

Honours Project Report

The BOLD Project: The BOLD Translator

Kyle Williams
kwilliams@cs.uct.ac.za

Supervised by: Dr Hussein Suleman
hussein@cs.uct.ac.za

| | Category | Min | Max | Chosen |
|---|---|------------|------------|---------------|
| 1 | Software Engineering/System Analysis | 0 | 15 | 0 |
| 2 | Theoretical Analysis | 0 | 25 | 0 |
| 3 | Experimental Design and Execution | 0 | 20 | 20 |
| 4 | System Development and Implementation | 0 | 15 | 10 |
| 5 | Results, Findings and Conclusion | 0 | 20 | 20 |
| 6 | Aim Formulation and Background Work | 10 | | 10 |
| 7 | Quality of Report Writing and Presentation | 10 | | 10 |
| 8 | Adherence to Project Proposal and Quality of Deliverables | 10 | | 10 |
| 9 | Overall General Project Evaluation | 0 | 10 | 0 |

Department of Computer Science
University of Cape Town

2009

Abstract

The Lloyd and Bleek Collection contains over 14000 dictionary pages with both an English word and its Bushman language translation. The notebooks in the Lloyd and Bleek Collection contain Bushman stories where in many cases English translations do not exist or are not clear. It is natural to assume that people making use of the notebooks would like to make use of the dictionary to translate words which appear in the notebooks. This, however, is not practical simply due to the magnitude of the dictionary. A need therefore exists to build a tool for interaction between the dictionary pages and the notebooks to allow for translation. A content based image retrieval (CBIR) system was built to do this and it was shown that it is possible to find the corresponding words in the dictionary by providing a single word from the notebooks as a search key. The system shows promising potential with well selected search keys returning relevant results.

Keywords

Digital libraries, information retrieval, cultural heritage preservation, image processing

Acknowledgements

I would first and foremost like to thank my supervisor, Hussein, for his commitment to the project, for sharing his knowledge and for always being available to assist me when problems arose - your cool, calm and collected approach kept me confident that what I was trying to achieve was possible.

I would like to thank my project partners, Lebogang and Sanvir, for their hard work on the BOLD Project and for always being around to discuss the project and the stress we were experiencing - guys, we've done it!

I would like to thank the computer science honours class of 2009 for the jokes and good times we had in the honours lab - thanks to you, this was one of my most enjoyable years at university.

I would like to thank my girlfriend, Kari, for putting up with me when I was distracted by this project and for being willing to listen as I worked my way through problems - you are my rock and your patience, understanding and support kept me going.

Most importantly I would like to thank my parents, Deborah and Vincent, for all the sacrifices that they made to give me an education and put me through university - without your sacrifices I would not be where I am today.

For my family...

Contents

| | | |
|----------|--|----------|
| 1 | Introduction | 1 |
| 1.1 | The BOLD Project | 1 |
| 1.1.1 | Motivation | 2 |
| 1.1.2 | The Framework | 2 |
| 1.2 | The BOLD Translator | 4 |
| 1.2.1 | Problem Statement | 4 |
| 1.2.2 | Motivation | 4 |
| 1.2.3 | Research Question | 4 |
| 1.2.4 | The Framework | 4 |
| 1.2.5 | Data Description | 6 |
| 1.2.6 | Ethical, Professional and Legal Issues | 7 |
| 1.3 | Summary of Report | 8 |
| 2 | Background | 9 |
| 2.1 | Introduction | 9 |
| 2.2 | The Lloyd and Bleek Collection | 9 |
| 2.3 | Preserving Cultural Heritage | 10 |
| 2.3.1 | The MEMORIAL Project | 10 |
| 2.3.2 | The MICHAEL Project | 10 |
| 2.3.3 | Greek Orthodox Archdiocese of America | 10 |
| 2.3.4 | Summary | 11 |
| 2.4 | Content Based Image Retrieval | 11 |
| 2.4.1 | Segmentation | 12 |
| 2.4.2 | Colour | 12 |
| 2.4.3 | Texture | 12 |
| 2.4.4 | Shape | 12 |
| 2.4.5 | Signatures | 13 |
| 2.4.6 | Comparing Signatures | 13 |
| 2.4.7 | CBIR Systems | 13 |
| 2.4.8 | Summary | 14 |
| 2.5 | Word Spotting | 14 |
| 2.5.1 | Word Spotting for the George Washington Collection | 14 |
| 2.5.2 | Word Spotting for Medieval Manuscripts | 15 |
| 2.5.3 | Summary | 15 |
| 2.6 | Word Segmentation | 16 |
| 2.6.1 | Line Segmentation | 16 |
| 2.6.2 | Gap Metrics | 16 |
| 2.6.3 | Scale Space | 16 |
| 2.7 | Discussion | 17 |

| | | |
|----------|--|-----------|
| 3 | Design and Implementation | 18 |
| 3.1 | Overview | 18 |
| 3.2 | Technology and Tools | 18 |
| 3.3 | Image Preprocessing | 19 |
| 3.4 | Segmentation | 21 |
| | 3.4.1 Identifying Underlining Lines | 21 |
| | 3.4.2 Identifying Word End Points | 22 |
| | 3.4.3 Identifying the Top of the Word | 22 |
| 3.5 | Feature Extraction | 23 |
| | 3.5.1 Features Used in the BOLD Translator | 23 |
| | 3.5.2 Inverted Files | 24 |
| 3.6 | Matching | 24 |
| | 3.6.1 Key Selection | 24 |
| | 3.6.2 Feature Weights | 25 |
| | 3.6.3 Variation | 25 |
| | 3.6.4 Feature Scores | 25 |
| | 3.6.5 Accurate Matching | 25 |
| 3.7 | Front End | 27 |
| 3.8 | Iterations | 27 |
| | 3.8.1 Iteration 1 | 29 |
| | 3.8.2 Iteration 2 | 29 |
| | 3.8.3 Iteration 3 | 30 |
| 3.9 | Portability and Maintainability | 31 |
| 3.10 | Issues | 32 |
| 3.11 | Discussion | 32 |
| 4 | Evaluation | 34 |
| 4.1 | Introduction | 34 |
| 4.2 | Segmentation | 36 |
| | 4.2.1 Experiment: Segmentation of words in an image | 36 |
| 4.3 | Features | 37 |
| | 4.3.1 Experiment: Using features to prune results | 37 |
| | 4.3.2 Experiment: Introducing variation into feature correspondence | 38 |
| | 4.3.3 Experiment: Evaluating the speed of feature based matching | 41 |
| 4.4 | Accurate Matching | 42 |
| | 4.4.1 Experiment: Evaluating the accuracy of each matching algorithm | 42 |
| | 4.4.2 Experiment: Evaluating the speed of each matching algorithm | 44 |
| 4.5 | End-to-end Testing | 44 |
| | 4.5.1 Experiment: Testing the optimal values | 45 |
| | 4.5.2 Experiment: Changing the matching algorithm | 46 |
| | 4.5.3 Experiment: Increasing variation | 47 |
| | 4.5.4 Experiment: Increasing scale and accuracy | 47 |
| | 4.5.5 Experiment: Increasing scale and performance | 47 |
| 4.6 | User Testing | 50 |
| | 4.6.1 Experiment: User usability and usefulness | 50 |
| 4.7 | Summary | 51 |
| 5 | Future Work | 52 |
| 5.1 | Continuous Improvements | 52 |
| 5.2 | Additional Testing | 52 |
| 5.3 | User Interface | 52 |
| 5.4 | Implementation of New Features | 52 |

| | | |
|----------|--|-----------|
| 5.5 | User Feedback | 53 |
| 5.6 | Word Spotting Index | 53 |
| 5.7 | Configurable System | 53 |
| 6 | Conclusions | 54 |
| | Appendices | A1 |
| A | Features and Variation | A1 |
| B | Accuracy of Matching Algorithms | B1 |
| C | User Testing Questionnaire and Feedback | C1 |

List of Figures

| | | |
|------|--|----|
| 1.1 | Overview of the BOLDProject framework | 3 |
| 1.2 | Overview of the BOLD Translator | 5 |
| 1.3 | A page from one of Wilhelm Bleek’s notebooks | 7 |
| 1.4 | A dictionary page | 8 |
| 2.1 | Original image for word spotting | 14 |
| 2.2 | Projection profile for word spotting | 14 |
| 2.3 | Upper profile for word spotting | 15 |
| 2.4 | Background-foreground transitions for word spotting | 15 |
| 2.5 | Overlapping and touching in handwritten text | 17 |
| 3.1 | A high resolution TIFF with the AOI highlighted | 19 |
| 3.2 | Results of the preprocessing | 20 |
| 3.3 | Output of the preprocessor | 21 |
| 3.4 | Workings of the segmentation algorithm | 23 |
| 3.5 | Features used in the BOLD Translator | 24 |
| 3.6 | Workings of XOR matching algorithm | 26 |
| 3.7 | BOLD Translator front ends | 28 |
| 3.8 | Iteration 1 screenshots | 30 |
| 3.9 | Iteration 2 screenshot | 31 |
| 3.10 | Iteration 3 screenshot | 32 |
| 4.1 | Search keys used in experiments | 35 |
| 4.2 | Search key selections | 35 |
| 4.3 | Analysis of different underlying line lengths for word segmentation | 37 |
| 4.4 | Recall for different feature weightings | 39 |
| 4.5 | Results of introducing variation into feature matching | 40 |
| 4.6 | Effect of dataset size on on feature performance | 41 |
| 4.7 | Comparison of matching algorithms for three selections of each key | 43 |
| 4.8 | Matching algorithms performance measures | 45 |
| 4.9 | Precision, recall and F-score for end-to-end testing using optimal values | 46 |
| 4.10 | Comparison of results using optimal values and optimal values with increased variation | 48 |
| 4.11 | Precision, recall and F-score as scale increases | 49 |
| 4.12 | Performance as scale increases | 49 |

Chapter 1

Introduction

The importance of cultural heritage preservation should not be underestimated, especially in a country like South Africa which has a rich and violent history. What is preserved today, will act as a reminder in the future of what has come to pass. Ancient wisdom provides us with new ways of looking at the world and provides us with insight into how some of the earliest people who walked the Earth lived and what they believed. To lose this cultural heritage would be to lose part of our history and part of who we are. It is therefore of utmost importance that we preserve it, for our sake and for the sake of future generations.

In his foreword to UNESCO's World Culture Report published in 2000, UNESCO Director-General Koichiro Matsuura writes that "UNESCO and its many partners have an urgent task in seeking ways of preserving the languages, customs, arts and crafts of the communities most vulnerable to sweeping change [UNESCO, 2000]". The |xam and !kun Bushman people of southern Africa, who are considered to be some of the first inhabitants of the Earth, are two such examples of these communities which are most vulnerable. Much of their wisdom, ancient knowledge, art, culture, customs and language have been lost due to the rapid onset of globalisation and western influence. Fortunately though, some of it has been preserved through a handwritten record of the |xam and !kun languages and exists in the form of the Lloyd and Bleek collection.

During the 1870s, Lucy Lloyd and Wilhelm Bleek recorded and translated the stories of a group of |xam and !kun men and women. These stories, along with art and dictionaries were preserved and have come to be known as the Lloyd and Bleek collection. The Lloyd and Bleek collection has become valuable as a source of insight into Bushman language and culture and has been recognised as a UNESCO Memory of the World Collection [Skotnes, 2009].

Testimony to the importance of cultural heritage preservation is the motto on the South African coat of arms, "*!ke e: /xarra //ke,*" which is written in the |xam language and literally means *diverse people unite* [Government, 2009] - without the records of Lucy Lloyd and Wilhelm Bleek, these words may have been lost forever.

1.1 The BOLD Project

Included in the Lloyd and Bleek collection is an English - |xam dictionary which is made up of over 14000 pages which contain English words and their corresponding |xam translations. These manuscripts could prove to be valuable to researchers and scholars worldwide, however, as with any historical manuscripts, physical handling significantly contributes to the degradation of the manuscripts and physical access to the manuscripts is a limiting factor. For these reasons,

digitising the collection and making it publicly accessible are important for preservation and ease of access.

The Bushman OnLine Dictionary (BOLD) Project is an attempt at building a digital repository for managing and interacting with the English - |xam dictionary (from now on referred to as the BOLD Dictionary). Important considerations in the building of this repository are:

- The ease with which administrators are able to manage the repository.
- The ability of users world wide to access and interact with the repository.
- The long term preservation of the BOLD Dictionary.
- The creation of a framework which can be adapted to other collections of a similar nature.
- The interaction and integration between the BOLD Dictionary and other digital collections.

1.1.1 Motivation

The Lloyd and Bleek collection suffers from the real risk of physical degradation and needs to be digitised to ensure long term preservation and ease of access. However, due to the rapid onset of globalisation, other physical collections of cultural heritage worldwide are also in danger of being lost forever. There are existing systems for digitising and managing cultural heritage collections, however, none of them are built specifically with the goal in mind of being able to store and interact with a large collection in the form of a visual dictionary - such as the BOLD Dictionary. This creates the need for this type of cultural heritage preservation system and is a need which the BOLD Project seeks to meet.

1.1.2 The Framework

The BOLD Project framework is made up of three distinct components which collectively seek to create a framework which can be adapted to other collections of a similar nature. The three distinct components which make up the BOLD framework are: the core digital repository, the search and browse interfaces and the image-based translation tool. A high level view of the framework is shown in Figure 1.1 and is discussed here in a bit more detail.

The Core Digital Repository

The core digital repository of the BOLD Project framework is responsible for the storage of the dictionary images as well as the administration of the collection. This component of the framework was implemented by Lebogang Molwantoa using the Fedora Commons Repository Software and administrators of the digital repository are able to browse, add and delete objects in the repository. This component of the framework seeks to meet the requirements of ease of use by administrators and long term preservation of the English - |xam dictionary.

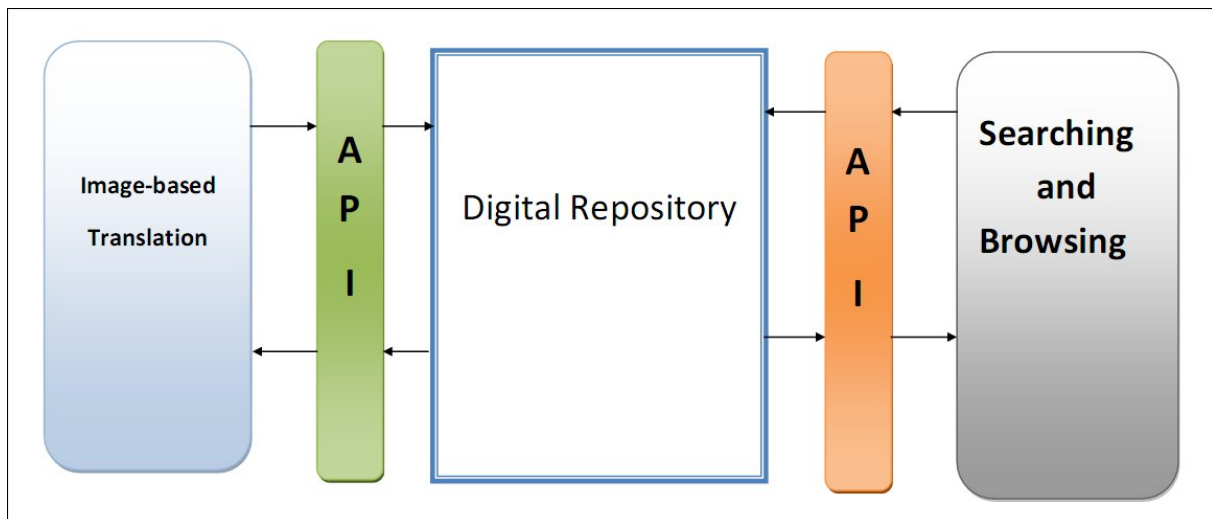


Figure 1.1: Overview of the BOLDProject framework

The Search and Browse Interfaces

The search and browse interfaces of the BOLD Project framework provide the means by which end users interact with the digital repository and were implemented by Sanvir Manilal. The search and browse interfaces are able to:

- Search by single or scrollable words.
- Create a scrollable list of words to aid browsing.
- Facilitate thumbnail browsing of envelopes.
- Use an online dictionary Web-service to retrieve word definitions.
- Create a history of words searched per session.
- Attempt to correct misspelled words.
- Incorporate an AJAX live search when searching.
- Query <http://lloydbleekcollection.cs.uct.ac.za> for links to drawings and notebooks related to searched words.

This component of the framework seeks to meet the requirements of allowing end users world wide to access and interact with the repository.

The Image-Based Translation Tool

The BOLD Translator is an image-based translation tool which attempts to meet the requirement of allowing for the interaction and integration between the BOLD Dictionary and other collections. It does this by providing a tool which allows users to select words from the notebooks containing |xam stories in the Lloyd and Bleek Collection and, using information retrieval and image processing techniques, performs image-based translation by matching the words selected from the notebooks with their corresponding entries in the BOLD Dictionary. This report will focus on the BOLD Translator as a component of the BOLD framework.

1.2 The BOLD Translator

1.2.1 Problem Statement

The current Lloyd and Bleek collection consists of notebooks which contain stories in the |xam and !kun Bushman languages. In some of these notebooks there are English translations alongside the Bushman words but in many cases those translations do not exist or are unclear. Given that the BOLD Dictionary contains images which have both an English word as well as its |xam translation it is natural to assume that people may want to make use of the BOLD dictionary to translate words which appear in the Lloyd and Bleek collection. The problem exists, however, that it is not practical to do this by hand simply due to the size of the BOLD dictionary.

1.2.2 Motivation

The motivation for the BOLD Translator is to solve the problems identified by the problem statement by making the translation of Wilhelm Bleek's notebooks a practical undertaking. The BOLD Translator has the potential to be of assistance to researchers who are studying Bushman language and culture.

1.2.3 Research Question

The research question the BOLD Translator seeks to address within the greater context of the BOLD Project is presented as follows:

Can image-based searching be done accurately and efficiently?

In answering this research question, the time taken in conducting searches and the correctness of searches, as measured by information retrieval measures, will be investigated.

1.2.4 The Framework

The BOLD Translator attempts to overcome the lack of practicability in doing manual translations 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. In this sense the BOLD Translator is a content based image retrieval (CBIR) system. Content Based Image Retrieval systems are systems for retrieving images from a collection based on the similarity of images in the collection to another image which is known as the search key [Rui et al., 1999]. More specifically, the BOLD Translator is a framework for CBIR systems for handwritten texts and thus is described here at a high level such that the framework can be implemented using any collection. Figure 1.2 shows an overview of the BOLDTranslator framework.

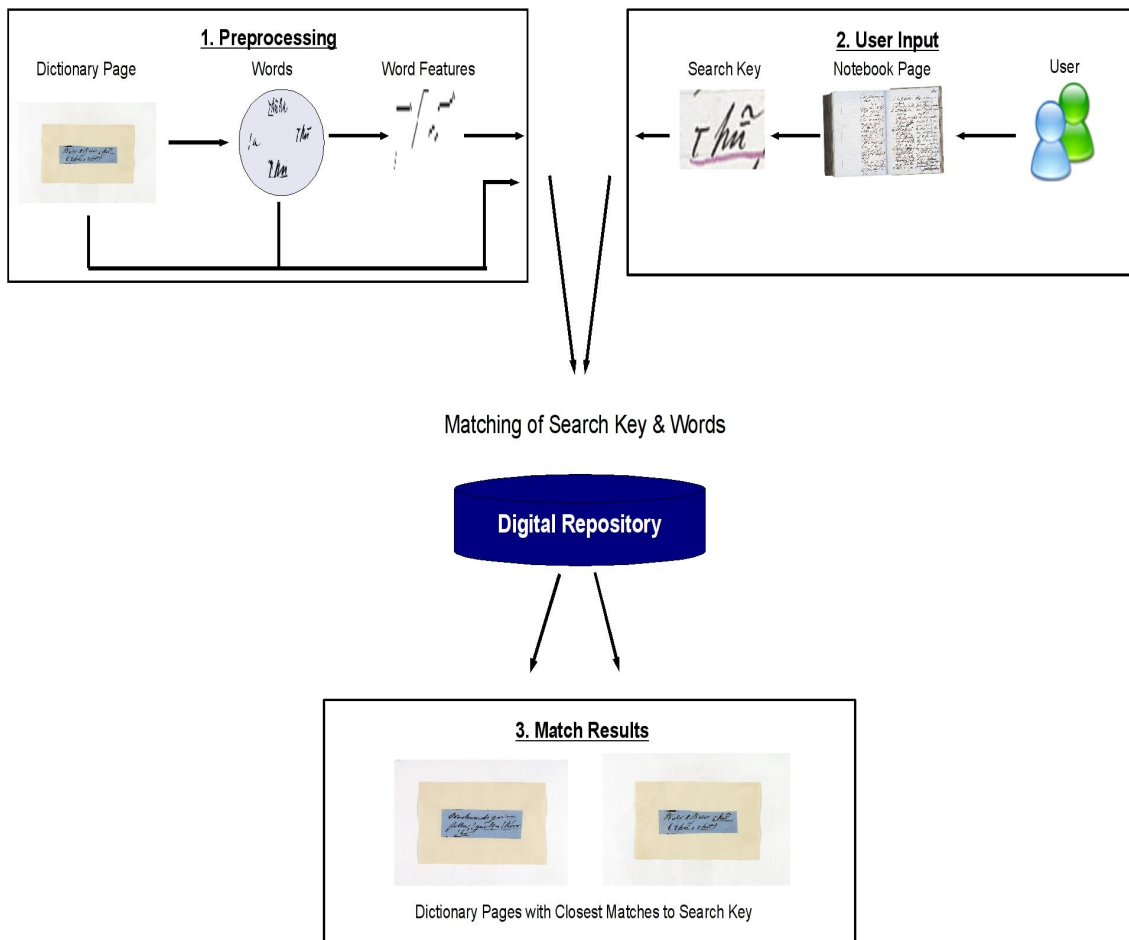


Figure 1.2: Overview of the BOLD Translator

Preprocessor

The following operations are performed on every image which is added to the repository:

1. The image is cleaned by smoothing to remove noise such that processing on it can take place.
2. Words in the image are segmented.
3. For each segmented word, a set of known features are extracted and stored in inverted files.
4. The original images, segmented words and inverted files are stored in the digital repository.

User input

User input involves the user selecting a search key which should be matched against the repository:

1. The user selects a search key.
2. The same set of features as extracted by the preprocessor are extracted from the search key.
3. The search key features and the search key image are compared to the contents of the digital repository.

Matcher

The matcher compares the search key features and the search key to the contents of the repository and returns the closest matches:

1. Inverted files are searched for words which have features that correspond to the features of the search key and each word is given a feature score.
2. Words which have a feature score above some threshold have more accurate matching performed on them.
3. The best matches are returned to the user as results.

1.2.5 Data Description

There are two sources of data which are relevant to the BOLD Translator. The first source of data is images of Wilhelm Bleek notebooks which contain Bushman stories (Figure 1.3). The second source of data is the images of the collection of dictionary pages which belong to the BOLD Dictionary (Figure 1.4). The majority of pages which appear in the BOLD Dictionary are written by Wilhelm Bleek and for this reason the BOLD Translator focuses on translating Wilhelm Bleek's notebooks and not Lucy Lloyd's. Wilhelm Bleek's notebooks serve as the location from which search keys are selected with matches of words being found in the BOLD Dictionary. In most cases, the physical objects in both Wilhelm Bleek's notebooks and the BOLD

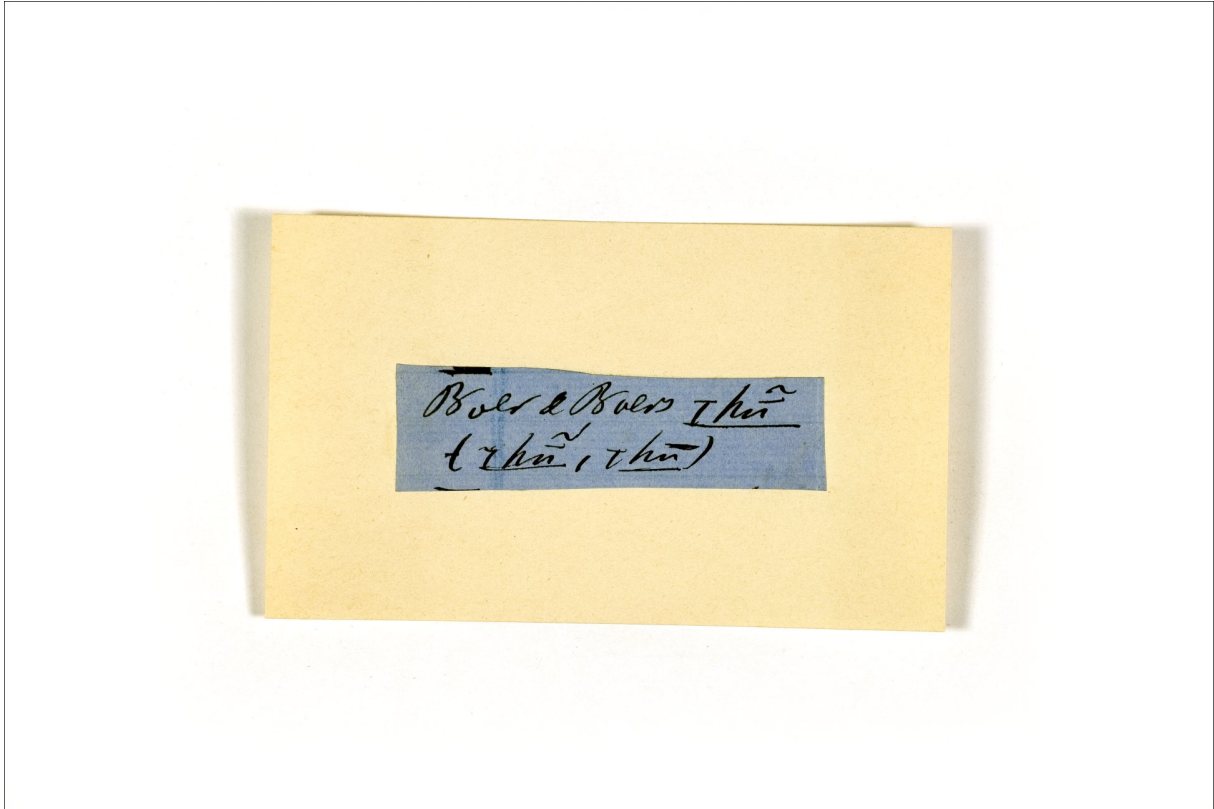


Figure 1.4: A dictionary page

1.3 Summary of Report

This report gives an overview of the BOLD Translator and starts by investigating similar work which has been done. Thereafter the design and implementation of the BOLD Translator is explained in greater detail. Evaluation of the system is carried out using various different experimental methods. Lastly, conclusions are made about the BOLD Translator and future work is noted.

Chapter 2

Background

2.1 Introduction

Much work has been done in the field of digital cultural heritage preservation with a number of museums and libraries curating large datasets with the goals of long term preservation and ease of access. In recent years there has also been exploration into Content Based Image Retrieval (CBIR) systems with varying degrees of success. This chapter sets out to set up a background to this project by exploring previous attempts at large scale digital cultural preservation with special emphasis on previous digital curation of the Lloyd and Bleek collection. Thereafter, previous work done on CBIR systems is investigated and a technique called word spotting for grouping similar images of words is explored. Lastly, previous work done in word segmentation - which is a requirement for this project - is explored.

2.2 The Lloyd and Bleek Collection

The Lloyd and Bleek collection is an archive of narratives, drawings and documents of and by the !xam and !kun people of southern Africa. Collected in the 19th century by Lucy Lloyd and Wilhelm Bleek, it is unique as an archive of indigenous life and thought, and has been recognised as a UNESCO Memory of the World Collection [Skotnes, 2009]. Of particular interest to this project is work done by Suleman [2007] in which a usable digital archive of 14128 images, from a total of 157 notebooks from the Lloyd and Bleek Collection was created. An XML-centric solution was chosen for this archive because it:

1. Required no installation from the end user.
2. Was platform independent.
3. Allowed for easier processing.
4. Had long term preservation benefits.

Suleman was able to show that the XML-centric approach did have many advantages over traditional database-based archives. However, scalability issues were identified as a limiting factor. The resulting product of the work done by Suleman is available online [Skotnes, 2009] and in book and DVD format [Skotnes, 2007]. Suleman's work is of particular interest because it forms the basis for part of this project - interaction between the existing Lloyd and Bleek collection and a new collection of dictionary images.

2.3 Preserving Cultural Heritage

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. 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 [Doumat et al., 2008]. These systems for managing digital collections have been used in a wide variety of projects for preserving cultural heritage [Antonacopoulos and Karatzas, 2004, Michael, 2009, Nicolakis et al., 2003].

2.3.1 The MEMORIAL Project

The MEMORIAL Project is a project funded by the European Union and undertaken by a multinational consortium. The goal of the MEMORIAL project is to enable the virtual distribution of paper-based archives which are currently held at museums and libraries [Antonacopoulos and Karatzas, 2004]. The official title of the project is: “A Digital Document Workbench for Preservation of Personal Records in Virtual Memorials,” which Antonacopoulos and Karatzas note suggests that the project’s focus is on information about people.

Antonacopoulos and Karatzas document the use of the MEMORIAL project to create a digital collection of World War II personal records which contain information about people at Nazi-run concentration camps. They discuss how the MEMORIAL project framework was used for document input, image analysis, optical character recognition (OCR) and the creation of a Web-based portal which would allow users to access and make use of the content. In their concluding remarks, they note that the MEMORIAL project was still underway, however the project’s website appears to have been taken down. Regardless of this fact, the MEMORIAL project is/was a good example of making use of digital collections to preserve cultural heritage and make it usable and accessible.

2.3.2 The MICHAEL Project

The Multilingual Inventory of Cultural Heritage in Europe (MICHAEL) 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 which will allow people to find and explore digital European cultural heritage on the Internet [Michael, 2009].

2.3.3 Greek Orthodox Archdiocese of America

The Greek Orthodox Archdiocese of America (GOA) undertook a project to digitise their large collection of religious and historical artifacts. As is the case with most digital collections of cultural heritage, the goal of the project was to preserve the artifacts as well as make them accessible by users [Nicolakis et al., 2003]. The GOA archive however was significantly different from other digital collections of cultural heritage in that the GOA had the need to make use of digital rights management (DRM) in order to protect and control access to the archive [Nicolakis et al., 2003].

2.3.4 Summary

There are a number of systems for digital preservation - three of which have been briefly discussed here. Annotations of digital objects in digital preservation systems is important to allow for the retrieval of these objects and is therefore an important consideration when designing a digital preservation system. It is evident that digital cultural heritage preservation is considered to be important by governments as well as private organisations.

2.4 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 [Rui et al., 1999]. 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 [Rui et al., 1999]. In this sense, images are used to search for other images in CBIR systems.

Information retrieval is a well studied discipline within the field of digital libraries and there are a number of international conferences, such as the Text REtrieval Conference (TREC) [TREC, 2009], which focus on issues surrounding information retrieval. CBIR systems, however, are relatively new within the larger context of information retrieval systems and have not been studied to the same extent as other information retrieval systems such as text retrieval systems. In this section the background to CBIR systems is explored and real world examples of CBIR systems are given.

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 meaning of images [Rui et al., 1999].

There are generally three types of CBIR: primitive queries (query by example), semantic retrieval and automatic retrieval [Eakins and Graham, 1999]. Primitive queries, or queries by example, are the most common types of queries in CBIR and therefore they are the only ones which will be discussed in full detail here. However, for completeness, the others will be discussed briefly.

In semantic retrieval, images are analysed and a set of possible interpretations are derived, each having some probability of being the correct meaning. Further semantic retrieval involves user feedback, which allows the system to learn about primitive features based on semantic concepts. There are few real life examples of automatic retrieval, however, one of its applications involves a CBIR system analysing colour in an image, determining whether the colours are “hot” or “cold”, and returning images which convey a similar “mood” or “feeling” [Eakins and Graham, 1999].

In primitive CBIR, images are analysed based on a number of primitive features, most notably colour, texture, shape and colour layout. This analysis usually takes place on segmented parts of the image, as it has been shown that the shape and colour layout analyses depend on good segmentation [Rui et al., 1999].

Once these features have been extracted from an image, it is possible to construct a signature for the image which can then be compared to signatures of other images in order to find matches.

2.4.1 Segmentation

In CBIR, images are often divided into parts, and then the features of each part are analysed separately. The goal of segmentation is to have more selective features in pixels, rather than more information about the image as a whole [Smeulders et al., 2000]. Smeulders et al. differentiate between four main types of segmentation:

- Strong segmentation, in which a segment contains the pixels of an object in the real world and nothing else.
- Weak segmentation, in which segments contain data which is homogeneous according to some criterion.
- Wigning, in which an object has a nearly fixed shape and a semantic meaning.
- Partitioning, in which a partition is simply a division of the data array.

As noted by Smeulders et al., the different types of segmentation allow for different features to be extracted from an image.

2.4.2 Colour

Colour is one of the most widely used features in CBIR, as it is relatively robust to background complication and independent of image size and orientation. In CBIR, the colour histogram is the most commonly used colour feature representation, as it shows the intensity of each of the three main colour channels [Rui et al., 1999]. These histograms allow for images to be compared, based on the similarity of their colour distributions. The colour histogram does, however, have one significant shortcoming in that it only shows the colour distribution of an image. Due to this, it could compute similar values for very different images if their colour distributions are similar [Rasheed et al., 2008]. Rasheed et al. suggest an alternative in the form of a colour correlogram, which not only shows the intensity of the colour distributions, but also the spatial information of pixels in the image. The approach of Rasheed et al. allows for significantly more meaning to be extracted from the colour in an image than the traditional colour histogram approach.

2.4.3 Texture

In CBIR, texture refers to the repetitive patterns which appear on the surfaces of images [Rui et al., 1999, Datta et al., 2008]. Textures provide a further basis of comparison for two images, as the images can be compared based on the features of their textures. Textures are often domain specific, such as the textures of aerial imagery and medical imagery. It has been shown that texture features can be extracted by a variety of methods such as the transformation of the original pixels of an image, or wavelet transformations [Datta et al., 2008].

2.4.4 Shape

Shape is another basis upon which two images can be compared. A large variety of ways of detecting shapes for comparison have been suggested. Examples are: making use of salient edges [Han and Guo, 2003]; Fourier descriptors [Zhang and Lu, 2002]; and making use of image properties which are indirectly related to shapes, rather than the shapes themselves [Gagaudakis and Rosin, 2002].

2.4.5 Signatures

The signature of an image is created based on its feature. Datta et al. [2008] identify three main types of signatures:

1. A feature vector in which a single vector is used to describe the whole image (global).
2. A region-based signature in which each region is described by a separate vector (local).
3. A summary of local feature vectors.

A local signature represents the specific details of an image and a global signature represents an image's "bigger picture" [Datta et al., 2008]. According to Datta et al., in recent years there has been a shift from global signatures towards local signatures. The reason for this shift is based on the growing understanding that local features often correspond with more meaningful aspects of an image, and these more meaningful aspects are more useful in CBIR.

2.4.6 Comparing Signatures

Signatures are compared based on some distance formula. The smaller the distance between images, the more similar they are. In this sense, images can be matched based on the similarity of their signatures. Datta et al. note that it becomes significantly more difficult to match signatures when region based (local) signatures have been used, due to the complexity of calculating the distances between the set of vectors.

2.4.7 CBIR Systems

- **retrievr & imgSeek**

retrievr [Langreiter, 2009] and imgSeek [Cabral, 2009] are two image-based search engines based on the fast multiresolution image querying algorithm developed by Jacobs et al. [1995]. 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! [Flickr!, 2009] 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 - that is, if there were 100 images in the sample, they were able to find the correct image almost every time.

- **IBM's Query By Image Content System**

IBM's Query By Image Content (QBIC) system is another CBIR system which performs searches based on the primitive features of an input image [IBM, 2009]. The system enhances the quality of its searches by allowing the user to provide key words in addition to the image used for searching. IBM has identified several real world uses of the QBIC system such as using it to search clothing catalogues to find certain styles of clothes.

2.4.8 Summary

CBIR systems are becoming increasingly important in the field of information retrieval as the need for new means of retrieving information increases. This is especially true if one considers the large amount of multimedia which is available on the internet and in multimedia databases such as Flickr!. Previous work done on CBIR systems has been explored and the need for CBIR systems has been discussed as well as the ways in which they are traditionally implemented. Three CBIR systems - retrievr, imgSeek and IBM's QBIC system - have been given as examples of real world CBIR systems which are currently available for public use.

2.5 Word Spotting

Word spotting is a technique for grouping occurrences of the same word, where it exists in multiple locations in a document. It is based on the user providing a key word, and then image matching is done to try and find other occurrences of that word in order to build an index [Rath and Manmatha, 2007]. Word spotting systems are different from CBIR systems in that they try to determine the similarity of all images in a collection, whereas CBIR systems try to retrieve similar images from a collection based on a key image. Word spotting is relevant to the BOLD Translator, however, in the sense that it particularly deals with words in handwritten manuscripts.

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 [Leydier et al., 2007]. Two implementations of word spotting systems are discussed.

2.5.1 Word Spotting for the George Washington Collection

A word spotting system developed by Rath and Manmatha [2007] showed that word spotting can be successfully applied to historical documents, by demonstrating its use on the George Washington collection.

It was shown by Rath and Manmatha that a word's profile can be created based on a few key features.

Figure 2.1 shows the original word image Rath and Manmatha were working with.

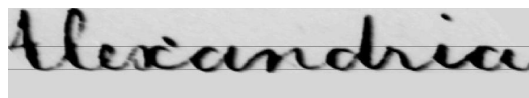


Figure 2.1: Original image for word spotting

The first feature captured by Rath and Manmatha was the distribution of ink along one of the two dimensions of an image, referred to as the projection profile. Figure 2.2 shows the projection profile image created by Rath and Manmatha .



Figure 2.2: Projection profile for word spotting

The next features captured were the upper and lower profiles (shapes) of a word. The upper profile image created by Rath and Manmatha are shown in Figure 2.3. These features were captured by applying background-foreground separation techniques.

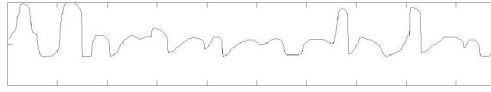


Figure 2.3: Upper profile for word spotting

The last feature extracted from a word was information about its “inner” content. The inner content images created by Rath and Manmatha are shown in Figure 2.4. This information was derived based on the number of background to text transitions which took place in an image.

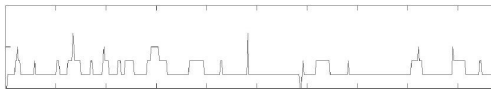


Figure 2.4: Background-foreground transitions for word spotting

The information extracted by Rath and Manmatha gave them three key pieces of information about a word:

1. The amount of ink used in an word.
2. The upper profile and lower profile of a word.
3. An idea about the inner structure of a word.

Armed with this information, they apply a dynamic time warping (DTW) algorithm for image matching. DTW is used because it allows for writing variation as it makes use of a common time axis. It is shown that DTW greatly outperforms other distance measuring techniques such as Euclidean Distance Matching.

Furthermore, Rath and Manmatha clearly show that word spotting is a viable and practical method for matching images containing handwritten text.

2.5.2 Word Spotting for Medieval Manuscripts

Leydier et al. [2007] show the application of word spotting to medieval manuscript images. The key idea revealed by the work done by Leydier et al. is that of domain specific word spotting, that is, they base the feature vector for each word on specific attributes of written medieval words. Their proposed approach involves no layout segmentation, no binarisation of images, and tolerates low quality and distorted images. Instead of focusing on the greyscale transformation of an image, the focus is instead placed on informative parts of an image, which are the most discriminant parts of an image and referred to as zones of interest (ZOIs). They found that the orientation of the gradients of strokes at these ZOIs is the most efficient description of a word’s shape. They furthermore showed that focusing on other features gave poor results.

2.5.3 Summary

It has been demonstrated that word spotting is a viable system for matching similar words in a collection. While the BOLD Translator is not a word spotting system, it does have many

similarities with word spotting systems in that it also requires the matching of similar handwritten words in historical manuscripts. The DTW matching technique implemented by Rath and Manmatha was implemented in the BOLD Translator as a matching algorithm and is described in further detail in section 3.6.5

2.6 Word Segmentation

Word segmentation is a prerequisite for any system which attempts to compare words which appear on a page, as is the case when using word spotting. There have been several attempts at performing segmentation of words which appear on a page. However, Manmatha and Srimal [1999] note that most systems which do this have been developed for machine printed text and that there is a lack of 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. In this section three techniques for the segmentation of handwritten words will be explored.

2.6.1 Line Segmentation

Line segmentation is one approach to word segmentation which works by segmenting an image containing words into lines. While not a complete word segmentation technique it is often a prerequisite for other 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 [Likforman-Sulem et al., 2007].

There are two important considerations when doing automatic line segmentation: overlapping and touching components. Overlapping occurs when a character from one line extends into another line and touching occurs when characters from two different lines touch (Figure 2.5). Line segmentation techniques need to be able to deal with overlapping, touching, as well as degraded quality which is a characteristic of historical documents [Likforman-Sulem et al., 2007]. For a complete discussion on line segmentation please refer to Likforman-Sulem et al. [2007].

2.6.2 Gap Metrics

Louloudis et al. [2009] propose an algorithm for word segmentation which makes use of the gap metrics between words. The algorithm takes a single line of words, such as those which are the output of line segmentation techniques, and then attempts to segment the words in that line. The problem is presented as a two-class classification problem to determine, given a distance (or gap) between two components, whether that gap is an inter-word space. Using this technique Louloudis et al. were able to achieve a segmentation success rate of 90.82%.

2.6.3 Scale Space

Manmatha and Srimal [1999] suggest a scale space technique for word segmentation which works by “analyzing the extent of ‘blobs’ in a scale space representation of an image.” The technique is built around scale based theory which deals with the importance of scale in observation. Manmatha and Srimal argue that in terms of document analysis characteristics can be thought

Chapter 3

Design and Implementation

3.1 Overview

The goal in the implementation of the BOLD Translator is to build a CBIR framework for handwritten words which requires no training of the system using a dataset. The omission of the requirement to train the system using a dataset allows for the framework to remain as general as possible. This chapter discusses the design and implementation of the BOLD Translator. The technology and tools used during the implementation of the BOLD Translator are detailed, then a high level design overview is given, followed by a more detailed low level design view. Lastly, the iterative implementation of the BOLD Translator is explained and results from each of the iterations are provided.

3.2 Technology and Tools

The majority of the BOLD Translator was written in C++ with the front end being written in PHP. C++ was chosen as the primary language for the BOLD Translator because of the importance of performance in the BOLD Translator. The front end to the BOLD Translator is trivial and can be easily re-implemented in any language.

The BOLD Translator makes extensive use of ImageMagick and Magick++ [ImageMagick, 2009]. ImageMagick is a software suite which is used to “create, edit and compose bitmap images.” It supports a wide variety of image formats and was used for much of the data preprocessing as well as regular processing. Magick++ is an object oriented C++ interface to ImageMagick. More information on ImageMagick and Magick++ can be found on the ImageMagick website [ImageMagick, 2009].

The Yahoo! User Interface (YUI) Library was used for the cropping tool which forms part of the interface and is available online at [Yahoo!, 2009].

The design of the BOLD Translator took place in three iterations with the first iteration being a proof of concept, the second implementation being a preliminary implementation and the third implementation closely representing the final product.

3.3 Image Preprocessing

The high resolution TIFF images which belong to the BOLD Dictionary contain large areas of space which are not necessary for the BOLD Translator. 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 Bushman words appear. In some cases Bushman words appear on the brown envelope sheet. In every high resolution TIFF there is a specific area of interest (AOI) which is the rectangular area of the TIFF which contains all the handwritten words. Figure 3.1 shows one of the original TIFFs with the large whitespace, the brown envelope and the AOI labeled. It is the goal of the preprocessor to identify the AOI in every image, crop the image around the AOI and provide as output a thresholded AOI which will be used for segmentation, feature extraction and matching.

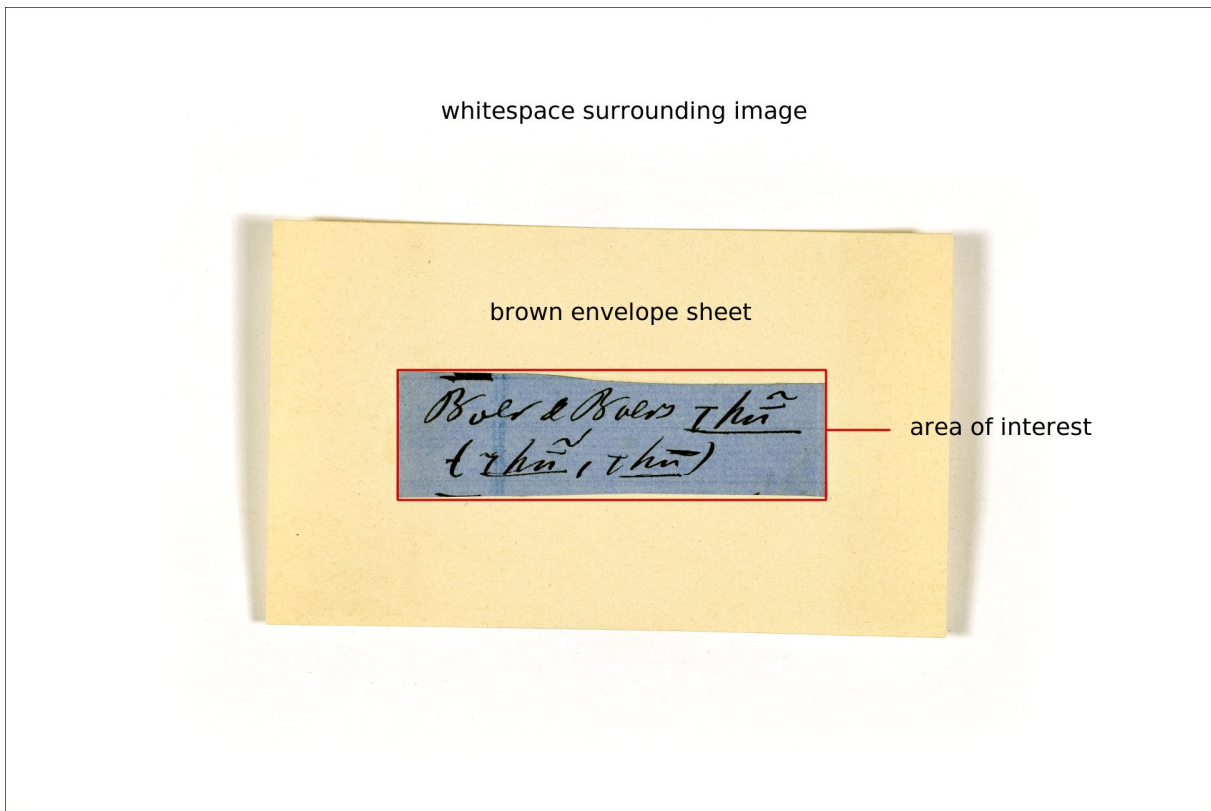


Figure 3.1: A high resolution TIFF with the AOI highlighted

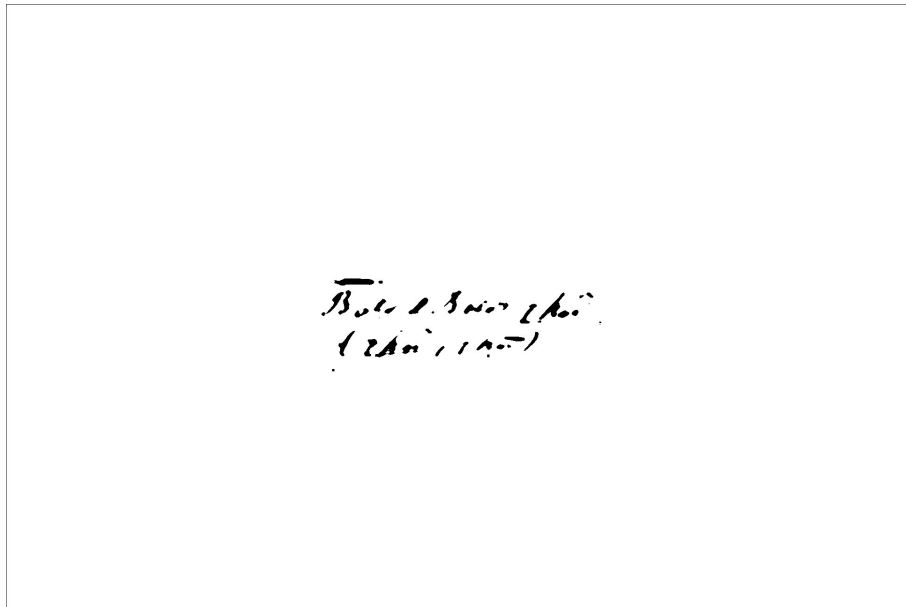
ImageMagick is used to crop the AOI in every image as follows:

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 which appears anywhere on the page.
2. The image is smoothed using a median filter in order to smooth away any black marks which might be isolated on the page and which are most likely noise.
3. The white area surrounding the image is trimmed, leaving only the area containing the handwriting.

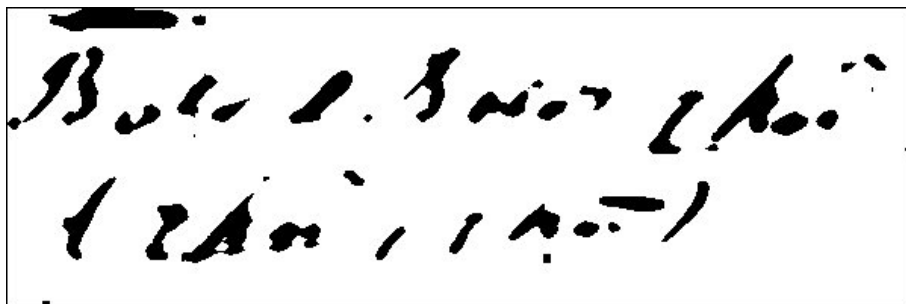
Figure 3.2 shows the results of the thresholding, smoothing and trimming.



(a) Thresholded image



(b) Smoothed image



(c) Trimmed image

Figure 3.2: Results of the preprocessing

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 where the AOI is (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 3.3 shows the AOI without any loss of information as well as the final output image from preprocessing.

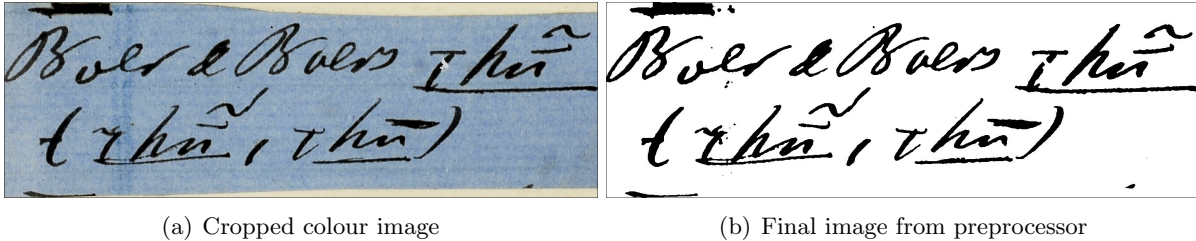


Figure 3.3: Output of the preprocessor

3.4 Segmentation

The role of segmentation is to - as accurately as possible - cut out the Bushman words which appear in each of the thresholded AOIs provided by the preprocessor. An important issue that needs to be considered is that AOIs contain both English as well as Bushman words. This creates the natural problem that by segmenting all the words in AOIs, non-Bushman words will be introduced into the dataset which has a negative effect on the system in that:

1. It increases the size of the dataset, resulting in performance issues.
2. It adds additional images which could be candidate matches and thus has a negative effect on accuracy.

The segmenter therefore works by exploiting known information about the collection and that is 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. The accuracy of the segmenter is evaluated in section 4.2.1.

3.4.1 Identifying Underlining Lines

Connected component analysis is used to identify the underlying lines. The algorithm works as follows:

1. For each pixel in an image, if the pixel is black create a candidate line.
2. If the adjacent pixel or either of the diagonal pixels is black, increase the length of the candidate line - continue doing this until a white pixel is found.
3. If the candidate line is at least as long $x\%$ of the image size, classify it as a line underlying a word, otherwise, discard it.

An important consideration is what length the underlying line length should be. In terms of precision and recall, if the required underlying line length is short then recall will likely be high since most underlying lines will be identified, however at the same time precision is likely to be low since lines which do not underline words will also be identified. On the other hand, if the required underlying line length is high then precision will likely be high while recall is likely to be low. The best underlying line length for this dataset is investigated in section 4.2.1.

3.4.2 Identifying Word End Points

Having detected the underlying lines for Bushman words the next step involves detecting the left and right ends of those words. In some cases the whole word is covered by the underlying line but in other cases the underlying line may fall short on either side. The algorithm works by finding the left and right end points of words as follows:

1. From the left end of the underlying line, project a vertical line 25 pixels up.
2. If any of those 25 pixels is black, then move the end one pixel to the left and project the vertical line again.
3. Keep doing this until there are no black pixels falling on the vertical line - this is then classified as the space between two words and the left end of the word being considered.
4. Do the same thing for the right end of the underlying line in order to find the right end of the word.

3.4.3 Identifying the Top of the Word

Having identified the underlying line as well as the left and right ends of a word, the last step in segmentation involves finding the top of the word. The algorithm works as follows:

1. Project a horizontal line equal to the size of the underlying line length 25 rows above the underlying line and count the number of black pixels.
2. Move the horizontal projection one row up and count the number of black pixels again - if this number is smaller than the previous number, then set this as the location of the minimum number of black pixels.
3. Continue moving one row up and keep track of where the minimum number of black pixels occurred until the number of black pixels in the current line exceeds the minimum by 5 - this is most likely the start of the next word.
4. Set the row for the top of the word to the location where the minimum number of black pixels occurred.

Having found the underlying line, the left and right end points and the top of the word a box can be drawn around the word and it can be segmented using this box as is shown below:

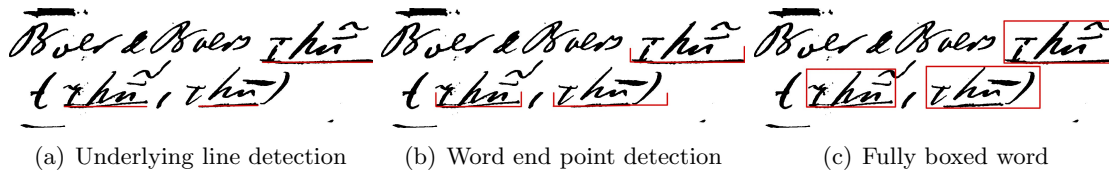


Figure 3.4: Workings of the segmentation algorithm

3.5 Feature Extraction

Features play a significant role in the BOLD Translator as they are used to prune a large dataset such that accurate matching can be performed on images which have similar characteristics as the key image. Once the words in an image have been segmented, features are then extracted from and stored in inverted files which are used to lookup words which have similar features when searching takes place. It is difficult to detect features perfectly due to the quality of the original manuscripts and inconsistencies in handwriting. For this reason, feature variation has been introduced into the system and is discussed in section 3.6.3.

3.5.1 Features Used in the BOLD Translator

The following features are used in the BOLD Translator: intersections, horizontal lines, long vertical lines and short vertical lines. With the exception of intersections, all the features are identified using connected component analysis.

Intersections

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

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 which are not necessarily horizontal lines in characters from being detected.

Long Vertical Lines

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 which are at least as long as 65% of the height of the word image.

Short Vertical Lines

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

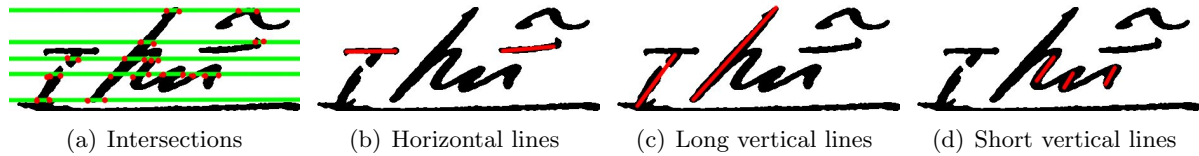


Figure 3.5: Features used in the BOLD Translator

3.5.2 Inverted Files

An inverted file is a file which contains a list of terms and for each term a list of corresponding document identifiers [Suleman, 2009]. An inverted file exists for each of the four features identified above and each inverted file contains all the possible values for that feature. After the features have been extracted from each word, the word identifier is added to the feature inverted files at the line where the feature value corresponds.

3.6 Matching

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

1. A key is selected by the user for matching and its features are extracted.
2. The user sets feature weights and variation allowance.
3. Based on these inputs, the inverted files are searched for words which have features which 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 which 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 which have a accurate matching score above some threshold t_2 are then displayed to the user.

Each of the steps involved in the matching process are described in more detail below.

3.6.1 Key Selection

Key selection is performed by the end user and involves selecting a word from one of the pages of the Lloyd and Bleek collection. Key selection is explored in more detail in section 3.7

3.6.2 Feature Weights

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 intersections and might result in better matches. Feature weights are set such that

$$\sum_{i=1}^4 W_i = 1 \quad (3.1)$$

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

$$Y = W_1 + W_2 + W_3 + W_4 \quad (3.2)$$

Various combinations of feature weights are investigated further in section 4.3.1.

3.6.3 Variation

As discussed in section 3.5 feature extraction is often not 100% accurate. Variation helps 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.

3.6.4 Feature Scores

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

$$Y = 0.1 + 0.3 + 0.3 + 0.3. \quad (3.3)$$

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 which has the highest feature score has a normalised feature score of 1. Words which 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.

3.6.5 Accurate Matching

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 which have an accurate matching score above some threshold t_2 are presented to the user as results. Four different accurate matching algorithms have been implemented with the first three all being variations of one another and the last one being completely different.

Difference Matching Algorithm

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 imposed on top of the key image and the sum of the absolute values of the differences between their corresponding pixels is calculated. This is repeated by 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.

XOR Matching Algorithm

Calculating the XOR of two images was mentioned by Manmatha and Croft [1997] 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 image such that the background becomes black and the foreground becomes white. Thereafter the algorithm works the same way as the difference algorithm, by resizing the images and imposing the word image on the key image as well as shifting the word image 1, 2, 3, 4, 5 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.

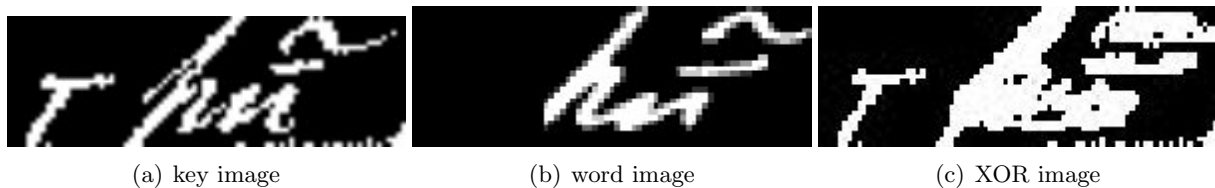


Figure 3.6: Workings of XOR matching algorithm

Euclidean Distance Matching Algorithm

The Euclidean Distance Matching (EDM) algorithm discussed by Manmatha and Croft [1997] works the same as the XOR matching algorithm, with the difference 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 which exists in a blob of white pixels will have a larger Euclidean distance 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.

Dynamic Time Warping Algorithm

Dynamic Time Warping (DTW) is a dynamic programming algorithm for computing the distance between two different time series and was proposed by Rath and Manmatha [2003] as a technique for word image matching. The DTW algorithm works by trying to find the optimal alignment between two signals and then calculates the distance between those two signals at that point. *Time warping* is a reference to the fact that the algorithm works by warping two signals such

that corresponding samples in the two signals appear in the same location on a common time axis. In this sense the DTW algorithm can account for variations in two occurrences of the same word where these variations can take on various forms such as spacing between characters, word length and slant.

For a word image I , feature vectors are extracted for each column c such that for each column a feature vector F_c exists where:

$$F_c = (f_1, f_2, \dots, f_n) \quad (3.4)$$

where $f_1 - f_n$ represent the n features for that column. The distance between two images a and b of length w is then calculated as:

$$D(a, b) = \sum_{i=1}^w (a(F_i) - b(F_i))^2 \quad (3.5)$$

The DTW distance between two images a and b is then presented as:

$$DTW = \min \begin{vmatrix} DTW(i-1, j) \\ DTW(i, j) \\ DTW(i, j-1) \end{vmatrix} + D(a, b) \quad (3.6)$$

The features used in this implementation of the DTW algorithm are the ones identified by Manmatha et al. and which are explored in section 2.5.

The four matching algorithms used in the BOLD Translator are evaluated for accuracy and performance in sections 4.4 and 4.4.2.

3.7 Front End

The front end to the BOLD Translator makes use of JavaScript for selecting search keys and PHP for displaying results. The front end was built to be simple to use though, due to time limitations and the project being more about information retrieval than usability, it was not designed according to any interface standards nor was any heuristic testing done on it. There are two front ends. The first one includes a page from one of Wilhelm Bleek's notebooks with a JavaScript selection tool for selecting a word from the notebook for translation, as well as text boxes and radio buttons for the user to set feature weights, variation, accurate matching algorithm and threshold levels. The second front end shows the results of the BOLD Translator and also allows the user to change the BOLD Translator parameters and re-conduct the search. Figure 3.7 shows screenshots of the two front ends.

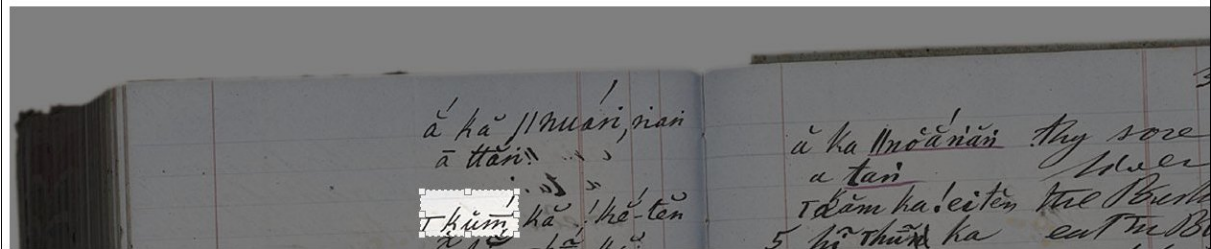
3.8 Iterations

The BOLD Translator was designed in three main iterations with the first iteration being a proof of concept, the second iteration being a preliminary implementation of the complete system and the third iteration being an almost complete system.

BOLDProject :: Translator

Highlight the word to translate and press the Translate! button

| | | |
|--|---|---|
| Intersections: | Weight: <input type="text" value="1"/> | Variation: <input type="text" value="0"/> |
| Horizontal Lines: | <input type="text" value="3"/> | <input type="text" value="1"/> |
| Long Vertical Lines: | <input type="text" value="3"/> | <input type="text" value="2"/> |
| Short Vertical Lines: | <input type="text" value="3"/> | <input type="text" value="2"/> |
| Threshold: | <input type="text" value="0"/> % | |
| Matcher: | <input checked="" type="radio"/> DIF <input type="radio"/> XOR <input type="radio"/> DTW <input type="radio"/> EDM <input type="radio"/> NONE | |
| <input type="button" value="Search!"/> | | |



(a) First front end for selection

Weight Variation

Intersections:

Horizontal Lines:

Long Vertical Lines:

Short Vertical Lines:

Width:

Matcher: DIF
 XOR
 DTW
 EDM

Pages

- [Boer \(Bleek\)](#)
- [Back \(Bleek\)](#)

BOLD Translator

Search Key: kūm

| | | |
|---|---|---|
| <p>1</p> <p>DICTIONARY_ENGLISH_B_0532.png Features: 9 Score: 43.6681</p> | <p>2</p> <p>DICTIONARY_ENGLISH_B_0659.png Features: 9 Score: 43.6681</p> | <p>3</p> <p>DICTIONARY_ENGLISH_B_0857.png Features: 9 Score: 42.0168</p> |
| <p>4</p> <p>DICTIONARY_ENGLISH_B_1183.png Features: 10 Score: 41.6667</p> | <p>5</p> <p>DICTIONARY_ENGLISH_B_0724.png Features: 10 Score: 40.4858</p> | <p>6</p> <p>DICTIONARY_ENGLISH_B_1117.png Features: 11 Score: 39.8406</p> |
| <p>7</p> <p>DICTIONARY_ENGLISH_B_0255.png Features: 9 Score: 39.6825</p> | <p>8</p> <p>DICTIONARY_ENGLISH_B_0867.png Features: 10 Score: 38.7597</p> | <p>9</p> <p>DICTIONARY_ENGLISH_B_0916.png Features: 9 Score: 37.3134</p> |

(b) Second front end for displaying results

Figure 3.7: BOLD Translator front ends

3.8.1 Iteration 1

The purpose of the first iteration was to show that each of the components which make up the BOLD Translator are technically viable. These three components are segmentation, feature extraction and accurate matching.

Segmentation

The first iteration did not perform any segmentation but showed that was possible to detect the spaces between words with the assumption that if it is possible to detect the spaces then it should be possible to segment words at those spaces. The first iteration showed the detection of horizontal white spaces between words.

Feature Extraction

The first iteration computed the upper profile feature of the key and each word image and showed that when comparing the upper profile of the search key and each of the word keys the most similar results were correct matches.

Accurate Matching

The first iteration performed the DIF matching algorithm on the full areas of interest and showed that it was possible to find correct matches this way.

Figure 3.8 shows screenshots from the first iteration.

3.8.2 Iteration 2

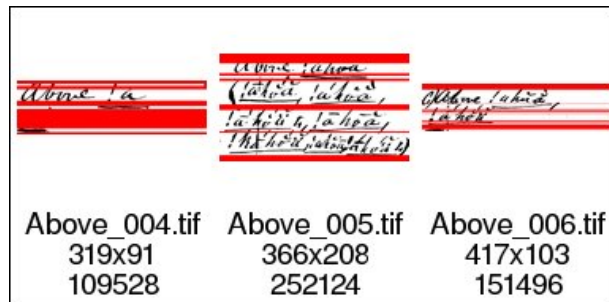
Iteration two involved bringing all the different parts of the BOLD Translator together into a single system.

Segmentation

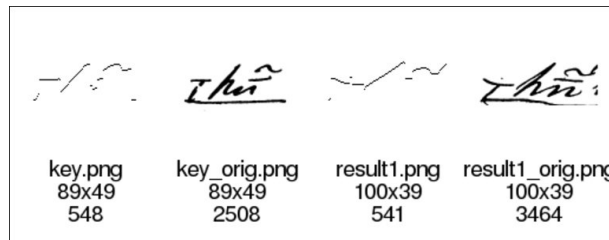
Iteration 2 implemented segmentation by identifying the lines underlying Bushman words. In this iteration, the segmentation did not detect the left and right ends of a word but rather segmented exactly at the left and right ends of the line. Similarly, the means by which the top of a word was found was more naive than in the final implementation.

Feature Extraction

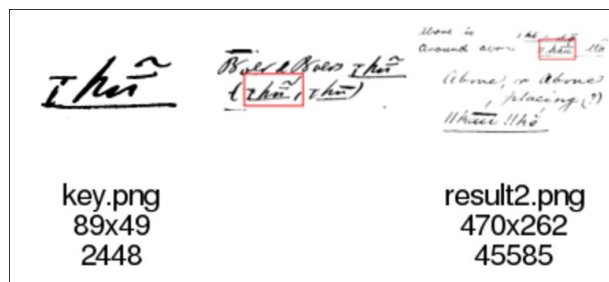
For each word which was segmented a single feature was extracted. This feature was an *average height of characters* feature and was implemented because it was easy to detect and could be used to demonstrate the use of inverted files to prune the dataset. The *average height of characters* feature was not included in the final implementation of the BOLD Translator.



(a) Segmentation



(b) Feature extraction



(c) Accurate matching

Figure 3.8: Iteration 1 screenshots

Accurate Matching

The DIF matching algorithm was improved in this iteration.

Figure 3.9 shows a screenshot from the second iteration.

3.8.3 Iteration 3

Iteration 3 was the final iteration and closely resembles the final system, except for a few modifications.

Segmentation

Iteration 3 implemented the current version of the word segmenter and enabled the functionality whereby the word segmenter attempts to identify the left and right ends of words as well as the tops of words.

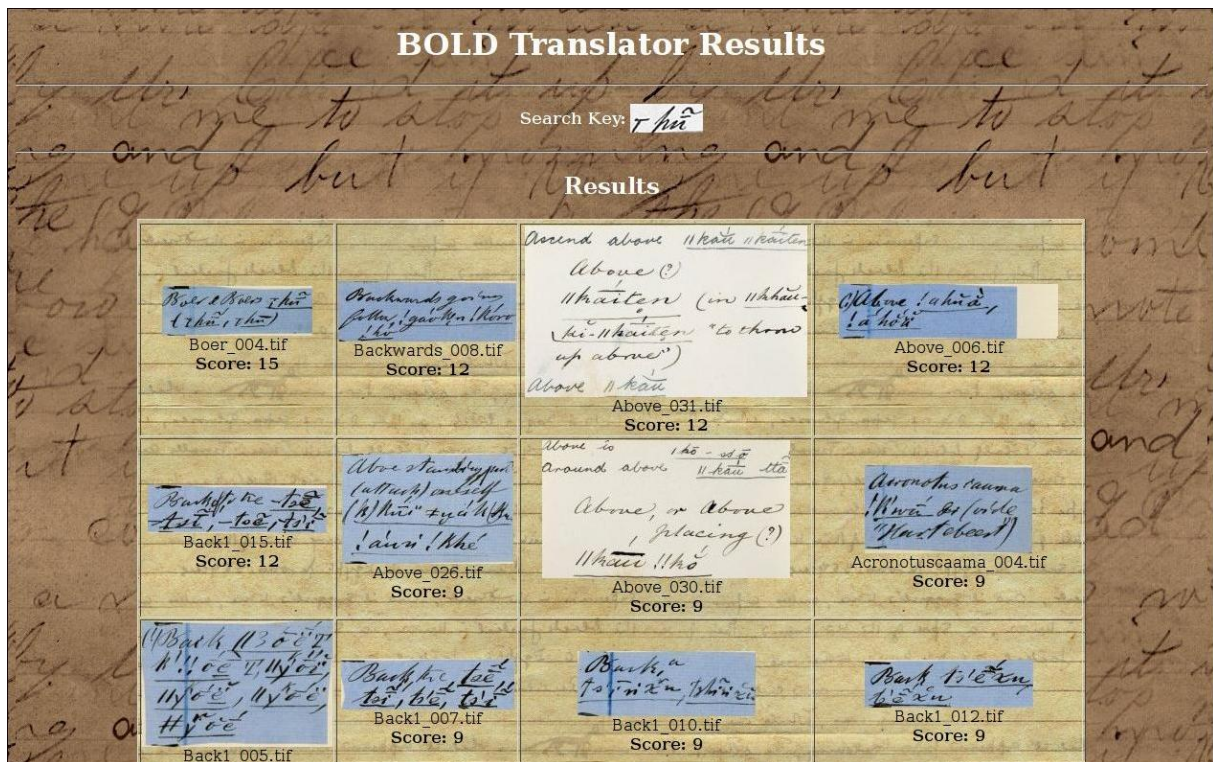


Figure 3.9: Iteration 2 screenshot

Feature Extraction

Iteration 3 implemented the full set of features which are used in the BOLD Translator. These features include the intersection feature, the horizontal lines feature, the long vertical lines feature and the short vertical lines feature. Iteration 3 also introduced feature weights into the system and allowed for variation in feature correspondence.

Accurate Matching

Iteration 3 introduced the full set of accurate matching algorithms which are used in the BOLD Translator.

Figure 3.10 shows a screenshot from the second iteration.

3.9 Portability and Maintainability

The BOLD Translator was written almost exclusively in C++ and makes use of the well established ImageMagick [ImageMagick, 2009] libraries to perform the image processing. ImageMagick libraries exist for all major platforms. The choice of C++ as a programming language as well as the wide support of the ImageMagick libraries ensures a high level of portability. Furthermore, interfaces for ImageMagick exist for almost every major programming language in use today, so it is possible to re-implement the BOLD Translator in any of these languages.

The structure of the BOLD Translator is modular and is split into the BOLD LineDetector, BOLD Segmenter, BOLD FeatureDetector, BOLD Preprocessor, BOLD Matcher and BOLD

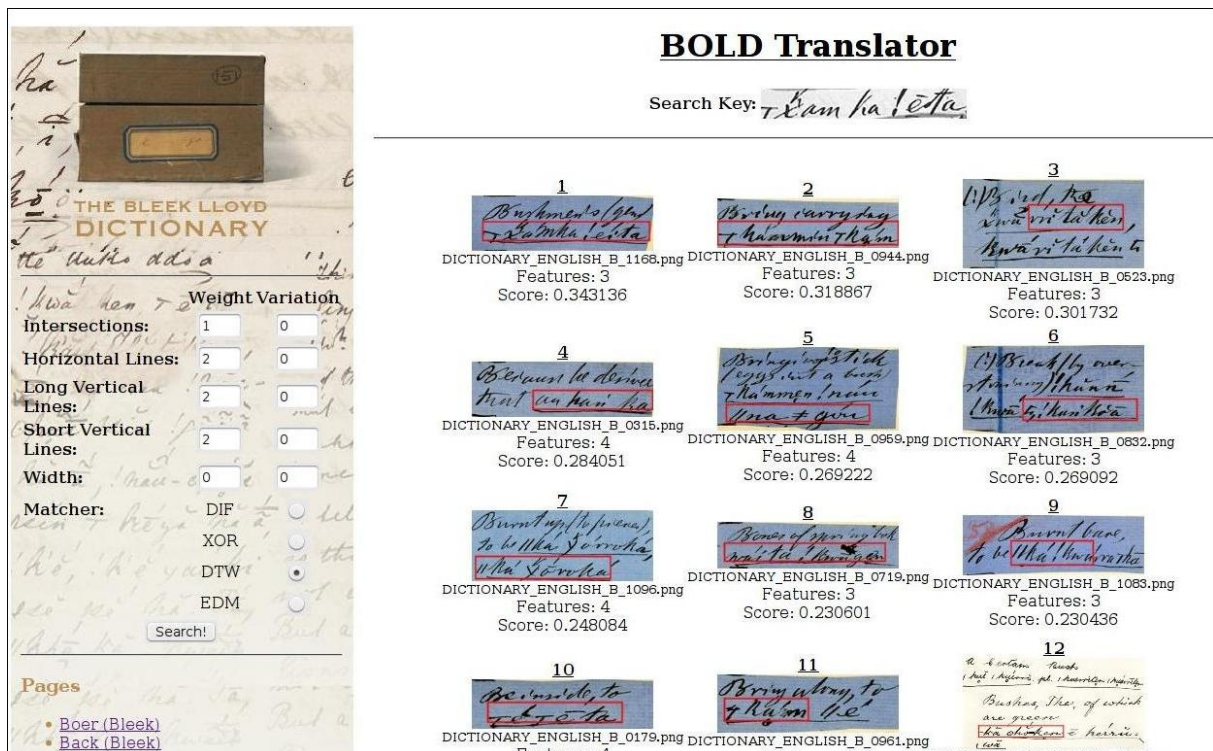


Figure 3.10: Iteration 3 screenshot

Translator. This structure makes it relatively easy to maintain the code as the different modules which make up the BOLD Translator can be developed, modified and maintained independently.

3.10 Issues

The use of thresholds and normalising of scores in the BOLD Translator has the potential to cause problems and poor results. This occurs when one word has a significantly higher matching score than all other words. When this occurs, then the normalisation process results in the word with the highest match having a normalised matching score of 1, and all other matches having low normalised matching scores. This means that the matching threshold needs to be extremely low to return results and it may not be evident to the user that this is the case.

In the project proposal for the BOLD Project it was said that, included in the image-based translation component, there would be the option for the user to enter a string of English words and have a string of images returned to them containing the corresponding |xam words. During implementation, however, it was found that this aspect of the BOLD Project was in fact much better suited to the search and browse interfaces, since they were built on the concept of allowing the user to enter an English word and get the corresponding dictionary image. For this reason, this function was not included in the BOLD Translator but rather forms part of the search and browse interfaces developed by Sanvir Manilal.

3.11 Discussion

This chapter has discussed the design and implementation of the BOLD Translator in detail. An effort has been made to describe the implementation such that it is possible to re-implement

the framework for other collections in any other major programming language by making use of the ImageMagick libraries. The iterative design of the BOLD Translator has been discussed demonstrating how the system has evolved in an experimental manner with each iteration building on the previous. Lastly, it has been noted that there is a design flaw in the use of thresholds which can sometimes have a negative effect on results.

Chapter 4

Evaluation

4.1 Introduction

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

$$\text{Precision} = \frac{\text{No. of relevant documents retrieved}}{\text{number of documents retrieved}} \quad (4.1)$$

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

$$\text{Recall} = \frac{\text{No. of relevant documents retrieved}}{\text{number of relevant documents in the collection}}. \quad (4.2)$$

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

$$F = \frac{2 \cdot \text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}} \quad (4.3)$$

[van Rijsbergen, 2009].

The layout for this evaluation chapter is as follows:

- The segmentation algorithm is evaluated in terms of accuracy.
- Features are evaluated to determine the extent to which they are able to prune the dataset for accurate matching.
- Variation is evaluated in order to determine whether or not they can improve the results of using features to prune the dataset.
- The performance of using features to prune the dataset is evaluated.
- The four matching algorithms are evaluated in terms of accuracy and performance.
- End-to-end testing is conducted to evaluate how the system as a whole performs as a whole using the optimal values identified in modular tests, as well as to determine what happens when the optimal value are changed. End-to-end evaluation is also conducted to determine the speed and accuracy of the system as scale increases.

- User testing is conducted in order to gain a better understanding into the usability and usefulness of the system.

For all experiments which use search keys, three different words (Figure 4.1) which are known to exist in the collection and which have varying characteristics were selected as search keys. Each of these words was selected in 10 different ways (Figure 4.2) and each experiment was carried out on 3 of the 10 possible selections since time limitations did not allow for all 10 selections to be used. The purpose of using different selections of the word is to determine the extent to which word selection affects results.


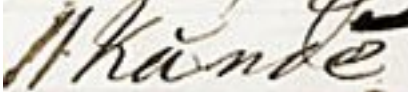
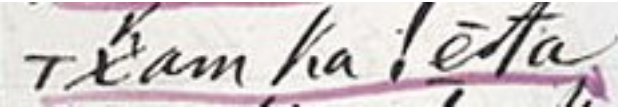
| Key Name | Image | Characteristics |
|----------|--|-----------------|
| Key 1 |  | Small |
| Key 2 |  | Medium |
| Key 3 |  | Large |

Figure 4.1: Search keys used in experiments


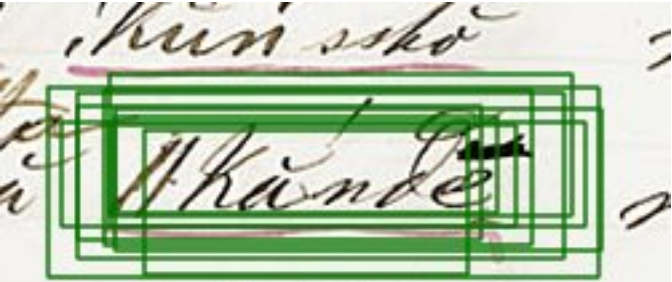
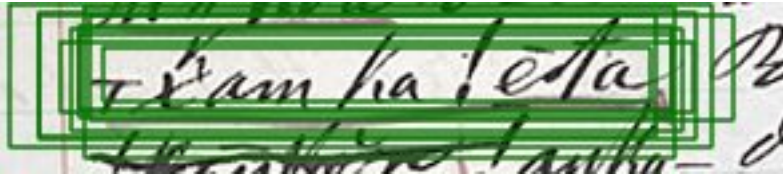
| Key | Selections |
|-------|--|
| Key 1 |  |
| Key 2 |  |
| Key 3 |  |

Figure 4.2: Search key selections

4.2 Segmentation

4.2.1 Experiment: Segmentation of words in an image

- **Purpose**

- To determine the extent to which segmentation can be successfully performed, as measured by precision and recall.
- To determine the optimum minimum line length for segmentation.

- **Procedure**

- A subset of the collection which consists of 10 images randomly picked for each letter of the alphabet is obtained. In cases where there are less than 10 images for any letter, all of the images for that letter are picked.
- The segmentation is performed on the subset of images for minimum word underlying line lengths of 10%, 20%, 30%, 40%, 50%, 60%, 70%, 80%, 90% and 100%.
- For each minimum underlying line length, the precision and recall is recorded for each individual image and then averaged out for the whole subset.
- The concepts of precision and recall are modified slightly to suit segmentation such that precision is calculated as

$$\text{Precision} = \frac{\text{No. of relevant segmented words retrieved}}{\text{number of segmented words retrieved}} \quad (4.4)$$

and recall is calculated as

$$\text{Recall} = \frac{\text{No. of relevant segmented words retrieved}}{\text{number of relevant segmented words}} \quad (4.5)$$

- For each minimum underlying line length, the weighted harmonic mean - or F-Score - for the average precision and recall of the subset is recorded and averaged out for the whole subset where

$$F = \frac{2 \cdot \text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}} \quad (4.6)$$

- **Results**

- The experimental results are summarised in Figure 4.3.

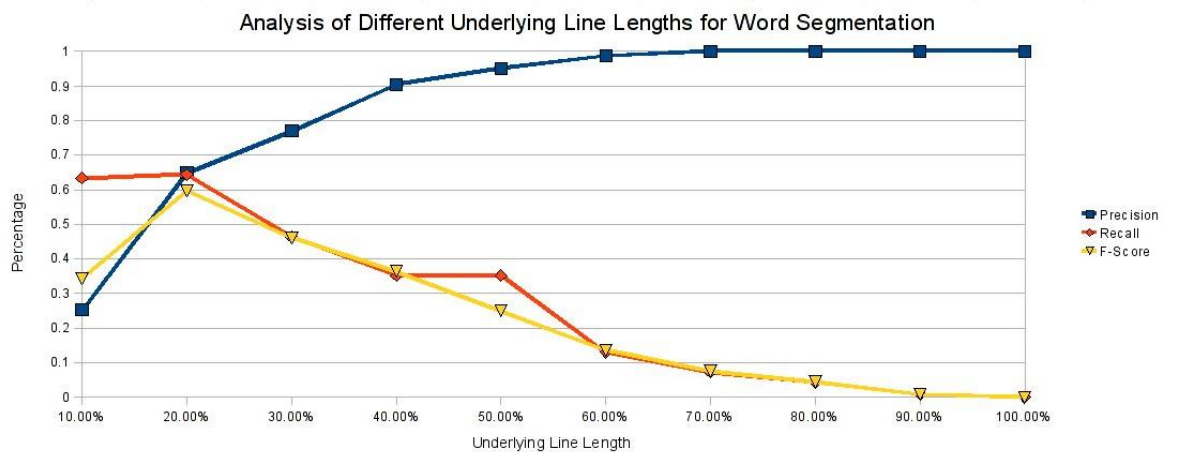


Figure 4.3: Analysis of different underlying line lengths for word segmentation

[h!]

- **Discussion**

- Figure 4.3 shows that there is a clear tradeoff between precision and recall.
- Precision increases consistently as the underlying line length increases. This is due to the algorithm becoming less likely to pick up lines which do not underline words such as the line which 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 which 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 optimal trade-off between precision and recall and exists at a minimum underlying length of 20%. 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 words in an image, it is reported that segmentation occurs with around 60% success.

4.3 Features

Features are used to prune the subset to allow for more accurate matching at a later stage. A number of experiments were carried out to determine the ability of features to prune the dataset, the best weights for the various features and performance issues when using features to prune the dataset.

4.3.1 Experiment: Using features to prune results

- **Purpose**

- To determine the extent to which features can be used to prune the data set for matching.
- To determine the optimum weights for each of the features.

- **Procedure**

- A subset equal to 20% of the full collection, which amounts to 2921 images is used in this experiment.
 - Each key image is matched with every word image and the similarity measure is based on feature correspondence only.
 - Different features weights are used in order to determine the best feature weights. Feature weights are calculated such that $\sum_{i=1}^4 W_i = 1$ where W_i = the weight given to feature i .
 - The following feature weights in the form of $Y = W_1 + W_2 + W_3 + W_4$ are 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. Additional feature weights are discussed in the future work section of this report in chapter 5.
- Feature scores are normalised over the range [0-1].
 - The experiment is carried out at threshold values of 100%, 80%, 60%, 40%, 20% and 0%.
 - For each threshold value the precision, recall and F-score is reported.

- **Results**

Figure 4.4 shows the results of this experiment for the first selection of the ten images for each key. The full set of results is available in Appendix A

- **Discussion**

- Precision and the F-score are always zero and are therefore not shown in the graphs.
- It is shown 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 appears to be no set which consistently outperforms the others, as is shown by the relatively constant recall across the different feature weight combinations tested.
- By using a subset of 20% of the dataset this experiment shows that a threshold can be used to prune the dataset. It is assumed that this remains true as the size of the collection increases and therefore experiments were not done on larger subsets.
- The low threshold required for positive recall as well as the lack of effect of different feature weights could be due to the 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 will test to see if allowing for variation in feature correspondence results in a decreased required threshold for recall as well as identifying a strong set of feature weights.

4.3.2 Experiment: Introducing variation into feature correspondence

- **Purpose**

- To determine if allowing feature correspondence variation as discussed in section 3.6.3 can improve accuracy.

Recall

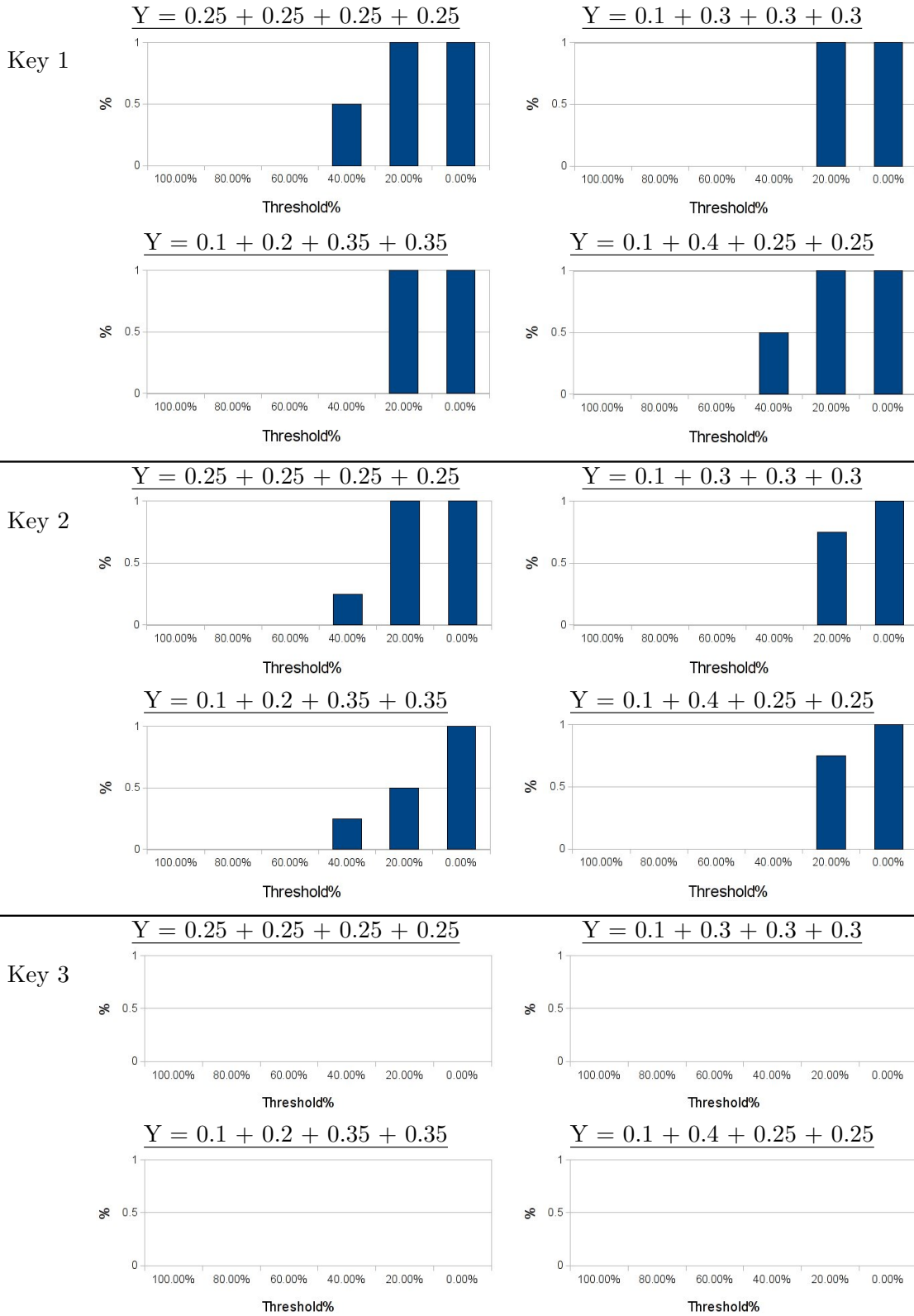


Figure 4.4: Recall for different feature weightings

- **Procedure**

- Feature variations of 1 and 2 are allowed, meaning that features differ from the key by $|1|$ and $|2|$.
- The equal feature weighting $Y = 0.25 + 0.25 + 0.25 + 0.25$ is used
- Feature scores are normalised over the range $[0-1]$.
- The experiment is carried out at threshold values of 100%, 80%, 60%, 40%, 20% and 0%.
- The recall is reported for each threshold value and allowed variation.
- The results of the matching with varying feature correspondence is compared to that without.

- **Results**

Figure 4.5 shows the results of this experiment for the first selection of the ten images for each key. The full set of results is available in Appendix A

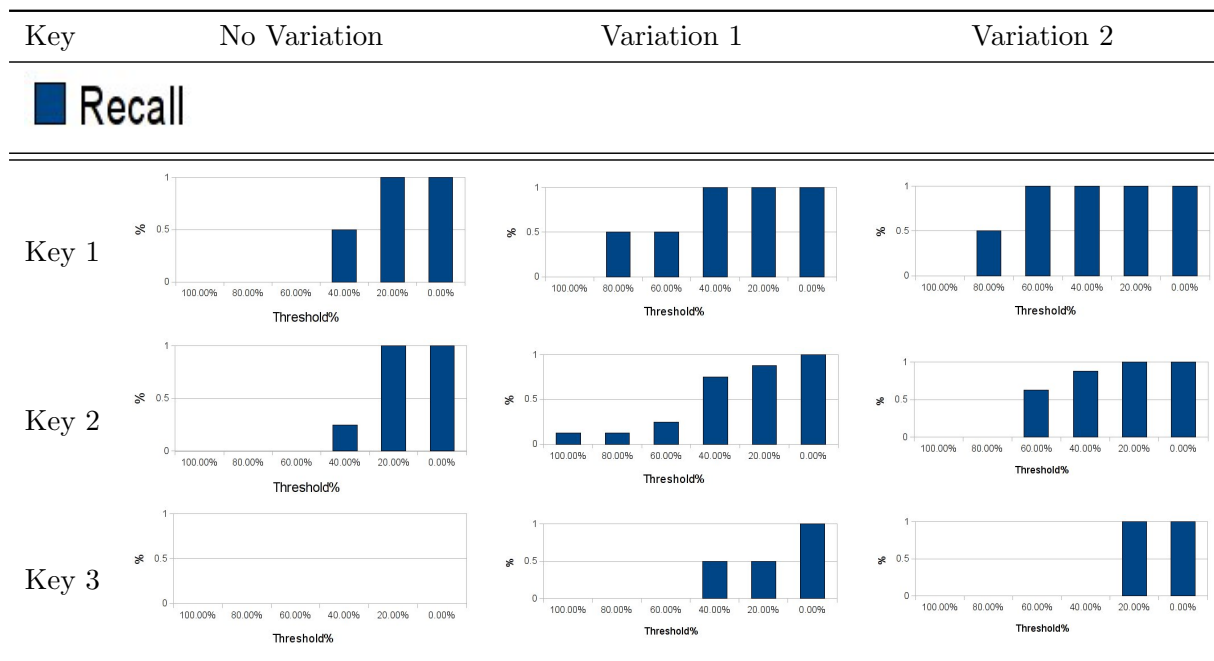


Figure 4.5: Results of introducing variation into feature matching

- **Discussion**

- It is clear from Figure 4.5 that variation has a positive effect on the required threshold for relevant matches.
- While allowing for a variation level of 1 almost always improves on results with no variation, allowing for a variation level of 2 does not necessarily improve results as is evident for keys 2 and 3.
- Examining Appendix A shows that there is still no set of feature weights which clearly out performs the others. However, there is slight evidence to suggest that the first set of feature weights performs slightly worse than the other sets of feature weights when variation is introduced. This is shown by the other sets of feature weights requiring a lower threshold to result in a positive recall.

- While introducing variation into feature weights has a positive effect on the required threshold, it most likely has a negative effect on the speed at which the system performs matches as well as the accuracy of the accurate matching. This is due to the possibility of one of the additional images, which is not a relevant result, scoring a higher accurate matching score than a relevant result.

4.3.3 Experiment: Evaluating the speed of feature based matching

- **Purpose**

- To determine the speed at which feature matching takes place for various sizes of subset.

- **Procedure**

- Subsets of size 20%, 40%, 60%, 80% and 100% of the dataset are used to determine the effect that dataset size has on the speed of pruning results using features.
- A single image is used to prune the results of each of the subsets and the time taken is recorded. This is repeated 10 times for each subset and the average time is calculated.
- An equal feature weighting with no variation is used for this experimental.
- The threshold for returning results is set at 50%.
- Time reported is system time.

- **Results** Figure 4.6 shows the results of measuring the speed of the system as scale increases.

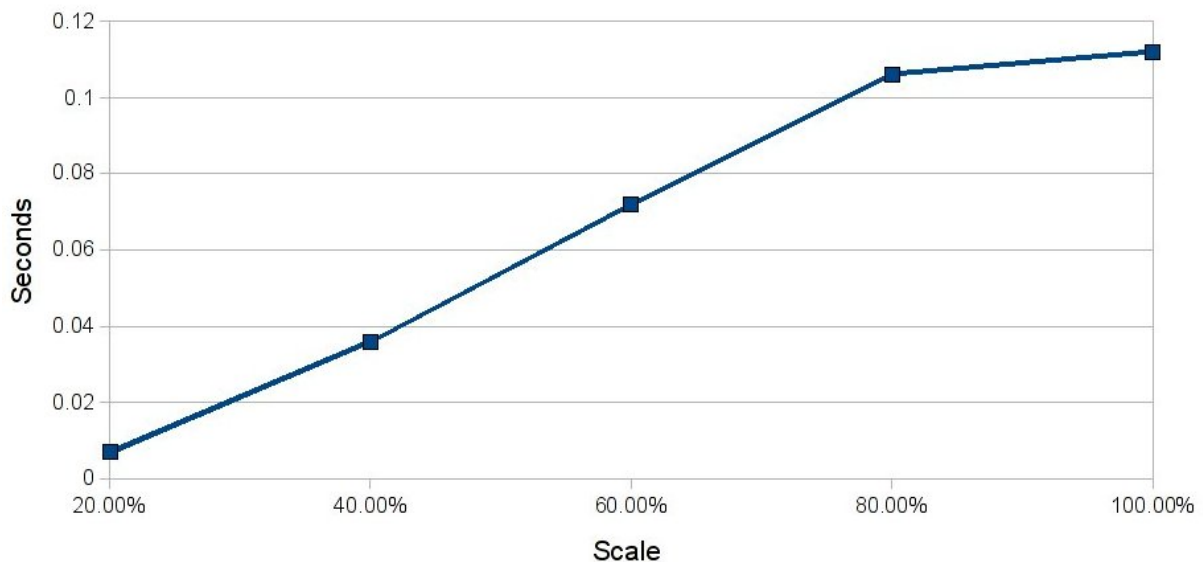


Figure 4.6: Effect of dataset size on on feature performance

- **Discussion**

- 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 which need to be compared, leading to larger inverted files and larger feature score calculation. This highlights potential scalability issues as the system might perform slowly as the size of the dataset increases. This will be investigated further in section 4.5.

- Inverted file optimisations may have an effect on the speed of feature pruning as the dataset increases and are discussed in chapter 5.

4.4 Accurate Matching

4.4.1 Experiment: Evaluating the accuracy of each matching algorithm

- **Purpose**

- To determine which matching algorithm provides the most accurate matching

- **Procedure**

- Each matching algorithm is run on a subset of images made up of 100 images from the dataset.
- The experiment makes use of the first three selections of each of the search keys.
- The match scores were recorded and normalised over the range [0-1].
- The experiment was carried out at threshold values of 100%, 80%, 60%, 40%, 20% and 0%.
- For each threshold value the precision, recall and F-score is reported.

- **Results**

Figure 4.7 shows the F-score for each result. Refer to Appendix B for the precision and recall. Table 4.1 shows the average F-score at each threshold for each of the matching algorithms for the 9 images used in the experiment.

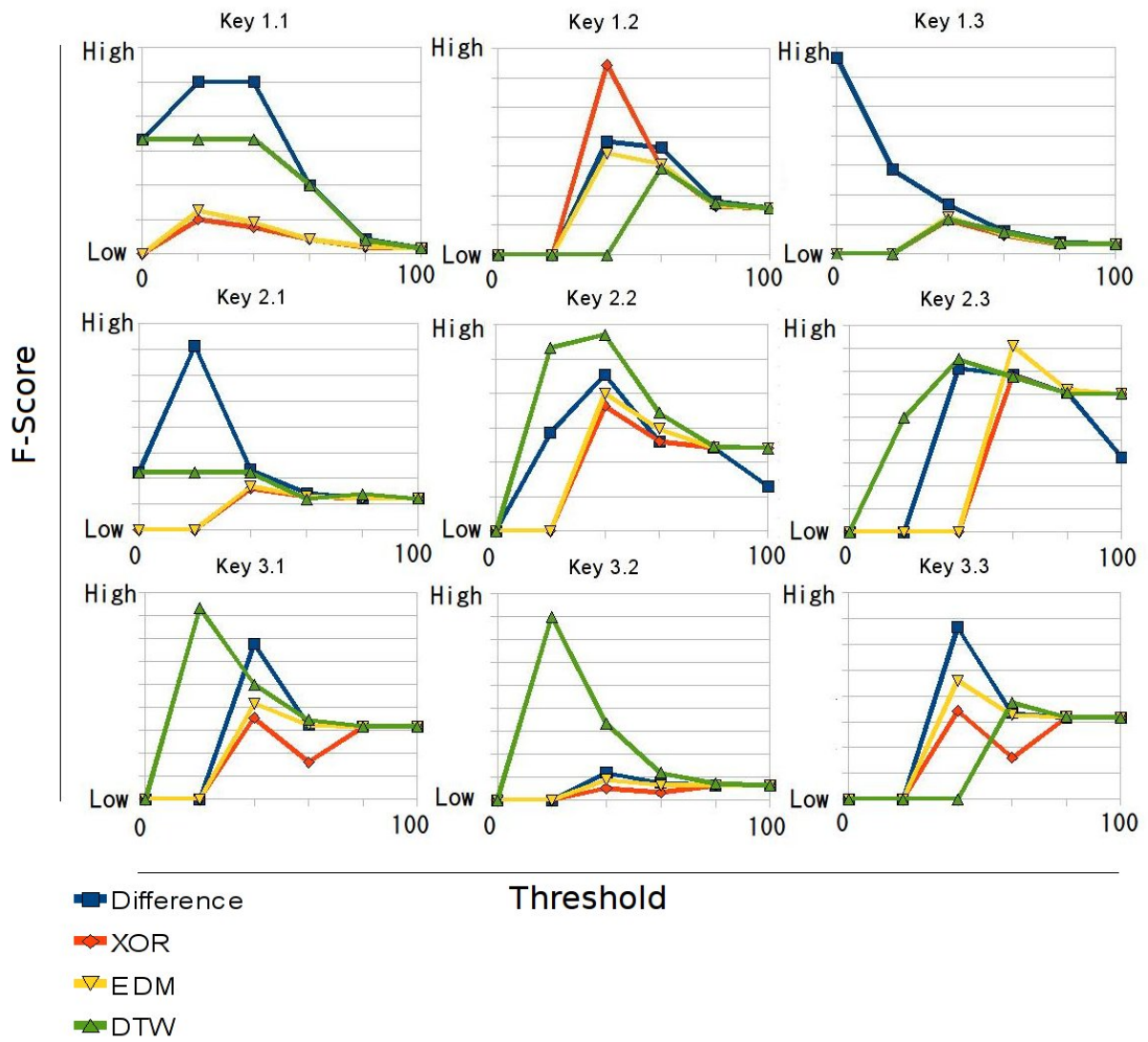


Figure 4.7: Comparison of matching algorithms for three selections of each key

| | 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 |
| DTW | 0.01 | 0.19 | 0.18 | 0.12 | 0.07 | 0.06 |

Table 4.1: Average F-score at each threshold for each of the matching algorithms for the 9 images used in the experiment

- **Discussion**

- The results above suggest that the DIF algorithm as well as the DTW algorithm generally result 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 and DTW algorithms appear to outperform the XOR and EDM algorithms, an important issue to consider is the time that each of these algorithms take to run. This will be investigated in the next experiment.

4.4.2 Experiment: Evaluating the speed of each matching algorithm

- **Purpose**

- To determine which matching algorithm is the fastest and how speed is affected as scale increases.

- **Procedure**

- The speed of each matching algorithm is recorded using subsets of 100, 200, 300, 400 and 500 images.
- Each matching algorithm is run five times and the average speed is reported.
- The time reported is user clock time.

- **Results** The results of the experiment are shown in Figure 4.8.

- **Discussion**

- The DIF and XOR matching algorithms are the fastest, EDM is the slowest, while DTW falls in between.
- Combining this result with the result in section 4.4.1 suggests that the DIF algorithm is the best algorithm for matching since it is one of the fastest algorithms and is also the most accurate.
- XOR, while fast, has poor accuracy and is not a viable algorithm.
- DTW, while not as fast as the DIF and EDM algorithms does have a high level of accuracy and thus may be suitable when the difference algorithm returns poor results.
- EDM is the slowest algorithm and it is only slightly more accurate than the XOR algorithm, also making it a non-viable algorithm for matching.

4.5 End-to-end Testing

The purpose of end-to-end testing is to determine how well the BOLD Translator works as a complete system.

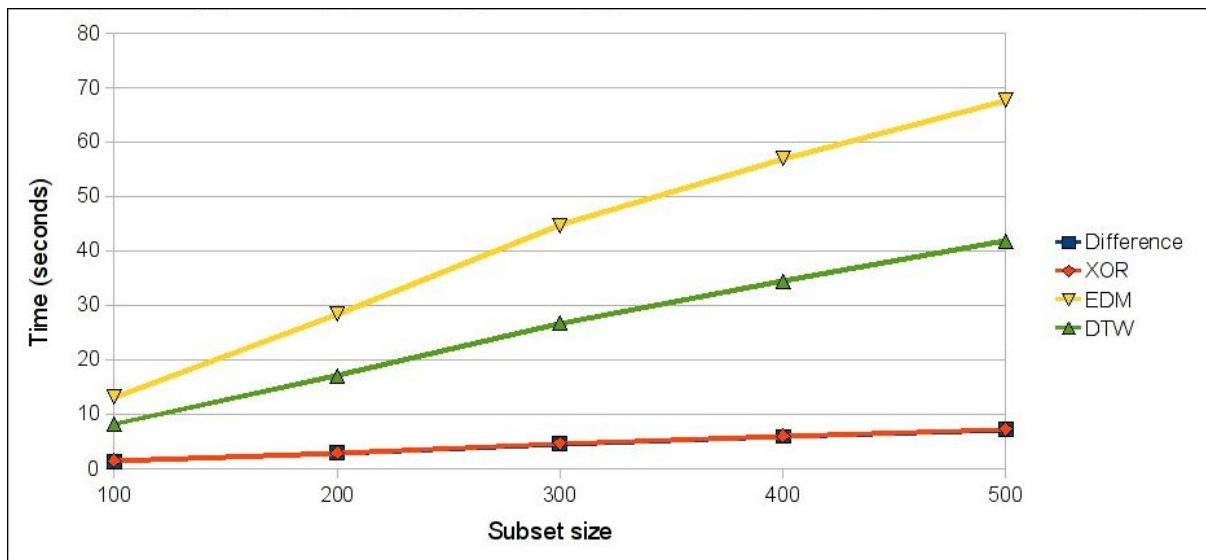


Figure 4.8: Matching algorithms performance measures

4.5.1 Experiment: Testing the optimal values

- **Purpose**

- To determine if the optimal values for each part of the BOLD Translator as identified by the previous experiments can produce good end-to-end results.

- **Procedure**

- A subset of 20% of the dataset is used, which amounts to 2921 images.
- Equal weights for features are used since no feature weight combinations were found to be optimal in section 4.3.1.
- Feature correspondence variation of 1 is used since it was shown in section 4.3.2 that this improves results.
- The DIF matching algorithm is used since it was shown in section 4.4.1 and section 4.4.2 that this is the best performing and most accurate matching algorithm.
- A threshold level of 80% is used for feature scores since it was shown in section 4.3.2 that positive recall occurs when allowed feature correspondence variation is 1 and the threshold is 80%. Similarly, a threshold level of 60% is used for matching as it was shown in section 4.4.1 that the difference algorithm works well at a 60% threshold. The reason that a matching threshold of 60% is used over 80% is that, while a threshold of 80% results in a higher F-score, a threshold of 60% results in a higher recall which means that more correct results are returned. At the same time, the F-score for a matching threshold of 80% is only slightly higher than the F-score for a matching threshold of 60%.
- The experiment is run using the first three selections of each key and the precision, recall and F-score are reported.

- **Results** The precision, recall and F-score for each of the 9 images used as search keys is shown in Figure 4.9.

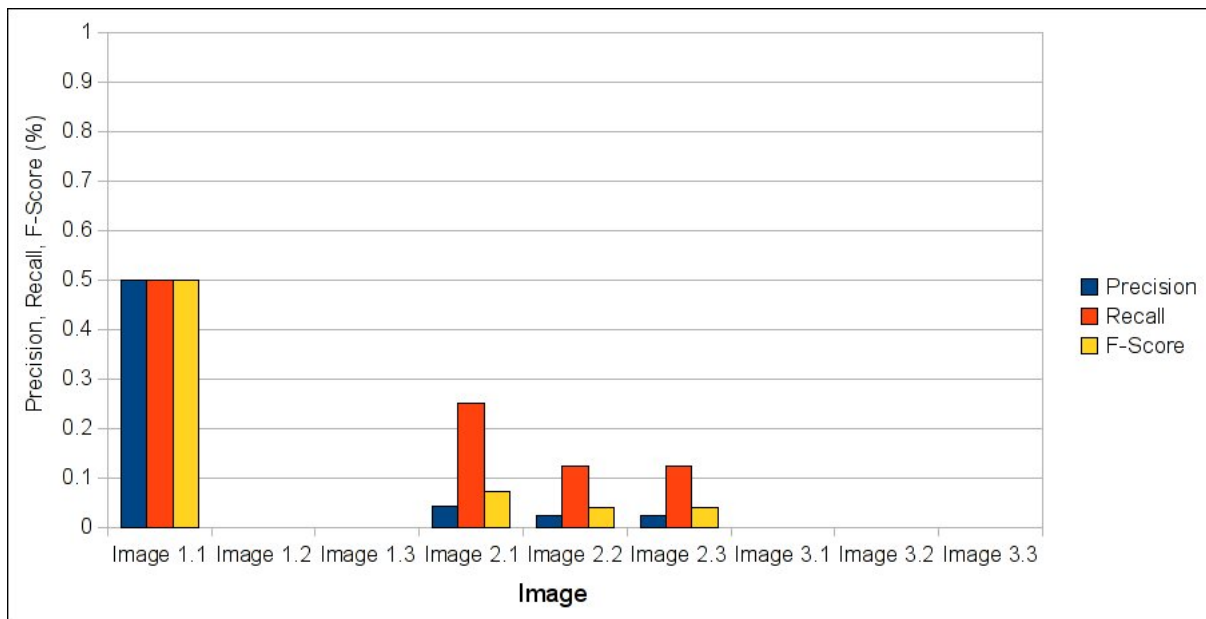


Figure 4.9: Precision, recall and F-score for end-to-end testing using optimal values

- **Discussion**

- Three of the nine image keys used for searching resulted in positive recall. These three keys, however, are slightly different from the other keys in that they have been “well selected,” meaning that the image keys generally include only the word and little surrounding noise. “Well selected” keys ultimately result in better matches since the feature matching, as well as accurate matching, is more accurate. This fact highlights the need for good selection of keys when making use of the BOLD Translator.
- It is shown that the optimal values do return positive results, however this only occurs $\frac{1}{3}$ of the time in the experiments carried out.

4.5.2 Experiment: Changing the matching algorithm

- **Purpose**

- To determine if using the DTW algorithm results in better matches than the DIF algorithm when doing end to end matching.

- **Procedure**

- The experimental procedure is exactly the same as that of section 4.5.1 except that the DTW algorithm is used for matching instead of the DIF algorithm.

- **Results**

The precision, recall and F-score when using the DTW algorithm were all zero.

- **Discussion**

- Using the DTW algorithm results in no results being returned when doing end-to-end evaluation. This conflicts with section 4.4.1 which showed that the DTW algorithm performs almost as well as the DIF algorithm. This requires more investigation in the future.

4.5.3 Experiment: Increasing variation

- **Purpose**

- To determine if increasing variation results in better matches when doing end to end matching.

- **Procedure**

- The experimental procedure is exactly the same as that of section 4.5.1 except that the allowed variation is 2 instead of 1

- **Results**

The precision, recall and F-score for each of the 9 images used as search keys is shown in Figure 4.10. The results are shown alongside the results when using optimal values for comparative purposes.

- **Discussion**

- As was the case in section 4.3.2, it is clear that increasing the variation results in too many incorrect matches ending up in the dataset and results in poorer results.

4.5.4 Experiment: Increasing scale and accuracy

- **Purpose**

- To determine the effect that increasing subset sizes have on accuracy.

- **Procedure**

- Keys which were shown to return results in section 4.5.1 are used to test the system at increasing scales.
- Subsets which are made up of 20%, 40%, 60%, 80% and 100% are used.
- For each subset, the precision, recall and F-score is reported.

- **Results**

The results for the increasing scales are shown in Figure 4.11

- **Discussion**

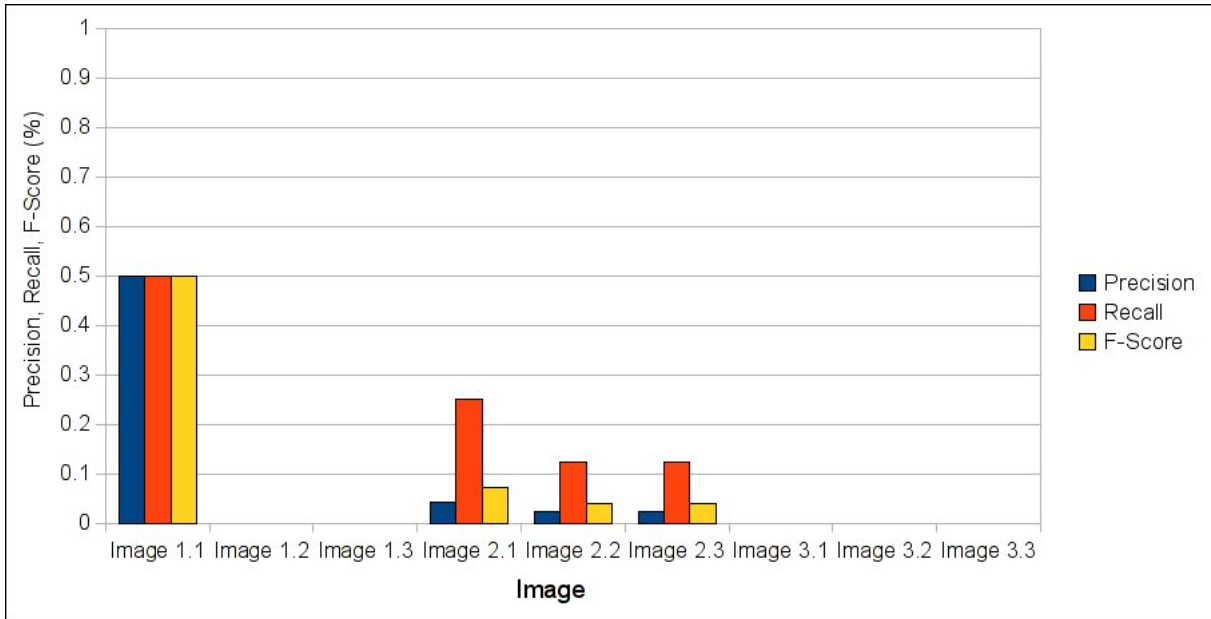
- The results of scale increase have a negative effect on precision in all cases.
- For keys 1.1 and 2.2 the recall remains constant while precision and thus the F-score decreases.
- The evidence suggests that strong matches will remain strong matches as the size of the subset increases, as is the case for keys 1.1 and 2.2, and that weak matches will disappear as the size of the subset increases, as is the case for keys 2.1 and 2.3.

4.5.5 Experiment: Increasing scale and performance

- **Purpose**

- To determine the effect that increasing subset sizes have on performance.

- **Procedure**



(a) Precision, recall and F-score for end-to-end testing with end-to-end testing with optimal values



(b) Precision, recall and F-score for end-to-end testing with increased variation

Figure 4.10: Comparison of results using optimal values and optimal values with increased variation

- Key 1.1 is used to test the system at increasing scales.
- Subsets which are made up of 20%, 40%, 60%, 80% and 100% are used.
- Optimal values are used for all variables.
- For each subset the user time is recorded 10 times and the average is reported.

• **Results** The results for the increasing scales are shown in Figure 4.12

• **Discussion**

- The system becomes increasingly slower as the subset size increases. By the time 100% of the dataset is being used the system takes 16 seconds to return results when

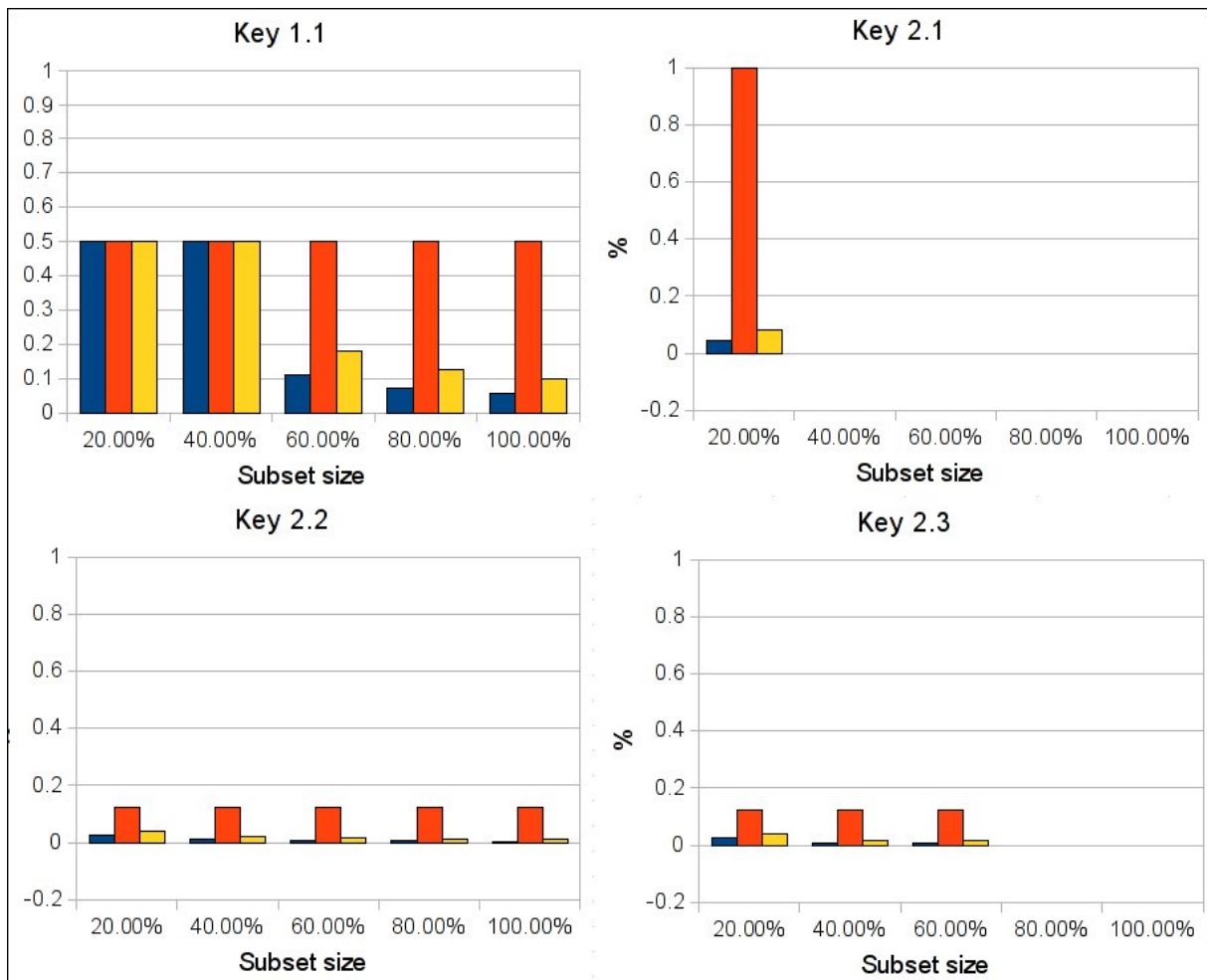


Figure 4.11: Precision, recall and F-score as scale increases

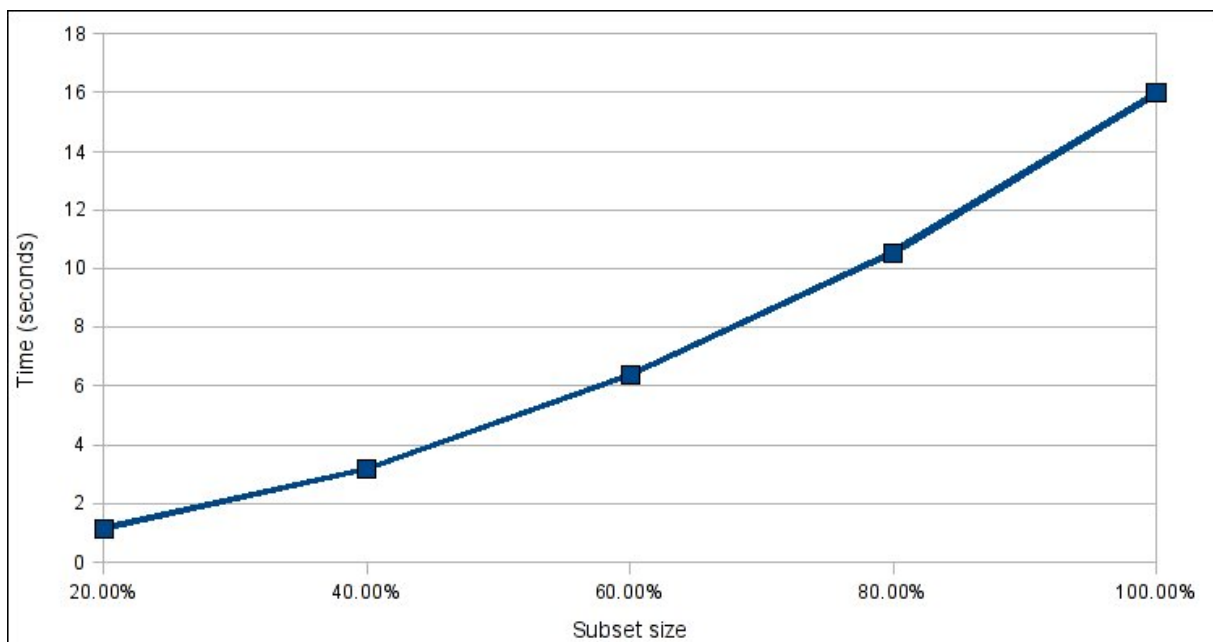


Figure 4.12: Performance as scale increases

the fastest matching algorithm is being used. This creates the need for optimisations throughout the system, which are discussed further in section 5.

4.6 User Testing

4.6.1 Experiment: User usability and usefulness

- **Purpose**

- To determine how users felt about the usability and usefulness of the system as a whole.

- **Procedure**

- Four users took part in this experiment.
- The experiment was conducted on a subset of the dataset which amounted to 1177 images which were all words which started with the letter B.
- Users were not able to set the system variables but instead equal feature weights were used for all features, feature correspondence variation of 1 was allowed and the DIF matching algorithm was used.
- The version of the system which was tested was a prototype system where the 100 words with the highest feature scores had accurate matching performed on them and the top 30 results from the accurate matching were presented to the user.
- Feature and accurate matching scores were not normalised over the range [0-1].
- The purpose of the system was explained to the users and they were asked to make use of the system to find a specific word which was pointed out to them. The word was the |xam word “*thu*” which in English means “*Boer.*”
- Once they had finished using the system to search for the word they were asked to answer a questionnaire which attempted to gain insight into the usability and usefulness of the system. The full questionnaire that users were given is available in Appendix C.

- **Results**

- All of the users who participated in this experiment, in some way, felt confused by the way that the results were displayed to them. They found it hard understand what the results meant and could not find where the translations for the words were. They all felt overwhelmed by the large number of results which were returned.
- All users except one were able to find a correct match in the results after spending some time looking through the full set of results which was returned to them.
- Users were generally quite happy with the way that words are selected and felt that the way of selecting was something that they were familiar with.
- All the users felt that a system of this nature might be helpful to researchers who are studying Bushman languages. Though it is noted that some work is needed.
- These results are in no way conclusive and are only potentially indicative of general user views on usefulness and usability.
- The full feedback provided by users is available in Appendix C

- **Discussion**

- It is evident from the user feedback that the way that results are displayed is not satisfactory. More effort needs to be put into the displaying of results as well as making it clear what the translations for each of the results are.
- Most of the users being able to find the correct match suggests that a system of this nature is suitable for use by end users. However, improvement is needed in order to improve the quality of matches and return fewer results to users.
- Users acknowledge that a system of this nature is useful, which suggests that they may see potential in the system. However, a lot more effort needs to be put into making the system more user friendly in the future.

4.7 Summary

Segmentation of words in an image has been evaluated and it has been shown that a success rate of around 60% was achieved. It was then shown that features are useful for pruning the collection in order to do more accurate matching and that by introducing variation, pruning can be improved. However, it was not possible to identify if an optimal set of feature weights exist. It was shown that the DIF algorithm out performs all other algorithms in terms of speed and accuracy and that for these reasons it is a good choice of matching algorithm. End-to-end testing showed that the optimal values identified in the modular testing remain optimal at an end-to-end level even when scale increases and that when key images are well selected, it is possible to achieve positive recall and precision. User testing showed that the system is not suitable for use by end users as the interface is unintuitive and confusing, however, the fact that users were able to find correct matches is a promising result.

The results of the evaluation suggest that the answer to the research question - *can image-based searching be done accurately and effectively?* - is yes. It has been shown that when a search key is well selected it is possible to find the correct match in the collection even as the size of the collection grows in size. Issues to deal with in this regard, however, include the incorrect matches which are returned at the same time and which can be overwhelming to users. In terms of the effectiveness of image based searching, when the subset size is just under 3000 images, results are returned in about 1 second of user clock time. When the size of the collection increases to over 14000 images, results are returned in about 16 seconds. It is important to note that these times take place in the absence of any optimisations and that implementing speed optimisation could potentially have a significant impact on the running time of the system.

Chapter 5

Future Work

There are limitless possibilities for future work on a project of this nature since it will always be desirable to improve performance, accuracy and usefulness to end users. Some possibilities for future development are discussed here.

5.1 Continuous Improvements

Continuous improvement can be made in terms of segmentation, feature extraction, accurate matching and performance. Such improvements involve increasing the segmentation success rate, increasing the accuracy of feature matching, implementing inverted file optimisations and improving matching algorithms.

5.2 Additional Testing

Due to time constraints limited testing of the system could be done. Future work involves doing more testing in order to gain additional insight into the inner workings of the system.

5.3 User Interface

The system depends heavily of good selection of search keys in order to perform good matches. Future work would involve developing an intelligent user interface which is able to detect the optimal word region in the area which the user selects. The user interface can also be developed using interface standards and subject to heuristic evaluations.

5.4 Implementation of New Features

New features can be implemented to allow for more criteria for dataset pruning using features. Additional features introduce more parameters which can be used to decide if a word in the collection is similar to the search key. More features could potentially result in large improvements to the accuracy of the system though it must be noted that there is the possibility that they will introduce performance issues.

5.5 User Feedback

The system could be developed such that it was capable of receiving user feedback. User feedback is useful in that it could be used to remove badly segmented words from the collection. Similarly, allowing users to rank results and identify where good results occur could allow the system to “learn” from user feedback.

5.6 Word Spotting Index

It is possible to extend the word spotting idea such that all Bushman words in the notebooks are identified and an English index could be made for all the words.

5.7 Configurable System

The framework could be extended such that it is capable of dealing with different types of collections such as printed text or handwritten text or texts using different character sets. In this sense the framework will allow for optimisations for different types of collections.

Chapter 6

Conclusions

This project set out to build a content based image retrieval (CBIR) framework for a collection of handwritten documents with the goal of being able to find specific images of words within a large collection. Background research showed that CBIR systems for other sorts of collections have been built before and that previous work has been done (most notably by Manmatha) in handwritten word recognition but that no effort has ever been made to put the two together. A framework was designed in order to allow for the implementation of systems of this type for any collection of handwritten documents.

It was shown that each component of the framework contributes to returning results and that the components act together to produce good results. An attempt was made to try to identify optimal values for variables used in the system and, while some insight was gained, further investigation would need to be done in order to gain a more complete picture.

The design of the system was iterative and proved to be a good method of development as each iteration could be evaluated and improved on.

Time constraints meant that not all possibilities for the different components of the system could be implemented but those which were implemented showed that success using this type of system could be achieved though scalability becomes an issue and the system tends to operate slower and less accurately on larger collections.

Possibilities for future development have been identified and discussed briefly.

It is noted that the system is experimental and not yet ready for use by end users but that results up to this point are promising.

In answering the research question - *can image-based searching be done accurately and efficiently?* - it was found that when a good search key is selected, then relevant matches can be found thereby suggesting that it is possible to do image-based searching accurately when a good search key is selected. Similarly, 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. These times are reported without any speed optimisations having taken place and it is suspected that speed optimisations could significantly improve the speed of the system.

Bibliography

- Apostolos Antonacopoulos and Dimosthenis Karatzas. Document image analysis for world war ii personal records. In *International Workshop on Document Image Analysis for Libraries (DIAL2004)*, pages 336–341. IEEE-CS Press, 2004.
- Ricardo Cabral. imgseek. <http://www.imgseek.net/>, 2009. Accessed: 06 May 2009.
- Ritendra Datta, Dhiraj Joshi, Jia Li, and James Z. Wang. Image retrieval: Ideas, influences, and trends of the new age. *ACM Comput. Surv.*, 40(2):1–60, 2008. ISSN 0360-0300. doi: <http://doi.acm.org/10.1145/1348246.1348248>.
- Reim Doumat, Elöd Egyed-Zsigmond, Jean-Marie Pinon, and Emese 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. ISBN 978-1-60558-081-4. doi: <http://doi.acm.org/10.1145/1410140.1410167>.
- J. Eakins and M. Graham. Content-based image retrieval. Technical report, JTAP, 1999. JISC Technology Applications Programme Report 39.
- Flickr! Flickr! <http://www.flickr.com/>, 2009. Accessed: 06 May 2009.
- George Gagaudakis and Paul L. Rosin. Incorporating shape into histograms for cbir. *Pattern Recognition*, 35(1):81–91, 2002. ISSN 0031-3203. doi: DOI:10.1016/S0031-3203(01)00043-7.
- GNU. Gnuce general public license. <http://www.gnu.org/licenses/gpl.html>, 2009. Accessed: 5 November 2009.
- South African Government. About government - coat of arms - linton panel. <p://www.info.gov.za/aboutgovt/symbols/coa/lintonpanel.htm>, 2009. Accessed: 1 November 2009.
- Jun Wei Han and Lei Guo. A shape-based image retrieval method using salient edges. *Signal Processing: Image Communication*, 18(2):141–156, 2003. ISSN 0923-5965. doi: DOI:10.1016/S0923-5965(02)00116-9.
- IBM. Query by image content (qbic). <http://domino.research.ibm.com/comm/pr.nsf/pages/rsc.qbic.html>, 2009. Accessed: 06 May 2009.
- ImageMagick. Imagemagick. <http://www.imagemagick.org/>, 2009. Accessed: 30 October 2009.
- Charles E. Jacobs, Adam Finkelstein, and David 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. ISBN 0-89791-701-4. doi: <http://doi.acm.org/10.1145/218380.218454>.
- Christian Langreiter. retrievr. <http://labs.systemone.at/retrievr>, 2009. Accessed: 06 May 2009.

- Yann Leydier, Frank Lebourgeois, and Hubert Emptoz. Text search for medieval manuscript images. *Pattern Recognition*, 40(12):3552–3567, 2007. ISSN 0031-3203. doi: DOI:10.1016/j.patcog.2007.04.024.
- Laurence Likforman-Sulem, Abderrazak Zahour, and Bruno Taconet. Text line segmentation of historical documents: a survey. *Int. J. Doc. Anal. Recognit.*, 9(2):123–138, 2007. ISSN 1433-2833. doi: <http://dx.doi.org/10.1007/s10032-006-0023-z>.
- G. Louloudis, B. Gatos, I. Pratikakis, and C. Halatsis. Text line and word segmentation of handwritten documents. *Pattern Recognition*, 42(12):3169 – 3183, 2009. ISSN 0031-3203. doi: DOI:10.1016/j.patcog.2008.12.016. New Frontiers in Handwriting Recognition.
- R. Manmatha and W. B. Croft. Word spotting: Indexing handwritten archives, 1997.
- R. Manmatha and Nitin 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. ISBN 3-540-66498-X.
- Michael. The michael project. <http://www.michael-culture.org/en/about/project>, 2009. Accessed: 06 May 2009.
- Theo Nicolakis, Carlos E. Pizano, Bianca Prumo, and Mitchell 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. ISBN 1-58113-786-9. doi: <http://doi.acm.org/10.1145/947380.947384>.
- Waqas Rasheed, Youngeun An, Sungbum Pan, Ilhoe Jeong, Jongan Park, and Jinsuk Kang. Image retrieval using maximum frequency of local histogram based color correlogram. In *AMS '08: Proceedings of the 2008 Second Asia International Conference on Modelling & Simulation (AMS)*, pages 322–326, Washington, DC, USA, 2008. IEEE Computer Society. ISBN 978-0-7695-3136-6. doi: <http://dx.doi.org/10.1109/AMS.2008.76>.
- Toni M. Rath and R. Manmatha. Word image matching using dynamic time warping. *Computer Vision and Pattern Recognition, IEEE Computer Society Conference on*, 2:521, 2003. ISSN 1063-6919. doi: <http://doi.ieeecomputersociety.org/10.1109/CVPR.2003.1211511>.
- Tony M. Rath and R. Manmatha. Word spotting for historical documents. *International Journal on Document Analysis and Recognition*, 9(2-4):139–152, APR 2007.
- Yong Rui, Thomas S. Huang, and Shih-Fu Chang. Image retrieval: Current techniques, promising directions, and open issues. *Journal of Visual Communication and Image Representation*, 10(1):39–62, 1999. ISSN 1047-3203. doi: DOI:10.1006/jvci.1999.0413.
- P. Skotnes. *Claim to the Country: the archive of Wilhelm Bleek and Lucy Lloyd*. Jacana Media and Ohio University Press, 2007.
- Pippa Skotnes. Lloyd bleek project. <http://www.lloydbleekcollection.uct.ac.za/index.jsp>, 2009. Accessed: 02 May 2009.
- Arnold W. M. Smeulders, Marcel Worring, Simone Santini, Amarnath Gupta, and Ramesh Jain. Content-based image retrieval at the end of the early years. *IEEE Trans. Pattern Anal. Mach. Intell.*, 22(12):1349–1380, 2000. ISSN 0162-8828. doi: <http://dx.doi.org/10.1109/34.895972>.
- H. Suleman. Digital libraries without databases: The bleek and lloyd collection. In *Proceedings of Research and Advanced Technology for Digital Libraries, 11th European Conference (ECDL 2007)*, pp. 392–403, 16-19 September, Budapest, Hungary., pages 392–403, 2007. crossref: DBLP:conf/ercimdl/2007.

- H. Suleman. Digital libraries course slides. *Department of Computer Science, University of Cape Town*, 2009.
- TREC. Trec. <http://trec.nist.gov/>, 2009. Accessed: 30 October 2009.
- UNESCO. *World Culture Report: Cultural Diversity, Conflict and Pluralism (2000)*. UNESCO Publishing, 2000. ISBN 92-3-103751-3.
- C. J. van Rijsbergen. Information retrieval (online book). <http://www.dcs.gla.ac.uk/Keith/Preface.html>, 2009. Accessed: 5 November 2009.
- Yahoo! Yui! libaray. <http://developer.yahoo.com/yui/>, 2009. Accessed: 1 November 2009.
- Dengsheng Zhang and Guojun Lu. Shape-based image retrieval using generic fourier descriptor. *Signal Processing: Image Communication*, 17(10):825–848, 2002. ISSN 0923-5965. doi: DOI: 10.1016/S0923-5965(02)00084-X.

Appendices

Appendix A

Features and Variation

P = Precision
R = Recall
F = F-Score

| | No Variation | | | | | | Variation 1 | | | | | | Variation 2 | | | | | |
|--------------------------------------|--------------|----|-----|-----|-----|-----|-------------|-----|-----|-----|-----|---|-------------|-----|-----|-----|----|---|
| Threshold | 100 | 80 | 60 | 40 | 20 | 0 | 100 | 80 | 60 | 40 | 20 | 0 | 100 | 80 | 60 | 40 | 20 | 0 |
| Y = 0.25 + 0.25 + 0.25 + 0.25 | | | | | | | | | | | | | | | | | | |
| Key 1.1 | | | | | | | | | | | | | | | | | | |
| P | 0 | 0 | 0 | 0 | 0 | 0 | 0 | .01 | 0 | 0 | 0 | 0 | 0 | .01 | 0 | 0 | 0 | 0 |
| R | 0 | 0 | 0 | .5 | 1 | 1 | 0 | .5 | .5 | 1 | 1 | 1 | 0 | .5 | 1 | 1 | 1 | 1 |
| F | 0 | 0 | 0 | 0 | 0 | 0 | 0 | .03 | 0 | 0 | 0 | 0 | 0 | .01 | 0 | 0 | 0 | 0 |
| Key 1.2 | | | | | | | | | | | | | | | | | | |
| P | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R | 0 | 0 | 0 | 0 | 0 | .5 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | .5 | 1 | 1 | 1 |
| F | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Key 1.3 | | | | | | | | | | | | | | | | | | |
| P | 0 | 0 | .01 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R | 0 | 0 | .5 | .5 | 1 | 1 | 0 | 0 | .5 | .5 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 1 |
| F | 0 | 0 | .01 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | .01 | 0 | 0 | 0 |
| Key 2.1 | | | | | | | | | | | | | | | | | | |
| P | 0 | 0 | 0 | 0 | 0 | 0 | .09 | .02 | .01 | 0 | 0 | 0 | 0 | 0 | .01 | 0 | 0 | 0 |
| R | 0 | 0 | 0 | .25 | 1 | 1 | .13 | .13 | .25 | .75 | .88 | 1 | 0 | 0 | .63 | .88 | 1 | 1 |
| F | 0 | 0 | 0 | .01 | .01 | .01 | .11 | .03 | .02 | .01 | .01 | 0 | 0 | 0 | .02 | .01 | 0 | 0 |
| Key 2.2 | | | | | | | | | | | | | | | | | | |
| P | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | .01 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R | 0 | 0 | 0 | .5 | .88 | .88 | 0 | 0 | .25 | .88 | 1 | 1 | 0 | 0 | .38 | .75 | 1 | 1 |
| F | 0 | 0 | 0 | .01 | .01 | .01 | 0 | 0 | .01 | .01 | .01 | 0 | 0 | 0 | .01 | .01 | 0 | 0 |
| Key 2.3 | | | | | | | | | | | | | | | | | | |
| P | 0 | 0 | 0 | 0 | 0 | 0 | 0 | .02 | .01 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R | 0 | 0 | 0 | .5 | .88 | .88 | 0 | .13 | .25 | .88 | 1 | 1 | 0 | 0 | .38 | .75 | 1 | 1 |
| F | 0 | 0 | 0 | .01 | .01 | .01 | 0 | .03 | .01 | .01 | .01 | 0 | 0 | 0 | .01 | .01 | 0 | 0 |
| Key 3.1 | | | | | | | | | | | | | | | | | | |
| P | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | .5 | .5 | 1 | 0 | 0 | 0 | 0 | 1 | 1 |
| F | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Key 3.2 | | | | | | | | | | | | | | | | | | |
| P | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R | 0 | 0 | 0 | 0 | .5 | .5 | 0 | 0 | 0 | .5 | 1 | 1 | 0 | 0 | .5 | 1 | 1 | 1 |

| | | | | | | | | | | | | | | | | | | |
|----------------|---|---|---|---|----|----|---|---|---|----|----|----|---|---|---|---|---|---|
| F | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | |
| Key 3.3 | | | | | | | | | | | | | | | | | | |
| P | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | |
| R | 0 | 0 | 0 | 0 | .5 | .5 | 0 | 0 | 0 | .5 | .5 | .5 | 0 | 0 | 0 | 0 | 1 | 1 |
| F | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

$$Y = .10 + .30 + .30 + .30$$

| | | | | | | | | | | | | | | | | | | |
|----------------|---|---|-----|-----|-----|-----|-----|-----|-----|-----|-----|----|---|-----|-----|-----|---|---|
| Key 1.1 | | | | | | | | | | | | | | | | | | |
| P | 0 | 0 | 0 | 0 | 0 | 0 | 0 | .01 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R | 0 | 0 | 0 | 0 | 1 | 1 | 0 | .5 | .5 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 1 |
| F | 0 | 0 | 0 | 0 | 0 | 0 | 0 | .01 | 0 | 0 | 0 | 0 | 0 | .01 | 0 | 0 | 0 | 0 |
| Key 1.2 | | | | | | | | | | | | | | | | | | |
| P | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R | 0 | 0 | 0 | 0 | .5 | .5 | 0 | 0 | 0 | .5 | 1 | 1 | 0 | .5 | .5 | 1 | 1 | 1 |
| F | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | .01 | 0 | 0 | 0 | 0 |
| Key 1.3 | | | | | | | | | | | | | | | | | | |
| P | 0 | 0 | .01 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R | 0 | 0 | .5 | .5 | 1 | 1 | 0 | 0 | .5 | 1 | 1 | 1 | 0 | .5 | 1 | 1 | 1 | 1 |
| F | 0 | 0 | .01 | .01 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Key 2.1 | | | | | | | | | | | | | | | | | | |
| P | 0 | 0 | 0 | 0 | .01 | 0 | .11 | .01 | .01 | .01 | 0 | 0 | 0 | .01 | .01 | 0 | 0 | 0 |
| R | 0 | 0 | 0 | 0 | .75 | 1 | .13 | .13 | .63 | .75 | 1 | 1 | 0 | .5 | .75 | .88 | 1 | 1 |
| F | 0 | 0 | 0 | 0 | .01 | .01 | .12 | .02 | .02 | .01 | .01 | 0 | 0 | .02 | .01 | .01 | 0 | 0 |
| Key 2.2 | | | | | | | | | | | | | | | | | | |
| P | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R | 0 | 0 | 0 | .25 | .88 | .88 | 0 | 0 | .38 | .88 | 1 | 1 | 0 | 0 | .5 | .88 | 1 | 1 |
| F | 0 | 0 | 0 | .01 | .01 | .01 | 0 | 0 | .01 | .01 | .01 | 0 | 0 | 0 | .01 | .01 | 0 | 0 |
| Key 2.3 | | | | | | | | | | | | | | | | | | |
| P | 0 | 0 | 0 | 0 | .01 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R | 0 | 0 | 0 | .25 | .88 | .88 | 0 | 0 | .25 | .5 | 1 | 1 | 0 | 0 | .38 | .88 | 1 | 1 |
| F | 0 | 0 | 0 | .01 | .01 | .01 | 0 | 0 | .01 | .01 | .01 | 0 | 0 | 0 | 0 | .01 | 0 | 0 |
| Key 3.1 | | | | | | | | | | | | | | | | | | |
| P | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | .5 | .5 | 1 | 0 | 0 | 0 | .5 | 1 | 1 |
| F | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Key 3.2 | | | | | | | | | | | | | | | | | | |
| P | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R | 0 | 0 | 0 | 0 | .5 | .5 | 0 | 0 | 0 | .5 | 1 | 1 | 0 | 0 | .5 | 1 | 1 | 1 |
| F | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Key 3.3 | | | | | | | | | | | | | | | | | | |
| P | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R | 0 | 0 | 0 | 0 | .5 | .5 | 0 | 0 | .5 | .5 | .5 | .5 | 0 | 0 | 0 | .5 | 1 | 1 |
| F | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | .01 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

$$Y = .10 + .20 + .35 + .35$$

| | | | | | | | | | | | | | | | | | | |
|----------------|---|---|---|-----|----|----|-----|-----|-----|-----|-----|---|---|-----|-----|-----|---|---|
| Key 1.1 | | | | | | | | | | | | | | | | | | |
| P | 0 | 0 | 0 | 0 | 0 | 0 | 0 | .01 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R | 0 | 0 | 0 | 0 | 1 | 1 | 0 | .5 | .5 | .5 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 1 |
| F | 0 | 0 | 0 | 0 | 0 | 0 | 0 | .01 | 0 | 0 | 0 | 0 | 0 | .01 | 0 | 0 | 0 | 0 |
| Key 1.2 | | | | | | | | | | | | | | | | | | |
| P | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R | 0 | 0 | 0 | 0 | 0 | .5 | 0 | 0 | 0 | .5 | 1 | 1 | 0 | .5 | .5 | 1 | 1 | 1 |
| F | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | .01 | 0 | 0 | 0 | 0 |
| Key 1.3 | | | | | | | | | | | | | | | | | | |
| P | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R | 0 | 0 | 0 | .5 | 1 | 1 | 0 | 0 | .5 | 1 | 1 | 1 | 0 | .5 | 1 | 1 | 1 | 1 |
| F | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Key 2.1 | | | | | | | | | | | | | | | | | | |
| P | 0 | 0 | 0 | .01 | 0 | 0 | .11 | .01 | .01 | 0 | 0 | 0 | 0 | .01 | .01 | 0 | 0 | 0 |
| R | 0 | 0 | 0 | .25 | .5 | 1 | .13 | .13 | .5 | .75 | .88 | 1 | 0 | .5 | .75 | .75 | 1 | 1 |

| | | | | | | | | | | | | | | | | | | |
|----------------------------------|---|---|-----|-----|-----|-----|-----|-----|-----|-----|-----|----|---|-----|-----|-----|---|---|
| F | 0 | 0 | 0 | .01 | .01 | .01 | .12 | .02 | .02 | .01 | .01 | 0 | 0 | .02 | .01 | .01 | 0 | 0 |
| Key 2.2 | | | | | | | | | | | | | | | | | | |
| P | 0 | 0 | 0 | .01 | 0 | 0 | 0 | 0 | .01 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R | 0 | 0 | 0 | .5 | .88 | .88 | 0 | 0 | .63 | .88 | 1 | 1 | 0 | 0 | .38 | .75 | 1 | 1 |
| F | 0 | 0 | 0 | .01 | .01 | .01 | 0 | 0 | .01 | .01 | .01 | 0 | 0 | 0 | .01 | 0 | 0 | 0 |
| Key 2.3 | | | | | | | | | | | | | | | | | | |
| P | 0 | 0 | 0 | 0 | .01 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R | 0 | 0 | 0 | .25 | .88 | .88 | 0 | 0 | .13 | .75 | 1 | 1 | 0 | 0 | .38 | .75 | 1 | 1 |
| F | 0 | 0 | 0 | .01 | .01 | .01 | 0 | 0 | .01 | .01 | .01 | 0 | 0 | 0 | .01 | 0 | 0 | 0 |
| Key 3.1 | | | | | | | | | | | | | | | | | | |
| P | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | .5 | .5 | 1 | 0 | 0 | 0 | 0 | 1 | 1 |
| F | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Key 3.2 | | | | | | | | | | | | | | | | | | |
| P | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R | 0 | 0 | 0 | 0 | .5 | .5 | 0 | 0 | 0 | .5 | 1 | 1 | 0 | 0 | .5 | 1 | 1 | 1 |
| F | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Key 3.3 | | | | | | | | | | | | | | | | | | |
| P | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R | 0 | 0 | 0 | 0 | .5 | .5 | 0 | 0 | 0 | .5 | .5 | .5 | 0 | 0 | 0 | .5 | 1 | 1 |
| F | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Y = .10 + .40 + .25 + .25 | | | | | | | | | | | | | | | | | | |
| Key 1.1 | | | | | | | | | | | | | | | | | | |
| P | 0 | 0 | 0 | 0 | 0 | 0 | 0 | .01 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R | 0 | 0 | 0 | .5 | 1 | 1 | 0 | .5 | .5 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 1 |
| F | 0 | 0 | 0 | 0 | 0 | 0 | 0 | .01 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Key 1.2 | | | | | | | | | | | | | | | | | | |
| P | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R | 0 | 0 | 0 | 0 | .5 | .5 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | .5 | .5 | 1 | 1 | 1 |
| F | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Key 1.3 | | | | | | | | | | | | | | | | | | |
| P | 0 | 0 | .01 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R | 0 | 0 | .5 | .5 | 1 | 1 | 0 | 0 | .5 | 1 | 1 | 1 | 0 | .5 | 1 | 1 | 1 | 1 |
| F | 0 | 0 | .02 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Key 2.1 | | | | | | | | | | | | | | | | | | |
| P | 0 | 0 | 0 | 0 | .01 | 0 | .11 | .01 | .01 | 0 | 0 | 0 | 0 | .01 | 0 | 0 | 0 | 0 |
| R | 0 | 0 | 0 | 0 | .75 | 1 | .13 | .13 | .38 | .75 | 1 | 1 | 0 | .5 | .63 | 1 | 1 | 1 |
| F | 0 | 0 | 0 | 0 | .01 | .01 | .12 | .02 | .01 | .01 | .01 | 0 | 0 | .02 | .01 | .01 | 0 | 0 |
| Key 2.2 | | | | | | | | | | | | | | | | | | |
| P | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R | 0 | 0 | 0 | .13 | .88 | .88 | 0 | 0 | .5 | 1 | 1 | 1 | 0 | 0 | .5 | .88 | 1 | 1 |
| F | 0 | 0 | 0 | 0 | .01 | .01 | 0 | 0 | .01 | .01 | .01 | 0 | 0 | 0 | .01 | .01 | 0 | 0 |
| Key 2.3 | | | | | | | | | | | | | | | | | | |
| P | 0 | 0 | 0 | 0 | .01 | 0 | 0 | .01 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R | 0 | 0 | 0 | .25 | .88 | .88 | 0 | .13 | .25 | .5 | 1 | 1 | 0 | 0 | .38 | .75 | 1 | 1 |
| F | 0 | 0 | 0 | .01 | .01 | .01 | 0 | .03 | .01 | .01 | .01 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Key 3.1 | | | | | | | | | | | | | | | | | | |
| P | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | .5 | .5 | 1 | 0 | 0 | 0 | .5 | 1 | 1 |
| F | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Key 3.2 | | | | | | | | | | | | | | | | | | |
| P | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R | 0 | 0 | 0 | .5 | .5 | .5 | 0 | 0 | 0 | .5 | 1 | 1 | 0 | 0 | .5 | .5 | 1 | 1 |
| F | 0 | 0 | 0 | .01 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Key 3.3 | | | | | | | | | | | | | | | | | | |
| P | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R | 0 | 0 | 0 | .5 | .5 | .5 | 0 | 0 | .5 | .5 | .5 | .5 | 0 | 0 | 0 | .5 | 1 | 1 |
| F | 0 | 0 | 0 | .01 | 0 | 0 | 0 | 0 | .01 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

Table A.1: Information retrieval measures for features with no feature correspondence variation, feature correspondence variation of 1 and feature correspondence variation of 2

Appendix B

Accuracy of Matching Algorithms

P = Precision

R = Recall

F = F-Score

| Threshold | 100 | | | 80 | | | 60 | | | 40 | | | 20 | | | 0 | | |
|----------------|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|---|-----|-----|---|-----|
| | P | R | F | P | R | F | P | R | F | P | R | F | P | R | F | P | R | F |
| Key 1.1 | | | | | | | | | | | | | | | | | | |
| DIF | 1 | .5 | .67 | 1 | 1 | 1 | 1 | 1 | 1 | .25 | 1 | .4 | .4 | 1 | .08 | .02 | 1 | .03 |
| XOR | 0 | 0 | 0 | .13 | .5 | .2 | .08 | 1 | .15 | .04 | 1 | .08 | .02 | 1 | .04 | .02 | 1 | .03 |
| EDM | 0 | 0 | 0 | .17 | .5 | .25 | .1 | 1 | .18 | .04 | 1 | .09 | .02 | 1 | .04 | .02 | 1 | .03 |
| DTW | 1 | .5 | .67 | 1 | .5 | .67 | 1 | .5 | .67 | .25 | 1 | .4 | .04 | 1 | .08 | .02 | 1 | .03 |
| Key 1.2 | | | | | | | | | | | | | | | | | | |
| DIF | 0 | 0 | 0 | 0 | 0 | 0 | .04 | .5 | .08 | .04 | 1 | .07 | .02 | 1 | .04 | .02 | 1 | .03 |
| XOR | 0 | 0 | 0 | 0 | 0 | 0 | .07 | 1 | .13 | .03 | 1 | .06 | .02 | 1 | .03 | .02 | 1 | .03 |
| EDM | 0 | 0 | 0 | 0 | 0 | 0 | .04 | .5 | .07 | .03 | 1 | .06 | .02 | 1 | .03 | .02 | 1 | .03 |
| DTW | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | .03 | 1 | .06 | .02 | 1 | .03 | .02 | 1 | .03 |
| Key 1.3 | | | | | | | | | | | | | | | | | | |
| DIF | 1 | .5 | .67 | .2 | .5 | .29 | .09 | 1 | .17 | .04 | 1 | .08 | .02 | 1 | .04 | .02 | 1 | .03 |
| XOR | 0 | 0 | 0 | 0 | 0 | 0 | .06 | 1 | .11 | .03 | 1 | .06 | .02 | 1 | .03 | .02 | 1 | .03 |
| EDM | 0 | 0 | 0 | .09 | .5 | 0 | .07 | 1 | .13 | .03 | 1 | .06 | .02 | 1 | .03 | .02 | 1 | .03 |
| DTW | 0 | 0 | 0 | 0 | 0 | 0 | .07 | .5 | .12 | .04 | 1 | .07 | .02 | 1 | .04 | .02 | 1 | .03 |
| Key 2.1 | | | | | | | | | | | | | | | | | | |
| DIF | 1 | .13 | .22 | .83 | .63 | .71 | .14 | .63 | .23 | .08 | 1 | .14 | .06 | 1 | .12 | .06 | 1 | .12 |
| XOR | 0 | 0 | 0 | .09 | .38 | 0 | .09 | .88 | .16 | .07 | 1 | .13 | .06 | 1 | .12 | .06 | 1 | .12 |
| EDM | 0 | 0 | 0 | 0 | 0 | 0 | .1 | .75 | .17 | .07 | 1 | .13 | .06 | 1 | .12 | .06 | 1 | .12 |
| DTW | 1 | .13 | .22 | 1 | .13 | .22 | 1 | .13 | .22 | .07 | .38 | .12 | .07 | 1 | .14 | .06 | 1 | .12 |
| Key 2.2 | | | | | | | | | | | | | | | | | | |
| DIF | 0 | 0 | 0 | .17 | .13 | .14 | .14 | .63 | .23 | .07 | .88 | .13 | .06 | 1 | .12 | .06 | 1 | .06 |
| XOR | 0 | 0 | 0 | 0 | 0 | 0 | .1 | .75 | .18 | .07 | 1 | .13 | .06 | 1 | .12 | .06 | 1 | .12 |
| EDM | 0 | 0 | 0 | 0 | 0 | 0 | .12 | .63 | .2 | .08 | 1 | .15 | .06 | 1 | .12 | .06 | 1 | .12 |
| DTW | 0 | 0 | 0 | .29 | .25 | .27 | .18 | .75 | .29 | .09 | 1 | .17 | .07 | 1 | .12 | .06 | 1 | .12 |
| Key 2.3 | | | | | | | | | | | | | | | | | | |
| DIF | 0 | 0 | 0 | 0 | 0 | 0 | .1 | .25 | .14 | .07 | .88 | .14 | .06 | 1 | .12 | .06 | 1 | .06 |
| XOR | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | .07 | 1 | .14 | .06 | 1 | .12 | .06 | 1 | .12 |
| EDM | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | .09 | .75 | .16 | .07 | 1 | .12 | .06 | 1 | .12 |
| DTW | 0 | 0 | 0 | .08 | .13 | .1 | .09 | .5 | .15 | .07 | 1 | .14 | .06 | 1 | .12 | .06 | 1 | .12 |
| Key 3.1 | | | | | | | | | | | | | | | | | | |
| DIF | 0 | 0 | 0 | 0 | 0 | 0 | .04 | 1 | .07 | .02 | 1 | .03 | .02 | 1 | .03 | .02 | 1 | .03 |

| | | | | | | | | | | | | | | | | | | |
|----------------|---|---|---|-----|----|-----|-----|----|-----|-----|---|-----|-----|---|-----|-----|---|-----|
| XOR | 0 | 0 | 0 | 0 | 0 | 0 | .02 | 1 | .04 | .01 | 1 | .02 | .02 | 1 | .03 | .02 | 1 | .03 |
| EDM | 0 | 0 | 0 | 0 | 0 | 0 | .02 | 1 | .04 | .02 | 1 | .03 | .02 | 1 | .03 | .02 | 1 | .03 |
| DTW | 0 | 0 | 0 | .05 | .5 | .08 | .03 | 1 | .05 | .02 | 1 | .03 | .02 | 1 | .03 | .02 | 1 | .03 |
| Key 3.2 | | | | | | | | | | | | | | | | | | |
| DIF | 0 | 0 | 0 | 0 | 0 | 0 | .03 | .5 | .06 | .02 | 1 | .04 | .02 | 1 | .03 | .02 | 1 | .03 |
| XOR | 0 | 0 | 0 | 0 | 0 | 0 | .01 | .5 | .03 | .01 | 1 | .02 | .02 | 1 | .03 | .02 | 1 | .03 |
| EDM | 0 | 0 | 0 | 0 | 0 | 0 | .02 | 1 | .04 | .02 | 1 | .03 | .02 | 1 | .03 | .02 | 1 | .03 |
| DTW | 0 | 0 | 0 | .33 | .5 | .4 | .1 | .5 | .17 | .03 | 1 | .06 | .02 | 1 | .04 | .02 | 1 | .03 |
| Key 3.3 | | | | | | | | | | | | | | | | | | |
| DIF | 0 | 0 | 0 | 0 | 0 | 0 | .04 | .5 | .07 | .02 | 1 | .03 | .02 | 1 | .03 | .02 | 1 | .03 |
| XOR | 0 | 0 | 0 | 0 | 0 | 0 | .02 | 1 | .03 | .01 | 1 | .02 | .02 | 1 | .03 | .02 | 1 | .03 |
| EDM | 0 | 0 | 0 | 0 | 0 | 0 | .02 | 1 | .05 | .02 | 1 | .03 | .02 | 1 | .03 | .02 | 1 | .03 |
| DTW | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | .02 | 1 | .04 | .02 | 1 | .03 | .02 | 1 | .03 |

Table B.1: Information retrieval measures for various matching algorithms

Appendix C

User Testing Questionnaire and Feedback

BOLD Translator User Evaluation

The purpose of this evaluation is to gain an understanding about how users feel about the BOLD Translator in terms of its usability and usefulness. You are requested to follow the instruction in this document and provide feedback in certain places. When providing feedback, please provide as much information as possible about how you feel about the system and the ways in which it might be useful.

Experiment: Selection a Word

1. Select the word *thu* which appears on the notebook page (the location of this word will be pointed out to you).
2. When selecting the word try to select only the word and nothing else. Try to fit the selection tool as closely around the word as possible and avoid including marks which are not part of the word in your selection.
3. Press the Search! button.
4. Look at the results which have been returned, keeping in mind that you are trying to find occurrences of the word *thu* in the results.
5. Click on the Boer (Bleek) link on the left side of the page and repeat this process but this time try and select the word *thu* in a different way.

Feedback

1. What did you think about the method of selecting words on a page?
2. How would you improve it?
3. What do you think about the way that results are displayed on the page?
4. Would you display them in any other way?

5. How accurate do you feel the system was in finding occurrences of the word *thu*?
6. Based on the results which have been returned, are you able to determine what the word *thu* means in English?
7. Do you think a system of this nature might be valuable to researchers who are studying Bushman languages?
8. If you were to build a system of this nature, how would you do it differently?

Thank you for your feedback!

Feedback: User 1

Age: 23

Occupation: *Musician/Producer*

Years using computers:

1. What did you think about the method of selecting words on a page?
It was user friendly and basic to use. Familiar method of selecting pictures, ie. cropping.
 2. How would you improve it?
Think it's fine.
 3. What do you think about the way that results are displayed on the page?
A bit confusing if you don't understand how to find what you're looking for.
 4. Would you display them in any other way?
Possibly vertically. Easier for the eye to spot the exact word you're looking for.
 5. How accurate do you feel the system was in finding occurrences of the word *thu*?
Accurate enough.
 6. Based on the results which have been returned, are you able to determine what the word *thu* means in English?
Yes.
 7. Do you think a system of this nature might be valuable to researchers who are studying Bushman languages? *Definitely. Just that the words are hard to make out.*
 8. If you were to build a system of this nature, how would you do it differently?
No clue.
-

Feedback: User 2

Age: 17

Occupation: *Student*

Years using computers: *5 years*

1. What did you think about the method of selecting words on a page?
It is a fair enough method, however, it can be a mission trying to exclude other squiggly lines.
 2. How would you improve it?
Maybe make it so that each word can be auto-selected when pressed on. This is less of a mission and more efficient.
 3. What do you think about the way that results are displayed on the page?
Instead of a number of results that need to be searched through. Have one result per a word. This will be easier if each word is on auto-select, because there can't be more than one possibility.
 4. Would you display them in any other way?
Yes. In standard letters that are easy to read, or if not - bigger. I spent a while searching to find the correct result. Again, if there is only one, no need to search.
 5. How accurate do you feel the system was in finding occurrences of the word *thu*?
Semi-accurate. It found "thu" but also found a number of random words that don't even look similar.
 6. Based on the results which have been returned, are you able to determine what the word *thu* means in English?
No, it says that it is from "DICTIONARY_ENGLISH_B_(number)" and says how many times it is featured and a score. But not English translation.
 7. Do you think a system of this nature might be valuable to researchers who are studying Bushman languages?
Yes, it is a quick and easy way to search words but would be more useful with some of my suggestions.
 8. If you were to build a system of this nature, how would you do it differently?
I would do it the way I suggested. At this point, the system requires that the person does a lot. Rather than the program itself doing the work which is the point.
-

Feedback: User 3

Age: 46

Occupation: *Research Project Manager*

Years using computers: *+20 years*

1. What did you think about the method of selecting words on a page?
Works relatively well, but rectangular selection tool does not sufficiently allow for "fine tuning".
 2. How would you improve it?
Not sure - is there a way of angling a selection tool?.
 3. What do you think about the way that results are displayed on the page?
I had 30 results, but was unsure of which were relevant to my search. May just have to do with search itself, ie. the word.
 4. Would you display them in any other way?
No - think the rows and columns work fine.
 5. How accurate do you feel the system was in finding occurrences of the word *thu*?
Had difficulty matching the actual results, so can't really tell.
 6. Based on the results which have been returned, are you able to determine what the word *thu* means in English?
Not really. Found it hard to read the actual text.
 7. Do you think a system of this nature might be valuable to researchers who are studying Bushman languages?
Certainly, but it does require some knowledge of the language and writing style of the day.
 8. If you were to build a system of this nature, how would you do it differently?
Perhaps add a feature where the initial results are displayed more legibly.
-

Feedback: User 4

Age: 50

Occupation: *Director*

Years using computers: *21 years*

1. What did you think about the method of selecting words on a page?
I liked the box selector but didn't see it in the beginning as it was too far to the left.
2. How would you improve it?
Put it in the middle/centre of the book more in view of the eye. Do initial training session with vocal instructions.
3. What do you think about the way that results are displayed on the page?
Not clear what the results mean. I expected to find translations but got too much on the screen and only later realised that there was 1 record that was correct with the right translation.
4. Would you display them in any other way?
Some of the results did not make sense and look like "thu", eg no. 17.
5. How accurate do you feel the system was in finding occurrences of the word *thu*?
No. 4 is the only pure occurrence.

6. Based on the results which have been returned, are you able to determine what the word *thu* means in English?

Boer.

7. Do you think a system of this nature might be valuable to researchers who are studying Bushman languages?

Yes a good way of training the eye to select the word as it help the "eye" to see it.

8. If you were to build a system of this nature, how would you do it differently?

It worked for me.
