IJDAR (1998) 1: 147–166

International Journal on IJDAR
Document Analysis and Recognition

A segmentation-free approach to text recognition with application to Arabic text

Badr Al-Badr, Robert M. Haralick

Department of Computer Science and Engineering, Mail Stop FR-35, University of Washington, Seattle, WA 98195, USA

Received March 5, 1998 / Revised July 28, 1998

Abstract. In recognizing cursive scripts, a major undertaking is segmenting cursive words into characters and isolating merged characters. The segmentation is usually the pivotal stage in the system to which a sizable portion of processing is devoted and a considerable share of recognition errors is attributed. The most notable feature of Arabic writing is its cursiveness. Compared to other features, the cursiveness of Arabic words poses the most difficult problem for recognition algorithms. In this work, we describe the design and implementation of an Arabic word recognition system. To recognize a word, the system does not segment it into characters in advance; rather, it recognizes the input word by detecting a set of "shape primitives" on the word. It then matches the regions of the word (represented by the detected primitives) with a set of symbol models. A spatial arrangement of symbol models that are matched to regions of the word, then, becomes the description of the recognized word. Since the number of potential arrangements of all symbol models is combinatorially large, the system imposes a set of constraints that pertain to word structure and spatial consistency. The system searches the space made up of the arrangements that satisfy the constraints, and tries to maximize the *a posteriori* probability of the arrangement of symbol models. We measure the accuracy of the system not only on words but on isolated characters as well. For isolated characters, it has a recognition rate of 99.7% for synthetically degraded symbols and 94.1% for scanned symbols. For isolated words the system has a recognition rate of 99.4% for noise-free words, 95.6% for synthetically degraded words, and 73% for scanned words.

Key words: Arabic characters – Optical character recognition – Segmentation – Cursive script – Degraded text recognition

Correspondence to: R.M. Haralick

1 Introduction

Since the advent of writing as a form of communication, stone, papyrus, and then paper have prevailed as the media for writing. Only recently have electronic media started to replace paper. Because of its improving space efficiency and its increasing speed of access, the use of electronic media is constantly on the rise. Paper's widespread use for communication and archiving, and the amount of information already on paper, press for quick and accurate methods to automatically read that information and convert it into electronic form.

Optical character recognition, OCR, is the branch of technology that deals with the automatic reading of text. The ultimate goal of OCR is to imitate the human ability to read – at a much faster rate – by associating symbolic identities with images of characters. As the emphasis shifts from recognizing individual characters to recognizing whole words and pages, more general terms being used include *optical text recognition* and *document image processing.*

This paper discusses a symbol recognition system that recognizes segmented noisy and cursive text words. To recognize symbols of a word, the system does not segment the word into characters in advance; rather, it recognizes the input word by detecting a set of "shape primitives" on the word. It then matches the regions of the word (represented by the detected primitives) with a set of symbol models. A spatial arrangement of symbol models that are matched to regions of the word, then, becomes the description of the recognized word. Since the number of potential arrangements of all symbol models is combinatorially large, the system imposes a set of constraints that pertain to word structure and spatial consistency. The system searches the space made up of the arrangements that satisfy the constraints and tries to maximize the *a posteriori* probability of the arrangement.

1.1 Features of Arabic script

The inadequate research activity on Arabic OCR can be attributed in part to the difficulties Arabic poses for

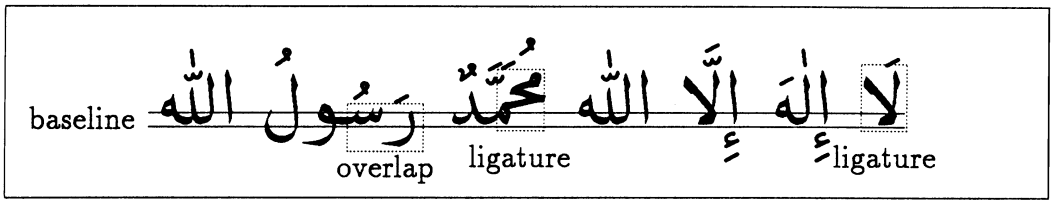


Fig. 1. A sample of written Arabic showing some of its characteristics

recognition. The calligraphic nature of Arabic script sets it apart from other languages in several ways:

- Arabic has 28 characters, of which 16 have from one to three dots above or below them. Dots differentiate otherwise similar characters. Additionally, three characters can have a zigzag-like complementary character (*Hamza* ء).
- Arabic text is read and written from right to left.
- Written Arabic is cursive. Within a word, some characters connect to the preceding and/or following characters, and some do not connect. Thus a word can have one or more connected components.
- The shape of an Arabic character depends on its position in the word. A character can have up to four different shapes depending on whether it is isolated, connected from the right (beginning form), connected from the left (end form), or connected from both sides (middle form).
- A distinguishing feature of Arabic writing is the presence of the baseline. The baseline is a horizontal line that runs through a connected portion of a word, and is formed from the concatenation of adjacent characters. Almost all points of connection between characters fall on this line.
- Characters in a word might overlap vertically (without touching) depending on their shapes.
- Arabic characters vary in size (height and width), even for different shapes of the same character.
- Several characters can combine vertically to form a ligature (combined character).
- The margins of typeset Arabic are justified by elongating the baselines of the words of a line, not by inserting inter-word spaces, as in English. This is accomplished by inserting the elongation, *tatweel*, symbol (ـ) at different places in the word. For example, the word (ســالمــين) is an elongated version of the word (سالمين).

Figure 1 demonstrates some of these characteristics on a typeset Arabic sentence consisting of seven words. Reading from right to left, the first word is a ligature made up of two characters, and the second word consists of three characters and two connected components. The short strokes at the top of the text are diacritic marks. The ligature in the middle of the figure consists of three vertically stacked characters.

Some of these characteristics greatly complicate recognition. One of the hardest problems with Arabic is its cursiveness; this is why segmentation is a crucial step for many Arabic character recognition systems. Many recognition errors are attributed to the segmentation phase, and a large portion of processing time is allocated to it. Recognizing isolated Arabic characters is not fundamentally different from recognizing Latin text (apart from the larger number of classes of Arabic). Surveys on Arabic recognition include those by Amin [6], Shoukry [38], and Jambi [25]. More recent and comprehensive surveys appear in [2] and [3]. Trenkle et al. [36] discusses a recent Arabic recognition system.

1.2 Motivation

In Arabic word recognition, the ultimate objective is to correctly recognize the symbols of a given word. Since Arabic words are cursive, it is not easy to determine where one symbol ends and the next one begins. In fact, it is possible for a symbol to end after the beginning of a succeeding symbol, as adjacent symbols can overlap as well as touch (as can be seen in Fig. 1).

Some recognition systems attempt to deal with the connectivity of Arabic text by performing two steps in order: (1) they segment a word into symbols first and (2) they then recognize the segmented symbols. Examples of this approach are in [7,14,26,28,30]. The difficulty of segmenting the symbols of a word leads to inaccuracies in the segmentation and, as a result, in the recognition. To overcome the difficulty in isolating the symbols, some systems over-segment an input word into pieces possibly smaller than symbols. It is up to the subsequent recognition phase to determine the true segmentation points. Under this approach, the recognition phase usually recognizes the symbol parts and then reassembles them to make symbols. Works that use this approach include [1,5,11,12,31,27,37,39,43].

Another approach is to scan the word in one direction (e.g., right to left) and assume that a consecutive set of image columns is a symbol and try to recognize that symbol, as in [33,35]. Whenever a symbol is recognized, the recognizer starts at where it left off and repeats the process. In this approach, if the system is unable to recognize a symbol that is in the middle of the word, it starts recognition from the other end and tries to recognize the symbols in reverse order.

A better approach is the recursive segmentation recognition approach [13]. Whenever the recognition step fails to recognize a segmented piece of text (symbol or part of a symbol), the word is re-segmented, and the process is repeated. The last approach is that of whole word recognition (e.g., [8]). This requires using features of the whole word and recognizing the whole word as a

unit. To do this, a system must be trained to recognize the shapes of all the different words it should recognize.

The problem with all of the above approaches is that there is no recourse if a symbol is recognized incorrectly, particularly when the error is due to segmentation. The preceding approaches find the solution that is best on a symbol-by-symbol basis, or that is best for the particular local area that contains (at most) a pair of symbols. Another way to state the problem is that these approaches are sequential: the recognition of a subsequent symbol depends on its correct segmentation and sometimes on the recognition of its preceding symbol.

The solution discussed here is to optimize both segmentation and recognition with respect to the whole word. The emphasis is on finding a recognition solution that maximizes an *a posteriori* word probability. The system uses a state-space search to find the best recognition of the word. In our approach, symbol recognition precedes boundary detection. Because symbol boundaries are not known, the search successively proposes sets of symbol boundaries. A competing approach to the one discussed here is the hidden Markov Model approach [29].

2 Problem statement

This section formalizes the word recognition problem as a state-space search problem. Let $\mathcal{P}$ be a set of predefined primitive types. Let $\mathcal{L}$ be a set of symbol classes, which is the symbol set that the system recognizes. When recognizing printed characters, it is the set of all the shapes to be recognized. A *symbol* is defined to be the shape or glyph of a character. In Arabic, a letter might have up to four shapes. Each symbol class has a model that defines it in terms of instances of the primitive types, $\mathcal{P}$, and their locations relative to one another. Let $\mathcal{M}$ be the set of symbol models that correspond to the symbols in $\mathcal{L}$. Further, let $\mathcal{X}$ be the set of all points in the Cartesian space, $\mathcal{X} = \mathcal{Z} \times \mathcal{Z}$, where $\mathcal{Z} = \{0, 1, 2, \dots\}$.

When recognizing a particular word, the input to the search problem is the image of the word, I, and the set, $\mathcal{S}$, of primitive instances detected on the word.

The system's goal is to spatially arrange the symbol models in the word space and find the spatial arrangement of symbol models with the maximum *a posteriori* probability. The system achieves its goal by matching symbol models with local regions in the word image.

Posing the problem as a state-space search problem, Σ, we can characterize it by four components:

$$\Sigma = \langle \mathcal{T}, \omega, s, \mathcal{G} \rangle.$$

The components have the following meaning:

- $\mathcal{T}$ is a set of states, $\mathcal{T} : \mathcal{E} \times \mathcal{S}$, where $\mathcal{E} = \langle \mathcal{M} \times \mathcal{X} \rangle$ is a sequence of zero or more pairs of symbol models and translations.
- ω is an operator that operates on a state and returns a set of states, $\omega : \mathcal{T} \to 2^{\mathcal{T}}$.
- $s \in \mathcal{T}$ is the start state.
- $\mathcal{G} \subseteq \mathcal{T}$ is a set of goal states.

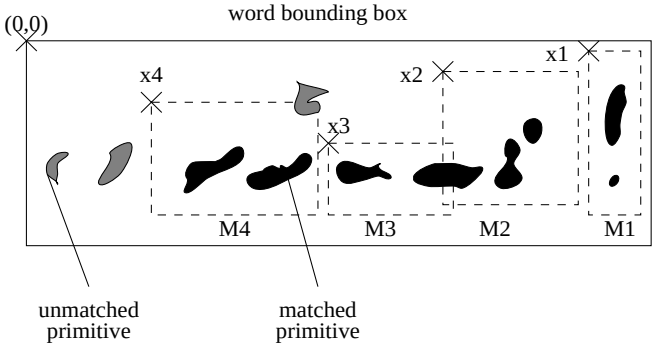


Fig. 2. The components of a state. The models are indicated by boxes with dashed lines

Each state t has two elements: (1) a sequence of pairs of models and their translations and (2) a set of unmatched primitives, as follows:

$$t = (\langle M_1, x_1 \rangle, \dots, \langle M_n, x_n \rangle, S)$$

where n is the number of symbol models already arranged (matched) in state t. The set S is the set of primitive instances from $\mathcal{S}$ that is not yet matched. As such, a state specifies a string of characters that is a (partial) solution to the recognition problem. This sequence of n models represents a string of symbols or a potential word that is ordered in word reading order.

Figure 2 illustrates the components of a state. The state has four matched models, $M_1, \dots, M_4$, indicated by dashed boxes. Their respective translations, $x_1, \dots, x_4$, are relative to the origin of the word's bounding box. The set of unmatched primitives, S, consists of the gray primitives. The partial word represented by the string of models is $l_1 \dots l_4$, where l_i is the symbol class of model M_i.

The start state, s, which is the root of the search tree, has no arranged symbol models. Its set of unmatched primitives equals the whole set of detected primitives, $s = (\emptyset, \mathcal{S})$.

Applying the operator ω to state t returns a set of states R. A state r in R uses a group of primitive instances from S to match an additional symbol model to a word region. This means that a descendent state r has one more translated model than its parent state t. Therefore:

$$r = (\langle M_1, x_1 \rangle, \dots, \langle M_{n+1}, x_{n+1} \rangle, T),$$

where T is the set of primitives remaining after matching model M_{n+1} at translation x_{n+1} with the primitives in set S.

A goal state $u \in \mathcal{G}$ is a leaf of the tree. The string of models of a goal state, u, must satisfy the word structure constraints specified below.

In this characterization of the recognition problem, segmentation and recognition are interleaved, while maintaining a global view of the process. The rest of this paper describes the design and implementation of experiments conducted on a system that recognizes Arabic words using the word optimization recognition methodology. The system has three major components, shown as

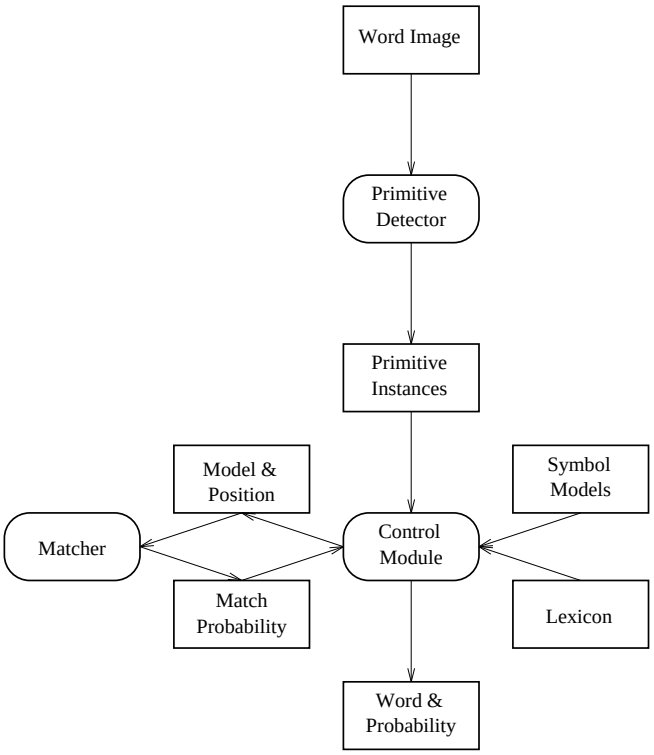


Fig. 3. Block diagram of the recognition system. The ovals designate processes, while the rectangles designate data structures

ovals in Fig. 3: (1) the primitive detector, (2) the control module, and (3) the matcher.

To recognize a given word, the system detects a predefined set of shape primitives on the isolated image of the word. Preprocessing and primitive detection are explained in detail in Section 3.

The control module takes the detected primitives of a word and hypothesizes a number of alternative strings as the recognition of the word. It then chooses the one that maximizes an *a posteriori* probability by conducting a state-space search in the space of symbol model arrangements.

At any particular point in the search, the control module decides on (1) a set of symbols that is expected at that location and (2) a location where it should match the next symbol model. It then asks the matcher to match each symbol model in the set with that particular region in the word. The control model is explained in detail in Section 4.

The matcher operates in two major modes: the training mode and the testing or recognition mode. During training, the matcher processes a training set comprised of a large number of degraded symbols that are labeled with their identity. The training mode essentially teaches the system the description of each symbol in terms of primitives.

During execution, the control module gives the matcher a symbol model and a translation of the model onto the word image. The matcher's job is to compute the probability of a match between the symbol model and the word image at the particular translation. The matcher is explained in detail in Section 5.

Prior work on search and model matching for handwriting recognition applications can be found in [44,45]. Some later work can can found in [16,23]. For the idea of using primitives in cursive script recognition see [32]. Fang and Hull [15] suggested the use of the A* algorithm to accomplish the search.

3 Preprocessing and primitive detection

The word recognition system is designed to recognize isolated Arabic words on binary images. Thus, it is assumed that input words have been isolated and have minimal skew.

The system uses information about the baseline of text to arrange the symbols. The baseline of Arabic text is the line at which the characters connect to one another. It usually coincides with the row of the word image with the highest density of black pixels. For that, the system detects the baseline using the horizontal projection profile.

The task of the primitive detector is to find instances of a set of *shape primitives* on a text image. Instances of primitives are found by applying the erosion morphological transform to an input block. A shape primitive is usually a small connected set of pixels that has a simple geometric structure. A shape primitive represents a structuring-element for erosion.

To detect instances of a shape primitive on a text image, the primitive detector morphologically erodes the image using the shape primitive as a structuring-element. This produces a new image, with instances of the shape primitive showing up as blobs at different locations. Each blob, which is a connected set of pixels, is termed a *primitive instance*. Each pixel in a blob specifies the location where an occurrence of the primitive was detected on the image. Gillies [18] and Stentiford [41] also use mathematical morphology to extract primitives. Other approaches that are more graded include [17,42].

Figure 4 shows the letter "A" and the structuring-elements for three shape primitives, which can be used to recognize the letter. Figure 5 shows the three images that result from detecting each of the three shape primitives on the image of the letter "A". In this example, each shape primitive had exactly one instance (blob).

Information about the blobs is extracted from the resulting image by a connected components labeling operation. The labeling operation assigns a unique label to each adjacent set of foreground pixels. Here we use eight neighbors, which means that diagonal pixels are taken to be neighbors.

The relevant information about a primitive instance (blob) is the location of its centroid and its area in pixels. Hence, the application of the operator that defines shape primitive p to a text image results in a number of primitive instance tuples of the form (p, r, c, a), where (r, c) are the row and column coordinates of the centroid of the primitive instance relative to image coordinates

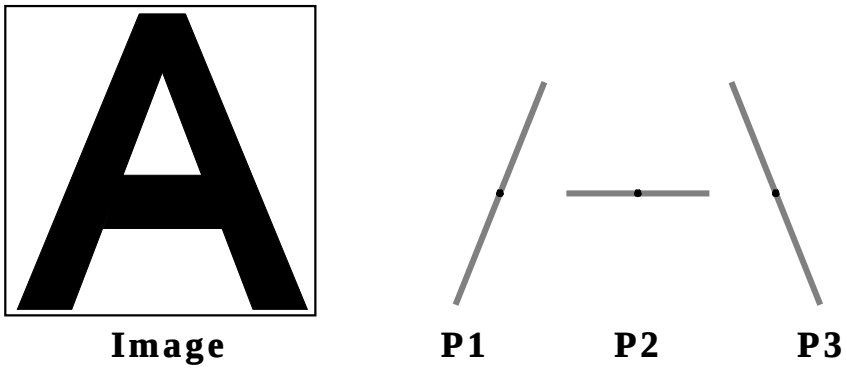


Fig. 4. An example of the definition and detection of shape primitives. On the left is an image of the character "A", and on the right are structuring-elements that designate three shape primitives, $P1$, $P2$, and $P3$, with their center points marked

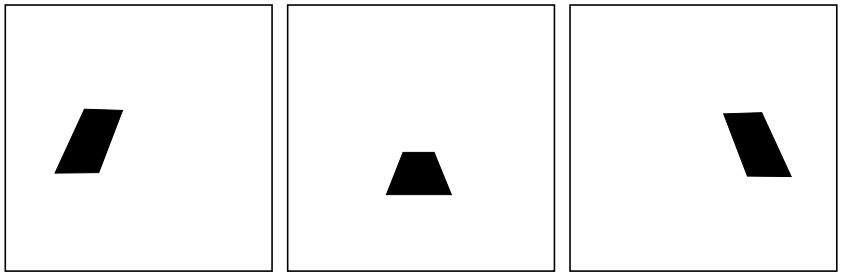


Fig. 5. The result of eroding the above image with each of the three structuring-elements, $P1$, $P2$, and $P3$. The blobs are the detected instances of the shape primitives

(the upper left corner), and a is the area of the instance in pixels.

In this system, shape primitives, or structuring-elements, are specified symbolically. A structuring-element can be a line, corner, or arc. As an example, a corner is defined by specifying the lengths of its two lines, the angle of the first line relative to the horizontal, and the angle of the second line relative to the first line. The origin of a corner structuring-element is the intersection (origination) point of the two lines.

The shapes of the primitives (or structuring-elements) that we use include: lines of different lengths and angles; corners (where each corner is formed by two lines that share an endpoint) of different line lengths and angles; elliptic arcs of different radii and beginning and extent angles; and circles and disks of different radii. Figure 6 shows a set of shape primitives.

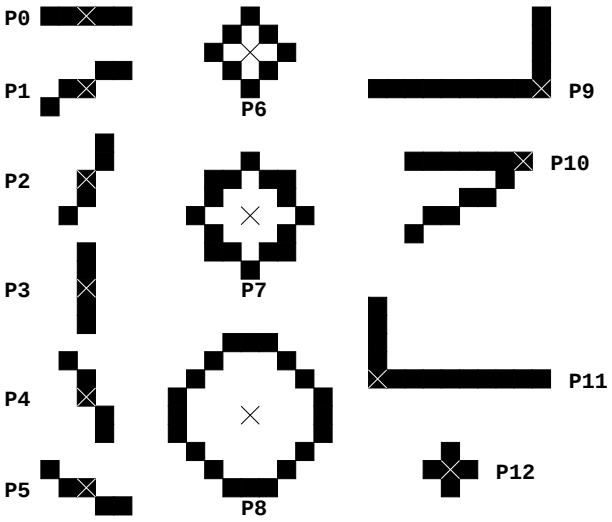


Fig. 6. The shape primitives (structuring-elements) used by the system. Each square corresponds to an image pixel. The origin of each shape is indicated by a cross

4 The control module

When recognizing a particular word, the input to the search problem is the image of the word, I, and the set of primitive instances detected on the word, $\mathcal{S}$. The task of the control module is to search for the spatial arrangement of symbol models with the maximum *a posteriori* probability. This spatial arrangement of models specifies a string of symbols or a recognition of the word.

The space to be searched is that of sets of translated symbol models. Since this space is exponentially large, the search algorithm must be efficient and effective in finding a good solution.

The search algorithm used here is a variant of depth-first branch-and-bound. It employs a list size cutoff to reduce the space of the search. The cutoff reduces the branching factor of the search and hence reduces the base of the exponent for the worst-case time complexity.

This strategy differs from beam search. In our approach, the cutoff is applied locally to the ordered list of descendents of a state; in beam search, the cutoff is applied to the open list, which is an ordered list of all nodes that have been generated but not expanded. When the cutoff is used, neither search strategy is guaranteed to find the optimal solution. The size of the list can be experimentally varied up to infinity (no cutoff) to find the best performance/time.

The tasks undertaken by search algorithm include:

1. Expanding a state and generating its successors by applying the operator ω to the state.
2. Checking if a state n is a goal state by satisfying the word constraints, $W(n) = \text{true}$.

3. Computing the *a posteriori* probability of a goal state n (word), $\Pr(n)$.
4. Ranking a set of states using the state ordering function τ.

Before advancing to the next sections that address these constraints and functions, we provide the following definitions that are used in the discussion. A state t in the search space has two components: a sequence of model-translation pairs and a set of unused primitives:

$$t = (\langle M_1, x_1\rangle, \ldots, \langle M_n, x_n\rangle, S)\,.$$

Let the name of the symbol represented by model M be denoted as $l(M)$. The string of symbols represented by the symbol models at state t is then $l(M_1)\ldots l(M_n)$.

State expansion is the process of applying the operator ω to a state t and generating a set of new states R. In this problem domain, state expansion is the process of creating a set of new states, where each state has an additional new model-translation pair added to the end of the string of models of the original state. This involves: (1) examining the region of the word image as well as the primitives that are unused by previous symbol models and (2) proposing a set of translated symbol models that are compatible with the current state.

The strategy of the search is to suggest models in a right-to-left order, so that the first model of a state will correspond to the first symbol of the word and the last model will correspond to the last symbol of the word. At the state-expansion level, this translates to proposing symbols in locations just to the left of the leftmost symbol model in the current state.

The space of all possible symbol model-translation pairs is very large. Further, many pairs are incompatible with parent states or are very unlikely to be part of a correct Arabic word. To reduce the number of searched states, we use a large lexicon of Arabic words and incorporate a number of constraints on word structure and the spatial arrangement of models. The advantage of using the word constraints at the state expansion level is to reduce the number of searched states. Kimura et al. [39] based their algorithm for unconstrained handwriting using a lexicon. Shridhar et al. [24] also found it useful to use a lexicon in his comparison of lexicon-free and lexicon-directed word recognition for handwritten words.

The lexicon constrains state expansion to propose symbol models that will construct words of the lexicon. It is customary in lexicons and dictionaries of Arabic to sort words in the root-pattern form, after stripping the prefixes and suffixes. This method is more compact than spelling out the words but is more complex when looking up words. Here, the lexicon as a list of words represented as a trie [40]. A base word might be included more than once to represent the word with different prefixes and/or suffixes. The Arabic lexicon that we use here is constructed from two sources: a corpus of about 212 000 words, and a set of 25 pages that were entered in-house. The lexicon has a total of over 42 000 different words. For comparison, the Unix spell command has around 50 000 (unique) words [9]. The sources of the corpus are Arabic-language newspapers, news magazines, and books. The root of the lexicon trie is the empty string and the children at the first level are all possible first characters of words. Subsequent levels branch off at character positions where lexicon words differ.

Table 1. The structural compatibility table for symbol models

Model n	Model $n+1$
$\emptyset$, digits, punctuation	Isolated and beginning forms, digits, punctuation
Beginning and middle forms	Middle and ending forms
Isolated and ending forms	Digits, punctuation

Figure 7 shows an example of a trie for nine Arabic words. As used here, the trie can be thought of as a finite state machine for recognition. Recognition starts at the empty string. Whenever a symbol is proposed, it advances to its state in the trie. Recognition can stop at any state that designates the end of a word (indicated by a double oval in the figure). Thus, when expanding a certain state, the symbol models to be proposed correspond to the characters that are the descendents of the trie node of the parent state.

The constraints on word structure include:

- The font type and size must be consistent within a word.
- A word must begin with a beginning form and end with an ending form.
- All adjacent positional shapes must be compatible within a word.

Word structure constraints restrict the set of symbol models proposed to those compatible with the last (leftmost) model in the parent state. Table 1 shows the types of symbol models that are proposed (second column) based on the last symbol model in the parent state being expanded (first column). An additional model that is considered a middle form is the elongation (*tatweel* ـ) symbol. The system permits a *tatweel* to occur between any pair of connecting letters.

Spatial constraints govern the location of the bounding boxes of symbol models relative to one another and to the word's bounding box. The spatial constraints specify that:

- The characters of a word are collinear; their baselines must coincide with the word's baseline.
- Bounding boxes of adjacent symbols have minimal overlap with one another.
- Bounding boxes of adjacent symbols have only small gaps between them.
- Bounding boxes of symbols are almost totally enclosed in the bounding box of the word.

Spatial constraints reduce the number of searched states because they remove states that will have low match probability. For example, one of the spatial constraints is to ensure that two symbol bounding boxes do not coincide. When two models coincide, the later one is bound to have a very low match probability because many of its primitives will have been used.

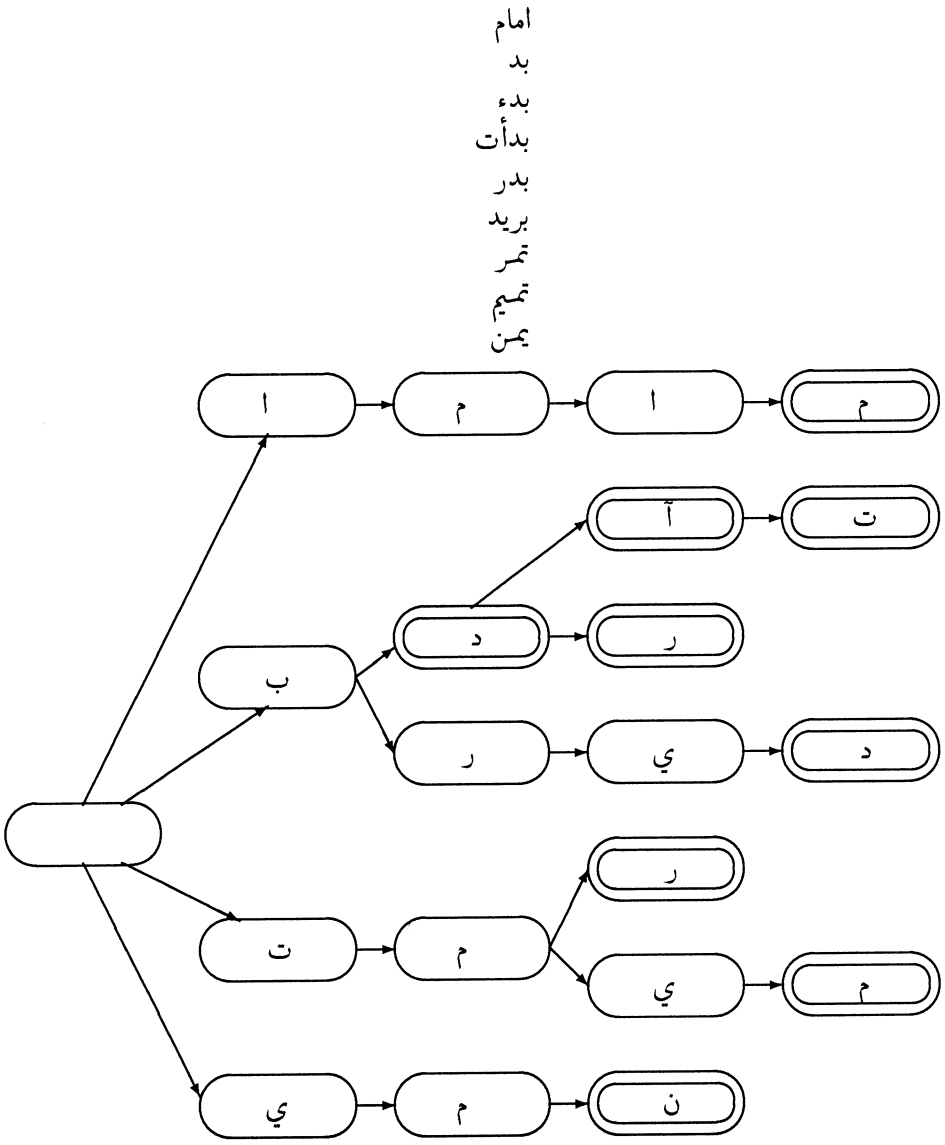


Fig. 7. An example of a trie for a set of nine Arabic words (shown at the top). The nodes with double ovals designate complete words

Now that we have a set of proposed symbol models that are structurally sound and can lead to a lexicon word, we propose translations for the models. To limit the number of descendent nodes to those likely to lead to a good solution, each proposed model will have only one translation. To find the best translation for a given model, we use the spatial constraints to find the most likely position of the model.

Since there is no guarantee that a proposed position, (x, y), of the symbol model is the best position, we try the set of adjacent positions $\{(x \pm m, y \pm n) | 0 \leq m, n \leq i\}$, where i is a small integer. At each of these translations, we match the model and retain the translation that produces the largest match probability. The assumption here is that the local optimization is likely to lead to the global optimum. The price paid for reducing the number of translations per model to one is possibly forsaking the optimal solution for a suboptimal one.

A goal state of the search tree is an arrangement of symbol models that corresponds to a word. This section addresses the constraints on a state that make it a goal and the *a posteriori* word probability model that is used to evaluate goal states.

Since the states are expanded under the lexicon, structural, and spatial constraints, the only remaining conditions that make a state a goal state are that the characters of the state make a proper lexicon word (and not only a subset) and that the last symbol model is an ending form.

In this problem, note that a goal state is not necessarily a leaf state. A string that might qualify as a word could be a substring of other words. This happens when the ending form and middle form of the last character of the word are the same as in the word "بد", which is also a substring of the words "بدء" and "بدأت", as can be seen in Fig. 7. For this reason, when such states satisfy the goal constraints, they should be searched further.

Furthermore, the first correct word to be found in the search might not be the best word. So reaching a goal state does not qualify as the termination condition. The algorithm must search all nodes that are not pruned. For

this reason, it is important for the search algorithm to be effective in pruning as many states as possible, but it must take care not to prune a state that leads to that best solution.

The system computes the word *a posteriori* probability for states that satisfy the word constraints. The *a posteriori* probability of such a state is the joint probability of each of its symbol models matching the region onto which it has been translated. The *word probability* is expressed as:

$$\Pr\left(\{\langle M_1, t_1\rangle, \ldots, \langle M_n, t_n\rangle\}, S \,|\, I\right).$$

The probability of each symbol matching the word region onto which it has been translated is determined by the matcher and is returned on an independent call to the matcher. Hence, we take the probability of each symbols' match to be independent of the other symbol matches. (The only interdependence results from restrictions on model selection and translation that an earlier model poses on a newer one.) If we assume that the symbols were matched in increasing subscript order, then the probability distribution is:

$$\Pr\left(\{\langle M_1, t_1\rangle, \ldots, \langle M_n, t_n\rangle\}, S \,|\, I\right) = \prod_{i=1}^{n} \Pr\left(\langle M_i, t_i\rangle \,|\, I\right) \times \Pr(S \,|\, I). \quad (1)$$

The first term on the right is the product of the model match probabilities. The model match probability distribution is detailed in Section 5.3.

When computing the word probability of a goal state, the detected primitives that remain unmatched are considered as extraneous primitives. The probability of the extraneous primitives, $\Pr(S \,|\, I)$, is the product of the probabilities of the individual extraneous primitives. Let the primitive instance $(p, r, c, a) \in S$ be an extraneous primitive instance, where p is its type, (r, c) is its centroid, and a is its area. We assume that extraneous primitives have a distribution that is similar to the primitive match distribution:

$$\Pr((p, r, c, a)) = k(\delta)\mathrm{e}^{-\delta a}, \quad (2)$$

where δ is the parameter of the distribution and $k(\delta)$ is the probability-normalizing constant. We make δ the minimum value for the exponential parameter, α, of the match probability over all primitives and models (Section 5.3.2), and we divide by an empirically determined constant.

The state ranking function, τ, is essentially a heuristic function that allows selecting states that are more likely to lead to the optimal solution. In this problem we are not interested in the goodness of the path to the solution, but in the goodness of the solution itself.

The figure of merit for a goal in this problem domain is the *a posteriori* probability of the goal state, formulated in Equation 1. Maximizing the merit of a solution is equivalent to minimizing the cost of the solution with a negated heuristic function [34].

For a heuristic function to be guaranteed to lead the search to the optimal solution, it should not underestimate the merit of any state. When a state is thought to be worse than it really is, the state that leads to the optimal solution might be pruned. This concept is analogous to the admissibility criteria of the heuristic of the A* algorithm.

Since the objective here is to maximize the probability of the solution, we compose a heuristic function of two parts: (1) $g(n)$, the actual match probability of the models in node n and (2) $h(n)$, an (over)estimate of the match probability of the remaining part of the word. For the heuristic to fulfill its rule in speeding up the search, it must be much faster to compute than expanding the node. For that, we define the probability per width, pw, of a symbol model, m, to be the ratio of the maximum possible match probability of m (i.e., under ideal conditions) to the width of the model. The match probability for a symbol model is maximal when it is matched with a noise-free image of the symbol.

The maximum probability per width over all symbol models is

$$mpw = \max_{m \in \mathcal{M}} pw(m).$$

We then take the estimate of the match probability of the remaining part of the word, $h(n)$, to be the product of the width of the yet unmatched part of the word, $w(n)$, times the maximum probability per width:

$$h(n) = w(n) \times mpw. \quad (3)$$

This assumes that the remaining part of the word can be matched with multiple instances of the symbol model that have the highest probability per width ratio, which is clearly an overestimate of the match probability. Putting all the parts together, the state evaluation function for state t is:

$$\tau(t) = \left[\prod_{i=1}^{n} \Pr\left(\langle M_i, t_i\rangle \,|\, I\right)\right] \times \left[w(t) \times mpw\right], \quad (4)$$

where $w(t)$ is the width of the unmatched part of the word at state t. By taking the log of τ, we can see the similarity between it and the A* heuristic:

$$\log(\tau(t)) = \underbrace{\sum_{i=1}^{n} \log\left(\Pr\left(\langle M_i, t_i\rangle \,|\, I\right)\right)}_{1} + \underbrace{\log(w(t) \times mpw)}_{2}. \quad (5)$$

Term 1 is the actual merit of the part of the word already matched in the state, while Term 2 is an overestimate of the merit of the expected match with the remaining portion of the word.

Figure 8 shows a block diagram of the control model. As mentioned earlier, the algorithm terminates when it searches all the states of the space that are not pruned. When the algorithm returns, it prints out the best word as the solution to the recognition problem. The next section discusses the operation of the matcher.

5 The matcher

The matcher compares a certain symbol model to a region in a word image onto which it has been translated.

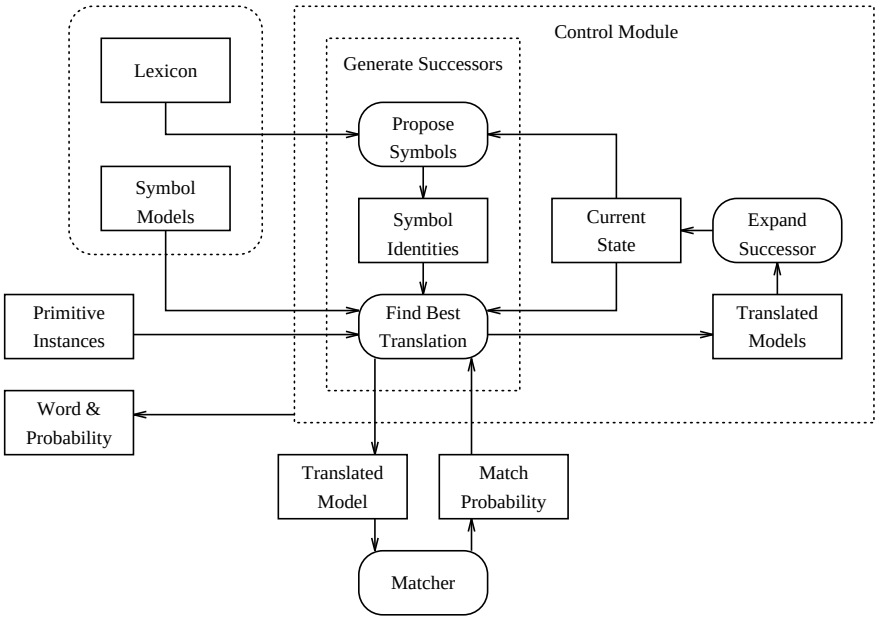


Fig. 8. Block diagram of the control model. The ovals designate processes, and the rectangles represent data structures

Called by the control model, the matcher is expected to return the probability of the symbol model matching the region. This section explains the components of symbol models, details how they are used in matching word regions, and describes how the match probability is computed.

5.1 Symbol models

The model for a particular symbol is a complete description of the primitives of an ideal image of the symbol, termed the *ideal primitives*, and the deformations expected under an assumed noise model. The symbol model contains the information necessary for the matcher to measure the similarity between its primitives and those of the match region of the word. The *match region* of the word is defined as the region covered by the bounding box of the symbol model when it is translated onto the word image. The matcher works by finding the correspondence between the primitives of the symbol model and the detected primitives that are in the match region of the word.

To find the detected primitives that correspond to a particular model primitive, each model primitive has a *correspondence region*. The *correspondence region* of a model primitive is a region that includes the expected deformations of the primitive, under the assumed noise model.

A symbol model contains a bounding box and a set of primitive specifications. The model, M, of a symbol is defined as:

$$M = (h, w, \{(p_i, r_i, c_i, A_i, R_i)\}_{i=1}^{n}) .$$

Components of the symbol model are:

- The height and width of the bounding box of the symbol, (h, w)
- The description of a set of n model primitives, each having the following components:
  - The shape primitive, p
  - The location of the centroid of the primitive relative to the model's upper left corner, (r, c)
  - The area of the ideal primitive, A
  - The correspondence region of the primitive, R.

The number of primitives per model, n, can vary from model to model.

The components of a symbol model are found by processing an ideal image of the symbol. The bounding box is taken to be the smallest box that includes the image of the symbol. The ideal primitives are found by detecting all the primitive operations of $\mathcal{P}$ on the ideal symbol image. The resulting connected components are the ideal primitives. In Fig. 9, (a) is the ideal image of a symbol. Some of the ideal primitives of the symbol are shown in (d) and (h) for primitives (c) and (g), respectively. Looking at the connected components in (d) and (h), the symbol has two ideal primitives of shape primitive (c) and four of primitive (g).

Computing the correspondence region of a primitive depends upon an assumed underlying process that generates noise. From observing a large number of degraded documents, one notices that the most visible degradation to words occurs near the boundary of text, and that the intensity of degradation is inversely proportional to the distance from the boundary. Degradation here refers to the occurrence of artifacts that are not part of the ideal text image.

The advantage of using a noise-generation model is the ability to predict the characteristics of the observed noise. The noise-generation model used here is the one explained in [21]. In that noise model, the probability of a pixel changing value due to noise is inversely proportional to its distance from the boundary of text. Further, noise is correlated, in that noise usually affects groups of adjacent pixels.

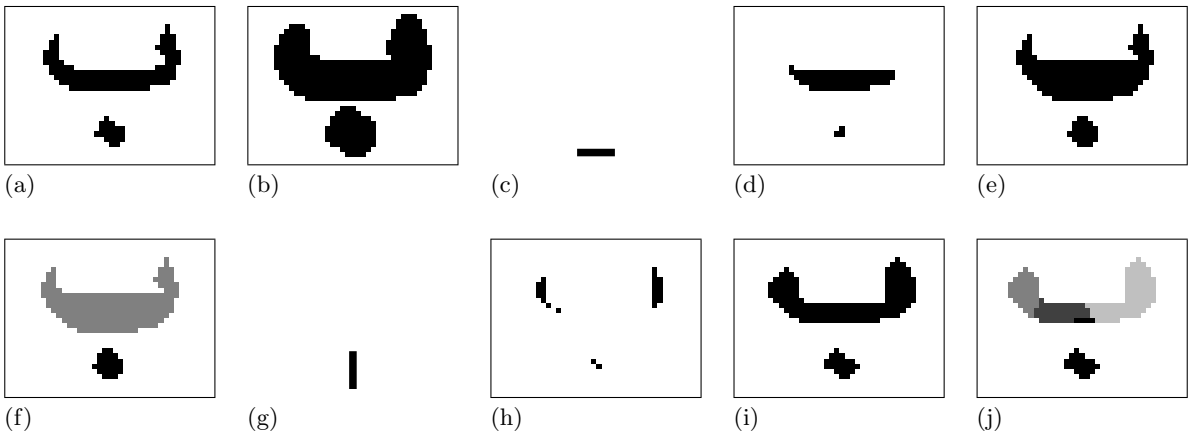


Fig. 9a–j. Generating the model of the isolated form of symbol Ba. **a** The ideal image of the symbol. **b** The result of dilating image **a** with a disk of radius 2. **c** A horizontal line structuring-element (shown enlarged). **d** The result of eroding image **a** with element **c**. **e** The result of eroding image **b** with element **c**. **f** The result of labeling image **c** with a unique label for every correspondence region. **g** A vertical line structuring-element (shown enlarged). **h** The result of eroding image **a** with element **g**. **i** The result of eroding image **b** with element **g**. **j** The result of labeling image **i** with a unique label for every correspondence region

This noise model is used to degrade text images when training the matcher and when testing the performance of the system. In Fig. 10, (a) shows an ideal paragraph of text, while (b) shows the same paragraph after being degraded with noise generated according to the noise model.

The *noise radius* of text is the range of distance from the boundary in which noise is likely to occur. The noise radius defines the correspondence region of primitives and is estimated from degraded images of text.

To determine the correspondence region of a model primitive, one works from the ideal image, corrupts it with noise that conforms to the specifications of the noise-generation model, and finds the effect on the primitive. Assume that the ideal image is I, the structuring-element is S, and the ideal primitive is P. To simplify notation, we first assume that there is one and only one instance of the primitive in the image; we later relax this assumption. The primitive is computed as:

$$P = I \ominus S.$$

The correspondence region, R, for an erosion primitive, P, and noise radius, r, is computed by dilating the ideal image with a disk of radius r and applying the primitive to this expanded image. Formally:

$$R = (I \oplus D_r) \ominus S, \tag{6}$$

where D_r is a disk of radius r.

Figure 9 shows the process of computing the correspondence regions for some primitives of a symbol. The two correspondence regions of the primitives in (d) are shown in (f), labeled with different shades. Likewise, the four correspondence regions of the primitives in (h) are shown in (j). Image (f) arises from image (e) by labeling the pixels with the identity of the nearest region of (d) (and likewise for sequence (h), (i), and (j)).

In the next sections, we discuss how to use a symbol model to match a word region and then how to estimate the parameters of and compute the match probability.

5.2 The matching process

This section addresses how to compute the match measurements for the match between a symbol model, $M = (h, w, \{(p_i, r_i, c_i, A_i, R_i)\}_{i=1}^{m})$, translated by point t and a region of a word image I. The bounding box of a symbol model that has been translated onto a word image delineates a match region in the image.

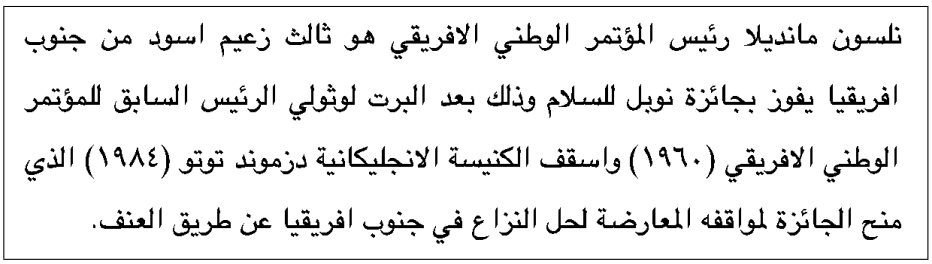
The process of model-region matching entails matching the detected primitives on the region to the model's primitives. A particular model primitive matches the detected primitives, or the parts of them, that are both: (1) of the same shape primitive and (2) fall into its correspondence region.

When matching a particular model primitive, its match measurement is the total area (in pixels) of the intersection between its correspondence region and the detected primitives. The detected primitives, or parts of them, that are matched are considered to be used or consumed. They cannot be used in subsequent matches. An example of the process of measuring the match is displayed in Fig. 11 for two correspondence regions, $R1$ and $R2$, that are to be matched with two detected primitives. In the figure, the match measurement for correspondence region $R1$ is $a_1 = C4$. The measurement for $R2$ is $a_2 = C_1 + C_2$. The area $C3$ is the unmatched part of $P1$.

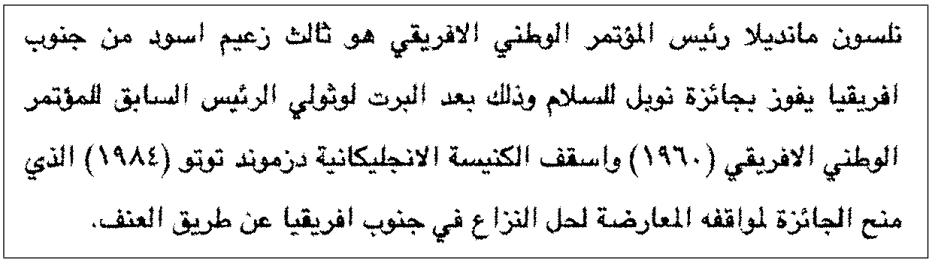
Figure 12 shows a scanned word being matched with the model of symbol . The match measurements are the number of pixels in from the black areas in the images.

5.3 The model-region match probability

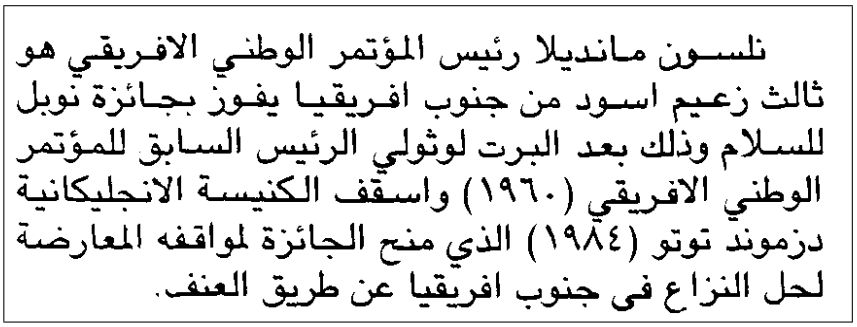
When matching a symbol model to a word region, and after calculating the match measurements for each model primitive, the matcher must compute and return the match probability.



(a)



(b)



(c)

Fig. 10a–c. Samples of input images: **a** ideal (noise-free) text; **b** synthetically degraded text (at degradation parameters $\alpha = \beta = 1.5$, $\alpha_0 = \beta_0 = 1.0$, $c_0 = 0$, and $e = 3$); and **c** scanned text

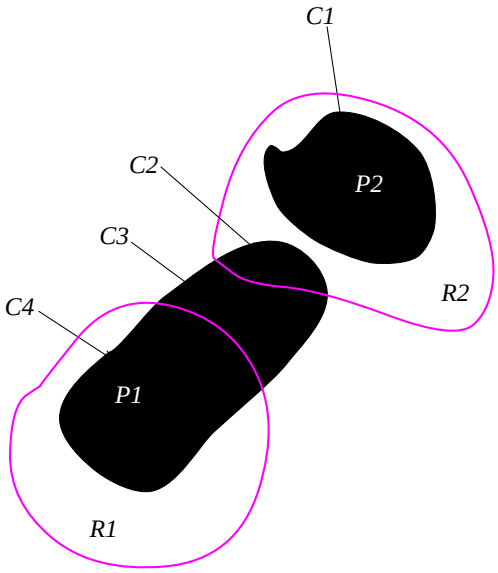


Fig. 11. Matching two detected primitives, $P1$ and $P2$, with two correspondence regions, $R1$ and $R2$

Assume that the matcher matched the N_m primitives of symbol model m, and that the match measurements for the primitives were $a_1, \ldots, a_{N_m}$. Let $s_1, \ldots, s_{N_m}$ be indicator variables, such that:

$$s_i = \begin{cases} 1 & \text{if } a_i > T_i \\ 0 & \text{otherwise,} \end{cases}$$

where $T_i \geq 0$ is the indicator threshold that is experimentally set in the next section. Typical values of T_i are 0 and $A_i/2$, where A_i is the ideal area of the primitive. After matching, we perform a change of variables and rewrite the model match probability as:

$$\Pr(\langle m, t\rangle \,|\, I) = \Pr(a_1, \ldots, a_{N_m}, s_1, \ldots, s_{N_m}).$$

The probability of symbol model m at translation t matching a region in I is then:

$$\begin{aligned} &\Pr(a_1, \ldots, a_{N_m}, s_1, \ldots, s_{N_m}) \\ &\quad = \Pr(a_1, \ldots, a_{N_m} | s_1, \ldots, s_{N_m}) \\ &\qquad \times \Pr(s_1, \ldots, s_{N_m}), \end{aligned} \tag{7}$$

by the probability chain rule. The match probability factors into the conditional probability of the matched primitives, $\Pr(a_1, \ldots, a_{N_m} | s_1, \ldots, s_{N_m})$, and the conditional probability of the indicator variables, $\Pr(s_1, \ldots, s_{N_m})$.

To simplify the model so that is could be computed, we assume that matched primitive probabilities are independent of one another and the indicators. So the dis-

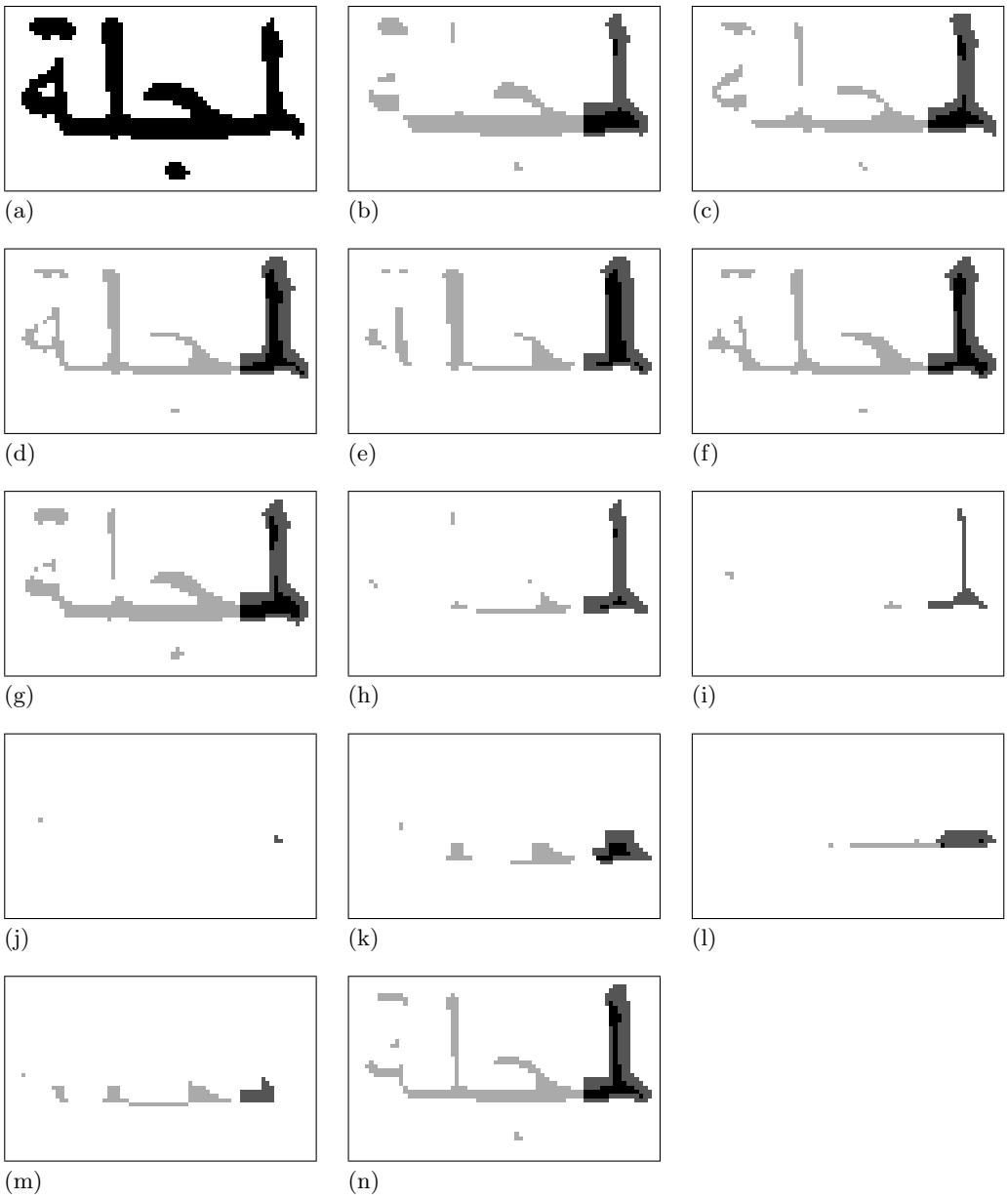


Fig. 12a–n. Matching a scanned word with symbol model . The top left image shows the scanned word. The other images in reading order correspond to the primitives of Fig. 6. Each image shows primitives detected on the word in light gray, the correspondence regions of the model in dark gray, and the areas in common in black

tribution reduces to:

$$\Pr(a_1,\ldots,a_{N_m},s_1,\ldots,s_{N_m}) = \prod_{i=1}^{N_m}\Pr(a_i) \times \Pr(s_1,\ldots,s_{N_m}), \quad (8)$$

the factors on the right-hand side being the distribution of each matched model primitive and the joint distribution of indicator variables, respectively. In the following sections, we will develop each of these distributions.

5.3.1 The joint distribution of indicator variables. Since indicator variables are binary, their joint probability distribution can be formulated in tabular form. The tables can be simply calculated from training instances of symbols. However, when using this simple scheme, the probability of an unseen configuration of indicators (in the training data) is zero. Since the number of variables (primitives) is large (on the order of 20-60), and the training sample size per symbol is on the order of 100, there are bound to be some possible configurations that do not show up in the training sample. To take care of this, we employ two modifications: (1) we reduce the joint probability model using the theory of probabilistic graphical models and (2) we use Bayesian estimation to estimate the table entries.

Simplifying an indicator model is illustrated by an example. Let a hypothetical vector of indicators have five indicators, $s_1,\ldots,s_5$. The complete table for the frequencies of the indicators is shown in Table 2. The

Table 2. The probability table for a hypothetical set of five indicators

			s4 = 0				s4 = 1			
			s2 = 0		s2 = 1		s2 = 0		s2 = 1	
			s1 = 0	s1 = 1	s1 = 0	s1 = 1	s1 = 0	s1 = 1	s1 = 0	s1 = 1
			0	0	0	0	0	0	0	0
			1	1	1	1	1	1	1	1
s5 = 0	s3 = 0	0	x1	x2	x3	x4	x5	x6	x7	x8
		1	x9	x10	x11	x12	x13	x14	x15	x16
s5 = 1	s3 = 0	0	x17	x18	x19	x20	x21	x22	x23	x24
		1	x25	x26	x27	x28	x29	x30	x31	x32

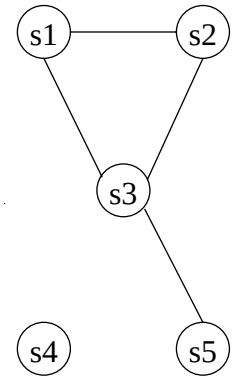


Fig. 13. The generated graphical model for the indicator example

maximum likelihood estimator of the probability of the configuration $s_1 = 1, s_2 = 0, s_3 = 1, s_4 = 0$, and $s_5 = 0$ is:

$$\Pr(s_1 = 1, s_2 = 0, s_3 = 1, s_4 = 0, s_5 = 0) = \frac{x_{10}}{t},$$

where $t = \sum_i x_i$.

By reducing the joint probability model using the theory of graphical models [46], assume that the simplified probability model is as shown in Fig. 13. The model has been simplified from one large clique of five variables into three cliques of three, two, and one variables. The probability model that corresponds to this graphical model is:

$$\Pr(s_1, \ldots, s_5) = \frac{\Pr(s_1, s_2, s_3) \cdot \Pr(s_3, s_5) \cdot \Pr(s_4)}{\Pr(s_3)}.$$

It can be seen here that the table of dimension $2^5 = 32$ (Table 2) has been reduced to other tables, the largest of which has dimension $2^3 = 8$. Now to compute the probability of the above configuration of the five variables, we need to compute $\Pr(s_1 = 1, s_2 = 0, s_3 = 1)$, $\Pr(s_3 = 1, s_5 = 0)$, $\Pr(s_4 = 0)$, and $\Pr(s_3 = 1)$.

After finding an appropriate model expressed as a set of cliques and separators, we estimate the distribution of each clique and separator. Since all the variables in a clique or separator are interdependent, the joint distribution of a set of binary variables of a clique or separator can be estimated by the maximum likelihood estimator: construct a table for the variables of the clique from the training sample; the probability of a certain configuration of variables is the count of the number of training cases with that configuration divided by the total number of training cases.

This simple method, however, is inadequate when the number of variables (the dimension) is high, as the number of possible configurations is exponential in the number of variables and the data will be sparse. Hence, we will assume that cells of a table are distributed as a multinomial distribution and use Bayesian estimation to estimate the parameters of the distribution given the data. The maximum likelihood estimator for $\Pr(s_3 = 1, s_5 = 0)$ is:

$$\frac{\sum_{i=9}^{16} x_i}{t},$$

while the Bayesian estimator for the same probability is:

$$\frac{1 + \sum_{i=9}^{16} x_i}{4 + t}.$$

5.3.2 The distribution of matched primitives. The remaining component of the match probability is the distribution of matched primitives. We assume that the distribution of a model primitive, (p, r, c, A, R), that has been matched to some detected primitives is a function of the measured area of match, a:

$$\Pr(a) = k(\alpha)\mathrm{e}^{-\alpha \frac{|a-A|}{A_R}}, \tag{9}$$

where $\alpha > 0$ and $k(\alpha) > 0$ are the parameters of the distribution, and A_R is the total area of the correspondence region. This form assumes that the probability of a measured area is a maximum when it equals the ideal area, and the probability is inversely proportional to the difference in area, normalized by the size of the correspondence region.

A more detailed description of the probability models and how their parameters are estimated can be found in [4].

6 Experiments

The system is currently implemented in C and runs on Unix workstations. It comprises almost 8000 line of code, excluding image reading and conversion and morphology code. The experiments were run on Sun Sparc 10 workstations.

The system operates in two major modes: a *training mode* and a *recognition mode.* During training, the system estimates the noise radius and generates the symbol models by computing the primitives of each model, including the measurements of the ideal primitives and their correspondence regions. The symbol models are the basis on which all parts of the systems depend.

The experiments performed on the system are of two types: recognizing individual symbols and recognizing whole words. The first type tests the matcher, which is

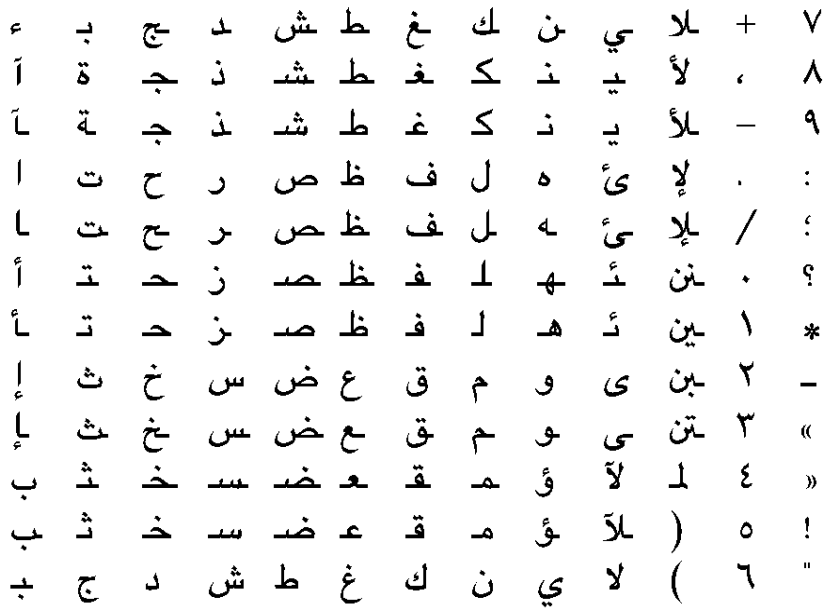


Fig. 14. The font sheet for the *Nadeem* font showing, the symbol shapes on which the system was trained

presented with a subimage containing exactly one symbol. The second type tests the performance of the whole system, which is presented with a subimage containing exactly one word. The experimental protocol followed here is explained in detail in [20].

6.1 Data sets

Experiments were conducted on three types of document images: ideal (noise-free), synthetically degraded, and scanned documents. Ideal documents come from synthetic images that are generated on an Apple Macintosh that has the Arabic language system installed. The ideal images are in Postscript and are converted to bitmap format using Ghostscript and the PBMPlus package.

The synthetically degraded images were generated from the ideal image using the noise model of Kanungo et al. [21]. By using this process, it is easy to generate many versions of the same passage of text that have different degradation parameters. The degradation model parameters were set as follows: $\alpha = 1.5$, $\alpha_0 = 1.0$, $\beta = 1.5$, $\beta_0 = 1.0$, $c_0 = 0.0$, and $e = 3$. For the same ideal image and the same degradation parameters, different degraded images are generated by using different seeds for the random-number generator. In Fig. 10: (a) shows a noiseless image of a text paragraph, and (b) shows the same paragraph after synthetically degrading it with the above degradation parameters.

The ideal images for the symbol experiments are the font contact sheets that are used to train the system. Figure 14 shows the font contact sheet for the *Nadeem* font. Each symbol (character shape) is surrounded by space and segmented automatically using the vertical and horizontal projection histograms. The symbol identities are stored as a list and correspond to symbol images, in a one-to-one manner. The degraded images that are used to test the system are different (i.e.,use different seeds) from the ones used to train the system.

The ideal images for the word recognition experiments are those generated by typesetting the text of seven pages (described below). After converting the pages into images, the words are isolated using the vertical and horizontal projections.

The scanned documents are a set of seven pages selected from a news magazine. By visual inspection, it was determined that these pages were most similar to the Macintosh *Nadeem* font type at a font size of 12 points. These pages were scanned at 300 dots per inch, in-house. They were then zoned, and their text was entered by two independent typists and verified. The details of the preparation process can be found in [10]. In Fig. 10, (c) shows a paragraph scanned from one of these documents.

When using scanned documents to train the system, words must be segmented into symbols. All the symbols on each of the seven pages were manually delineated by bounding boxes, and each box was labeled with the symbol identity. This allows training the matcher and also testing the recognition rate of the individual symbols.

The document images described above are packaged into several sets and then used to train and test the system. In devising the sets, care is taken to eliminate any overlap between the training and testing sets.

The training sets are symbol-based – i.e., in the form of symbols delineated by bounding boxes – because they are used for training the matcher. Two different *training sets* are alternatively used to train the system:

1. The degraded set consists of the symbols in 500 synthetically degraded instances of the font contact sheet (Fig. 14), each degraded with a different random seed. Since the font has 156 symbols, the total number of symbols is 78 000.

Table 3. Symbol recognition experiments and results

Test Set	Recognition Rate	Confidence Interval
Ideal	100%	
Degraded	99.69%	0.1%
Scanned	94.1%	0.5%

2. The mixed set consists of symbols in four of the seven scanned pages. These pages include 10 712 symbols. In addition, the set includes the symbols of 75 synthetically degraded images of the font contact sheet. This was necessary to increase the number of training symbols and to represent some of the symbols that were not on the documents. The total number of symbols in this set is 22 412.

The second training set is used to find the effect of training with scanned symbols on the recognition of scanned symbols and words.

The testing sets are of two types. Symbol testing sets are used to test the performance of the matcher; each symbol is delineated by a bounding box. The word testing sets are used to test the performance of the whole system; each word is delineated by a bounding box.

The two *symbol test sets* are as follows:

1. The synthetic set consists of the symbols in 100 synthetically degraded images of the font contact sheet. The random seeds used to generate these images differ from the seeds of the two training sets. This set contains a total of 15 600 symbols.
2. The scanned set consists of symbols in three of the seven scanned pages. These pages include 10 267 symbols.

The three *word test sets* are as follows:

1. The ideal set consists of the words of the seven ideal document pages described above. This set includes a total of 4317 words.
2. The degraded set consists of the words of a synthetically degraded version of the above set.
3. The scanned set consists of the words in the seven scanned pages.

The number of words tested from each set is listed below.

6.2 Symbol experiments

In the symbol experiments, many settings for the system parameters were tried. A full description of the parameters and the experiments can be found in [4]. Table 3 summarizes the best results attained for each test set. The table also shows a 95% confidence interval for each rate. The method for computing the confidence interval assumes that experiments are independent Bernoulli trials with identical success probability and then uses the normal approximation to the binomial distribution.

The experiments indicated that a larger set of shape primitives significantly improves performance. The results also indicate that when all the indicators of the indicator joint probability (last term of Equation 8) are assumed to be independent (no use of graphical models), the performance becomes significantly worse, and that when this term is removed altogether, is performance is insignificantly different.

Table 4. Confusion pairs for the scanned set. This table accounts for 70% of the errors

true identity	recognized identity	% of total
لا	لأ	7.3%
ج	ح	5.9%
لا	لأ	4.7%
ز	ر	3.8%
لا	لآ	3.7%
أ	ا	3.5%
خ	ح	2.6%
»	«	2.5%
«	»	2.5%
		2.5%
ا	!	2.2%
ا		2.2%
ز	ر	2.2%
		2.1%
ذ	د	2.0%
·		2.0%
ث	ت	1.8%
		1.8%
	-	1.8%
ز	·	1.7%
		1.4%
		1.4%
	·	1.4%
	ر	1.4%
ز	د	1.3%
		1.2%
،	.	1.0%
ا		0.9%
·		0.9%
-		0.9%

Table 4 lists the most frequent confusion pairs, which account for 70% of the errors for a test run on scanned symbols. Figure 15 shows the discrepancy in the match for the most frequent error of the table. The bottom row shows that the primitives that belong to the left stem of the symbol (white) do not fall in the correspondence regions (gray). This is due to the different angle of the stem in the scanned font compared to the training font.

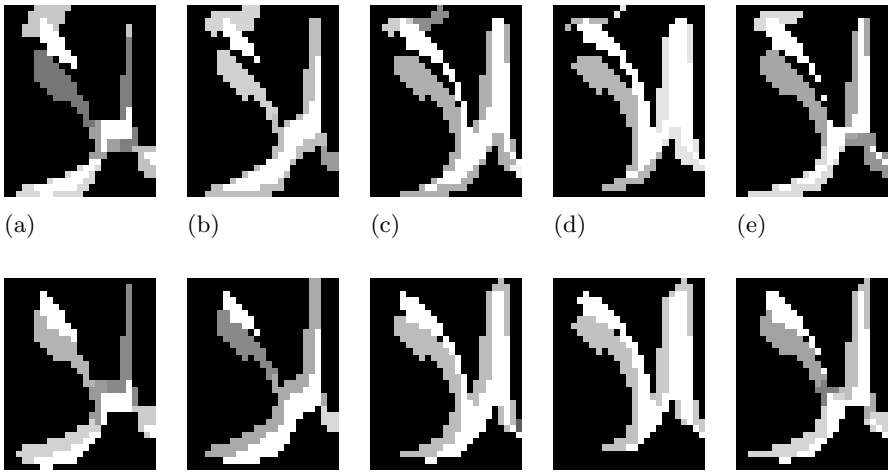


Fig. 15a–j. The matched primitives for the most frequent error case of Table 4. Each image shows the correspondence regions of the model (in shades of gray) and the primitives of the actual symbol (superimposed in white). The top row shows the model with the highest probability, while the bottom row shows the correct model. The images in columns 1 to 4 are for primitives $P0$–$P3$ of Fig. 6, respectively. The last column is for primitive $P12$

Table 5. Word recognition experiments and results. The average time is in milliseconds

Test Set	# Words	Recognition Rate	Time/Word (ms)	Confidence Interval
Ideal	3787	99.39%	16 718	0.3%
Degraded	3522	95.60%	59 760	0.7%
Scanned	830	73.13%	224 593	6%

The frequency of these errors can be reduced by making the correspondence regions more true to the font of the scanned symbols. For this, we try a new method of generating correspondence regions. Using this method, we extract all the shape primitives from all training instances. For every symbol model and for every shape primitive, we create an image that is the union of all training images for the symbol and primitive. This image then serves as the expanded image from which to construct the correspondence regions. Using this method, the recognition rate increased by 1.14%.

6.3 Word experiments

Table 5 summarizes the experiments that have been conducted on words. The time figures should be treated *very* cautiously, since they assume that the time to recognize a word is within the clock limit of 36 minutes. This means that time figures are lower bounds on the true time.

The most noteworthy observation from the word experiments is that the search process in word recognition takes an extremely long time. Table 5 shows that even for noise-free words it takes on average more than 16 seconds to recognize one word. Recognition time is almost four times larger for synthetically degraded words, and many more times larger for scanned words. These times are for un-optimized program code.

For the ideal and the degraded test sets, the words are of the same font and are formatted identically, which means that their search spaces are identical. The difference is in the order and number of states that are searched from that space. When the symbols are degraded, the dissimilarity between symbols, measured by the match probability, becomes less. This reduces the effectiveness of pruning and the heuristic function in trimming the states, both of which exploit variation in match probability.

The increase in search time is even more for scanned words. One reason is that the matcher is less effective in identifying scanned symbols because the scanned font is different from the font used to generate the synthetic images. Another equally important reason is the increased length of the scanned words. The scanned words come from magazine columns. Many of the words have been elongated using the *tatweel* symbol to align the margins of the columns (see Section 1.1). The presence of the elongation symbol adversely affects recognition time in two ways. First, it obviously increases the length of the word to be recognized. The amount of time needed to recognize a word is proportional to its length, as the depth of the search tree is determined by the length of the word. Second, it obscures the process of proposing translations for symbols.

The recognition-rate figures for word experiments show that the system is effective in recognizing noise-free words and synthetically degraded words. Using the method of [19], the recognition rate for noise-free words is determined to be above 99% with a certainty of 94%. The recognition rate is above 95% with even higher certainty.

7 Discussion and conclusions

The most distinguishing feature of this work is that it does not follow the classical recognition paradigm of a sequence of segmentation, feature extraction, and classification steps, which is sometimes hooked up in a feedback loop. The premise underlying this work is that accurate segmentation is difficult. For that the system does not commit itself to a segmentation of the word, rather it simulates trying different segmentation points and then chooses the best set of segmentation points that provides the best recognition. As we explained in Section 1.2 this system optimizes the segmentation with respect to the whole word. The price paid for the optimization is having to use a time-consuming search.

This system successfully uses primitives extracted by mathematical morphology operations to recognize words. The features we use are structural and statistical at the same time. They are structural in the sense that they indicate the presence of shape primitives in the input image. They are statistical in that they indicate the size of the matched primitives and are used to calculate match probabilities.

At the outset, the matching technique used here resembles template matching in that there is a reference template (symbol models here) that is matched with local regions in the word. The main difference, though, is that the matcher does not analyze the whole region covered by the symbol model; it analyzes only the region covered by the correspondence regions of the symbol model. Detected primitives that are covered by a symbol model but not within a correspondence region can be used again to match another symbol model or are considered extraneous primitives if left unused. Other differences are that the quality of match is computed according to an assumed probability model and that the match is flexible because the correspondence regions were constructed according to an assumed noise model.

When using probabilities for character recognition, a common practice is to assume the conditional independence of the components of a text segment. This simplifies the estimation procedure and calculating the probabilities. The other extreme is not to assume independence and take all variables that correspond to components to be dependent. This is very hard to estimate and calculate for more than a few variables, and is harder when variables can take more than two values. Even if it were computationally feasible to do so, it is not necessarily the best model to explain the data.

Here, we sought a middle ground. We break down the match probability into an area part and indicator part. Though we assume the areas to be conditionally independent given the indicators, we use the theory of graphical models to find a suitable model for the joint distribution of the indicators. The experiments did not show a significant difference in performance among the cases of using graphical models and not using the indicators at all. However, assuming that all indicators are independent had an adverse effect on performance.

During the process of generating ground truth for the seven scanned pages, we have experienced firsthand the amount of work required to prepare the pages. Not only the text had to be entered, but also words had to be manually segmented into symbols. We have used a graphical tool, built in-house, to delineate the symbols with bounding boxes. Though this tool simplified matters, the process was still very time consuming, taking a few hours per page just to segment the symbols. A similar experience is reported in [22] when generating a training set.

One of the design goals of this system was to reduce the effort needed in preparing the training set. For that we have relied on a noise model to degrade an ideal version of the symbols and produce many degraded instances of the symbols. Using the synthetically degraded symbols to train the system, the system achieved a matched symbol recognition rate of over 90%. The use of mixed symbols, however, boosted the rate by about 4%. This indicates that the noise model needs to be modified to better resemble actual symbols.

The main theoretical contribution of this work is in laying the foundation for a segmentation-free approach for Arabic word recognition. Recognition is based on maximizing the probability of the word given the detected primitives. In this work we have integrated the use of morphological primitives, with state-space search, a noise generation model, a noise effect model, graphical probabilistic models and Bayesian estimation.

The main practical contributions of this work are in implementing a system that satisfies its performance requirements for noise-free and synthetically degraded text. The system also requires minimal effort to train. Though the performance was not impressive for scanned text, several ways are suggested to improve its performance on such text. This work should be regarded as the beginning of a path that is to lead to an alternative and better recognition strategy.

The highest symbol recognition rates that were attained in the experiments were 99.69% for synthetic symbols and 94% for scanned symbols. The number of data points used to calculate those two rates are above 10 000 cases each.

When comparing the performance of this system to other systems one runs into difficulty. The test sets are different, the performance measures are not clearly defined (nor necessarily similar), and many of the results reported lack statistical analysis. Most systems report on experiments conducted on a few data samples, and a few tens at most. Further, it is not made clear whether the training and testing sets are mutually exclusive as they should be. For the reason of limited test data sets, and because the systems in the literature seldom report on the type, font, size, and source of their test sets, and also on the definition of recognition rate, one cannot directly compare the percentages reported for different systems. Many works in the literature report recognition rates of up to 99%. Some of those results are listed in the review papers of [2] and [3]. It is interesting to note that on the first data set on which the system was tested, the system had a 100% success rate. On further experimentation on a large sample the system was found to have a much lower success rate.

7.1 Future work

In this work we have assumed that the words were already segmented from document pages and that they were not rotated. A practical recognition system must be able to robustly extract the words and align skew. Because of time and resource issues, we did not deal with these aspects which are discussed in the research literature.

To improve the speed of search, one can search fewer states of the space, reduce the space of the search, or reduce the time necessary to generate and expand a state. To reduce the number of searched states, a better heuristic is needed. The heuristic must be more effective in pruning states that do not lead to the correct solution. The other two options require more radical changes to the design of the system.

One way to reduce the space of the search is to examine the word to be recognized and suggest a small set of segmentation points, instead of trying out many translations for symbols. The object of the search or optimization would then be to find the best subset of segmentation points. If the set of proposed segmentation points always includes the true segmentation points, then, in addition to reducing the space of the search, this approach can find the optimal solution. It remains to be investigated how this method actually performs. The use of segmentation points can also help in solving the problem of inaccurate translations described above, since it uses more information.

The key to optimizing word recognition is in evaluating the word probability and not just considering the goodness of match of individual symbols or local regions. By doing so, the word probability can take into account the probability of the segmentation points being the true ones.

At the time that this study was done, there was a lack of fully ground truthed[1] Arabic document image data bases. So we made do with synthetically generated and synthetically degraded Arabic document images whose ground truth would be automatically created at the time of image generation. However, the font used for this generation and the font that appeared in the real scanned images had differences. We believe that these differences are a major contributor to the lower recognition rate on the real scanned images when we used the real scanned images for the test data set. It would be appropriate to repeat this study using an extensive set of real scanned Arabic document images.

References

1. H.Y. Abdelazim, M.A. Hashish, Automatic reading of bilingual typewritten text. Proc. Comp. EURO '89 VLSI and Computer Peripherals, Hamburg, Germany, May 1989. pp. 2/140–144
2. P. Ahmed, M.A.A. Khan, Computer recognition of Arabic scripts based text – the state of the art. Proc. 4th International Conference and Exhibition on Multi-lingual Computing (Arabic and Roman Script), University of Cambridge, London, U.K., April 1994, pp. 2.2.1–2.2.15
3. B. Al-Badr, S. Mahmoud, Survey and bibliography of Arabic optical text recognition. Signal Processing 41(1): 49–77 (1995)
4. B. Al-Badr, A segmentation-free approach to text recognition with application to Arabic text. PhD Thesis, University of Washington, Seattle, WA, 1995
5. H. Almuallim, S. Yamagochi, A method of recognition of Arabic cursive handwriting. Pattern Recognition 9(5): 715–722 (1987)
6. A. Amin, State of the art on character recognition. In: Arabic Language Meeting, Paris, January 1985
7. A. Amin, S. Al-Fedaghi, Machine recognition of printed Arabic text utilizing natural language morphology. IEEE Trans. Systems, Man, and Cybernetics 35, 769–788 (1991)
8. A. Amin, G. Masini, Machine recognition of cursive Arabic words. SPIE Vol 359 Applications of Digital Image Processing IV, San Diego, CA, August 1982, pp. 286–292
9. C.H. Chen, J.L. Decurtins, Word recognition in a segmentation-free approach to OCR. Proc. 2nd International Conference on Document Analysis and Recognition, Japan, 1993, pp. 573–576
10. I.T. Phillips, S. Chen, R.M. Haralick, CD-ROM document database standard. Proc. International Conference on Document Analysis and Recognition, Tsukuba, Japan, 1993, pp. 478–482
11. J.H. Chiang, P.D. Gader, Hybrid fuzzy-neural systems in handwritten word recognition. IEEE Trans. Fuzzy Systems 5(4): 497–511 (1997)
12. P.D. Gader, M. Mohamed, J.H. Chiang, Handwritten word recognition with character and inter-character neural networks. IEEE Trans. Systems Man and Cybernetics 27(1): 158–165 (1996)
13. C.E. Dunn, P.S.P. Wang, Character segmentation techniques for handwritten text – a survey. Proc. 11th IAPR International Conference on Pattern Recognition, The Hague, Netherlands, August 1992, Vol. 2, pp. 577–580
14. M. Fakir, C. Sodeyama, Machine recognition of Arabic printed scripts by dynamic programming matching method. IEICE Trans. Information and Systems 76(2): 235–242 (1993)
15. C. Fang, J. Hull, A hypothesis testing approach to word recognition using an A* search algorithm. Proc. 3rd International Conference on Document Analysis and Recognition, Montreal, Canada, 1995, pp. 360–363
16. A.M. Gillies, D. Hepp, P.D. Gader, A system for recognizing handwritten words. In: ERIM Technical Report to U.S. Postal Service, Ann Arbor, MI, 1992
17. P.D. Gader, M. Khabou, Automatic feature generation for handwritten digit recognition. IEEE Trans. Pattern Analysis and Machine Intelligence 18(12) 1256–1262 (1996)
18. A.M. Gillies, Automatic generation of morphological template features. SPIE Conference on Image Algebra and Morphological Image Processing, San Diego, CA, 1990, pp. 252–261
19. R.M. Haralick, Performance assessment of near-perfect machines. Machine Vision and Applications 2(1): 1–16 (1989)

[1] As complete as the UW English Document Image Data Bases.

20. R.M. Haralick, Performance characterization in computer vision. Proc. Computer Analysis of Images and Patterns. 5th International Conference, CAIP '93 Proc., Budapest, Hungary, September 1993, pp. 1–9
21. T. Kanungo, R.M. Haralick, I.T. Phillips, Global and local document degradation models. Proc. International Conference on Document Analysis and Recognition, Tsukuba, Japan, 1993, pp. 730–734
22. K.M. Hassibi, Machine-printed Arabic OCR using neural networks. Proc. 4th International Conference and Exhibition on Multi-lingual Computing (Arabic and Roman Script), University of Cambridge, London, U.K., April 1994, pp. 2.3.1–2.3.12
23. M.J. Ganzberger, R.M. Rovner, A.M. Gillies, D.J. Hepp, P.D. Gader, Matching database records to handwritten text. In: SPIE Conference on Document Recognition, San Diego, CA, 1994
24. M. Shridhar, G. Houles, F. Kimura, Handwritten word recognition using lexicon free and lexicon directed word recognition algorithms. In: 4th International Conference on Document Analysis and Recognition, Ulm, Germany, 1997
25. K.M. Jambi, Arabic character recognition: many approaches and one decade. Arabian J. Engineering and Sciences 16(4): 499 (1991)
26. K.M. Jambi, A system for recognizing handwritten Arabic words. Proc. 13th National Computer Conference, November 1992, pp. 472–482
27. G. Kim, V. Govindaraju, Handwritten word recognition for real-time applications. Proc. 3rd International Conference on Document Analysis and Recognition, Montreal, Canada, 1995, pp. 24–28
28. B.M. Kurdy, A. Joukhadar, Multifont recognition system for Arabic characters. Proc. 3rd International Conference and Exhibition on Multi-lingual Computing (Arabic and Roman Script), University of Durham, U.K., December 1992, pp. 7.3.1–7.3.9
29. J. Makhoul, R. Schwartz, C. Lapre, I. Bazzi, Z. Lu, P. Natarajan, A language-independent methodology for OCR. Symp. Document Image Understanding Technology, Annapolis, MD, 1997, pp. 99–108
30. V. Margner, Sarat – a system for the recognition of Arabic printed text. Proc. 11th IAPR International Conference on Pattern Recognition, The Hague, Netherlands, September 1992, pp. 561–564
31. P.D. Gader, J.M. Keller, R. Krishnapuram, J.H. Chiang, M.A. Mohamed, Neural and fuzzy methods in handwriting recognition. Computer 30(2): 79–86 (1996)
32. M. Parizeau, R. Plamondon, A fuzzy-syntactic approach to allograph modeling for cursive script recognition. IEEE Trans. Pattern Analysis and Machine Intelligence 17(7): 702–712 (1995)
33. S.S. El-Dabi, R. Ramsis, A. Kamel, Arabic character recognition system: a statistical approach for recognizing cursive typewritten text. Pattern Recognition 23(5): 485–495 (1990)
34. E. Rich, Artificial Intelligence. New York: McGraw-Hill 1983
35. H.Y. Abdelazim, A.M. Mousa, Y.R. Saleh, M.A. Hashish, Arabic text recognition using a partial observation approach. Proc. 12th National Computer Conference, Riyadh, Saudi Arabia, October 21–24, 1990, pp. 427–437
36. J. Trenkle, A. Gillies, S. Schlosser, An off-line Arabic recognition system for machine-printed documents. Symp. Document Image Understanding Technology, Annapolis MD, 1997, pp. 155–162
37. M.F. Tolbaand, E. Shaddad, On the automatic reading of printed Arabic characters. Proc. IEEE International Conference on Systems, Man and Cybernetics 1990, pp. 496–498
38. A. Shoukry, Arabic character recognition state of the art. Proc. 11th National Computer Conference, Dhahran, Saudi Arabia, March 1989, pp. 382–390
39. F. Kimura, M. Shridhar, N. Narasimhamurthi, Lexicon directed segmentation – recognition procedure for unconstrained handwritten words. Proc. Third International Workshop on Frontiers in Handwriting Recognition, Buffalo, NY, 1993, pp. 122–132
40. T.A. Standish, Data Structure Techniques. Reading, MA: Addison-Wesley 1980
41. F. Stentiford, Automatic feature design for optical character recognition using an evolutionary search procedure. IEEE Trans. Pattern Analysis and Machine Intelligence 7, 349–355 (1985)
42. P.D. Gader, B.D. Forester, A.M. Gillies, M.J. Ganzerger, R.C. Vogt, J.M. Trenkle, A segmentation-free neural network classifier for machine-printed numeric fields. United States Postal Service Advanced Technology Conference, Washington D.C., 1992, pp. A137–A151
43. F. Kimura, M. Shridhar, S. Tsuruoka, Z. Chen, Context directed handwritten word recognition for postal service applications. United States Postal Service Advanced Technology Conference, Washington D.C., 1992, pp. 199–214
44. A.M. Gillies, P.D. Gader, M.P. Whalen, B.T. Mitchell, Application of mathematical morphology to handwritten zip code recognition. SPIE Conference on Visual Communications and Image Processing IV, Philadelphia, PA, 1989
45. B.T. Mitchell, A.M. Gillies, M.P. Whalen, P.D. Gader, A model-based computer vision system for the recognition of handwritten address zip codes. In: ERIM Technical Report to U.S. Postal Service, Ann Arbor, MI, 1989
46. J. Whittaker, Graphical Models in Applied Multivariate Statistics. New York: Wiley 1990

Dr. Haralick occupies the Boeing Clairmont Egtvedt Professorship in the Department of Electrical Engineering at the University of Washington. Professor Haralick is responsible for developing the gray scale co-occurrence texture analysis technique and the facet model technique for image processing. He has developed shape analysis and extraction techniques using mathematical morphology, the morphological sampling theorem, and fast recursive morphology algorithms. In the area of document image understanding, Professor Haralick, along with Professor Ihsin Phillips, developed a comprehensive ground-truthed set of over 1500 document images most in English and some in Japanese. He has also developed algorithms for document image skew angle estimation, zone delineation, word and text line bounding box delineation. His most recent research is in the area of computer vision performance characterization.

Professor Haralick is a Fellow of IEEE for his contributions in computer vision and image processing and a Fellow of IAPR for his contributions in pattern recognition, image processing, and for service to IAPR. He has published over 470 papers and has just completed his term as the president of the International Association for Pattern Recogntion.

Al-Badr. 1995 University of Washington Seattle, WA, USA Ph.D. Computer Science and Engineering. 1991 University of Washington Seattle, WA, USA M. Sc. Computer Science and Engineering. 1987 University of Petroleum & Minerals (UPM) B.Sc. Computer Science. Experience: Currently an Assistant Professor at Computers and Electronics Research Institute, King Abdulaziz City for Science and Technology (KACST), Riyadh, Saudi Arabia. Conducts research in Arabic optical charcter recognition and Arabic Internet technologies.