Word Spotting: Indexing Handwritten Archives.

R. Manmatha and W.B. Croft
Center for Intelligent Information Retrieval,
Computer Science Department
University of Massachusetts,
Amherst, MA-01003
U.S.A.
(413)-545-3623.
{manmatha,croft}@cs.umass.edu

Abstract

There are many historical manuscripts written in a single hand which it would be useful to index. Examples include the early Presidential papers at the Library of Congress and the collected works of W. B. DuBois at the library of the University of Massachusetts. The standard technique for indexing documents is to scan them in, convert them to machine readable form (ASCII) using Optical Character Recognition (OCR) and then index them using a text retrieval engine. However, OCR does not work well on handwriting. Here, an alternative scheme is proposed for indexing such texts. Each page of the document is segmented into words. The images of the words are then matched against each other to create equivalence classes (each equivalence classes contains multiple instances of the same word). The user then provides ASCII equivalents for say the top 2000 equivalence classes.

The current paper deals with the matching aspects of this process. Due to variations in even a single person's handwriting, it is expected that the matching will be the most difficult step in the whole process. Two different techniques for matching words are discussed. The first method, based on Euclidean distance mapping, matches words assuming that the transformation between the words may be modelled by a translation (shift). The second method, based on an algorithm developed by Scott and Longuet-Higgins, matches words assuming that the transformation between the words may be modelled by an affine transform.

Experiments are shown demonstrating the feasibility of the approach for indexing handwriting.

1. Introduction

The explosion of information in today's society has led to a need for indexing the information. If the information is in machine readable form (ASCII), it can be indexed using text retrieval engines. However, much of today's information is multi-media in nature. It is available on paper or on videos and not in machine readable format. A number of chapters in this book discuss the problem of retrieving and indexing multi-media information. For example, in chapter 2, Petkovic et al discuss the QBIC system to query images based on attributes like color, texture and shape, while in chapter 5, Jones and his collaborators discuss the problem of retrieving video mail by indexing on speech.

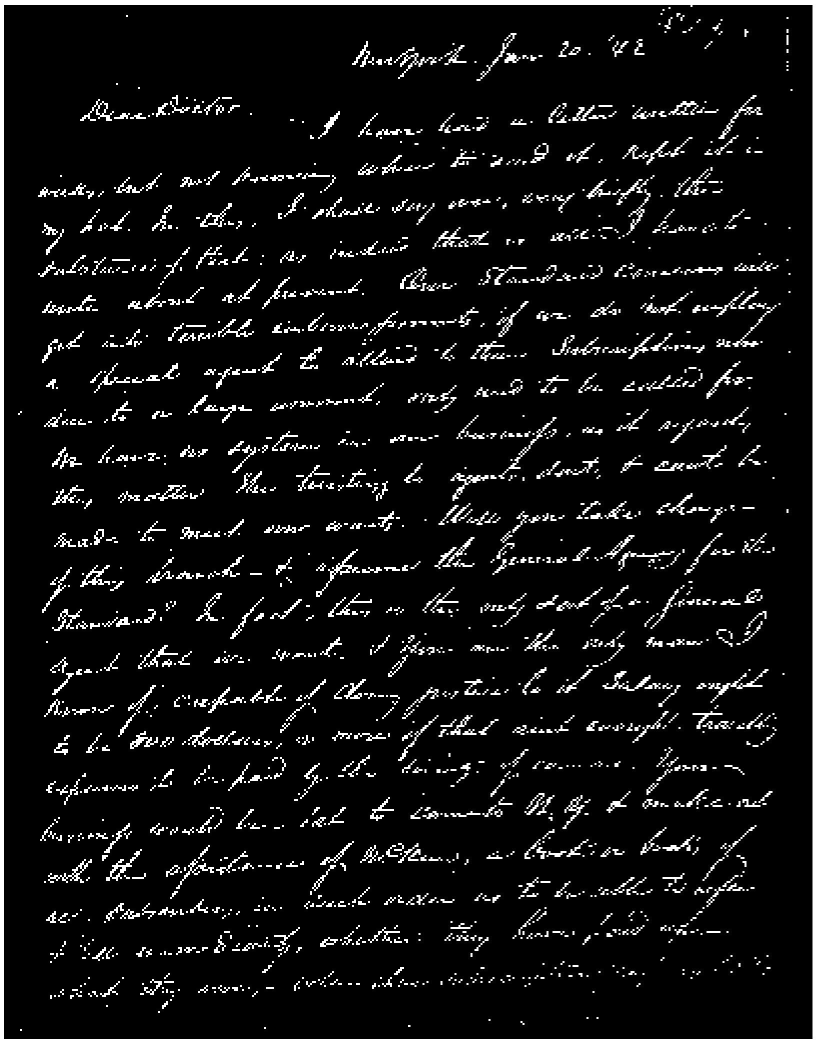
There is, however, a large amount of textual information on paper that needs to be indexed and retrieved efficiently. One solution for converting scanned paper documents into ASCII is to use Optical Character Recognition (OCR). Existing OCR technology works well with good machine printed fonts against good clean backgrounds. It works poorly if the text is handwritten. We propose an alternative solution for indexing handwritten text when a large corpus of texts written by a single person exists.

Specifically the problem being addressed in this paper is the indexing of historical manuscripts. These manuscripts are largely written in a single hand and most of them are unpublished. For example, even the collected works of well known people like W. E. B. Du Bois, the African American civil rights leader, and Margaret Sanger, a pioneer in birth control are mostly unpublished. Both left a substantial amount of their work and correspondence written in their own hand. It is unlikely that all of this material will ever be published.

Such manuscripts are, however, valuable resources for scholars as well as others who wish to consult the original manuscripts. It would, therefore, be useful to index them to allow rapid perusal. Since conventional OCR and text retrieval engines cannot be used, this paper proposes an alternative strategy for indexing such documents.

The indexing scheme proposed here also simplifies reading documents where the handwriting is hard to read. A scanned page from the correspondence of Erasmus Darwin Hudson (1809-1880) - an anti-slavery organizer and pioneer orthopaedic surgeon - is shown in Figure 1. This page is part of a letter from James S. Gibbons to Erasmus Hudson. The authors are still unable to decipher some of the words on this page - although the indexing scheme suggested here did help in deciphering some of the other words.

Since the document is written by a single person, the assumption is that the variation in the word images will be small. The proposed solution will match the actual word images against each other to create equivalence classes. Each equivalence class will consist of multiple instances of the same word. Each word will have a link to the page it came from. The number of words in each



New York. Jan 20. '42

Dear Doctor.

[illegible]

Figure 1: Manuscript from the Hudson collection (1842).

equivalence class will be tabulated. Those classes with the largest numbers of words will probably be stopwords i.e. conjunctions like "and" or articles like "the". Classes containing stopwords are eliminated (since they are not very useful for indexing). A list is made of the remaining classes. This list is ordered occuring to the number of words contained in them. The user provides ASCII equivalents for a representative word in each of the top m (say m = 2000) classes. The words in these classes can now be indexed. This technique will be called "wordspotting" as it is analogous to "wordspotting" in speech processing [GFJ95].

The proposed solution completely avoids machine recognition of handwritten words as this is a difficult task [MOR92]. Robustness is achieved compared to OCR systems for two reasons

1 Matching is based on entire words. This is in contrast to conventional OCR systems which essentially recognize characters rather than words.

2 Recognition is avoided. Instead a human is placed in the loop when ASCII equivalents of the words must be provided.

The present paper deals with the first part of the problem where the scanned document is segmented into word images and the word images are matched against each other. A future paper will deal with the rest of the system. The matching phase of the problem is expected to be the most difficult part of the problem. This is because unlike machine fonts, there is some variation in even a single person's handwriting. This variation is difficult to model.

In this paper, two different matching techniques are discussed. The first models the transformation as a translation (i.e. shift) while the second models it as a general affine transformation.

2. Prior Work

The traditional approach to indexing documents involves first converting them to ASCII [BOK92], and then using a text based retrieval engine [SAL88,TUR92]. Scanned documents can be converted into ASCII by first segmenting a page into words and then running them through an OCR [BOK92]. The OCR segments the words further into characters and then attempts to recognize the characters using statistical pattern classification [BOK92,MOR92]. This approach has been highly successful with good clean machine fonts against clean backgrounds. It has had much more limited success when handwriting is used. Primarily, this is because character segmentation is much more difficult in the presence of handwriting and also because of the wide variability in handwriting (not only is there variability between writers, but a given person’s writing itself varies).

An approach similar to ours has been used to recognize words in documents which use machine fonts [KHO93]. The word images are compared against each other and divided into equivalence classes. The words within an equivalence class - all of which are presumably identical - are used to construct a noisefree version of the word. This word is then recognized using an OCR. Recognition rates are much higher than when the OCR is used directly [KHO93].

Machine fonts have a number of advantages over handwriting. Multiple instances of a given word printed in the same font are identical except for noise. This situation does not hold for handwriting. Multiple instances of the same word on the same page by the same writer show variations. The variations are many - these include scaling of the words with respect to each other, small changes in orientation, and changes in the lengths of descenders and ascenders.

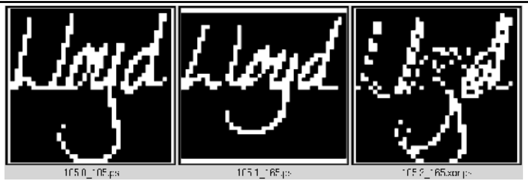


Figure 2: XOR of images.

In Figure 2, the first two images are two instances of the same word from the same document, written by the same writer. The third image which is the XOR image under optimal translation shows that the two words are written slighly differently. It may thus be necessary to account for these variations.

3. Outline of Algorithm

1. A scanned greylevel image of the document is obtained.

2. The image is first reduced by half by Gaussian filtering and subsampling.

3. The reduced image is then binarized by thresholding the image (note the thresholding is done in such a way that the characters are white and the background black).

4 . The binary image is now segmented into words. this is done by a process of smoothing and thresholding described later.

5. A given word image (i.e. the image of a word) is used as a template. and matched against all the other word images. This is repeated for every word in

the document. The matching is done in two phases. First, the number of words to be matched is pruned using the areas and aspect ratios of the word images - the word to be matched cannot have an area or aspect ratio which is too different from the template. Next, the actual matching is done by using a matching algorithm. Two different matching algorithms are tried here. One of them only accounts for translation shifts, while the other accounts for affine matches. The matching divides the word images into equivalence classes - each class presumably containing other instances of the same word.

6. Indexing is done as follows. For each equivalence class, the number of elements in it is counted. The top n equivalence classes are then determined from this list. The equivalence classes with the highest number of words (elements) are likely to be stopwords (i.e. conjunctions like ‘and’ , articles like ‘the’, and prepositions like ‘of’) and are therefore eliminated from further consideration. Let us assume that of the top n, m are left after the stopwords have been eliminated. The user then displays one member of each of these m equivalence classes and assigns their ASCII interpretation. These m words can now be indexed anywhere they appear in the document.

We now discuss these techniques in detail.

4. Word Segmentation

Since the purpose of this paper is to demonstrate the feasibility of word spotting, a simple technique is used for segmenting words. The method works reasonably well on the images tested so far. It is expected that this technique will be improved with further use.

The technique assumes that a binary image of each page is available and further assumes that the words are white against a dark background (if it is otherwise in the original image, the image can be inverted). Since the spacing between adjacent characters in a word is smaller than the spacing between adjacent words, a new image is constructed using a smoothing and thresholding operation. If two white pixels are separated by less than a certain distance k, the intermediate pixels are made white. This is done in the horizontal direction k_{horiz} . In the case of handwriting, this procedure also needs to be performed in the diagonal direction - mainly to prevent descenders from breaking up. k_{diag} . Note that each of these window operations may be viewed as a smoothing and thresholding operation or as a morphological closure operation. Connected components are now recovered from this image. A minimum bounding rectangle is now constructed using the connected components. The minimum bounding rectangles essentially give a segmentation of the page into words. Figure 3 shows an example. Certain errors do occur; for example the dot over the i is segmented as a separate word. This is ignored by requiring that word images have a minimum size. Other errors in segmentation may also occur because the writer left a large gap between parts of a word in one instance but did not do so when writing the word again.

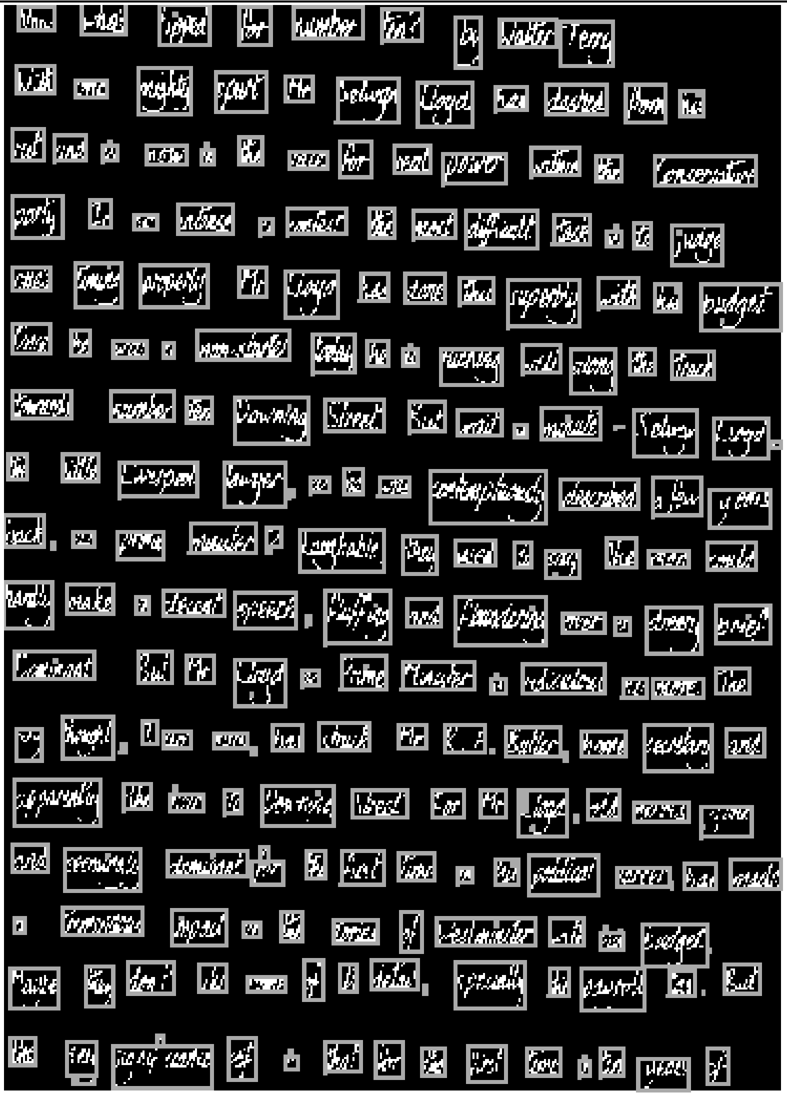


Figure 3: Page segmentation of the Senior document.

A number of algorithms exist in the literature for segmenting words from binary images and essentially any of them can be used [WAH82,WAN89].

5. Determination of Equivalence Classes

The matching is done in a number of phases. First, the number of possible words that need to be matched is pruned by using the areas and aspect ratios of the words. Since, the entire document is written by the same hand, it is expected that variations in size will be small. Thus the pruning can be done on the basis of the area of the word images and the aspect ratios of the word images.

5.1. Pruning

It is assumed that;

$1/\alpha <= A_{word}/A_{template} <= \alpha$.

where $A_{template}$ is the area of the template and A_{word} is the area of the word to be matched. It is also assumed that

$1/\beta <= Aspect_{word}/Aspect_{template} <= \beta$.

where $Aspect_{template}$ is the aspect ratio (width/height) of the template and $Aspect_{word}$ is the aspect ratio of the word to be matched.

α and β should not be too small so that valid words are omitted, nor too large so that too many words are passed onto the matching phase. The average value of the area ratio and the ratio of aspect ratios determine a lower bound or minimum value for α and β. These average values may be determined statistically by sampling a small set of known documents.

The average of the area ratio over all matching words is computed as follows. Assume that all possible matches for every word are known. The area ratio is then computed for all pairs of matching words. If any of these numbers is less than one, that value is replaced by its inverse (taking the average directly would give a number close to 1.0). The average of the resulting area ratios is then taken.

It turns out that words with only one or two characters may have large area ratios and bias the results. However, most words with only one or two characters are stop words which are not useful for indexing. The average is, therefore, computed by considering words of length 3 or greater (alternatively, words of length 4 or greater could be used but the former gives a more conservative estimate).

The minimum value of β may be computed in the same manner. The actual values of α and β used are larger than the minimum values so that valid words may not be missed. There is considerable leeway in the choice of these parameters. In the experimental section it is shown that the minimum value of the average area ratio for the two documents used here is 1.20 and that the results do not differ significantly whether α is chosen to be 1.22 or 1.3.

Typical values of α used in the experiments range between 1.2 and 1.3 while typical values of β used in the experiments range between 1.4 and 1.7.

5.2. Matching

The template is then matched against the word of each image in the pruned list (actually the number of words to be matched can be further restricted by eliminating all words which have already been placed in equivalence classes). The matching function must satisfy two criteria

1. It must produce a low match error for words which are similar to the template.
2. It must produce a high match error for words which are dissimilar.

Two matching algorithms have been tried. The first algorithm - Euclidean Distance Mapping (EDM) [DAN80] - assumes that no distortions have occured except for relative translation and is fast. This algorithm usually ranks the matched words in the correct order (i.e. valid words first, followed by invalid words) when the variations in words is not too large. Although, it returns the lowest errors for words which are similar to the template, it also returns low errors for words which are dissimilar to the template. The second algorithm [SCO91], referred to as SLH here, assumes an affine transformation between the words. It thus compensates for some of the variations in the words. It is shown in the experiments that the average precision for the SLH algorithm is much better than that for the EDM algorithm. However, as currently implemented the SLH algorithm is much slower than the EDM algorithm (we expect to be able to speed it up).

6. Using Euclidean Distance Mapping for Matching

This approach is similar to that used by [KHO93] to match machine generated fonts. Consider two images to be matched. There are three steps in the matching:

1. Alignstep: First the images are roughly aligned. In the vertical direction, this is done by aligning the baselines of the two images. The baseline is computed as follows. The difference in the number of white pixels between adjacent scan lines is computed. The point at which the difference is maximum is declared to be the baseline. The baseline computation is performed for both images, and the images then shifted so that they are aligned. In the horizontal direction, the images are aligned by making their left hand sides coincide. The alignment is, therefore, expected to be accurate in the vertical direction and not as good in the horizontal direction. This is borne out in practice.
2. XORstep: Next the XOR image is computed. This is done by XOR'ing corresponding pixels. An example of two images and the corresponding XOR image is shown Figure 2. A match error E_{XOR} may be computed by finding the number of white pixels in the XOR image. The XOR image match error is in general not accurate enough for matching. Notice that XOR images may consist of either isolated pixels or pixels in a blob. The error measure computed above gives equal weight to both. However, an isolated pixel in the XOR image may be due to noise while a blob may be due to a major mismatch. Therefore, blobs should be given more weight. This can be done by using an Euclidean distance mapping.
3. EDMstep: An Euclidean distance mapping [DAN80] is computed from the XOR image by assigning to each white pixel in the image, its minimum distance to a black pixel. Thus a white pixel inside a blob will get a larger distance than an isolated white pixel. An error measure E_{EDM} can now be computed by adding up the distance measures for each pixel.
4. Although the approximate translation has been computed using step 1, this may not be accurate and may need to be fine-tuned. Thus steps 2 and 3 are repeated while sampling the translation space in both x and y. A minimum error measure E_{EDMmin} is computed over all the translation samples.

7. SLH Algorithm for Matching

The EDM algorithm does not discriminate well between good and bad matches. In addition, it fails when there is significant distortion in the words. This happened with the writing of Erasmus Hudson (Figure 1). Thus a matching algorithm which models some of the variation is needed. A second matching algorithm (SLH) which models the distortion as an affine transformations was, therefore tried (note that it is expected that the real variation is probably much more complex).

An affine transform is a linear transformation between coordinate systems. In two dimensions, it is described by **r**' = **Ar** + **t** where **t** is a 2-D vector describing the translation, **A** is a 2 by 2 matrix which captures the deformation, **r**' and **r** are the coordinates of corresponding points in the two images between which the affine transformation must be recovered. An affine transform allows for the following deformations - scaling in both directions, shear in both directions and rotation.

The literature describes a number of algorithms to recover affine transforms [BER92, GOL94, MAN94A, MAN94B, SCO91, SZE94]. A number of criteria restrict the choice of algorithms.

1. One of the requirements of the problem being considered here is that the algorithm must recover both the correspondence between images and the affine transform simultaneously.
2. Greylevel matching techniques are not necessarily appropriate for matching binary images.

These criteria restrict the choice of algorithm to those that operate on points. Scott and Longuet-Higgins [SCO91] proposed an algorithm to recover the correspondence between two sets of points I and J under an affine transform (actually the Scott and Longuet-Higgins algorithm does not require that the correspondence between the two sets of points be affine but only in the case of affine transforms has it been shown to recover the correct correspondence). This algorithm will now be described.

Two sets of points I and J are created as follows. Every white pixel in the first image is a member of the set I. Similarly, every white pixel in the second image is a member of set J. First, the centroids of the point sets are computed and the origins of the coordinate systems is set at the centroid. An adjacency matrix **G** is then computed. The entries G_{ij} are Gaussian weighted distances between a point i in set I and a point j in set J. Each entry G_{ij} is given by $G_{ij} = \exp(- r_{ij}^T r_{ij}/(2\,\sigma^2))$ where r_{ij} is the Euclidean distance between i and j. The matrix **G** is then diagonalized using singular value decomposition (SVD) to give **G = T D U** where **D** is a diagonal matrix and **T** and **P** are orthogonal matrices. The diagonal entries in **D** are replaced by 1's to give an m by n matrix **E**. The pairing matrix **P**

= T E U indicates the strength of the attraction between points i and j. Thus a correspondence between two points i and j is posited only if the entry P_{ij} is the greatest element in row i and the greatest element in column j. Intuitively **P** is the matrix which correlates best with the **G** matrix in the sense of maximizing the trace of $\mathbf{P}^T\mathbf{G}$. The transformation can then be computed using the recovered correspondence. Scott and Longuet-Higgins showed that if σ is chosen large enough , the method would compute the correspondence correctly for translations, scale changes (i.e. expansions, contractions) and shears. Here, as in intensity based algorithms large values of σ are useful in recovering large translations. However, the method cannot be shown to compute the correct correspondence if a rotation is involved. In practice, small rotations can be handled most of the time.

Note that some points will have no correspondence i.e what the algorithm returns is a one to one correspondence between some subset of I and some subset of J.

Given the (above) correspondence between point sets I and J, the affine transform can be computed in a straightforward manner. The correct affine transform $\mathbf{A},\mathbf{t}$ is that transform which minimizes the following least mean squares criterion: $E_{SLH} = l\ (I_l - \mathbf{A}\ J_l - \mathbf{t})^2$ where I_l,J_l are the (x,y) coordinates of point I_l and J_l respectively.

The values of $\mathbf{A},\mathbf{t}$ can be computed in closed form by minimizing the above expression (i.e. differentiating and setting it to zero). The values are then plugged back into the above equation to compute the error E_{SLH} . The error E_{SLH} is an estimate of how dissimilar two words are and the words can, therefore, be ranked according to it.

One disadvantage of computing the affine parameters is that in certain situations two very different words can give a low error rate E_{SLH} (this is similar to the fact that given enough parameters any continuous function can be fitted by a polynomial). If, however, the range of values of the affine parameters is constrained, this is unlikely to occur. It will, therefore, be assumed that the variation for valid words is not too large. This implies that if A_{11} and A_{22} are considerably different from 1, the word is probably not a valid match.

The affine matching algorithm is much more accurate than the Euclidean distance mapping technique. The current implementation of this technique is slow because of the need to compute the SVD of a large matrix (often the matrix may have a few hundred rows and columns). However, the **G** matrix is sparse (since the values of σ are low). The computation of the SVD can, therefore, be speeded up by utilizing methods which compute the SVD of a sparse matrix quickly . This will be done in future implementations.

Note: The SLH algorithm assumes that pruning on the basis of the area and aspect ratio thresholds is performed.

8. Experiments

The performance of both techniques was tested on two handwritten pages, each written by a different writer. The first page was obtained from the DIMUND document server on the internet. This page can be obtained from http://documents.cfar.umd.edu/resources/database/handwriting.database.html and was scanned by Andrew Senior (this page will be referred to as the Senior document). The handwriting on this page is fairly neat. The second page is from an actual archival collection - the Hudson collection from the library of the University of Massachusetts. The page used is a letter written by James S. Gibbons to Erasmus Darwin Hudson. The handwriting on this page is difficult to read and in fact the indexing technique helped in deciphering some of the words.

The experiments will show examples of how the matching techniques work. The experiments show rankings and match errors for a few selected words. Recall precision curves for both documents are also presented. The recall precision curves are generated by considering queries (templates) for which there is at least one other (besides itself) possible match in the document. All rankings were produced by matching the template with every word left in the pruned class. However, only a few of the matches are displayed in the figures and tables.

For page segmentation (see section 4), $k_{horiz} = 9$ and $k_{diag} = 3$ were chosen. The parameters were determined empirically by varying them and choosing one which gave the best segmentation. Table 1 shows the number of words in each document, the number of words which have length greater than 3 (i.e. 3 or more characters) and the number of words of length greater than 4.

Table 2 shows statistical information determined from the documents for the purpose of determining the thresholds for pruning. The numbers are calculated for words with three or more characters in them.This is done so that words with two characters or less do not skew the results. Such words (with two characters or less) are likely to be stopwords with little usefulness as far as indexing is concerned.

Document	# of words	# of words of length >= 3	# of words of length >= 4
Senior	192	155	130
Hudson	153	113	101

Table 1: Number of words in each document.

The first column in Table 2 lists the documents for which statistical information was collected. The second and third columns list the average value of the area ratio and the average of the ratio of aspect ratios for words with three or more characters in the document.

Document	Avg. area ratio	Avg. ratio of aspect ratios
Senior	1.09	1.10
Hudson	1.20	1.15

Table 2: Statistical information for words with 3 or more characters.

The minimum value of the thresholds (α_{min} and β_{min}) for pruning may now be determined. They are obtained by finding the maximum over both documents of the averages of the area ratio and the ratio of aspect ratios. Using Table 2, they are given by:

$$\alpha_{min} = 1.20,\ \beta_{min} = 1.15$$

The actual values used for α and β are much higher to allow for some variation. For the Euclidean Distance Matching technique, $\beta = 1.4$ was used for both documents. Note that this is so large compared to β_{min} that very few valid matches are likely to be eliminated. The EDM algorithm was tried with $\alpha = 1.22$ and 1.3. The experimental results in the following subsection show that both values of α give roughly the same results. There is, therefore, considerable leeway in choosing the pruning thresholds.

The current implementation of the SLH algorithm is slow. Therefore, the EDM algorithm is run, a threshold picked and words which have a match error under this threshold are then processed by the SLH algorithm. The actual value of the threshold is not crucial. Before the EDM algorithm is run, the words are pruned as before using the area and aspect ratios. To ensure that the SLH algorithm did most of the matching and pruning, the thresholds were picked to be conservative. $\alpha = 1.4$ and $\beta = 1.7$ were chosen.

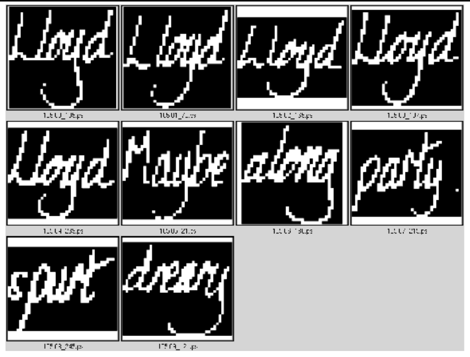


Figure 4: Rankings for the template "Lloyd" using the EDM algorithm.

8.1. Experiments Using the EDM Algorithm.

The EDM algorithm was run on both documents. All experiments were conducted by matching the template with every word in the document - the Senior document has 192 words and the Hudson has 153 words (See Table 1). The translations were sampled to within ± 4 pixels in the x direction and ± 1 pixel in the y direction. Increasing the translation sample space did not change the results.

Some of the figures below show examples of the rankings achieved. In these figures, the first word is the template. The template is followed by words ranked according to the error measure. A cut-off threshold is used to limit the number of words displayed. This threshold is common to all the experiments.

Token	Word	Area	$E_{EDM\ min}$	X	Y
105	Lloyd	1360	0.000	0	0
70	Lloyd	1224	0.174	0	0
165	Lloyd	1230	0.175	-2	0
197	Lloyd	1400	0.194	4	0
239	Lloyd	1320	0.197	-3	0
21	Maybe	1147	0.199	-1	0
180	along	1156	0.200	1	0
215	party	1209	0.202	1	0
245	spurt	1170	0.205	-1	0
121	dreary	1435	0.206	3	0

Table 3: Rankings and match errors for the template "Lloyd" using EDM algorithm.

On the Senior document, the EDM algorithm does quite well. This performance is to be expected because the handwriting is fairly neat. A typical example is shown in Figure 4. In the figure, the first word is the template "Lloyd". The figure shows that the four other instances of "Lloyd" are ranked before any of the other words. Table 3 shows that the match error for the other instances of "Lloyd" is less than that for any other word. In the table, the first column lists the Token number, the second column gives a transcription of the word, the third column shows the area in pixels, the fourth gives the match error and the last two columns specify the translation in the x and y directions respectively. Note the significant change in area of the words.

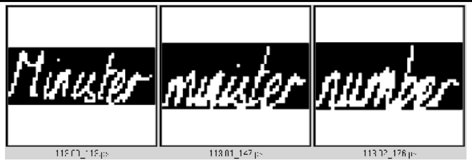


Figure 5: Rankings for the template "Minister" using the EDM algorithm.

In English, the first letter in a word is capitalized when the word begins a sentence and not otherwise (unless it is a proper noun). Thus it is desirable that the technique be relatively insensitive to this capitalization. Figure 5 and Table 4 show an example of this. The word "minister" is the highest ranked word obtained for the template "Minister" inspite of the fact that "minister" begins with a lower case letter while "Minister" starts with an uppercase letter.

Token	Word	Area	E_{EDM}	X	Y
113	Minister	1134	0.000	0	0
147	minister	1078	0.210	-1	0
176	number	1104	0.285	2	0

Table 4: Rankings and match errors for the template "Minister" using the EDM algorithm.

The algorithm performs poorly in two respects. It shows poor discrimination between valid words and invalid words. For example, in Table 3 the last "Lloyd" has a match error of 0.197 while the next word in the ranking "Maybe" has a match error of 0.199. Thus it is difficult to discriminate between valid and invalid words using the error measure.

The performance of a retrieval algorithm is often evaluated in terms of its recall and precision. Recall is defined as the "proportion of relevant material actually retrieved in answer to a search request" [van79] while precision is defined as the "proportion of retrieved material that is actually relevant" [van79].

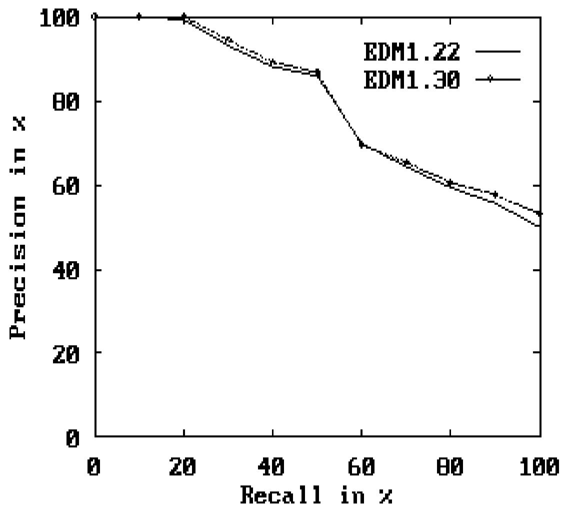


Figure 6: Recall precision graph for the Senior document using the EDM algorithm.

Figure 6 shows a graph of precision versus recall for the Senior document using the EDM algorithm. The plots were generated by using only words with more than 3 characters as queries. Only those words were used as queries for which there was at least one other instance of the word in the document. The number of queries was 59.

The two plots were generated using different values of the area pruning threshold (α = 1.22 and α = 1.3). Figure 6 shows that there is no significant difference in performance using either pruning threshold. The average precision using α = 1.22 is 78.7% while for α = 1.3 it is 79.7%.

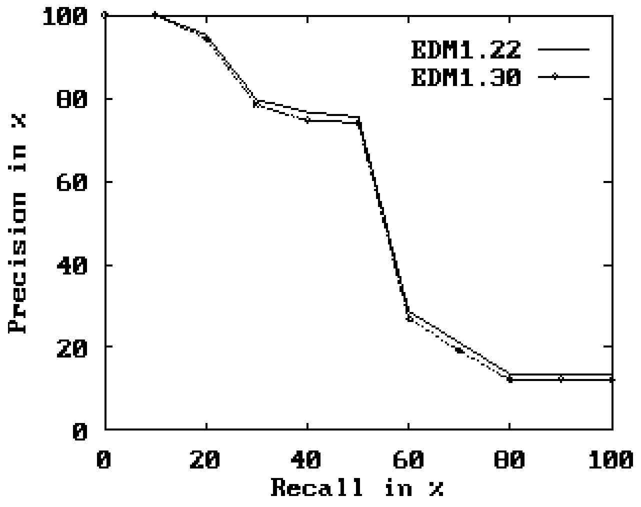


Figure 7: Recall precision graph for the Hudson document using the EDM algorithm.

The EDM algorithm was also tested on the Hudson document. Figure 7 shows the recall precision graph for the Hudson document. The

Token	Word	Area	E_{EDMmin}	X	Y
280	Standard	1530	0.000	0	0
239	comment	1722	0.203	-4	0
94	come to	1241	0.212	1	0
45	whether	1258	0.212	1	0
186	branch	1743	0.218	0	0
56	subscribes	1900	0.228	-4	0
283	substances	1479	0.231	1	0
167	Standard	1440	0.231	1	0

Table 5: Rankings and match errors for the template "Standard" using the EDM algorithm.

average precision using $\alpha = 1.22$ was 56.1% while for $\alpha = 1.3$ it was 57.9%. The poorer performance on the Hudson document can be attributed to the handwriting. The handwriting in the Hudson collection (Figure 1) is difficult to read even for humans looking at grey-level images at 300 dpi.

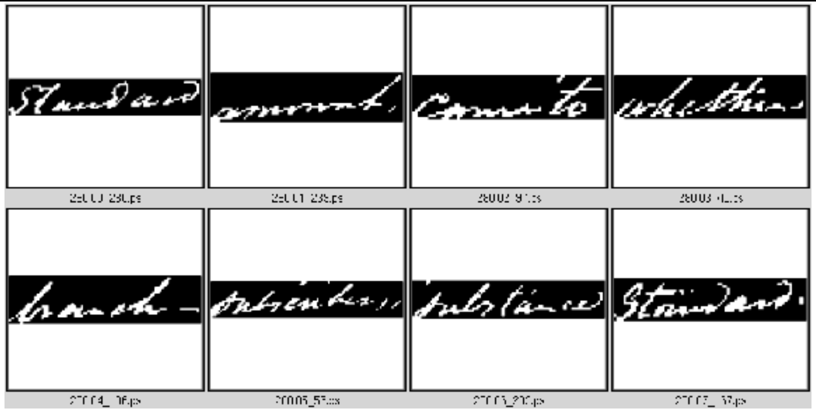


Figure 8: Rankings for the template "Standard" using the EDM algorithm.

An example of failure from the Hudson collection is now shown. The word “Standard” from the Hudson collection was matched. Figure 8 and Table 5 show the results of this matching. The performance is not very good. The reason is that the words are written differently. In the template, there is a gap between the “t” and the “a”. However, in the second example of “Standard” there is no gap. This implies that a technique which models some kind of distortion may be needed.

8.2. Experiments Using the SLH Algorithm

Experiments were performed using the Senior document. Since the current

Token	Word	Area	Pts.	E_{SLH}	A		T
105	Lloyd	1368	233	0.00	1.00	0.00	0.00
					0.00	1.00	0.00
197	Lloyd	1400	199	1.302	0.96	-0.04	1.58
					0.01	1.04	0.14
70	Lloyd	1224	176	1.356	0.94	0.09	-1.02
					0.03	0.92	-1.38
165	Lloyd	1230	189	1.631	1.03	0.05	-0.43
					-0.01	0.87	-2.60
239	Lloyd	1320	203	1.795	0.99	-0.05	1.44
					0.03	1.07	2.21
157	lawyer	1518	185	3.393	0.96	-0.03	1.89
					0.05	1.11	0.03
240	Selwyn	1564	188	3.673	0.94	0.06	-4.23
					0.05	1.05	-0.75
91	thought	1178	181	3.973	0.97	0.03	2.33

Table 6: Rankings and match errors for the template "Lloyd" using the SLH algorithm.

version of the SLH algorithm is slow, the initial matches were pruned using the EDM algorithm and then the SLH algorithm run on the pruned subset.

To account for the large variations in the Hudson papers, the area threshold α was fixed at 1.4 and the aspect ratio threshold at 1.7. The value of σ depends on the expected translation. Since it is small, $\sigma = 2.0$. A lower value of $\sigma = 1.5$ yielded poorer results.

The matches for the template “Lloyd” are shown in Table 6. The succesive columns of the table, tabulate the Token Number, the transcription of the word, the area of the word image, the number of corresponding points recovered by the SLH algorithm, the match error E_{SLH} using the SLH algorithm and the affine transform. The entries are ranked according to the match error E_{SLH}. If either of A_{11} or A_{22} is less than 0.8 or greater than 1/0.8, that word is eliminated from the rankings. A comparison with Table 3 shows that the rankings change. This is not only true of the invalid words (for example the sixth entry in Table 1 is “Maybe” while the sixth entry in Table 5 is “lawyer”) but is also true of the “Lloyd’’s. Both tables rank instances of “Lloyd” ahead of other words. The technique also shows a much greater discrimination in match error - the match error for “lawyer” is almost double the match error for the fifth “Lloyd”.

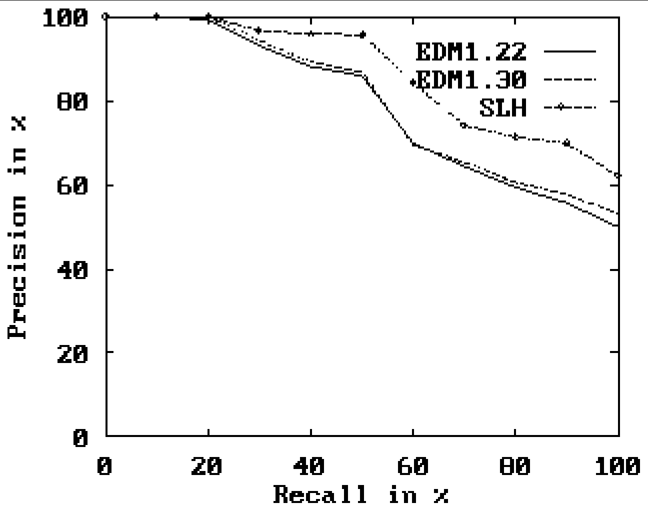


Figure 9: Recall precision graph for the SLH and the EDM algorithms on the Senior document.

Figure 9 compares the recall and precision of the EDM algorithm and the SLH algorithm on the Senior document. Note the significant improvement in performance. As before, words with three or more characters of which there was at least one other instance were used as queries. For the SLH algorithm, the average precision came out to be 86.3% compared to 79.7% for the EDM algorithm.

Token	Word	Area	Pts.	E_{SLH}	A		T
1	they	899	108	0.000	1.00	0.00	0.00
					0.00	1.00	0.00
43	they	891	97	0.636	0.92	0.05	-0.93
					0.05	1.01	1.62
156	only	775	85	3.172	0.89	-0.22	1.53
					0.03	1.20	-0.38
191	this?	696	83	8.466	0.97	-0.15	1.40
					-0.05	1.14	7.23

Table 7: Rankings and match errors for the template "they" using the SLH algorithm.

Token	Word	Area	Pts.	E_{SLH}	A		T
280	Standard	1530	251	0.000	1.00	0.00	0.00
					0.00	1.00	0.00
167	Standard	1440	183	4.36	1.03	0.10	5.07
					-0.01	0.94	0.33
56	subscribers	1900	196	7.816	0.99	0.20	1.27
					0.00	0.94	-0.38
283	substance	1479	183	39.18	0.92	0.12	-1.39
					-0.02	0.82	1.02

Table 8: Rankings and match errors for the template "Standard" using the SLH algorithm.

The SLH algorithm was also run on the Hudson document (Figure 1). This document is particularly difficult because of the poor handwriting. The writing is difficult for people to read.

Performance on templates like “they” is good as shown in and Table 7. Good discrimination between valid and invalid words is also obtained using the error measure E_{ESH} . (In this particular case, the EDM algorithm also ranks correctly, but the discrimination is not so good).

Finally, we look at the word “Standard” on which the EDM method did not rank correctly The SLH method produces the correct ranking inspite of the significant distortions in the word (see Figure 10) and Table 8. As discussed before the first instance of “Standard” is written with additional gaps between the “t” and the “a” and the “d” and the “a” (visible in Figure 10).

8.3. Comment

It is clear that the SLH algorithm ranks words correctly almost all the time. In some situations, the discrimination between valid and invalid words needs to be improved. However, it seems to be a reasonable algorithm to base wordspotting on.

9. Conclusion

The work clearly demonstrates the feasibility of indexing handwritten words when there exists a corpus of words written by a single author. Two algorithms were used for ranking matches of handwritten words with a template. The first (EDM) based on Euclidean distance mapping does not account for any distortions and thus performs poorly when the handwriting is bad. The second (SLH) algorithm, based on an algorithm of Scott and Longuet Higgins, produces the correct rankings almost always - this is true even if the handwriting is bad.

Two areas need to be improved - speed and the discrimination between valid and invalid words.

10. Acknowledgements

This material is based on work supported in part by the National Science Foundation, Library of Congress and Department of Commerce under cooperative agreement number EEC-9209623, by NSF contract number CDA 9502639 and by NRaD contract number N66001-94-D-6054. Any opinions, findings and conclusions or recommendations expressed in this material are the authors and do not necessarily reflect those of the sponsors.

Chengfeng Han and David Hirvone provided assistance with programming. Jonathan Lim and Bob Heller provided help with systems software. The page from the Hudson collection was scanned in and provided by Gail Giroux of the University of Massachusetts W. B. DuBois library.

11. References.

[BER92] Bergen, J. R., Anandan, P., Hanna, K. J., and Hingorani, R., Hierarchical model-based motion estimation, in Proc. 2nd European Conference on Computer Vision, 237--252, 1992.

[BOK92] Bokser, M., Omnidocument technologies, in Proceedings IEEE, 80(7):1066--1078, 1992.

[DAN80] Danielsson, Per-Erik, Euclidean distance mapping, in Computer Graphics and Image Processing, 14:227--248, 1980.

[GFJ95] Jones, G. J. F., Foote J. T., Sparck Jones K., and Young, S. J.,
Video mail retrieval: The effect of word spotting accuracy on precision, in International Conference on Acoustics, Speech and Signal Processing, vol 1, 309--316, 1995.

[GOL94] Gold S., Rangarjan, A., Lu, C. P., Pappu, S., and Mjolsness, E.,
New algorithms for 2d and 3d point matching: Pose estimation and correspondence, to appear in Pattern Recognition.

[KHO93] Khoubyari, S. and Hull, J.J., Keyword location in noise document image, Second Annual Symposium on Document Analysis and Information
Retrieval, UNLV, Las Vegas, 217--231, 1993.

[MOR92] Mori, S., Suen, C. Y. and Yamamoto, K., Historical review of OCR research and development, Proceedings of the IEEE, 80(7):1029--1058, 1992.

[MAN94A] Manmatha, R., Measuring the affine transform using gaussian filters.
in Proc. 3rd European Conference on Computer Vision, 159--164, 1994.

[MAN94B] Manmatha, R., A framework for recovering affine transforms using points, lines or image brightnesses, in Proc. Computer Vision and Pattern Recognition Conference, 141--146, 1994.

[SAL88] Salton, G., Automatic Text Processing, Addison-Wesley, 1988.

[SC091] Scott, G. L. and Longuet-Higgins, H. C., An algorithm for associating the features of two patterns, in Proc. Royal Society of London B, B244:21--26, 1991.

[SHA92] Shapiro, L. S. and Brady, J. M., Feature-based correspondence: An eigenvector approach, in Image and Vision Computing, 10:283--288, 1992.

[SZE94] Szeliski, R. and Coughlan, J., Hierarchical spline-based image registration in Proc. Computer Vision and Pattern Recognition Conference, 194--201, 1994.

[TUR92] Turtle, H.R. and W.B. Croft, W. B., A comparison of text retrieval models,in Computer Journal, 1992.

[van79] van Rijsbergen, C. J., Information Retrieval, Butterworths, London, 1979.

[WAH82] Wahl F., Wong. K., and Casey, R., Block segmentation and text extraction in mixed text/image documents, in Computer Vision Graphics and Image Processing, 20:375--390, 1982.

[WAN89] Wang D., and Srihari, S. N., Classification of newspaper image blocks using texture analysis, in Computer Vision Graphics and Image Processing, 47:327--352, 1989.