Universität Hamburg
Department Informatik
Cognitive Systems Laboratory, KOGS

Manuscript Analysis: Bag-of-Feature representation for Writer Retrieval

Project Report

Master Project Image Processing

Soubarna Banik

Matr.Nr. 6640587

4banik@informatik.uni-hamburg.de

15.03.2016

Abstract

Writer retrieval finds the documents from the database that are written by the same writer as the input document. This task is specially relevant for analysis of large historical archives, where the information about the writer is often unknown. This project report presents a method for writer retrieval using SIFT and Bag-of-Feature model. For evaluation, IAM dataset for handwriting recognition is used.

Contents

1 Introduction

Writer identification is the process of identifying the writer of a queried handwritten document with reference to a known database of documents written by multiple writers [3]. Whereas, writer retrieval is the process of retrieving all documents similar to the query document from a database. It also provides a ranking of the similarity of the retrieved documents [3]. In manuscript analysis, searching similar documents from a large archive is a common scenario. It is also needed to verify if a piece of manuscript is part of a known document in the archive or not. Often these archives are partially classified or not classified at all. In this context, finding similar documents manually becomes very difficult, tiresome and hence, an expensive job. Automated writer retrieval makes the target of the query much narrower and in turn, makes the job much easier for the person concerned. Hence, writer retrieval is more relevant in manuscript analysis where the task is to find similar documents from a partially known or completely unknown database [3], whereas writer identification is more relevant to forensic investigation, security and financial activities where the database is known [5]. However, the approach for both of the tasks is quite alike.
Writer retrieval is conceptually similar to handwriting recognition, which is different from optical character recognition (OCR). The former is concerned about the handwriting styles, textual features of the document while OCR tries to recognize the textual content of the handwritten documents. To achieve this, OCR needs to ignore the variations of same characters written by multiple writers, but writer retrieval needs to focus on these variations as these contribute towards the identification of the writing style [4].
The main goal of this project is to study the state of the art of writer retrieval in the context of historical document analysis and to design an effective writer retrieval system to aid historians and other professionals working with such documents. This project report is structured as follows - Section 2 gives a brief description of the ongoing work in writer retrieval. Section 3 discusses the architecture of this project in detail and the experimental results are evaluated in section 4. Finally a conclusion is reached in section 5.

2 Related Work

There are two prominent approaches for handwriting recognition - approaches based on global features and local features respectively. As stated in [10], the global features are computed directly from the entire document or by scanning the document line-by-line. Features like number of exterior and interior contours [1] [2], slope or distribution of the writing directions [10], weighted energy of the document [4] or the run-length of background [1] signify the global characteristics of the document. On the other hand, the local features are extracted around multiple interest points in the document. In the global approach, the above mentioned feature extractors are designed manually and hence they are dependent on the

input script. A particular feature extractor that works well for a script may not be suitable for another language. In this respect local feature extractors are more adaptable to a new script. The global features are also extracted at word or character level. As a result, this approach is dependent on a robust segmentation algorithm for separating the foreground from the background. However, this is not an easy task. Old historical documents usually suffer from the problems of blurred ink, faded texts and ink bleed-through, which make it very difficult to segment the lines, words or characters properly. Moreover, for texts which are written in cursive style, often the letters and the lines overlap and the segmentation algorithms fail to identify the boundaries.
An approach which involves local features is Bag-of-Feature (BoF). Fiel et al. implemented BoF for writer identification and retrieval in [3]. In this approach, local features are extracted from the input image and a histogram is generated for the input image. Documents having similar histograms are considered to be similar.

3 Project Architecture

The general approach of BoF is to represent an image as an orderless collection of features - thus the name bag-of-feature. The set of local features extracted from all training images are clustered to form a vocabulary, also called a codebook. Each cluster denotes a representative feature and is called a word. Once an image is queried, features extracted from it are mapped to the nearest clusters or the words and the image is represented as a histogram of the words. This is also known as "term vector". This process is depicted in Figure 1.

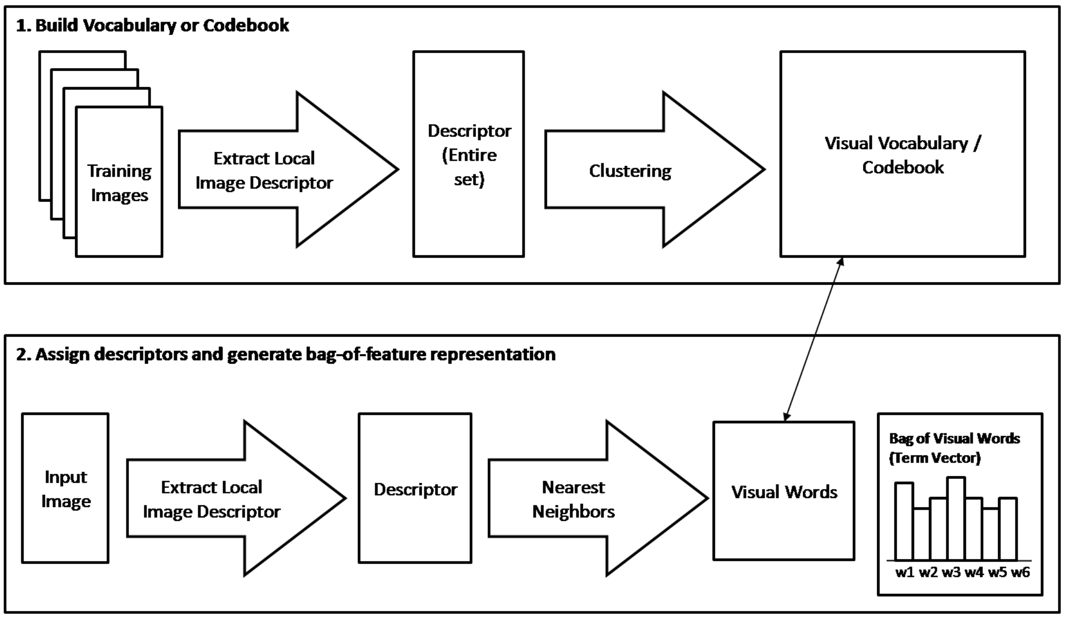

1. Build Vocabulary or Codebook
Training Images
Extract Local Image Descriptor
Descriptor (Entire set)
Clustering
Visual Vocabulary / Codebook
2. Assign descriptors and generate bag-of-feature representation
Input Image
Extract Local Image Descriptor
Descriptor
Nearest Neighbors
Visual Words
Bag of Visual Words (Term Vector)
w1 w2 w3 w4 w5 w6

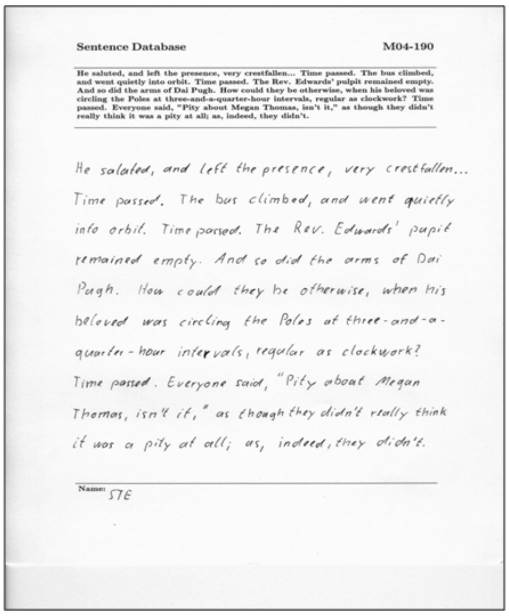
Figure 1: Process for Bag-of-Features Image Representation based on image from [9]

3.1 Dataset

The IAM dataset by Marti and Bunke [7] is used for the experiment. The dataset contains handwriting of 657 writers in English. A collection of texts were used to generate forms and subsequently the writers were asked to fill the forms by copying the text. The number of forms copied by the writers varied from 1 to 59. The total number of forms are 1539, out of which 356 writers have only one sample each. The resolution of the images was set to 300 dpi at a grey level resolution of 8 bit.

3.2 Image Pre-processing

The IAM forms have a fixed format. There is a section of typewritten text at the top of the form, which is bounded by two horizontal lines as depicted in Figure 2. The handwritten text follows the typewritten text which is also bounded by another horizontal line from bottom. In the pre-processing stage, the typewritten text is removed and only the handwritten text is identified and cropped. As the length of the typewritten section varies, this cropping is not done manually. A program is written to identify the three horizontal lines and the section between the last two horizontal lines is cropped. In case the line detection algorithm fails to detect the lines, the image is cropped at default locations, which has been decided after analyzing multiple images.



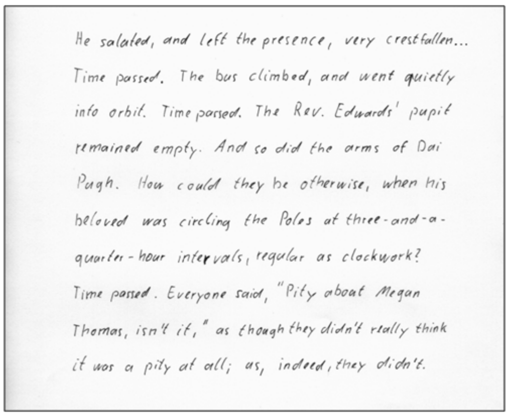
Sentence Database **M04-190**

He saluted, and left the presence, very crestfallen... Time passed. The bus climbed, and went quietly into orbit. Time passed. The Rev. Edwards' pulpit remained empty. And so did the arms of Dai Pugh. How could they be otherwise, when his beloved was circling the Poles at three-and-a-quarter-hour intervals, regular as clockwork? Time passed. Everyone said, "Pity about Megan Thomas, isn't it," as though they didn't really think it was a pity at all; as, indeed, they didn't.

He salated, and left the presence, very crestfallen...
Time passed. The bus climbed, and went quietly
info orbit. Time parsed. The Rev. Edwards' pupit
remained empty. And so did the arms of Dai
Pugh. How could they be otherwise, when his
beloved was circling the Poles at three-and-a-
quarter-hour intervals, regular as clockwork?
Time passed. Everyone said, "Pity about Megan
Thomas, isn't it," as though they didn't really think
it was a pity at all; as, indeed, they didn't.

Name: STE

Figure 2: Sample form from IAM Dataset



He salated, and left the presence, very crestfallen...
Time passed. The bus climbed, and went quietly
info orbit. Time parsed. The Rev. Edwards' pupit
remained empty. And so did the arms of Dai
Pugh. How could they be otherwise, when his
beloved was circling the Poles at three-and-a-
quarter-hour intervals, regular as clockwork?
Time passed. Everyone said, "Pity about Megan
Thomas, isn't it," as though they didn't really think
it was a pity at all; as, indeed, they didn't.

Figure 3: Pre-processing: Sample form after cropping

The shadowed region in the left edge of the sample image and the fixed heading above the typewritten section, as seen in Figure 2 are removed manually before cropping the image. To get rid of the texture of the background paper, the images are smoothed using a Gaussian kernel of size 3x3 and binarized.

3.3 Feature Extraction

Once the image is pre-processed, features are extracted from it. This is done in two phases - Keypoint or interest point detection and Descriptor extraction.

3.3.1 Keypoint Detection

For detecting keypoints, Scale Invariant Feature Transform (SIFT) [6] keypoint detector is used. The objective of this module is to find stable interest points across all possible scales. The range of scales is called scale space. As described in [6] this is achieved by convolving the image intensity function, $I(x, y)$ with Gaussians, $G(x, y, \sigma)$ over the scale space, where σ denotes the scale. The keypoints are detected at the scale space extrema in the difference of Gaussian function, $D(x, y, \sigma)$ convolved with the image.

This can alternatively be explained by a Laplacian of Gaussian function (LoG)

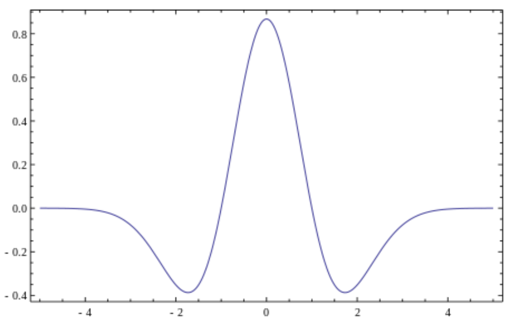


Figure 4: Laplacian of Gaussian (LOG) Operator in 1D

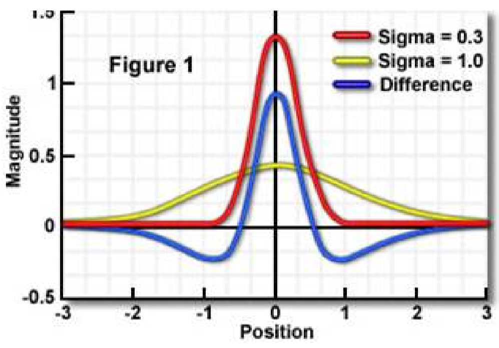


Figure 5: Approximation of LOG - Difference of Gaussian Operator in 1D

which is used as an edge detector and can be approximated by difference of Gaussian function as explained in Figure 5.

$$D(x, y, \sigma) = (G(x, y, k\sigma) - G(x, y, \sigma)) * I(x, y) \quad (1)$$

So the input image is convolved with Gaussians of incremental scales by a factor of k and the interest points are detected by subtracting adjacent Gaussian images and finding the extrema (Equation 1). This is repeated for multiple scale octaves, where the initial image of an octave is produced by downsampling the last image of the last octave by 2. This is depicted in the Figure 6. The default parameters, empirically derived by Lowe [6] are as follows: $k = \sqrt{2}$, initial $\sigma = 1.6$, number of octaves= 4 and number of scale levels= 5. In the project implementation, the Opencv SIFT library is used. A sample output of SIFT keypoint

Figure 7: Sample output of SIFT keypoint detection

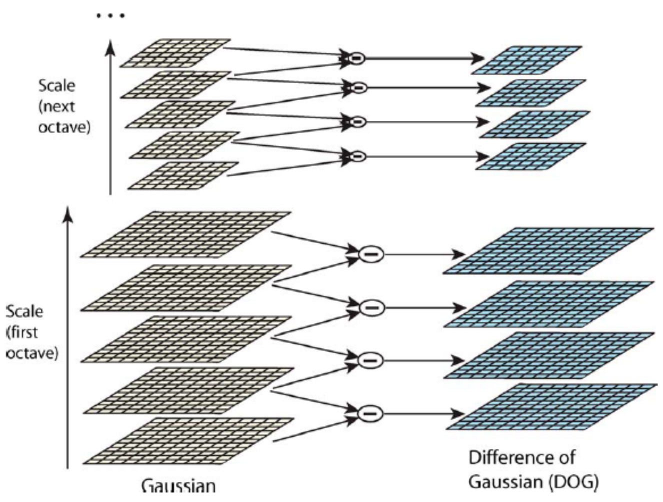


Figure 6: SIFT Keypoint Detection [6]

detection method on the IAM dataset image can be found in Figure 7. The larger circles denote the larger scales.

As quite a few spurious keypoints were detected in the background, the parameters contrast threshold and initial σ are varied to see the effect. The contrast threshold parameter is used to check the intensity values at the extrema points and to reject those having values less than the threshold. The default value is 0.03. In order to remove features in low-contrast regions, threshold is increased. As expected, higher threshold removed the unwanted keypoints (Figure 8). As the variation of sigma does not improve the situation (Figure 9), the default value has been retained.

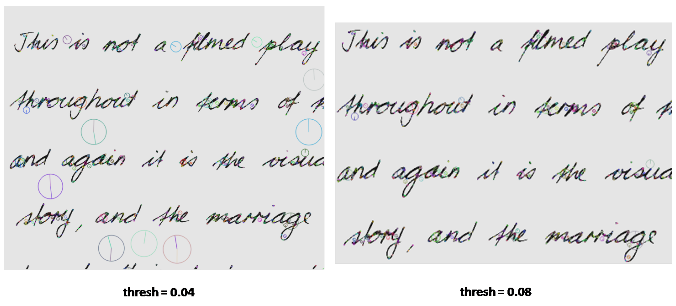


Figure 8: SIFT Keypoint detection with increasing threshold

3.3.2 Descriptor Extraction

Once the keypoints are detected, feature descriptors are extracted around each point. For this too SIFT descriptor is used. SIFT computes the intensity gradients in a 16x16 neighboring window around the detected keypoint. The window is divided into 16 smaller 4x4 sub-windows as shown in Figure 10. In each sub-window, the gradients are computed in 8 directions and a 8 bit histogram is computed. The gradients are weighted by their magnitude and the scale of the keypoint. All 16

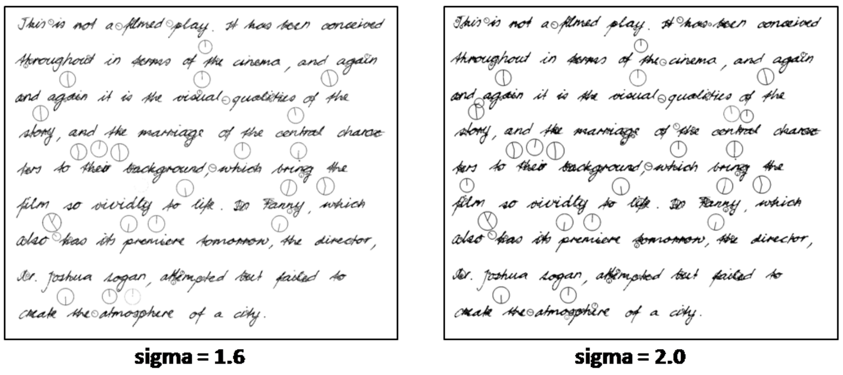


Figure 9: SIFT Keypoint detection with increasing sigma

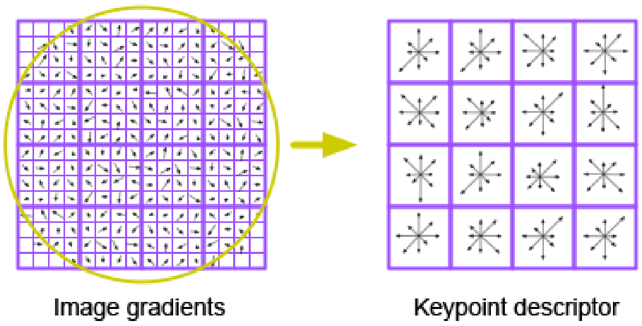


Figure 10: SIFT Keypoint descriptor [6]

histograms are then concatenated to form a 128 bit long descriptor vector.

3.4 Vocabulary generation

Once features are extracted from all training images, a codebook or vocabulary needs to be generated. The vocabulary should be made of representative features computed from the set of all descriptors. For this kMeans++ clustering algorithm is run on the feature vectors. kMeans++ is based on the kMeans algorithm. Though kMeans algorithm is widely used, it can suffer from very slow converge rate. kMeans++ overcomes this by choosing the initial cluster centers rationally. The kMeans and kMeans++ algorithms are listed in Algorithm 1 and 2. The size of the codebook is determined empirically.

3.5 Offline generation of Term Vector

Each cluster center in the vocabulary is called a word or a *codeword*. Once the vocabulary is created, all training images are represented in terms of the computed codewords. The features extracted from an image are mapped to these codewords and the image is represented as a histogram of the frequencies of codewords. The histogram is called the *term vector* and the mapping process is called *Quantization*.

Algorithm 1 kMeans algorithm

```
1: Randomly choose $k$ cluster centers $C = c_1, c_2, ....., c_k$
2: repeat
3:     Assign each data point $x^{(i)}$ to its closest cluster $c^{(i)} := argmin||x^{(i)} - c_j||^2$
4:     Update the position of the cluster centers by the mean of all data points
       belonging to the cluster $C_i$. $c_i = \frac{1}{|C_i|} \sum_{x \in C_i} x, \forall i \in 1, ..., k$
5: until Centers do not change
```

Algorithm 2 kMeans++ algorithm

```
1: Choose the first cluster center $c_1$ randomly
2: repeat
3:     For each data point $x$, compute $D(x)$, the distance between $x$ and the nearest
       center.
4:     Select $x \in X$ as the next cluster center $C_i$ with probability $\frac{D(x)^2}{\sum_{x \in X} D(x)^2}$
5: until $k$ centers are chosen
6: Run kMeans() steps 2 - 5
```

In practice, there are two ways of quantization - hard and soft quantization. In hard quantization, a feature is assigned to its closest matching codeword. On the other hand, in soft quantization a feature is not assigned to any specific codeword, rather the probability of belonging to each codeword is computed.
In this project features are assigned to its closest codeword by using Fast Approximate Nearest Neighbors (FANN) algorithm [8]. As the number of features extracted by SIFT are high and each feature vector is 128 bit long, simple kNN works similar to the expensive linear search - with complexity $O(Nd)$, where N is the number of codewords and d is the dimension of feature vector. On the other hand, FANN analyzes the data to check parameters such as the correlations between dimensions, the size of the data and decides accordingly to either operate hierarchical kMeans tree or multiple randomized kd-trees on the data to find the approximate nearest neighbor.
At the end of this step, a database of term vectors representing the training images is created.

4 Experimental Result

For writer retrieval, when an image of a document is queried to the database, first features are extracted from the input image as described in section 3.3 and a term vector is generated for the input image by mapping the features to the nearest codewords. To get the resultant images from the database, the input term vector is compared with all term vectors from the database and the nearest matches are returned to the user.

To learn the vocabulary, the dataset is divided into training and test sets. For each writer, two samples are considered for training and the rest are kept for testing purpose. However, many writers have only one or two samples altogether. In case of two samples, one is considered for training. For writers having only one sample, it is split into two parts, which are distributed equally between the training and test sets.
Features are extracted on the training images using both SIFT default parameters and for contrast threshold=0.08. Also, the vocabulary is learned for different codebook size, varying from 200 to 1500. In order to compare two term vectors, the distance between the vectors is calculated. For this two alternative distance metrics are considered - χ^2 distance and Cosine similarity of the vectors. χ^2 distance is a weighted Euclidean distance, where each squared term is weighted inversely by the average value (Equation 2).

$$d(v, w) = \sum_{i=1}^{|v|} \frac{(v_i - w_i)^2}{(v_i + w_i)} \tag{2}$$

Cosine similarity between two vectors is a measure that calculates the cosine of the angle between them. In other words, it measures the similarity of the vectors' orientations rather than magnitude. It is computed from the equation of dot product.

$$\theta = \arccos \frac{\vec{v}\dot{\vec{w}}}{||\vec{v}||||\vec{w}||} \tag{3}$$

For each query the top 5 matches are retrieved according to both distance metrics. The overall success rate for chi-square distance and cosine similarity metric are shown in Figure 11 and Figure 12 respectively. Both graphs show the rates for default parameters and contrast threshold=0.08 as well. If out of 5 results, there is at least one successful match, it is considered as success. Top 3 is also calculated in a similar way. As expected, the cosine similarity metric gives better result than the chi-square metric, which basically compares the magnitude of the two vectors. For cosine similarity metric, the best result is achieved at codebook size 600, which is 75.5% for top 5 and 72% for top 3. Also, it is apparent from the graph (Fig 12) that the thresholding has improved the performance. However, the actual performance can be judged from the recall rate. The recall rate for top 5 and top 3 results are 37.09% and 50.56% respectively.
Two major features that need to be considered for classifying handwriting are the curvature and the orientation of characters. Both of these characteristics are ignored, due to the rotation invariance property of SIFT keypoint detection. If the SIFT feature descriptors are computed by following the dense sampling approach, the performance can be improved. Apart from this, after analyzing the top 5 cosine similarity result for codebook size 600, it is observed that out of the 265 cases, where no successful match is found, 81.13% cases are for the writers who have at most two samples. This proves that the shortage of training samples for these writers has adversely affected the performance.
In the histogram generation phase, the FANN algorithm is used to find the nearest

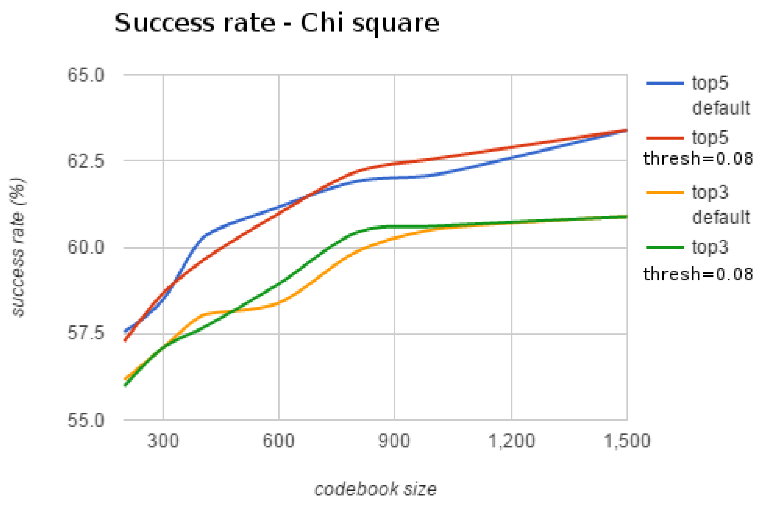


Figure 11: Success rate of prediction (Chi-square)

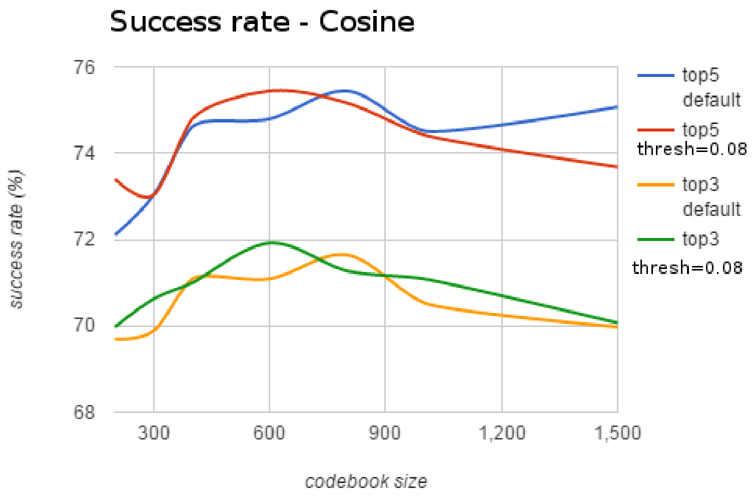


Figure 12: Success rate of prediction for (cosine similarity)

cluster. FANN produces an approximate nearest neighbor and compares Euclidean distance between the vectors. Also, the opencv kMeans++ algorithm internally uses Euclidean distance. The reason for the failure of χ^2 distance metric is also applicable for Euclidean distance. To improve the result, the kMeans algorithm using cosine similarity metric was implemented. However, the performance of the algorithm could not be optimized during the limited project schedule. It can be taken up as a future work. Also while computing the histogram, a soft quantization approach can be considered in stead of hard quantization which is prone to loss of information.

5 Conclusion

A simple method for writer retrieval is presented in this project report. Once queried with a document, the system returns a list of documents of the same writer ranked according to the similarity of the handwriting of the documents. The model uses SIFT features and a bag-of-feature approach. A vocabulary is generated from all extracted features and a database of histogram of the codewords is created for all training image. The histogram of the queried image is compared with the database and both cosine similarity distance and χ^2 distance are computed. A success rate of 75.5% is achieved with the cosine similarity metric. Few techniques that can be investigated for improving the result are dense sampling approach for extracting SIFT descriptors, a nearest neighbor matcher based on cosine similarity and soft quantization.

References

[1] Marius Bulacu, Lambert Schomaker, and Louis Vuurpijl. Writer identification using edge-based directional features. In *null*, page 937. IEEE, 2003.

[2] Daniel Fecker, Abedelkadir Asit, Volker Margner, Jihad El-Sana, and Tim Fingscheidt. Writer identification for historical arabic documents. In *2014 22nd International Conference on Pattern Recognition (ICPR)*, pages 3050–3055. IEEE, 2014.

[3] Stefan Fiel and Robert Sablatnig. Writer identification and writer retrieval using the fisher vector on visual vocabularies. In *Document Analysis and Recognition (ICDAR), 2013 12th International Conference on*, pages 545–549. IEEE, 2013.

[4] Zhenyu He, Xinge You, and Yuan Yan Tang. Writer identification using global wavelet-based features. *Neurocomputing*, 71(10):1832–1841, 2008.

[5] Georgios Louloudis, Nikolaos Stamatopoulos, and Basilios Gatos. Icdar 2011 writer identification contest. In *Document Analysis and Recognition (ICDAR), 2011 International Conference on*, pages 1475–1479. IEEE, 2011.

[6] David G Lowe. Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60(2):91–110, 2004.

[7] U-V Marti and Horst Bunke. A full english sentence database for off-line handwriting recognition. In *Document Analysis and Recognition, 1999. ICDAR'99. Proceedings of the Fifth International Conference on*, pages 705–708. IEEE, 1999.

[8] Marius Muja and David G Lowe. Fast approximate nearest neighbors with automatic algorithm configuration. *VISAPP (1)*, 2:331–340, 2009.

[9] Stephen O'Hara and Bruce A Draper. Introduction to the bag of features paradigm for image classification and retrieval. *arXiv preprint arXiv:1101.3354*, 2011.

[10] Sargur N Srihari, Chen Huang, and Harish Srinivasan. Search engine for handwritten documents. In *Electronic Imaging 2005*, pages 66–75. International Society for Optics and Photonics, 2005.