THE UNIVERSITY OF SOUTH ALABAMA

School of Computing

# Multi-Script Handwriting Identification by Fragmenting Strokes

BY

Joshua Jude Thomas

A Thesis Proposal

Submitted to the Graduate Faculty of the

University of South Alabama

in partial fulfillment of the

requirements for the degree of

Master of Science in Computer Science

in

Electrical and Computer Engineering

May 2024

Approved: Date:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Chair of Thesis Committee: Dr. Ryan, Benton

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Committee Member: Dr. Tom, Johnsten

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Committee Member: Dr. Prakash, Duraisamy

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Chair of Department: Dr. James, Van Haneghan

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Director of Graduate Studies: Dr. Debra, Chapman

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Dean of the Graduate School: Dr. J. Harold Pardue

# Multi-Script Handwriting Identification by Fragmenting Strokes

A Thesis

Submitted to the Graduate Faculty of the

University of South Alabama

in partial fulfillment of the

requirements for the degree of

Master of Science

in

Computer Science

by

Joshua Jude Thomas

B.S., University of South Alabama, 2021

A.S., Coastal Alabama Community College, 2017

April 2023

# TABLE OF CONTENTS

Page

[LIST OF TABLES iv](#_Toc161048197)

[LIST OF FIGURES v](#_Toc161048198)

[ABSTRACT vii](#_Toc161048199)

[CHAPTER I 1](#_Toc161048200)

[1.1 Verification and Identification Model 2](#_Toc161048201)

[Multi-Script Writer Identification 4](#_Toc161048202)

[Assumptions 4](#_Toc161048203)

[CHAPTER II 6](#_Toc161048204)

[Handwriting Identification 6](#_Toc161048205)

[Single-Script Handwriting Identification 6](#_Toc161048206)

[Multi-Script Handwriting Identification 7](#_Toc161048207)

[Stroke Decomposition 9](#_Toc161048208)

[CHAPTER III 10](#_Toc161048209)

[Methodology of Study 10](#_Toc161048210)

[Stroke Fragmentation Method 11](#_Toc161048211)

[Training and Evaluation 16](#_Toc161048212)

[Model and Method of Analysis 17](#_Toc161048213)

[CHAPTER IV 18](#_Toc161048214)

[Experiments 18](#_Toc161048215)

[CHAPTER VI 23](#_Toc161048216)

[Future Work 23](#_Toc161048217)

[REFERENCES 25](#_Toc161048218)

# LIST OF TABLES

Table Page

1. Experimental results from experiments 1 (train CN test EN), 2 (train EN, test CN), and 3 (train EN+CN, test MIXED) 23

# LIST OF FIGURES

Table Page

1. The Handwriting Identification Model 3
2. (From left to right) Chinese, Bengali, Tamil, and English. Writing Scripts of three different origins that look very different on the word level 5
3. Visualization of the stroke fragmentation process before the filtering criterion

15

1. Visualization of class distributions in the CERUG-CN and CERUG-EN datasets vs, the total number of stroke fragments extracted from each 22

**LIST OF ABBREVIATIONS**

KD Known Document

QD Questioned Document

SVM Support Vector Machine

LLR Log Likelihood Ratio

CNN Convolutional Neural Network

HIM Handwriting Identification Model

HVM Handwriting Verification Model

SD Stroke Decomposition

CERUG Chinese-English database of the University of Groningen

OCR Optical Character Recognition

MSWI Multi-Script Writer Identification

# ABSTRACT

Joshua Jude Thomas, M. S., University of South Alabama, April 2023. Multi-Script Handwriting Identification by Fragmenting Strokes. Chair of Committee: Ryan Benton, Ph.D.

This study tests the effectiveness Multi-Script Handwriting Identification after simplifying character strokes. Character simplification is performed through fragmentation of the character by branching-points and end-points, called **stroke fragmentation** in this study. This process shares similarities with the concept of **Stroke Decomposition** in OCR which attempts to recognize characters through the writing strokes that make them up. The main idea of this study is that the characters of different writing scripts (English, Arabic, Mandarin, etc.) have common shapes which can be extracted and used in the handwriting identification process. The effectiveness of the processes described in this study is tested on the CERUG dataset from the University of Groningen. While not achieving state of the art performance, the results of this study imply that simplifying characters shows promise in use for handwriting identification.

# CHAPTER I

**Introduction**

**Handwriting Identification** is the process of classifying the writer of a Handwritten Questioned Document based on handwriting habits contained in that document. According Harralson and Miller in *Huber and Headrick’s Handwriting Identification: Facts and Fundamentals*, forensics experts commonly compare twenty one “discriminating elements of handwriting” that deal with properties such as word size, word placement, margin sizes, abbreviation choices, etc. [1].

Computationally assisted Handwriting Identification is performed by extracting visual features from a set of Known Documents (KD) and Questioned Documents (QD). Features from each of the of the KD and QD are extracted. In general, the features of a QD are compared to a database of writers having a set of KD to which the QD is compared against. The goal is to attribute the QD to one of the known writers by detecting similar features; the accuracy of this attribution, generally, increases as the set of KD increases [2].

## 1.1 Verification and Identification Model

Srihari et al. describe two main frameworks for performing handwriting identification, the **Handwriting Verification Model** (HVM), and the **Handwriting Identification Model** (HIM) [2]. Much of the research literature falls under one of the two categories. The **HVM** estimates the likelihood that two documents were written by the same person (2-class classification) [2]. The extracted feature vectors of two documents are compared by a model to produce a similarity score representing **same writer** (very similar) or **different writer** (dissimilar). Alternatively, a dissimilarity score can also be used. The **HIM**, compares the features of a QD to a model of known writers, and directly classifies the writer (n-class classification) [2]. The HIM is typically the more popular of the two frameworks and is used in this study.

Diagram

Description automatically generated with medium confidence

Figure 1. The Handwriting Identification Model

## Multi-Script Writer Identification

In **Multi-Script Handwriting Identification** (MSWI), writers are not limited to one language or writing script. A writer can produce handwritten documents in multiple languages, such English, and Mandarin, write different languages that share a writing script, or even write in the same language but use different writing-scripts across the set of documents. The goal of MSWI, according to the 2018 ICFHR competition paper on the subject, is to find “… writing patterns that are common across different scripts [and] may be exploited to identify the writer”. This problem is based on the assumption that there are ingrained patterns in a person’s handwriting that are stable across different writing scripts [3].

## Assumptions

This study assumes that there is only one writer per document. A document may contain multiple writing scripts, but there is a 1-to-1 correspondence between a document and the set of writers. This study also assumes that the habits of a writer do not vary across a document due to influences such as time [1]. One implicit assumption that may or may not affect the performance of the generated model(s), is that the writing medium and utensil is the same, or similar enough, between each of the datasets.

A picture containing text, document

Description automatically generated

Figure 2. (From left to right) Chinese, Bengali, Tamil, and English. Writing Scripts of three different origins that look very different on the word level.

This study attempts to tackle the Multi-Script Identification problem by breaking down characters into simpler, hopefully more common, shapes. With the idea being that there are similar looking shapes in the sub-parts of a handwritten character that can be used for compare handwritten documents across different writing scripts.

# CHAPTER II

**Literature Review**

## Handwriting Identification

In general, the methodologies presented in the Handwriting Identification literature differ mainly by the features they extract from a handwritten document.

“Individuality of Handwriting” by Srihari et al. is an old and well-regarded paper that tests the hypothesis that “handwriting is individual” [2]. The hypothesis is tested by implementing features based on the twenty-one features by Huber and Headrick, plus a set of “computational features” consisting of a set of eleven “macro-features”, and “micro-features”. The features are tested in both the HVM and HIM, described in the study.

### Single-Script Handwriting Identification

Foroozandeh et al. used a deep transfer-learning approach to perform signature verification [4]. Several popular CNN architectures were used as feature extractors then classified as a genuine signature or a forgery using an SVM. Nguyen et al. uses a CNN to extract the local features of randomly sampled sub-images of a document [5]. The local features were aggregated via a pooling operation and then used to classify the writer. Shaikh et al. use a “Hybrid Deep Learning” approach to perform writer verification [6]. They pair one of two “Auto-Learned” features, a CNN and an Auto Encoder, and one of two “Human Engineered” features, SIFT descriptors and GSC descriptors. The four resulting combinations were trained to classify pairs of “AND” images and the Log-Likelihood Ratio was computed. Wu et al. extracted SIFT key points from segmented word regions to generate a codebook used to compute a similarity score [7]. Jain, Rajiv and Doermann, David approximate the contours of handwritten characters into “k-adjacent” segments [8]. The contours are approximated into lines by a line-fitting algorithm and then sets of 2-4 line segments are taken and described through a feature vector for use in writer identification. Tan et al. extract features from a bounding box and a bounding quadrilateral for writer identification [9].

### Multi-Script Handwriting Identification

The ICFHR 2018 Multi-Script Handwriting Identification competition reports on the successes of four different systems submitted to the competition [3]. These systems are the LIMPAF-I, LIMPAF-II, Tokyo System, and the Nuremberg System. The LIMPAF-I and LIMPAF-II were submitted by the same group; LIMPAF-I uses Uniform Complete Local Binary Patterns (U-LBP) [10] for its feature extraction while the LIMPAF-II uses Oriented Basic Image Features (oBIF) [11]. For classification, both systems used a multi-class SVM. The Tokyo system used two CNNs to extract features from randomly selected sub-images of a writing sample. Features extracted from writing samples were passed into a “Transfer Neural Net” to transform the extracted features, of writing samples of possibly different writing scripts into a more uniform representation, which were then used in classification. The Nuremberg system was actually based on another paper ([12]) which extracted features by a pre-trained CNN. The extracted features were then “PCA-Whitened” and encoded in a visual bag of words algorithm called VLAD. The feature vectors of the Nuremberg systems were classified by measuring the Cosine-Distance between each sample. He et al. tested the power of handwriting junctions on the writer identification task [13]. Junctions are grouped into L-junctions representing points where writing strokes are sufficiently curved, and T, Y and X-junctions where two handwriting strokes intersect. A junction feature is defined containing the center-point, scale (defined as minimum branch-length, where a branch is one part of an intersecting stroke), two to four angles representing the directions the branches of a junction point in, and the “strength of a branch” in a set number of directions. This junction feature is used to form a “junctlet” codebook of common junctions. Finally, the codebook is used to create histogram containing the number of times a type of junction was detected in a document, and the histogram is used as a feature vector for classification. This study also introduces the Chinese-English database of the university of Groningen (CERUG) consisting of a collection of Documents written in Chinese, English, or a combination of both[[1]](#footnote-2). Semma et al. use CNN features encoded into modified VLAD vectors for handwriting identification [14]. Sub-images of a handwritten document are taken around key-points found via the Harris corner detector. And then CNN features are extracted, processed into fixed length VLAD vectors, and classified. Abbas *et al* combines two features descriptors, LBP and oBIF to create a histogram of both over the whole range of the document and then classifying the writer with an multi-class SVM [15]. Ahmed et al. focus on the “ending strokes”, parts of a character appearing at the tail-end, for writer identification by assembling their contours into a code-book [16].

## Stroke Decomposition

**Stroke Decomposition** is a method of reducing a handwritten character into approximate individual strokes. While similar concepts have been applied in Handwriting Identification, This technique typically appears in research on Optical Character Recognition (OCR) which tries to convert a handwritten text into a typed, digital format rather than identify the writer of the text.

Kim et al. decompose Chinese characters into individual strokes by first performing a morphological thinning to reduce each character to a single pixel width [17]. They then segment the characters based on branching points (areas where strokes overlap) and excessively curved segments. The segments are grown morphologically using two modifications on a morphological dilation which use vectors both parallel and perpendicular (elongation and fattening) to the direction of each segment; A more standard dilation, named isotropic expansion, is performed on segments that are not long enough for the elongation step (less than five pixels in the paper). Both Fattening and Isotropic expansion are constrained by an approximate convexity measure. Finally, Grown stroke segments that have intersecting parts are then potentially merged using the same convexity measure as a conditional [17]. Chen et al. convert handwritten Chinese characters into stroke sequences (strings of numbers that indicate the type of stroke by number and the order the stroke was written by the position of that number). They use an encoder-decoder architecture to convert the character images to stroke sequences [18].

# CHAPTER III

## Methodology of Study

In this study, the Stroke Fragmentation process is performed as a separate process from Writer Identification and is performed by constructing a set of filters for each handwritten document. A typical Handwriting Dataset consists of a set of digitized handwritten document images (scanned or photographed). A set of filters, representing simplified stroke shapes extracted from the related document, is constructed for each. Sub-images containing the simplified stroke shape are extracted from the document with the bounding box coordinates of the filters, and then unneeded pixel values (not corresponding to the filter pixels) are removed. Finally, the filtered sub-images are saved to disk, and then later loaded to be trained on a modified ResNet50 CNN model [19]. The final evaluation metrics well be measured on a different writing script dataset. The evaluation metrics used in this study are: Top-1 Accuracy (Categorical Accuracy), Top-10 Accuracy, Precision, and Recall.

## Stroke Fragmentation Method

The main idea of the stroke fragmentation process is to split handwritten characters into simpler shapes, which are hopefully more common between different writing scripts. The handwritten characters of a document are split up using **critical points** consisting of the branching points, and end points the character. The end goal of the fragmentation process is a collection of simplified shapes extracted from each document that could then be used with more conventional handwriting identification methods.

A mathematical graph representation is used to facilitate the finding of the critical points in a document. In this setting, all the characters contained in a handwritten document are represented as a disconnected, planar, multi-graph: , where is a set containing all the critical points of the document and is a multi-set of pairs . Each pair in represents that the critical points are connected by a part of a stroke (A.K.A. a stroke fragment). The simplified shapes we want to extract can then be thought of as subgraphs of the connected portions of the larger document graph. Note that the actual pixel coordinates corresponding to the handwritten character in a document are not represented as the edges or nodes but are instead included as attributes in the nodes and edges of the graph. When extracting a simplified stroke shape, the pixel coordinate attributes are taken from the corresponding nodes and edges and are combined to form the filter of that simplified shape.

General Steps of Stroke Fragmentation:

1. Preprocessing: transform the document image into an inverted grayscale image for feature extraction, and an inverted binary image for forming the filters used in the extraction step.
2. Reduce the characters in the binary image to single pixel width through morphological skeletonization.
3. Convert the morphological skeleton into a graph representation to locate the critical points.
4. Apply an integer label to each edge and take the attributed pixel coordinates of the edge and it’s connecting nodes to form a skeleton of the filter
5. Perform region growing between the skeleton of the filter and the binary image to form different filters.
6. Remove any filter whose bounding box is too small or two large in either two dimensions (filtering criterion)
7. Use the filter(s) and its bounding box to extract the corresponding sub-image from the grayscale image, then remove any pixel values for pixels not coinciding with the pixels in the filter

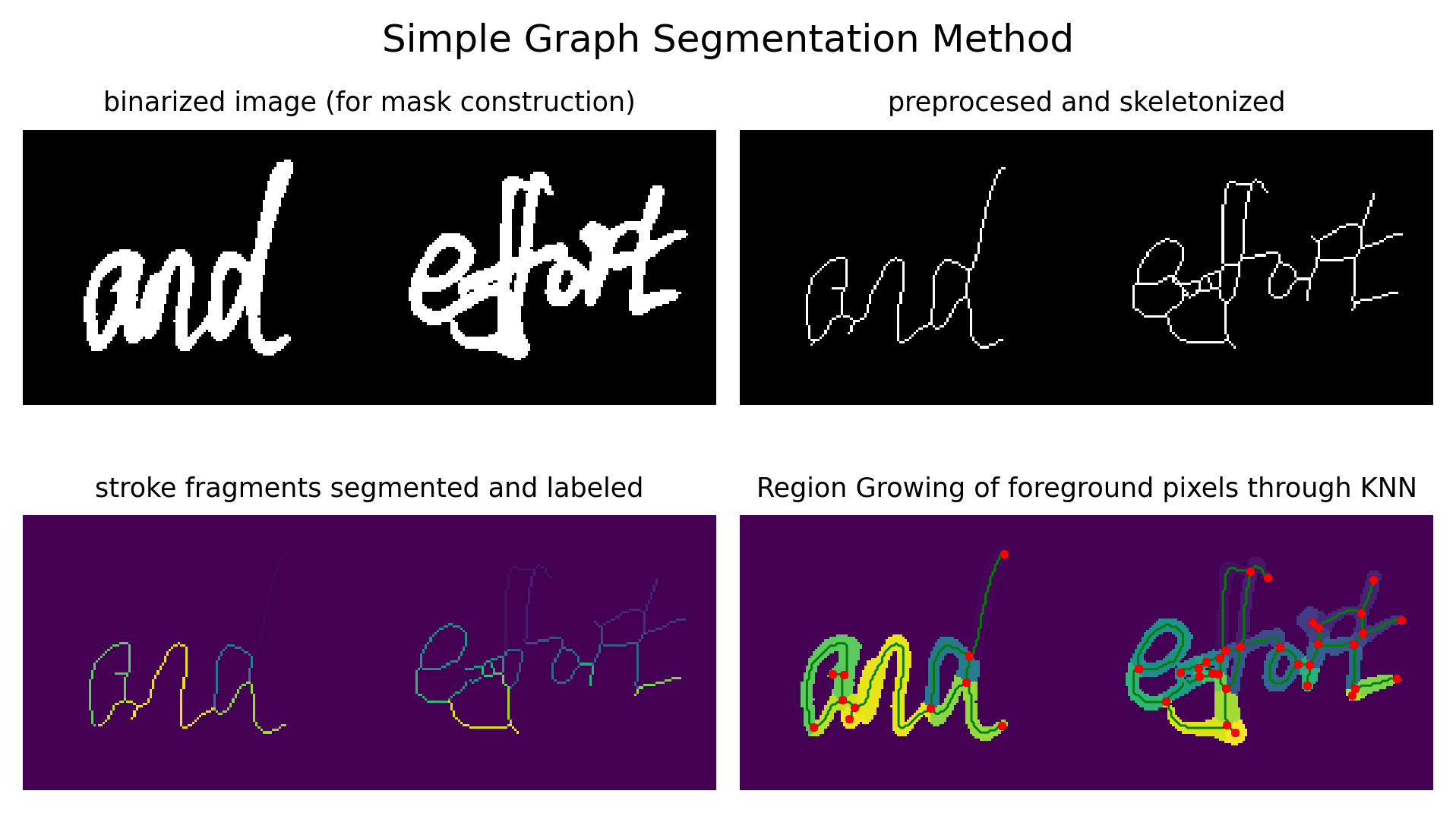


Figure 3. Visualization of the stroke fragmentation process before the filtering criterion. Starting with a raw binary image to construct the filters. It is skeletonized, segmented on the branching points, and the regions of the stroke fragments are grown. Note that the vertex and edge points are plotted over the region grown image for visualization[[2]](#footnote-3).

(step 1) When performing stroke fragmentation on a document, it is converted into a grayscale representation, and a corresponding binary representation. The grayscale representation of the document is kept for the actual extraction stage at the end of stroke fragmentation. The binary image is obtained via Otsu’s method [20], which is then used to form the filters. When processing the binary image(s), an inherent amount of noise will exist in the images of the handwriting dataset, depending on both the method and quality of digitization, and may influence the graph representation. Possible examples are smudges on the paper, visible paper edges from poor digitization, etc. So a gaussian blur which is applied to the image document before binarization (though the blur is not present when actually extracting the strokes).

(step 2) The documents, now binarized and filtered, then go through morphological skeletonization. The skeletonization of the binary shapes in the image reduce them to a single pixel width and try to preserve the general shape of the character as much as possible.

(step 3) The skeletonized handwriting components are then converted into a graph theory representation[[3]](#footnote-4) to facilitate finding the critical points of each character. The graph representing a handwriting document is defined as such that the vertices of the graph correspond to pixel considered critical points in the document. A critical point is all the endpoints of the strokes in the document (pixels connected to only one other pixel) and the branching points of the document (pixels who are connected to three or more other pixels). Every pixel that does not fit the above criteria is considered an edge pixel; The edges of a graph represent critical points connected by a line of edge pixels. When the graph representation is made, the critical point pixels are stored in the corresponding vertices and the edge pixels are stored in the corresponding edge.

(step 4 and 5) During the construction of the filters, each the attributed edge-pixels of each edge is assigned an integer label . These labeled edge pixels represent the skeleton of a simplified stroke fragment. Afterwards, region growing is performed, via KNN, between the filter skeletons and the binarized image to form different filters. That is, the filter skeletons are used as the training data to fit the KNN model, which is then used to segment the pixels of the binary image into the final set of filters.

(step 6) Some artefacts may remain in the image even after the blur is applied. Large artefacts such as visible page edges will be viewed as handwriting strokes by the stroke decomposition algorithm will result in a noisier dataset. In addition to large artefacts, some filters may end up being too small to give any meaningful information and may result in a noise dataset as well. A **filtering criterion** is defined to remove large artefacts and small filters from the final output. The bounding box of each filter is found and used for the feature extraction step (step 7). When the bounding box is found the width and height of the filter is taken. If either dimension is smaller than 3 pixels or larger than 75% the width/height of the image document, the filter is rejected and not included in the output.

(step 7) Finally, each filter is used to extract the corresponding pixels in the inverted grayscale image. The bounding box coordinates are used to define the location of the sub-image containing the desired stroke pixels. After the sub-image boundary is obtained any pixel not coinciding with the pixels in the filter is zeroed out.

The result of the Stroke Decomposition process is filtered sub-images, where is the number of found filters for a document and is the number of filters rejected by the filter criterion. The filtered sub-images into a dataset representing a particular writing-script, and then further into a writer class; A writer may have a writer class for each different writing script collected in the dataset (or in some cases a class corresponding to a mix of multiple writing scripts). After this stroke fragmentation process is performed a standard CNN model can then be trained to the data to perform MSWI.

## Training and Evaluation

In the Stroke Decomposition stage, the scanned handwritten documents of a Dataset are transformed into sets of filtered sub-images containing the approximate strokes fragments making up the characters of that document. The sub-images are saved to a directory split into the different writing scripts of the document (e.g. Arabic, Chinese, English). Each of the writing scripts then contains a directory for each of the writers (who created a document using that writing script). In general, a writing script is treated as a separate dataset from the other writing scripts, and used as the training or testing sets for the modified resnet-50 model.

Writing script directories are loaded as datasets via Kera’s image\_dataset\_from\_directory() method, which infer the writer classes from the names of the subdirectories. The writer class of the sub-images are one-hot encoded and used to predict the probabilities that the sub-image comes from a document written by any of the writers. What writing scripts are used in the Training and Test splits will depend on the experiment being performed. Training sets will be further split into training and validation datasets, using an 80-20 split.

General Steps of the Writer Identification Part:

1. Training
   1. Load training set via image\_dataset\_from\_directory().
   2. Initialize ResNet50 CNN Model with top layers modified to predict writer classes of each dataset.
      1. One model per dataset/experiment due to differing numbers of writer classes.
   3. Train modified CNN on training set and return metrics.
      1. Top-1 (Categorical Accuracy)
      2. Precision
      3. Recall
2. Testing
   1. Load the training set via image\_dataset\_from\_directory().
   2. Evaluate model from training set using top-1, top-10, precision, and recall.

### Model and Method of Analysis

A modified resnet-50 model was chosen as the CNN for this study. In addition to the top-1, precision, and recall metrics, a top-10 accuracy metric is also selected as it is a popular evaluation metric in the literature. Top-N as defined in the ICFHR 2018 Competition on Multi-Script Writer Identification is defined as “… the scenario where the genuine writer of a query document is present within the list of N most probable writers received by the system” [3]. Top-10 is selected to compare with the results of this study with [13].

# CHAPTER IV

## Experiments

The Chinese-English Database of the University of Groningen (CERUG), defined in [13] contains different writing script per writer, for 105 writers. The writing scripts are partitioned into: CERUG-CN for Chinese writing in page 1 and 2, CERUG-EN in page 3 (split over two images), and CERUG-MIXED in page 4 which consists of a mix of Chinese and English. Three experiments are performed in this study (mimicking the tasks in [3]). Experiment 1 uses the CERUG-CN as the training-set, and CERUG-EN as a test set. Experiment 2 uses CERUG-EN as the training set and CERUG-CN as the test set. Experiment 3 merges CERUG-CN and CERUG-EN together for use as the training-set and uses CERUG-MIXED as the test-set.

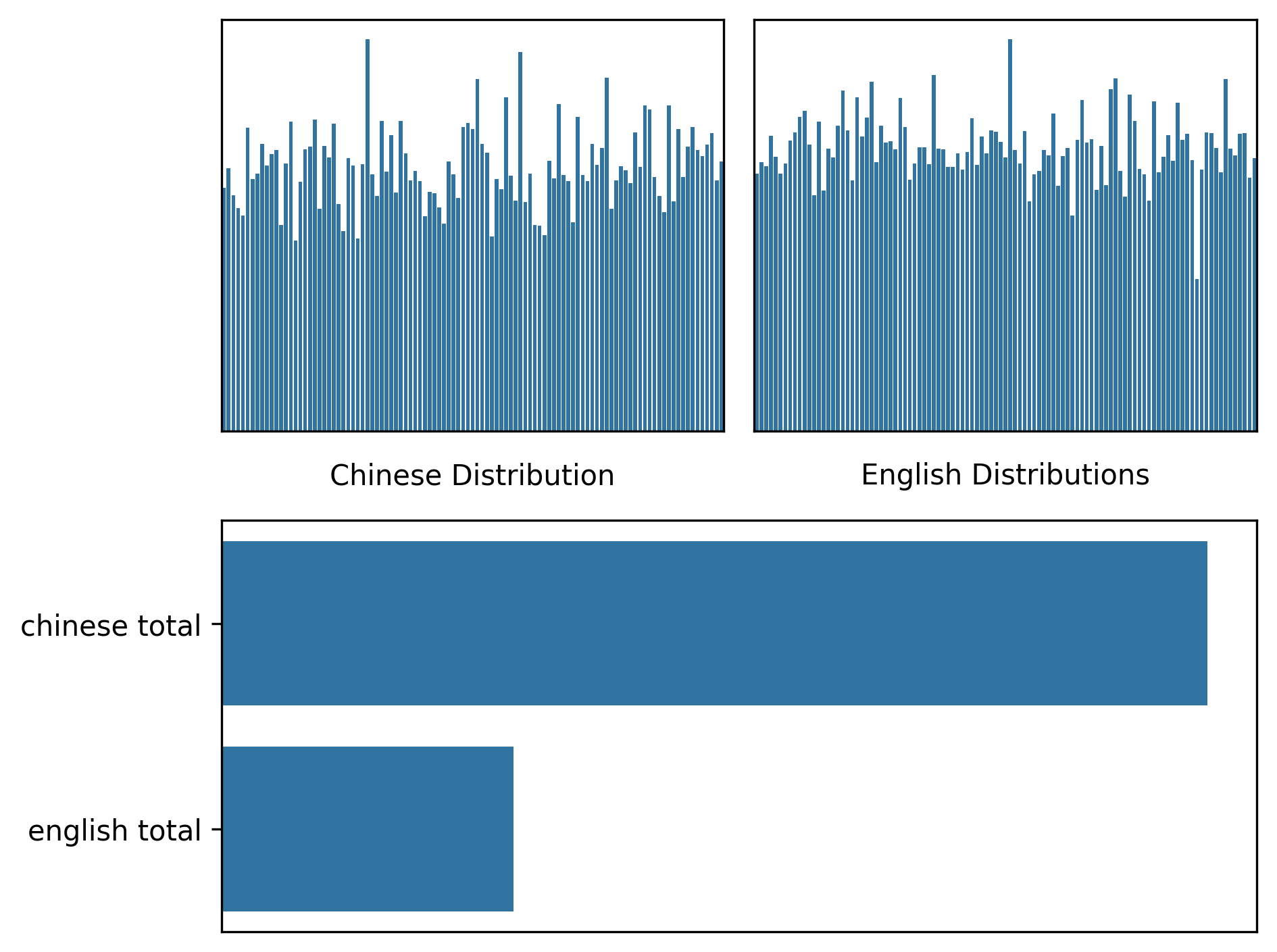


Figure 3. Visualization of class distributions in the CERUG-CN and CERUG-EN datasets vs, the total number of stroke fragments extracted from each.

Figure 3. shows the class distribution CERUG-CN and CERUG-EN vs. the distribution of stroke fragments extracted from each. is of interest to note that, while the individual classes seem relatively balanced, the different writing scripts themselves constitute a data imbalance. In Total: there were 463,507 sub-images extracted from CERUG-CN, 137,258 sub-images extracted from CERUG-EN, resulting in different drastically differing performances in all three experiments.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Experiment Number | Top-1 Accuracy | | Top-10 Accuracy | | Precision | Recall |
|  | Proposed | Junctlets | Proposed | Junctlets | Proposed | Proposed |
| Experiment 1 | 0.365 | 0.907 | 0.865 | 0.967 | 0.524 | 0.266 |
| Experiment  2 | 0.214 | n/a | 0.715 | n/a | 0.318 | 0.138 |
| Experiment  3 | 0.477 | n/a | 0.920 | n/a | 0.662 | 0.355 |

Table 1. Experimental results from experiments 1 (train CN test EN), 2 (train EN, test CN), and 3 (train EN+CN, test MIXED).

Table 1 presents the experimental results from all three experiments. The performance results of the junctlets feature, from [13], are also added for reference. Note that the two metrics are not exactly comparable because the junctlets feature compute a global feature (the occurrence histogram) over the entire document while this study tests CNN accuracy across the fragmented strokes of all documents of a particular script (with an additional split along the training dataset for verification). However the results from [13] are still a good baseline metric.

While not performing nearly as well as the junctlets feature proposed in [13], the evaluation performance does show that the method of simplifying stroke shapes has promise in use for writing identification. A completely random classifier would have an average top-1 (categorical accuracy) score of about 1% for the 105 writer classes of CERUG. The range of 21%-47% implies that there is some information of the writer carried in the stroke fragments produced in this experiment. Furthermore, the top-10 accuracy, having a range of 71-92%, implies that the writer producing a stroke is classified as being in the ten most likely writers, out of 105 writers, for at least 71% of all the strokes in the dataset.

While not directly important to the study, it is of interest to note the overall precision and recall of the trained CNN model for each experiment. The recall performance is noticeably less than the precision performance, and the trend between precision and recall matches the training size between all three experiments (with experiment 3 having the highest, respectively). These scores seem to imply that while the model is not able to correctly classify many of the stroke fragments of a writer, the strokes it does classify as belonging to a particular writer have a, relatively, higher chance of actually being produced by that writer.

As expected with the imbalance of the writing scripts, Experiment 2 is the worst performer of all three experiments due to CERUG-EN being much smaller, at least in the number of extracted stroke fragments, than CERUG-CN. Experiment 3, naturally, has the highest performance since it includes both CERUG-CN and CERUG-EN as the training set.

# CHAPTER VI

**Conclusion**

Multi-Script Handwriting Identification attempts to classify the writer of a handwritten document in a setting where there can be multiple writing scripts or languages in use, and with the possibility that a writer can create documents in more than one writing script. As such, Multi-Script Handwriting Analysis seeks common features between the different writing scripts that are both effective and consistent. This study tested the effectiveness of breaking down the characters of a document into simpler shapes and performed this by breaking down a character along its critical points. While not state-of-the-art, the results of the three experiments performed on the CERUG dataset show that the simplifying character shapes have a promising potential for being used in Handwriting Identification[[4]](#footnote-5).

## Future Work

In this study, the characters are broken down into simpler shapes by segmenting the fragments of a stroke along the “critical points” found in the document. Performance might be increased by adding an additional step that re-merges these stroke fragments into larger, but still simpler, shapes than the whole character, shapes. It may also be interesting to try and train a model directly on the curvature of the skeletons produced during the stroke fragmentation process. The contours of a skeleton, produced during the stroke fragmentation process, may be used to classify the writer by converting those skeleton edges into a chain-code, or similar, representation.

# REFERENCES

[1] H. Harralson and L. Miller, *Huber and Headrick’s Handwriting Identification: Facts and Fundamentals*, 2nd ed. CRC PRess, 2017.

[2] S. N. Srihari, Sung-Hyuk Cha, H. Arora, and S. Lee, “Individuality of Handwriting,” *Journal of Forensic Science*, vol. 47, no. 4, p. 17, Jul. 2002.

[3] C. Djeddi, S. Al-Maadeed, I. Siddiqi, G. Abdeljalil, S. He, and Y. Akbari, “ICFHR 2018 Competition on Multi-Script Writer Identification,” in *2018 16th International Conference on Frontiers in Handwriting Recognition (ICFHR)*, Aug. 2018, pp. 506–510. doi: 10.1109/ICFHR-2018.2018.00094.

[4] A. Foroozandeh, A. Askari Hemmat, and H. Rabbani, “Offline Handwritten Signature Verification and Recognition Based on Deep Transfer Learning,” in *2020 International Conference on Machine Vision and Image Processing (MVIP)*, Feb. 2020, pp. 1–7. doi: 10.1109/MVIP49855.2020.9187481.

[5] H. T. Nguyen, C. T. Nguyen, T. Ino, B. Indurkhya, and M. Nakagawa, “Text-independent writer identification using convolutional neural network,” *Pattern Recognition Letters*, vol. 121, pp. 104–112, Apr. 2019, doi: 10.1016/j.patrec.2018.07.022.

[6] M. A. Shaikh, M. Chauhan, J. Chu, and S. Srihari, “Hybrid Feature Learning for Handwriting Verification,” Aug. 2018, pp. 187–192. doi: 10.1109/ICFHR-2018.2018.00041.

[7] X. Wu, Y. Tang, and W. Bu, “Offline Text-Independent Writer Identification Based on Scale Invariant Feature Transform,” *IEEE Transactions on Information Forensics and Security*, vol. 9, no. 3, pp. 526–536, Mar. 2014, doi: 10.1109/TIFS.2014.2301274.

[8] R. Jain and D. Doermann, “Offline Writer Identification Using K-Adjacent Segments,” in *2011 International Conference on Document Analysis and Recognition*, Sep. 2011, pp. 769–773. doi: 10.1109/ICDAR.2011.159.

[9] J. Tan, J. Lai, C. Wang, and M. Feng, “A Stroke Shape and Structure Based Approach for Off-line Chinese Handwriting Identification,” *IJISA*, vol. 3, no. 2, pp. 1–8, Mar. 2011, doi: 10.5815/ijisa.2011.02.01.

[10] Z. Guo, L. Zhang, and D. Zhang, “A Completed Modeling of Local Binary Pattern Operator for Texture Classification,” *IEEE Transactions on Image Processing*, vol. 19, no. 6, pp. 1657–1663, Jun. 2010, doi: 10.1109/TIP.2010.2044957.

[11] A. J. Newell and L. D. Griffin, “Writer identification using oriented Basic Image Features and the Delta encoding,” *Pattern Recognition*, vol. 47, no. 6, pp. 2255–2265, Jun. 2014, doi: 10.1016/j.patcog.2013.11.029.

[12] V. Christlein and A. Maier, “Encoding CNN Activations for Writer Recognition,” in *2018 13th IAPR International Workshop on Document Analysis Systems (DAS)*, Apr. 2018, pp. 169–174. doi: 10.1109/DAS.2018.9.

[13] S. He, M. Wiering, and L. Schomaker, “Junction detection in handwritten documents and its application to writer identification,” *Pattern Recognition*, vol. 48, no. 12, pp. 4036–4048, Dec. 2015, doi: 10.1016/j.patcog.2015.05.022.

[14] A. Semma, Y. Hannad, I. Siddiqi, S. Lazrak, and M. E. Y. E. Kettani, “Feature learning and encoding for multi-script writer identification,” *IJDAR*, vol. 25, no. 2, pp. 79–93, Jun. 2022, doi: 10.1007/s10032-022-00394-8.

[15] F. Abbas, A. Gattal, C. Djeddi, I. Siddiqi, A. Bensefia, and K. Saoudi, “Texture feature column scheme for single- and multi-script writer identification,” *IET Biometrics*, vol. 10, no. 2, pp. 179–193, 2021, doi: 10.1049/bme2.12010.

[16] A. A. Ahmed, H. R. Hasan, F. A. Hameed, and O. I. Al-Sanjary, “Writer Identification on Multi-Script Handwritten Using Optimum Features,” *KJAR*, vol. 2, no. 3, pp. 178–185, Aug. 2017, doi: 10.24017/science.2017.3.64.

[17] J. W. Kim, K. I. Kim, B. J. Choi, and H. J. Kim, “Decomposition of Chinese character into strokes using mathematical morphology,” *Pattern Recognition Letters*, vol. 20, no. 3, pp. 285–292, Mar. 1999, doi: 10.1016/S0167-8655(98)00147-0.

[18] J. Chen, B. Li, and X. Xue, “Zero-Shot Chinese Character Recognition with Stroke-Level Decomposition.” arXiv, Jun. 22, 2021. doi: 10.48550/arXiv.2106.11613.

[19] K. He, X. Zhang, S. Ren, and J. Sun, “Deep Residual Learning for Image Recognition.” arXiv, Dec. 10, 2015. doi: 10.48550/arXiv.1512.03385.

[20] N. Otsu, “A Threshold Selection Method from Gray-Level Histograms,” *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 9, no. 1, Art. no. 1, Jan. 1979, doi: 10.1109/TSMC.1979.4310076.

1. The CERUG dataset is used for this study and the results are compared to the presenting paper. [↑](#footnote-ref-2)
2. Due to the color-map used, the stem of the d in ‘and’ may not appear visible. [↑](#footnote-ref-3)
3. Credit to <https://github.com/Image-Py/sknw/tree/master/sknw> for providing the code to generate a graph theory representation. [↑](#footnote-ref-4)
4. The Experimental Results, Publication, Code, and Figures can be found at <https://github.com/justjude97/MultiScript-Handwriting-Identification-with-Stroke-Decomposition> [↑](#footnote-ref-5)