Segmentation-Based And Segmentation-Free Methods for Spotting Handwritten Arabic Words

Gregory R. Ball, Sargur N. Srihari, Harish Srinivasan

Segmentation-Based And Segmentation-Free Methods for Spotting Handwritten Arabic Words

Gregory R. Ball, Sargur N. Srihari, Harish Srinivasan
Center of Excellence for Document Analysis and Recognition (CEDAR)
University at Buffalo, State University of New York
Amherst, New York 14228
{grball, srihari, hs32}@cedar.buffalo.edu

Abstract

Given a set of handwritten documents, a common goal is to search for a relevant subset. Attempting to find a query word or image in such a set of documents is called word spotting. Spotting handwritten words in documents written in the Latin alphabet, and more recently in Arabic, has received considerable attention. One issue is generating candidate word regions on a page. Attempting to definitely segment the document into such regions (automatic segmentation) can meet with some success, but the performance of such an algorithm is often a limiting factor in spotting performance. Another approach is to directly scan the image on the page without attempting to generate such a definite segmentation.

A new algorithm for word spotting and a comparison of recent algorithms which act on previously unsegmented Arabic handwritten text is presented. The algorithms considered are an automated word segmentation method presented previously and a "segmentation free" algorithm which performs spotting directly on lines of unsegmented text. The segmentation free approach performs spotting and segmentation concurrently using a sliding window. The spotting method used to judge the performance of the algorithms is a character based method, but the results are independent of the actual spotting method used.

The segmentation-free method performs an average of 5-10% better than the automated segmentation method, and manages to have a lower per query cost on unprocessed images. However, it has a larger per query cost on preprocessed documents.

Keywords: segmentation, word spotting, Arabic, scanned document search, word recognition

1. Introduction

Given a set of handwritten words, a common query for a user to ask is whether or not a specific word is among that set. More generally, given a page of handwritten text, a common query is whether or not a word or words of interest appear on the page. Such a query allows a user to sift through a set of documents for a subset that is of most interest to the user. This is the motivation behind the *word spotting problem*: given a scanned image of a handwritten document and a UNICODE sequence of characters (the query word), the *word spotting problem* asks if the image contains a handwritten image of the query word and if so, at what coordinates in the image the word exists.

Spotting handwritten words in documents written in the Latin alphabet has received considerable attention in literature ([4], [5], [6], [7], [8]). More recently, exploration of word spotting in documents written in Arabic has begun ([1], [2]). While word recognition methods have been pursued for some time, the methods described here and in [1], [2] represent the first attempts known to the authors of word spotting based image retrieval for Arabic documents. Specific challenges to Arabic word spotting include a much larger potential character set (alphabet) due in part to characters written in a variety of forms, the presence of characters distinguished from one another by the presence of dots, either above or below the main body of the character, and characters which cannot easily be segmented horizontally. While the Arabic alphabet contains only 28 basic letters, they are often written in a variety of forms depending upon where in the word they appear, and depending upon which letter precedes them. This expands the number of character classes significantly–to about 100.

For a word spotting application to have practical utility, it is essential to operate on unconstrained pages of text. While line segmentation generally poses relatively little difficulty, many unresolved issues remain as to segmenting individual component words in a line. In word spotting, algorithms often incorporate a method to segment lines into words, such as in [1] and [2]. They can then generate a ranking of scores for the candidate words. Such segmentation requirements can present a severe bottleneck in terms of word spotting performance. Generally, a segmentation method which correctly generates only a percentage of the words in a line as candidates immediately results in a corresponding reduction in the maximum possible spotting rate of the algorithm. Regardless of the spotting method, if the desired word is not among the candidate images (e.g., it was split into multiple pieces) it cannot individually receive a score. Experimentally, this fact was confirmed in [2] and others where increases in the effectiveness of the segmentation algorithm proportionally increased the results of the spotting methods.

1.1. Segmentation Algorithms

Automatic word segmentation, as presented in [2], is based on taking several features on either side of a potential segmentation point and using a neural network for deciding whether or not the segmentation is between two distinct words. Some of the differences between the tasks of segmenting Arabic script and segmenting Latin script are the presence of multiple dots above and below the main body in Arabic and the absence of upper case letters at the beginning of sentences in Arabic. The method presented was found to have an overall correctness of about 60%.

The segmentation free method attempts to perform spotting and segmentation concurrently. Rather than a candidate word image, an entire line image acts as input. The line is split into segments based on an algorithm similar to the ligature-based segmentation algorithm used in [3]. All realistic combinations of adjacent connected components are considered as potential areas where the desired word may appear. This approach more exhaustively searches a line, looking for a given word image, while at the same time keeping the number of evaluations manageable by considering only a small subset of potential regions in the image.

1.2. Word Spotting Algorithm

A recently developed lexicon-driven word spotting approach is used to judge the performance of the approaches for unsegmented data. A measure of the effectiveness of the segmentation approaches is made by comparing the results of the word spotting methods on manually segmented images to the performance on unsegmented data. If the word spotting methods can perform equally well on segmented as well as unsegmented images, the lack of *a priori* manual segmentation is insignificant.

The word spotting algorithm is based on segmenting Arabic words into candidate characters, given an Arabic search word lexicon in UNICODE representation. It makes use of the sequence of characters for a word in the lexicon in order to select sets of prototype images representing the characters forming them. This candidate word image is first split into a sequence of segments (as in Figure 1), with the ideal result being individual characters in the candidate word being separated. In word spotting, the lexicon size is one.

The segmentation algorithm used "oversegments" words in the hopes of avoiding incorrectly putting more than a single character into a segment. For dealing with this oversegmentation, the segments need to be rejoined into candidate characters. To do this, all reasonable combinations of candidate characters are considered with dynamic programming, using the sequence of characters for deciding upon a reasonable range for the number segmentation points in a candidate word. Each subcharacter is assigned to a character appearing in the lexicon word. The subcharacters are placed into consecutive groups corresponding to the letter known to be in the same relative location in the word (i.e. after the group of subcharacters assigned to the first character, the group of subcharacters for the second character begin). The features of the composed characters are compared to features of prototype images of the characters. The groupings are maximized such that the overall score for the characters (groups of subcharacters) is best (the composed characters are a closest match to some of the prototypes). A score that represents the match between the lexicon and the candidate word image is then computed. The score relates to the individual character recognition scores for each of the combined segments of the word image. The prototype characters are from a library of many images of each possible character written by different writers, manually cut from documents where they appeared in words. While the images appearing in the figures are some of the more legible, many images of poorly written characters are included as well. When creating the sample set, the writers were told to make no effort to write neatly. These documents are the same as those described [2] where ten writers wrote 10 documents each.

The segmentation algorithm is essentially the same as the segmentation algorithm used in the "segmentation free" method; it is discussed more thoroughly in that section. The number of segmentation points is kept to a minimum. In [3] they were able to use a maximum of four segmentation points per character because the maximum number of segments a prototype character was segmented into was four. When the Arabic prototype characters were segmented, the maximum size was five due to the nature of the different style of character writing. Further issues of undersegmenting unique to Arabic are dealt with using compound character classes.

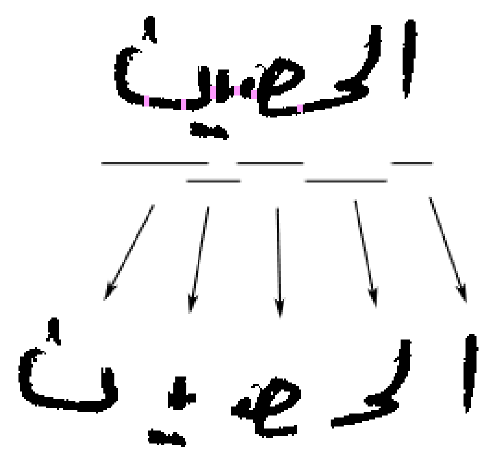


Figure 1. Arabic word "alhusian" (a family name), with segmentation points and corresponding best segments shown below

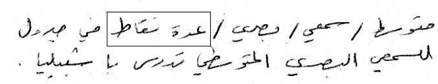
The features used for this algorithm are the WMR (word model recognition) feature set described in [10], which consists of 74 features. Two are global features–aspect and stroke ratio of the entire character. The remaining 72 are local features. Each character image is divided into 9 subimages. The distribution of the 8 directional slopes for each subimage form this set (8 directional slopes $\times$ 9 subimages $=$ 72 features). $F_{l_{i,j}} = s_{i,j}/N_i S_j$, $i = 1, 2, ..., 9$, $j = 0, 1, ..., 7$, where $s_{i,j} =$ number of components with slope j from subimage i, where $N_i =$ number of components from subimage i, and $S_j = max(s_{i,j}/N_i)$. These features are the basis of comparison for the character images derived from the segmentation of words to be recognized. To date, this appears to be the first application of WMR features to Arabic recognition.

As in [3], the objective is to find the best match between the lexicon and the image. In contrast to [3], up to five adjacent segments are compared to the character classes dictated as possibilities by a given lexicon entry for the reasons described earlier. In the first phase of the match, the minimum Euclidean distance between the WMR features of candidate supersegments and the prototype character images is computed. In the second phase, a global optimum path is obtained using dynamic programming based on the saved minimum distances obtained in the first matching phase. The lexicon is ranked, the entries with the lowest total scores being the closest matches. Figure 1 shows the the Arabic word “alhusian”, a family name, split into its best possible segments.

2. Image Processing

Image processing is accomplished via a method similar in part to that described in [3]. First, a chain code [12], [13], [14], [3] representation of the binary image’s contours is generated. Binary images are converted into a chain code representation by coding the boundary contours of image components while preserving the positional and directional information of adjacent pixels. Noise is removed, and based on analysis, slant, skew, stroke width, and size are corrected, and the image is smoothed. The algorithms are designed to work on documents which have already undergone preprocessing, so no effort was made to include “poor quality” images/images containing excessive artifacts.

One interesting issue raised by this image processing procedure is the removal of dots. Some dots critical to recognizing an individual character are removed from the Arabic image. While this may not pose a large issue to languages such as English where the number of characters with dots is small, it removes the ability to distinguish certain Arabic letters from one another. However, this rarely adds ambiguity to an entire Arabic word; that is, the number of words in practice that are distinguished from each other only by the presence of characters with the same similar base shape, but having or not having associated dots is not large. Further experiments are currently ongoing to determine if attempting to keep some of the dots might help boost performance in word spotting (or recognition).



(a) Region of a document image

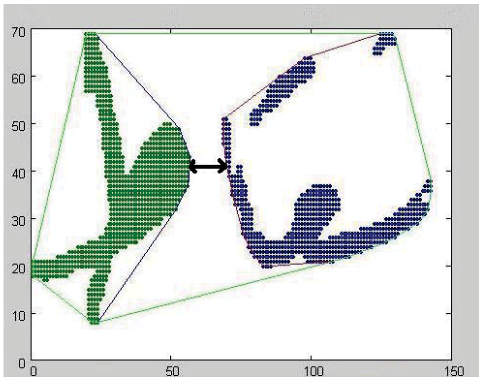
(b) Connected component exterior and interior contours

Figure 2. Connected components of a small section of the image.

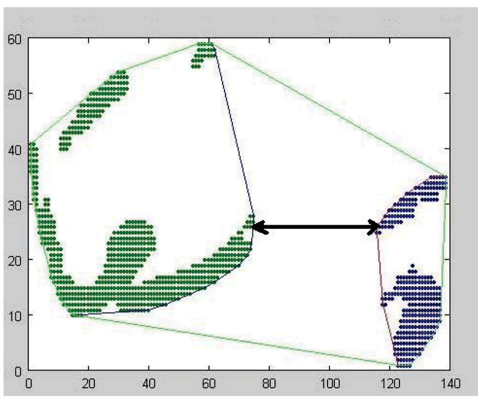
3. Automatic Word Segmentation

The process of word segmentation begins with obtaining the set of connected components for each line in the document image. Figure 2 shows the connected components (exterior and interior contours) of a small region of a document image. The interior contours or loops in a component are ignored for the purpose of word segmentation as they provide no information for this purpose. The connected components are grouped into clusters, by merging minor components such as dots above and below a major component. Also particular to Arabic, many words start with the Arabic character “Alef”. The presence of an “Alef” is a strong indicator that there may be a word gap between the pair of clusters. The height and width of the component are two parameters used to check if the component is the character “Alef”. Figure 4 shows samples of the Arabic character “Alef”. Every pair of adjacent clusters are candidates for word gaps. 9 features are extracted for these pairs of clusters and a neural network is used to determine if the gap between the pair is a word gap. The 9 features are: width of the first cluster, width of second cluster, difference between the bounding box of the two clusters, flag set to 1 or 0 depending on the presence or absence of the Arabic character “Alef” in the first cluster, the same flag for the second cluster, number of components in the first cluster, number of components in the second cluster, minimum distance between the convex hulls enclosing the two clusters and the ratio between, the sum of the areas enclosed by the convex hulls of the individual clusters, to the total area inside the convex hull enclosing the clusters together. The minimum distance between convex hulls is calculated by sampling points on the convex hull for each connected component and calculating the minimum distance of all pairs of such points. Figure 3 shows two pairs of adjacent clusters tested for word gaps.

A neural network was trained using these 9 features with feature vector labeled as to whether it is a word gap or not. This is similar to the neural network approach used for English postal addresses [9], but with different features.



(a) Not word gap



(b) Word gap

Figure 3. Two pairs of adjacent clusters. Dotted lines indicate convex hulls around the individual clusters. Solid lines indicate convex hull around the two clusters taken together. (a) The gap between the convex hulls is truly not a word gap, and (b) The gap between the convex hulls is truly a word gap.

Figure 4. Samples of Arabic character “Alef”. The height and width are two parameters that are used to detect the presence of “Alef” in the clusters.

3.1 Automated Word Segmentation Performance

When applied to the document set described earlier with correctly segmented lines, the overall performance is about 60% using a set of seven segmentation features. In [2], the authors noted that a more complex set of features is expected to yield a higher level of performance.

4. Segmentation Free Line Processing

The segmentation free algorithm processes the words on a per line basis rather than relying on pre-segmented words. The algorithm can be viewed as a sequence of steps. First, the image is processed into component lines. Candidate segmentation points are generated for a given line. The line is scanned with a sliding window, generating candidate words and scoring them, as well as filtering out nearly equivalent candidates.

4.1 Candidate Segmentation Algorithm

The segmentation algorithm used on the line is essentially the same as the one used to generate candidate character segmentation points in candidate words in the actual spotting step. It is performed via a combination of ligatures and concavity features on an encoded contour of the components of the image. Average stroke width is estimated and used to determine the features.

Ligatures, as noted in [3] are strong candidates for segmentation points in cursive scripts. Ligatures are extracted in a similar way as in [3]–if the distance between y-coordinates of the upper half and lower half of the outer contour for a x-coordinate is less than or equal to the average stroke width, then the x-coordinate is marked as an element of a ligature. Concavity features in upper contour and convexities in the lower contour are also used to generate candidate segmentation points, which are especially useful for distinct characters which are touching, as opposed to being connected. A ligature will cause any overlapped concavity features to be ignored. For a given x-coordinate, if a concavity and convexity overlap, a segmentation point is added for that x-coordinate.

While the character based method described in [3] uses this segmentation method to split a word into candidate characters, the segmentation free line processing method uses it to split the line, the motivation being to generate candidate word regions on the line. Arabic has predicable breaks in a word based on non-connective characters. Therefore, the number of connected components in a word is predictable as well.

4.2 Line Scanning

The method utilizes a sliding window, starting from the left of the line and proceeding to the right (in the opposite direction to the way Arabic is written), although the direction of the scan is unimportant because all realistic combinations of connected components will be considered.

Each character class c in Arabic is associated with a minimum and a maximum durational length ($minlen(c)$ and $maxlen(c)$ respectively). These lengths are generated by segmenting a representative dataset of characters with the same segmentation algorithm, and taking the min and max for each character. Due to the nature of the Arabic character set, the upper bound for all characters is 5, not 4 as in [3].

The scanning algorithm will scan for candidate words consisting of a range of segments. For a given search word W of length n, for each character $c_i \in W$, the minimum length $minlen(W)$ considered is $\sum_{i=0}^{n-1} minlen(c_i)$ and the maximum considered length $maxlen(W)$ is $\sum_{i=0}^{n-1} maxlen(c_i)$.

The scanning algorithm starts at each segmentation point p on a line. For a given point p_i, if $i = 0$ or if $p_i.left > p_{i-1}.right$ (i.e., there is horizontal space to the left of the segmentation point) it is considered a valid start point. Similarly, for a given point p_i, if $i = max(p)$ or if $p_i.right < p_{i+1}.left$ (i.e., there is horizontal space to the right of the segmentation point) it is considered a valid endpoint. The algorithm considers candidate words to be ranges of segments between two segmentation points p_i and p_j where p_i is a valid start point, $minlen(W) \leq j - i + 1 \leq maxlen(W)$, and p_j is a valid endpoint.

While this generally results in more candidate words than the other segmentation method, since each Arabic word is only broken into a few pieces separated by whitespace it does not result in a dramatic decrease in performance.

4.3 Filtering

Often, a candidate word influences neighboring candidate words' scores. Neighboring candidate words are those words with overlapping segments. Often, a high scoring word will result in high scores for neighbors. The largest issue rises when the high scoring word is, in fact, an incorrect match. In this case, the incorrect choice and several of its neighboring words receive similarly good scores, pushing the rank of the actual word lower in the list. Another issue is if the word being searched for appears multiple times in a document. The best matching words' neighboring candidates depresses the second occurrence's rank. Various ways of dealing with the overlap meet with different degrees of success.

The approach taken in the current incarnation of the algorithm is to keep the candidate word that has the highest score out of the overlapping words. Unfortunately, this occasionally removes the correct word completely from the list. Alternate methods of filtering are being explored.

5 Experimental Results

All methods were tested on a subset of the documents mentioned earlier; ten writers writing 10 documents each, making no particular effort to produce neatly written samples. For both methods, the documents were initially segmented into lines using a simple clustering algorithm.

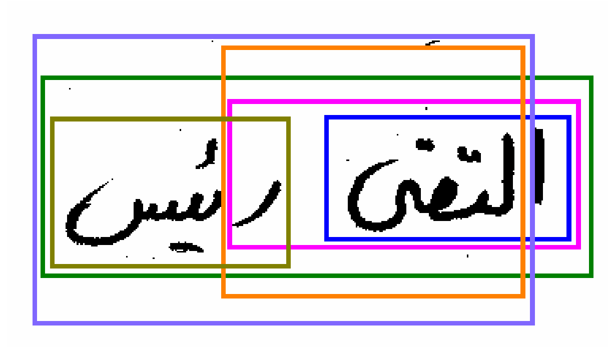


Figure 5. Candidate word regions

Figure 6 shows a precision-recall curve comparing the automated segmentation method and the segmentation-free method, when applied to the character based spotting method. Figure 7 shows a recall comparison of the segmentation methods.

5.1 Automatic Segmentation

It took about 2.0s to process an image into candidate words, and 491ms to segment the candidate words. After this document processing time of approximately 2.5s, each query ran in about 812ms.

5.2 Segmentation Free

It took an average of 38.5ms to process and segment a line, with an average of 18 lines per document tested. After segmentation was completed, it took an average of 120ms to score words in a line and filter the results. This gives an average time of 693ms to process a document, and 2160ms to spot a word in a processed document.

6 Conclusions and Future Directions

The segmentation free method shows a significant boost over the automated segmentation method for spotting Arabic words. While the algorithm is better both in performance and versatility than the automated segmentation method, there is significant room for improvement, as evidenced by the gap in performance between the algorithm acting on the segmentation free method and the manually segmented data. Preliminary experiments using additional heuristics have had some success in closing this gap. One such idea for a heuristic is limiting (or penalizing) a disproportionate amount of whitespace in the interior of a word. Another idea is based on the fact that Arabic words are naturally and predictably segmented into one or more pieces by whitespace. Incorporating scores matching the obvious segments in the word to the predicted separated subwords may be able to drive the segmentation free performance higher.

Running word spotting using the segmentation free method is slower than when running the same spotting method on a document segmented by the automatic segmentation method described in [2]. This reflects the fact that, on average, Arabic words consist of 2-4 sub-

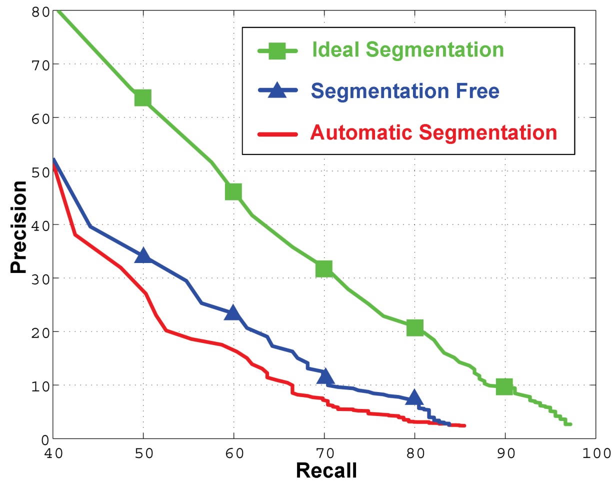


Figure 6. Precision-Recall comparison of Segmentation Methods

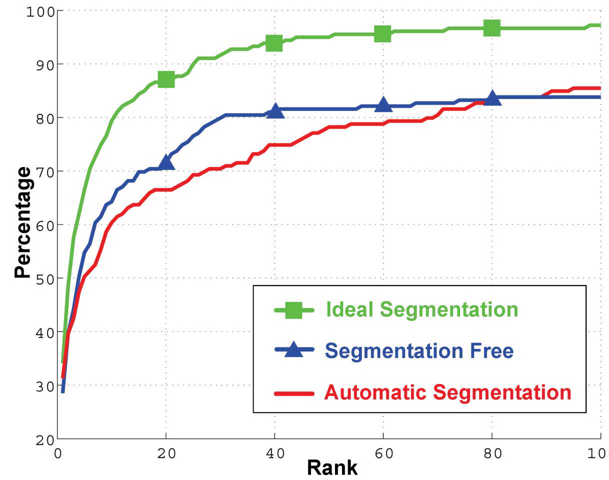


Figure 7. Recall comparison of Segmentation Methods

words which result in possibilities for word segmentation points. However, the segmentation free algorithm is faster on average when spotting a single word in an unprocessed document (2853ms for segmentation free vs. 3303ms for automatic segmentation). Interesting future directions include improved heuristics for the segmentation free method or combining the two methods into a single ranking.

References

[1] S. N. Srihari and H. Srinivasan and P. Babu and C. Bhole, "Handwritten Arabic Word Spotting using the CEDARABIC Document Analysis System", *Proc. Symposium on Document Image Understanding Technology (SDIUT-05)*, College Park, MD, November, 2005, pp 123–132.

[2] S. N. Srihari and H. Srinivasan and P. Babu and C. Bhole, "Spotting Words in Handwritten Arabic Documents", *Proceedings SPIE*, San Jose, CA, 2006, pp 606702-1–606702-12.

[3] G. Kim and V. Govindaraju, "A Lexicon Driven Approach to Handwritten Word Recognition for Real-Time Applications", *IEEE Transactions on Pattern Analysis and Machine Intelligence 19(4)*, April, 1997, pp. 366–379.

[4] A. Kolcz and J. Alspector and M. Augusteijn and R. Carlson and G. Viorel Popescu, "A Line-Oriented Approach to Word Spotting in Handwritten Documents", *Pattern Analysis & Applications*, Volume 3, Issue 2, Jun 2000, pp 153–168.

[5] M. Burl and P. Perona, "Using hierarchical shape models to spot keywords in cursive handwriting data", *IEEE-CS Conference on Computer Vision and Pattern Recognition*, June 23-28, 1998, pp 535–540.

[6] S. Kuo and O.E. Agazzi, "Keyword spotting in poorly printed documents using 2-d hidden Markov models", *IEEE Trans. Pattern Analysis and Machine Intelligence*, Volume 16, 1994, pp 842–848.

[7] R. Manmatha and T. M. Rath, "Indexing of handwritten historical documents-recent progress", *Symposium on Document Image Understanding Technology (SDIUT)*, 2003, pp 77–85.

[8] B. Zhang and S. N. Srihari and C. Huang, "Word Image Retrieval Using Binary Features", *Document Recognition and Retrieval XI*, SPIE Vol. 5296, 2004, pp 45–53.

[9] G. Kim and V. Govindaraju and S. N. Srihari, "A Segmentation and Recognition Strategy for Handwritten Phrases", *International Conference on Pattern Recognition*, ICPR-13, 1996, pp 510–514.

[10] Sargur N. Srihari and Catalin I. Tomai and Bin Zhang and Sangjik Lee, "Individuality of Numerals", *ICDAR '03: Proceedings of the Seventh International Conference on Document Analysis and Recognition*, 2003, Washington, DC.

[11] B. Zhang and S. N. Srihari, "Binary vector dissimularity measures for handwriting identification", *Proceedings of the SPIE, Document Recognition and Retrieval*, 2003, pp 155–166.

[12] H. Freeman, "Computer Processing of Line-Drawing Images", *Computing Surveys*, vol. 6, no. 1, 1974, pp 57-97.

[13] C. C. Lu and J.G. Dunham, "Highly Efficient Coding Schemes for Contour Lines Based on Chain Code Representations," *IEEE Trans. Communications*, vol. 39, no. 10, 1991, pp. 1511-1514.

[14] G. Kim and V. Govindaraju, "Efficient Chain Code Based Image Manipulation for Handwritten Word Recognition," *Proc. SPIE Symp. Electronic Imaging Science and Technology (Document Recognition III)*, San Jose, Calif., vol. 2, 660, Feb. 1996, pp. 262-272.