High-Performance Reading Machines

Sargur N. Srihari
Department of Computer Science
State University of New York at Buffalo
Buffalo, New York 14260-0001

Abstract

The design of reading machines is a specialized computer vision domain that has many practical applications and lends itself to research involving complete high-performance systems. Two major subtasks in reading are the extraction of textual regions of interest, and the reading of printed and handwritten text. Both of these tasks require the exploitation of contextual constraints to overcome ambiguity. Working in this domain, we are exploring a range of issues in building smarter and faster vision systems. This includes computational theories, algorithms and hardware implementations. The paper describes our approach to developing reading machines for the postal domain: for finding address text on mail pieces and reading ZIP codes in handwritten and poorly printed addresses. For each subsystem a computational model is developed to demonstrate competence. The design is then modified to achieve real-time performance.

Keywords: pattern recognition, computer vision, optical character recognition, reading machines.

I. Introduction

Pattern recognition - the attempt to automate understanding of the symbolic content of images or signals - is one of the earliest goals of computer science, predating and prefiguring the recent explosive interest in artificial intelligence and neural networks. In the 1960's and 1970's, the difficulty of the subject stimulated a gradual specialization of research, giving rise to distinct disciplines including image processing, the psychophysics of vision, and computer vision. Of these, only computer vision retained the original goal of automatically producing a complete symbolic interpretation of an image. Research in computer vision itself, in recent years, has undergone further specialization, and has become dominated by work on a few subtopics, such as early vision primitives and surface modeling of depth images. Comparatively little attention is being paid today to architectural studies of specialized but complete high-performance vision systems.

Studying high-performance systems involves taking a vertical slice through several levels of the design hierarchy. For any information processing task, as pointed out by Marr [1], a machine carrying out the task must be understood at three levels: computational theory, representation and algorithms, and hardware implementations. Computational theory pertains to questions such as: what is the goal of the computation, what is the logic of the strategy, and why it is appropriate. The representation and algorithm level answers questions such as: what is the representation for the input and the output, what is the algorithm for the transformation, and whether the performance is satisfactory. The hardware implementation level concerns questions of how the representation and algorithm can be realized physically to achieve the desired speed or throughput. The three levels are interdependent, as the hardware implementation can influence the representation-algorithms level and the computational theory itself.

Issues in high performance pattern recognition system research include robustness, *i.e.*, ability to handle a variety of inputs, performance measured over realistic test sets, and considerations of speed and throughput. All three levels of the Marr hierarchy have to be traversed, possibly several times, before we arrive at a final design.

Our research group at the State University of New York at Buffalo is working on developing high-performance systems for document reading. The domain poses challenging subtasks such as finding regions of interest (*e.g.*, text) in a document, and reading poorly printed and handwritten text while exploiting contextual constraints. Working in this domain, we are exploring a range of issues in building smarter and faster pattern recognition systems.

This includes computational theories, algorithms and hardware implementations.

The two major subsystems in building a machine for performing any given reading task are: finding text to be read and reading the text. In this paper we will describe design approaches to both subsystems in developing high-performance reading systems, specialized to the domain of postal mail sorting. In this case, the subsystems specialize to: (i) finding address blocks on mail pieces, and (ii) determining ZIP codes from handwritten and poor quality machine-printed postal addresses. The first task is one of finding regions of interest in either a structured or cluttered environment; the regions of interest are blocks of machine-printed or handwritten text. The second task involves reading digits and handwritten words and consulting with a ZIP code database so as to determine the five- or nine-digit destination ZIP code. The role of each of these subsystems in a larger address recognition system is indicated in Figure 1. This system is an instance of a more general document image understanding architecture [2].

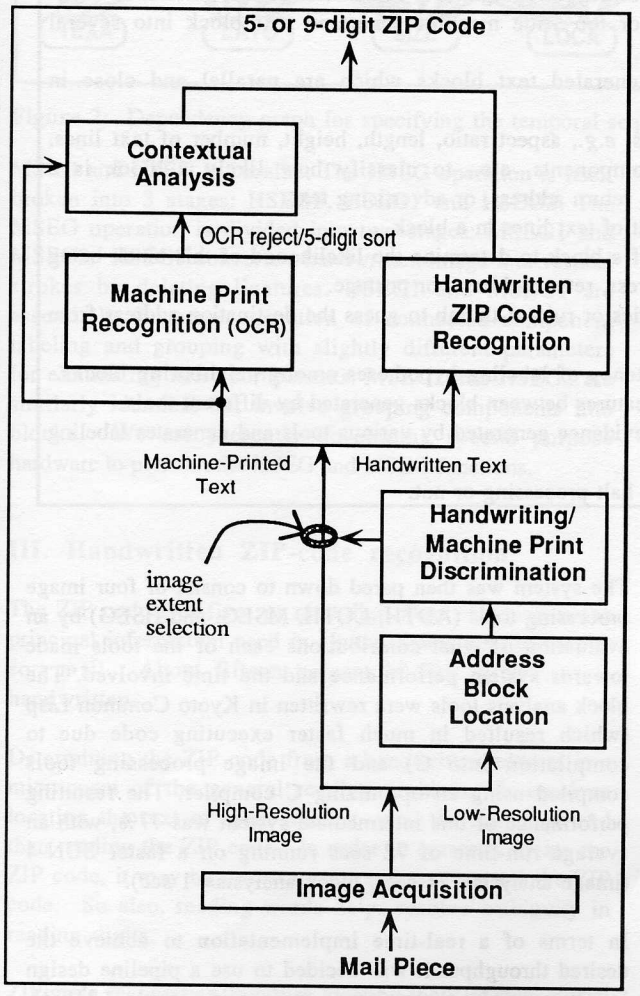

Figure 1: Address Recognition System Architecture.

Section II describes Address Block Location; the handwriting/machine print discrimination function is a byproduct of this operation. Section II describes Handwritten ZIP Code recognition. Section IV describes Contextual Analysis of machine-printed addresses; its goal is to analyze rejects and low-level sorts of a conventional Optical Character Reader (OCR) to improve the level of sortation.

II. Address Block Location

In the address block location (ABL) problem, the input is an image of the face of an arbitrary mail piece. The objective is to produce as output one or more rectangular subregions of the image as candidates for the destination address. This task is to be performed with a low to medium resolution image (100 ppi) so that the actual reading of the hypothesized destination address region is deferred to a later stage. This later stage, which is not part of the address block location task, will use a higher resolution image to attempt to read the address.

A typical mail piece has several regions or blocks that are meaningful to mail processing, *e.g.*, address blocks (destination and return), postage (meter mark or stamp) as well as extraneous blocks such as advertising text, logos and graphics. The significant blocks have certain characteristic properties and spatial relationships that can be derived from images and used to assist in the task.

The computational approach is to develop a set of tools to segment the image and to detect characteristic properties that are relevant to assigning appropriate labels to the segmented regions.

The initial design of the ABL system was to demonstrate system competence in locating address blocks. Once this was done, the goal was to produce a final system design that exhibited competence in terms of throughput (roughly of the order of 12 per second). The initial design was arrived at by examining a large data base of characteristics of mail pieces relevant to address block location. The design uses several specialized tools [3, 4]. The tools were divided into two categories: image analysis tools and block analysis tools (see Table I). The image analysis tools take images as input. Their output consists of either other images (*e.g.*, thresholded image, images of candidate blocks), or extracted image features (*e.g.*, whether printed or handwritten, whether a blob is rectangular). The block analysis tools take symbolic descriptions as input, *e.g.*, extents of different blocks, confidences associated with blocks for different labels, and produce other symbolic descriptions.

The tools are invoked and coordinated using a blackboard system architecture [5]. The blackboard architecture is

Table I: Functional Descriptions of Tools in initial ABL system.

Category	*Tool*	*Description*
Image Analysis Tools	**DIGI**	Produces digitized images (RGB and gray-level) of the original picture.
	ADTH	Adaptive thresholding to convert a gray-level image into a binary image using local contrast.
	COTH	Color thresholding to extract regions of specified color in RGB images.
	SHAP	Measures the degree of rectangularity of a blob.
	MSEG	Bottom-up segmenter to group machine-generated characters into words, lines and blocks.
	HSEG	Bottom-up segmenter to group hand-generated thin strokes into lines and blocks.
	HWDE	Regularity analyzer to detect the likely regions of hand-generated text.
	HWDI	Regularity analyzer for distinguishing machine-generated versus hand-generated address block.
	TEXA	Texture discriminator for address block type characters and non-address block type characters.
	ICDE	Postal icon detector to detect rectangular postal icons.
	UVDE	Postage detector to detect the postage locations (stamp, meter-mark) on UV illuminated image.
Block Analysis Tools	**ZIPM**	Merges ZIP code block in lower right of a destination address candidate.
	BLCS	Splits a too high or too wide machine-generated text block into several smaller text blocks.
	BLCM	Merges machine-generated text blocks which are parallel and close in proximity.
	SIZE	Uses block features, *e.g.*, aspect ratio, length, height, number of text lines, and number of components, etc., to classify how likely a block is a destination address, return address, or advertising text.
	LAYO	Examines the layout of text lines in a block.
	LOCA	Uses the location of a block to determine the likelihoood of this block being the destination address, return address, or postage.
	HEUR	Uses spatial heuristics or rule-of-thumb to guess the destination address from a list of candidates.
	COVF	Verifies the consistency of labeling hypotheses among neighboring blocks.
	UNIF	Unifies the block features between blocks generated by different tools.
	EVHP	Pools together the evidence generated by various tools and generates labeling hypotheses.
	STOP	Decides whether to halt processing or not.

particularly suited to incorporate several tools, or knowledge sources, in an initial design. The order of invokation of the tools is determined by a dependency graph as shown in Figure 2.

The initial system consists of ten image processing tools and eleven block analysis tools. The system was able to correctly locate the address block in 81% of the cases presented; the test set consisted of 174 images, many of which were selected because they were determined by inspection to be cases that are difficult to handle. This system, running on a SUN-3 with 8 MBytes of memory took, on an average, 627 secs per mail piece image. The image analysis tools, written in C, consumed most of the time (543 secs). The block analysis tools, written in LISP (Sun Franzlisp), took the remainder (84 secs).

The system was then pared down to consist of four image processing tools (ADTH, COTH, MSEG, and HSEG) by an evaluation of what contributions each of the tools made towards system performance and the time involved. The block analysis tools were rewritten in Kyoto Common Lisp (which resulted in much faster executing code due to compilation into C) and the image processing tools compiled using an optimizing C compiler. The resulting performance of this intermediate system was 77%, with an average run-time of 72 secs running on a faster SUN-4 (image analysis= 71 secs, block analysis= 1 sec).

In terms of a real-time implementation to achieve the desired throughput it was decided to use a pipeline design involving only three image processing tools (ADTH, MSEG and HSEG). This design feeds the result of ADTH to

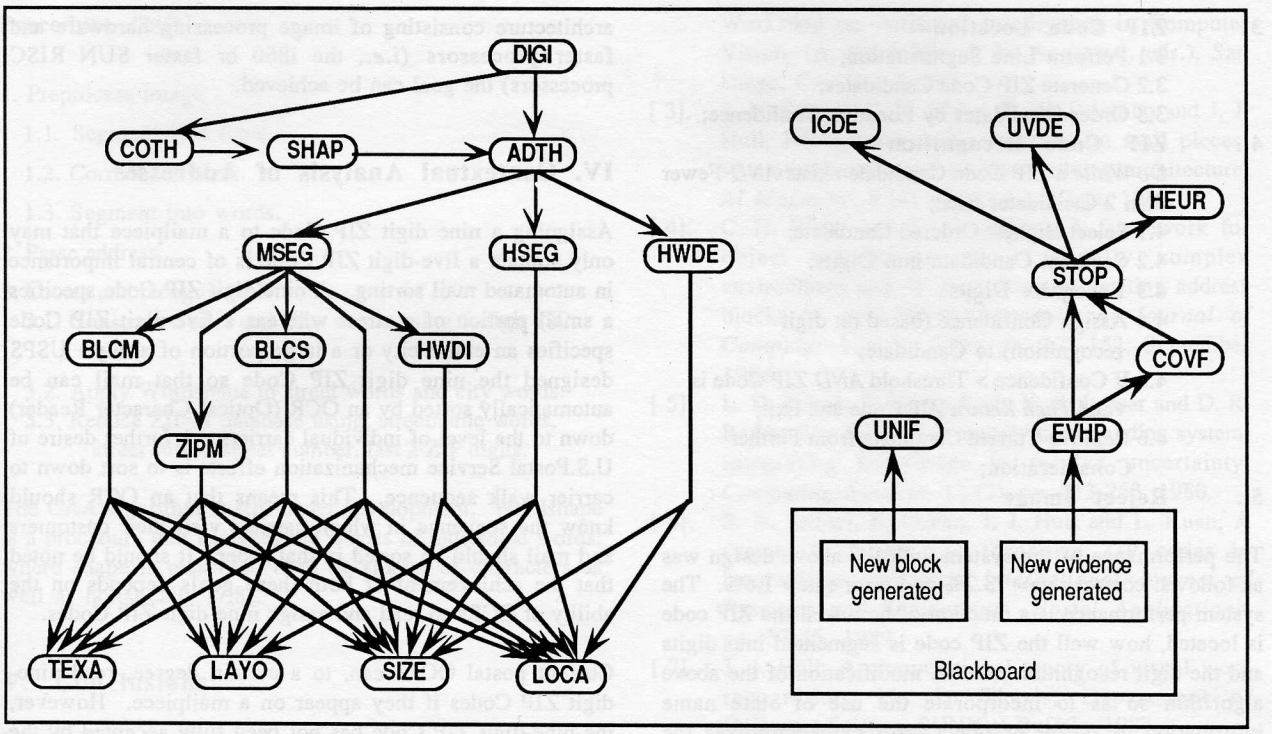

Figure 2: Dependency graph for specifying the temporal sequence of applying specialized tools in ABL.

MSEG and HSEG in parallel. The HSEG operation is itself broken into 3 stages: HSEGF, HSEG1 and HSEG2. The MSEG operation is divided into two stages: MSEG1 and MSEG2. HSEGF converts a handwritten image into vertical strokes by deleting ligatures. HSEG1 and MSEG1 are essentially the same operation of connected component labeling and grouping with slightly different parameters for eliminating noise components. MSEG2 and HSEG2 are similarly identical but involve grouping components into blocks. We are presently developing special purpose hardware to perform the HSEG and MSEG functions.

III. Handwritten ZIP-code recognition

The ZIP code is a five- or nine-digit number which is the principal information used in United States post offices to sort mail. About fifteen percent of first class mail is handwritten.

Determining the ZIP code from a handwritten address is a microcosm of the general reading problem. It involves locating the text of main interest, *viz.*, the ZIP code and then reading the ZIP code. In order to correctly locate the ZIP code, it may be necessary to read text beyond the ZIP code. So also, reading words helps resolve ambiguity in reading digits.

ZIP code recognition involves locating the ZIP code within the address, and recognizing digits with the help of redundant contextual information present in the city, state and street address part of the address. There are various characteristics that make the problem challenging: the ZIP code is often on the last line (90% of cases), but not always; underlines make segmentation of lines and digits difficult; poorly formed addresses without a ZIP code may be incorrectly identified as having a ZIP code.

A complete system for determining ZIP codes from addresses written with unconstrained handwriting requires many components[6]. It may be necessary to first perform preprocessing operations such as thresholding and underline removal. The text lines have to be determined, separated into words, and candidates for the ZIP code and other keywords (*e.g.*, city, state, P.O. Box, etc.) have to be determined. The flow of control in an initial handwritten ZIP code recognition system is given below. The input is a high-resolution (300 ppi) image of a handwritten address block.

The initial design for handwritten ZIP code recognition (HZR) follows. It does not involve reading city, state and street words.

Procedure HZR

1. Input grey-level image
2. Preprocessing
2.1 Remove Horizontal Lines;
2.2 Label Connected Components;

3. **ZIP Code Location**
   3.1 Perform Line Segmentation;
   3.2 Generate ZIP Code Candidates;
   3.3 Order Candidates by Location Confidence;
4. **ZIP Code Recognition**
   Do While a ZIP Code Candidate exists *AND* Fewer than 2 Candidates tried;
   4.1 Select Highest Ordered Candidate;
   4.2 Segment Candidate into Digits;
   4.3 Recognize Digits;
   4.4 Assign Confidence (based on digit recognition) to Candidate;
   4.5 If Confidence > Threshold *AND* ZIP Code is valid *Then Return* ZIP Code and Exit;
   4.6 Remove Current Candidate from Further Consideration;
5. **Reject Image**

The performance of the system with the above design was as follows: correct rate= 73.2% and error rate= 1.6%. The system performance is a function of how well the ZIP code is located, how well the ZIP code is segmented into digits and the digit recognition rate. A modification of the above algorithm so as to incorporate the use of state name information to accept or reject error cases improves the correct recognition rate to 75.2% with a 1.6% reject rate. We are presently working on boosting system performance to the 80% level with a 1.5% reject rate.

Many considerations must be taken into account in evolving a real-time handwritten ZIP Code recognition system from its non real-time counterpart. As an example, consider our recognition system that achieves a 70% correct rate and a 1.5% error rate. Using a SUN 4/260, the average execution time for this system per mail piece is over 200 seconds. To reduce the time to an average of 0.25 seconds per piece, a multi-layered modification scheme has been adopted. Code revisions that keep algorithm functionality are an integral part of the first layer since algorithms for demonstrating competence are usually coded without execution speed as a primary objective. One goal that we have adopted in this layer is the reduction of the amount of information that must be considered at each stage in the computation. On the second layer, some time consuming operations such as connected component labeling become candidates for hardware implementation while the more complex time consuming algorithms must seriously be considered for replacement. Such replacements offer a reduced execution time but they also may negatively impact upon performance since the replacement algorithm is an approximation to the original algorithm. Consequently, these tradeoffs must be seriously considered. On the third level, image processing algorithms are being targeted for existent special purpose image processing hardware. The modifications in these three layers lead to a system that still exceeds 0.25 seconds per mail piece but, by moving the system into a pipelined architecture consisting of image processing hardware and faster processors (*i.e.*, the i860 or faster SUN RISC processors) the goal can be achieved.

IV. Contextual Analysis of Addresses

Assigning a nine digit ZIP Code to a mailpiece that may only contain a five-digit ZIP Code is of central importance in automated mail sorting. A nine digit ZIP Code specifies a small portion of a street whereas a five-digit ZIP Code specifies an entire city or a large portion of a city. USPS designed the nine digit ZIP Code so that mail can be automatically sorted by an OCR (Optical Character Reader) down to the level of individual carriers. A further desire of U.S.Postal Service mechanization efforts is to sort down to carrier walk sequence. This means that an OCR should know the sequence in which carriers visit their customers and mail should be sorted in that order. It should be noted that the achievement of both these goals depends on the ability of OCRs to read and assign nine-digit ZIP Codes.

Current postal OCRs can, to a certain degree, read nine-digit ZIP Codes if they appear on a mailpiece. However, the nine-digit ZIP Code has not been fully accepted by the American public and only a small amount of mail bears the nine-digit ZIP Code. To counter this problem, USPS has been deploying OCRs that can read several lines of text in an address and lookup the proper nine-digit ZIP Code in the ZIP+4 database. However, this strategy has met with limited success because of processing speed requirements for the OCRs (about ten pieces per second) and the fact that the machines were designed with computer technology from the 1960s and 1970s that has not allowed them to incorporate advances in recognition techniques.

Our solution to the problem can be summarized as follows:
1. Read binary address image and the OCR results for this image.
2. Preprocess image to remove as much noise and other artifacts as possible.
3. Segment image into lines and words.
4. Determine five digit ZIP Codes by analyzing words in the street line.

The analysis of words in the last step refers to determining the word or words that can match the image of a word. These words are termed the *neighborhood* of the word image. A neighborhood with only one word in it is equivalent to a complete recognition of the word image. The neighborhoods of the various words in an address are then used to access the USPS nine digit ZIP Code Database and to improve the sort assignment of the piece.

The basic algorithm for Contextual Address Analysis (CAA) is given below:

Procedure CAA

1. Preprocess image
   1.1. Segment into lines.
   1.2. Correct for skew.
   1.3. Segment into words.
2. Parse address
3. Locate and recognize ZIP code
   3.1. Constrain Street and City words using 5-digit ZIP Code.
   3.2. Apply Wordshape to street words and city words.
   3.3. Reduce ZIP+4 database using: streetname words, street suffix, street number, last 2 ZIP digits.

The CAA algorithm is still under development. Wordshape is a procedure that analyzes the shapes of individual words; where shape is measured both by shapes of characters as well as holistically [7,8].

V. Conclusions

Efforts in high performance vision system development involve developing computational theories for significant subsystems, where each computational theory is influenced by its algorithm and eventual hardware implementation.

The design of reading machines involves two major subsystems: finding text to be read and reading text. Finding text can be performed with a lower resolution image than for reading text.

In developing high-performance subsystems for a reading machine, a two pass approach has been taken. During the initial design, a computational theory was developed for each subsystem and a prototype to demonstrate competence (in terms of ability to do the desired task without regard to speed) was developed. During the final design, the computational theory, algorithms and representations were significantly changed to achieve real-time performance while striving to maintain the original recognition rates. For instance, in the case of finding text to be read (address block location), the black-board architecture employed in the initial design was modified to a pipe-line architecture in the final design.

References

[1]. D. Marr, *Computer Vision*, W. H. Freeman, San Francisco, 1986.

[2]. S.N. Srihari, R. K. Srihari, S. Lam and V. Govindaraju, Document image understanding: Using knowledge to read newspapers, Proc. IEEE-CS Workshop on Artificial Intelligence in Computer Vision, (A. Rosenfeld, J. K. Aggarwal, eds.), San Diego, CA, June 1989.

[3]. S. N. Srihari, C.-H. Wang, P. W. Palumbo and J. J. Hull, Recognizing address blocks on mail pieces: specialized tools and problem-solving architecture, *AI Magazine*, 8 (4), pp. 25-40, Winter 1987.

[4]. C.-H. Wang and S. N. Srihari, A framework for object recognition in a visually complex environment and its application to locating address blocks on mail pieces, *International Journal of Computer Vision*, 2 (2), pp. 125-152, September 1988.

[5]. L. D. Erman, F. Hayes-Roth, V. R. Lesser and D. R. Reddy, The Hearsay-II speech understanding system: integrating knowledge to resolve uncertainty, *Computing Surveys*, 12 (2), pp. 213-253, 1980.

[6]. S. N. Srihari, E. Cohen, J. J. Hull and L. Kuan, A system to locate and recognize ZIP codes in handwritten addresses, *International Journal of Research and Engineering - Postal Applications*, pp. 37-45, 1989.

[7]. J. J. Hull, A computational theory of visual word recognition, Ph. D. Thesis, Department of Computer Science, SUNY at Buffalo, 1987.

[8]. T. K. Ho, J. J. Hull and S. N. Srihari, Combination of structural classifiers, *Proc. SSPR-90: Workshop on Structural and Syntactic Pattern Recognition*, Murray Hill, NJ, June 1990.