Indexing for a Digital Library of George Washington's Manuscripts: A Study of Word Matching Techniques

T. M. Rath, S. Kane, A. Lehman, E. Partridge and R. Manmatha[*]
Center for Intelligent Information Retrieval
Computer Science Department
University of Massachusetts
Amherst, MA 01003
[trath,manmatha]@cs.umass.edu

ABSTRACT

In a multimedia world, one would like electronic access to all kinds of information. But a lot of important information still only exists on paper and it is a challenge to efficiently access or navigate this information even if it is scanned in. The previously proposed "word spotting" idea is an approach for accessing and navigating a collection of handwritten documents available as images using an index automatically generated by matching words as pictures. The most difficult task in solving this problem is the matching of word images. The quality of the aged documents and the variations in handwriting make this a challenging problem. Here we present a number of word matching techniques along with new normalization methods that are crucial for their success. Efficient pruning techniques, which quickly reduce the set of possible matches for a given word, are also discussed. Our results show that the best of the discussed matching algorithms achieves an average precision of 73% for documents of reasonable quality.

Keywords

Multimedia content analysis, audio/image/video processing, content-based multimedia retrieval

1. INTRODUCTION

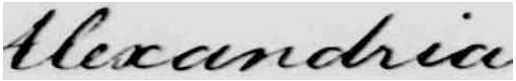
There exist numerous collections of handwritten historical manuscripts which could serve as a valuable resource to historians and others who wish to read them. Examples include the early Presidential papers at the Library of Congress and the collected works of W.E.B. Du Bois - the African American civil rights leader - at the University of Massachusetts Library. Unfortunately, many such collections, although available through public libraries or the internet, are only loosely organized, i.e. without any index information or just manually annotated. Since manual document transcription - which is the current solution to this problem - is a very expensive task, an automatic way to build document indexes for easy navigation is preferred. Previously the "word spotting" idea has been proposed to solve this problem [8, 7, 6]. In word spotting, an index is automatically generated for a collection of handwritten documents, which are only available as images by matching words as pictures. After segmenting all pages in a given corpus into pictures of words, image matching techniques are used to cluster all occurrences of a word throughout the collection. Then, the largest clusters (most frequent words in the collection) form the index terms. Here we compare a number of algorithms for matching word images and we present the extensive normalization techniques that make these algorithms work.

A preliminary study in [8, 7, 6] identified two algorithms for matching images, Euclidean Distance Matching (EDM) - which compensates for translations - and the SLH algorithm (proposed by Scott and Longuet-Higgins) [12] which handles affine transforms. However, techniques which work on carefully scanned images fail on the George Washington collection where the noise and the range of distortions is larger. The previously discussed EDM technique only handled translations. Here, we propose new normalization schemes to handle these distortions and show that the resulting affine-normalized EDM technique performs much better than four other image matching techniques including SLH, SSD (correlation) and the shape context matching algorithm.

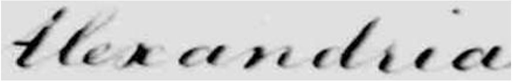
Word spotting is a challenging problem, because of the quality of the data that has to be dealt with. The most prominent factors are the following:

- Age of the documents: historic manuscripts suffer from fading ink (for example see Figures 1(b) or 7), ink that shines through pages and non-uniform backgrounds.
- Bad scanning quality: our collection of George Washington's letters was scanned from microfilm (possibly

[*]This material is based on work supported in part by the Center for Intelligent Information Retrieval and in part by the National Science Foundation under grant numbers EIA-9820309 and IIS-9909073 . Any opinions, findings and conclusions or recommendations expressed in this material are the author(s) and do not necessarily reflect those of the sponsor(s).



(a) Scanned from a well preserved document



(b) From document with slightly faded ink

Figure 1: Examples of "good" quality images.

to protect the originals), which causes considerable degradation in the quality of the scanned images.

- Compression artifacts: the test collection we received was scanned at 300dpi and saved in JPEG format with a high compression ratio, in order to reduce the storage needs (see Figure 2 for an example of the magnitude of the JPEG artifacts).

Additionally, since most historical documents were written by hand, the words are difficult to recognize. This is a result of the variations in handwriting, which affect the appearance of a word at both the word- and the character-level: throughout a document by the same author, words vary in scale, length, slant, the way their characters are written and other factors. For these reasons, optical character recognition (OCR) approaches, which perform well on machine printed documents against clean backgrounds, do not work well in the handwriting domain. Our research indicates that one of the best performing algorithms on the MNIST set of handwritten digits [1] does not perform well on our collection of documents.

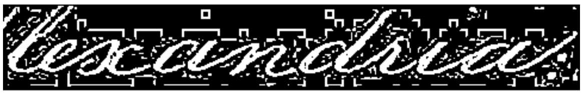


Figure 2: Edge image of an instance of the word *Alexandria*: JPEG compression artifacts result in "dirt" and phantom edges (e.g. horizontal edges).

The previously presented word spotting idea provides an alternative solution for indexing large handwritten texts written by a single person (or a few people). The goal is to automatically create an index similar to the one at the back of a book: the index would contain a list of words, each word associated with a set of links, one link for each occurrence of that word in the collection. With these indexes, libraries can put their collections of manuscripts on line, easily accessible to anyone who needed to do research on them. The word spotting technique accomplishes this by creating classes of word images, where each class consists of multiple instances of the same word. A human could then provide an ASCII equivalent for the classes. Such a system would of course be applicable more widely, not just to the George Washington collection. The matching techniques could also be useful for other applications - for example in PDAs (address/appointment retrieval or similar tasks). Previous work has discussed the feasibility of word spotting [8, 7, 6] based on a small sample of images which were scanned carefully.

The word spotting algorithm is outlined as follows:

1. A scanned greylevel image of the document is obtained.
2. The page is divided into units (words) which can be compared singly. Our approach uses the algorithm presented in [9].
3. Each word image is taken as a template and compared against all other word images[1].
4. Words are grouped into clusters/classes based on their similarity.
5. A human provides an ASCII equivalent of the most frequently occuring words (large clusters)[2]. Another way to present the index in a user-interface would be to display the members of a cluster as clickable thumbnails which point to the respective document locations.[3]

There are a number of advantages of word spotting over conventional OCR. Most OCR systems recognize a character at a time - with some language modeling constraints on sequences of valid character strings. Using the entire word image often provides better context and will make the process easier (for example individual characters written by people are often unrecognizable). In fact, there is empirical evidence that humans employ a similar "holistic" word matching technique for reading [5]. Additionally, word spotting avoids the automatic *recognition* process entirely and focuses on matching word images. Recognition is often difficult to do and requires extensive training. Furthermore, we believe that the word matching process is easier to solve when the documents are written by one person (or a few).

Here we report the extensive image normalization methods we used to account for a number of common variations in handwriting. These techniques are crucial for the successful matching of word images. We also discuss five algorithms for matching words and compare the performance of four of them on a set of 10 documents. Unless the quality of the scanned images is very poor, our approach can achieve an average precision of 73%. An online demonstration which provides an index for a small set of George Washington's pages has been implemented.

2. BACKGROUND

The traditional approach to indexing documents involves first converting them to ASCII and then using a text based

[1] For a large collection this is obviously prohibitive; we reduce the number of possible matches using pruning techniques.

[2] The largest clusters will likely be stop-words (for example *the* and *of*), which are not useful for indexing. Such words should be omitted.

[3] The demonstration at http://ciir.cs.umass.edu/research/wordspotting uses this interface.

retrieval engine [13, 11]. 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]. This approach has been highly successful with standard 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 also varies). The successes of handwritten character recognition have occured in domains where the lexicon is small and/or the words are highly constrained. For example, the US post office uses OCR for reading addresses on envelopes [10] (many of which are machine printed) by taking advantage of the highly structured nature of such addresses and the fact that different fields mutually constrain each other (for example, the street name must be consistent with the town name and zip). These machines also have databases of all the streets in every city in the country. On the other hand, the current state of the technology is such that OCR is unlikely to be successful in the foreseeable future for handwritten archival documents.

An approach similar to ours has been used to recognize words in documents which use machine fonts [4]. 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 noise-free version of the word. This word is then recognized using an OCR. Recognition rates are much higher than when the OCR is used directly [4].

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 variations in ink, scaling of the words with respect to each other, small changes in orientation, and changes in the lengths of descenders and ascenders. Figure 9 shows a number of identical words from the same document, written by the same writer but nevertheless with a fair amount of variation. It may thus be necessary to account for these variations.

Word spotting requires that the text be previously segmented into words. This is already by itself a non-trivial problem and for the work done here, the scale space algorithm discussed in [9] is used to segment the document into words. Figure 3 shows a page of Washington's manuscripts which has been segmented using this algorithm. Note that although this page is of good quality, variations in the ink and dark borders (artifacts of the microfilming/scanning process) are present. The segmentation technique has proven to work well (77-96 percent accuracy on Washington's manuscripts), and has been used to segment our collection of 6000 pages of George Washington's scanned manuscripts.

The output of the segmentation process is used as the input to the word spotting algorithm. Here the segmented words are grouped into clusters/classes. This is done by comparing one segment, the template, against the others, and ranking matches by similarity scores. The top matches then form the class.

Threshold α	Recall	Precision	% Returned
4	1.000	0.012	88.03
3	1.000	0.014	77.57
2	0.982	0.019	53.67
1.5	0.960	0.032	32.47
1.2	0.851	0.060	15.83
1.15	0.739	0.068	11.97

Table 1: Pruning results for *similar length* constraint (equation 1).

Because all of the matching algorithms are very resource-intensive, it is desirable to prune the images which are likely not to be matches. This cuts down on the number of comparisons to be done by the matching algorithms. Section 3 outlines the techniques we used and their performance. Images which are not discarded by the pruning process are handed to the matching algorithm (see section 5). Our results show that even for good quality scans extensive filtering, normalizing and transforming of these images is necessary for producing good matching results.

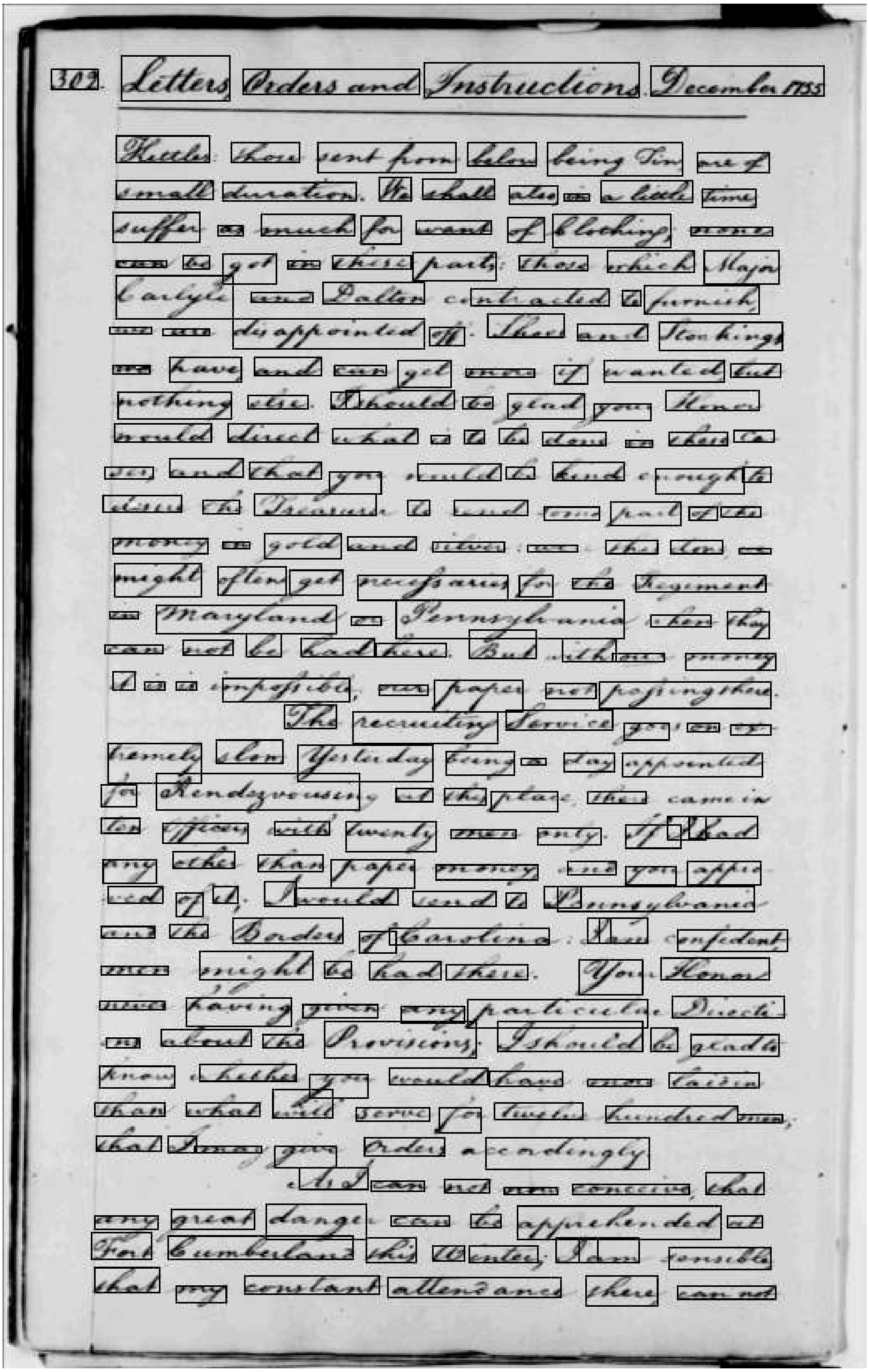
3. PRUNING

Pruning is the process of determining which images are not likely to be matches before running the scoring algorithm by using easy-to-measure properties of the image. Since there are $O(n^2)$ image pairs for n word-images, and due to the resource-intensive nature of the matching algorithms, it is desirable to prune the search space as much as possible, while being careful not to dismiss valid matches. Several types of pruning criteria were tested: image length, image area, aspect ratio, number of ascenders and descenders, and number of black-to-white transitions in a cross-section of the image.

The first pruning method used the assumption that words from the same class will have similar lengths in terms of the widths of their bounding boxes. Thus, the following constraint was used to determine whether a given word is a possible match for a template:

$$\frac{1}{\alpha} \leq \frac{templatelength}{imagelength} \leq \alpha \qquad (1)$$

Table 1 shows the results obtained with the *similar length* pruning constraint: *Recall* is the number of relevant words that were left after pruning at a given threshold value divided by the total number of relevant words. (a word image is relevant to the query if it has the same ASCII translation.) *Precision* is the percentage of relevant words in the returned words. *% Returned* is the average percentage of word images left after pruning at the corresponding threshold (total word images before pruning: 2381). Ideally one would like to maximize precision without lowering recall too much. Due to the variations in handwriting, words belonging to the same category can often have quite different lengths, so this pruning method is likely to discard positive matches. For



302. Letters Orders and Instructions December 1755

Kettles: those sent from below being Tin are of small duration. We shall also in a little time suffer as much for want of Cloathing; none can be got in these parts: those which Major Carlyle and Dalton contracted to furnish us are disappointed of. Shoes and Stockings we have and can get more if wanted but nothing else. I should be glad your Honor would direct what is to be done in these cases, and that you would be kind enough to desire the Treasurer to send some part of the money in gold and silver: as we then might often get necessaries for the Regiment in Maryland or Pennsylvania when they can not be had here. But without money it is impossible, our paper not passing there.

The recruiting Service goes on extremely slow. Yesterday being the day appointed for Rendezvousing at this place there came in ten officers with twenty men only. If I had any other than paper money and you approved of it, I would send to Pennsylvania and the Borders of Carolina: I am confident men might be had there. Your Honor never having given any particular Direction about the Provisions, I should be glad to know whether you would have more laid in than what will serve for twelve hundred men, that I may give Orders accordingly.

As I can not now conceive that any great danger can be apprehended at Fort Cumberland this Winter, I am sensible that my constant attendance there can not

Figure 3: Page from George Washington's manuscripts after the segmentation process. This is a low resolution version of a larger image. Note that there is some variation in ink even in this good quality image.

Threshold β	Recall	Precision	% Returned
4	0.996	0.014	71.06
3	0.994	0.017	58.97
2	0.939	0.024	39.31
1.5	0.863	0.037	23.27
1.2	0.644	0.054	10.84
1.15	0.473	0.052	7.98

Table 2: Pruning results for *similar area* constraint (equation 2).

Threshold δ	Recall	Precision	% Returned
4	1.000	0.012	96.56
3	1.000	0.012	90.72
2	1.000	0.015	72.03
1.5	0.951	0.025	48.64
1.2	0.737	0.039	24.36
1.15	0.661	0.044	18.10

Table 3: Pruning results for *similar aspect ratio* constraint (equation 3).

this reason, two other pruning constraints, one based on the number of pixels forming a word ("area") and one on the aspect ratios of the bounding boxes, were tried:

$$\frac{1}{\beta} \leq \frac{templatearea}{imagearea} \leq \beta \qquad (2)$$

and

$$\frac{1}{\delta} \leq \frac{templateaspect}{imageaspect} \leq \delta \qquad (3)$$

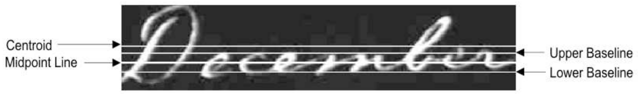


Figure 4: Visualization of upper and lower baseline.

The first constraint asserts that words in a cluster have similar areas, the second one asserts that the bounding boxes of such words have similar aspect ratios. The first assumption obviously does not hold true if words in the same cluster occur at different scales (words in the heading versus in the body of a text), yet it has good pruning capabilities without lowering the recall too much (the < 1 recall values even for high thresholds are most probably due to words from the same cluster, which occur at different scales). The second constraint seems more sound, but since it makes a more moderate assumption its pruning capability is not as strong as the first constraint. These two techniques together provided a better estimate than length alone for the pruning step (see tables 2 and 3), but still did not achieve the desired high level of pruning while not removing valid words from consideration.

To improve upon these techniques, we examined the number of ascenders and descenders. For example, the word "example" has one ascender, the "l", and one descender, the "p", while the word "one" has neither ascenders nor descenders. To find the number of ascenders, first the upper and lower baselines of the main segment of the word were found. To compute the baselines of the image, the vertical projection of the black and white image is computed. The upper and lower baselines are taken as the two points where the changes in this projection are greatest. Figure 4 shows the resulting baselines for an example image. Each connected component above the upper baseline was then counted as an ascender. Each connected component below the lower baseline was counted as a descender. (To avoid counting false ascenders and descenders, the components had to be larger than 10 pixels to be counted.)

While counting ascenders in this way was not very successful, we were able to accurately determine the number of descenders in a great number of cases. Furthermore, constraining the number of descenders in matching candidates to the same number as in the template turned out to be a good pruning technique. As a result, a combination of examining area (with $\beta = 2$), aspect ratio (with $\delta = 1.5$), and number of descenders (only words with the same number of descenders are allowed) were used for the tests in the subsequent steps. With these parameters, 87% of the possible comparisons were avoided, and 94% of the relevant words were retained. Note that this combined reduction is much more efficient than that of its components (see Tables 2 and 3) and that lowering the area and aspect ratio thresholds any further would drastically lower the recall.

4. NORMALIZATION

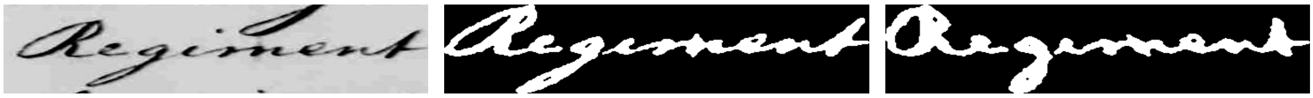
Originally (see [8, 7, 6]), images that were not discarded by the pruning step were passed on directly to the matching algorithm. We have found that applying a number of normalization techniques before words are actually matched can improve the accuracy of the match scores . With the exception of the SLH and SC algorithms, all of the matching algorithms presented in section 5 can only compensate for translations of a match candidate relative to the template. Since handwritten words in a collection are frequently scaled[4], skewed[4] and rotated with respect to each other, the success of a matching algorithm that can only compensate for rigid translations is limited. For this reason we apply a number of normalization techniques to match candidates. The algorithms presented here compensate for different scale (both directions) and skew (horizontal, i.e. word slant) in a template and a match candidate.

4.1 Binarization and Artifact Removal

Since all of our matching algorithms work on black-and-white images, the template and match candidate are both binarized using a threshold which was picked by inspection. A more robust thresholding scheme, such as automatically selecting a threshold from a grey level histogram, may be more robust for different collections. Figure 5(b) shows an example output of the binarization procedure (Figure 5(a) contains the original image).

A result of using bounding boxes as stencils to extract words from document images are dangling descenders and ascen-

[4] Scaling and skewing frequently occurs independently in horizontal and vertical direction.



(a) Original greyscale image (b) Binarized, artifacts removed (c) Slant-corrected

Figure 5: Different processing/normalization steps on a greyscale image of the word *Regiment*.

ders from other lines above and below which appear in the segmented image. These artifacts negatively affect the matching algorithms and thus need to be removed while preservin the actual word. Our solution to this problem was to draw a white box between the upper and lower baselines (see Figure 4) and compute the 8-connected components of the resulting image. All connected components but the largest one were removed from the original binarized image. Figure 5(b) shows a typical result obtained with a word-image with a dangling descender (original in Figure 5(a)).[5]

4.2 Scale Normalization

The heights of the template and the candidate image were normalized by scaling them so that the distance between the lower baseline and the top for both images were the same.

Three normalization techniques have been tried for the width normalization:

1. Normalization of the image width (same length of bounding box): while better than no normalization, this caused problems in cases where the segmentation routine missed the first or last letter of a word.
2. Normalization of the distance between two centroids: two centroids were calculated for each image - one for the left, and one for the right half of the image. This method performed best among all of the width normalization techniques.

4.3 Slant Correction

Most of the words in the George Washington collection are slanted, but the slant angle is not consistent throughout all documents. Therefore, we deslanted all words prior to matching them. To do so, we estimated the slant angle of a word from the average angle of the left edge of a word's first letter. These angle estimates were collected between the upper and lower baseline of the word. With this angle information a shear transformation was performed on the image, in order to produce a slant-corrected version of it. A typical result can be seen in Figure 5(c). Curls or tails on the first letter of a word can interfere with the angle estimation, which could be overcome by determining the slant angle at several points in a word image, not just from the first letter. Slight rotations or shears in the vertical direction were not corrected so far.

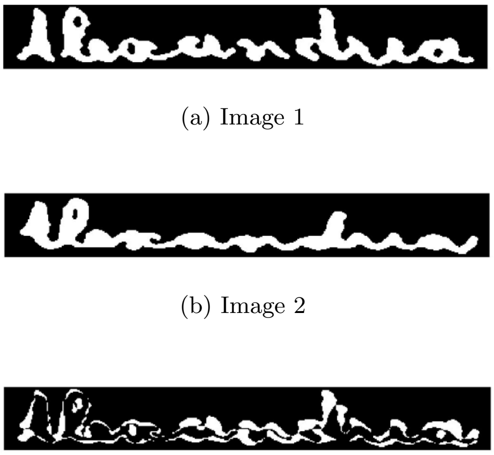
4.4 Image Alignment

Before running matching algorithms on the normalized images, we roughly compensate for their relative translations by aligning their lower baselines (vertical displacement) and leftmost pixels (horizontal displacement). The matching algorithms in the next section will refine this alignment.

5. MATCHING ALGORITHMS

Five different matching algorithms were tried, ranging from a very simple XOR computation to a much more sophisticated calculation of thin-plate spline transformations. All the matching algorithms took one image, designated it as a template, and took each of the other images left after the pruning step and compared them to the template to generate a match error. Then the images were ranked according to how well they matched the template (low match errors first).

5.1 XOR



(a) Image 1

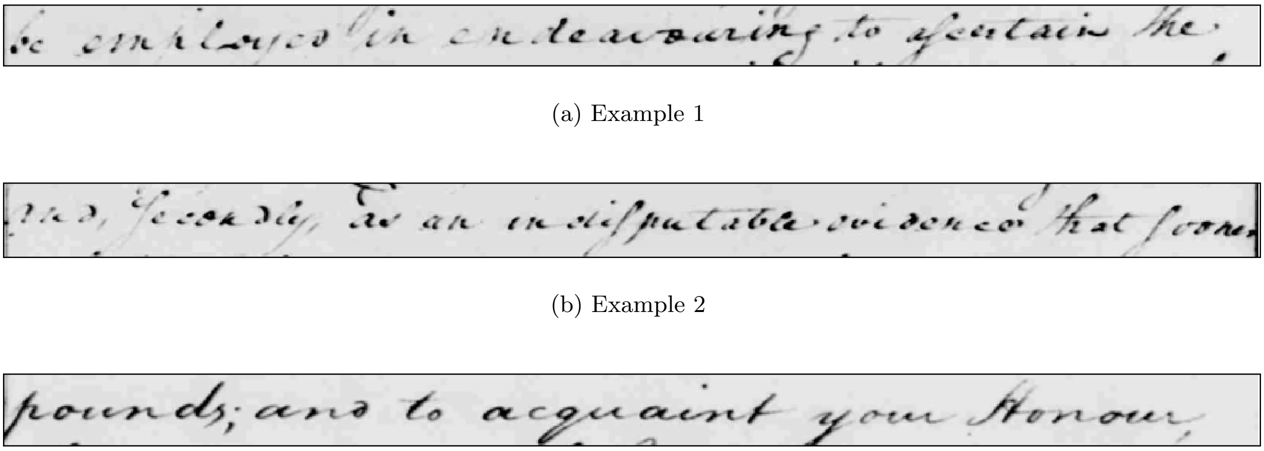
(b) Image 2

(c) XOR-Image of Image 1 and 2

Figure 6: XOR-image for two occurrences of the word *Alexandria*.

The simplest of the implemented matching techniques calculated a XOR-image, which contains white pixels in locations where the two original images differ (see Figure 6 for an example). The number of white pixels is counted and divided by the total number of pixels in the XOR-image to form the match error. This process is repeated for a number of small translations of one of the images in order to compensate for imperfect image alignments. The minium of the obtained match errors is used as the final error measure. This technique was very fast, but did not produce accurate match errors. The EDM algorithm, which has a slightly more sophisticated approach, performs significantly better.

[5] Note that dots, such as that of the "i" in this example, are removed as well.


be employed in endeavouring to ascertain the
secondly, as an indisputable evidence that form
pounds; and to acquaint your Honour

(a) Example 1

(b) Example 2

(c) Example 3

Figure 7: Representative lines extracted from data sets 1, 2 and 3, respectively. Note the uniformity in ink in the third line as compared with the other two. The difference in quality is more pronounced when the images are displayed on a screen and also magnified when the images are binarized.

5.2 EDM

While the XOR matching algorithm provides a natural way of assessing the similarity of two images, it has a flaw: each mismatched pixel contributes the same amount to the error measure. But isolated white pixels in the XOR image are more likely to be the result of noise or small misalignments in the compared images, whereas blobs of pixels are indicators of a more serious mismatch, possibly due to non-negligible differences in the two images. In order to account for this observation, we applied the Euclidean Distance Mapping (EDM) algorithm to the XOR images. Using this technique, each white pixel in a XOR image contributes an amount to the error measure, which is proportional to the distance between that pixel and the nearest black pixel. In our implementation we used the low-error EDM variant *Sequential Octagonal Distance Mapping* [3].

Similar to the XOR error measure, the EDM measure needs to be normalized in order to remove bias introduced by different image and template sizes. This was done by dividing the EDM error by the area of the XOR image. We calculated the EDM error measure for a number of small translations (of the template relative to the other image) in horizontal and vertical direction and used the minimum of the observed error measures. The resulting word rankings were significantly better than those obtained with the XOR method.

5.3 SSD

This algorithm computes the correlation between an image and a template using the Sum of Squared Differences (SSD) technique. The smaller of the two images to be compared was used as the template. For a number of horizontal and vertical displacements x and y, the normalized SSD value was computed using

$$SSD(x,y) = \frac{\sum_i \sum_j [I_{(x+i),(y+j)} - T_{i,j}]^2}{\sqrt{\sum_i \sum_j (T_{i,j})^2}\sqrt{\sum_i \sum_j (I_{(x+i),(y+j)})^2}} \qquad (4)$$

where T is the template image and I is the image to be tested against (subscripts indicate the coordinates of the pixel to be evaluated). The normalization factors in the denominator remove the SSD-measure's bias towards matches with smaller templates (otherwise smaller templates would yield smaller SSD values). The results obtained with this technique were roughly comparable to those of the XOR-image method, but required more processing time.

5.4 SLH

All of the presented algorithms so far compare images pixel-by-pixel. The following algorithm, which was also tested on our collection, takes a different approach: Scott and Longuet-Higgins [12] (SLH) suggested an algorithm which explicitly accounts for affine transforms between two sets of sample points. The correspondences between the sets are recovered and used to calculate an affine transform which best maps one set of samples onto the other set. The estimated affine transform is of the form

$$\mathbf{r}' = \mathbf{A}\mathbf{r} + \mathbf{t} \qquad (5)$$

where $\mathbf{A}$ is a 2 x 2 matrix and $\mathbf{t}$ is a 2-dimensional vector. This type of transform can describe scaling, shear, translation in both directions, and rotation between two sets of points. The reader is referred to [12, 6] for details on how sample point correspondences are computed using this algorithm.

When comparing two images, the first step is to reduce the number of pixels that form the words in the images. We tried

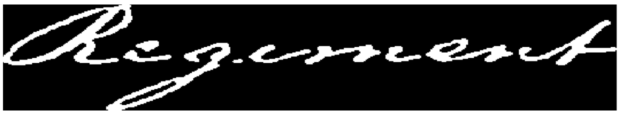


Figure 8: Laplacian of Gaussian filter applied to the image in Figure 5(a).

both Gaussian derivative edge-filters to extract word contours and filtering with Laplacian of Gaussian filters. The latter produced the best results (see Figure 5.4 for an example output) and was used for our experiments in section 7.

The filtered images are used to generate two sets of sample points I and J. These are simply all non-zero pixels in the images. The sample points are then passed to the SLH algorithm which recovers the correspondences between the sets I and J and calculates an affine transform $(\mathbf{A}, \mathbf{t})$ using least mean squares. The error (ESLH) between the affine transformed set of points J' and I may be computed as follows:

$$ESLH = \sum_v (I_v - \mathbf{A}J'_v - \mathbf{t})^2 \tag{6}$$

where the subscript v indicates a single point in I or J'.

Prior research using this method has provided encouraging results [6]. However, while SLH had previously been used on a set of images pruned with EDM (without normalization!), here the SLH method was used as a separate matching algorithm. Its performance could not reach that of the EDM algorithm and the time needed by the algorithm was far greater than was needed for the XOR, EDM and SSD algorithms (see Table 7).

6. SHAPE CONTEXT

The matching algorithm based on the shape context descriptor (SC) currently achieves the best results in the domain of handwritten digit recognition [1]. This algorithm matches images by recovering correspondences between pairs of points on the edge contours of the two images. The correspondences are used to find a transformation which maps one image onto the other. Thus it is possible to use traditional image matching techniques for determining the similarity between two given images.

The similarity between pairs of points is measured using the shape context descriptor, which describes the context of a given point relative to the rest of the shape of which it is a part: lines originating in the given point, equally dividing the full 360° range and concentric circles around that point with radiuses on a logarithmic scale mark the bins of a histogram. Each histogram bin counts the number of points on the shape contour which fall into it. The extent of the shape context histograms (largest circle marking a bin) is chosen so that it roughly covers the whole shape, thus describing the "greater" context of a point within the shape.

The point correspondences are recovered by finding an assignment between pairs of points which minimize a global cost that is based on the similarity of the shape contexts of all assigned points.

While we have had success in applying this algorithm to other types of shapes, our tests with the freely available demonstration code were not successful for matching word images. An inspection on a number of similar and dissimilar images showed no correlation between the similarity of tested image pairs and the shape context cost measures. As a result, the SC algorithm was not used in the tests described in the next section.

The algorithm seemed to have problems finding the correct correspondences, possibly due to the coarse similarity of words: contour points are almost always densely distributed between the upper and lower baseline with few ascenders and descenders. When points are picked from roughly the same region in an image, they will have very similar shape contexts and thus are not distinguishable. This can result in falsely identified correspondences between neighboring characters. In the case of shapes with less variation this problem is not as pronounced. Another drawback of the algorithm is its high computational demand, which results from the extraction of correspondences from a cost matrix which is of size $n \times n$ for n contour sample points. While this is not an issue for contours which can be approximated with 100 sample points (for example digits), it becomes a problem for words: for a rough approximation of the contour of the word *Alexandria* we picked 280 random sample points, which resulted in a running time of about 64 minutes on a 550MHz PentiumIII.

7. EXPERIMENTAL RESULTS

For a set of test data on which to test the algorithms, three sets of 10 consecutive pages were chosen from George Washington's letters. These letters are part of the archival collection at the Library of Congress, where they have been scanned in and kept on microfiche. The images of the documents used in the experiments were scanned from the microfiche. Because these were third generation copies, there was significant degradation in the quality of the image; unwanted artifacts had been introduced and image quality had been lost. Additionally, the high JPEG compression-ratio which was used for storing the collection introduced additional image artifacts.

7.1 Preparing the Test Sets

All documents in the 3 test sets were segmented into word images using the techniques described in [9]. Each word was manually tagged with its ASCII equivalent. Even for such a small document set, this was a very tedious task and it became very clear how useful word spotting is for indexing documents. The segmentation algorithm was not perfectly accurate, resulting in some cropped words. These were treated as follows: if less than a letter was missing, the entire word was used as the ASCII equivalent. If more than that was missing, however, only the part of the word that was included was used as the ASCII equivalent. If no letters could be made out, a control character ('#') was used as the word tag.

With the image tags, relevance between images could be judged. An image was deemed relevant to another if the two images had the same ASCII equivalent. When a ranked list

	Data Set 1	D. Set 2	D. Set 3
Average Precision	0.4956	0.3365	0.7338
R-Precision	0.4852	0.3319	0.6846

Table 4: Comparison of recall-precision scores (using 'affine" normalized EDM for matching) across the three data sets.

	XOR	SSD	EDM	SLH
Average Precision	0.5414	0.5266	0.7338	0.4243
R-Precision	0.5011	0.4706	0.6846	0.3846

Table 5: Comparison of recall-precision scores for the four matching algorithms.

of matches was returned by one of the matching algorithms, recall-precision scores were computed.

After running tests on the three data sets, it became apparent that two of the sets of 10 pages were of extremely low quality and did not allow any of the matching algorithms to work well. Figure 7 shows three typical lines, one from each data set. Clearly, the lines from the first two test sets are harder to recognize. These pages also caused the most segmentation errors and difficulty in transcription. Because the quality of the images within these two sets were so poor, all tests reported here were run on the third set. In addition, the third data set contained the most repeated non-stopwords that would be useful to index and could be chosen for tests. Table 4 shows the differences in average recall-precision scores between the three sets.

15 words of varying lengths (2 to 14 characters) and varying numbers of occurrences (4 to 110 appearances) were chosen to form a test set for assessing the quality of the algorithms. Each of the words was run through the matching algorithms to generate a ranked list, and the average recall-precision scores were calculated over all 15 queries. Table 5 shows the differences in how the algorithms performed over these queries.

7.2 Test Results

The results found in the experiments indicate that the EDM algorithm is roughly as fast as XOR, but more importantly is the most accurate matching method of those tested, with recall-precision scores well above those achieved using either SSD or SLH. This is a surprising result, since SSD is a widely used matching algorithm in image processing and SLH is a very flexible matching method, compensating for affine transformations between a pair of images. Note, however, that the normalization to some extent compensates for scale, shear and positional differences for all algorithms except SLH. In previous work on good quality images it was noticed that SLH performed better [6]. This might indicate that the performance of SLH is sensitive to the quality of the image.

	No Normalization	With Norm.
Average Precision	0.0832	0.7338
R-Precision	0.0645	0.6846

Table 6: Comparison of recall-precision scores for EDM with and without image normalization.

XOR	SSD	EDM	SC	SLH(half)	SLH(full)
13.0	72.0	14.3	3855.1	121.4	5508.8

Table 7: Running times in seconds *including* normalization for matching a pair of images of the word *Alexandria* on a 550MHz PentiumIII. The SLH algorithm was run for full- and half-size images.

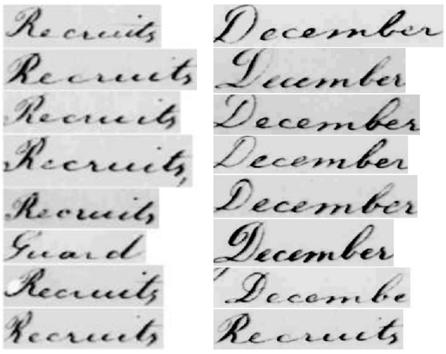

(a) Matches for Recruits
(b) Matches for December

Figure 9: Top 8 matches from rankings, as determined by the EDM algorithm. The template is the top ranked image in each list (Recruits and December respectively). The average precision values for these two queries are 0.9526 and 0.7764 respectively. The word images shown are the original word images (before pre-processing). Note the wide variation in size and also the matching error in each case.

Table 6 shows the differences in EDM performance with and without normalization of the images to be matched. The results clearly indicate the importance of normalization, especially for images of poor quality.

Table 7 shows the time required to perform a single comparison using each algorithm. Note that these times include the time required for image normalization for each algorithm, which explains why the cost of the EDM and XOR algorithms is about the same.

7.3 Test Summary

The above results indicate that normalization is an important step for achieving good performance. Further, the "affine" normalized EDM algorithm achieves about 73% average precision on documents with reasonable quality. The success of EDM could not have been predicted by previous work in image matching. When the scanned images are of really poor quality the precision falls further. This indicates that the simplest way of improving the performance of the algorithms is to produce good quality scans in the first place.

Some insight into possible improvements in the EDM algorithm can be gathered by viewing the results in the ranked lists presented in Figure 9. One observation is that the last entry in the list for "December" is actually the same word image as the second entry in the "Recruits" list. Thus, in an actual implementation of the matching algorithms, where word classes were to be formed, the "Recruits" ranked eighth in the "December" list would not be included in the "December" word class, as it appears more highly ranked in another list. By gathering these ranked lists into actual word classes, the recall-precision scores could be improved. Other possibilities are discussed in the next section.

The EDM algorithm has been used to create an example implementation for word indexing on a small set of documents. The demonstration can be visited at http://ciir.cs.umass.edu/research/wordspotting. Note that the demonstration does not constitute a complete implementation, its purpose is rather to illustrate the idea of word spotting.

8. FUTURE WORK

Several directions to continue the work in this area have already been proposed. These fall into three categories: pruning and normalization methods, additional testing, and implementation of other matching algorithms.

We consider the pruning and normalization steps far from finished. Pruning is a crucial part of the word spotting process. In order to allow the approach to scale to large document collections, more efficient pruning techniques must be developed: as the number of words n in a collection grows, the number of possible word pairs grows with $O(n^2)$. An efficient pruning technique should be able to quickly reduce the set of possible matches, so that the remaining set is only of size $O(n)$ or slightly worse. Pruning methods with this characteristic and high recall-values still remain to be found. We are currently implementing a demonstration with the EDM algorithm for a test set of 100 documents. This will provide further insight into scaling issues.

This work showed the importance of normalization for matching a pair of images. Some of the techniques presented here are empirical, for example image binarization with a manually selected threshold. These processes will be automated in order to minimize the need for fine-tuning by the user of an implemented system. Finally, we are currently investigating a number of techniques for matching word images and hope to further improve the current results.

9. CONCLUSION

There are two problems central to indexing large collections of handwritten documents in the presence of faded ink, scanning and compression artifacts: the segmentation of pages into words (largely solved by the scale-space techniques discussed in [9]) and the matching of word images in order to build word clusters. The results here indicate that normalization is extremely important in achieving good performance - this is to be expected given the quality of the images. EDM, the best performing algorithm, gives roughly 73% average precision on documents with reasonable quality. When the scanned images are of really poor quality the precision falls further. This indicates that the simplest way to achieve good results is to produce good quality scans in the first place.

10. ACKNOWLEDGMENTS

We thank Fangfang Feng for assistance. The last author would also like to thank Joshua Sarro for his assistance. We thank the authors of [1] for making their shape context code publicly available at http://www.cs.berkeley.edu/projects/vision/shape/.

11. REFERENCES

[1] S. Belongie, J. Malik and J. Puzicha. Shape Matching and Object Recognition Using Shape Contexts. *IEEE Transactions on Pattern Analysis and Machine Intelligence* **24**:24 (2002) 509–522.

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

[3] P.-E. Danielsson. Euclidean distance mapping. *Computer Graphics and Image Processing*, **14** (1980) 227–248.

[4] S. Khoubyari and J. J.Hull. Keyword location in noisy document image. In *Second Annual Symposium on Document Analysis and Information Retrieval, UNLV, Las Vegas*, pp. 217–231, 1993.

[5] S. Madhvanath and V. Govindaraju. The role of holistic paradigms in handwritten word recognition. IEEE Transcations on *Pattern Analysis and Machine Intelligence*, **23**:2 (2001)

[6] R. Manmatha and W. B. Croft. Word spotting: Indexing handwritten manuscripts. In M. Maybury, editor, *Intelligent Multi-media Information Retrieval*, pp. 43–64. AAAI/MIT Press, 1997.

[7] R. Manmatha, C. Han, and E. M. Riseman. Word spotting: A new approach to indexing handwriting. In *Proc. Computer Vision and Pattern Recognition Conference*, pp. 631–637, 1996.

[8] R. Manmatha, C. Han, E. M. Riseman, and W. B. Croft. Indexing handwriting using word matching. In *Digital Libraries '96: 1st ACM International Conference on Digital Libraries*, pp. 151–159, 1996.

[9] R. Manmatha and N. Srimal. Scale space technique for word segmentation in handwritten manuscripts. In *In the Proc. of the Second International Conference on Scale-Space Theories in Computer Vision (Scale-Space'99)*, pp. 22–33, Sep. 1999.

[10] P. W. Palumbo and S. N. Srihari. Postal address reading in real time. *International Journal of Imaging Systems and Technology*, **7** (1996) 370–378.

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

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

[13] H. Turtle and W. Croft. A comparison of text retrieval models. *Computer Journal*, **35**:3 (1992) 279–290.