A Two-stage Approach for Word Searching in Handwritten Document Images

A thesis

submitted in partial fulfillment of the requirement for the Degree of

Master of Technology in Computer Technology of

Jadavpur University

By

Pronita Mukherjee

Registration No: 137125 of 2016-2017

Examination Roll No: M6TCT19022

Under the Guidance of

Dr. Ram Sarkar

Associate Professor

Department of Computer Science and Engineering

Jadavpur University, Kolkata-700032

India

2019

FACULTY OF ENGINEERING AND TECHNOLOGY
JADAVPUR UNIVERSITY

Certificate of Recommendation

This is to certify that the dissertation entitled "A Two-stage Approach for Word Searching in Handwritten Document Images" has been carried out by Pronita Mukherjee (University Registration No: 137125 of 2016-2017, Examination Roll No: M6TCT19022) under my guidance and supervision and be accepted in partial fulfillment of the requirement for the Degree of Master of Technology in Computer Technology. The research results presented in the thesis have not been included in any other paper submitted for the award of any degree in any other University or Institute.

……

Dr. Ram Sarkar (Thesis Supervisor)
Associate Professor
Department of Computer Science and Engineering
Jadavpur University, Kolkata-32

Countersigned

……

Dr. Mahantapas Kundu
Professor & Head, Department of Computer Science and Engineering,
Jadavpur University, Kolkata-32.

……

Prof. Chiranjib Bhattacharjee
Dean, Faculty of Engineering and Technology,
Jadavpur University, Kolkata-32.

FACULTY OF ENGINEERING AND TECHNOLOGY
JADAVPUR UNIVERSITY

Certificate of Approval*

This is to certify that the thesis entitled "A Two-stage Approach for Word Searching in Handwritten Document Images" is a bonafide record of work carried out by Pronita Mukherjee in partial fulfillment of the requirements for the award of the degree of Master of Technology in Computer Technology in the Department of Computer Science and Engineering, Jadavpur University during the period of July 2016 to June 2019. It is understood that by this approval the undersigned do not necessarily endorse or approve any statement made, opinion expressed or conclusion drawn therein but approve the thesis only for the purpose for which it has been submitted.

……..

Signature of Examiner 1

Date:

……..

Signature of Examiner 2

Date:

*Only in case the thesis is approved

FACULTY OF ENGINEERING AND TECHNOLOGY
JADAVPUR UNIVERSITY

Declaration of Originality and Compliance of Academic Ethics

I hereby declare that this thesis entitled "A Two-stage Approach for Word Searching in Handwritten Document Images" contains literature survey and original research work by the undersigned candidate, as part of her Degree of Master of Technology in Computer Technology.

All information has been obtained and presented in accordance with academic rules and ethical conduct.

I also declare that, as required by these rules and conduct, I have fully cited and referenced all materials and results that are not original to this work.

Name: Pronita Mukherjee
Registration No: 137125 of 2016-2017
Exam Roll No: M6TCT19022
Thesis Title: A Two-stage Approach for Word Searching in Handwritten Document Images

……..
Signature with Date

Acknowledgement

I would like to start by thanking the holy trinity for helping me deploy all the right resources and for shaping me into a better human being.

I would like to express my deepest gratitude to my advisor, **Dr. Ram Sarkar**, Associate Professor, Department of Computer Science and Engineering, Jadavpur University for his admirable guidance, care, patience and for providing me with an excellent atmosphere for doing research.

I would like to thank **Dr. Mahantapas Kundu**, Professor & Head, Department of Computer Science and Engineering, Jadavpur University, for providing me with moral support at times of need. I would also like to thank **Prof. Mita Nasipuri**, Department of Computer Science and Engineering, Jadavpur University for her amazing psychological assistance and guidance.

I would like to thank my senior Mr. Samir Malakar without whom this thesis could not be completed as he played a major role in educating me about technical aspect. I would like to specially mention Mr. Nirmal Das, who provided me with the template for this thesis.

Most importantly none of this would have been possible without the love and support of my family, especially my mother for her constant motivation and my bambino sacrificed unbelievably.

This thesis would not have been completed without the inspiration and support of a number of wonderful individuals — my thanks and appreciation to all of them for being part of this journey and making this thesis possible.

……..
Pronita Mukherjee
Registration No: 137125 of 2016-2017
Exam Roll No: M6TCT19022
Department of Computer Science & Engineering
Jadavpur University

Table of Content

List of Figures

List of Tables

Chapter 1

1. Introduction

Use of handwritten paper documents is still playing an importance role, despite growing use of electronic documents in our day to day life. Current technologies allow convenient and inexpensive means to capture, store, compress and transfer digitized images of documents. But, the process of semi-automatic document image processing requires specialized technology to extract document's contents correctly. Retrieval of information from Digital Libraries is primarily done by using typed textual queries.

The main purpose of Document Image Analysis is the information retrieval properly from document images which contain either textual or pictorial or structural information. The correct understanding of such information represents a step forward towards shortening the semantic gap between recognizing individual visual objects and understanding the whole document content in a given context. It does not involve in pure transcription of documents, but the retrieval and the linkage of semantic knowledge from large collections of document images stored in digital repositories.

There are lots of historical handwritten documents, which can be used for several projects and studies. There are two ways available to extract the information:

- Transcribing documents (word-to-word)
- Word spotting.

The DIAR community is interested in preserving these documents and wants to extract all valuable information from these documents. Apart from these, office automation also demands digitized storage, manipulation, retrieval of documents in electronic format, i.e. handwritten documents can be managed properly. The solution of this is to transfigure document images into electronic form and then process the same with an optical character recognition (OCR) engine [1]. The current handwritten OCR work ailing for large lexicon sizes [2]. The alternative solution is to retain the documents in digital form with proper tagging.

1.1 Optical Character Recognition

OCR is the mechanical or electronic conversion of images of typed, handwritten or printed text into machine-encoded text, whether from a scanned document, a photo of a document. It is a common method of digitizing documents so that they can be electronically edited, stored more compactly, and used in machine translation, text-to-speech conversion, data and text mining. OCR is a field of research in artificial intelligence, pattern recognition and computer vision.OCR system is used to convert physical documents into machine-readable text. It is a type of software that can automatically evaluate scanned text document and fit it into a form that a computer can process more easily [3]. OCR is the heart of everything from handwriting analysis. Once placed in this soft copy, users can edit, format and search the document as if it was created with a word processor.

Suppose there was only one alphabet: A. Then, it can be seeing that OCR would face a tricky problem—because every person writes the letter A in a marginally different way. Even with printed text, there is an issue, because many different typefaces (fonts) and the letter A can be printed in many subtly different forms.

Figure 1.1: A fair bit of variation of a capital letter A, basic similarity: almost all of them are prepared from two angled lines that meet in the middle at the top with a horizontal line between.

Broadly idiom, there are two altered ways to solve this problem, either by recognizing characters in their entirety i.e. pattern recognition or by detecting the individual lines and strokes characters are made from feature detection and identifying them.

- *Pattern recognition* – OCR programs are fed examples of text in various fonts and formats which are then used to compare, and recognize, characters in the scanned document.

- *Feature detection* – OCR programs apply rules regarding the features of a specific letter or number to recognize characters in the scanned document. Features could include the number of angled lines, crossed lines or curves in a character for comparison. For example, the capital letter "A" may be stored as two diagonal lines that meet with a horizontal line across the middle.

When a character is identified, it is converted into an ASCII code that can be used by computer systems to handle further manipulations. Users should be able to correct basic errors, proof-read and make sure complex layouts were handled properly before saving the document for future use.

1.2 Problem with Optical Character Recognition

OCR works accurately for machine printed text. But, OCR performance is unstable, when documents contain either handwritten text or symbols or graphical structures. Though OCR is one of the ways to collect and analyze large volume of physical (paper) data quickly, it can still be incredibly difficult and time consuming to use. It must be assured that the document is in a language, and the OCR software can recognizes; not all engines are trained to recognize all languages [4]. Low contrast in documents can diminish OCR accuracy; contrast can be attuned in a photo manipulation tool. Text created earlier to 1850 or by a typewriter can be more challenging for OCR software to read. OCR software is unable to read handwriting; while the handwritten notes are digitized.

Technologies overall have been advanced over the past few years, OCR technology really hasn't changed in over a decade. There hasn't been any driving force behind the adoption of new technology because OCR works just well enough to be acceptable for machine printed text. Unfortunately, this has left many users to become accustomed to lengthy turnaround times and to view them as just "part of the process". As long as, there is limited demand for faster OCR, there is no momentum to improve upon the OCR technology. As a consequence, conventional OCR technology is both slow and unpredictable. For a technology that is intended to improve upon the speed of document management processes, slow speeds can be exceptionally problematic and the lack of accuracy in OCR can also present challenges.

OCR requires large amounts of both technical and human resources in word spotting. It will often require huge volumes of memory and processing speed, which slow down the system and makes it more difficult to scan large volumes of documents. OCR tends to have high levels of inaccuracy, especially with low quality documents, such as, handwritten text or handwritten manuscripts or symbols or graphical structures. Hence OCR requires more and more manual review to verify the outcome results.

By considering the above issues with OCR, it is still a big scientific challenge to retrieve information from documents, which contain information in the form of either handwritten text or symbols or graphical structures.

1.3 Applications

There is an increasing interest to preserve historical documents digitally and to provide access to users to historical document collections in libraries, museums and archives. Conversion of historical documents to digital records is of major importance to society, both in terms of information accessibility and long-term preservation. Handwritten documents are found in historical document collections.

- Unique manuscripts written by well-known scientists, artists or writers;
- Trade forms or administrative documents kept by rural community or municipalities.
- Different types of map.
- Patient dieses background documents, which include prescription, reports etc.
- Government's critical files.

OCR engines have been advanced into many kinds of domain-specific OCR applications, such as receipt OCR, invoice OCR, and check OCR, legal billing document OCR. They can be used for,

- Data entry for business documents, e.g. check, passport, invoice, bank statement and receipt. Automatic number plate recognition.
- In airports, for passport recognition and information extraction.
- Automatic insurance documents key information extraction.
- Extracting business card information into a contact list.
- Converting handwriting in real time to control a computer (pen computing)
- Assistive technology for blind and visually impaired users

1.4 Categorization

Handwriting word spotting is a pattern recognition task that consists in detecting words present in handwriting document images. In this thesis, the documents, where all pages are written by the same author or few different authors, same word images of multiple instances are likely to look almost similar. Word spotting [5] treats a collection of documents as a collection of words.

Depending on how the input is specified, there exist two types of word-spotting approaches:

- Query-by-string
- Query-by-example.

In *query-by-string*, character models have been trained in advance and the character models are combined to form words during time of execution and the probability of each word is evaluated. On the other hand, in *query-by-example*, the input is an image of word to search and the output is a set of the most representative images of the query word.

Word spotting model can be classified into two namely like segmentation based approach [6] and segmentation-free approach [8] approach. First one is based on pre-segmented word or text-line. The second one always tries to search the word in the document image without page segmentation. In recognition-based approach, the off line sample word are used for preparing the training model. On the contrary, recognition free approaches always try to match the search word with the help of different matching techniques. The present work is based on *query-by-example*, recognition-free word-spotting technique.

1.5 Challenges

Centuries ago, the ink was used to write, had some oxide particles, which was contributed to degrade the paper of the document and caused damage in the words as well. This effect is known as bleed through. Nowadays, some methods have been developed for improving the quality of the images [7, 8].

Document degradation over time is a challenge for word spotting. Degradation of documents can be caused due to lifetime usage. Degradation can appear for several reasons:

- Non stationary noise due to illumination changes,
- Curvature of the document,
- Ink and holes in the document,
- Ink show through is the appearance of text or graphics on scanned image back side,
- Low contrast,
- Warping effect.

Handwritten word-spotting refers to the problem of identifying specific keywords in handwritten document images.

1.6 Motivation

In recent years there has been an increasing interest in digitizing the vast amounts of pre-digital age handwritten documents that exist throughout the world. Many of the emerging digitizing initiatives are aimed at dealing with huge collections of handwritten documents, for which automatic recognition is not yet as mature as for printed text Optical Character Recognition (OCR).

There are vast collections of historical handwritten documents available that contain a plenty of valuable information. The extraction of this information has become a crucial task among the document analysis researchers and practitioners. Preserving of these documents in digital form is the need of current circumstances. But, only digitalization of these documents might not fulfill user's demand. Over the past few decades, there has been growing interest in addressing handwritten document information extraction by using word spotting, which is getting reflected by continuously increasing number of approaches. However, there are very few comprehensive studies exist, which analyze the various aspects of different word spotting techniques.

The recognition free approach extracts geometrical/shape of word image [9] and then some similarity metric for word matching is applied. Though this approach is fast, it retrieves more inappropriate words with respect to the search word. On the other hand, as matching techniques is applied in recognition-based approach [10], tried to spot a query word in target document images by identifying all the words present in the document. Thus, these approaches not only consume more time, but also spot more irrelevant target words.

Keeping these facts in the mind, a novel approach is proposed in this thesis. In this work, a pre-selection is performed beforehand that gets the benefits of the recognition-free word spotting technique and there after a matching schema is used which is recognition-based model. In the present work, an approach of matching a search word with target word is modeled. The experiment is carried out on handwritten English document database, called "QUWI" database, which was used in the writer identification competition in the International Conference on ICDAR 2015 [11]. It is a two-stage approach, the first stage is the pre-selection of search words and in the second stage a voting schema is performed over the five distance-matching scores. First one is the voting system where two or more distance-matching scores, vote for positivity of the word as a target word, like wise second one performing matching-score for more than three, whereas in the last case if three distance metrics vote for positivity as the target word the model store the same in the target folder. At the end, a comparative study is depicted for different voting systems. The aim of this current work is to review the recent approaches as well as to fill the research gaps found in the literature survey.

1.7 Organization of the Thesis

In *Chapter1*, a general introduction is presented along with the idea of OCR System, its application, categorization as well as short-falls. This chapter ends with the challenges and motivation of the present work done.

In *Chapter 2*, Literature Survey is presented in detail.

In *Chapter 3*, Methods and Methodologies applied to implement the present work are described in details with some examples to understand the ideas described therein.

In *Chapter 4*, the experimental results are depicted in tabular format along with the plotted graph to get the visual idea of the results obtained in the different voting procedure.

In *Chapter 5*, the present work is concluded along with the feature scope of the present model to improve the performance of it.

Chapter 2

2. Literature Survey

From literature survey, it is perceived that, till to date, many researchers have applied numerous distance-based approaches [12, 13] for probing a word from a document image following the QbE way. Word-spotting was originally introduced to detect words in speech messages. Later, it was used in text documents for matching and indexing of words. In this context, it was first suggested in [14], which was written by Manmatha and later, a good numbers of different word matching [15, 16] algorithms were explored.

In [16], Rath et al, proposed an automatic retrieval system for historical handwritten documents, which consists of two different statistical models to retrieve words (within a text query) from large collection of handwritten manuscripts. Both statistical models were using a set of transcribed page images to study a joint probability distribution in-between feature computed from word images and their transcriptions. Later, this automatic retrieval system was used to retrieve handwritten documents unlabeled images as well.

In generic word spotting approach, the image of word is not segmented into smaller parts, but it is considered as a whole shape. Thus, the word matching recognition is using shape matching algorithm to perform, in terms of the features of images can be calculated at some key interest points. A recent comparative [7] study in between a number of interest detectors points is presented to show that corner can be detected with the Harris detector [17], but as always, such detector is having drawback of its sensitiveness and responsiveness to noise.

In [10], a five-stage process was prescribed for query word spotting in document images. At the beginning, both the query word and document image were processed through a preprocessing step for noise reduction. Document images are usually having noises due to, i) document quality, ii) document age and iii) scanning device imperfection. Therefore, Adaptive Thinning Framework (ATF) was used in preprocessing step for noise reduction and input normalization. 1-pixel width representation of images was produced by Adaptive Thinning Framework, which was more robust against noise compared with conventional thinning algorithms. In second step, image features were extracted from components and represented as feature vectors. Contour Points Distribution Histogram (CPDH) shape descriptor was used for feature extraction; where for each component of query and document image, a feature vector was extracted based on the shape points distribution within the shape enclosing circle in polar coordinates. Next, the point distribution was represented in a two-dimensional histogram. In third step, corresponding query component's feature vectors and document image component were matched to calculate similarity scores, stored in a similarity matrix. In fourth step, locations of candidate image occurrences are searched within document image by using

similarity matrix. In final step, relevant occurrences were detected and stored after filtering out of irrelevant patterns. This research was reported as an application-independent and segmentation-free approach for multipart queries in document images spotting. The proposed approach was used to find occurrences of a query in document image by introducing 5 steps, which remove irrelevant pixels by means of feature matching. Preliminary experimental results were promising in performance and there was enough possibility of further improvement.

In [11], feature extraction was done by using 'Column based features' and 'Slit style Histogram of Oriented Gradient (SSHOG) based features'. Eight statistical features are computed from left to right on each pixel columns for an image with a width of N pixels in 'Column-based features' extraction. On the other side, in SSHOG based features extraction; a fixed sized window slides over image in horizontal direction to extract HOG [17] features for each slit. For experiments, classical word spotting framework was used. Due to handwriting quality variability, words belong to the same category can often have different lengths and sizes. Two different pruning techniques were used to tackle this situation. One was based on area of bounding boxes and another one was a rough estimation of number of characters in word image. Irrelevant word images (with respect to size of query) were pruned by using simple properties of images for GW-90 and Bentham dataset before performing. Single, primitive and simple threshold values were used to avoid fine tuning of threshold. Finally, dynamic time warping method was used to identify the optimal warping path between two different time series. This warping path maintains the following three constraints, i) boundary conditions, ii) continuity and iii) monotonicity.

In [15], two-stage approach was proposed, which consist pre-selection of target words and confirmation of pre-selected word(s) as search word, had been introduced within document page images to search a word. HOG is a proven texture based feature descriptor and it computes gradient information at first for a cell (a primitive sub-block) in different directions. Secondly, a round of cell normalization is performed for each block to form a pattern. Each cell orientation measurement is performed by dividing total orientation angle (0°–360° or 0°–180°) into different ranges. These angle ranges are called "range" or "bin". HOG feature was extracted into b bins from a c × c cell size of image by considering 2c × 2c block size. Images were padded with zeros to convert image dimension in multiples of c, before applying HOG feature. Next, extracted information mean and standard deviation for each of b bins were considered as feature values, which resulted a feature vector of length 2 * b for each and every image. This modification helped to generate a feature vector of equal length by preserving actual size of image and this was reduced feature vectors dimension a certain level. These two features were extracted from an image including upper & lower parts of an image and then separated by principal and non-principal diagonal. In later phase, all these features were again extracted from all the sub-images to collect local information. A length 3 feature vector was extracted from all the words from a document image, to filter in only relevant word in search. To confirm the relevant filter in words as expected search word(s), a holistic word recognition

approach was used then, where, a feature vector, which comprising topological features and a modified HOG feature descriptor, was extracted from each word image, to classify with an Multi-Layer Perceptron classifier.

In [18], seven features based system was proposed, which extract every capable word similarities and discarding related differences due to noise or different style of fonts. The seven feature set were, i) width to height ratio, ii) word area density, iii) center of gravity, iv) vertical projection, v) top - bottom shape projections, vi) upper grid features and vii) down grid features. In first set, by considering word shape, width to height ratio of the word outline is important information. In second set, word area density feature represents black pixels percentage included in word bounding box. In third set, Center of gravity represents Euclidean distance from word's center of gravity (C.G) to the upper left corner of word bounding box. The vertical and horizontal center of gravity should be determined in order to calculate this. In fourth set, vertical projection feature consists of a vector with only 20 elements, extracted from word image after applying smoothing and vertical projection normalization. These elements were corresponded to first 20 coefficients of smoothed and normalized vertical projection of discrete cosine transform. In fifth set, top - bottom shape projections were considered as signature of word shape. These signatures consisted of 50 elements feature vector; where first 25 values were the first 25 coefficients of smoothed and normalized top shape projection discrete cosine transform and the rest 25 values were equal to first 25 coefficients of smoothed and normalized bottom shape projection discrete cosine transform. Word image was scanned from top to bottom in order to calculate top shape projection. All the following pixels of the same column were converted to black after first time a black pixel was found. The bottom shape projection is found similarly; word image was scanned from bottom to top and all the pixels were converted to black until a black pixel was found. In six set, upper grid features (UGF) was a 10 elements binary value vector extracted from upper part of each word image. Initially the image's horizontal projection is extracted, and from it, the upper part of the word is determined. In seven set, down grid features were similar to upper grid features, but down grid features were extracted from the lower part of word image. The down grid features were calculated by using the method of the upper grid features, but this time the search was starting from the bottom of the horizontal projection histogram. The output was again a 10 elements binary values vector.

In [8], an Arabic learning-based word spotting system was proposed. This proposed system was based on a hierarchical classifier, which integrates partial segmentation of lexicon words into Pieces of Arabic Words (PAWs), to spot or reject a word by using language models. The system was built on lexicon of Arabic handwritten words and tested with Arabic handwritten documents only. PAWs of lexicon words were re-grouped corresponding to their location within the word and each group was used to train a single classifier. This was resulted a sequence of classifiers that formed a hierarchical classifier. PAWs of document text lines were passed to this hierarchical classifier. Along with this, a set of graphs with confidence values above a predefined threshold, were created for the PAWs. Then, the paths of these graphs

were evaluated to determine (within the path) whether to spot or reject the word. Additionally, a pruning model that includes a change in internal structure of classifier was introduced and compared with default internal structure of the classifier. This proposed system had attempted to build a language independent word spotting system.

In [6], four pre-processing steps were applied on digital image of word. Historical Arabic handwritten manuscripts (HAH manuscripts) was first obtained from hardcopy of HAH manuscript, by using a 300 x 300 resolution scanner and transformed the text into a digitized image. Then, these digital images were processed through four pre-processing steps. First step of pre-processing phase was binarization, which converts a gray-scale image into a binary image. Simply, binarization was to mark pixels, which belongs to foreground to ON (value as one) and background to OFF (value as zero), by utilizing a threshold value. In second step of pre-processing, a 3-By-3 median filter was used to remove noise for HAH manuscripts. Noises were introduced during scanning and appeared as isolated small regions or as irregular characters edges. In third step of pre-processing, smoothing was used to remove contour discontinuities, which were found after applying 3-By-3 median filter in previous pre-processing step. In final step of pre-processing, thinning was used to reduce binary image regions into skeletons of region. These four pre-processing steps were important as they had to compensate for poor quality of original manuscript and/or poor scanning of original manuscript.

In [17], a word spotting model was proposed, that was motivated by some human visual characteristics. This word spotting approach was based on human perception, visualization and image understanding. The proposed bio-inspired model was worked in two different levels. First, to define several candidate zones, a Global Filtering module was enabled. In this module, the whole document was first scanned for a global view, which was used to search the element (e.g., query) that was looked for. Then, the selection of good retrieved results was done by using a filtering module facilitates. In this module, it was attempted to find out the exact query within processed documents in a lower scale. This allowed reduction of the number of false positives. These modules were based on an accumulation of voting processes which resulted from the generalized application. Thus, they proposed a multi-scale word spotting approach, which spreads from rough to fine scale. Segmentation of documents was not needed for this proposed model. The proposed model was tested with George Washington Database and results were measure against state-of-the-art performances.

In [9], a word spotting system was proposed, which was based on Hidden Markov Models (HMM) characteristic. A key property of HMM for modeling handwriting is that, they are able to deal with the problem, where characters are connected in cursively handwritten text. Several processing steps were included within this proposed word spotting system. In preprocessing stage, input text line images were normalized in order to deal with different writing styles. In this step, individual text lines were extracted after binarized the handwritten document images. This text line extraction and binarization; both were depending on the

documents type and quality. Afterwards, a series of local feature vectors were extracted using a sliding window of one pixel width moving from left to right over the image. In the training phase, each HMMs character was trained for each alphabet that was based on transcribed text line images. At the recognition stage, the trained HMMs character was connected to keyword text line model to calculate the probability score of the input text line. This probability score was finally normalized against a general filler text line model before it was compared to a threshold. This system was forced to contain the exact sequence of key-word characters at the beginning, in the middle or at the end of the text line, separated by the space character. The rest of the text line should be an arbitrary sequence of characters that was modeled with the filler text line.

In [19], a method on learning-free QbE keyword spotting for handwritten documents was proposed. This method was having three fundamental steps, namely preprocessing, feature extraction and matching. These three steps were to address the critical variations of text images; tilt, translation, different writing styles etc. Contrast normalization was part of preprocessing step, which was aiming to overcome the appearance descriptor shortcomings. Storage requirements for document collection were significantly reduced with the help of this preprocessing step. A series of descriptors were generated using a novel appearance descriptor, during the feature extraction step. This novel appearance descriptor referred as modified Projections of Oriented Gradients (POG). Distances in-between query and word sequences were efficiently computed by using the proposed *Selective Matching* algorithm. This algorithm was further extended to manage an augmented set of images, which were originating from single query image. The proposed method efficiencies were established by experimentation conducted on seven publicly available datasets. In these experiments, the proposed method considerably outperformed against state-of-the-art learning-free techniques.

In [20], an efficient segmentation-free word spotting method was proposed. This method was applicable in the context of the historical document collections, by following the query-by-example pattern. A patch-based framework was used for this method; where local patches were described by bag-of-visual-words model powered by SIFT descriptors. To perform word retrieval, the QbE pattern was used as stated earlier, where users input the system a sample image of the sought word. This proposed method was sampling densely the SIFT descriptors at first based on the user inputted query word. Then, it was supposed to quantize SIFT descriptors into visual words using the codebook. Afterwards, by accumulating visual words into different bins of spatial pyramid histograms, the patch descriptors were obtained. These sets of putative local patch descriptors, which were visually similar to the given query, were first obtained by this segmentation free approach. Then, a voting scheme was used which aim to find the location within the document page images with a high chances to find the search word. It was highly possible to efficiently index document information both in terms of memory and time by using these patch descriptors with latent semantic analysis technique to a topic space and then compressed the patch descriptors with product quantization method. The proposed method was evaluated using four different collections of historical documents (like,

GW1500 dataset, BCN dataset) on both handwritten and typewritten scenarios. In evaluation, the proposed method considerably outperformed against state-of-the-art keyword spotting approaches.

In [16], one of the most commonly used feature comparison algorithm in handwriting words recognition is the DTW. DTW is an algorithm for measuring similarity between two strings of array sequence, which may vary in length or time or speed. DTW has been commonly used in the field of speech processing, bio-informatics and on-line handwriting communities to match 1-D and/or 3-D signals for quite some time now. By using DTW algorithm method, it is possible to convert an image features in 1-dimension, even though, the corresponding image features are in 2-dimensions in general; but the possibility is there to slack the association between column features of images. DTW algorithm method is used to minimize the variations in between the image features vectors. In general, DTW is a method, which allows a computer to find an optimal match between two given strings of array sequence.

In [21], the image of words was not segmented into smaller parts, rather considered as a whole shape. Thus, the word matching recognition was used as shape matching algorithm. A comparative study in between a number of interest detectors points is presented to show that corner can be detected with the Harris detector [8], but as always, such detector is having drawback of its sensitiveness and responsiveness to noise.

Word-spotting is an attractive alternative to the apparently obvious recognize-then-retrieve approach to historical manuscript recovery. Word-spotting is having the capability of identifying indexing terms automatically, making it possible to overcome costly human labor effort.

There are some gaps in literature survey; the very important one is the relationship between the feature level extraction, the nature of the data and the resolution. Finding a correlation between these may sidestep confusions of choice between DTW, Fréchet distance, Hausedorff distance, Smith-Waterman, LCS methods (or to select directly the appropriate method). The results of the existing approaches can be further amplified by refining both the adaptation of the filter selection to the query word image and to the document characteristics and in the binarisation step [15].

Chapter 3

3. Methods and Methodologies

In this chapter, the theoretical foundation of present work is explained. The concept of Histogram of Oriented Gradients (HOG), Dynamic Time Wrapping (DTW), Hausdorff distance, Fréchet distance, Smith-Waterman, Largest Common Substring (LCS) are described here.

3.1 Histogram of Oriented Gradients

HOG is an efficient feature descriptor proposed by Dalal and Triggs [17] and it is extensively used for the purposes like target tracking, automatic target detection and recognition. HOG descriptor extract features, which are used broadly in image processing to detect image object. It uses a technique to counts gradient orientation occurrences in localized portions of an image. This image extraction method is similar to Edge Orientation Histograms (EOG), Scale-Invariant Feature Transform (SIFT) descriptors.

The essential thought behind HOG descriptors, are that local object appearance and shape within an image can be described by the distribution of intensity gradients or edge directions. The usage of HOG descriptors can be achieved by dividing an image into small connected regions, called cells and for each cell compiling a histogram of either gradient directions or edge orientations for the pixels within the cell. These histograms combinations then represent descriptor. Toward improving accuracy, these local histograms can be contrast-normalized by calculating a measure of the intensity across a larger region of the image, called a block, and then using this value to normalize all cells within the block. This normalization results in better invariance to changes in illumination or shadowing.

The HOG descriptor maintains couple of key advantages over other descriptor methods. This method upholds invariance to geometric and photometric transformations as HOG descriptor operates on localized cells. This is an exception for object orientation. This invariance to geometric and photometric transformation changes will only appear in larger spatial regions.

HOG feature extraction consists of many histograms of orientated gradients in localized areas of an image. HOG image feature extraction technique is divided into the following three steps.

Step I – Gradient computation:

In HOG image feature extraction, the magnitude G and direction $\theta$of the calculated gradients are computed by below equations correspondingly:

$$G = \sqrt{{G_x}^2 + {G_y}^2}$$

$$\theta = \arctan \frac{G_x}{G_y}$$

*Step II – Histogram gener*ation:

The image is divided into 8×8 cells and a histogram of gradients is calculated for each 8×8 cells. An 8×8 image patch contains 8x8x3 = 192 pixel values. The gradient of this patch contains 2 values (magnitude and direction) per pixel which adds up to 8x8x2 = 128 numbers, these 128 numbers are represented using a 9-bin histogram which can be stored as an array of 9 numbers. Not only is the representation more compact, calculating a histogram over a patch makes this representation more robust to noise. Individual gradients may have noise, but a histogram over 8 × 8 patches makes the representation much less sensitive to noise. 8×8 cells in a photo of a pedestrian scaled to 64×128 are big enough to capture interesting features. The histogram is essentially a vector (or an array) of 9 bins (numbers) corresponding to angles 0, 20, 40, 60 … 160. A bin is selected based on the direction, and the vote (the value that goes into the bin) is selected based on the magnitude. If the angle is greater than 160 degrees, it is between 160 and 180, and we know the angle wraps around making 0 and 180 equivalent. The contributions of all the pixels in the 8×8 cells are added up to create the 9-bin histogram. The histogram has a lot of weight near 0 and 180 degrees, which is just another way of saying that in the patch gradients are pointing either up or down [figure 3.1 (c)].

Step III - Histogram normalization:

Ultimately, large histogram is formed by combining altogether the generated histograms belong to a block, which consists of cells.

In order to reduce variance influence in enlightenment and contrast, L1 normalization is adopted for this paper. The normalization follows following equation, after having the large histogram:

$$v = \frac{V_k}{\|V_k\| + \varepsilon}$$

Where:

- ε is a constant variable

- V_k is the combined histogram vector for a block
- v is normalized vector, the final HOG extracts one dimensional array

The first stage is optional and it applies Image re-sizing. In this step, an inputted image can be re-sized to get a desire size for better image extraction. Additionally, inputted image are converted into gray-scale.

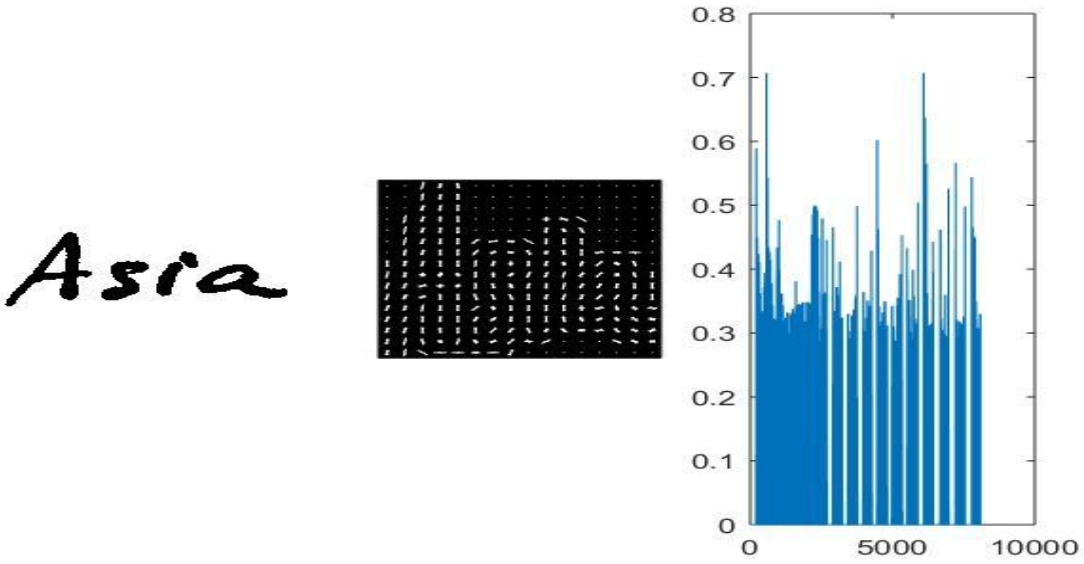
The second stage computes order image gradients. These capture shape, outline and some surface information, while providing further resistance to enlightenment variations. The locally dominant color channel is used, which provides color invariance to a great extent. Variant method might compute second order image derivatives, which acts as primitive bar detectors – a useful feature for capturing bar like structures.

The third stage produces an encoding, which is sensitive to local image content, while left behind resistant to small changes in pretense or appearance. The entire image window is divided into small square regions, called "cells". For each cell, it is accumulated a local one dimension histogram of either gradient orientations or edge orientations, over all pixels in the said cell. This combined cell-level one dimension histogram forms the basic "orientation histogram" representation. Each orientation histogram divides the gradient angle range into a fixed number of predetermined bins. The gradient magnitudes of all pixels in the cell are used to take part into the orientation histogram.

The fourth stage computes normalization across blocks, which takes local groups of cells and direction normalizes their overall responses before moving to next stage. Normalization introduces better invariance to enlightenment, shadowing and edge distinction. It is performed by accumulating a measure of local histogram "energy" over local groups of cells, which called "blocks". The result is used to normalize each cell in the block. Typically each individual cell is shared between several blocks, but its normalization is block dependent and thus different. Thus the cell appears several times in the final output vector with different normalization. This might seem redundant, but it improves the performance.

The final stage computes an optional global image normalization, which is designed to reduce the influence of enlightenment effects. In practice, gamma (power law) compression is used either computing the square root or the log of each color channel. Image texture strength is typically proportional to the local surface enlightenment, so, this compression helps to reduce the effects of local shadowing and enlightenment variations.

Following example contains ‘input image’, ‘visualization of HOG’ and ‘plotting of histogram’:



(a)Sample Image (b)Visualization of sample image (c)Plotting of HOG

Figure 3.1: The HOG feature extraction from the sample word image Asia.

3.2 Dynamic Time Warping

Dynamic Time Warping (DTW) is an algorithm for measuring similarity between two strings of array sequence [16], which may vary in length. For instance, walking similarities could be detected using DTW, even if, there were acceleration and deceleration during observation period or if one person is walking faster than the other. DTW can be applied to chronological sequences of video, audio and graphics data; any data which can be converted into a linear sequence of array.

DTW is a method, which calculates an optimal match between two given strings of array sequences. Let's assume:

String 1 = {5, 4, 3, 1, 1, 1, 9}
String 2= {5, 5, 4, 4, 3, 1}

DTW is used to find out the optimal match between these two above strings of array sequences.

In DTW, to start with, the distance between two points, d(k, l) are defined, where k and l represent the two points. Let,

d(k, l) = |k - l| //absolute difference

In next step, a 2D matrix is formed by using these two given strings of array sequence. The distances between each point of one given string of array sequence with every points of other given string of array sequence will be calculated and the same will be used vice-versa as well to find out the optimal match between these two above strings of array sequences.

In matrix calculation, for the first row, if it is taken no values from String 1, the distance between this and String 2 will be infinity. So, let's put infinity on the first row. Same goes for the first column. If it is taken no values from String 2, the distance between this one and String 2 will also be infinity. And the distance between 0 and 0 will simply be 0. So, the outcome is,

Now onwards, for each step, it will be considered the distance between each point in concern and add it with the minimum distance found so far. This will give the optimal distance of two sequences up to that position.

The value at Table represents the maximum distance between these two given strings of array sequence. Here the maximum distance between String 1 and String 2 is 8.

Now if it is backtracked from last point, all the way back towards starting point, it will give a long line that moves horizontally, vertically and diagonally [22].

It will continue like this, till it reaches starting point (0, 0). Each move has its own meaning:

- A horizontal move represents deletion. That means String 2 sequence gets accelerated during this interval.
- A vertical move represents insertion. That means String 2 sequence gets decelerated during this interval.
- A diagonal move represents match. During this period String 1 and String 2 are same.

Table 3.2.1: Dynamic Time Warping calculation (a-d)

	0	5	5	4	4	3	1
0							
5							
4							
3							
1							
1							
1							
9							

(a)

	0	5	5	4	4	3	1
0	0	inf	inf	inf	inf	inf	inf
5	inf						
4	inf						
3	inf						
1	inf						
1	inf						
1	inf						
9	inf						

(b)

	0	5	5	4	4	3	1
0	0	inf	inf	inf	inf	inf	inf
5	inf	0	0	1	2	4	8
4	inf	1	1	0	0	1	4
3	inf	3	3	1	1	0	2
1	inf	7	7	4	4	2	0
1	inf	11	11	7	7	4	0
1	inf	15	15	10	10	6	0
9	inf	19	19	15	15	12	8

(c)

	0	5	5	4	4	3	1
0	0	inf	inf	Inf	inf	inf	inf
5	inf	0	0	1	2	4	8
4	inf	1	1	0	0	1	4
3	inf	3	3	1	1	0	2
1	inf	7	7	4	4	2	0
1	inf	11	11	7	7	4	0
1	inf	15	15	10	10	6	0
9	inf	19	19	15	15	12	8

(d)

The formula will be,

Table[i][j] := d(i, j) + min(Table[i-1][j], Table[i-1][j-1], Table[i][j-1])

For the first one, d(1, 1) = 0, Table[0][0] represents the minimum. So, the value of Table[1][1] will be 0 + 0 = 0. For the second one, d(1, 2) = 0. Table[1][1] represents the minimum. The value will be: Table[1][2] = 0 + 0 = 0.

Pseudo-code will be:

```
Procedure DTW (Sample, Test):
n := Sample.length
m := Test.length
Create Table[n + 1][m + 1]
for i from 1 to n
   Table[i][0] := infinity
```

```
end for
for i from 1 to m
   Table[0][i] := infinity
end for
Table[0][0] := 0
for i from 1 to n
   for j from 1 to m
      Table[i][j] := d(Sample[i], Test[j])
                  + minimum(Table[i-1][j-1],      //match
                            Table[i][j-1],        //insertion
                            Table[i-1][j])        //deletion
   end for
end for
Return Table[n + 1][m + 1]
```

The complexity of computing DTW is O(m * n) where m and n represent the length of each sequence.

3.3 Longest Common Subsequence

The longest common subsequence problem is finding the longest matching sequence exists in between two given strings of array sequences, which may vary in length.

Subsequence

Let's consider a sequence S = $<s_1, s_2, s_3, s_4, \ldots, s_n>$.

Another sequence Z = $<z_1, z_2, z_3, z_4, \ldots, z_m>$ over S, is called a subsequence of S, if and only if it can be derived from S by deletion of some elements.

Common Subsequence

Let's consider, X and Y are two strings of array sequence over a finite set of elements. It can say that Z is a common subsequence of X and Y, if Z is a subsequence of both X and Y.

Naïve Method

Let's X be a sequence of length *m* and Y a sequence of length *n*. Check for every subsequence of X whether it is a subsequence of Y and return the longest common subsequence found.

There are 2^m subsequences of X. Testing the sequence whether or not it is a subsequence of Y takes $O(n)$ time. Thus, the naïve algorithm would take $O(n^{2m})$ time.

Longest Common Subsequence

If a set of two strings of array sequence (which may vary in length) is given, the longest common subsequence problem is to find a common subsequence among all the sub-sequences that is of maximal length.

The longest common subsequence problem is a classic computer science problem.

Let's assume:

String 1 = {5, 4, 3, 1, 1, 1, 9}
String 2= {5, 5, 4, 4, 3, 1}

In longest common subsequence, to start, the distance between two points, d(k, l) is defined, where k and l represent the two points. The value of d(k, l) is always '1' in longest common subsequence method.

In next step, a 2D matrix is formed by using these two given strings of array sequence. The distances between each point of one given string of array sequence with every points of other given string of array sequence will be calculated and the same will be used vice-versa as well to find out the optimal match between these two above strings of array sequences [figure 3.3.1 (a)].

Here, Table 3.3.1.b represents the optimal distance between two strings of array sequences; if it is considered the sequence up to String 1[i] and String 2[j], considering all the optimal distances observed.

For the first row, if it is taken no values from String 1, the distance between this and String 2 will be zero. So, let's put all zero on the first row. Same goes for the first column. If it is taken no values from String 2, the distance between this one and String 2 will also be zero. And the distance between 0 and 0 will simply be 0. So, the outcome is,

Table 3.3.1: Longest Common Subsequence calculation (a)-(b)

	0	5	5	4	4	3	1
0							
5							
4							
3							
1							
1							
1							
9							

(a)

	0	5	5	4	4	3	1
0	0	0	0	0	0	0	0
5	0						
4	0						
3	0						
1	0						
1	0						
1	0						
9	0						

(b)

Now onwards, for each step, it will be considered the distance between each point in concern and add it with the distance as '1' found so far. This will give the difference of two sequences up to that position.

The formula will be,

Table[i][j] := d(i, j) + Table[i-1][j-1]>> If String1[i] = String2[j]

:=max(Table[i-1][j], Table[i][j-1])>> If String1[i] <> String2[j]

The maximum value found in Table as '4' in position [4][6]. Now if it is backtracked from this point, all the way back towards ZERO value, it will give a long line that moves horizontally, vertically and diagonally.

Algorithm: LCS-Length-Table-Formulation (X, Y)

```
m := length(X)
n := length(Y)
for i = 1 to m do
  C[i, 0] := 0
for j = 1 to n do
  C[0, j] := 0
for i = 1 to m do
  for j = 1 to n do
    if xi = yj
      C[i, j] := C[i - 1, j - 1] + 1
      B[i, j] := 'D'
    else
      if C[i -1, j] ≥ C[i, j -1]
        C[i, j] := C[i - 1, j] + 1
        B[i, j] := 'U'
      else
```

```
        C[i, j] := C[i, j - 1]
        B[i, j] := 'L'
return C and B

Algorithm: Print-LCS (B, X, i, j)
if i = 0 and j = 0
  return
if B[i, j] = 'D'
  Print-LCS(B, X, i-1, j-1)
  Print(xi)
else if B[i, j] = 'U'
  Print-LCS(B, X, i-1, j)
else
  Print-LCS(B, X, i, j-1)
```

This algorithm will print the longest common subsequence of X and Y [23]. To populate the table, the outer for loop iterates m times and the inner for loop iterates n times. Hence, the complexity of the algorithm is O(m, n), where m and n are the length of two strings.

Table 3.3.2: Longest Common Subsequence calculation (c)-(d)

		0	5	5	4	4	3	1
0	0	0	0	0	0	0	0	0
5	0	1	1	1	1	1	1	1
4	0	1	1	2	2	2	2	2
3	0	1	1	2	2	3	3	3
1	0	1	1	2	2	3	4	4
1	0	1	1	2	2	3	4	4
1	0	1	1	2	2	3	4	4
9	0	1	1	2	2	3	4	4

(c)

		0	5	5	4	4	3	1
0	0	0	0	0	0	0	0	0
5	0	1	1	1	1	1	1	1
4	0	1	1	2	2	2	2	2
3	0	1	1	2	2	3	3	3
1	0	1	1	2	2	3	4	4
1	0	1	1	2	2	3	4	4
1	0	1	1	2	2	3	4	4
9	0	1	1	2	2	3	4	4

(d)

It will continue like this, till it reaches starting point (0, 0). Each move has its own meaning:

- A horizontal move represents deletion. That means String 2 sequence accelerated during this interval.
- A vertical move represents insertion. That means String 2 sequence decelerated during this interval.
- A diagonal move represents match. During this period String 1 and String 2 are same.

3.4 Smith–Waterman Algorithm

Smith-Waterman uses dynamic programming to obtain the optimal alignment between two strings of array sequence, which may vary in length.

The algorithm computes a cost matrix H of which each element $H_{i,j}$ depends on $H_{i-1,j-1}$, $H_{i-1,j}$ and $H_{i,j-1}$. At the beginning of the algorithm, no parallelism is possible. But as the algorithm progresses, a growing "wave front" of independent values becomes calculable. The cost matrix can be divided into sub-matrices, which exhibit the same data dependency pattern.

Let's X = $\{x_1x_2x_3...x_n\}$ and Y = $\{y_1y_2y_3...y_m\}$ are two strings of array sequence to be aligned, where *n* and *m*are the lengths of X and Y respectively.

Step I. First step is to determine both substitution matrix and gap penalty scheme.

- s(a,b) - Similarity score of the two strings of array sequence elements
- W_k- The penalty of a gap that has length *k*

Step II. Build scoring matrix H; initialize its first row and first column as all zeroes. The size of the scoring matrix is $[(n + 1)*(m + 1)]$. The used indexing is zero based:

$$H_{k0} = H_{0l} = 0; \text{ for } 0 \le k \le n \text{ and } 0 \le l \le m$$

Step III. Populate rest of the scoring matrix H by using below formula:

$$H_{ij} = \max \begin{cases} H_{i-1,j-1} + s(a_i, b_j), \\ \max_{k \ge 1}\{H_{i-k,j} - W_k\}, \\ \max_{l \ge 1}\{H_{i,j-l} - W_l\}, \\ 0 \end{cases} \qquad (1 \le i \le n, 1 \le j \le m)$$

Where:

- $H_{i-1,j-1} + s(a_i,b_j)$ is the score of a_i and b_j aligning
- $H_{i-k,j} - W_k$ is the score, if a_i is at the end of a gap of length k
- $H_{i,j-l} - W_l$ is the score, if b_j is at the end of a gap of length l
- 0 means there is no similarity up to a_i and b_j

Step IV. Final step is to trace back. Trace back is starting from the highest score in the scoring matrix H and ending at a matrix cell that has a score of 0; trace back is based on the source of each score recursively to generate the best possible local alignment.

Let's assume:

String1 = {5, 4, 3, 1, 1, 1, 9}
String2 = {5, 5, 4, 4, 3, 1}

Smith-Waterman is used to obtain the optimal alignment between two strings of array sequence.

In first step, a Two Dimensional Matrix Table is formed by using these two given strings of array sequence. The distances between each point of one given string of array sequence with every points of other given string of array sequence will be calculated and the same will be used vice-versa as well to find out the optimal match between these two above strings of array sequences. Here, the optimal distance between two strings of array sequences; if it is considered the sequence up to String 1[i] and String 2[j], considering all the optimal distances observed.

For the first row, if it is taken no values from String 1, the distance between this and String 2 will be zero. So, let's put all zero on the first row. Same goes for the first column. If it is taken no values from String 2, the distance between this one and String 2 will also be zero. And the distance between 0 and 0 will simply be 0. So, the outcome is,

Table 3.4.1: Smith–Waterman Algorithm calculation table (a-b)

	0	5	5	4	4	3	1
0							
5							
4							
3							
1							
1							
1							
9							

(a)

	0	5	5	4	4	3	1
0	0	0	0	0	0	0	0
5	0						
4	0						
3	0						
1	0						
1	0						
1	0						
9	0						

(b)

Now onwards, for each step, it will be considered the distance between each point in concern and add it with the distance found so far depending upon formula. This will give the difference of two sequences up to that position.

The formula will be,

Table[i][j := Table[i-1][j-1] + 3 >> If String1[i] = String2[j]

$$:= \max \begin{cases} = \text{Table}[i-1][j-1] - 3 \\ = \text{Table}[i-1][j] - 2 \\ = \text{Table}[i][j-1] - 2 \end{cases} \quad >> \text{If String1}[i] <> \text{String2}[j]$$

<u>Note:</u> No negative value in table; negative value will be replaced with ZERO.

Table 3.4.2: Smith–Waterman Algorithm calculation table (c-d)

	0	5	5	4	4	3	1
0	0	0	0	0	0	0	0
5	0	3	3	1	0	0	0
4	0	1	1	6	4	2	0
3	0	0	0	4	3	7	5
1	0	0	0	2	1	5	10
1	0	0	0	0	0	3	8
1	0	0	0	0	0	1	6
9	0	0	0	0	0	0	4

(c)

	0	5	5	4	4	3	1
0	0	0	0	0	0	0	0
5	0	3	3	1	0	0	0
4	0	1	1	6	4	2	0
3	0	0	0	4	3	7	5
1	0	0	0	2	1	5	10
1	0	0	0	0	0	3	8
1	0	0	0	0	0	1	6
9	0	0	0	0	0	0	4

(d)

The maximum value found in Table as ‘10’ in position [4][6]. Now if it is backtracked from this point, all the way back towards ZERO value; it will give a long line that moves horizontally, vertically and diagonally.

The backtracking procedure will be:

```
if Table[i-1][j-1] >= Table[i-1][j] and Table[i-1][j-1] >= Table[i][j-1]
        i := i - 1
        j := j - 1
else if Table[i-1][j] > Table[i-1][j-1] and Table[i-1][j] >= Table[i][j-1]
        i := i - 1
else
        j := j - 1
end if
```

It will continue like this, till it reaches starting point (0, 0). Each move has its own meaning:

- A horizontal move represents deletion. That means String 2 sequence accelerated during this interval.
- A vertical move represents insertion. That means String 2 sequence decelerated during this interval.
- A diagonal move represents match. During this period String 1 and String 2 are same.

3.5 Hausdorff Distance

Named after Felix Hausdorff (1868-1942), Hausdorff distance is the maximum distance of a set to the nearest point in the other set. More formally, Hausdorff distance from set A to set B is a maxi-min function, defined as:

$$h(A,B) = \max_{a \in A} \left[\min_{b \in B} \{ d(a,b) \} \right]$$

Where,

- a and b are points of sets A and B respectively, and.
- d(a, b) is any metric between these points.

For simplicity, we will take d(a, b) as the Euclidian distance between a and b. If for instance A and B are two sets of points, a brute force algorithm would be:

1. h = 0
2. for every point a_i of A,
   - 2.1 shortest = Inf ;
   - 2.2 for every point b_j of B
     - $d_{ij} = d(a_i , b_j)$
     - if d_{ij} < shortest then
       - shortest = d_{ij}
   - 2.3 if shortest > h then
     - h = shortest

This algorithm is illustrated in below figure.

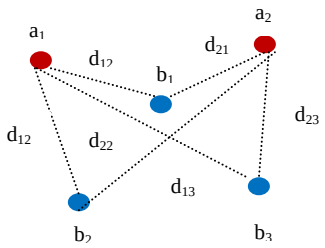


Figure 3.5 - Hausdorff Distance

It should be noted that Hausdorff distance is oriented, we could say asymmetric as well), which means that most of times h(A, B) is not equal to h(B, A). This general condition also holds for the example of above figure, as h(A, B) = d(a1, b1), while h(B, A) = d(b2, a1). This asymmetry is a property of maxi-min functions, while mini-min functions are symmetric.

A more general definition of Hausdorff distance would be:

$$H(A, B) = \max \{ h(A, B), h(B, A) \}$$

Which defines the Hausdorff distance between A and B, while above equation is applied to Hausdorff distance from A to B (also called directed Hausdorff distance). The two distances h(A, B) and h(B, A) are sometimes termed as forward and backward Hausdorff distances of A to B.

If sets A and B are made of lines or polygons instead of single points, then H(A, B) applies to all defining points of these lines or polygons, and not only to their vertices. The brute force algorithm could no longer be used for computing Hausdorff distance between such sets, as they involve an infinite number of points.

3.6 Fréchet Distance

The Fréchet distance measures distance between two strings of array sequence having same length. It is defined as the minimum cord-length sufficient to join a point traveling forward along with one string of array sequence and one traveling forward along with other string of array sequence, although the rate of travel for either point might not necessarily be uniform.

Fréchet distance can be explained [24] by a classic imagination about a man is traversing a finite path, while walking with his dog on a strap, the dog traversing in a separate path. Assuming that, the dog varies its speed to keep as looser as possible of its strap. The Fréchet distance between these two curves is the length of shortest strap sufficient for both to traverse in their separate paths. Note that, Fréchet distance definition is symmetric with respect to two curves—Fréchet distance would be the same, if the dog is walking with its owner.

Let's consider, S is a metric space. A curve 'A' in S is a continuous map from the unit interval into S, i.e., $A : [0,1] \rightarrow S$. A re-parameterization α of [0,1] is a continuous and non-decreasing surjection $\alpha : [0,1] \rightarrow [0,1]$.

Let's 'A' and 'B' are two given curves in S, where S is a metric space. The Fréchet distance between 'A' and 'B, is defined as the infimum (*infimum → the largest quantity that is less than or equal to each of a given set or subset of quantities*) over all re-parameterizations of α and β of [0,1] of the maximum over all $t \in [0,1]$ of the distance in S between $A(\alpha(t))$ and $B(\beta(t))$. In mathematical notation, the Fréchet distance F (A,B) is represented as:

$$F(A,B) = \inf_{\alpha,\beta} \max_{t\in[0,1]} \left\{ d\Big(A(\alpha(t)),\, B(\beta(t))\Big) \right\}$$

Where, d is the distance function of S.

We can think of parameter '*t*' as "time". Then, A(α(t)) is the position of the dog and B(β(t)) is the position of the dog's owner at a specific time '*t*' or vice versa. The length of the strap between them at that specific time '*t*' is the distance between A(α (t)) and B(β(t)). Taking the infimum over all possible re-parameterizations of [0,1] corresponds to choosing the walk along the given paths, where the maximum strap length is minimized. The restriction is that, α and β should be non-decreasing, means that, neither the dog nor its owner can backtrack.

The Fréchet metric takes into account the flow of the two curves because the pairs of points, whose distance contributes to the Fréchet distance sweep continuously along their respective curves. This makes the Fréchet distance a better measure of similarity for curves than alternatives, such as the Hausdorff distance, for arbitrary point sets. It is possible for two curves to have small Hausdorff distance but large Fréchet distance.

3.7 Proposed Model

Any word searching model, consists of two leading steps, namely (a) page segmentation (text lines and/or words are extracted from document images) and (b) word matching schema (i.e. confirming a target word in a document image as the searched word).

A two-stage approach, comprising of pre-selection of target words as search words and confirmation of pre-selected words as search word, has been presented for searching a word from document images. A feature vector of length 3 has been extracted from all the words in a document image to clean out irrelevant words with respect to the search word [15].Generally, in a document, there exists a varying number of characters in different words. That is why searching a word in a whole document is incorporate mismatched word images in the fetched word image and also increases the time consumption to complete the task. Keeping this idea in mind, the words having different number of character with respect to the search word are discarded at the beginning as preprocessing.

To confirm the outstanding words in the document page as probable search word, a voting based approach has been used. For doing this, a modified HOG feature descriptor is extracted from each word image, then 5 distance-matching metrics are calculated, fed to a voting schema with the help of threshold value of each metrics, calculated beforehand. Here 3 types of voting is performed, first 2 ,with the varying no of metrics vote for positivity of the search

word and in the last one three distance metrics are used among which if more than one votes for the positivity the model will indicate the word as a search word.

3.7.1 Pre-selection of Search Words for a given Search Word

Assume that, a binarized word image can be represented like $B_p = \{f(i, j) : 1 \leq i \leq h \wedge 1 \leq j \leq w\}$, where h and w are height and width of B_p, respectively, and $f(i, j) \in \{0, 1\}$ (as, 0 and 1 represent non-data pixels and data pixels, respectively).

At first, Bp is segmented into three non-overlapping regions horizontally (e.g., upper, middle, and lower zones as shown in Figure 3.7.1), for feature extraction. This figure is having four horizontal lines, viz. R1, R2, R3, and R4, to distinguish the regions horizontally. The lines R1 and R4 are calculated respectively:

$R1 = \min \{i : f(i, j) = 1 \wedge (i, j) \in [1, h] \times [1, w]\}$
$R4 = \max \{i : f(i, j) = 1 \wedge (i, j) \in [1, h] \times [1, w]\}$

The identification R2 and R3 is complicated. To identifying these lines, the number of transition points between data and non-data pixels and vice versa along with each row of Bp is calculated.

$Ti = |\{j : ((f(i, j) = 1 \wedge f(i, j + 1) = 0) \vee (f(i, j) = 0 \wedge f(i, j + 1) = 1)) \wedge j \in [1, w - 1]\}|$

The mean of all such transition point counts (μTP) of B is estimated by

$\mu TP = 1/N\Sigma\sum_{i=1}^{H} Ti$, where $N = |\{i : Ti \neq 0\}|$, $i \in [1, H]$.

Now, R2 and R3 are calculated as follows:

$R2 = \min_{i=1,2,...,H}\{i : Ti > \mu TP\}$
$R3 = \max_{i=1,2,...,H}\{i : Ti > \mu TP\}$

Finally, the F1(= (f 1, f 2, f 3)) is estimated as

$f1 = 1/M\Sigma\sum_{i=R2}^{R3} Ti,$
where $M = |\{i : Ti \neq 0\}|$, $\forall i \in [R2, R3]$.

$f2 = |\{C : \theta(C) = 1\}|$.

f3 = |{C : ∅(C) = 1}| .

Now, let θ(.) and∅(.) be functions that represent the belongingness of a CC in upper/lower zone, respectively, which are defined by

$$\theta(C) = \begin{cases} o, if \min\{i : f(i,j) \in C\} \leq (R1 + R2)/2 \text{ and } \max\{i : f(i,j) \in C\} = R2 - 1 \\ 1, otherwise \end{cases}$$

$$\emptyset(C) = \begin{cases} o, if \max\{i : f(i,j) \in C\} \leq (R3 + R4)/2 \text{ and } \min\{i : f(i,j) \in C\} = R3 + 1 \\ 1, otherwise \end{cases}$$

Figure 3.7.1: Partitioning of a Word Image into Zones.

3.7.2 Decision Rule Creation

To filter out words that are irrelevant with respect to a given search word, a decision rule has been created. For that, decision boundaries (lower and upper bounds) for each of extracted feature values are estimated. These decision boundaries are then set by considering the mean (μ_{fi}, where i = 1,2,3) and standard deviation (σ_{fi}, i = 1, 2, 3) of feature values extracted from manually chosen N word image samples for a given search word. Let, L_{fi} and U_{fi} be the lower and upper bounds of feature value fi (i = 1, 2, 3), respectively and can be defined by

$L_{fi} = \mu_{fi} - \sigma_{fi}$, where i = 1, 2, 3
$U_{fi} = \mu_{fi} + \sigma_{fi}$, where i = 1, 2, 3

Finally, a word is pre-classified as a probable candidate for the given search word by the decision rule.

3.7.3 Deciding a Pre-selected Candidate Word as a Search Word

A texture based feature descriptor; HOG is very useful in pattern based recognition. HOG computes the gradient of a cell in various directions and then the values are normalized for the pattern description of each block. The total orientation angle (0^0-180^0 or 0^0-360^0) can be

divided into different ranges (like 8, 9, etc.), each of which is known as bin. Here we used 9 bin over the total feature length extracted from HOG, in case of Hausdorff distance, Fréchet distance, DTW distance measures, as these distance metrics always check foremost corner to the last corner of all extracted feature values which may decries the required performance. For rest two matching metric (Waterman, LCS) total feature length is compared.

From all the search words, 15 samples are selected manually as the candidate word. For a candidate word, all pre-selected words are gone through the present module. At the beginning five distance-matching metrics for each of these words (d_1, d_2, d_3, d_4, d_5) are calculated. Then, the obtained five scores are compared with the threshold range (Table 4.2.0), if the scores are in the corresponding threshold range, then that metric will increase the value of matching flag variable by 1. According to these, if M distance-matching metric votes for a search word as a target word then this model will store or load the word image in the designated folder. When M varies the performance of the model will change accordingly (Table 4.2.6-4.2.8).

It is observed that, among the five metrics, distance metrics are providing better performance while compare with the matching metrics. The working principle of these matching metrics is based on the longest substring retrieval from the extracted features of the word images. At the time of matching score generation these metrics are producing huge matching scores, which belong to the threshold range in many cases though they are wrongly retrieved.

For example, in case of fetching a key word image from the document word images, many images are wrongly retrieved as the target word from the present voting system some example of these situations are shown in Table 4.5.2. The matching metrics votes as the target word though they are not, which increases the false positive count as well as decreasing the performance of the model. To increase the performance of the present model, a voting system is designed with only 3 distance metrics, which is retrieving less no of wrong words i.e. the no of false positive is decrease, increasing the performance of the model.

Let's say, when it is searched for word image "ASIA", it picked the following two images along with others, where one image (Figure 3.7.3.a) is correct, though the handwritten version of this word is not good. But, it also picked the other image (Figure 3.7.3.b) as word 'ASIA'; despite the fact that, this word image is actually stands for 'Also'.

(a) (b)

Figure 3.7.3: Fetched word images for the search word ASIA

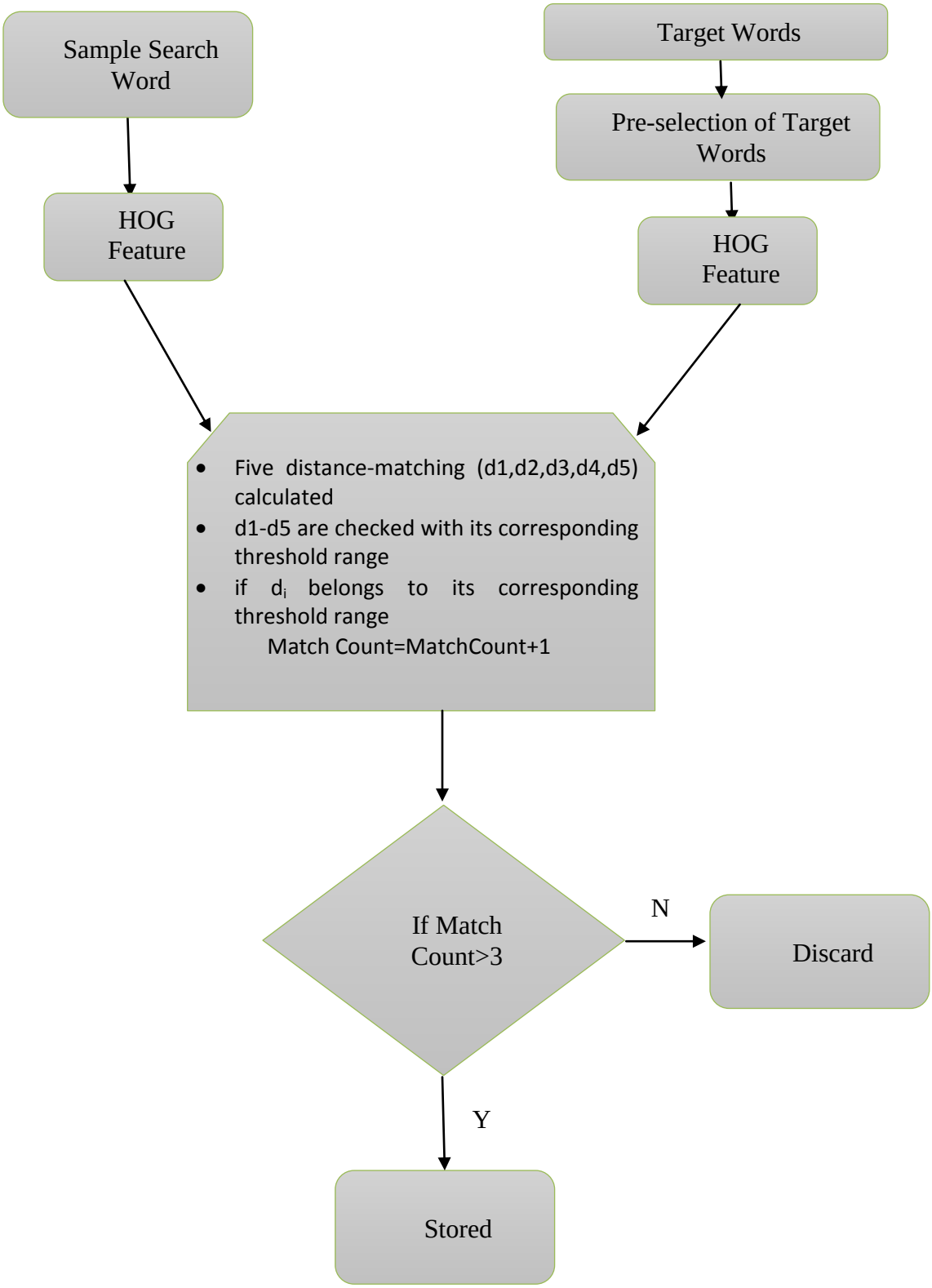


Figure 3.7: Block Diagram of the Present Word Spotting Model

Chapter 4

4. Result and Discussion

4.1 Database Description

Handwritten word spotting is a classification task to recognize pattern which is mainly used to detect specific keyword(s) within handwritten document images. In this current work, the QUWI database [25] is preferred as it is available to a certain extent for public use in different forums. Such preference for QUWI database is significant as a huge number of diversified (e.g., age, sex, nationality, background etc.) of writers. A good number of writing materials variations are available in terms of colors and thicknesses of pen/pencil.

Around 300 writers' handwritten documents in English and Arabic, are uploaded for public use in ICDAR 2015 competition on multi-script writer identification and gender classification using QUWI database. Mostly same text with117 words are kept in for each scripts. A page can have minimum of 99 words, as observed. Following constrains have been added with these document pages to increase searching complexity:

- Wrongly spelled words
- Use of different abbreviated forms
- Different spellings of same word

The document is alienated into 1:5 ratio. The first set of document pages are for the evaluation of present word searching algorithm, while the rest of document pages are useful to confirm values require to design the searching algorithm different parameters.

In this current work, a word searching model for handwritten document images is presented. A two-stage approach for word searching that discussed earlier is introduced for this purpose. In this sub-section experimental outcomes are described in details.

It has been mentioned at the start of this section that, the QUWI database is used to carry out the experiment. A set of 15 words are selected from document page images as query words, as shown in Table 4.1, for the evaluation of designed word searching technique. The choices of selecting word(s) to be searched are mainly depended on multiple occurrences of the searching word(s) within the respective document page.

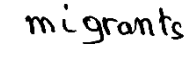
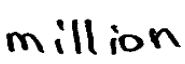
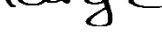
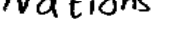
Additionally, following word pairs that are almost impossible to differentiate are also included here in searching word set:

- In terms of shape (e.g., "today" and "today's")
- Stemming words (e.g., "Asia" and "Asian" or "migrant" and "migrants")
- Derived forms (e.g., "large" and "largest")
- Arbitrary words (e.g., "international" and "nations")

Distinguishing same words, which are starting with upper case or lower case increases searching complexity within handwritten document images.

To identify the search word within pre-selected candidate words is extracted from manually chosen set of fifteen words among all in our experimental setup. The word instances are:

Table 4.1: Search words instances.

SW	Image Instance	SW	Image Instance	SW	Image Instance
SW01		SW06		SW11	
SW02		SW07		SW12	
SW03		SW08		SW13	
SW04		SW09		SW14	
SW05		SW10		SW15	

SW stands for "Index number of the particular Search Word image instances".

4.2 Evaluation Metrics

Results of present work are measured in terms of accuracy in searching process. The evaluation of the present searching model has been done using recall, precision and F-measure scores [12].

- Recall is the ratio of the relevant document that are retrieved successfully, can be depicted as–

$$Recall = \frac{\{Relevantwordimage\ \} \cap \{Retrivedwordimage\ \}}{\{Relevantwordimage\ \}}$$

- Precision is the ratio of retrieved document with respect to relevant query, can be depicted as–

$$Precision = \frac{\{Relevantwor\ \ dimage\ \} \cap \{Retrivedwordimage\ \ \}}{\{Retrivedwordimage\ \ \}}$$

- F-measure or balanced F-score is harmonic mean of recall and precision, can be depicted as–

$$F - Measure = \frac{2 \times recall \times precision}{recall\ + precision}$$

4.3 Experimental Outcomes and Analysis

In this section, the experimental results are carried out using this model. Among two stages, the first stage, pre-processing is carried out in [3].

In the present work, 15 word images are selected as search word of this model, and then threshold range is calculated for DTW, Fréchet, Hausdorff, Waterman and LCS methods, in Table 4.3.1.

Table 4.3.1: Search Word-based Threshold Range of Word Searching Model using DTW, Fréchet, Hausdorff, Waterman, LCS.

SW	DTW	Fréchet	Hausdorff	Waterman	LCS
America	0-0.2128	0-0.1097	0-0.0518	7.8704-100	47.2222-100
Asia	0-0.2730	0-0.1359	0-0.0399	6.2500-100	40.9722-100
Asian	0-0.2797	0-0.1396	0-0.0438	6.9444-100	43.0556-100
Europe	0-0.1621	0-0.0834	0-0.0576	9.7222-100	41.6667-100
immigrants	0-0.2577	0-0.1300	0-0.0576	9.9537-100	49.3056-100
immigration	0-0.2314	0-0.1190	0-0.0498	13.889-100	46.5278-100
International	0-0.1541	0-0.0919	0-0.0444	13.789-100	59.0278-100
large	0-0.2426	0-0.1177	0-0.0288	6.4815-100	39.5833-100
largest	0-0.2744	0-0.1357	0-0.0548	6.4815-100	33.3333-100
migrant	0-0.2916	0-0.1776	0-0.0647	9.0278-100	45.8333-100
migrants	0-0.2783	0-0.1703	0-0.0589	8.9935-100	46.3261-100
million	0-0.1827	0-0.0984	0-0.0327	9.0278-100	52.7778-100
Nations	0-0.1658	0-0.0897	0-0.0453	11.111-100	53.4722-100
today	0-0.3204	0-0.1488	0-0.0428	6.7130-100	40.2778-100
Today's	0-0.2344	0-0.1232	0-0.0444	6.2500-100	38.1944-100

"SW" stands for search word index.

The testing outcomes of this work are reported separately into eight tables, among those first table contains the values of Recall, Precision and F-Measure of the 15 search words passed through DTW.

Table 4.3.2: Search Word-wise Recall, Precision, F-Measure Values of present Word Searching Model using DTW.

SW	Recall	Precision	F-Measure
SW01	0.7100	0.1224	0.2088
SW02	0.3069	0.0920	0.1416
SW03	0.6458	0.0286	0.0548
SW04	0.1020	0.7692	0.1802
SW05	0.4950	0.1707	0.2539
SW06	0.5200	0.2321	0.3210
SW07	0.4600	0.7667	0.5750
SW08	0.3878	0.07567	0.1267
SW09	0.7500	0.0540	0.1007
SW10	0.7347	0.0350	0.0667
SW11	0.5657	0.1000	0.1699
SW12	0.3036	0.4573	0.3650
SW13	0.4222	0.0526	0.0936
SW14	0.7083	0.0865	0.1542
SW15	0.5400	0.0415	0.0770
Average	**0.5101**	**0.2056**	**0.1926**

In Table 4.3.3, the values of Recall, Precision and F-Measure of the 15 search words passed through Hausdorff Distance technique are reflected.

Table 4.3.3: Search Word-wise Recall, Precision, F-Measure Values of present Word Searching Model using Hausdorff.

SW	Recall	Precision	F-Measure
SW01	0.9400	0.1414	0.2458
SW02	0.7327	0.0846	0.1516
SW03	0.5625	0.0254	0.0486
SW04	0.6735	0.0565	0.1042
SW05	0.5455	0.1709	0.2602
SW06	0.5200	0.1831	0.2708
SW07	0.5800	0.6304	0.6042
SW08	0.3673	0.0387	0.0700
SW09	0.8333	0.0330	0.0635
SW10	0.8571	0.0398	0.0762
SW11	0.5960	0.1021	0.1743
SW12	0.4291	0.3072	0.3582
SW13	0.4667	0.0332	0.0620
SW14	0.8125	0.0400	0.0762
SW15	0.6400	0.0287	0.0549
Average	**0.6371**	**0.1277**	**0.1747**

In table 4.3.4, the values of Recall, Precision and F-Measure of the fifteen search word passed through Fréchet Distance technique is reflected.

Table 4.3.4: Search Word-wise Recall, Precision, F-Measure Values of present Word Searching Model using Fréchet.

SW	Recall	Precision	F-Measure
SW01	0.6500	0.1451	0.2372
SW02	0.3069	0.1076	0.1594
SW03	0.6250	0.0309	0.0588
SW04	0.1224	0.8000	0.2124
SW05	0.4848	0.1868	0.2697
SW06	0.4700	0.2186	0.2984
SW07	0.4800	0.7742	0.5926
SW08	0.3265	0.1391	0.1951
SW09	0.6458	0.0555	0.1021
SW10	0.6531	0.0304	0.0581
SW11	0.7071	0.1094	0.1894
SW12	0.3036	0.4601	0.3658
SW13	0.4000	0.0627	0.1084
SW14	0.6458	0.1308	0.2175
SW15	0.5200	0.0384	0.0715
Average	**0.4894**	**0.2193**	**0.2091**

In table 4.3.5, the values of Recall, Precision and F-Measure of the fifteen search word passed through LCS matching technique is reflected.

Table 4.3.5: Search Word-wise Recall, Precision, F-Measure Values of present Word Searching Model using LCS.

SW	Recall	Precision	F-Measure
SW01	0.7400	0.1080	0.1885
SW02	0.6238	0.0809	0.1432
SW03	0.5625	0.0288	0.0548
SW04	0.5918	0.0599	0.1088
SW05	0.4041	0.1270	0.1932
SW06	0.5200	0.1751	0.2620
SW07	0.1800	0.7500	0.2903
SW08	0.6531	0.0217	0.0421
SW09	0.7500	0.0264	0.0510
SW10	0.5306	0.0273	0.0519
SW11	0.5051	0.0806	0.1391
SW12	0.1822	0.3309	0.2350
SW13	0.4444	0.0359	0.0664
SW14	0.6250	0.0294	0.0561
SW15	0.5000	0.0162	0.0313
Average	**0.5208**	**0.1265**	**0.1276**

In table 4.3.6, the values of Recall, Precision and F-Measure of the 15 search words passed through Waterman matching technique are reflected.

Table 4.3.6: Search Word-wise Recall, Precision, F-Measure Values of present Word Searching Model using Waterman.

SW	Recall	Precision	F-Measure
SW01	0.7300	0.1059	0.1850
SW02	0.6040	0.0546	0.1001
SW03	0.6042	0.0243	0.0467
SW04	0.1939	0.1418	0.1638
SW05	0.5354	0.1715	0.2598
SW06	0.5100	0.2267	0.3138
SW07	0.6400	0.3299	0.4354
SW08	0.6939	0.0274	0.0528
SW09	0.6042	0.0247	0.0475
SW10	0.4082	0.0456	0.0820
SW11	0.7071	0.1165	0.200
SW12	0.4089	0.3146	0.3556
SW13	0.4222	0.0341	0.0631
SW14	0.5625	0.0265	0.0507
SW15	0.5800	0.0184	0.0356
Average	**0.5470**	**0.1108**	**0.1595**

In Table 4.3.7, voting technique is performed using the present model by applying DTW, Hausdorff, Fréchet, where we assume that if more than two processes vote for yes for a certain word then that will be considered as the target word.

Table 4.3.7: Search Word-wise Recall, Precision, F-Measure Values of present Word Searching Model using voting among five metrics (match count>2).

SW	Recall	Precision	F-Measure
SW01	0.7300	0.1148	0.1984
SW02	0.6040	0.1871	0.2857
SW03	0.6042	0.0322	0.0612
SW04	0.1939	0.7600	0.3089
SW05	0.5051	0.1629	0.2463
SW06	0.5100	0.2267	0.3138
SW07	0.5400	0.8438	0.6585
SW08	0.3673	0.0428	0.0766
SW09	0.7292	0.0311	0.0597
SW10	0.6531	0.0309	0.0589
SW11	0.5859	0.0853	0.1489
SW12	0.1903	0.2670	0.2222
SW13	0.4444	0.0424	0.0774
SW14	0.7500	0.0729	0.1328
SW15	0.5200	0.0232	0.0449
Average	**0.5285**	**0.1949**	**0.1929**

In Table 4.3.8, voting technique is performed using the present model by applying DTW, Hausdorff, Fréchet where we assume that if more than three processes vote for yes for a certain word then that will be considered as the target word.

Table 4.3.8: Search Word-based Recall, Precision, F-Measure Values of present Word Searching Model using voting among five metrics (match count>3).

SW	Recall	Precision	F-Measure
SW01	0.6800	0.1331	0.2226
SW02	0.3168	0.0982	0.1499
SW03	0.5625	0.0329	0.0622
SW04	0.1225	0.8000	0.2124
SW05	0.4646	0.1631	0.2415
SW06	0.4800	0.2233	0.3048
SW07	0.4200	0.8077	0.5526
SW08	0.2653	0.0684	0.1088
SW09	0.5208	0.0383	0.0714
SW10	0.4694	0.0241	0.0458
SW11	0.4949	0.0851	0.1452
SW12	0.1822	0.4369	0.2571
SW13	0.3778	0.0547	0.0956
SW14	0.7083	0.1411	0.2353
SW15	0.4800	0.0344	0.0642
Average	**0.4363**	**0.2094**	**0.1846**

In Table 4.3.9, voting technique is performed using the present model by applying DTW, Hausdorff, Fréchet where we assume that if more than one process vote for yes for a certain word then that will be considered as the target word.

Table 4.3.9: Search Word-wise Recall, Precision, F-Measure Values of present Word Searching Model using voting among DTW, Fréchet and Hausdorff (match count>1).

SW	Recall	Precision	F-Measure
SW01	0.7200	0.1309	0.2215
SW02	0.4158	0.1170	0.1826
SW03	0.6250	0.0287	0.0549
SW04	0.2143	0.0732	0.1091
SW05	0.5051	0.1742	0.2591
SW06	0.5800	0.2589	0.3581
SW07	0.5000	0.8065	0.6173
SW08	0.5102	0.1202	0.1946
SW09	0.7500	0.0534	0.0997
SW10	0.7347	0.0439	0.0829
SW11	0.6667	0.0629	0.1149
SW12	0.3239	0.4124	0.3628
SW13	0.6000	0.0520	0.0957
SW14	0.7708	0.1000	0.1770
SW15	0.5800	0.0408	0.0762
Average	**0.5664**	**0.1650**	**0.2004**

4.4 Comparison with State-of-the-art Methods

The outcomes of present method are compared with state-of-the-art methods [12, 26]. Among these two, the method presented in [12] is recognition based and another method [26] is recognition free searching method. It is observed that the authors of [26] compared several time series matching techniques. Average performances over mentioned state-of-the-art methods including the present method are reflected in table 4.4.

Table 4.4: Comparative study of the Present Method with other state-of-the-art Word Searching Methods.

Methods along with Publication year	Feature Extracted from		Average	
		Recall	Precision	F-measure
M1: Mondal et al. [46], 2016	Each column of word image	0.7045	0.0504	0.0901
M2: Mondal et al. [11], 2018	Each column of word image	0.5721	0.06324	0.1057
M3:Present Study with 5 metrics having Match Count>2	Entire word	0.528481	0.194867	0.192935
M4:Present Study with 5 metrics having Match Count>3	Entire word	0.436347	0.209409	0.184615
M5:Present Study with 3 metrics having Match Count>1	Entire word	0.566431	0.164999	0.200428

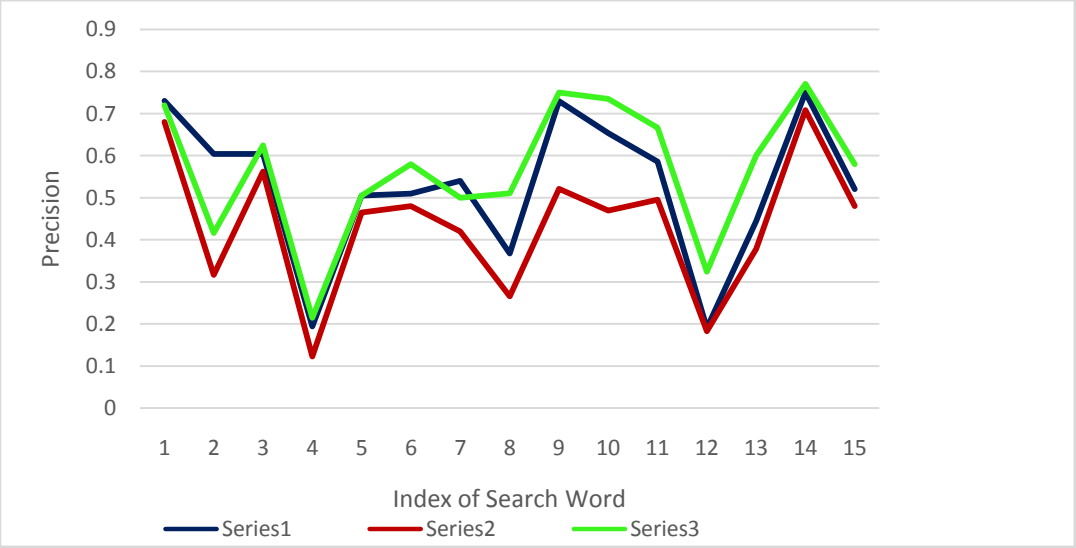


Figure 4.4.1: Comparative study of the Precision values of three Methods (M1:Series1, M2:Series2, M3:Series3) of words searching techniques.

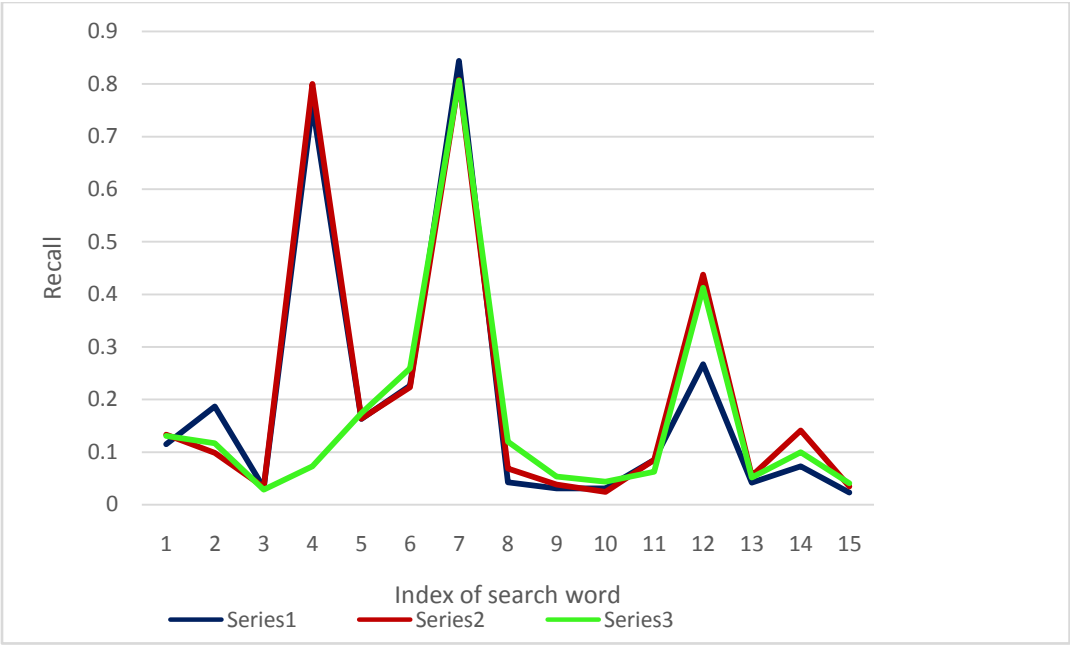


Figure 4.4.2: Comparative study of Recall values of three Methods (M1:Series1, M2:Series2, M3:Series3) of word searching techniques.

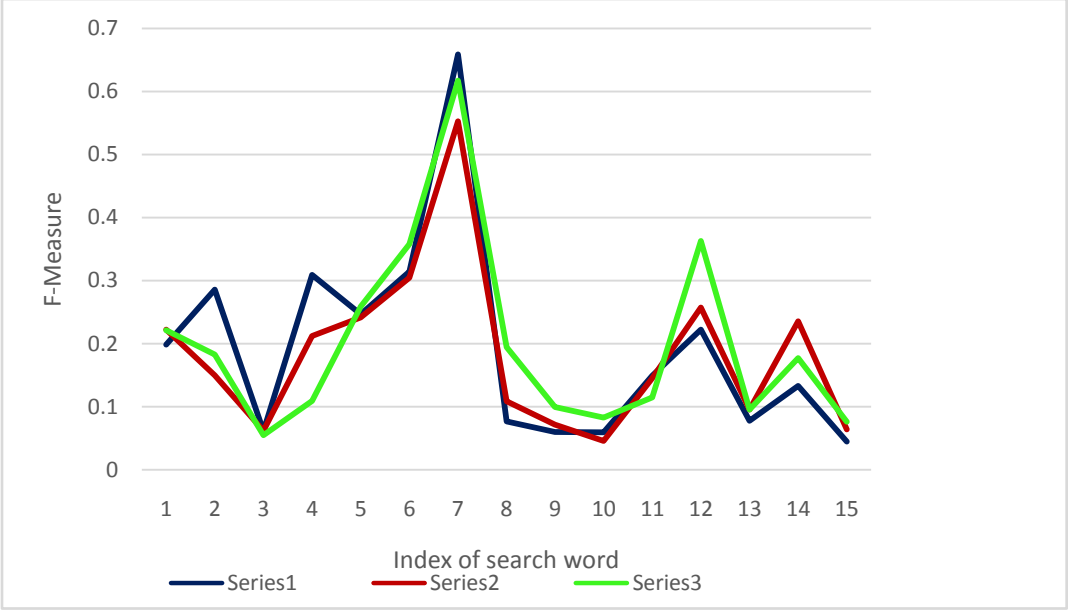


Figure 4.4.3: Comparative study of F-Measure values of three Methods (M1:Series1, M2:Series2, M3:Series3) of word searching techniques.

4.5 Error Analysis

Let, n_1:= the actual number of search words in the document
And, n_2:= the number of fetched target words

Error $\Delta = n_1 - n_2$

Ideally $\Delta=0$ (i.e., $n_1=n_2$), when the accuracy level is 100%.In other words, the model can recognize all the words successfully. If $n_1>n_2$ we call it as positive error and if $n_1<n_2$ it is known as negative error.

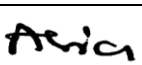
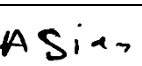
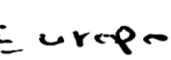
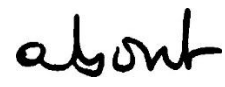
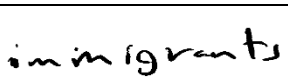
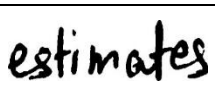
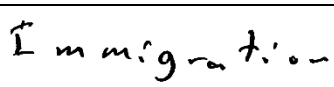
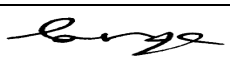
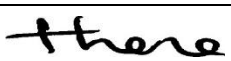
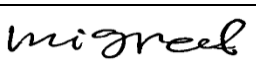
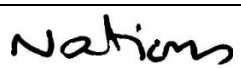
In our experimental setup we get the value of Δ for different SWs in different metrics, among those the result of DTW is shown in Table 4.5.1.

Table 4.5.1: Error calculation (Δij) table over DTW for different SWs.

	DTW			
	ACTUAL IN PSW	NO. OF RETRIEVED	NO. OF CORRECTLY RETRIEVED	NO. OF WRONGLY RETRIEVED
SW1	816	580	71	509
SW2	1245	337	31	306
SW3	1529	1084	31	1053
SW4	1192	13	10	3
SW5	339	287	49	238
SW6	303	224	52	172
SW7	100	30	23	7
SW8	1582	251	19	32
SW9	1391	667	36	631
SW10	1077	1030	36	994
SW11	740	560	56	514
SW12	680	164	75	89
SW13	838	361	19	342
SW14	1854	393	34	359
SW15	1637	651	27	624

In case of searching a word image, some wrong word image as a searched word as well as some unbelievably good result are retrieved which is deformed badly. In Table 4.5.2 we can observe some of these examples.

Table 4.5.2: Retrieved (correct, wrong) word images.

Search Word	Successful Retrieval	Unsuccessful Retrieval
America		
Asia		
Asian		
Europe		
Immigrants		
Immigration		
International		
large		
largest		
immigrant		

Chapter 5

5. Conclusion

Word spotting appears as an attractive alternative to obvious recognize-then-retrieve approach when information retrieval is required from historical, ancient or contemporary handwritten document images. With the capability of matching images of word in a quick and accurate way, word collections might be achieved with reasonable accuracy and limited human interaction. Word spotting has the capability to automatically identify word in search and making it possible to use costly human labor more carefully when a full transcription would require.

A two-stage approach for word searching in handwritten document images is an approach to solve the difficulty of word searching in handwritten document page is reflected here. The irrelevant word images are discarded in the pre-selection section. At the stage of confirmation, pre-selected candidate words retrieved from a document image are passed over a voting system. The outcomes of the current model indicate that it leads to a satisfactory performance. Based on voting, confirmed words are stored as search words in designated folder. The experimental outcomes indicate that the proposed two-stage approach of word searching in handwritten document images yields acceptable retrieval performance.

In spite of this achievement, there are still some areas for improvement. First and notable, a high-dimensional feature extraction can be used for word searching purpose. Therefore, application of a feature selection algorithm would be a worthy choice to increase the retrieval performance of the proposed approach. Also, some context-sensitive features in the first stage of the present work could be applied in the recent future.

The results obtained in this work are encouraging preliminary to continue with a further research in different directions. One continuation path might be improving this work in order to manage possible scalability problem that presents some problems to solve, such as different handwriting patterns, different documents structure or variations in structure even within instances of same words. This will lead to ease the process of digitalization of hand written documents in near future.

References

[1] S. Basu, N. Das, R. Sarkar, M. Kundu, M. Nasipuri and D. K. Basu, A hierarchical approach to recognition of handwritten Bangla characters, Pattern Recogn. 42 (2009), 1467–1484.

[2] S. Bhowmik, S. Malakar, R. Sarkar, S. Basu, M. Kundu and M. Nasipuri, Off-line Bangla handwritten word recognition: a holistic approach, Neural Computing. Appl. (2018), 1–16

[3] https://www.explainthatstuff.com/how-ocr-works.html

[4] https://publish.illinois.edu/commonsknowledge/2017/07/19/what-to-do-when-ocr-software-doesnt-seem-to-be-working/

[5] T.M. Rath and R. Manmatha. Word spotting for historical documents. International Journal of Document Analysis and Recognition (IJDAR), 9:139{152, August 2006.

[6] Zaher Al Aghbari and Salama Brook, "HAH manuscripts: A holistic paradigm for classifying and retrieving historical Arabic handwritten documents", Expert Systems with Applications 36 (2009) 10942–10951

[7] J.L. Rothfeder, S. Feng, and T.M. Rath. Using corner feature correspondences to rank word images by similarity. Computer Vision and Pattern Recognition Workshop, 2003.CVPRW'03.Conference on, 3:30{35, 2003.

[8] MunaKhayyat, Louisa Lam and Ching Y. Suen, "Learning-based word spotting system for Arabic handwritten documents", Pattern Recognition 47 (2014) 1021-1030.

[9] Andreas Fischer, Andreas Keller, Volkmar Frinken and Horst Bunke, "Lexicon-free handwritten word spotting using character HMMs", Pattern Recognition 33 (2012) 934-942.

[10] Houssem Chatbri, Paul Kwan, and Keisuke Kameyama, "An Application-Independent and Segmentation-Free Approach for Spotting Queries in Document Images", 2014 22nd International Conference on Pattern Recognition.

[11] T. Mondal, N. Ragot, J. Y. Ramel and U. Pal, Comparative study of conventional time series matching techniques for word spotting, Pattern Recogn. 73 (2018), 47–64.

[12] S. Sudholt and G. A. Fink, PHOCNet: a deep convolutional neural network for word spotting in handwritten documents, in: Proceedings International Conference on Frontiers in Handwriting Recognition, pp. 277–282, 2016.

[13] W. Pantke, M. Dennhardt, D. Fecker, V. Margner and T. Fingscheidt, An historical handwritten Arabic dataset for segmentation-free word spotting – HADARA80P, in: Proceedings of International Conference on Frontiers in Handwriting Recognition, pp. 15–20, IEEE, 2014.

[14] R. Manmatha and W.B. Croft. Word Spotting: Indexing Handwritten Archives IEEE Transactions on Pattern Analysis and Machine Intelligence, 15:446-451, 1997.

[15] Samir Malakar, Manosij Ghosh, Ram Sarkar and MitaNasipuri, "Development of a Two-Stage Segmentation-Based Word Searching Method for Handwritten Document Images", 2018.

[16] T.M. Rath and R. Manmatha. Word image matching using dynamic time warping. IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'03), 2:521–527, 2003.

[17] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection", IEEE Computer Society Conference on Computer Vision and Pattern Recognition, vol. 1, pp. 886–893, IEEE, 2005.

[18] Konstantinos Zagoris, Kavallieratou Ergina and Nikos Papamarkos, "A Document Image Retrieval System", Engineering Applications of Artificial Intelligence 23 (2010) 872–879.

[19] George Retsinas, GeorgiosLouloudis, NikolaosStamatopoulos and Basilis Gatos," Efficient Learning-Free Keyword Spotting", IEEETransactions on Pattern Analysis and Machine Intelligence,2018

[20] M. Rusinol, D. Aldavert, R. Toledo and J. Llados, Efficient segmentation-free keyword spotting in historical document collections, Pattern Recogn. 48 (2015), 545–555.

[21] A. Ebrahimi and E. Kabir. A pictorial dictionary for printed Farsi sub-words. Pattern Recognition Letters, 29:656, April 2008.

[22] https://riptutorial.com/algorithm/example/24981/introduction-to-dynamic-time-warping

[23] https://www.tutorialspoint.com/design_and_analysis_of_algorithms/design_and_analysis_of_algorithms_longest_common_subsequence.htm

[24] http://cgm.cs.mcgill.ca/~athens/cs507/Projects/2002/StephanePelletier/

[25] S. Al Maadeed, W. Ayouby, A. Hassaïne and J. M. Aljaam, QUWI: An Arabic and English handwriting dataset for offline writer identification, in: Proceedings of International Conference on Frontiers in Handwriting Recognition, pp. 746–751, IEEE, 2012.

[26] P. Wang, V. Eglin, C. Garcia, C. Largeron, J. Llados and A. Fornes, A novel learning-free word spotting approach based ongraph representation, Proceedings of International Workshop on Document Analysis Systems, pp. 207–211, IEEE, 2014.