Word Spotting: A New Approach to Indexing Handwriting

R. Manmatha, Chengfeng Han and E. M. Riseman, *
Center for Intelligent Information Retrieval,
Computer Science Department,
University of Massachusetts, Amherst, MA-01003.
(manmatha)@cs.umass.edu

Keywords: Image Indexing, Matching, Matching under Affine transforms, Document Analysis, Novel Applications, Word Spotting.

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.

*This research was supported by the Center for Intelligent Information Retrieval and by ARPA grant number N66001-94-D-6054.

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 on paper or on videos not in machine readable format. One solution is to use Optical Character Recognition (OCR) to convert scanned paper documents into ASCII. 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. However such manuscripts are 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 aprt 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.

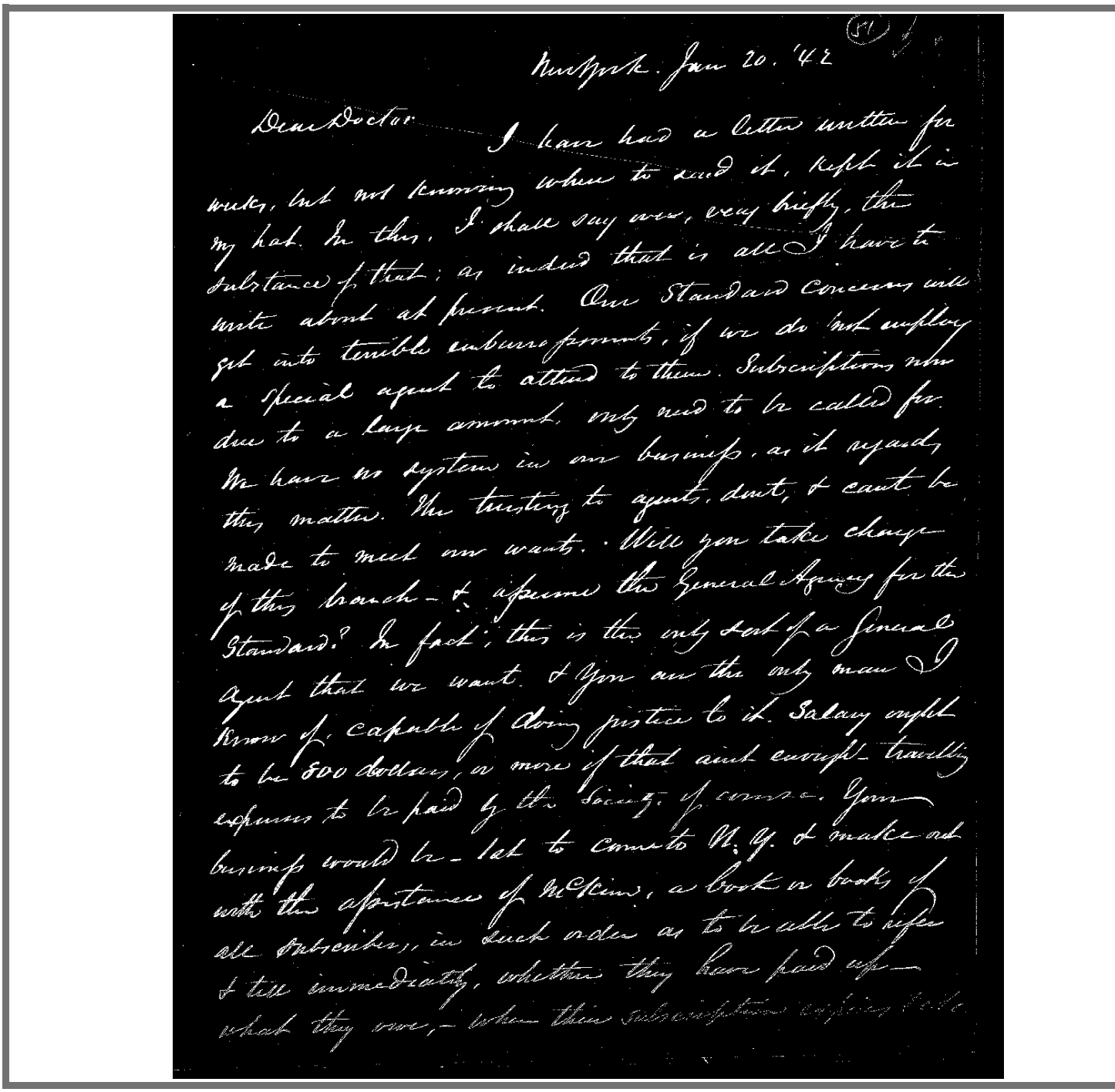

Figure 1: Manuscript from the Collected Papers of the Hudson Family

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 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 [7].

The proposed solution completely avoids machine recognition of handwritten words as this is a difficult task [12]. 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. Figure (2) shows two examples of the word Lloyd written by the same person. The last image is produced by XOR'ing these two images. The white areas in the XOR image

indicate where the two versions of "Lloyd" differ. This result is not unusual. In fact the differences are sometimes even larger.

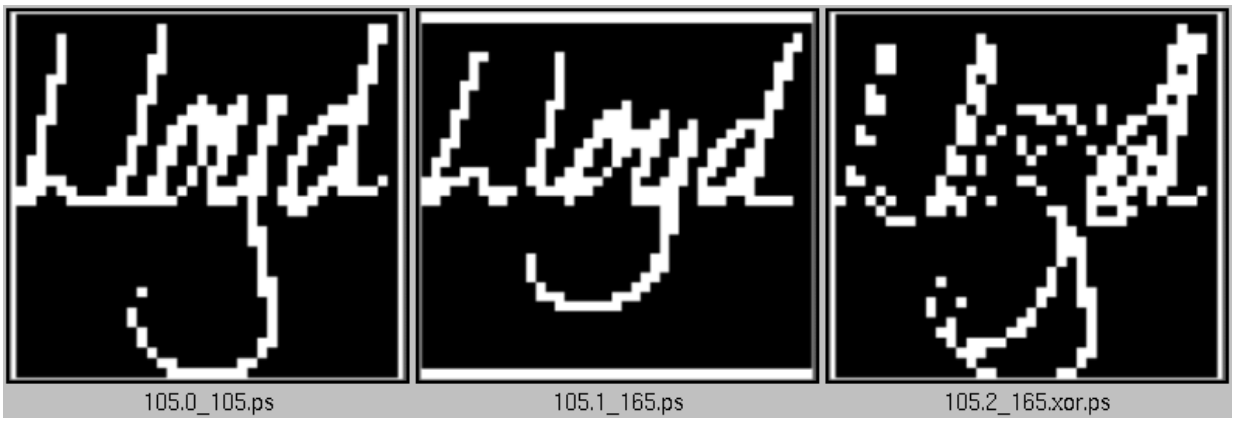

Figure 2: Two Examples of the Word "Lloyd" and the XOR image

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 and then using a text based retrieval engine [17, 13]. Scanned documents can be converted into ASCII by first segmenting a page into words and then running them through an OCR [2]. The OCR segments the words further into characters and then attempts to recognize the characters using statistical pattern classification [2, 12]. 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 [8]. 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 [8].

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. 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.

3.1 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. Other errors in segmentation may also occur because the writer left a large gap between parts of a word.

A number of algorithms exist in the literature for segmenting words from binary images and essentially any of them can be used [18, 5].

4 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.

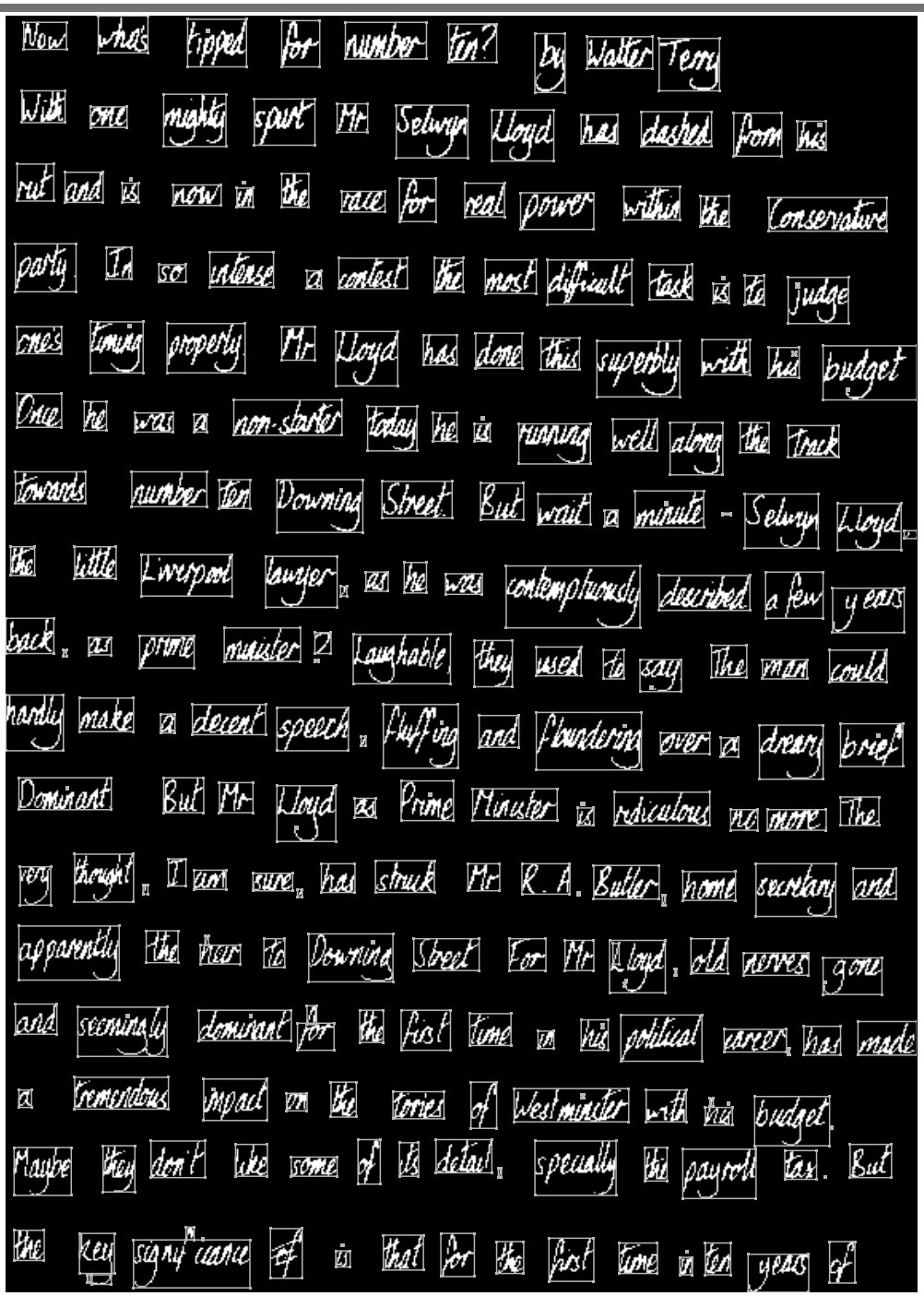

Figure 3: Segmentation of Page

4.1 Pruning

It is assumed that

$$\frac{1}{\alpha} \leq \frac{A_{word}}{A_{template}} \leq \alpha \tag{1}$$

where $A_{template}$ is the area of the template and A_{word} is the area of the word to be matched. Typical values of α used in the experiments range between 1.2 and 1.3. A similar filtering step is performed using aspect ratios (ie. the width/height ratio). It is assumed that

$$\beta \leq \frac{Aspect_{word}}{Aspect_{template}} \leq \beta \tag{2}$$

. Values of β used in the experiments range between 1.4 and 1.7. In both the above equations, the exact factors are not important but it should not be too large so that valid words are omitted, nor too small so that too many words are passed onto the matching phase.

4.2 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) - 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 [14],referred to as SLH here, assumes an affine transformation between the words. It thus compensates for some of the variations in the words. This algorithm not only ranks the words in the correct order for all examples tried so far, it also seems to be able to discriminate valid words from invalid words. As currently implemented the SLH algorithm is much slower than the EDM algorithm (we expect to be able to speed it up).

5 Using Euclidean Distance Mapping for Matching

This approach is similar to that used by [6] to match machine generated fonts. Consider two images to be matched. There are three steps in the matching:

1. 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. 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. However, 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. An Euclidean distance mapping [3] 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.

6 Experiments Using the EDM Algorithm

Experiments were performed on a handwritten page obtained from the DIMUND document server on the internet (this will be referred to as the Senior document) The handwriting on this page is fairly neat. The page was segmented into words using $k_{horiz} = 9$ and $k_{diag} = 3$ - the output is shown in Figure (3). The algorithm was then run on the segmented words. In the following figures, the first word shown is the template. After the template, the other words are ranked according to the match error E_{EDM}. The pruning was done with an area threshold of $\alpha = 1.2$ and an aspect ratio threshold of $\beta = 1.4$. 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.

In Figure (4), the template is the word "Lloyd". The figure shows that the four other instances of "Lloyd" are ranked before any of the other words. As Table (1) shows the match errors for other instances of "Lloyd" is less than that for any other word. In the table, the first column is the Token number (this is needed for identification purposes), the second column is 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 4: Rankings for template "Lloyd" using the EDM Algorithm(the rankings are ordered from left to right and from top to bottom).

Figure (5) and Table (2) display the results when the template "the" is used. In this case, there are a few instances where other words are ranked ahead of two instances of "the". Token numbers 191,33 and 161 are ranked ahead. Note that two of these 191 and 161 are actually instances of "he" which is fairly close to the correct word. In general it is expected that small words will have the largest errors. However, most small words are stopwords and are not useful for indexing. Therefore, the errors are not necessarily serious.

Token Number	Word	Area	E_{EDM}	Xshift	Yshift
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 1: Rankings and Match Errors for template "Lloyd" using the EDM Algorithm.

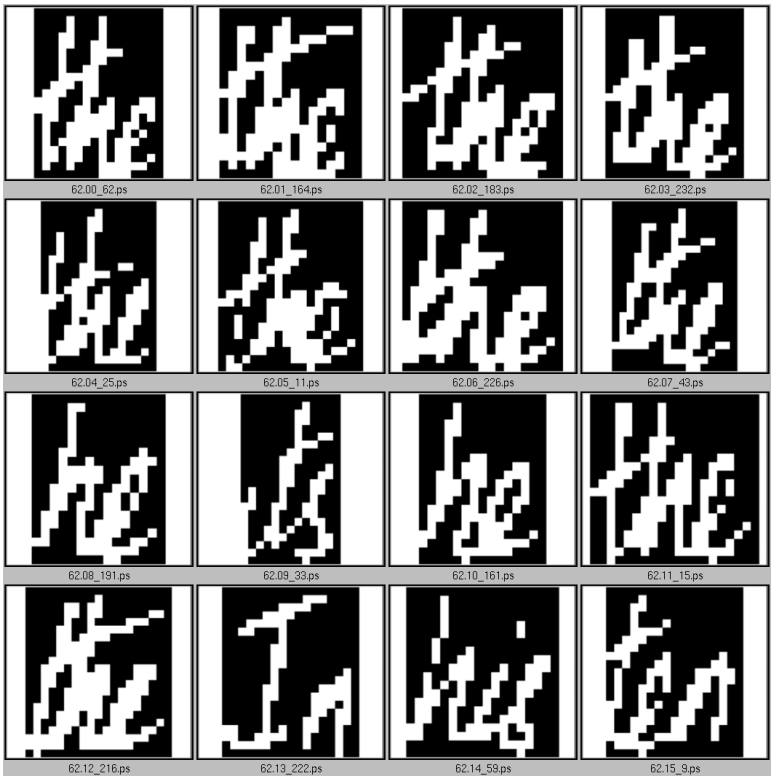

Figure 5: Rankings for template "the" using the EDM Algorithm (the rankings are ordered from left to right and from top to bottom).

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 (6) and Table (3) shows an example of this. The word "minister" is the highest ranked word obtained for the template "Minister"

Token Number	Word	Area	E_{EDM}	Xshift	Yshift
62	the	336	0.000	0	0
164	the	304	0.143	-3	0
183	the	380	0.149	-1	0
232	the	396	0.170	1	0
25	the	330	0.193	0	0
11	the	378	0.223	0	0
226	the	380	0.256	0	0
43	the	391	0.259	0	0
191	he	285	0.265	2	0
33	its	286	0.265	2	0
161	he	300	0.271	1	0
15	the	400	0.280	-1	0
216	the	418	0.289	0	0
59	his	357	0.312	3	0
222	In	360	0.315	0	0
9	ten	357	0.333	1	0

Table 2: Rankings and Match Errors for template "the" using the EDM Algorithm.

inspite of the fact that "minister" begins with a lower case letter while "Minister" starts with an uppercase letter.

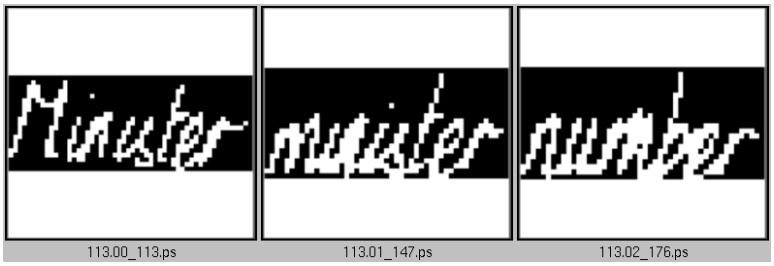

Figure 6: Rankings for template "Minister" using the EDM algorithm (the rankings are ordered from left to right and from top to bottom).

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

Table 3: Rankings and Match Errors for 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 (1) 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 algorithm also performs poorly when the hand writing is bad. For example, the handwriting in the Hudson collection (1) is difficult to read even for humans looking at grey-level images at 300 dpi (legend has it that Erasmus Hudson was berated by his wife for his bad handwriting). An example from the Hudson collection is now shown.

The word "Standard" from the Hudson collection was matched. Figure (7) and Table (4) 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.

Figure 7: Rankings for template "Standard" using the EDM Algorithm (the rankings are ordered from left to right).

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 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

Token Number	Word	Area	E_{EDMmin}	Xshift	Yshift
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 4: Rankings and Match Errors for template "Standard" using the EDM Algorithm.

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

$$\mathbf{r}' = \mathbf{A}\mathbf{r} + \mathbf{t} \tag{3}$$

where $\mathbf{t}$ is a 2-D vector describing the translation, $\mathbf{A}$ is a 2 by 2 matrix which captures the deformation, $\mathbf{r}'$ and $\mathbf{r}$ 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 [1, 10, 9, 16, 14, 15, 11] describes a number of algorithms to recover affine transforms. 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 [14] 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

$$G_{i,j} = exp(-r_{ij}^T r_{ij}/(2\sigma^2)) \tag{4}$$

where r_{ij} is the Euclidean distance between i and j. The matrix **G** is then diagonalized using singular value decomposition (SVD) to give

$$\mathbf{G} = \mathbf{TDU} \tag{5}$$

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**

$$\mathbf{P} = \mathbf{TEU} \tag{6}$$

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 $P^T G$. 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 sigma 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:

$$E_{SLH} = \sum_l (I_l - \mathbf{A}J_l - \mathbf{t})^2 \tag{7}$$

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.

The values of the affine transform $\mathbf{A}, \mathbf{t}$ are not directly meaningful except in a broad sense. This is because the distortion between different copies of the handwritten words is not affine - all we are doing is to approximately model it as an affine. In addition when the words are dissimilar, the affine parameters are not meaningful at all. One disadvantage of computing the additional 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 sigma are low). The computation of the SVD can, therefore, be speeded up by utilizing methods which compute the SVD of a sparse matrix quickly [4]. 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 Using the SLH Algorithm

Experiments were performed using the Senior document. Since the current version of the SLH algorithm is slow, the initial matches were pruned using the EDM algorithm and then the SLH algorithm run on the prunedd subset.

To account for the large variations in the Hudson papers, the area threshold α was fixed at 1.3 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 (5). The succesive columns of the table, tabulate the Token Number, the transcription of the word, the area of the word image, the match error due to the EDM method, 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 (1) 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".

Token Number	Word	Area	E_{EDM}	Corres.no	E_{SLH}	A		T
105	Lloyd	1368	0.000000	233	0.00	1.00	0.00	0.00
						0.00	1.00	0.00
197	Lloyd	1400	0.194	199	1.302	0.96	-0.04	1.58
						0.01	1.04	0.14
70	Lloyd	1224	0.174	176	1.356	0.94	0.09	-1.02
						0.03	0.92	-1.38
165	Lloyd	1230	0.175	189	1.631	1.03	0.05	-0.43
						-0.01	0.87	-2.60
239	Lloyd	1320	0.197	203	1.795	0.99	-0.05	1.44
						0.03	1.07	2.21
157	lawyer	1518	0.236	185	3.393	0.96	-0.03	1.89
						0.05	1.11	0.03
240	Selwyn	1564	0.307	188	3.673	0.94	0.06	-4.23
						0.05	1.05	-0.75
91	thought	1178	0.208	181	3.973	0.97	0.03	2.33
						-0.01	1.08	2.91

Table 5: Rankings and Match Errors for template "Lloyd" Using SLH Algorithm.

The template "the" was matched. Table (6) and Figure (8) show the rankings. All the instances of "the" are ranked in the correct order (which was not true of the EDM algorithm - see Table (2) and they all have match errors less than 1 while the invalid words have a match error greater than 1.

The word "Minister" was then matched. Table (7). Again, the correct ranking is established. The upper and lower case "m" does not seem to cause a problem. The discrimination is poorer i.e. the match error of "minister" is close to that of the next word. This is not completely unexpected. The word "minister" will have a somewhat larger error because its first letter does not correspond to the template.

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

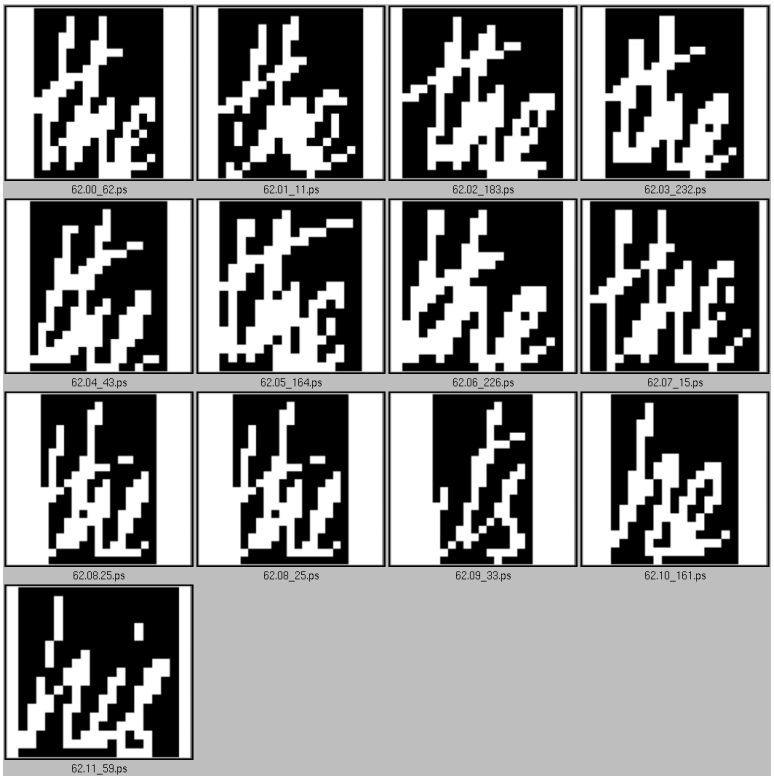

Figure 8: Rankings for template "the" (the rankings are ordered from left to right).

shows wide variations in size - for example, the area of the word "to" varies by as much as a 100% ! However, this large a variation is not expected to occur and is not seen when the words are larger.

The difficulty with small words is demonstrated in the following example. [Figure (9) and Table (8)] (? indicates uncertain transcription). The template is "to". The technique recovers 9 out of 13 "to"'s. Two of the 13 "to"'s have an area a 100% larger than that of the template and are therefore pruned by the area threshold (using a larger area threshold would increase the number of false matches). Two other "to's" have a large affine distortion and are eliminated by the affine pruning technique. (The figure shows some of the large variations in the "to's"). Fortunately small words such as "to" are not useful for indexing. Such large distortions are not seen with the larger words.

In any case, the performance of this algorithm is significantly better than the EDM

Token Number	Word	Area	E_{EDM}	Corres.no	E_{SLH}	A		T
62	the	336	0.000	116	0.000	1.00	-0.00	-0.00
						-0.00	1.00	-0.00
11	the	378	0.223	95	0.456	1.01	-0.02	1.08
						0.05	0.97	-0.05
183	the	380	0.149	105	0.548	0.95	0.00	-1.74
						0.03	1.02	-0.28
232	the	396	0.170	105	0.596	0.92	0.00	-0.32
						0.02	0.98	-0.72
216	the	418	0.289	108	0.650	0.92	-0.03	-0.39
						-0.06	0.96	-0.06
43	the	391	0.259	96	0.659	0.99	-0.04	-0.26
						-0.03	0.95	-0.45
164	the	304	0.143	98	0.683	1.02	0.02	0.19
						-0.01	1.03	0.17
226	the	380	0.256	105	0.718	0.88	0.02	-0.13
						0.00	1.03	0.12
15	the	400	0.280	100	0.760	0.90	0.07	-1.25
						-0.06	1.01	0.37
25	the	330	0.193	93	0.834	0.98	0.03	-0.83
						0.06	0.98	-0.04
33	its	286	0.265	69	1.173	0.88	0.11	-0.47
						0.14	1.01	-0.68
161	he	300	0.271	85	1.233	0.89	0.04	0.23
						0.02	0.89	0.03
59	his	357	0.312	72	1.323	1.02	0.01	1.09
						0.01	0.90	0.07

Table 6: Rankings and Match Errors for template "the" Using SLH Algorithm.

Token Number	Word	Area	E_{EDM}	Corres.no	E_{SLH}	A		T
113	Minister	1134	0.000	254	0.000	1.00	0.00	0.00
						0.00	1.0	0.00
147	minister	1078	0.210	189	1.705	1.04	0.00	1.94
						-0.04	1.04	2.62
156	derived	1334	0.352	215	1.93	1.00	0.04	-4.94
						-0.02	0.96	1.09
257	number	1248	0.342	219	1.958	1.02	0.020	1.54
						-0.05	0.99	1.83
176	number	1104	0.285	197	2.034	-0.03	1.04	1.27
						1.05	0.046	1.33

Table 7: Rankings and Match Errors for template "Minister" Using SLH Algorithm.

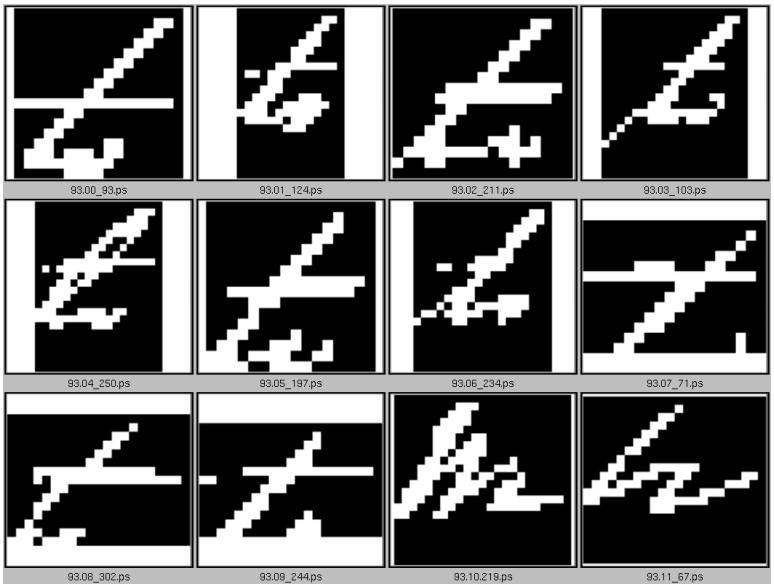

Figure 9: Rankings for template "to" for the SLH algorithm (the rankings are ordered from left to right and from top to bottom.

algorithm.

Performance on larger templates like "they" is good as shown in Figure (10) and Table (9). 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 (11) and Table (10). 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 (7)).

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.

Token Number	Word	Area	E_{EDM}	Corres.no	E_{SLH}	A		T
93	to	289	0.000	63	0.000	1.00	0.00	0.00
						0.00	1.00	0.00
124	to	224	0.356	52	0.567	1.03	0.13	0.05
						-0.04	1.03	0.04
211	to	255	0.121	49	0.981	0.92	0.14	-0.96
						0.00	1.05	0.17
103	to	342	0.263	48	0.986	0.99	-0.07	3.61
						0.08	0.88	2.21
250	to	324	0.457	59	1.005	0.89	0.04	0.04
						0.05	0.88	-0.62
197	to	272	0.131	55	1.034	0.96	0.06	0.43
						-0.00	0.98	-0.39
234	to	288	0.280	49	1.238	0.90	0.08	-1.35
						0.02	1.03	0.16
71	to	288	0.280	49	1.560	0.87	0.16	-1.63
						0.11	0.85	-1.08
302	to	336	0.176	50	1.602	0.946	-0.04	0.27
						0.13	0.88	-0.51
244	to	315	0.170	53	1.708	0.85	0.14	-0.84
						-0.01	0.97	0.01
219	We?	368	0.526	52	2.710	0.81	0.23	-0.75
						0.08	0.97	-0.31
67	be	308	0.308	42	2.828	1.19	-0.22	0.80
						-0.04	0.87	-0.13

Table 8: Rankings and Match Errors for template "to" Using SLH Algorithm.

Sorting list of matching words for 'they' (Token Number = 1)								
Token Number	Word	Area	E_{EDM}	Corres.no	E_{SLH}	A		T
1	they	899	0.000	108	0.000	1.00	0.00	0.00
						0.00	1.00	0.00
43	they	891	0.145	97	0.636446	0.92	0.05	-0.93
						0.05	1.01	1.62
156	only	775	0.182	85	3.172	0.89	-0.22	1.53
						0.03	1.20	-0.38
191	this?	696	0.222	83	8.466	0.97	-0.15	1.40
					-0.05	1.14	7.23	

Table 9: Rankings and Match Errors for template "they" Using SLH Algorithm.

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

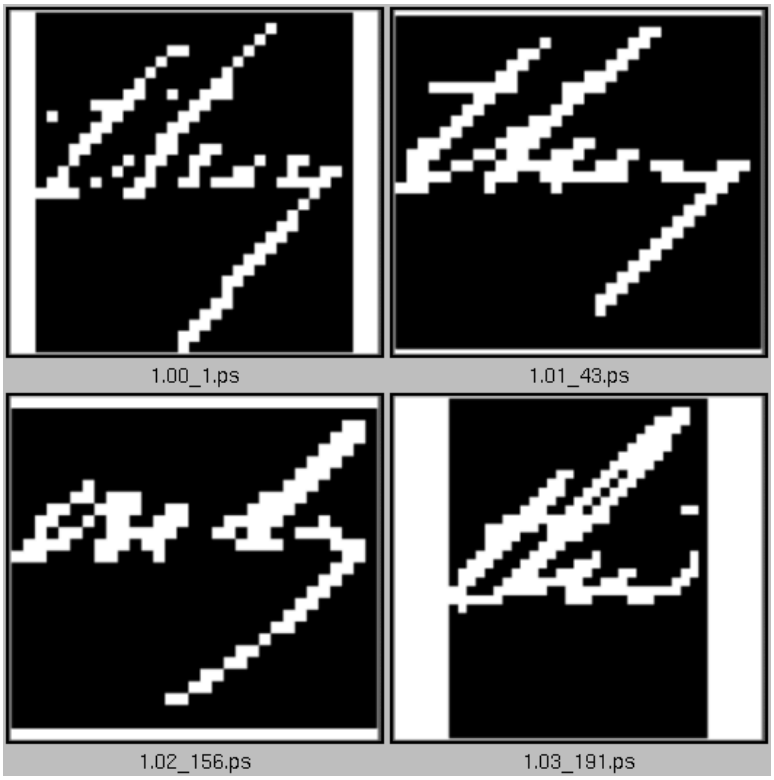

Figure 10: Rankings for template "they" for the SLH algorithm (the rankings are ordered from left to right and from top to bottom.

Token Number	Word	Area	E_{EDM}	Corres.no	E_{SLH}	A		T
280	Standard	1530	0.000	251	0.000	1.00	0.00	0.00
						0.00	1.00	0.00
167	Standard	1440	0.228	183	4.36	1.03	0.10	5.07
						-0.01	0.94	0.33
56	subscribers	1900	0.224	196	7.816	0.99	0.20	1.27
						0.00	0.94	-0.38
283	substance	1479	0.223	183	39.185	0.92	0.12	-1.39
						-0.02	0.82	1.02

Table 10: Rankings and Match Errors for template "Standard" Using SLH Algorithm.

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

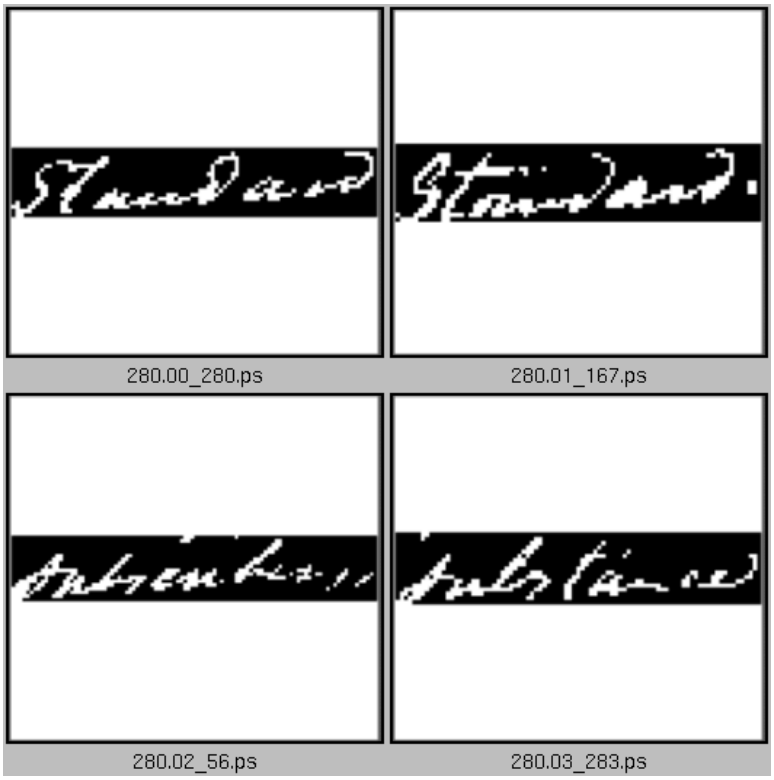

Figure 11: Rankings for template "Standard" for the SLH algorithm (the rankings are ordered from left to right and from top to bottom.

words.

10 Acknowledgements

The original idea of using word spotting to index handwritten documents was suggested by Bruce Croft. The page from the Hudson collection was scanned in and provided by Gail Giroux of the University of Massachusetts W. B. DuBois library. The other scanned document (the Senior page) was obtained through the DIMUND document server. The Senior page can be obtained from:

http://documents.cfar.umd.edu/resources/database/handwriting.database.html

and was scanned by Andrew Senior. Jonathan Lim and Bob Heller provided help with systems software.

References

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

[2] M. Bokser. Omnidocument technologies. *Proceedings IEEE*, 80(7):1066–1078, 1992.

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

[4] Michael Berry et al. Svdpackc user's guide. Technical Report CS-93-194, Dept. of Computer Science, Univ. of Tennessee,Knoxville, TN, 1993.

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

[6] Paul Filiski and Jonathan J. Hull. Keyword selection from word recognition results using definitional overlap. In *Third Annual Symposium on Document Analysis and Information Retrieval, UNLV, Las Vegas*, pages 151–160, 1994.

[7] K. Sparck Jones G. J. F. Jones, J. T. Foote and S. J. Young. Video mail retrieval: The effect of word spotting accuracy on precision. In *International Conference on Acoustics, Speech and Signal Processing*, volume 1, pages 309–316, 1995.

[8] Siamak Khoubyari and Jonathan J.Hull. Keyword location in noise document image. In *Second Annual Symposium on Document Analysis and Information Retrieval, UNLV, Las Vegas*, pages 217–231, 1993.

[9] R. Manmatha. A framework for recovering affine transforms using points, lines or image brightnesses. In *Proc. Computer Vision and Pattern Recognition Conference*, pages 141–146, 1994.

[10] R. Manmatha. Measuring the affine transform using gaussian filters. In *Proc. 3rd European Conference on Computer Vision*, pages 159–164, 1994.

[11] C. P. Lu S. Gold, A. Rangarjan, S. Pappu, and E. Mjolsness. New algorithms for 2d and 3d point matching: Pose estimation and correspondence. In *To appear in Pattern Recognition*, pages –.

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

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

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

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

[16] R. Szeliski and J. Coughlan. Hierarchical spline-based image registration. In *Proc. Computer Vision and Pattern Recognition Conference*, pages 194–201, 1994.

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

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