PERGAMON

Pattern Recognition 32 (1999) 305–316

PATTERN RECOGNITION
THE JOURNAL OF THE PATTERN RECOGNITION SOCIETY

A knowledge-based segmentation algorithm for enhanced recognition of handwritten courtesy amounts

Karim M. Hussein[a,]*, Arun Agarwal[b], Amar Gupta[a], Patrick S.P. Wang[c]

[a] *Sloan School of Management, Massachusetts Institute of Technology, Cambridge, MA 02139, U.S.A.*
[b] *Department of Computer Science, University of Hyderabad, Hyderabad, India*
[c] *College of Computer Science, Northeastern University, Boston, MA, U.S.A.*

Received 13 January 1998

Abstract

A knowledge-based segmentation algorithm to enhance recognition of courtesy amounts on bank checks is proposed in this paper. This algorithm uses multiple contextual cues to enhance segmentation and recognition. The system described extracts context from the handwritten numerals and uses a syntax parser based on a *deterministic finite automaton* to provide adequate feedback to enhance recognition. Further feedback is provided by a simple legal amount decoder that determines word count and recognizes several key words (e.g. thousand and hundred). This provides an additional semantic constraint on the dollar section. The segmentation analysis module presented is capable of handling a number of commonly used styles for courtesy amount representation. Both handwritten and machine written courtesy and legal amounts were utilized to test the efficacy of the preprocessor for the check recognition system described in this paper. The substitution error was reduced by 30–40% depending on the input check mix. © 1999 Pattern Recognition Society. Published by Elsevier Science Ltd. All rights reserved.

Keywords: Automata; Check processing; Character recognition; Classification; Knowledge-based; Parser; Segmentation; Syntactic

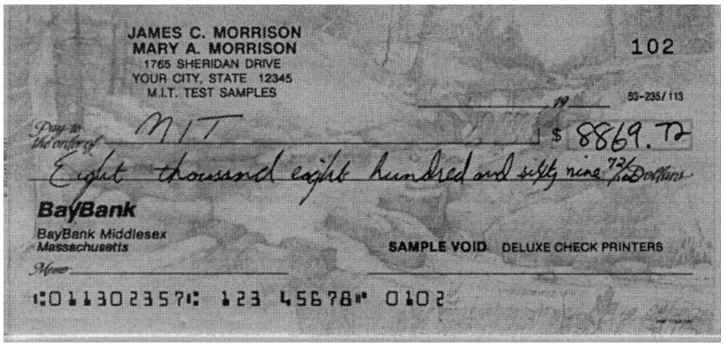
1. Introduction

Approximately 61 billion checks are written in the United States each year, making processing expenses of these checks a significant financial burden on banking institutions. Image processing technology offers the potential for significantly reducing costs involved in processing checks by banks [1]. Traditional check processing involves a series of manual processing steps. When a check shown in Fig. 1 is deposited for credit into one's account, the depositor's bank is interested in two numerical fields: the account number, which is already written in MICR ink and can be handled using automated techniques with near-perfect accuracy, and the amount of the check, which is currently read and keyed in by a human operator.

The other fields, such as the name of the recipient, the date, and the signature on the check are largely ignored in routine check processing, unless the transaction is contested or the check is presented for immediate encashment to a bank employee.

The amount of the check is written in two ways: in textual format and in numerical format. The textual format, called the *legal amount*, was originally intended to be the version used for all transaction-related

* Corresponding author.



JAMES C. MORRISON
MARY A. MORRISON
1765 SHERIDAN DRIVE
YOUR CITY, STATE 12345
M.I.T. TEST SAMPLES

102

19 53-235/113

Pay to the order of MIT $ 8869.72

Eight thousand eight hundred and sixty nine 72/100 Dollars

BayBank
BayBank Middlesex
Massachusetts

SAMPLE VOID DELUXE CHECK PRINTERS

Memo

:011302357: 123 45678" 0102

Fig. 1. A typical bank check.

purposes. The numerical format, called the *courtesy amount*, is the version currently utilized by human operators to key in the amount that appears at the bottom of cancelled checks.

Automated reading of numerical fields has been attempted for a number of application areas. One such area is the reading of postal zipcodes in the addresses written or typed on letters [2–7]. Reading courtesy amounts is more difficult than reading zipcodes due to a number of key differences in the nature of the handwritten material. First, the number of digits in zipcodes is fixed and known a priori, which is not true for courtesy amounts. Second, unlike the case of zipcodes, the courtesy amount consists of two components: the non-fractional component (the dollar portion) and the fractional component (the cents portion). The courtesy component is written in more than a dozen different styles in the U.S., and some of these styles are extremely difficult to handle via automated techniques. Fig. 2 shows possible styles. However, bank checks have a redundancy in the representation of the check amount (i.e. check amount is represented in numerals as well as in words – the legal amount) which can significantly aid the segmentation and recognition processes.

The focus of this work is to improve the segmentation accuracy of digits by developing a knowledge-based segmentation analysis mechanism for automated reading of the courtesy amount. Our new algorithm extracts the context available from the typical styles shown in Fig. 2 to aid the segmentation and recognition subsystems within the overall check processing system. The system also takes advantage of the redundancy in checks provided by the handwritten legal amount. This paper is comprised of six sections. Section 2 describes the overall check recognition system architecture. Section 3 concentrates on theory and prior work that served as the foundation for our research. Section 4 focuses on the details of the knowledge-based segmentation analysis module and section 5 presents results. Section 6 contains the conclusions and discusses areas for future work.

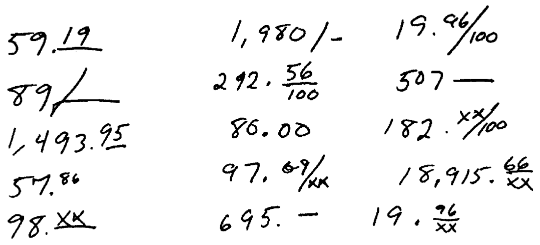


Fig. 2. Possible styles for writing courtesy dollar amount.

2. Check recognition system architecture

Automated recognition can be performed in two environments: on-line and off-line. In the on-line case, recognition is performed as the characters are being written and hence dynamic information like stroke sequence, speed, pressure, and pen-up and/or pen-down positions can be utilized to overcome some of the difficulties created by the non-uniformity and connectivity of unconstrained handwriting [8]. In off-line recognition, only static information contained in the image of the numeral strings is available, which makes this enviroment more difficult to tackle. This paper deals with off-line recognition of check courtesy amounts.

The architecture of the prototype system for reading handwritten numerals on checks, shown in Fig. 3, consists of six modules and is discussed below.

Image handler: Image enhancement and information location indentification operations are conducted by this module. A courtesy amount block locator algorithm [9] locates and extracts the position of the handwritten amount for recognition by the system. The background of the check (US checks sometimes have colorful and ornate backgrounds) is removed using a dynamic thresholding algorithm. Finally, several noise reduction filters are employed to enhance image quality.

Segmentation module: The image of the courtesy amount and the legal amount is split into component primitives by the segmentation module. This module

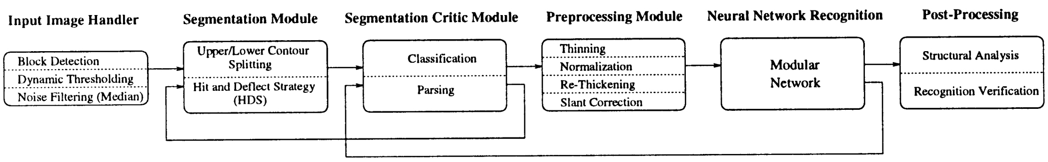


Fig. 3. System architecture.

attempts to segment the courtesy amount into digits and delimiters, and the legal amount into words. Three algorithms are used to segment the handwritten numerals: connected component extraction, upper and lower contour splitting, and the hit and deflect strategy (HDS). See Refs. [10,11] for a complete description of these algorithms. Section 4.3 describes the word segmentation of the legal amount in detail.

Segmentation analysis module: This module performs several contextual checks on the scanned numerical representation. The module classifies the digits into primitive sets (including digits, periods and hyphens); then the ordered primitives are analyzed for syntactic correctness. A legal amount estimator unit decodes the handwritten word representation to determine the number of digits in the numerical string. The parser output and the extimator output are decoded by an evaluation unit that determines the validity of the segmentation and resegments if necessary. Section 4 provides a complete description of this module.

Preprocessing: Additional image enhancement process are carried out after segmentation. These include slant reduction, uniform line thickness adjustment, as well as normalization of the component image. These algorithms are described in detail in Refs. [12–14].

Neural-network-based recognizer: The components are then classified into numerals by an adaptive modular neural network [15,16].

Postprocessing: This final stage of the recognition system verifies the recognition process by performing a detailed analysis of some digits that are commonly misclassified. This module deals particularly with the '5', '8', '1' and '7' digits. It employs structural analysis techniques to refine the pattern classification process as described in Ref. [17].

All the modules, except the segmentation analysis module, were developed previously [18]. This paper focuses on the segmentation analysis module which deals with enhancing the ability to separate individual characters from a handwritten string.

The granularity of segmentation determines the primitives produced and the subsequent ability to recognize the individual characters. A given string of connected characters, can be separated using two basic approaches: according to individual characters or according to the strokes that make up these characters. Our approach has been to use a single numeral or delimiter as our base primitive. Algorithms to perform the segmentation are based on Sparks et al. [10]

Since the correct segmentation cannot be guaranteed using these bottom-up techniques, we have encoded our knowledge of courtesy amount representation styles. The encoded styles provide additional information concerning the interplay between the primitive classes and the correlations between them. In previous work on segmenting numerals, especially work done on postal zip code recognition, the number of digits in a string was known *a priori* thereby simplifying the segmentation task. Our algorithm aims to provide a pattern-directed feedback mechanism to improve our segmentor. We have now used a syntax-based classification approach [19], which permits a large set of complex patterns to be described using small sets of simple pattern primitives and grammatical rules, and successive portions of the input pattern to be classified on a recursive basis. This approach starts with the entire string of input characters and attempts to partition the problem into subgoals and proceeds until the last subgoal has been either attained or exhausted.

This paradigm has been incorporated in the segmentation analysis described to improve the segmentation process by providing feedback, to the appropriate module regarding erroneous segmentation of the components.

3. Segmentation framework

The human reading model is highly adaptive and flexible, and the understanding of words is highly dependent on the reader's current purpose and context. Word and numeral recognition in humans involves information analysis on multiple planes including graphic orthographic phonetic, semantic, morphological and syntactic. Several methods have been adopted to simulate human reading in computer systems. These approaches have included syntactic and structural analysis mechanisms, as well as various forms of sematic matching processes.

Syntactic and structural approaches for pattern recognition are more popular than conventional statistical methods because they are hierarchical in nature. Moreover, using such syntactic and structural approaches, one can directly take advantage of powerful data structures using grammatical rules, trees, and directed labeled graphs, which are widely used in dealing with linguistic problems [19–21]. Further, it has been shown that syntactic and structural approaches can overcome some disadvantages found in the classic decision-theoretic (statistical) approach, which has difficulty in distinguishing between two very similar patterns (characters).

A critical assumption in the pattern recognition techniques described above is that one has provided the specific region or primitive to be recognized. This issue of segmentation or *chunking*, which involves separation of an object from a particular scene or a character from a handwritten string is a non-trivial task. Deciding granularity of segmentation will determine the primitives produced. For example, a given string of connected characters, can be separated using two basic approaches: according to individual characters or according to the strokes that make up these characters. Our approach has been to use a single numeral or delimiter as our base primitive. Several dissection algorithms have been developed to perform this segmentation which are described in Sparks et al. [10]. However, the correct segmentation cannot be guaranteed using these bottom-up techniques.

The process of such decomposition (bottom-up) discussed above is inherently narrowly focused. The segmentation and classification processes reach a point where *one cannot see the forest from the trees*. Any successful recognition system must retain a general understanding of the environment within which it operates. Such an environment or context provides additional information concerning the interplay between the classes and the correlations between them. Our aim was to provide such a pattern-directed feedback mechanism to improve our segmentor. In previous work on segmenting numerals, especially work done on postal zip code recognition, the number of digits in a string was known beforehand, thereby simplifying the segmentation task. Our algorithm aims to provide an analysis of primitives within a context that has been predetermined (i.e., courtesy amount styles). In reading the courtesy amount on checks, we can only use the fact that certain delimiters are used in particular styles to separate the cents from dollars. This provides a constraint on the cents portion, ie. an indication of the length of the string. The *comma* delimiter also provides an indication of how many numerals are in the dollar amount. These are the constraints that are encoded by our segmentation analysis module. However, a major difficulty is that the most common delimiters, the comma and period, could also be noise. Therefore, initial noise filtering is critical for the proper functioning of our segmentation analysis module.

We have used a syntax-based classification approach [22], which permits a large set of complex patterns to be described using small sets of simple pattern primitives and grammatical rules, and successive portions of the input pattern to be classified on a recursive basis. This approach starts with the entire string of input characters and attempts to partition the problem into subgoals (and corresponding subsets of characters) and proceeds until either the last subgoal has been attained or exhausted. This provides a natural way of hypothesizing the global properties of a configuration at an early stage of the recognition procedure.

We have developed and incorporated this paradigm into the segmentation analysis process described below to improve the segmentation process by providing feedback, based on context, to the module regarding errant segmentation of the components.

4. Segmentation analysis module

The segmentation process delineated in Section 2 splits the input binary image of the courtesy amount into a set of binary images that are probable primitives. The module described in this section verifies the segmentation process using a knowledge base of courtesy amount syntax as well as information regarding the number of digits from the preliminary recognition of the legal amount. The segmentation analysis module is comprised of four parts: a classifier, a syntax parser, a legal amount estimator and an evaluator (see Fig. 4). The classifier unit matches coarsely classifies the input components into one of seven predefined primitives (these include digits and delimiters such as comma, period and dash–see Fig. 5). The syntax parsing unit encodes the syntax of the typical styles for numeral representation shown in Fig. 2. The legal amount estimator determines the number of digits in the courtesy amount through a coarse segmentation and recognition of the numerical value of the check in words. Finally, the evaluator unit takes input from the syntax parser and legal amount estimator units and determines whether the segmentation was valid. If the segmentation is invalid, the evaluator directs the segmentation module and the classifier unit with the appropriate action to be taken.

4.1. Classifier unit

The segmentation module described in Section 2 uses three different algorithms to attempt an initial segmentation of the numerical string. These algorithms are applicable in different connectivity situations. The components generated should represent one of the primitives in Fig. 5. The component attributes that are encoded collectively constitute the mean bounding rectangle (MBR) represented by x_{min}, x_{max}, y_{min} and y_{max}. Scale and positional information are not absolute since the syntactic relationship is based on the relative positions of the components. The following heuristic features are derived from the

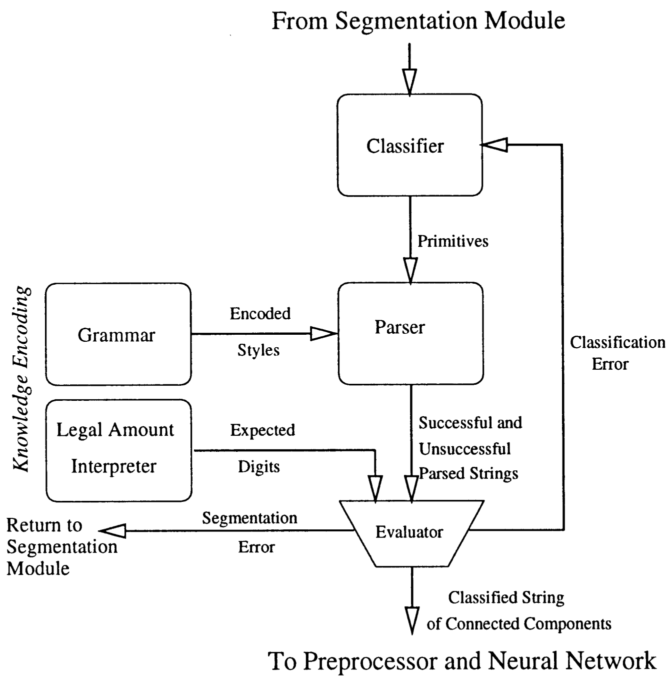


Fig. 4. A block diagram of the segmentation analysis module.

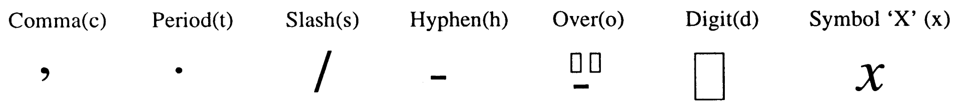


Fig. 5. Permitted component primitives and their respective classes.

component image and its MBR:

$$\text{Standard height (SH)} = \frac{1}{n}\sum_{i=1}^{n}(y_{i_{max}} - y_{i_{min}}),$$

$$\text{Midline (M)} = \frac{SH}{2},$$

$$\text{Midline offset (MO)} = \frac{y_{min} - M}{SH},$$

$$\text{Aspect ratio (AR)} = \frac{y_{max} - y_{min}}{x_{max} - x_{min}},$$

$$\text{Relative height (RH)} = \frac{y_{max} - y_{min}}{SH},$$

Principal axis width (PA) = width of axis of least inertia (see Ref. [23]),

$$\text{Above (AB)} = \frac{x_{(i-1)_{max}} - x_{i_{min}}}{SH},$$

$$\text{Span (SP)} = \frac{y_{i_{max}} - y_{i_{min}}}{y_{(i-1)_{max}} - y_{(i-1)_{min}}}.$$

Given these features, the components are coarsely classified using the algorithm shown in Fig. 6.

As shown in the algorithm, each heuristic feature has a threshold value that breaks reduces the classification set. These thresholds are indicated by a letter "T" with a subscript showing the corresponding heuristic feature used. The "x" primitive is determined through a specialized backpropogation neural network that returns a confidence value that must be greater than T_X.

The thresholds indicated in the classification algorithm have a range of limits. In the preliminary implementation, there are two values for each threshold: a constained value and a relaxed value. Initial classification is performed with the constrained values. These thresholds are relaxed depending on the feedback from the evaluator module.

4.2. Syntax parsing unit

Multiple techniques have been explored in the literature for the storage of context in pattern recognition applications [24]. The key criteria for the choice of contextual encoding are the storage size and the speed

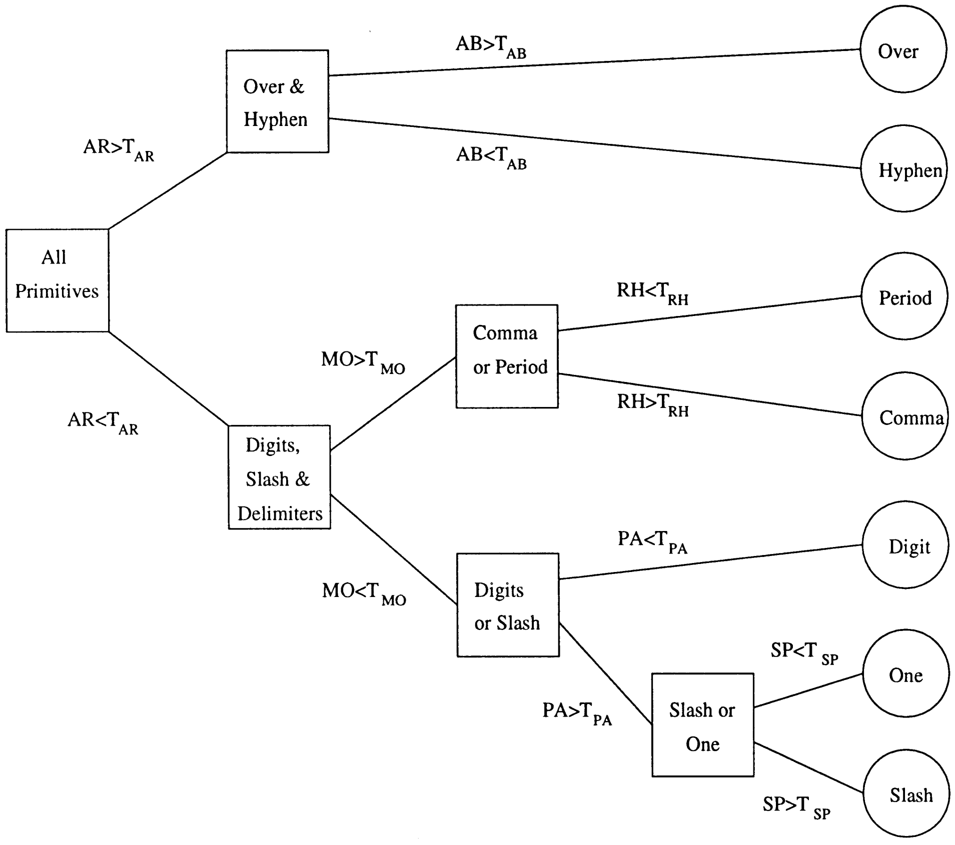


Fig. 6. Classification algorithm.

of retrieval of contextual knowledge. An exhaustive dictionary of possible primitive combinations is prohibitively inefficient for the storage of numeral syntax. Hence, we have used a compact representation based on a finite-state automaton (FSA). This representation provides quick retrieval with a simple matching structure. However, finite-state automata are not easily extensible and can entail a lengthy encoding process. In this particular application, the syntax was relatively simple and fixed making FSAs the ideal contextual repository.

The sequence classified components generated by the classifier unit are validated by syntax parser. The parser based on a deterministic finite automaton (DFA) can be defined by the quintuple $M = (V, \Sigma, \delta, s, F)$ where

- V is the finite set of states of the DFA,
- Σ is the language utilized by the DFA,
- δ is the state transition function from $V \times \Sigma$ to V,
- s is the set of start states of the DFA,
- F is the set of final states of the DFA.

The language Σ is defined by an *alphabet* that is composed of the set of primitives (digit, comma, period, slash, over, hyphen, and x) that are contained in the string of connected components within the input. The DFA inforces ordering constraints on the string to eliminate arbitrary sets of strings, thereby defining a language. V in this application corresponds to the states q_0-q_{19} and q_F and s, the start state is q_0. The final state is q_F.

The state transition matrix (δ) for the DFA is shown in Table 1. This defines the rules by which the automaton M chooses the next state given an input state and an input character from Σ. Hence, if M is in state $q \in V$ and the symbol read from the input string is $\sigma \in \Sigma$, then $\delta(q, \sigma) \in V$ is the uniquely determined state to which M passes. The DFA was further optimized using the algorithm in Aho et al. [25].

The language encoded in the state transition matrix for the DFA indicates two major contextual constraints, the number of numerals in the dollar segment and the style and length of the cents segment. The first nine states (q_0–q_8) provide the dollar constraint by ensuring that there are only three digits after a comma delimeter. The remaining states encode the 15 different cents encoding styles shown in Fig. 2. The transition from the dollar encoding states (q_0–q_8) to the initial cents states (q_9–q_{14}) occurs when a period, slash, over or hyphen delimiter is

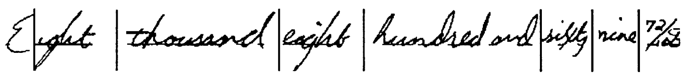


Fig. 7. White space segmentation.

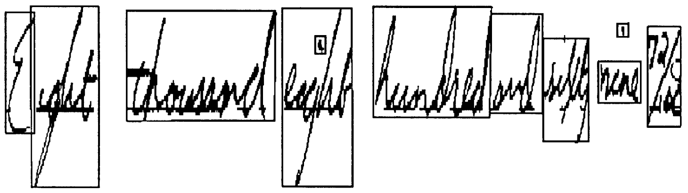


Fig. 8. Connected component extraction.

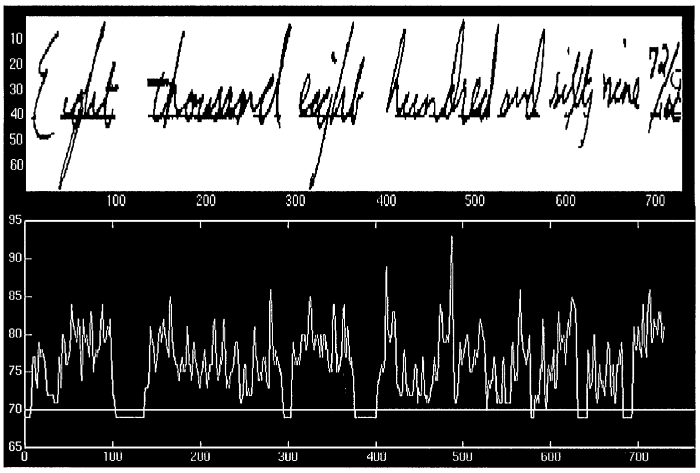


Fig. 9. Minima analysis.

encountered. The remaining states constrain the length and style of the cents portion. From the above discussion it is clear that the available context for the cents portion is richer than the dollar section thereby suggesting that errors in cent segmentation are more easily detected than errors in dollar segmentation. This is due to the lack of syntactic constraints on the dollar section. This defficiency is addressed by the legal amount estimator described in Section 4.3 which provides some additional semantic content.

If the DFA syntax parser reaches a terminal state for a given input string, the parsing is successful, otherwise the input string does not match the courtesy amount syntax. The output is then passed to the evaluator unit for verification of string length using the legal amount estimator and the possible reclassification or resegmentation due to string length or parsing errors.

4.3. Legal amount estimator module

The legal amount (the courtesy amount in words) is used as an additional semantic constraint on the length of the dollar section in the numerical segmentation system. The amount in words is segmented using a three phase approach. A simple wholistic word recognizer is used to distinguish the words in the limited vocabulary of the legal amount. The output is used by the Evaluator unit to determine the validity of the preliminary segmentation and classification of the numerical string.

The first phase of the word segmentation process involves a linear density measurement of the legal amount. The density is the horizontal projection of the words in the legal amount. Segmentation points are chosen based on the regions with zero linear density in the image. Since many words do not contain clearly separated features, this phase is not sufficient for the

Table 1
Parser state transition matrix (S indicates intermediate state; B indicates an intermediate state that can be a final state and F indicates a final state)

		Input symbol						
State	Type	d	c	t	o	x	h	s
Q0	S	Q1		Q19			Q19	
Q1	B	Q2	Q3	Q11		Q12	QF	Q9
Q2	B	Q2	Q3	Q11	Q14	Q12	QF	Q10
Q3	S	Q4						
Q4	S	Q5						
Q5	S	Q6						
Q6	B	Q7	Q3	Q11		Q12	QF	Q9
Q7	S	Q8						
Q8	S	Q6			Q14			Q14
Q9	S						QF	
Q10	S	Q15				Q17	QF	
Q11	B	Q18		Q11		Q12	QF	Q9
Q12	S					Q13		
Q13	B				Q14			Q14
Q14	B	Q15				Q17		
Q15	S	Q16						
Q16	S	QF						
Q17	S					QF		
Q18	S	Q13						
Q19	S	Q18		Q19				
QF	F							

segmentation of the many overlapping sequences of words on bank checks.

The second phase of segmentation extracts connected components from the binary image. This is a simple recursive fill algorithm that extracts all components that are connected using an 8 connectivity grid. An efficient algorithm is used with minimal recursion to limit the need for large memory resources. This method is useful for cursive connected words. A postprocessing step is used to eliminate discontinuities in a single word by calculating the average spacing between characters within a single word and using that as a threshold for word boundaries.

The final phase searches for peaks in the horizontal projection of the binary image and searches for minima within the vertical projection. Segmentation is performed at these minima based on average word length.

After the components are segmented, key words in the legal amount are recognized using a segmentation-free algorithm based on Hidden Markov models described in Ref. [26]. The key words are *million*, *thousand*, and *hundred*. Further classification of the word patterns is performed to separate the nine basic digits (one through nine) from other composite terms that signify double digits (11, 40, 90, etc.). Once these five classes are determined, the number of digits in the courtesy amount can be approximated and fed into the evaluation module for verification of the digit segmentation. For example, the number 8869 contains the words *eight* (1 digit) and *thousand* (3 digits) which results in a total of 4 digits that represent the amount. The counting algorithm is described below:

```
TotalDigits = 0

n = firstword.digits
current_word = first_word
TotalDigits = n

LOOP through n words
  prev_word = current_word
  current_word = next_word

    if(current_word.digits > prev_word.digits)
        TotalDigits = TotalDigits + n

END LOOP
```

4.4. Evaluator unit

The evaluator unit takes the input from the syntax parser and the legal amount estimator to validate the segmentation process. The evaluator also provides the appropriate feedback to the segmentation system if the segmented string is invalid. The evaluator unit provides two forms of feedback to the segmentation process: feedback to the classifier and feedback to the segmentation module. The classifier feedback is used to correct misclassification of primitives due to the similarity between some of the primitives (e.g. the comma and period, and the one and slash). The classifier feedback is provided in the form of the relaxation of the classification threshold. The evaluator provides segmentation feedback by identifying the component that is most likely a candidate for further segmentation. The resegmentation is performed by choosing a finer segmentation algorithm (e.g. from connected component segmentation to hit and deflect segmentation).

The appropriate feedback is based on a heuristic encoding of the most likely causes of failed parses of the connected component string. The knowledge is encoded in the form of rules that are prioritized to provide a deterministic solution to the process. The rules are a subset of all possible failures that were selected based on the frequency of the failure and the relative simplicity in implementation. The rules are encoded in the following grammar:

Given Failed Parse State $\mathscr{R} | \mathscr{R} \in \mathscr{V}$:

if < condition > then < action >

< condition > ⇒ < condition > AND < subcondition > |

$< \text{condition} >$ OR $< \text{subcondition} >$ |
$< \text{subcondition} >$
$< \text{subcondition} > \Rightarrow \text{String}(n) = P$ where $(0 < n < \text{len(String)}$ and $(P \in \Sigma)$.
$< \text{action} > \Rightarrow$ Threshold Adjustment |
Concatenate last two characters |
Adjust segmentation criteria.

The actions taken by the evaluator are in the form of threshold adjustments, character concatenation and segmentation adjustment. Threshold adjustment corresponds to the relaxation of the heuristic thresholds discussed in Section 4.1. Character concatenation involves the "glueing" of two components. This is necessary when portions of the character are detached from the body as is commonly the case with the digit "5". Finally, segmentation adjustment refers to the use of alternate segmentation algorithms to further refine the splitting of a component.

Table 2
Parser reaction table (a small sample)

STATE	Condition	Action
Q0	c	relax T_{RH}
	s	Relax T_{SP}
	x	Constrain T_X
	o	Relax T_{AB}
	o AND T_{AB_R}	Concatenate
Q1	o	Relax T_{AB}
	o AND T_{AB_R}	Concatenate
Q2	NULL	NULL
Q3	s	Relax T_{SP}
	x	Constrain T_X
	h,o	Concatenate
Q4	c,t	Check Segmentation
	End	Check c at Q2 relax T_{RH}
	s	Relax T_{SP}
	x	Constrain T_X
	h,o	Concatenate
.	.	.
.	.	.
Q10	c,t,s	Check s at String(n-1) relax T_{SP}
	o	Relax T_{AB}
	o AND T_{AB_R}	Concatenate
.	.	.
.	.	.
Q18	c,t	Check Segmentation
	End	Check Segmentation
	x	Constrain T_X
	s	Relax T_{SP}
	h,o	Concatenate (if error Check Segmentation)

The conditions and actions encoded are listed in the evaluation table shown in Table 2. The left column corresponds to the condition and the right column corresponds to the appropriate action. The threshold adjustments correspond to the thresholds indicated in Section 4.1.

5. Results

The segmentation analysis system significantly enhanced our overall handwritten numeral recognition system. It successfully flagged most incorrect segmentations in the cents portion. Overall, the segmentation analysis was effective at locating substitution errors (i.e. comma/period or slash/one substitution), and locating segmentation errors when the amount of the dollar section was successfully estimated (either via the legal amount estimator or the detection of a comma delimiter). The analysis system also removed all comma/period and slash/one substitution errors.

The results are based on recognition by a modular neural network developed by our group [15]. For a training set of 5993 digits, the training phase was terminated after a mean square error of 0.25 was reached, with an accuracy of over 99.5% on the training set. We subsequently used a sample of 1000 checks obtained from several sponsor banks. The overall substitution error fell by 30–40% depending on the characteristics of the test batch checks. This result shows that the parser tends to make the segmentation process more informed to get better segments for recognition. The parser algorithm has been very successful at locating the source of the segmentation error (locating 10% of all segmentation errors). Most unflagged segmentation errors had occurred in the dollar portion, while most segmentation errors in the cents portion were flagged.

In order to illustrate the effectiveness of the system, consider the initial scanned input image shown in Fig. 1. After the courtesy amount block is detected using Agarwal et al. [9], the image is dynamically thresholded to seperate the ink object points from the background. The result is shown in Fig. 10. The segmentation process splits the image as shown in Fig. 11. The components are then passed to the classifier unit of the segmentation analysis module resulting in the string *dddtd*. Next the parser generates the following output describing the failure stage:

```
Char => d, State => Q0
Char => d, State => Q1
Char => d, State => Q2
Char => t, State => Q2
Char => d, State => Q11
Char => End, State => Q18
Failed Parse at State Q18 with Input 'End'
```

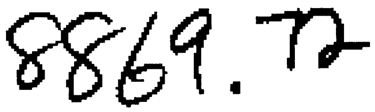


Fig. 10. Thresholded courtesy amount.

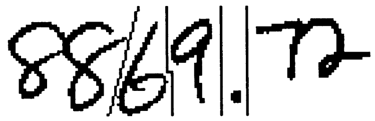


Fig. 11. Initial segmenatation.

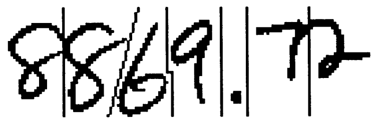


Fig. 12. Final segmenatation.

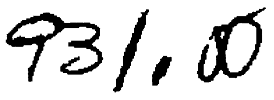


Fig. 13. Thresholded courtesy amount.

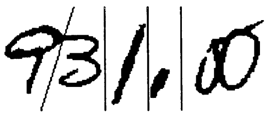


Fig. 14. Initial segmenatation.

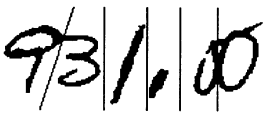


Fig. 15. Final segmenatation.

As described in Table 2 the evaluator then prompts the segmentor to reduce the aspect ratio threshold for the cent portion, resulting in the string *dddtdd*. This is again rejected by the evaluator due to the detection of the words "eight" and "thousand" by the legal amount estimator which indicates that there should be 4 digits in the dollar portion. The aspect ratio threshold is again relaxed for the dollar section and the resulting string is *ddddtdd*. The string of components is then passed to the recognizer resulting in the proper courtesy amount of 8869.72.

Another example courtesy amount is shown in Fig. 13. The initial segmentation results in the image in Fig. 14. The classifier unit generates the following string output *ddscd*. Next the parser generates the following syntax failure:

```
Char => d, State => Q0
Char => d, State => Q1
Char => s, State => Q2
Char => c, State => Q10
Failed Parse at State Q10 with Input "c"
```

As described in Table 2 the evaluator relaxes the spanning threshold T_{SP} resulting in the string *dddcd*. This again fails the parser resulting in a failure at state **Q4**. The evaluater then relaxes the T_{RH} threshold and the comma is transformed to a period. The string *dddtd* again fails the parsing process at state **Q18**. The encoded reaction table then prompts the segmentor to relax its aspect ratio threshold (yielding the string *dddtdd*) resulting in the segmentation shown in Fig. 15. Finally, the correct string of components is generated and passed to neural net recognition module, yielding the correct answer of 931.00.

6. Concluding remarks

One of the most challenging tasks in the creation of a pattern recognition system for unconstrained handwritten material is the segmentation of the input image into meaningful components. The segmentation analysis approach suggested in this paper aids the segmentation process significantly by encoding the context knowledge associated with handwritten courtesy dollar amounts.

The proposed segmentation analysis system is also very efficient and adds minimal overhead to the overall check recognition system. The results discussed in Section 5 show significant improvement in the performance of the system to warrant the limited overhead associated with the analysis module. Further work is being conducted in the word recognition of the legal amount to verify the recognition and segmentation of the handwritten numerical amount on bank checks.

References

[1] S. Helm, Banks check into image processing, Comput. Banking 7 (1990) 25–32.

[2] R. Stern, From intelligent character recognition to intelligent document processing, in Proc. Int. Electronic Imaging Exposition and Conf., Prentice-Hall, Englewood Cliffs, NJ, 1987, pp. 236–245.

[3] E. Cohen, J. Hull, S. Srihari, Understanding handwritten text in a structured environment: determining zip codes from addresses, in: P.S.P. Wang, (Ed.), Character and Handwriting Recognition: Expanding Frontiers, World Scientific, Singapore, 1991, pp. 221–264.

[4] C.Y. Suen, C. Nadal, R. Legault, T.A. Mai, L. Lan, Computer recognition of unconstrained handwritten numerals, Proc. IEEE 80 (1992) 162–1180.

[5] O. Matan, H.S. Baird, J. Bromley, C.J.C. Burges, J.S. Denker, L.D. Jackel, Y. Le Cun, E. Pednault, W. Satterfield, C. Stenard, T. Thompson, Reading handwritten digits: a zip code recognition system. IEEE Comput. 25 (1992) 59–64.

[6] M.A. Mohamed, P. Gader, Handwritten word recognition using segmentation-free hidden Markov modeling and segmentation-based dynamic programming techniques, IEEE Trans. Pattern Anal. Machine Intell. 18(5) (1996) 548–554.

[7] P. Gader, J. Keller, R. Krishanpuram, J. Chiang, M.A. Mohamed, Neural and fuzzy methods in handwriting recognition. IEEE Comput. 30(2) (1997) 79–86.

[8] C.C. Tappert, C.Y. Suen, T. Wakahara, On-line handwriting recognition – a survey, in Proc. Int. Conf. on Pattern Recognition, Rome, 1992, pp. 1123–1132.

[9] A. Agarwal, K. Hussein, A. Gupta, P.S.P. Wang, Detection of courtesy amount block on bank checks, J. Electronic Imaging 5(2) (1996) 214–224.

[10] P. Sparks, M.V. Nagendraprasad, A. Gupta, An algorithm for segmenting handwritten numerical strings. Technical Report IFSRC 214-92, MIT Sloan School of Management, 1992.

[11] H. Baird, H. Bunke, P.S.P. Wang, Document Analysis and Recognition, World Scientific, Singapore, 1994.

[12] L. Lam, S.W. Lee, C.Y. Suen, Thinning methodologies — a comprehensive survey, IEEE Trans. Pattern Anal. Machine Intell. 14(9) (1992) 868–885.

[13] C.Y. Suen, P.S.P. Wang (Eds), Thinning Methodologies for Pattern Recognition. World Scientific, Singapore, 1994.

[14] M.V. Nagendraprasad, P.S.P. Wang, A. Gupta, Algorithms for thinning and rethickening digital patterns, J. Digital Signal Process. 3(2) (1993) 97–102.

[15] L. Mui, A. Agarwal, A. Gupta, P.S.P. Wang, An adaptive modular neural network with application to unconstrained character recognition, Int. J. Pattern Recognition Artificial Intell. 8(5) (1994).

[16] I. Guyon, P.S.P. Wang (eds), Advances in Pattern Recognition Systems Using Neural Network Technologies. World Scientific, Singapore, 1994.

[17] M. Duggan, Enhancing accuracy of automated numerical recognition. S.b. Thesis, Massachusetts Institute of Technology, 1992.

[18] A. Agarwal, A. Gupta, K. Hussein, P.S.P. Wang, Bank check analysis and recognition by computers, in H. Bunke, P.S.P. Wang, (Eds.), Handbook on Optical Character Recognition and Document Image Analysis, World Scientific, Singapore, 1997, pp. 623–651.

[19] K.S. Fu, Syntactic Pattern Recognition and Applications. Prentice-Hall, Englewood Cliffs, NJ, 1982.

[20] T. Pavlidis, Structural Pattern Recognition, Springer, New York, 1980.

[21] A. Rosenfeld, Picture Languages: Formal Models for Picture Recognition. Academic Press, New York, 1979.

[22] P. Rothman, Syntactic pattern recognition, AI Expert 7 (1992) 40–51.

[23] B. K. P. Horn, Robot Vision, MIT Press, Cambridge, MA, 1986.

[24] D. Elliman, I. Lancaster, A review of segmentation and contextual analysis techniques for text recognition. Pattern Recognition 23(3/4) (1990) 337–346.

[25] A. Aho, R. Sethi, J. Ullman, Compilers, Principles, Techniques and Tools, Addison-Wesley, Reading, MA, 1986.

[26] M. Mohamed, P. Gader, Handwritten word recognition using segmentation-free Hidden Markov Modelling and segmentation-based dynamic programming techniques, IEEE Trans. Pattern Anal. Machine Intell. 18(5) (1996) 548–554.

[27] R. Casey, E. Lecolinet, A survey of methods and strategies in character segmentation, IEEE Trans. Pattern Anal. Machine Intell. 18(7) (1996) 690–706.

About the Author—KARIM HUSSEIN received his Doctor of Science degree in Information Systems from the Intelligent Engineering System Group at the Massachusetts Institute of Technology (MIT). He holds a dual BS degree in electrical engineering and computer engineering from Carnegie Mellon University, and an MS degree in civil and environmental engineering from MIT. His research interests include digital image processing, machine vision, neural computing, distributed database systems, and computer support for collaborative engineering.

About the Author—ARUN AGARWAL is a reader (associate professor) in computer science at the University of Hyderabad in India. The work in this paper was carried out while he was a visiting scientist at the Sloan School of Management, Massachusetts Institute of Technology. He received his Ph.D in computer science from the Indian Institute of Technology, New Delhi, and a bachelor's of technology in electrical engineering from the same school. He has refereed and attended many conferences in India as well as several international conferences. He has written one book and has published more than 30 technical papers and articles. Dr. Agarwal's research interests include computer vision, pictorial information systems, distributed problem solving, and neural computing.

About the Author—AMAR GUPTA is codirector of the Productiviy from Information Technology (PROFIT) Initiative at the Sloan School of Management, Massachusetts Institute of Technology. He received his Ph.D. in computer science from the Indian Institute of Technology, New Delhi, a master's degree in management from MIT and a B.Tech. degree in electrical engineering from the Indian Institute of Technology, Kanpur. Dr. Gupta is a member of the IEEE. He serves as a consultant to a number of corporations and government agencies on various aspects of computer technology. Dr. Gupta has been active in research in the areas of multiprocessor architectures, distributed homogeneous and heterogeneous databases, expert systems, and information technology. He has written more than 100 technical articles and papers, and has produced eight books in these areas.

About the Author—PATRICK S.P. WANG has been a tenured full professor of computer science at Northeastern University since 1983, a research consultant at the MIT Sloan School since 1989, and an adjunct faculty member of computer science at Harvard University Extension School since 1985. He was also elected Otto-Von-Guericke distinguished guest professor of Magdeburg University of Germany in fall of 1996. He received his Ph.D. degree in computer science from Oregon State University, his MS in ICS degree from Georgia Institute of Technology, his MSEE degree from National Taiwan University and his BSEE degree from National Chiao Tung University. He was on the faculties of the University of Oregon and Boston University, and senior researcher at Southern Bell, GTE Labs, and Wang Labs prior to his present position. He has published more than 100 technical papers and 16 books and has three patents. He has organized numerous international conferences and workshops and served as reviewer for many journals and grant proposals. Prof. Wang is a fellow of IAPR and a senior member of IEEE. He is founding editor in chief of International Jornal of Pattern Recognition and Artificial Intelligence since 1987. Prof. Wang's research interests include software engineering, programming languages, pattern recognition, k artificial intelligence, image technology, and automation.