Translating Handwritten Bushman Texts

Kyle Williams
Department of Computer Science
University of Cape Town
Private Bag X3, Rondebosch, 7701
South Africa
kwilliams@cs.uct.ac.za

Hussein Suleman
Department of Computer Science
University of Cape Town
Private Bag X3, Rondebosch, 7701
South Africa
hussein@cs.uct.ac.za

ABSTRACT

The Bleek and Lloyd Collection is a collection of artefacts documenting the life and language of the Bushman people of southern Africa in the 19th century. Included in this collection is a handwritten dictionary that contains English words and their corresponding |xam Bushman language translations. This dictionary allows for the manual translation of |xam words that appear in the notebooks of the Bleek and Lloyd collection. This, however, is not practical due to the size of the dictionary, which contains over 14000 entries. To solve this problem a content-based image retrieval system was built that allows for the selection of a |xam word from a notebook and returns matching words from the dictionary. The system shows promise with some search keys returning relevant results.

Categories and Subject Descriptors

H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval; H.3.3 [**Information Storage and Retrieval**]: Digital Libraries; I.4.6 [**Image Processing and Computer Vision**]: Segmentation—*edge and feature detection*

General Terms

Algorithms, Experimentation, Design, Performance

Keywords

Information retrieval, cbir, cultural heritage preservation, digital libraries, handwritten manuscripts, image processing

1. INTRODUCTION

The Bleek and Lloyd Collection consists primarily of handwritten notebooks that contain stories in the |xam and !kun Bushman languages. In some of these notebooks there are English translations alongside the Bushman words but in other cases these translations do not exist or are unclear.

In addition to notebooks, the Bleek and Lloyd collection also contains a dictionary for the |xam Bushman language in which each entry contains an English word and its corresponding |xam translation. This dictionary can be used to translate |xam words that appear in the Bleek and Lloyd notebooks. The problem, however, is that it is not practical to do this by hand simply due to the size of the dictionary, which contains over 14000 pages.

A solution to the impracticality of manual translation is to build a system that can assist in the automatic translation of words that appear in the notebooks. The script that the |xam language was recorded with, however, can not be represented using Unicode, thereby making optical character recognition unsuitable for this task. However, it is important that the dictionary can still be used to assist researchers and scholars in understanding and interpreting the stories in the notebooks in the Bleek and Lloyd Collection. To overcome this problem a content based image retrieval (CBIR) system, the Bushman OnLine Dictionary (BOLD) Translator, was built to allow for the matching and translation of |xam words that appear in the notebooks. The CBIR system is separated into two parts: a preprocessor and a matcher. The preprocessor segments words, extracts features from them and stores these features in inverted files. The matcher takes a word image as a search key and identifies candidate matches based on feature similarity. Candidate matches then have more accurate pixel-level matching performed on them and the results are presented to the user.

This paper begins by introducing the Bleek and Lloyd collection and discusses related work. The design of the system is then discussed followed by an evaluation of its speed and accuracy and, lastly, conclusions are drawn.

2. BLEEK AND LLOYD COLLECTION

The Bushmen people of southern Africa are widely considered as being some of the oldest human inhabitants of the Earth. Unfortunately, due to the rapid onset of globalisation and Western influence, much of their wisdom, ancient knowledge, art, culture, customs and language have been lost. Some of it, however, has been preserved through a handwritten record of the |xam and !kun languages made by Lucy Lloyd and Wilhelm Bleek during the 1870s. These stories, along with art and dictionaries were preserved and have come to be known as the Bleek and Lloyd collection. The collection consists of 157 notebooks containing 14128 pages, 752 drawings and over 14000 dictionary pages for the |xam dictionary [15]. There are three types of objects in the |xam dictionary: envelopes, slips and entries. For each

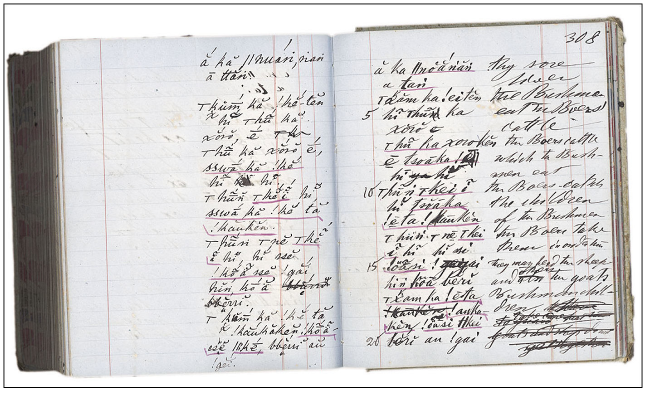


Figure 1: A notebook page in the Bleek and Lloyd collection.

English word there is at least one envelope that contains the English word written on the front. Inside each envelope there is exactly one slip that contains the English word written at the top and one or more inserts that contain the English words and their corresponding |xam translations. The BOLD Translator concentrates on the inserts in the dictionary since they can be used for translating |xam words.

The Bleek and Lloyd Collection is jointly owned by the National Library of South Africa, the Iziko National Museum of South Africa and the University of Cape Town, and in 2007 the collection was recognised as a UNESCO Memory of the World Collection. In 2003, the Lucy Lloyd Archive and Research Centre at the Michaelis School of Fine Arts undertook the the task of producing high resolution scans of all the artefacts that make up the Bleek and Lloyd Collection, for which digital library systems have been and are currently being built. Figures 1 and 3 show examples of a notebook page and a dictionary insert respectively.

In 2007, Suleman [15] built a digital library system for the notebooks and the art in the Bleek and Lloyd collection using an XML-centric solution. An XML-centric solution was chosen for this archive because it: required no installation from the end user; was platform independent; allowed for easier processing; and had long term preservation benefits. Suleman showed that an XML-centric approach had many advantages over traditional database-based archives. However, scalability issues were identified as a limiting factor.

3. RELATED WORK

3.1 Cultural Heritage Preservation

Museums and libraries world-wide digitise their valuable historical documents with the goals of long term preservation and ease of access. A key requirement for digital collections is that objects are annotated in order for them to be accessible and exploitable [2]. Annotation needs to be done either manually or automatically. The type of annotation required for the BOLD Translator includes the mapping of words in the notebooks to their corresponding dictionary entries. Manual annotation suffers from a key problem in that it is extremely tedious and expensive. In response to this, automatic systems have been built for storing, accessing and annotating digital collections. These systems for managing digital collections have been used in a wide variety of projects for preserving cultural heritage such as the following:

- Europeana[1] is a project, funded by the European Commission, that aims to bring together Europe's cultural heritage and contains links to over 6 million digital objects.
- American Memory[2] is a gateway that provides free and open access to the digitised collections of the Library of Congress and other institutions.
- Aluka[3] is an international effort at building a digital library made up of scholarly content about and from Africa.
- The MEMORIAL Project is a project funded by the European Union to enable the virtual distribution of paper-based archives that are currently held at museums and libraries [1].
- The Multilingual Inventory of Cultural Heritage in Europe (MICHAEL)[4] is a project funded by the European Commission to establish a new service for European cultural heritage. The project's vision is to create a service that will allow people to find and explore digital European cultural heritage on the Internet.
- The Greek Orthodox Archdiocese of America (GOA) is digitising their large collection of religious and historical artifacts [12].

Having briefly mentioned several cultural heritage preservation attempts, the next section will discuss related work done within the fields of content based image retrieval, word segmentation and a technique for handwritten word matching called word spotting.

3.2 Content Based Image Retrieval

Text based image retrieval systems date back to the early 1970s, where a popular method for image retrieval involved manually annotating images and then conducting searches based on these annotations [14]. Content Based Image Retrieval (CBIR) was proposed in the early 1990s as an alternative technique for image retrieval where, instead of basing retrieval on text-based annotations, the visual properties of an image, such as colour and shape, are used for retrieval. In this sense, images are used to search for other images in CBIR systems.

CBIR came about as a result of two fundamental shortcomings of text-based image retrieval: the amount of effort required to annotate images in large databases, and the subjectivity of human perception to the meanings of images [14]. There are generally three types of CBIR: primitive queries (query by example), semantic retrieval and automatic retrieval [3], with primitive queries being the most common and therefore the only type discussed in detail here.

In primitive CBIR, images are analysed based on a number of primitive features, most notably colour, texture, shape and colour layout [14]. This analysis usually takes place on segmented parts of the image, as it has been shown that the

[1]http://www.eurepeana.er
[2]http://memory.loc.gov/ammem/index.html
[3]http://www.aluka.org/
[4]http://www.michael-culture.org

shape and colour layout analyses depend on good segmentation. Once these features have been extracted from an image, it is possible to construct a signature (or feature vector) for the image that then can be compared to signatures of other images in order to find matches.

retrievr[5] and imgSeek[6] are two image-based search engines based on the fast multiresolution image querying algorithm developed by Jacobs et al [4]. The fast multiresolution image querying algorithm, and thus retrievr and imgSeek, uses a hand drawn sketch or low quality scan of the image to be retrieved. retrievr makes use of the hand drawn sketch or low quality scan to search Flickr![7] for similar photos, whereas imgSeek is a photo collection manager with built in CBIR. Jacobs et al tested their algorithm by using sketches and low quality scans of actual images to see if they could find the correct image in a database of sample images. They found that their algorithm was extremely fast and effective and able to pinpoint the correct image to within a 1% subset of the original sample.

Word spotting is a technique for grouping occurrences of the same word, where it exists in multiple locations in a collection of documents. It has been shown that word spotting is well suited to the case of handwritten historical documents, where optical character recognition (OCR) techniques do not work well [6]. However, word spotting falls short of meeting the requirements of the BOLD Translator because word spotting systems generally focus on determining the similarity of all images in a collection, in order to build an index [13]. The BOLD Translator, on the other hand, requires matching of images to be performed on demand.

There have been several attempts at building CBIR systems for words, such as a system for matching textual queries to word images by translating the textual queries into word images using LaTeX [11], and, in an approach more similar to that of the BOLD Translator, a CBIR system was built for archives of handwritten Ottoman documents [17].

3.3 Word Segmentation

Word segmentation is a prerequisite for any system that attempts to compare words that appear on a page. There have been several attempts at performing segmentation of words that appear on a page. However, Manmatha et al [10] note that most systems that do this have been developed for machine-printed text and that there are not many systems that deal with handwritten text. Furthermore, they note that, of the few systems developed for handwritten text, most focus on special kinds of texts, such as cheques or addresses on letters.

Line segmentation is often a prerequisite for word segmentation techniques. There are a number of approaches to doing line segmentation, with common approaches including projection profiles, Hough transforms, smearing methods, grouping methods, repulsive-attractive methods and stochastic methods [7]. Louloudis et al [8] propose an algorithm for word segmentation that makes use of the gap metrics between words. Using this technique, Louloudis et al were able to achieve a segmentation success rate of 90.82%. Manmatha et al [10] suggest a scale space technique for word segmentation and, using their technique, were able to achieve segmentation success rates of 77-96%, with an average segmentation success rate of 87%.

Significant work has been done in the areas of digital cultural heritage preservation, CBIR systems and word segmentation. These areas of research all play a role in the BOLD Translator and are loosely combined to create the system.

[5]http://labs.systemone.at/retrievr
[6]http://www.imgseek.net/
[7]http://www.flickr.com

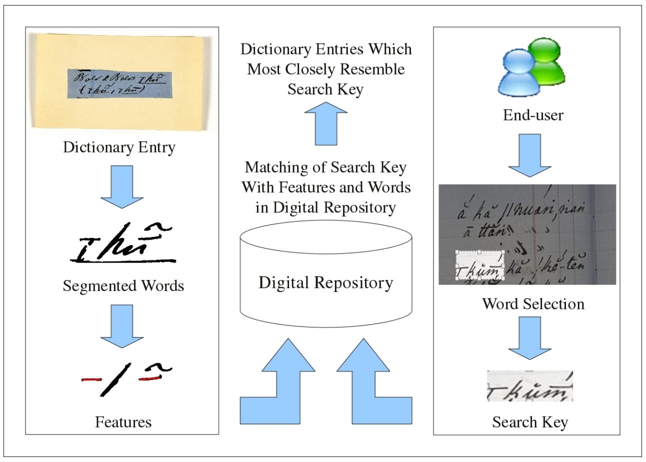

Dictionary Entry
Segmented Words
Features
Dictionary Entries Which Most Closely Resemble Search Key
Matching of Search Key With Features and Words in Digital Repository
Digital Repository
End-user
Word Selection
Search Key

Figure 2: Overview of the BOLD Translator

4. IMPLEMENTATION

The BOLD Translator attempts to overcome the lack of practicality in doing manual translations between the Bleek and Lloyd notebooks and dictionaries by allowing end users to search for the English translation of a |xam word using only an image of the |xam word as a search key. The BOLD Translator is designed such that it requires no training of a dataset and could be adapted to work with other collections. A brief overview of the system is given here followed by a more detailed description of the various components that make up the system. Figure 2 gives an overview of the system.

- **Preprocessor:** The following operations are performed on every image that is added to the repository:
  - The image is cleaned by smoothing to remove noise.
  - Words in the image are segmented.
  - For each segmented word, a set of known features are extracted and stored in inverted files.
  - The colour image, segmented words and inverted files are stored in the digital repository.
- **User Input and Matcher:** User input involves the user selecting a search key that is then matched against the repository. The matcher compares the search key features and the search key to the contents of the repository and returns the closest matches. This is done as follows:
  - The user selects a search key.
  - The same set of features as extracted by the preprocessor is extracted from the search key.

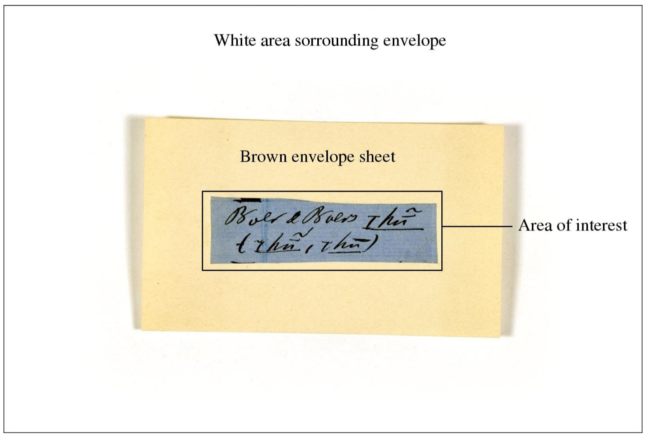


Figure 3: A high resolution TIFF with the AOI highlighted

- Inverted files are searched for words that have features that correspond to the features of the search key and each word is given a feature score.
- Words that have feature scores above some threshold t_1 have more accurate matching performed on them.
- Words that have accurate matching scores above some threshold t_2 are returned to the user as results.

4.1 Preprocessor

Preprocessing in the BOLD Translator involves cleaning images such that processing can take place on them, segmenting words in an image and lastly extracting features from the segmented words and storing these features in inverted files.

4.1.1 Image Cleaning

The high resolution TIFF images that belong to the |xam dictionary contain large areas of space that are not necessary for translation. These areas include the large whitespace surrounding the image, as well as the brown envelope sheet on which the blue sheet of paper containing the |xam words appear. In every high resolution TIFF the specific area of interest (AOI) is the rectangular area of the TIFF that contains all the handwritten words. Figure 3 shows one of the original TIFFs with the large whitespace, the brown envelope and the AOI labeled. The goal of the image cleaning step of the preprocessor is to identify the AOI in every image, crop the image around the AOI and provide as output a thresholded AOI that will be used for segmentation, feature extraction and matching.

ImageMagick[8] is used to crop the AOI in every image as follows:

1. The image is thresholded in order to convert everything in the image to white except the blue sheet of paper containing the Bushman words and any writing that appears anywhere on the page.

2. A median filter is used in order to smooth away any black marks that might be isolated on the page and that are most likely noise.

[8]http://www.imagemagick.org

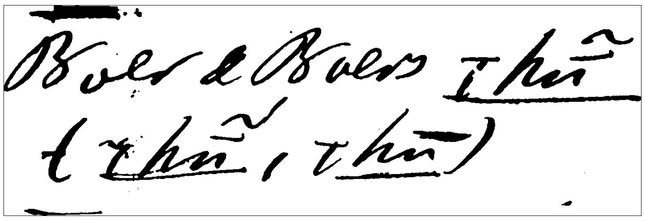


Figure 4: Final output from image cleanup

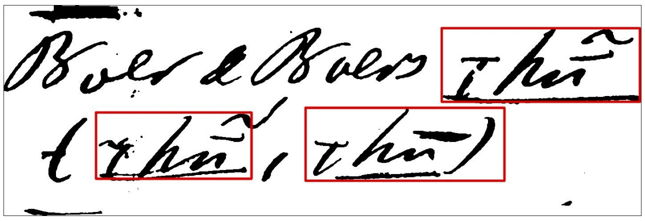


Figure 5: Fully boxed word for segmentation

3. The white area surrounding the image is trimmed, leaving only the area containing the handwriting.

There is a negative effect of performing this preprocessing on the image in that the smoothing results in a lot of information in the image being lost. Therefore, instead of performing the changes on the actual image, the coordinates of the AOI (as identified by the trim function) are stored. The actual image is then cropped at these coordinates, resulting in a full colour image without any information loss. After the colour image has been cropped, it is then thresholded and smoothed slightly so as not to lose a lot of information in the image. Figure 4 shows the final output of the AOI.

4.1.2 Word Segmentation

The role of segmentation is to, as accurately as possible, identify the Bushman words that appear in each thresholded AOI provided by the preprocessor. AOIs contain both English and Bushman words. This creates the problem that, by segmenting all the words in AOIs, non-Bushman words will be introduced into the dataset. This could have a negative effect on the system in that it could increase the size of the dataset, resulting in performance issues, and, it could add additional images that could be candidate matches and thus have a negative effect on accuracy.

The segmenter therefore exploits known information about the collection - that almost every Bushman word is underlined by a solid horizontal line. The segmenter first attempts to identify every line underlying Bushman words in an image, then attempts to identify the left and right end points of every word and finally identify the top of every word. Figure 5 shows the bounding boxes of words identified by the segmenter.

4.1.3 Feature Extraction

Features play a significant role in the system as they are used to prune a large dataset such that accurate matching can be performed on images that have similar features to the key image. Once the words in an image have been segmented, features are then extracted from them and stored in inverted files that are used to lookup words that have similar features when searching takes place.

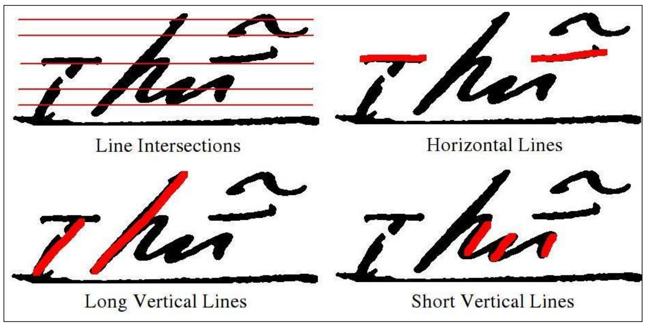


Figure 6: Features used in the system

The following features are used in the system: intersections, horizontal lines, long vertical lines and short vertical lines. With the exception of intersections, all the features are identified using connected component analysis [5]. Figure 6 shows the features used in the system.

The intersection feature is determined by projecting horizontal lines through every word at the following locations: 10% from the top, $\frac{1}{3}$ from the top, $\frac{1}{2}$ of the way through the image, $\frac{1}{3}$ from the bottom and 10% from the bottom. For each of these horizontal projections the number of intersections with line strokes are counted and this represents the intersection feature.

Horizontal lines in words are detected using connected component analysis. A minimum threshold is set for the length of horizontal lines in order to prevent horizontally connected components that are not necessarily horizontal lines in characters from being detected.

Long vertical lines are detected using connected component analysis by rotating the word so as to make vertical lines horizontal and then using horizontal line detection. Long vertical lines are defined as lines that are at least as long as 65% of the height of the word image.

Short vertical lines are detected in the same way as long vertical lines, except that short vertical lines are defined as lines that are at least as long as 30% of the height of the word image but less than 65%.

4.2 Matching

Matching involves taking a word as input (referred to as the key) and finding images in the dataset that are similar. Matching works as follows:

1. A key image is selected by the user for matching and its features are extracted.

2. The user sets feature weights and variation allowance (discussed in next section).

3. Based on these inputs, the inverted files are searched for words that have features that correspond to the features of the search key and each word gets a feature score based on its similarity to the search key.

4. Words that have feature scores above some threshold t_1 go through a process of more accurate matching and each of those words gets an accurate matching score.

5. Words that have an accurate matching score above some threshold t_2 are then displayed to the user.

Key selection is performed by the end user and involves selecting a word from one of the pages of the Bleek and Lloyd collection using a JavaScript box select tool that makes use of the Yahoo! User Interface (YUI) Library[9].

The user is able to set how important each of the features are for matching. The motivation behind this is that some features, such as horizontal lines, might be more unique than others, such as intersections, and therefore might result in better matches.

Feature weights are set such that:

$$\sum_{i=1}^{4} W_i = 1 \tag{1}$$

where W_i = the weight given to feature i. Feature sums can be presented in the form of

$$Y = W_1 + W_2 + W_3 + W_4 \tag{2}$$

Various combinations of feature weights are investigated further in section 5.2.1.

It is difficult to detect and match features perfectly due to the quality of the original manuscripts and differences in style and handwriting. For this reason, feature variation has been introduced into the system in order to overcome this shortcoming by allowing for variation in feature correspondence. In this sense, instead of features corresponding perfectly, variation allows for them to differ to some certain extent. Variation allows for flexibility in pruning the dataset for accurate matching, however it does have the drawback that it returns more results, which affects precision and performance.

The feature score of two words is a measure of how similar they are, based on features alone, and is calculated using feature weights. To demonstrate this further, consider the feature sum

$$Y = 0.1 + 0.3 + 0.3 + 0.3.$$

For every word in the dataset for which the first feature corresponds with the first feature in the key, increase that word's feature score by 0.1. Similarly, for every word in the dataset for which the second feature corresponds with the second feature in the search key, increase that word's feature score by 0.3, and so on for all the features. The result of this is a list of words for which each word has a feature score. This list of words is then ranked and the feature scores are normalised over the range [0-1] such that the word that has the highest feature score has a normalised feature score of 1. Words that have a normalised feature score above some threshold t_1 are passed on to the next step for more accurate matching to take place on them.

Accurate matching involves determining which images are most similar to the search key at a pixel level. Accurate matching scores are normalised over the range [0-1], with the best match having a score of 1. Matches that have an accurate matching score above some threshold t_2 are presented to the user as results. Three different accurate matching algorithms, which are all variations of one another, have been implemented.

The difference (DIF) matching algorithm works by first resizing each word image and the key so that they are the same size. The word image is then superimposed on top of the key image and the sum of the absolute values of the

[9]http://developer.yahoo.com/yui/

differences between their corresponding pixels is calculated. This is repeated several times by shifting the word image 1, 2, 3, 4 and 5 pixels in every direction. The lowest value of the sum of the absolute values of all the shifts is returned as a similarity score, where a perfect match has a similarity score of 0.

Calculating the XOR of two images was mentioned by Manmatha et al [9] as a prerequisite to the next matching algorithm explored: Euclidean distance mapping. Calculating the XOR of two images can also be used as a matching algorithm. The algorithm works by first inverting the images such that the backgrounds become black and the foregrounds become white. Thereafter the algorithm works the same way as the difference algorithm, by resizing the images and superimposing the word image on the key image as well as shifting the word image 1, 2, 3, 4, and 5 pixels in every direction. The XOR of each set of corresponding pixels is then determined and the number of white pixels are counted to determine a similarity score, where a perfect match has a similarity score of 0.

The Euclidean Distance Matching (EDM) algorithm discussed by Manmatha et al works the same as the XOR matching algorithm, with the difference being that instead of counting the number of white pixels, the Euclidean distance from each white pixel to the closest black pixel is determined. Using this method, a white pixel that exists in a blob of white pixels will have a larger Euclidean distance to a black pixel than a white pixel in isolation. A similarity score is calculated where the similarity score is equal to the sum of the Euclidean distances from every white pixel to its closest black pixel. As with the other matching algorithms, a perfect match has a similarity score of 0.

Having given an overview of the system design and implementation, the evaluation of the system will now be discussed.

5. EVALUATION

The BOLD Translator was evaluated using both modular testing for each of the components that make up the system as well as end-to-end testing for the system as a whole. Specifically, the system was tested in terms of performance measures related to speed as well as information retrieval measures such as precision, recall and the F-score [16]. Precision is the proportion of relevant documents retrieved in a query and, in general, is calculated as:

$$\text{Precision} = \frac{\text{No. of relevant documents retrieved}}{\text{number of documents retrieved}}. \quad (3)$$

Recall is the proportion of relevant documents in a collection that are actually retrieved in a query and, in general, is calculated as:

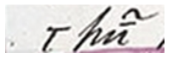
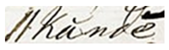
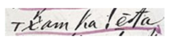
$$\text{Recall} = \frac{\text{No. of relevant documents retrieved}}{\text{number of relevant documents in the collection}}. \quad (4)$$

The F-score is the weighted harmonic mean of the precision and recall and, in general, is calculated as:

$$\text{F} = \frac{2 \cdot \text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}. \quad (5)$$

For all experiments that use search keys, three different words that are known to exist in the collection and have varying characteristics and sizes were selected as search keys (Table 1). Each of these words was selected in 3 different ways and each experiment was carried out on these three selections. The key selections are referred to as 1a, 1b and 1c for selections 1, 2 and 3 of the first key, 2a, 2b and 2c for the selections of the second key and, similarly, 3a, 3b and 3c for the selections of the third key. The purpose of using different selections of each key is to determine the extent to which word selection affects results.

Table 1: Search keys used in experiments

Key	Image	Size	Translation
Key 1		Small	Boer (farmer)
Key 2		Medium	Brother
Key 3		Large	Bushmen's gems

5.1 Word Segmentation

|xam words are the content that is retrieved in the BOLD Translator and therefore their successful segmentation is important as it has a significant effect on the overall ability of the system to retrieve correct matches. An experiment was conducted to evaluate the accuracy at which segmentation can be automatically performed on |xam words, as well as to identify the optimum minimum underlying line length for segmentation. The length of the underlying line is important as it needs to be able to distinguish between lines that underline words, and lines that make up the characters in the words. A subset of the collection that consisted of 10 images randomly picked for each letter of the alphabet was used. In cases where there were less than 10 images for any letter, all of the images for that letter were picked. The segmentation was performed on the subset of images for minimum word underlying line lengths of 10%, 20%, 30%, 40%, 50%, 60%, 70%, 80%, 90% and 100% of the width of the image. For each minimum underlying line length, the precision and recall was recorded for each individual image and then averaged out for the whole subset.

The concepts of precision and recall were applied as follows:

$$\text{Precision} = \frac{\text{No. of relevant segmented words retrieved}}{\text{number of segmented words retrieved}}. \quad (6)$$

$$\text{Recall} = \frac{\text{No. of relevant segmented words retrieved}}{\text{number of relevant segmented words}}. \quad (7)$$

For each minimum underlying line length, the weighted harmonic mean - or F-Score - for the average precision and recall of the subset was recorded. The results of the experiment are summarised in Figure 7, which shows that there is a clear tradeoff between precision and recall. Furthermore, precision increases consistently as the underlying line length increases. This is due to the algorithm becoming less likely to pick up lines that do not underline words, such as the line that crosses a "t." This is of course beneficial because it results in less noise being introduced into the dataset through bad segmentation. However, it has a negative effect on recall. As the underlying line length increases, it becomes less likely for the algorithm to detect lines that underline short words and this is shown by the decreasing recall as the underlying line length increases.

The F-score is a good measure of the trade-off between

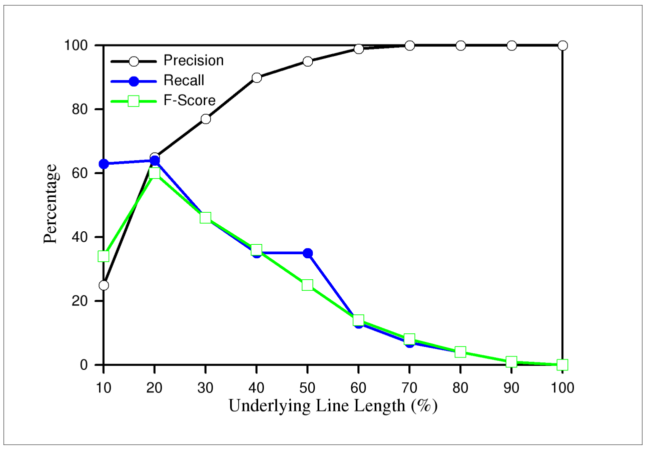


Figure 7: Analysis of different underlying line lengths for word segmentation

precision and recall and the optimal F-score exists at a minimum underlying length of 20% of the width of the image. At this point, recall is maximised and precision is at about 65%. Using the F-score as a measure of the ability to successfully segment the |xam Bushman words in an image, it is shown that segmentation occurs with around 60% success. Good segmentation is important in order to increase the chances of returning correct matches and thus the 60% success rate, while satisfactory, could be improved in order to improve the accuracy of the system as a whole.

5.2 Features

Features are used to prune the dataset to allow for more accurate matching at a later stage. Feature pruning should return a subset of the collection that contains word images that are likely to match the search key provided by the user. The size of this subset of candidate matches is determined by the threshold level for matches. The size of this subset is important since a high number is likely to slow down processing and decrease accuracy while a low number is likely to speed up processing, but also decrease accuracy. A number of experiments were carried out to determine the ability of features to prune the dataset, determine the best weights for the various features and identify performance issues when using features to prune the dataset.

5.2.1 *Using Features to Prune Results*

A subset equal to 20% of the collection, which amounted to 2921 images, was used to determine the extent to which features can be used to prune the dataset for matching, as well as to identify the optimum weights for each of the features. Each key image was matched with every word image and the similarity was measured based on feature correspondence only. Different feature weights were used in order to determine if a best set of feature weights exists. Feature weights were calculated according to Equation 1. The following feature weights in the form of

$$Y = W_1 + W_2 + W_3 + W_4$$

were used for the four features considered:

- Equal Weighting: $Y = 0.25 + 0.25 + 0.25 + 0.25$
- Reduced intersections: $Y = 0.1 + 0.3 + 0.3 + 0.3$
- Low intersections, high vertical lines: $Y = 0.1 + 0.2 + 0.35 + 0.35$
- Low intersections, high horizontal lines: $Y = 0.1 + 0.4 + 0.25 + 0.25$

These feature weights were selected to get a general feeling for how different feature weights can affect results. The experiment was carried out at threshold values of 100%, 80%, 60%, 40%, 20% and 0% and, for each threshold value, the precision, recall and F-score were recorded.

The experiment found that features can be used to prune the dataset but generally rely on a low threshold of 20-40% in order to have a positive recall. Of the various sets of feature weights considered, there appeared to be no set that consistently outperformed the others and this was shown by a relatively constant recall across the different feature weight combinations tested. The low threshold required for positive recall as well as the lack of effect of different feature weights could be due to the style and handwriting differences between word images. The algorithm works by extracting features from words. However, when two words are the same, but written in a different way, their features may not correspond. The next experiment set out to test if allowing for variation in feature correspondence improved the ability of features to prune the dataset.

5.2.2 *Introducing Variation into Feature Correspondence*

Feature variation allows for words with features that differ slightly due to style or handwriting to still have matching features. Feature variations of 1 and 2 are allowed, meaning that features differing from the key by |1| and |2| will still count as matches. In conducting this experiment, the equal feature weighting $Y = 0.25 + 0.25 + 0.25 + 0.25$ was used and the experiment was carried out at thresholds of 100%, 80%, 60%, 40%, 20% and 0%. For each threshold value, the precision, recall and F-score were recorded. Table 2 shows the results of the experiment for three different search keys both with and without variation. The experiment showed that introducing feature variation had a positive effect on the required threshold for relevant matches. While allowing for a variation level of 1 almost always improved on results with no variation, allowing for a variation level of 2 did not necessarily improve results. By introducing variation into the system, positive results were returned for threshold levels of 100%, 80% and 60% whereas, without variation, positive results were only returned at a threshold level of 40%. The evidence suggests that introducing variation into the system has a positive effect on results by reducing the required

Table 2: Recall for different thresholds (%) when variation is introduced

Key	Variation	100	80	60	40	20	0
1a	0	0.00	0.00	0.00	0.50	1.00	1.00
	1	0.00	0.50	0.50	1.00	1.00	1.00
	2	0.00	0.50	1.00	1.00	1.00	1.00
2a	0	0.00	0.00	0.00	0.00	0.75	1.00
	1	0.13	0.13	0.25	0.75	0.88	1.00
	2	0.00	0.00	0.63	0.88	1.00	1.00
3a	0	0.00	0.00	0.00	0.00	0.00	0.00
	1	0.00	0.00	0.00	0.50	0.50	1.00
	2	0.00	0.00	0.00	0.00	1.00	1.00

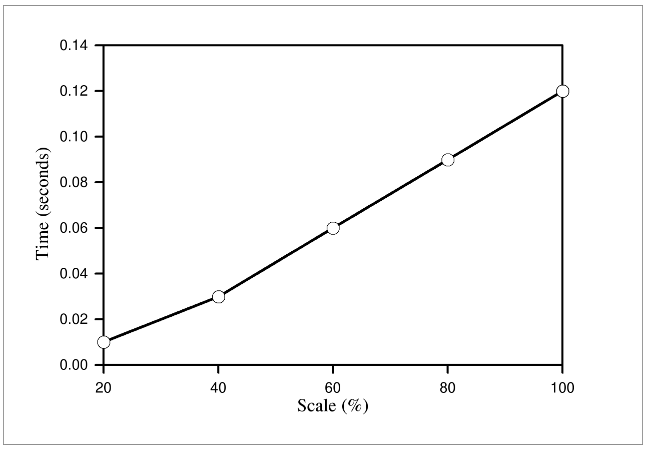


Figure 8: Effect of dataset size on feature performance

threshold and thereby reducing the number of words that have more accurate matching performed on them. However, introducing variation can potentially have a negative effect since variation increases the chance of bad matches exceeding the feature similarity threshold.

5.2.3 *Evaluating the Speed of Feature Based Matching*

Having conducted experiments to determine how well features can be used to prune the dataset, both with and without variation, an experiment was conducted to determine the speed at which feature matching takes place for various dataset sizes. Subsets of size 20%, 40%, 60%, 80% and 100% of the total collection were used to determine the effect that dataset size has on the speed of pruning results using features. A single key image was used to prune the results of each of the subsets and the time taken was recorded. This was repeated 10 times for each subset and the average time was calculated. An equal feature weighting with no variation was used for this experiment and the threshold for returning results was set at 50% feature similarity. The time reported is system time. Figure 8 shows the results of measuring the speed of the system as scale increases and it is clear that the size of the dataset has an effect on the speed at which feature pruning can occur. This is due to an increase in size, leading to more images that need to be compared, leading to larger inverted files and a larger feature score calculation. This highlights potential scalability issues as the system might perform slowly as the size of the dataset increases.

5.3 Accurate Matching

It has been shown that features can be used to quickly identify a subset of candidate images that are potentially matches for the search key provided by the user. The next set of experiments set out to determine how well the algorithms used in the BOLD Translator can be used to perform accurate matching.

5.3.1 *Evaluating the Accuracy of Each Matching Algorithm*

An experiment was conducted to determine which matching algorithm provided the most accurate matching. Each matching algorithm was run on a subset of 100 word images

Table 3: Average F-score at each threshold for each of the matching algorithms for the 9 images used in the experiment

%	100	80	60	40	20	0
DIF	0.17	0.24	0.23	0.12	0.07	0.05
XOR	0	0.02	0.09	0.07	0.06	0.06
EDM	0	0.03	0.01	0.08	0.06	0.06

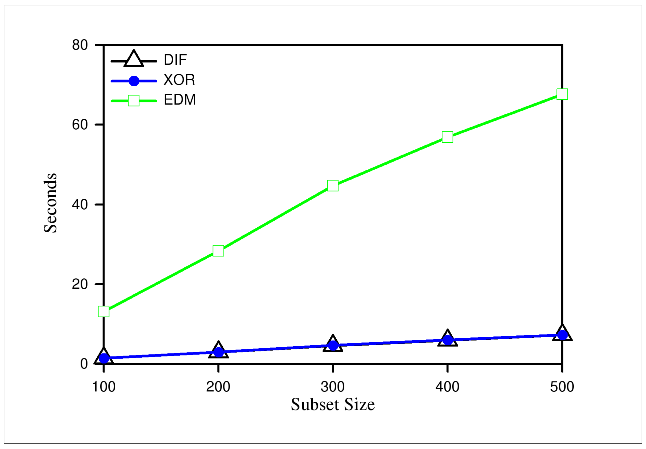


Figure 9: Matching algorithms performance measures

that had been automatically segmented by the preprocessor. Included in this subset were word images that matched the search keys used. The full range of search keys were used in this experiment and the experiment was carried out for similarity threshold values of 100%, 80%, 60%, 40%, 20% and 0%. For each threshold value, the precision, recall and F-score were recorded. Table 3 shows that average F-score for each matching algorithm and it is shown that the DIF algorithm generally results in better matches at higher thresholds. High thresholds are ideal since they result in a smaller number of results being returned to the user. While the DIF algorithm appears to outperform the XOR and EDM algorithms in terms of accuracy of matching, an important issue to consider is the time that each of these algorithms takes to run. This is investigated in the next experiment.

5.3.2 *Evaluating the Speed of Each Matching Algorithm*

An experiment was conducted to determine which matching algorithm is the fastest and how speed is affected as scale increases. The speed of each matching algorithm was recorded using subsets of 100, 200, 300, 400 and 500 images and each matching algorithm was run five times and the average speed was recorded. The time reported is user clock time. Figure 9 shows that the DIF and XOR matching algorithms are the fastest, while EDM is the slowest. Combining this result with the result in section 5.3.1 shows that the DIF algorithm is the best algorithm for matching since it is one of the fastest algorithms and is also the most accurate.

5.4 End-to-end Testing

Modular testing showed how each of the components that make up the BOLD Translator contribute to returning relevant results. The purpose of end-to-end testing was to de-

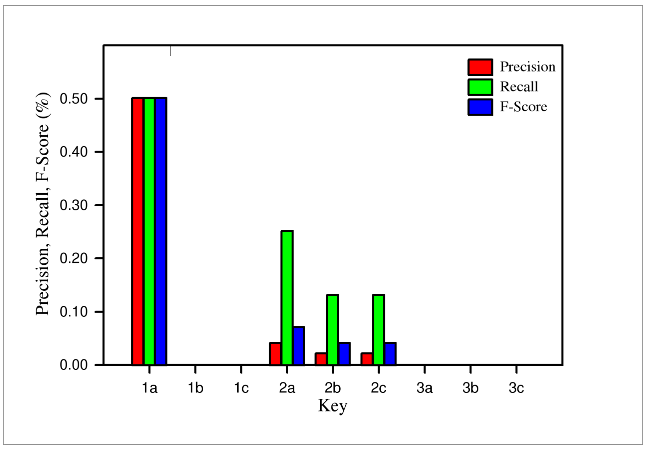


Figure 10: Precision, recall and F-score for end-to-end testing using optimal values

termine how well the BOLD Translator worked as a complete system. To do this a number of tests were carried out. Firstly, a test of the optimal values was carried out in that variable values that had been found to work well in modular testing were used in testing the end-to-end system. Thereafter, experiments were conducted to see how well the system performed and how accurate it was as scale increased.

5.4.1 *Testing the Optimal Values*

An experiment was conducted to determine if the optimal values for each component of the system, as identified by the modular experiments, can produce good end-to-end results. A subset of 20% of the collection, which amounted to 2921 images, was used and equal weights for features were used since no feature weight combinations were found to be optimal in section 5.2.1. Feature correspondence variation of 1 was used since it was shown in section 5.2.2 that this had a positive effect on results. The DIF matching algorithm was used since it was shown in section 5.3.1 and section 5.3.2 that this was the most accurate and best performing matching algorithm. A threshold level of 80% was used for feature scores since it was shown in section 5.2.2 that positive recall occurs when allowed feature correspondence variation is 1 and the threshold is 80%. Similarly, a threshold level of 60% was used for matching as it was shown in section 5.3.1 that the difference algorithm works well at a 60% threshold. The experiment was run using the full range of search keys and the precision, recall and F-score were recorded. Figure 10 shows that four of the nine image keys used for searching resulted in positive recall. These four keys result in better matches because they were tightly constrained when selected. This highlights the need for good selection of keys when making use of the system. It was shown that the optimal values do return positive results, however, this only occurred for 4 of the 9 search keys. Having shown that the system can be accurate when keys are tightly constrained when selected, the next two experiments set out to evaluate how accurate the system was and how well it performed as scale increased.

5.4.2 *Increasing Scale and Accuracy*

An experiment was conducted to determine the effect that increasing dataset sizes have on accuracy. Keys that were shown to return results in section 5.4.1 were used to test the system at increasing scales. Subsets that were made up of 20%, 40%, 60%, 80% and 100% of the full collection were used and, for each subset, the precision, recall and F-score was reported.

Table 4: Precision, recall and F-score as scale increases for keys that return positive results

Key	Measure	20%	40%	60%	80%	100%
1a	Precision	0.50	0.50	0.11	0.07	0.06
	Recall	0.50	0.50	0.50	0.50	0.50
	F-Score	0.50	0.50	0.18	0.13	0.10
2a	Precision	0.04	0.00	0.00	0.00	0.00
	Recall	1.00	0.00	0.00	0.00	0.00
	F-Score	0.08	0.00	0.00	0.00	0.00
2b	Precision	0.02	0.01	0.01	0.01	0.01
	Recall	0.13	0.13	0.13	0.13	0.13
	F-Score	0.04	0.02	0.02	0.01	0.01
2c	Precision	0.02	0.01	0.01	0.00	0.00
	Recall	0.13	0.13	0.13	0.00	0.00
	F-Score	0.04	0.02	0.02	0.00	0.00

Table 4 shows that a scale increase has a negative effect on precision in all cases. For keys 1a and 2b the recall remains constant while precision, and thus the F-score, decrease. The evidence suggests that strong matches will remain strong as the size of the dataset increases, as is the case for keys 1a and 2b, and that weak matches will disappear as the size of the dataset increases, as is the case for keys 2a and 2c. Key 2a resulted in better results for a subset size of 20% of the collection than key 2b for the same subset. However, key 2a failed to return positive results as the scale increased and thus is considered a weak match, while key 2b continued to return positive results as the scale increased and thus is considered a strong match.

5.4.3 *Increasing Scale and Performance*

An experiment was conducted to determine the effect that increasing dataset sizes have on performance. A single key was used to test the performance of the system at increasing scales. Subsets that were made up of 20%, 40%, 60%, 80% and 100% of the collection were used and optimal values were used for all variables. For each subset the user time was recorded 10 times and the average is reported. Figure 11 shows that the system becomes increasingly slower as the subset size increases. When 100% of the dataset was used, the system took 16 seconds to return results. This suggests that there is a need for optimisations throughout the system.

6. CONCLUSIONS

This paper described a content based image retrieval system for a collection of handwritten documents that cannot be represented using Unicode. The system was built with the goal of being able to find specific images of words within a large collection. The system includes a preprocessor that automatically segments words and extracts features from them, and a matcher, that identifies a subset of candidate matches based on feature similarity and then performs more accurate matching on this subset and returns the highest scoring results to the user.

Automatic segmentation of |xam Bushman words was per-

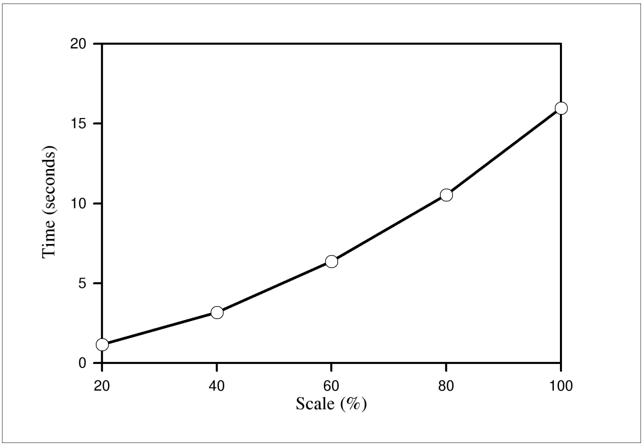


Figure 11: Performance as scale increases

formed with around 60% accuracy and it was shown that features can be used to prune the dataset for more accurate matching to take place, especially when variation is introduced. The DIF algorithm was shown to outperform the other algorithms implemented in the system in terms of accuracy as well as speed. End-to-end testing showed that when a good search key is selected, then relevant matches can be found. It was shown that the system performs well with matches taking approximately 1 second on a collection size of around 3000 images and 16 seconds on a collection size exceeding 14000 images. The evidence suggests that it is possible to do image-based translation of this nature.

Future work involves improvements at all levels of the system in order to increase the chances of correct translation. The system could further be extended by incorporating the use of predictive language modelling.

It is believed that a system of this nature has great potential in assisting researchers and scholars worldwide in their understanding of the |xam Bushman language as well as other dictionaries that are part of the Bleek and Lloyd Collection. Furthermore, it is believed that this system could be adapted to manuscripts of other languages which can not be represented using Unicode and provide a means of simple and efficient matching and translation of words.

7. REFERENCES

[1] A. Antonacopoulos and D. Karatzas. Document image analysis for world war ii personal records. In *DIAL '04: Proceedings of the First International Workshop on Document Image Analysis for Libraries (DIAL'04)*, page 336, Washington, DC, USA, 2004. IEEE Computer Society.

[2] R. Doumat, E. Egyed-Zsigmond, J.-M. Pinon, and E. Csiszar. Online ancient documents: Armarius. In *DocEng '08: Proceeding of the eighth ACM symposium on Document engineering*, pages 127–130, New York, NY, USA, 2008. ACM.

[3] J. Eakins and M. Graham. Content-based image retrieval. Technical report, Newcastle upon Tyne, United Kingdom: University of Northumbria at Newcastle, Institute for Image Data Research, 1999. JISC Technology Applications Programme Report 39, http://www.jisc.ac.uk/uploaded_documents/jtap-039.doc.

[4] C. E. Jacobs, A. Finkelstein, and D. H. Salesin. Fast multiresolution image querying. In *SIGGRAPH '95: Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*, pages 277–286, New York, NY, USA, 1995. ACM.

[5] T. Y. Kong and A. Rosenfeld, editors. *Topological Algorithms for Digital Image Processing*. Elsevier Science Inc., New York, NY, USA, 1996.

[6] Y. Leydier, F. Lebourgeois, and H. Emptoz. Text search for medieval manuscript images. *Pattern Recogn.*, 40(12):3552–3567, 2007.

[7] L. Likforman-Sulem, A. Zahour, and B. Taconet. Text line segmentation of historical documents: a survey. *Int. J. Doc. Anal. Recognit.*, 9(2):123–138, 2007.

[8] G. Louloudis, B. Gatos, I. Pratikakis, and C. Halatsis. Text line and word segmentation of handwritten documents. *Pattern Recogn.*, 42(12):3169–3183, 2009.

[9] R. Manmatha and W. B. Croft. Word spotting: indexing handwritten manuscripts. In *Intelligent multimedia information retrieval*, pages 43–64, Cambridge, MA, USA, 1997. MIT Press.

[10] R. Manmatha and N. Srimal. Scale space technique for word segmentation in handwritten documents. In *SCALE-SPACE '99: Proceedings of the Second International Conference on Scale-Space Theories in Computer Vision*, pages 22–33, London, UK, 1999. Springer-Verlag.

[11] S. Marinai, E. Marino, and G. Soda. Indexing and retrieval of words in old documents. In *ICDAR '03: Proceedings of the Seventh International Conference on Document Analysis and Recognition*, page 223, Washington, DC, USA, 2003. IEEE Computer Society.

[12] T. Nicolakis, C. E. Pizano, B. Prumo, and M. Webb. Protecting digital archives at the greek orthodox archdiocese of america. In *DRM '03: Proceedings of the 3rd ACM workshop on Digital rights management*, pages 13–26, New York, NY, USA, 2003. ACM.

[13] T. M. Rath and R. Manmatha. Word spotting for historical documents. *International Journal on Document Analysis and Recognition*, 9(2-4):139–152, APR 2007.

[14] Y. Rui, T. S. Huang, and S.-F. Chang. Image retrieval: Current techniques, promising directions, and open issues. *Journal of Visual Communication and Image Representation*, 10(1):39–62, 1999.

[15] H. Suleman. Digital libraries without databases: The bleek and lloyd collection. In *Proceedings of Research and Advanced Technology for Digital Libraries, 11th European Conference*, pages 392–403. Springer-Verlag, March 2007.

[16] C. J. van Rijsbergen. *Information Retrieval 2nd Edition*. Butterworth-Heinemann, London, 1979.

[17] I. Z. Yalniz, I. S. Altingovde, U. Güdükbay, and O. Ulusoy. Ottoman archives explorer: A retrieval system for digital ottoman archives. *J. Comput. Cult. Herit.*, 2(3):1–20, 2009.