Pseudo Two-Dimensional Hidden Markov Models for Document Recognition

Oscar E. Agazzi
Shyh-shiaw Kuo

Hidden Markov models (HMM) have become the most popular technique for automatic speech recognition. Extending this technique to the two-dimensional domain is a promising approach to solving difficult problems in optical character recognition (OCR), such as recognizing poorly printed text. Hidden Markov models are robust for OCR applications due to:

- Their inherent tolerance to noise and distortion,
- Their ability to segment blurred and connected text into words and characters as an integral part of the recognition process,
- Their invariance to size, slant, and other transformations of the basic characters, and
- The ease with which contextual information and language models can be incorporated into the recognition algorithms.

We give a brief overview of OCR algorithms based on two-dimensional hidden Markov models, and we present three case studies that show their remarkable strengths.

Introduction

During the past 30 years, a variety of methods have evolved in the field of optical character recognition (OCR). The oldest, template matching, is used by many commercial OCR machines and software. Other methods include feature analysis, followed by classification using statistical classifiers, and neural networks. For a recent review of character-recognition methods, see Govindan et al.[1] and Mori et al.[2] We will discuss a new approach to OCR, based on two-dimensional hidden Markov models (HMMs). In OCR, HMMs provide a powerful tool for predicting the structure of a badly degraded image. A brief overview of HMMs is presented in Panel 2.

Most current OCR methods perform properly only when working with clean text. These methods fail with text that is blurred and degraded by noise and geometric distortions, as often happens after a text goes through multiple reproductions and fax transmissions.[3,4] Documents blurred and severely contaminated with noise are quite common. Figure 1 shows examples of a document that seriously challenges present OCR systems. In Figure 1a, the characters often touch, and the segmentation of the text into individual characters is difficult, due to blur and noise. In Figure 1b, we see a clean example of the word "operation" that could easily be recognized by a commercial OCR system. However, we also see an example of the word with severe vertical geometric distortion, due to the irregularity of the paper speed in a facsimile transmission, an impairment quite common in many fax machines. As noted, all commercial, and most experimental, OCR systems fail drastically when presented with these kinds of distortions.

Two-dimensional hidden Markov models provide a powerful technique to solve this problem, since they:

- Have inherently high tolerance to distortion and blur.
- Allow the operations of image *normalization, segmentation,* and *recognition*—steps traditionally performed independently in OCR—to be done simultaneously, which allows these functions to be optimized jointly. The result is enhanced robustness.

Panel 1 - Acronyms and Terms Used in This Paper

1D HMM — One-dimensional hidden Markov model

2D HMM — Two-dimensional hidden Markov model

HMM — Hidden Markov model. See Panel 2 for a description.

Normalization — Image normalization refers to the process of identifying the transformations that have affected the image and their parameters, and applying the inverse transformations to create a new image in which a large part of the variability found in typical documents is eliminated. Examples of common transformations are scaling and slanting.

Observable Markov model - A set of models that are used to determine the probability of observable states, such as whether a system is running properly (Good) or not (Bad). See Panel 2 for a description.

OCR — Optical character recognition

PHMM — Pseudo two-dimensional, or planar, hidden Markov model

Segmental k-means — An algorithm that iteratively improves the parameters of a set of HMMs, given an initial set with coarse approximations to the parameters.

Segmentation — Reducing an image to its individual characters or other identifiable components.

Recognition — Identifying an image either deterministically or statistically.

Viterbi algorithm — An algorithm used to find the optimal sequence of states in a Markov model, to represent a certain image.

In the traditional approach, errors introduced in the image normalization and segmentation operations inevitably induce errors in the recognition operation. These steps will be discussed below.

- Allow transition probabilities among characters and contextual information to be incorporated in a natural and elegant way.

Previous work in the area of degraded text recognition is described by Chou,[5] who discusses two-dimensional, stochastic (or probabilistic), context-free grammars used to recognize equations, and Bose et al.[6], who discusses one-dimensional (1D) HMMs used to recognize connected and degraded text. The work presented in this article is a continuation of the work described by Chou and Bose et al.

Pseudo Two-Dimensional Hidden Markov Models

One-dimensional hidden Markov models have been used with great success in speech recognition for a number of years.[7] One-dimensional (1D) HMMs also have proved to be effective in character recognition. Bose[6] describes a connected-text recognition system that successfully recognizes extremely degraded words. 1D HMMs also have been used successfully in keyword spotting systems.[8] The effectiveness of 1D HMMs in these problems suggested to the authors that even greater benefits could be obtained by extending their flexibility to two dimensions, that is, to determine what an image is by examining both its horizontal and vertical characteristics.

One objective of a character-recognition method is to represent the shape of different characters in the alphabet with enough flexibility that variations in the shape—caused either by the use of different typefaces or sizes, or by degradations in the document—can be captured in the structure used to represent them. At the same time, the representation should be tolerant to noise, blur, and printing defects, such as broken characters. These defects often appear in a computer representation of an image. They result from slight misadjustments in the threshold of the scanner, even when the original document was clean.

Many of these sources of variation can be adequately captured when the character shapes are represented by two-dimensional, stochastic, finite-state networks known as *pseudo two-dimensional* or *planar hidden Markov models* (PHMM). They are two-dimensional generalizations of one-dimensional hidden Markov models.

Each character of the alphabet is represented by a PHMM. Thus, to represent the English alphabet, we would need 52 PHMMs (26 for lower case letters, and 26 for capital letters). In addition, we would need 10 more PHMMs to represent the digits 0-9, plus a PHMM for every punctuation mark and symbol. A complete OCR system using HMMs may require several hundred HMMs to represent all the possible variations of text.

The structure of these 2D HMM networks, and the way in which they represent characters, is illustrated in Figure 2. The character image to be scanned is divided into a number of regions, each having

Panel 2: A Brief Overview of Hidden Markov Models

Let's consider a system that may, at any time, be in one of a set of distinct states. At regularly spaced, discrete times the system either undergoes a change of state, or remains in the same state, according to a set of probabilities associated with the state. Also, let's assume that the probability of being in a certain state at a specific time (t) depends only on the state of the system at the previous time interval ($t-1$), *independent* of any prior intervals ($t-2$, $t-3$, etc.).

As an example of such a system, let's consider a simple 3-state model of the weather. Assume that once a day the weather is observed as being in one of the following states:

State 1: Rainy
State 2: Cloudy
State 3: Sunny

In this case, we can assume, based on past experience, that the probability of a certain weather condition depends only on the weather condition the day before. Figure 11 on page 71 shows a diagram describing this system. The figure also shows possible numerical values of the *transition probabilities* as labels on the arrows that represent the transitions. For example, according to this model, the probability of having a sunny day tomorrow, given that it is raining today, is 0.3.

Since the state of the weather is directly accessible to the observer, this is an example of an *observable* Markov model.

For a *hidden* Markov model, the state of the system is not directly observable. What is instead observable is some random variable, or set of random variables, whose statistical distribution depends on the state of the system.

The example of an observable Markov model, the weather outside, could be converted to an example of a hidden Markov model if the observer was locked inside a room without windows, and the only source of knowledge about the weather condition was measurements of the temperature and humidity taken at regular intervals. In this case, the observer cannot know directly the state of the system. The best that can be done is to *estimate* the sequence of states of the system. This estimate would be based on the temperature and humidity measurements, the *a priori* knowledge of the transition probabilities of Figure 11, and, as a starting point, knowledge of the state of the weather at the time the person was locked in the room.

Although the temperature and humidity do not uniquely identify the state of the weather, their statistical distributions depend on it. Therefore, the measurements provide some information about the possible states of the weather. Using this information, as well as the knowledge of the transition probabilities and the initial state of the weather, the observer can estimate the entire sequence of weather states since entering the room.

approximately similar horizontal cross-sections. In Figure 2a, for example, the first of these regions is the space above the character. The second region is the ascender of the letter "h." The third is the horizontal bar, and the fourth is the "legs" of the "h" below the horizontal bar. In this example, each region has a constant horizontal cross-section. In other cases, there may be some variability of the horizontal cross-section from the top to the bottom. This variability should be small, however, or the region should be subdivided into smaller regions. The exact number of regions required depends on the types of characters.

Each region is represented by a 1D "left-to-right"[7] HMM, which models the sequence of pixel observations as we scan a horizontal section of the image. First, let's look at the horizontal areas singly. In this example, we are looking at two states with our HMM, the background and foreground regions in the character image. That is, we are looking for either the existence or absence of pixels to recognize the image. For simplicity, let's assume the image is binary, that is, just black and white. In this case, shades of gray would be assigned by the scanner as either black or white states, based on the density of the pixels and other factors. The observations in each HMM state are represented by binary pixel values. For example, a 1 can be used to represent a black (foreground) pixel and a 0 to represent a white (background) pixel.

Bear in mind, the "h" we are looking at is a clear, crisp image. Indeed, it is so clean, so perfect an image, the observations in each state would be deterministic. That is, the image could be determined by matching it to

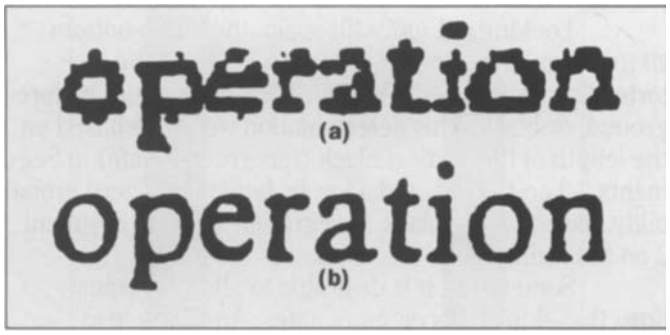

Figure 1. The illustration (a) shows examples of degraded documents, that is, blurred and "noisy" text. In (b) we see a clean example of the word "operation." (a) is an example of the word with geometric distortion due to faxing, which would not be recognized by commercial OCR systems.

a stored template, rather than processing it through a HMM to determine its identity. In this case, the *hidden* Markov model would be replaced by an *observable* Markov model.[7]

But in many cases, where images are corrupted by noise or other defects, such as in Figure 2b, the observations are stochastic, rather than deterministic, and must be conjectured by describing them in terms of their probability distribution. In this figure, the distortion at the bottom of the image suggests it could be an "h," "b," or even a stylized "6". The HMM would analyze the foreground and background states to determine the probability that the distortion at the base of the image indicates the image is either an "h," or a "b" or "6." In this case, the black, or foreground pixels between the "legs," will be "weighed" by the HMM. This determines if they are foreground or just noise, and whether the white, or background pixels distorting the "legs," are background or foreground pixels. Foreground states typically emit foreground pixels with high probability, unless the degradation is extreme, but the probability of a background pixel is still greater than zero. Similarly, background states emit foreground pixels with low, but nonzero, probability.

The probability of one state transitioning to another state is associated with the length of the image segments represented by those states. Long segments are represented by states with a low probability of transition to the next state. Similarly, short segments are represented by states with a high probability of transitioning

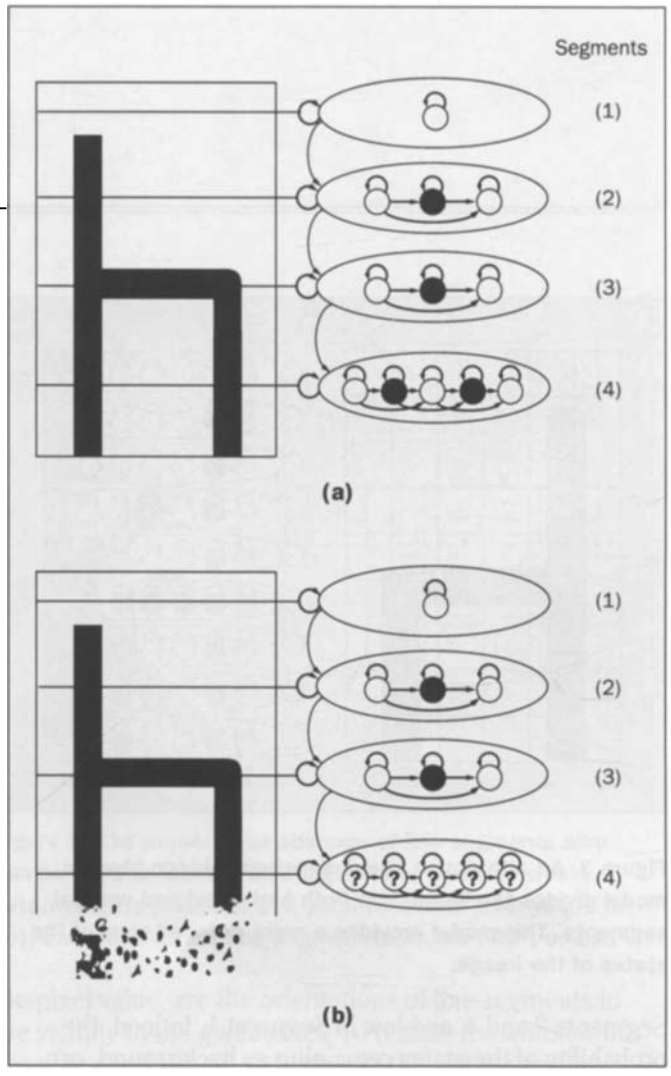

Figure 2. In (a), a pseudo two-dimensional hidden Markov model divides the image, in this case, a clearly recognizable "h," into horizontal segments. It then determines the "state" of each segment. Note that in Segment 1, only one state, background, was determined. In contrast, in the Segment 4, five states were determined, three of background (white), and two of foreground (black). In (b), the portion of the image in Segment 4 has been distorted, and noise has been introduced. This image could be an "h" or, if the distortion actually represents foreground, it could be a "b" or stylized "6." The HMM algorithm would evaluate the probability of the "legs" being either foreground or background by means of the pixel value, the states in the segments above and below this segment, and the density of the noise between the legs.

to the next state. More specifically, it can be shown that the probability of a transition from state i to the state $i+1$ is given by: $a_{i,i+1} = 1/l_i$ where l_i is the length in pixels of the segment that corresponds to state i. Thus, in Figure 2b, the probability of the states remaining as foreground, or black, is high across Segment 3, moderate in

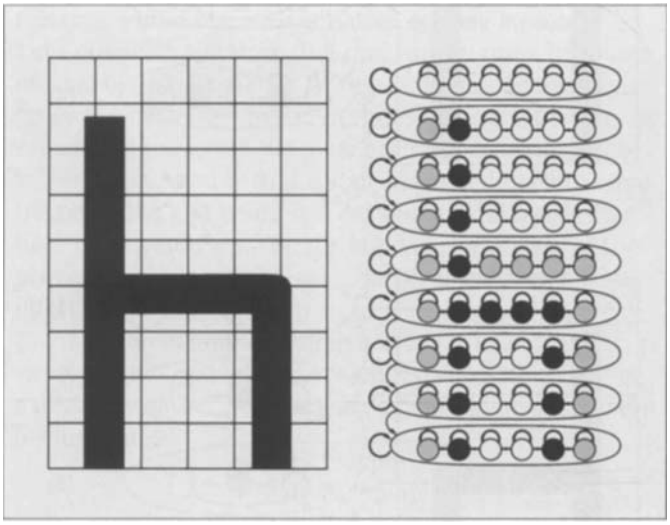

Figure 3. An alternative two-dimensional hidden Markov model divides the image into both horizontal and vertical segments. This model provides a more detailed view of the states of the image.

Segments 2 and 4, and low in Segment 1. Indeed, the probability of the states remaining as background, or white, is high across Segment 1.

So far, we have employed a 1D HMM to analyze the individual horizontal segments of the image. Now we will look at the vertical structure of the image. This vertical structure is captured by embedding the set of 1D HMMs, which is associated with horizontal movement in the image, into another HMM, which is associated with a movement from top to bottom in the image. By analogy with left-to-right HMMs, we could call this a "top-to-bottom" HMM. If this top-to-bottom HMM looked only at the vertical aspects of the image, it also would be a 1D HMM. However, the observations of the top-to-bottom HMMs are affected by the state transitions in the left-to-right 1D HMMs. For this reason, this HMM is called a 2D HMM, and the states of this new HMM are called "superstates," because the observations contained in them correspond to complete horizontal sections of the image, and their probability is determined by the probability of the associated horizontal 1D HMM. In a way that is entirely similar to the horizontal HMMs, the transition probabilities from superstate to superstate relate to the height of the associated region in the image.

Looking at Figure 2b again, the top-to-bottom 2D HMM would assign a high probability that the distorted "legs" area of the "h" in Segment 1 should be foreground, or black. This determination would be based on the length of the vertical black (foreground state) in Segments 2-3 on the left, and a lower, but still not-zero probability, based on the black (foreground state) in Segment 2 on the right.

Sometimes, it is desirable to allow for transitions that skip states or superstates. An example of skipping a state is shown in Figure 2. Segment 1 is represented by only one state (0), Segments 2 and 3 are represented by just three states (0,1,0), and Segment 4 is represented by five states (0,1,0,1,0). These transitions can be used to model image defects. For example, a transition skipping the black state in Segment 2 could model a break in the ascender of the character "h." Similarly, when characters are connected, the white states at the edges may be skipped.

The fact that the transitions from state to state, or superstate to superstate, are stochastic, instead of deterministic, allows the network to model variations in the basic character shape. These include size variations, various degrees of slant, or the geometric distortions introduced by fax transmission, as shown in Figure 1b. It is clear, however, that the network of Figure 2 does not allow for totally general connections among states. Therefore, not every conceivable character distortion can be modeled by these networks. For this reason, we call them "pseudo two-dimensional," instead of fully two-dimensional, HMMs. Nevertheless, the class of distortions that can be adequately modeled by PHMMs is large enough to cover almost all cases of interest in character recognition.

The relation between a PHMM and a character image, shown in Figure 2, is not the only one possible. An alternative, shown in Figure 3, is to divide the image into a uniform lattice, and relate to each cell in the lattice a state of the model. In this case, the probability of observing a foreground pixel in a given state is associated with the density of foreground pixels in the corresponding cell of the image.

Another possible variation is to use horizontal, instead of vertical, superstates. This possibility is illustrated in Figure 4. This structure has a disadvantage: It cannot model slant transformations as well as the

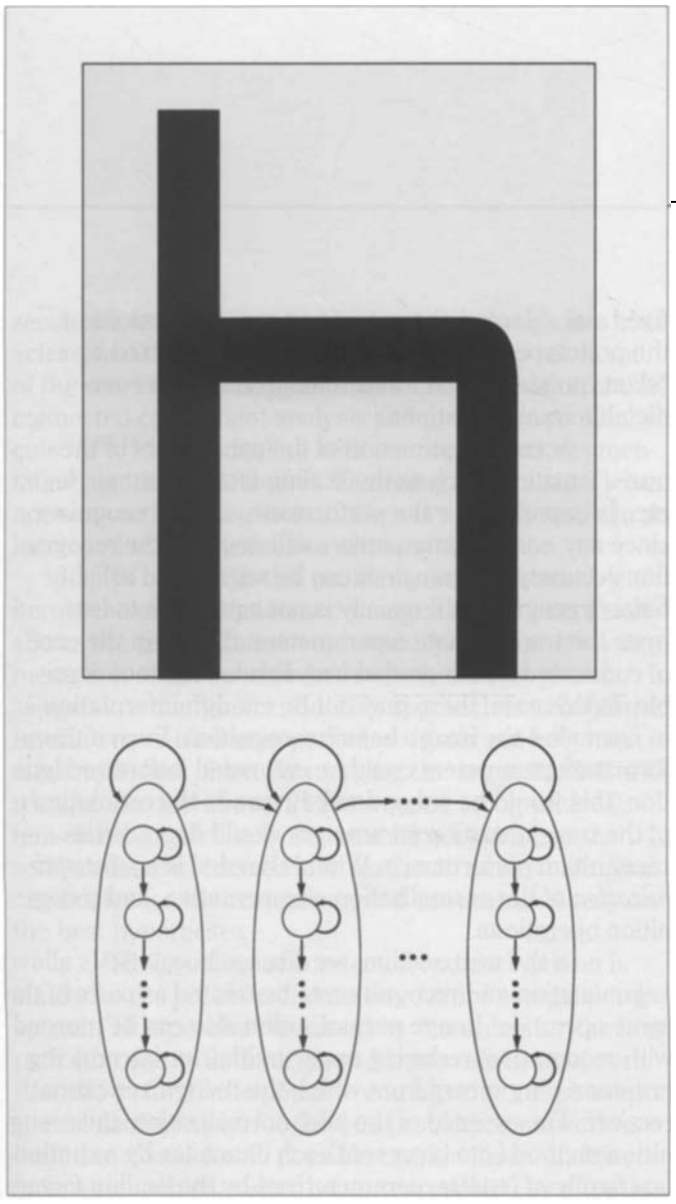

Figure 4. Another alternative pseudo two-dimensional hidden Markov model makes it easier to connect to other similar structures to form models of strings, instead of just isolated characters. One disadvantage is that it doesn't model slant transformations very well.

horizontal structure. However, it is much easier to interconnect to other similar structures to form models of entire strings, instead of isolated characters. This is important in the problem of connected text recognition, which we discuss in the section, "Connected Text Recognition."

Although in our discussion we have considered the pixel values as the only observations in each state of the PHMM, it is possible to use more elaborate observations, or "features." The observations can be seen as the components of a multidimensional "feature vector." For example, other features that can be used, in addition to

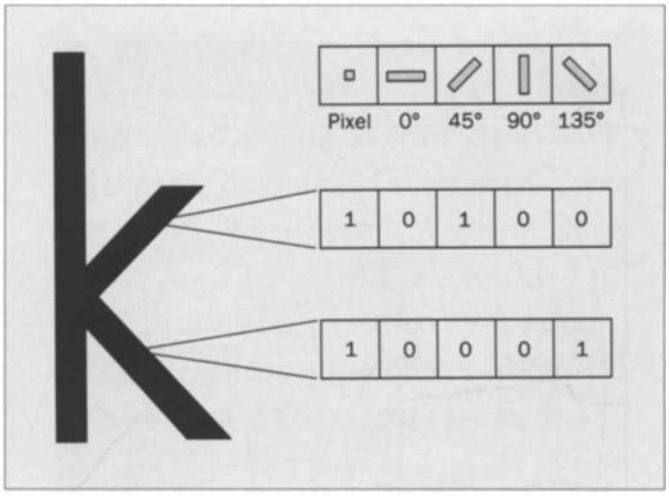

Figure 5. The presence or absence of line segments also can be used to identify an image. Here, an HMM would determine the presence of a pixel (encoded 1 for yes, 0 for no), and whether the line segment is 0°, 45°, 90°, or 135°.

the pixel value, are the orientations of line segments in the vicinity of the given pixel. To reduce the number of possible dimensions, or angles, of the feature vector, these orientations can be quantized into a small number of values, indicated by such angle variations as 0°, 45°, 90°, and 135°. This is illustrated by Figure 5. Other possible features are discussed in Kuo et al.[9]

Recognition. Recognition is the process of selecting, from among the many PHMMs available in a library (typically one or more per character in the alphabet), the one that best represents the analyzed image. Toward this end, the probabilities of the image being generated by the different models in the library are evaluated by giving them *scores*, using a two-dimensional version of the *Viterbi algorithm*. The algorithm is used to find an optimal sequence of states, given a certain image. After the scores of all models and the given image have been computed, the model with the best score, that is, the highest probability of being correct, is selected. A detailed description of the two-dimensional Viterbi algorithm can be found in the references.[9,10]

So far, we have not considered the problem of how to obtain the PHMMs that best represent a given character set. This is usually accomplished by *training* the models, using a set of images representative of the

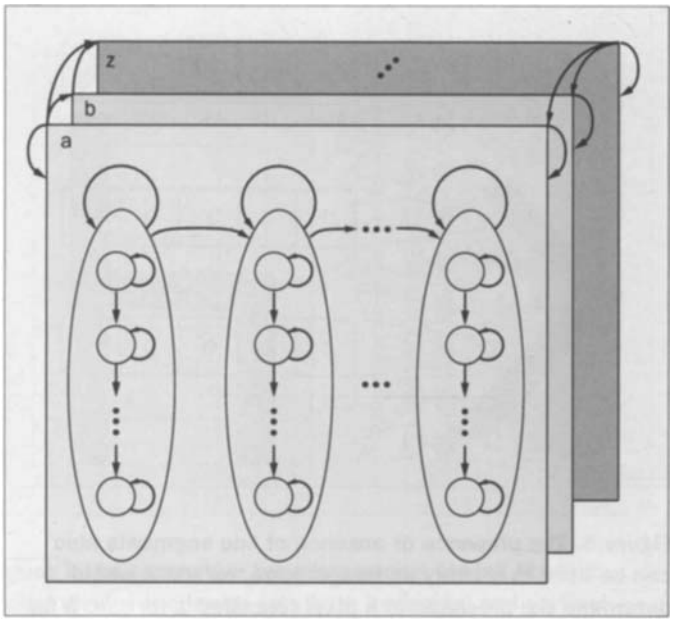

Figure 6. The problem of connected-text recognition can be solved by building a network that allows for all possible combinations of characters, each character being represented by a planar hidden Markov model. The network allows for arbitrary transitions from character to character.

statistical properties of the images that must be recognized in the applications. A training algorithm frequently used is the *segmental k-means* algorithm. Given a set of PHMMs with some initial values of parameters, the segmental-k means algorithm iteratively refines the parameters until nearly optimal values are obtained. The initial values of parameters needed to start the segmental-k means iteration are typically computed using simple heuristics. This algorithm, first applied with excellent results in speech recognition, is described in Rabiner.[11]

Transformation Invariance. In most document recognition applications, the recognizer must be able to handle a wide variety of character sizes, as well as slanted or otherwise transformed characters. The method most commonly used to deal with this variety calls for *normalizing* the image before handing it over to the recognizer. Image normalization refers to the process of identifying the transformations that have affected the image and the parameters of the transformations, then applying the inverse transformations. Characters of different sizes are treated as scaled versions of certain prototype characters of fixed size. Slanted characters are seen as versions of the prototype characters that have been affected by a "slant transformation," and, similarly, by other predictable transformations.

Accurate estimation of the parameters of the transformation, such as the scaling factor, slant angle, etc., is important for the performance of the recognizer, since any normalization errors will degrade the recognition accuracy. If characters can be segmented reliably before recognition, it usually is not a problem to estimate the transformation parameters. But, as in the case of connected and degraded text, this is often not possible. In this case, there may not be enough information to normalize the image before recognition. Even if transformation parameters could be estimated before recognition, this would be suboptimal. Errors in the estimation of the transformation parameters would degrade the recognition performance. What is needed is a *joint optimization* of the normalization, segmentation, and recognition operations.

In the next section, we discuss how PHMMs allow segmentation and recognition to be treated as parts of the same operation. Image normalization also can be merged with recognition, reducing opportunities for introducing preprocessing errors from which the recognizer cannot recover. The essence of the joint normalization and recognition method is to represent each character by a continuous family of PHMMs, parameterized by the scaling factor, slant angle, and other transformation parameters.

A single pass of the Viterbi algorithm is enough to search over each one of these continuous families, yielding both the score for each family and the values of the parameters that maximize the score. These values are then used to compute transformation-independent scores for the PHMMs under test. In this way, estimation of transformation parameters, inverse transformation (normalization), segmentation, and recognition are all simultaneously optimized. Accurate estimation of the transformation parameters is a useful byproduct of this process. The values of these parameters can be used by higher-level modules in an intelligent document-recognition system to provide a more complete description of the document structure.

Connected Text Recognition

One of the most interesting applications of PHMMs is the recognition of connected text. Traditionally, string

recognition algorithms first segment the image into characters, and then recognize individual characters. Some of the most common algorithms for segmentation are connected-component analysis and projection-profile cuts.[12,13,14] When dealing with connected text, segmentation algorithms often fail. The recognizer normally cannot recover from segmentation errors, which result in an increased error rate for the whole system.

To cope with this problem, some systems[4,15] iterate the segmentation and recognition phases. This allows the segmentation engine to try an alternative segmentation of the substrings to which the recognizer assigns low confidence. But this approach is suboptimal. Invalid segmentation hypotheses should not be generated in the first place. By merging the segmentation and recognition operations, a PHMM-based recognition system eliminates unlikely segmentation hypotheses as soon as there is enough information to make them poor candidates for the recognizer, and concentrates only on the best hypotheses.

The problem of connected-text recognition is similar to the problem of connected word recognition in speech. This problem is solved by searching for the best path (the one with the highest likelihood) in a network that allows for all the possible combinations of phonetic HMMs. Analogously, for the OCR problem, a network is built that allows for all possible combinations of characters, with each character being represented by a PHMM. The structure of this network is illustrated in Figure 6. Notice that this network is itself a hidden Markov model; it allows arbitrary transitions from character to character. The transition probabilities among character-level PHMMs correspond to the transition probabilities among characters, or probabilities of bigrams, or pairs of characters of letters, in the English language (or any other language, according to the application).

When using HMMs, segmentation and recognition are not treated as two separate operations. Instead, they are done simultaneously in a single Viterbi search. In this way, the two operations are optimized jointly. Unlike the traditional approach, which can generate invalid segmentation hypotheses (which may or may not be detected later during recognition), the Viterbi search automatically eliminates unlikely segmentation hypotheses early in the search procedure. It does this by concentrating on just those hypotheses that have the highest likelihood when considered as complete strings.

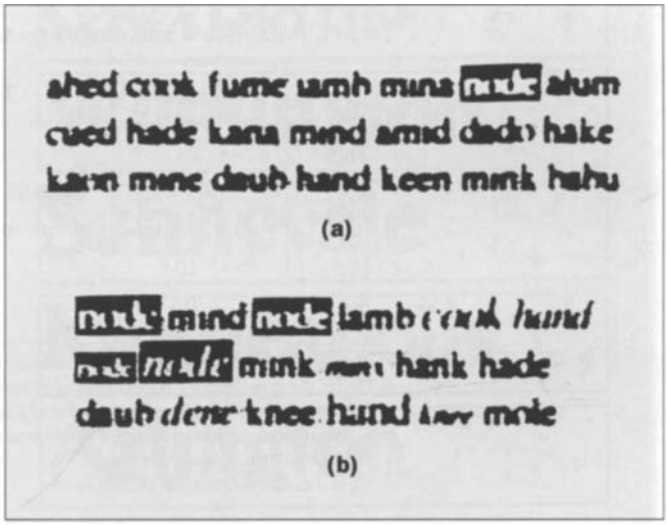

Figure 7. In (a), the keyword "node" is correctly spotted on this document, which contained, in one test, 2,650 key words and 16,000 extraneous words. In (b), the keyword "node" also was successfully spotted in a document that included both size and slant transformations of the word.

Incorporation of Contextual Information. The simplest form of contextual information that can be incorporated into the recognition algorithm is the use of transition probabilities among characters, or probabilities of bigrams, as was mentioned in the previous section. This does not require any modification of the recognition algorithm, and does not introduce any penalties in computation speed. All that is needed is to attach the proper weights to the arcs of Figure 6, which correspond to transitions among the different PHMMs representing the possible images.

A more powerful, and also computationally more expensive, way to add contextual information is to use grammars. In speech recognition, grammars are used to reduce the possible number of sentences that can be legally generated, and recognized, according to the application. They impose both lexical and syntactic constraints. The same can be done in character recognition. In the particular example of the section to follow, "City Names Database," we attempt to recognize words that are part of a limited lexicon. Use of a dictionary as a post-processor, to check the recognized words and correct errors, is one traditional approach to exploit this information to improve the accuracy of the recognizer.

Table I. Parameters and ranges for training and testing sets.

	Training Set	Testing Set I	Testing Set II	Testing Set III
Thld	78	75, 78, 83	75, 78, 83	75, 78, 83
P_n	0.05, 0.4	0.05, 0.1, 0.2, 0.3, 0.4	0.05, 0.1, 0.2, 0.3, 0.4	0.05, 0.1, 0.2, 0.3, 0.4
σ_n	0 ~ 1.9 pixels	0 ~ 2.2 pixels	0.4 ~ 2.5 pixels	0.4 ~ 2.5 pixels
Pt	10	10	8, 10, 12, 14	8, 10, 12, 14
Stg	0°	0°	0°	0°, 20°

Testing Sets II and III include words with different sizes and higher degradation. Testing Set III also includes slanted words.
- Thld is the threshold for binarization of images.
- P_n is the probability of flipping the value of a binary pixel.
- σ_n is the standard deviation of the 2D Gaussian-blur function.
- Pt is the point size of a printed word.
- Stg is the slant degree of a printed word.

However, the use of a grammar prevents the errors from being made in the first place, and is, therefore, an optimal way of exploiting the lexical constraints. The strategy of using grammars to incorporate contextual constraints, together with the previously discussed strategies of joint normalization, segmentation, and recognition, allows hard decisions to be avoided until all the available information has been used.

In addition to linguistic constraints, grammars could be used to incorporate other forms of contextual information, as well. Font and size usually remain constant within a word, for example. They also change relatively infrequently within a document. These constraints could be easily incorporated using grammars, although this has not been done in the experiments reported in the next section, "Applications."

Other forms of contextual information, such as semantic and pragmatic context, can be exploited to enhance the accuracy of the recognizer. Nagy[16] has an interesting and entertaining review of this subject.

Applications

To assess their effectiveness, the techniques described in this article have been applied to three case studies, which we will now discuss.

Keyword Spotting. The objective of *keyword spotting* is to find instances of a selected, and small, group of keywords embedded in large volumes of text. Although this objective could be accomplished by recognizing the entire document, and then finding the keywords in a machine-readable format by using a text editor, finding keywords directly in the image is more efficient. Moreover, accuracy can be much higher when only a small set of keywords, rather than an unconstrained text, must be recognized.

In keyword spotting, PHMMs represent entire words instead of characters. One model is built for each desired keyword, and one additional model, called the *extraneous word* model, is built representing all non-keywords. Discrimination of keyword versus non-keyword is accomplished by comparing the scores of the associated keyword and extraneous word PHMM, given the input word image. A similar modeling strategy has been used with great success in speech recognition.[17]

We assume that the text has been scanned, page preprocessing has been performed, and the word to be recognized has been presented to our system. Characters forming the word may be badly degraded and connected. The input word is preprocessed by a nonlinear filter,[6] to reduce noise, while retaining character sharpness and connectivity. Prior knowledge about the keywords is then used to eliminate unlikely candidates, based on word length, ratio of word length to height, presence or absence of ascenders and descenders, etc.

Thresholds can be obtained from the training set, allowing some margin (±20% in our experiments) for unexpected variations. For example, if the width of a keyword in the training set is between 20 and 30 pixels, then any input word with a width smaller than 16 pixels or larger than 36 pixels will be considered as an extraneous word and removed from further consideration. This

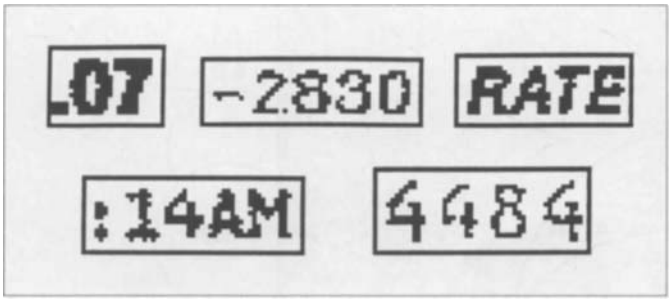

Figure 8. Images rejected by commercial OCRs.

Table II. Sizes of training and testing sets.

Training Set	100 keywords + 240 extraneous words
Testing Set I	345 keywords + 2,000 extraneous words
Testing Set II	1325 keywords + 8,000 extraneous words
Testing Set III	2650 keywords + 16,000 extraneous words

Table III. Accuracy of 1D and 2D HMMs.

Modeling Technique	1D	2D
Testing Set I	97%	99%
Testing Set II	70%	96%
Testing Set III	-	94%

procedure reduces the number of candidates to a small fraction of the total.

If the input word passes the prechecking stage, scores are calculated, using the Viterbi algorithm, for the keyword model and the extraneous word model. The input word is considered as a keyword if its score is higher than the score of the extraneous word. Otherwise, it is not a keyword.

The keyword-spotting system has been evaluated on a computer-simulated database. In these experiments, the word "node" is first specified as the keyword. Then a set of words that have similar shape structures to that of "node" are selected from a dictionary and composed onto a page. All of these words have successfully passed the prechecking step of an algorithm that determines if each is a possible key word. Various clean pages also are formed by using different point sizes and slanted versions of these words. The test-word database then is generated by distorting the clean pages. An example of

Figure 9. Images of city names Charlotte, Dallas, Cleveland, Wilmington, Sunnyvale, Minneapolis, and Arlington form a database with simulated distortion. Recognition of these highly degraded images is enhanced by the use of the grammar in Figure 10.

the database, showing both distorted words and slant and size transformations, is shown in Figure 7.

To test the robustness of the algorithm, the range of the parameters in the training set is chosen as a subset of those in the test sets. Table I lists all the parameters and their ranges for all these sets. Test sets have been made increasingly difficult. The ranges of the first two parameters are the same for all the test sets. However, Test Sets II and III include words with different sizes and higher degradation. In addition, Test Set III also includes slanted words. Table II lists the size of each set. Table III shows the experimental results.

For the purpose of comparison, the results of a 1D HMM algorithm are also given in Table III. The false alarm rates are not listed, since they are small (< 1%) in all cases. Both modeling techniques perform equally well on Test Set I, and both can successfully spot the keyword in that page, as indicated in Figure 7. In other words, both modeling techniques are robust enough to spot the keyword embedded in a poorly printed page, if the normalization and slant correction are not required for the 1D HMM.

The PHMM performs better than the 1D HMM in

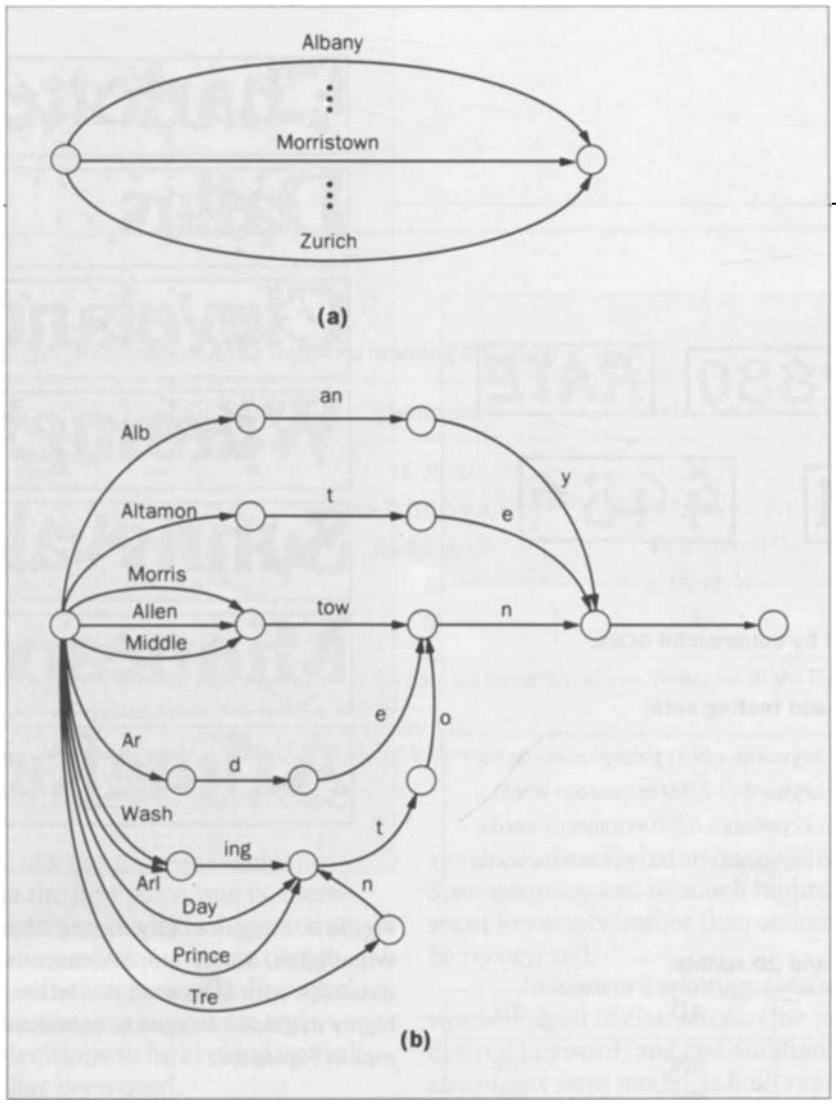

Figure 10. The grammar used in the city names experiment. For clarity, only a few nodes are shown. (a) shows a simple grammar, and (b) shows a grammar obtained by the minimization of (a).

terms of the size and slant independence. If the data set contains words with different sizes, as in Test Set II, then size normalization has to be added to the 1D HMM approach. This results in distortion of the original data and deteriorates performance—only 70% as indicated in Table III. However, the PHMM performs at a 96% accuracy rate on the same test set. In case the data set also contains slanted words, as in Test Set III, then much worse performance can be expected of the 1D HMM. This is a result of the slant correction step being added to the approach, although it has not been evaluated in our experiment.

On the other hand, size normalization and slant correction are not required for the PHMM approach, because it posesses the "elastic matching" property in both horizontal and vertical directions. Performance remains roughly the same for all three test sets. The slight difference of performance among sets is simply due to the fact that the more variation the set has, the less accurately the system can perform. The performance of the algorithm on Test Sets II and III can easily be improved by including these extra variations in the training set.

Figure 7b shows an interesting page in which words are formed by three different sizes and two different fonts, roman and italic (different in slant). The PHMM approach can recognize the keywords embedded in that page, as indicated by black windows in the figure.

Reading Commercial OCR Rejects. Our next experiment involved character recognition on a string of characters that were rejected by a commercial OCR machine with 99.5% accuracy. This machine is used in a high-volume application. The 0.5% rejects, although a small fraction of the total number of characters processed, still represent a large number of characters that must be corrected by a human operator. Using an alternative algorithm to recognize at least part of these rejects will significantly reduce the need for manual intervention and will, therefore, reduce cost.

The strings of images rejected by the commercial OCR machine were random. They came from many fields in the document—some of them numerical, others alphanumerical—so that they were completely unstructured. For this reason, no contextual information was

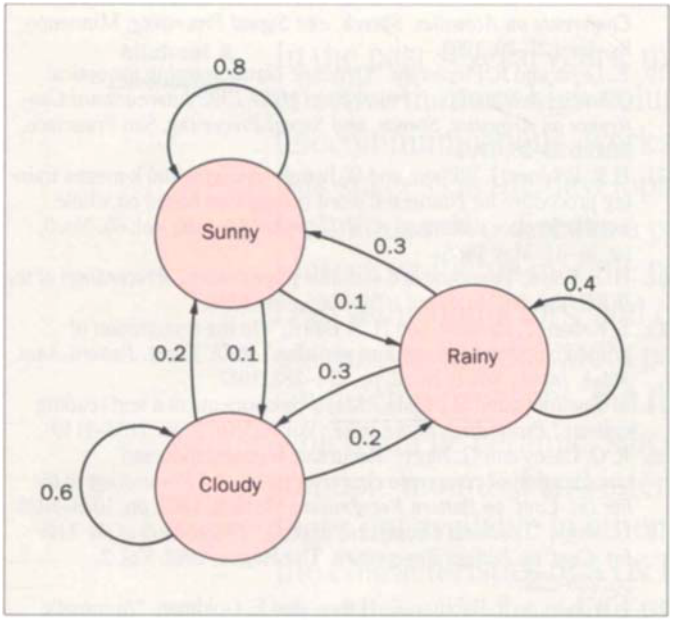

Figure 11. Using this hidden Markov model, a person in a windowless room could estimate the weather conditions outside, using the transition probabilities of this model, the temperature and humidity conditions outside the room, and the state of the weather when the person first entered the room.

directly available to constrain, or help focus, the search in these experiments. However, this contextual information is usually available in most applications, and can be incorporated into the search algorithm, for example, by using grammars.

Figure 8 shows some typical images in this database. The main challenges in obtaining high accuracy on this database are the low resolution and poor quality of the images, the fact that there is a mixture of different fonts and sizes and, as mentioned, the absence of contextual information. We used 6,126 images in our experiment; half were used for training and the rest for testing. The results of our test on the database of rejects by the commercial OCRs was 98.76% for character accuracy, and 96.13% for string accuracy.

If this level of accuracy could be maintained in the field, the compounded error rate of the combination of the commercial OCR machine with the PHMM algorithm would be reduced to 0.0062%, or one error in 16,000 characters. This is almost two orders of magnitude better than the current state of the art.

This excellent result could be improved even further by use of contextual information. A large fraction of the residual errors reported are substitutions caused by highly confusable pairs, such as O and 0. If these errors could be corrected by context, the accuracy would improve to more than 99.5%, and the compounded error rate would fall below one error in 40,000 characters.

City Names Database

The database described in the previous section tested our algorithm in an interesting "real world" application. But it did not exercise to the fullest one of the main strengths of PHMMs—which is the capability to recognize highly connected text. For this reason, we created a simulated database of more than 20,000 blurred and noisy instances of 205 city names. Half of the images were used for training, and the rest for testing. These images were distorted to closely resemble the effects of multiple reproductions and/or fax transmissions. Some of the images were difficult to recognize, even for a human observer, as shown in Figure 9.

Recognition accuracy can be significantly improved, in this application, by using a grammar to constrain the search space. The grammar forces the recognized string to be one of the 205 city names. Figure 10 shows two examples of grammars used in this experiment. Figure 10a simply creates word-level models for the legal city names by concatenating character-level models, that is, segments of words.

Use of this grammar makes this experiment equivalent to a keyword spotting experiment with 205 keywords, except for the fact that an extraneous word model is not used here. The grammar of Figure 10b is obtained from the first by minimization, using a standard finite-state network minimization algorithm.[18] This form of the grammar saves some computation by sharing arcs that are common to more than one city name.

The results were that, with a noise probability of 0.15 and 0.20, 100% of both the characters and strings were recognized. When the noise probability was increased to 0.25, character recognition was 99.65%, and string recognition was 99.62%. One should note that, in the absence of a grammar, the string accuracy would typically be significantly lower than the character accuracy. However, as a result of the use of a grammar, the two percentage values are very similar.

Conclusions

As a result of their ability to combine normalization, segmentation, and recognition, hidden Markov models excel in recognizing connected and degraded text. The three case studies presented in this article demonstrate the potential for this application. Additional work is needed to assess their applicability to the recognition of handprinted text and to script recognition. Further work is also needed to establish the feasibility of special purpose hardware architectures that efficiently can handle the high computational demands of these algorithms.

Acknowledgements

We are grateful to Nikil Jayant, who has been instrumental in identifying the problem, and for providing valuable suggestions and continuous motivation during the course of the work. We are also grateful to E. Levin, R. Pieraccini, and C. H. Lee for many useful discussions and valuable consultation.

References

1. V. K. Govindan and A. P. Shivaprasad, "Character recognition—a review," *Pattern Recognition*, Vol. 23, No. 7, pp. 671–683, 1990.
2. S. Mori, C. Y. Suen, and K. Yamamoto, "Historical review of OCR research and development," *Proceedings of the IEEE*, Vol. 80, No. 7, pp. 1029–1058, July 1992.
3. S. V. Rice, J. Kanai, and T. A. Nartker, "A report on the accuracy of OCR devices," *Information Science Research Institute*, University of Nevada, Las Vegas.
4. M. Bokser, "Omnidocument Technologies," *Proceedings of the IEEE*, Vol. 80, No. 7, pp. 1066–1078, July 1992.
5. P. Chou, "Recognition of equations using a two-dimensional stochastic context-free grammar," *SPIE Conference on Visual Communications and Image Processing*, Philadelphia, November 1989.
6. C. B. Bose and S. Kuo, "Recognition of connected and degraded text using hidden Markov models," *Proceedings of the 11th International Conference on Pattern Recognition*, The Hague, August 31–Sept. 3, 1992.
7. L. R. Rabiner, "A tutorial on hidden Markov models and selected applications in speech recognition," *Proceedings of the IEEE*, Vol. 77, No. 2, pp. 257–286, February 1989.
8. S. Kuo and O. E. Agazzi, "Keyword spotting in poorly printed texts using pseudo 2D hidden Markov models," *Proceedings of the International Conference on Computer Vision and Pattern Recognition*, New York, June 1993.
9. S. Kuo and O. E. Agazzi, "Machine vision for keyword spotting using pseudo 2D hidden Markov models," *1993 International Conference on Acoustics, Speech, and Signal Processing*, Minneapolis, April 27–30, 1993.
10. E. Levin and R. Pieraccini, "Dynamic planar warping for optical character recognition," *Proceedings of the 1992 International Conference on Acoustics, Speech, and Signal Processing*, San Francisco, March 23–26, 1992.
11. L. R. Rabiner, J. Wilpon, and B. Juang, "A segmental k-means training procedure for connected word recognition based on whole word reference patterns," *ATT Technical Journal*, Vol. 65, No. 3, pp. 21–31, May 1986.
12. H. S. Baird, "Anatomy of a versatile page reader," *Proceedings of the IEEE*, Vol. 80, No. 7, pp. 1059–1065, July 1992.
13. S. Kahan, T. Pavlidis, and H. S. Baird, "On the recognition of printed characters of any font and size," *IEEE Trans. Pattern Anal. Mach. Intell.*, Vol. 9, No. 2, pp. 274–288, 1987.
14. S. Tsujimoto and H. Asada, "Major components of a text reading system," *Proceedings of the IEEE*, Vol. 80, No. 7, pp. 1133–1149.
15. R. G. Casey and G. Nagy, "Recursive segmentation and classification of composite character patterns," *Proceedings of the 6th Int. Conf. on Pattern Recognition*, Munich, 1982, pp. 1023–1026.
16. G. Nagy, "Teaching a computer to read," *Proceedings of the 11th Int. Conf. on Pattern Recognition*, The Hague, 1992, Vol. 2, pp. 225–229.
17. J. Wilpon, L. R. Rabiner, C. H. Lee, and E. Goldman, "Automatic recognition of keywords in unconstrained speech using hidden Markov models," *IEEE Trans. Acoust. Speech Signal Processing*, Vol. 38, pp. 1870–1878, November 1990.
18. J. E. Hopcroft and J. D. Ullman, "Introduction to automata theory, languages, and computation," *Addison Wesley*, 1979.

(Manuscript approved November 1993)

Oscar E. Agazzi *is a distinguished member of technical staff in the Signal Processing Research Department at AT&T Bell Laboratories in Murray Hill, New Jersey. His work involves image analysis and recognition. He joined Bell Labs in 1987. He has a B.S.E.E. and an M.S.E.E. from the University of Cordoba in Argentina, and a Ph. D. in electrical engineering from the University of California at Berkeley.*

Shyh-shiaw (Stanley) Kuo *is a member of technical staff at AT&T Image Solutions in Somerset, New Jersey. He is working on the next generation image-fax-phone device. He joined AT&T Bell Laboratories in 1991. Mr. Kuo received an M.S. and Ph.D. degrees in electrical and computer engineering from Rutgers, the State University of New Jersey.*