defining the component is added to the message string. To specify the exact written information (e.g. *Month* word), the user only needs to click on the tear-off menu and the word is automatically added to the message string. Different colors are applied to the tagged components in order to inform the user which components have not been tagged and imply the tagging sequence as well.

The *date_zone* images are tagged from left to right. All the three fields, *Day*, *Month* and *Year*, as well as punctuations and the printed digits "19" are tagged according to the sequence of their appearance on the date image (refer to Figure 16 for detail). The characters ';', ',' and ':' are placed in the message string in order to separate the tagged objects. The word "Century" designates the printed digits "19". Separate characters and digits are also tagged for future use.

Based on the message string, it is easy to extract any desired part from the date image. Shown in Figures 15 and 16, ':' indicates the start of a pair of coordinates which uniquely identify a certain connected component, and ';' indicates the start of handwritten information. As an example, the word "Oct" is composed of two connected components, which are identified by (4, 27) and (18, 85), i.e. the leftmost coordinates on the first line of 'O' and 'ct' respectively.

In fact, four dedicated databases can be created for date image processing:

- A database for *Month* words, which consists of both English and French versions, as well as their abbreviations;
- A database for *Day*, which consists of numerals from 1 to 31;

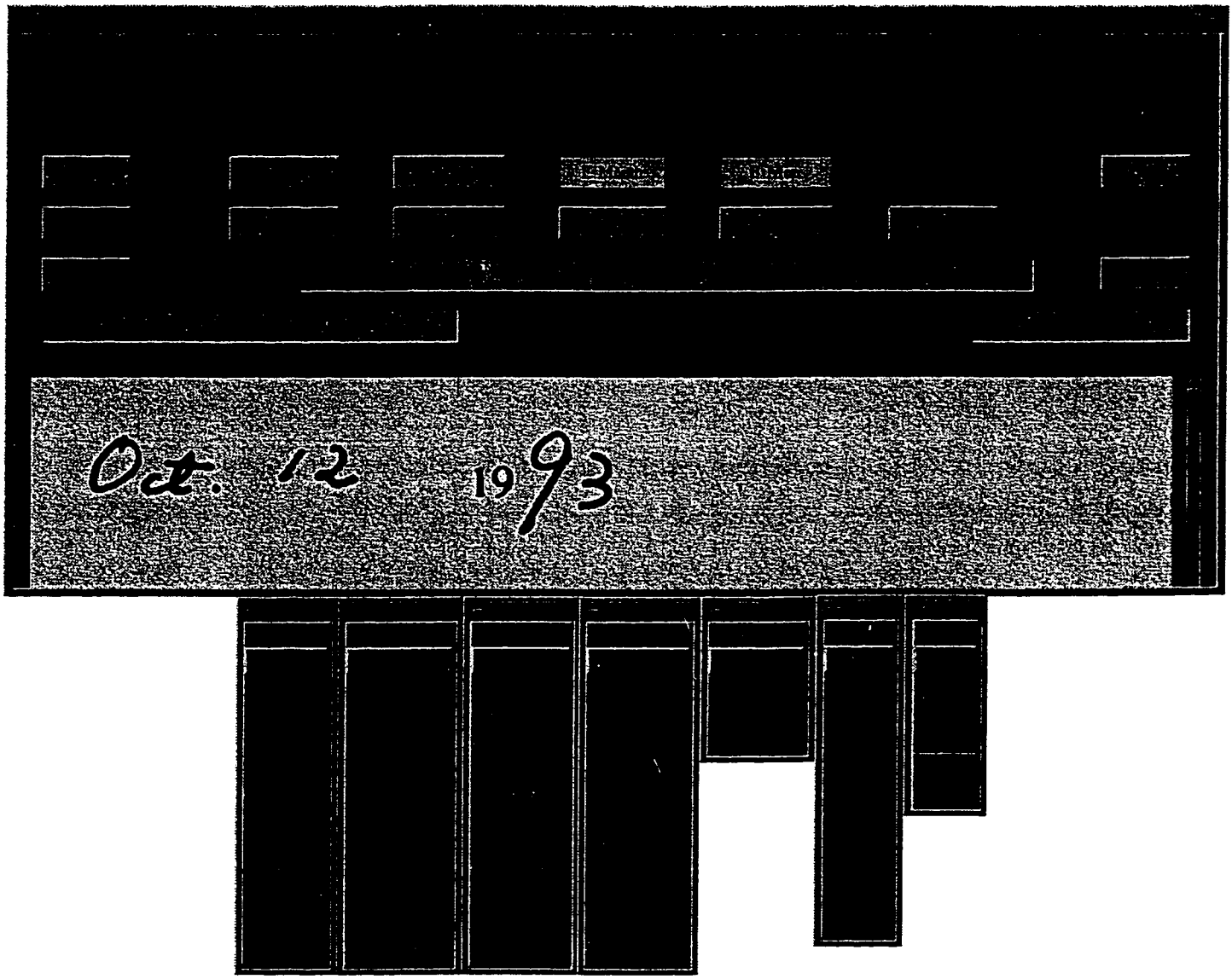


Figure 15: User interface for the tagging of *date_zone* images

- A database for *Month* (if written in digits), which consists of numerals from 1 to 12;
- A database for *Year*, which consists of numerals ranging from 00 to 99.

Apart from the above, a variety of useful statistics can also be gathered utilizing the message strings from tagged *date_zone* images. In Chapter 2, *date_zone* writing styles are precisely analyzed by giving detailed statistics. This set of data is obtained by extracting all the handwritten information, i.e. exact date as well as punctuations, from message strings. As will be shown later, performance of segmentation, punctuation and maximum gap detection are also discussed. Since in the message string each connected component is uniquely identified and sufficient contextual information is retained, such statistics can be collected by comparing the segmentation or detection result against the counterpart extracted from the message string.

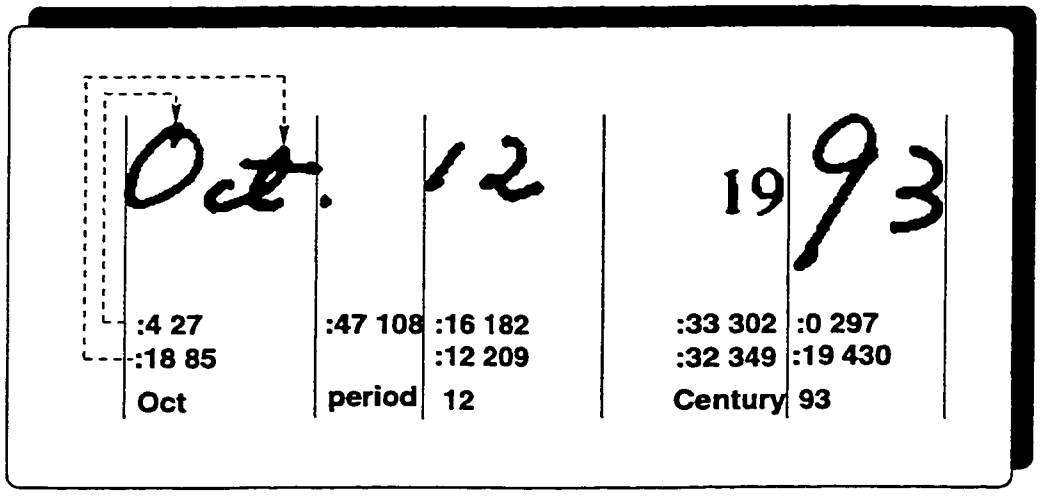


Figure 16: *Date_zone* image message after the tagging

4.2 Result of Segmentation

Date_zone image segmentation is a critical part of date image processing, because it determines what kind of information is sent to the digit and word recognizers. If incorrect subimages are provided by the segmentation system, even though the

error only occurs on one subimage (for *Day* or *Month* or *Year*), there is no way that the recognition system can produce a correct output. In this case, the entire date processing fails.

For this reason, statistics on the performance of *date_zone* segmentation is produced specifically. This will assist detailed analysis of the performance of the entire date processing system, and also facilitate the identification of the areas where further improvements are needed.

4.2.1 Result on *CENPARMI cheques*

During the development phase of cheque processing system, *CENPARMI cheques* were the only data available, the date processing system was therefore designed and implemented mostly based on this set of data. Table 4 shows the performance of *date_zone* image segmentation on its 4564 images, where "correct" means that the cutting positions are properly located and hypotheses are correctly generated so that each field can be sent to the appropriate recognizer (digit or cursive word).

Table 4: Performance of segmentation on *CENPARMI cheques*

	Total no.	Correct (%)	Reject (%)	Error (%)
CENPARMI cheques	4564	74.96	7.82	17.22

4.2.2 Result on *real cheques*

Due to the high complexity involved in extracting handwritten information from *real cheques*, the quality of *date_zone* images extracted from *real cheques* is not as good as that of the *CENPARMI cheque* database. In addition to the lower image quality, invalid *date_zone* images which were not seen among *CENPARMI cheques* also appeared in the *real cheque* database. As illustrated by the examples in Figure 17, invalid *date_zone* images here refer to those where date information is not extracted, or there is excessive black or white noise which badly distorts handwritten information, or some fields are missing. The date processing system, originally designed and implemented only on "ideal" images, was not yet fully capable of dealing with "bad"

images or rejecting any invalid images. For this reason, a set containing 494 *real cheques* was created to "adjust" the system in a real application environment. This set is thus called *Adjustment set.*

The second set, containing all the 391 remaining *real cheques*, was used to test the performance and robustness of the system at single execution. These cheques were designated as *Testing set.*

Table 5 shows the performance of *date_zone* image segmentation on both the *Adjustment set* and the *Testing set.*

Table 5: Performance of segmentation on *real cheques*

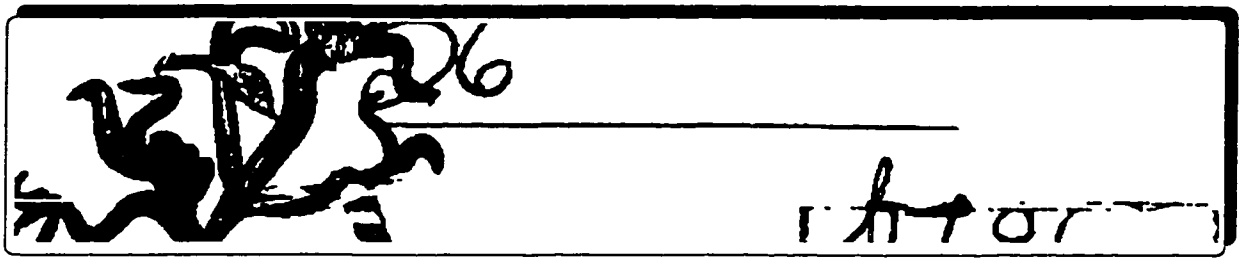
	Total no.	Correct (%)	Reject (%)	Error(%)
Adjustment set	494	61.74	11.34	26.92
Testing set	391	65.73	12.28	21.99

4.2.3 Performance analysis of segmentation

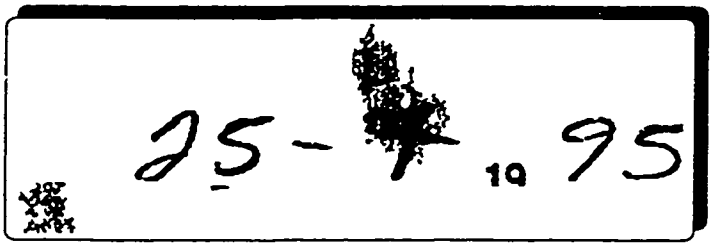
As the first step of date image segmentation, the proposed method for detecting printed digits '1' and '9' performs very well. On the *CENPARMI cheque* database, it can successfully locate all the printed '19' provided the digits are not touched by the handwritten data. On the *real cheque* database, this method is applied when no additional information (detailed description in Section 3.4) is available. Four errors in detection occur among the 299 images to which this method is applied.

In Tables 4 and 5, the rejection of a *date_zone* image can be due to one or more of the following reasons:

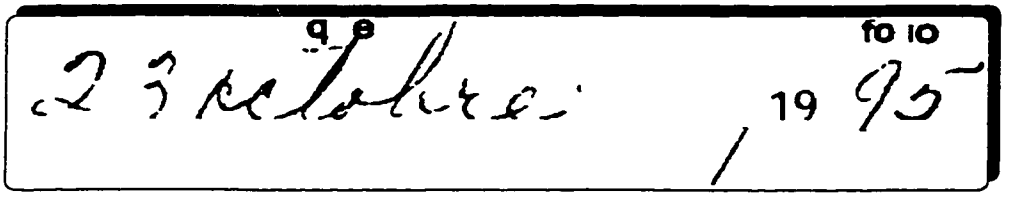
- If an image contains too many connected components, the segmentation system will not only become more difficult, but also tend to produce erroneous results. If a large number of broken pieces appear on the image due to improper binarization, quite a few of them might be mis-interpreted as punctuations since the system is very "sensitive" to small components. If the excessive number of components come from information other than date (Figure 18), there will be



(a) Date information not extracted



(b) Excessive black noise



(c) Excessive white noise

(d) Missing *Day & Month*

Figure 17: Examples of invalid *date_zone* images

too much noise introduced in each subimage. Therefore, in order to generate more reliable results, such images are simply excluded from further processing.

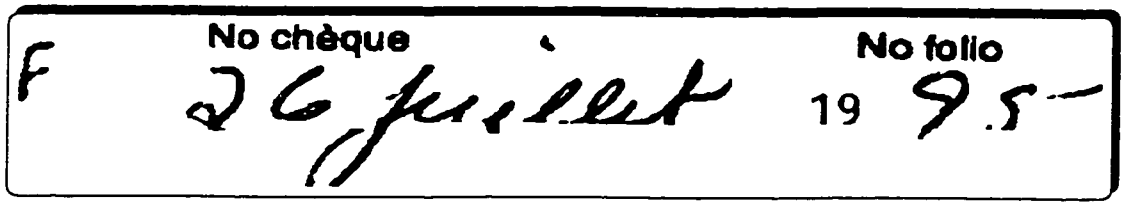


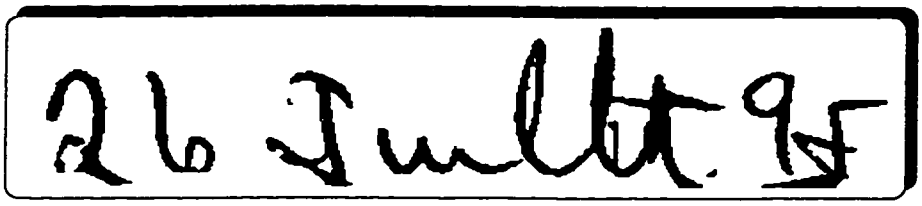
Figure 18: An example of *date_zone* image with too many components

- On the other hand, in some images there exists only one connected component representing the entire *date_zone*. This provides no clue to segmentation, and hence rejection is the only choice.

- In the *real cheque* database, the occurrence of invalid *date_zone* images proves to be a serious problem. As almost no valuable result could be produced from such images, the segmentation system is programmed to be capable of validating input images to a certain extent, e.g. rejecting those composed of only one connected component or containing excessive black or white noise, and terminating the processing when failure is encountered during validation.

- Printed digits "19" cannot always be detected in *date_zone* (refer to the examples in Figure 19). In some images, printed "19" are almost completely overwritten by other fields. In other images, '1' and '9' are touching each other, or touching the rest of the image. In addition, in quite a few images, '1' and '9' are distorted so that their features can not be detected properly. As locating "19" is the key step in *date_zone* segmentation, an image can be rejected if "19" cannot be identified.

- However, even if printed "19" are correctly detected, a date image can still be rejected. For instance, if the *Day&Month* subimage contains only one connected component, it provides no clue to further separate *Day* and *Month* from each other. If *Year* is not written after printed "19" (Figure 20), the image will be rejected as well since it conflicts with the *date_zone* segmentation assumptions (i.e. *Year* is always written after printed "19").



(a) Printed "19" overwritten by other fields

(b) '1' and '9' touching each other

(c) '1' and '9' touching with other fields

(d) '1' and '9' distorted

Figure 19: Examples of *date_zone* images where printed "19" are not detected

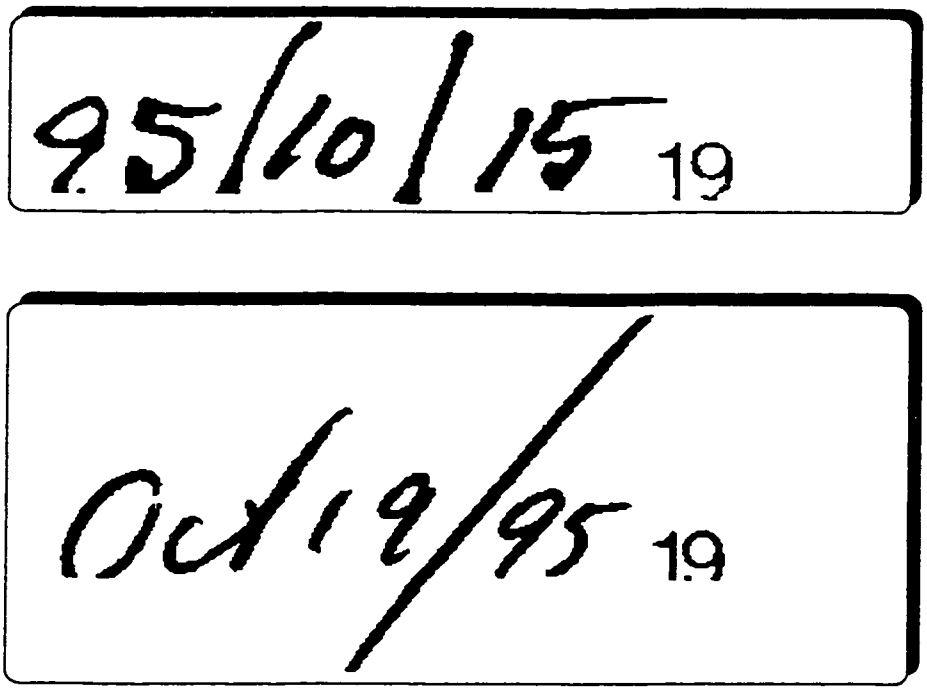


Figure 20: Examples of *date_zone* images where *Year* written before "19"

Errors can occur in *date_zone* image segmentation. A breakdown of error rate is shown in Table 6:

Table 6: Error rate analysis of *date_zone* segmentation

	Error (%)	Error1 (%)	Error2 (%)
CENPARMI cheques	17.22	2.74	2.91
Adjustment set	26.92	6.07	4.25
Testing set	21.99	4.60	6.39

- Since both the *CENPARMI* and *real cheque* databases were collected around the Montréal area where there is a significant French-speaking population, "Le" appears in a number of *date_zone* images as part of the *Day* information. However, no proper method has yet been found to detect such a short handwritten word. Approaches such as trying to locate "Le" from the leftmost part of the image using cursive word recognizer, have been tried. But the results were not encouraging. "Error1" refers to those incorrect segmentations caused by the word

"Le".

- At *date_zone* segmentation, assumptions are made that if slash '/' or hyphen '-' is detected then *Month* is written in numerals, if period '.' or comma ',' is detected then *Month* is written in word, if no punctuation is detected then *Month* is written in word, etc.. These assumptions are true most of the time (as indicated by Tables 1 and 2 in Chapter 2). However there are exceptions, which cause the segmentation system to interpret the writing style of *Month* incorrectly or designate *Day* subimage as month and vice versa, even though the *Day&Month* subimage is cut at the correct position. "Error2" refers to such errors (illustrated in Figure 21).

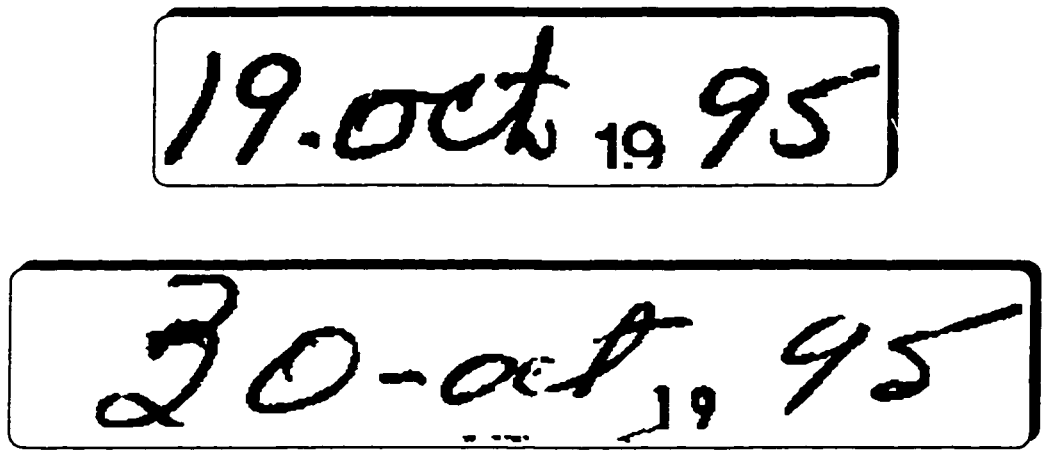


Figure 21: Examples of *date_zone* images conflict with segmentation assumptions

- Punctuation detection is also a critical step in *Day&Month* subimage segmentation, since different heuristic rules are applied to conduct processing based on the types and locations of punctuations found. Table 7 gives the performance of each punctuation detection algorithm on *CENPARMI cheques*. It shows that the methods proposed by this research work are effective at locating punctuations written on *date_zone* images.

  However, quite a few errors also occur during this step, i.e. mis-interpret components of other fields as punctuations. For example, broken strokes (most probably due to binarization) are mis-interpreted as punctuations (Figure 22),

digit '1' or letter 'l' as slash '/' (Figure 23), lower part of letter 'i' as comma ',' (Figure 24), etc.. Detailed statistics of each punctuation type on 4564 *CENPARMI cheques* is provided in Table 8.

Table 7: Performance of punctuation detection on *CENPARMI cheques*

	Slash (%)	Hyphen (%)	Period (%)	Comma (%)
Correct	94.72	94.02	92.80	93.63

Table 8: Erroneous punctuation detections on *CENPARMI cheques*

	Slash	Hyphen	Period	Comma
No. of error detections	351	176	68	132

- Correctly locating the maximum gap becomes very important if no punctuation is detected in *Day&Month* subimage. However, due to a variety of factors such as slanting of handwriting, strokes extending from one field to its adjacent field, the flexibility in leaving gaps between *Day* and *Month* when writing a cheque, and similar factors, errors also occur here. As already mentioned, the performance of gap detection reaches 80.11% on 4564 *CENPARMI cheques*.

4.3 System Integration

The input to the date processing system consists of binary *date_zone* images extracted from either *CENPARMI cheques* or *real cheques*. The segmentation program is first executed on the input image in order to obtain subimages for *Day*, *Month* and *Year* respectively. An assumption of whether *Month* is written in numeric or alphabetic format is made at the same time.

To further process the three subimages, two recognizers developed specifically for cheque processing application are involved. Numeric subimages, including *Day*, *Year* and *Month* (if written in numerals), are sent to the digit recognizer used for courtesy

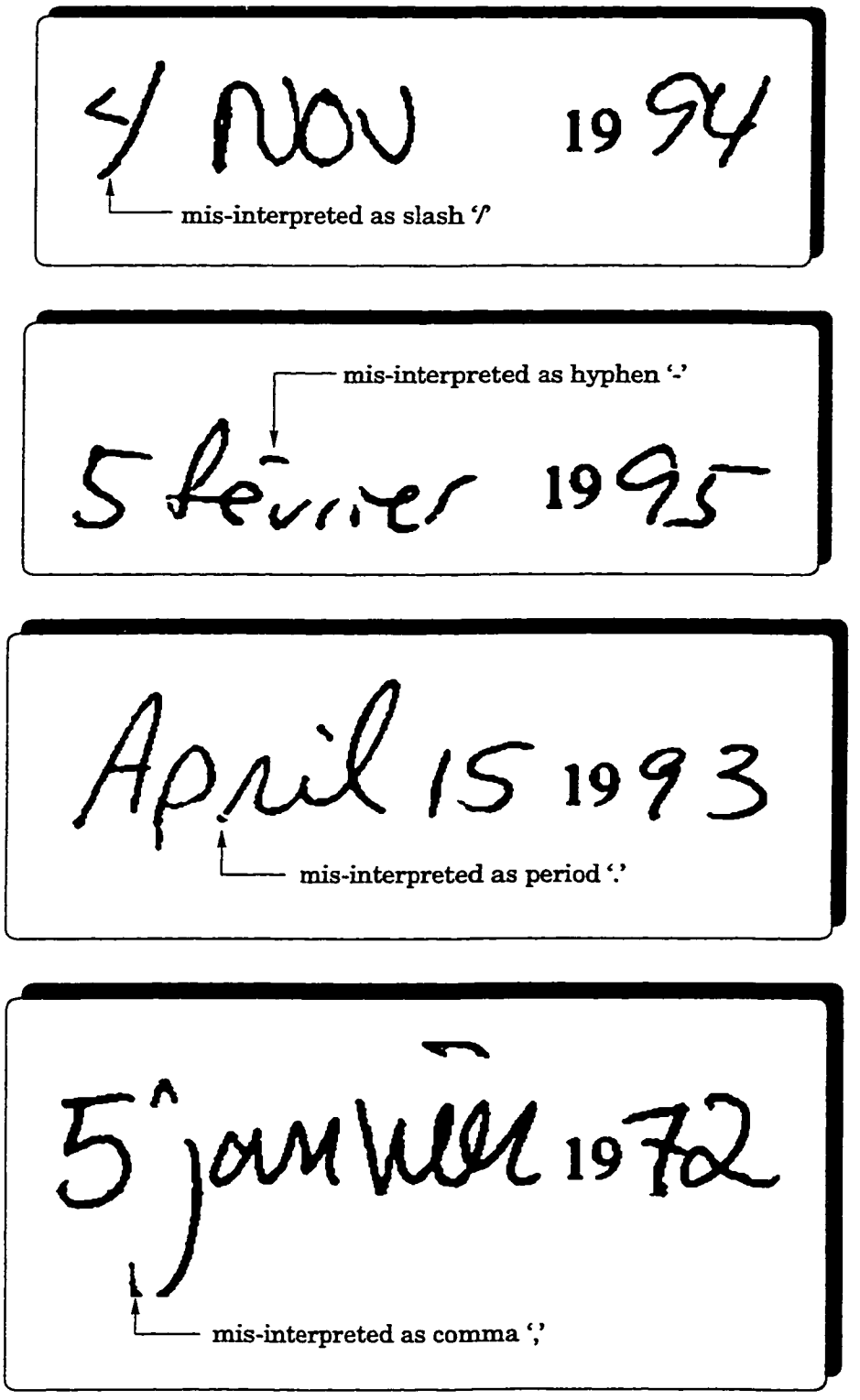
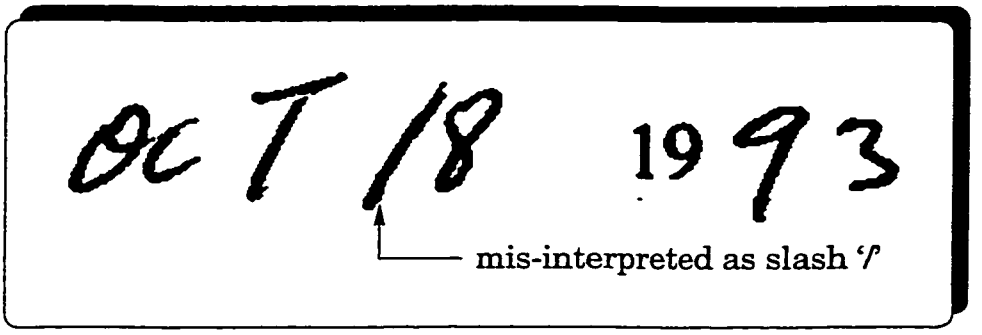


Figure 22: Examples of mis-interpretation of broken strokes



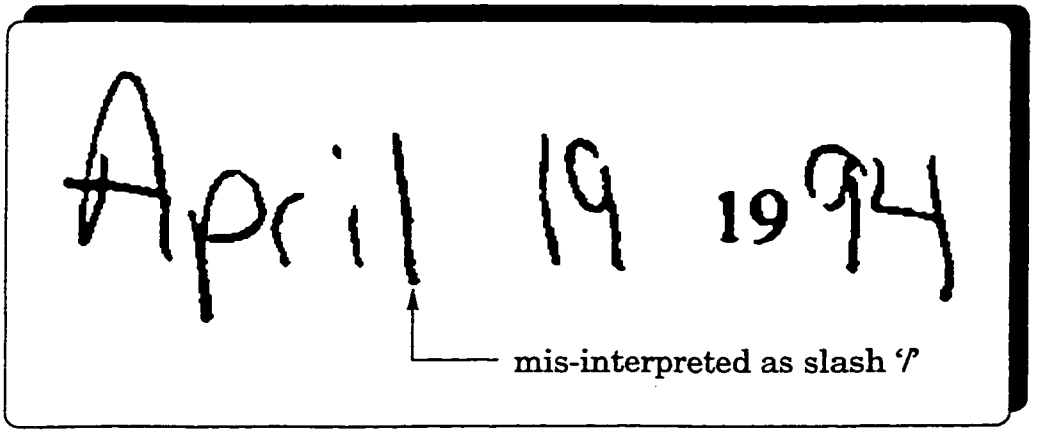


Figure 23: Examples of mis-interpretation of digit '1' and letter 'l'

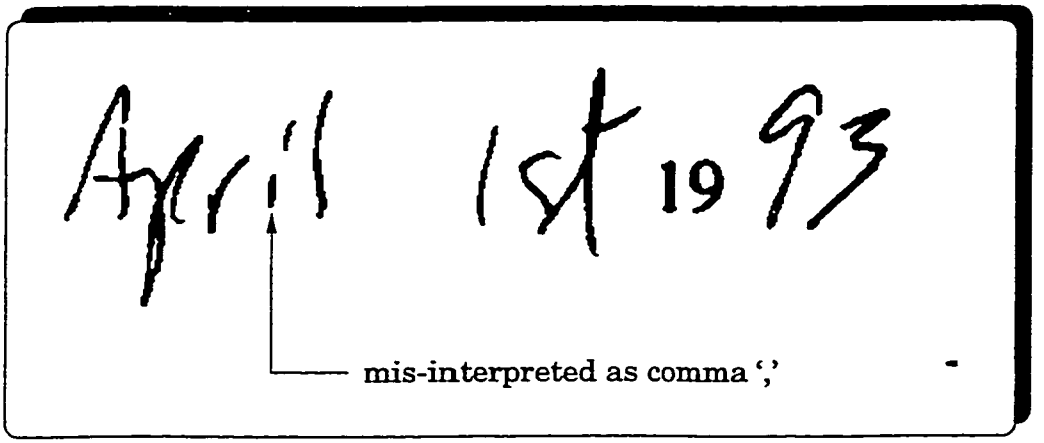


Figure 24: Examples of mis-interpretation of letter 'i'

amount processing [SLG+96]. Alphabetic field, i.e. *Month* if written in cursive scripts, is sent to the cursive word recognizer used for legal amount processing [GS95]. This recognizer has been trained on the lexicon of both English and French month words as well as their abbreviations, altogether 42 classes. Due to the similarity between English and French month words, the major task of the word recognizer is to find out which month is written, instead of how it is written. In other words, the word recognizer is considered to have provided correct information if it outputs "September" when the exact month word written is "Septembre".

However, when the recognition results are put together, they may not represent a valid date. For example, the recognition result for numeric subimage may contain more than two digits, whereas such field can contain only one or two digits. This is due to the fact that the digit recognizer used to process numeric date field is actually developed for courtesy amount recognition, in which case the number of output digits is not restricted at all. Therefore, it is necessary to send all the recognition results to a parser for further validation and interpretation. Thus, invalid results such as "January 32, 1997", "June 31, 1997", "September 3, 19011", etc. can be rejected automatically.

4.3.1 Result on *CENPARMI cheques*

Table 9 shows the performance of the entire date processing system on 4564 *date_zone* images extracted from *CENPARMI cheques*. The statistics are collected under two different conditions, "Without parsing procedure" and "With parsing procedure". Among the 4564 *CENPARMI cheques*, a significant amount, i.e. 12.78%, are rejected by the parser because the results are not valid dates. In most cases, invalid dates are produced by the date processing system. For instance, when *date_zone* is incorrectly segmented, a *Month* word subimage might be sent to the digit recognizer. By introducing the parser, some of the incorrect classifications become rejections and hence the error rate is reduced (refer to Table 9 for details). However, users also make mistakes. For example, "fév 30 1994", "fév 31 1994" and "31 nov 1993" are observed in the database. There are altogether 5 such invalid images and they can be identified by the parser. This explains why there is rejection even though all the three fields are correctly recognized.

Table 9: Performance of date processing on *CENPARMI cheques*

	Correct (%)	Error (%)	Reject (%)
Without parsing procedure	21.98	70.20	7.82
With parsing procedure	21.87	57.53	20.60

Three types of statistics, collected in the case where parsing procedure is not involved, are presented in Table 10. "Completely correct" means the result of date processing is exactly the same as what is written in the *date_zone*. "Any 2 fields correct" means two fields out of three (could be *Day* and *Month*, or *Day* and *Year*, or *Month* and *Year*) match the *date_zone*. "Any 1 field correct" means only one field, *Day* or *Month* or *Year*, matches its corresponding field in the *date_zone*.

Table 10: Performance of field processing on *CENPARMI cheques*

	Completely correct (%)	Any 2 fields correct (%)	Any 1 field correct (%)
Without parsing procedure	21.98	54.65	81.88

It is worth noting that in general, the processing of *Year* field presents better result than that of *Month* and *Day* due to the fact that *Year* is better segmented.

4.3.2 Result on *real cheques*

The system was further tested on the *real cheque* database which is composed of one *Adjustment set* and one *Testing set*, each consisting of 494 and 391 images respectively. Tables 11 and 12 present the performance of *date_zone* processing on these two sets, while Table 13 shows the same group of statistics as collected in Table 10. The results on the *Testing set* was obtained through a single execution.

4.3.3 Performance analysis of date processing

Automatic date processing is a very challenging problem, especially when processing real bank cheques. Three major steps, background removal and *date_zone* extraction, *date_zone* segmentation, and field recognition (including cursive word and digit

Table 11: Performance of date processing on *Adjustment set*

	Correct (%)	Error (%)	Reject (%)
Without parsing procedure	14.98	73.68	11.34
With parsing procedure	14.98	52.23	32.79

Table 12: Performance of date processing on *Testing set*

	Correct (%)	Error (%)	Reject (%)
Without parsing procedure	12.53	75.19	12.28
With parsing procedure	12.53	58.31	29.16

recognition), are involved. The performance of each step is dependent on that of the previous steps. However, errors could be made in every step, thus severely affecting the following steps. Hence, it is fair to say that it is very difficult to obtain a perfect result when processing handwritten dates.

The great difficulties involved in removing background and extracting handwritten information have been described in Section 4.1.2. It should be reiterated here that the two recognizers utilized in date recognition were actually developed for applications other than handwritten date processing.

The digit recognizer was developed for processing the courtesy amount written on bank cheques. Unlike *Day*, *Year* or *Month* (when written in numerals) each of which contains maximum two digits, the courtesy amount can be any number. Therefore, the digit recognizer does not place proper restrictions on the number of output digits or the range of output number (since *Day* should never exceed 31, and *Month* should never exceed 12).

The cursive word recognizer was designed for processing legal amount handwritten

Table 13: Performance of field processing on *real cheques*

	Completely correct (%)	Any 2 fields correct (%)	Any 1 field correct (%)
Adjustment set	14.98	36.24	69.03
Testing set	12.53	41.94	69.82

on bank cheques. It is a nearest neighbour recognizer which highly relies on features such as ascenders and descenders [GS94]. However, among the 42 classes of English and French month words, a few of them are quite short which produce less features compared with long words. Different month words may exhibit similar ascender and descender features, for example "Jan", "Avr" and "Dec". In addition, many people tend to "print" abbreviated month words, in which case the ascender and descender information is totally lost. All these make it the more difficult for the legal amount recognizer to achieve a better performance on month words.

The performance of the two recognizers on *CENPARMI cheque* database is shown in Table 14. This set of statistics is collected under the condition that the subimages sent to each recognizer are correctly segmented and hypothesized.

Table 14: Performance of the recognizers on *CENPARMI cheques*

	Untrained Digit recognizer	Untrained Word recognizer
Correct (%)	74.10	48.89
Error (%)	25.90	51.11
Total	100% 7303 numbers	100% 2855 words

Chapter 5

Conclusions

5.1 Contributions

This thesis proposed a method of automatically processing date information handwritten on standard bank cheques. This method is based on *date_zone* image segmentation, and includes digit recognition as well as cursive month word recognition. It was also implemented into an automatic date processing system. The system was further tested on real bank cheques, under the conditions that no restrictions were imposed on the writing of the cheques, no *a priori* knowledge was available about the written date, and only a single execution was allowed. It is worth mentioning that during the research and implementation phase of this work, there was no literature published on the same topic. Being the first handwritten date processing system developed and published, it proved to be feasible and efficient in tackling this unsolved OCR problem. Its potential of success upon further research and development is also demonstrated.

A date segmentation system is studied in detail. Features describing both *shape* and *spatial* aspects of any given connected component have been developed and analyzed in order to detect possible segmentation points on date images. A set of heuristic rules have been developed to finalize segmentation, and also make assumptions about the *date_zone* writing style.

A date image truthing tool is designed and implemented as well in this research work. This application is built on top of the MOTIF environment. It is completely

mouse-driven, which makes the tedious job easy and effective. Different colors are used throughout the tagging process to reduce human error rate. With minor modifications, this user friendly interface can be extended to other image processing applications for different purposes.

5.2 Strengths and Weaknesses

The date image processing system described in this thesis is not restricted only to the English/French handwritten date application. With a limited amount of extra training, it can be extended to process any date information provided that it has a similar layout.

For such a complicated problem involving both cursive words and numerals, the method proposed here is rather simple and efficient. It breaks down one problem into several sub-problems, and also makes it possible to apply mature algorithms for cursive word and connected digit recognition. This could lead to a significant impact on the research progress.

However, the interaction between segmentation and recognition has not been properly established. It lacks a proper way of evaluating different segmentation strategies, e.g. if a punctuation is detected, segmentation should be done based on the location and type of punctuation or by ignoring the punctuation (i.e. cut by interword gap). When the segmenter makes a mistake in either date image segmentation or writing style assumption, the recognizers cannot indicate with sufficient confidence that the segmentation result is incorrect.

5.3 Future Work

In an ideal model for date image processing, the segmentation system and the recognizers should better "coordinate" with each other to produce a final result. There should be alternative strategies provided by the segmentation system, in case one or even more strategies could not produce recognition results with high confidence. Recognizers should be able to provide feedback to the segmentation system, e.g. the digit recognizer should be able to reject letters or cursive words, and the cursive word

recognizer be able to reject numerals.

Recognizers used in this research work were developed for processing legal and courtesy amounts handwritten on cheques, i.e. applications other than date processing. Different approaches, such as Hidden Markov Model [GS97] and human reading model [CCLS97], have been studied and tested on recognizing legal amount. Eventually, recognizers intended specifically for date processing should be developed and trained using a large and comprehensive database. This is especially useful for word recognizers since over 80% of the general public write the months in words.

In terms of *date_zone* segmentation, correct detection of the French word "Le"/"le" is definitely a problem that needs to be solved, especially if the application is aimed at a French population. The confusions in punctuations should also be investigated, since the precise location of punctuations is a great asset to date image segmentation as well as date processing.

Bibliography

[Bar96] N. Bartneck. The role of handwriting recognition in future reading system. In *Proc. of the Fifth International Workshop on Frontiers in Handwriting Recognition*, pages 147–163, Essex, England, September 1996.

[Cas94] Sean Casey. Checks still america's favorite payment method. *Item Processing Report*, page 7, December 1994.

[CCLS97] M. Côté, M. Cheriet, E. Lecolinet, and C.Y. Suen. Automatic reading of cursive scripts using human knowledge. In *Proc. of the fourth Internation Conference on Document Analysis and Recognition*, pages 107–111, Ulm, Germany, August 1997.

[CHS94] E. Cohen, J.J. Hull, and S. N. Srihari. Control structure for interpreting handwritten addresses. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 16(10):1049–1055, Oct 1994.

[DGI+96] G. Dimauro, A.R. Galiano, S. Impedovo, I. Pansino, G. Pirlo, and A. Salzo. Bankcheck recognition systems: Re-engineering the design process. In *Proc. of the Fifth International Workshop on Frontiers in Handwriting Recognition*, pages 329–334, Essex, England, September 1996.

[FLS95] R. Fan, L. Lam, and C.Y. Suen. Reading and recognition of dates on bank cheques. Technical report, CENPARMI, Concordia University, Montréal, Québec, Canada, July 1995.

[FLS96] R. Fan, L. Lam, and C.Y. Suen. Processing of date information on cheques. In *Proc. of the Fifth International Workshop on Frontiers in Handwriting Recognition*, pages 207–212, Essex, England, September 1996.

[GS94] D. Guillevic and C.Y. Suen. Cursive script recognition: A sentence level recognition scheme. In *Proc. of the Fourth International Workshop on Frontiers in Handwriting Recognition*, pages 216–223, Taipei, Taiwan, December 1994.

[GS95] D. Guillevic and C.Y. Suen. Cursive script recognition applied to the processing of bank cheques. In *Proc. of the third Internation Conference on Document Analysis and Recognition*, pages 11–14, Montréal, Canada, August 1995.

[GS97] D. Guillevic and C.Y. Suen. HMM word recognition engine. In *Proc. of the fourth Internation Conference on Document Analysis and Recognition*, pages 544–547, Ulm, Germany, August 1997.

[Gui95] D. Guillevic. *Unconstrained handwriting recognition applied to the processing of bank cheques.* PhD thesis, Concordia University, Montréal, Québec, Canada, July 1995.

[HAS+96] 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 International Association for Pattern Recognition Workshop on Document Analysis Systems*, pages 495–546, Malvern, Pennsylvania USA, October 1996.

[HF94] D. Heller and P.M. Ferguson. *Motif Programming Manual*, volume Six A. O'Reilly & Associates, Inc., second edition, 1994.

[KBPS96] S. Knerr, O. Baret, D. Price, and J.C. Simon. The A2iA recognition system for handwritten checks. In *Proc. of the International Association for Pattern Recognition Workshop on Document Analysis Systems*, pages 431–494, Malvern, Pennsylvania USA, October 1996.

[LSN96] K. Liu, C.Y. Suen, and C. Nadal. Automatic extraction of items from cheque images for payment recognition. In *Proc. of the Internation Conference on Pattern Recognition*, pages 798–802, Vienna, Austria, August 1996.

[Mil96] U. Miletzki. Documents on the move: DA&IR-driven mail piece processing today and tomorrow. In *Proc. of the International Association for Pattern Recognition Workshop on Document Analysis Systems*, pages 547–563, Malvern, Pennsylvania USA, October 1996.

[SC94] G. Seni and E. Cohen. External word segmentation of off-line handwritten text lines. *Pattern Recognition*, 27(1):41–52, 1994.

[SLG+96] C.Y. Suen, L. Lam, D. Guillevic, N.W. Strathy, M. Cheriet, J.N. Said, and R. Fan. Bank check processing system. *International Journal of Imaging Systems and Technology*, 7:392–403, 1996.

[Sri92] S.N. Srihari. High-performance reading machines. *Proceedings of the IEEE*, 80(7):1120–1132, July 1992.

[Sri96] S.N. Srihari. Recent advances in off-line handwriting recognition at CEDAR. In *Proc. of the Fifth International Workshop on Frontiers in Handwriting Recognition*, pages 1–15, Essex, England, September 1996.

[Str93] N.W. Strathy. A method for segmentation of touching handwritten numerals. Master's thesis, Concordia University, Montréal, Québec, Canada, September 1993.