Recognizing Handwriting Styles in a Historical Scanned Document Using Scikit-Fuzzy c-means Clustering

Sriparna Majumdar[1,2], Aaron Brick[1]

[1] Computer Science Department, City College of San Francisco, San Francisco, United States of America

[2] Department of Psychology, Santa Clara University, Santa Clara, United States of America

Correspondence: smajumda@mail.ccsf.edu

Abstract:

The forensic attribution of the handwriting in a digitized document to multiple scribes is a challenging problem of high dimensionality. Unique handwriting styles may be dissimilar in a blend of several factors including character size, stroke width, loops, ductus, slant angles, and cursive ligatures. Previous work on labeled data with Hidden Markov models, support vector machines, and semi-supervised recurrent neural networks have provided moderate to high success. In this study, we successfully detect hand shifts in a historical manuscript through fuzzy soft clustering in combination with linear principal component analysis. This advance demonstrates the successful deployment of unsupervised methods for writer attribution of historical documents and forensic document analysis.

Introduction:

Handwriting recognition and separating handwriting into different scribes are computationally challenging tasks. Scribe attribution to handwriting is central to the forensic document analysis of new or historical handwritten documents ([1]). This process requires automation to reduce human bias. In the recent times many research groups in computer vision have focused efforts to build digital optical character recognition tools to facilitate handwriting based classification of historical archived documents([2, 3]), offline signature verification([4]) and associate quality of handwriting to visual cognition and motor activities in human subjects of normal, diseased and aged groups([5, 6]). The handwriting recognition problem for historical documents can be broken down into two smaller problems 1) One writer, many languages and 2) Many writers, one language. While the first problem requires some idea of basic units of languages or lexicons, the latter would not require much idea about the language, vocabulary or grammar used, what each character meant, how they should look like and how they should be strung together to satisfy the semantics. Handwritten documents written in a single language can be classified as single writer or multi writers, solely based on how many different handwriting styles are detected. The individual cursive handwriting is often a combination of all the benchmark cursive forms we are aware about ([7]). Two writers would characteristically use variable lengths of connected strokes, would lift pen at different frequencies, have characteristic stroke width pertaining to the quality of the tip of the pen and pressure applied, introduce characteristic

slants, loops, tails and other flares to the beginning and end of characters or words, have characteristic height and width of individual characters ([7, 8]).

Many of the previous works rely heavily on segmentation of words into graphs of characters ([9]). Handwriting features were extracted from the graphs using Hidden Markov model (HMM) ([10]) and other mathematical feature extracting algorithms, like Levenshtein distance analysis that calculates the cost of converting one grapheme into another ([11]), combining stroke width with direction of stroke ([12]), discrete cosine transform that calculates the sum of cosine functions to represent a finite sequence of data points ([13]), cloud of line distribution that yields unique characteristics of characters based on angle information between dominant points ([14, 15]), to name a few. Those features were then classified into scribes using linear or non-linear component reduction and supervised support vector machine algorithms (SVM) ([16, 17]), convolutional neural network (CNN) ([18]) or recurrent neural network (RNN) ([19, 20]). These supervised and semi-supervised methods work moderately well in the writer classification, with up to 97% accuracy in best cases. Refer to Eskenazi et al. for a comprehensive survey ([21]).

Fewer studies have looked into the use of unsupervised clustering to distinguish handwriting. Vuurpijl and Schomaker introduced a cursivity index which compared the number of letters and the number of strokes conjoining the letters ([22]). Cursivity index of 1 would mean one word was written in just 1 pen stroke and all letters in the word touched their immediate neighbors. Using stroke samples they trained a Kohonen Self-Organizing Map neural network which generated 14 features. Using these features they could cluster print, cursive and mixed styles of writing samples into agglomerative hierarchical clusters. Korovai and Marchenko extended this study by introducing curvature and slants of strokes as features ([23]). They used 12 dynamic features and 22 spatial features and k-means algorithms to group handwriting samples into clusters.

For this study we asked ourselves "Given a set of black & white handwritten documents, can we cut out word-sized pieces and cluster them, to reveal how many different scribes did the writing?" To answer that we extracted 10 handwriting features characteristic of individual writers from scanned historical manuscripts written in Spanish ([24]), taken from the "Archives of California" available at the Bancroft Library. We analyzed the 445 pages long volume "BANC MSS C-A 35", and hereafter refer to it as C-A 35. We have used tesseract to define textual lines and words image bounding boxes (bbox) and scikit-learn, scikit-image, SciPy and OpenCV Python libraries, available in Anaconda for Windows 10.0, to extract the following features from those bboxes: stroke width as a measure of pen tip used and the pressure applied; connected component, the area of an image bbox containing writing connected by a continuous curved line, as a measure of cursive flow which is dependent on the frequency at which the pen was lifted; corner angle, the angle between corner inked pixel and its neighboring highest intensity center pixel, as a measure of loops in writing and flares seen at the beginning and tail ends of characters and words; orientation of the letters/words/lines as a measure of forward or backward slant; convex area, as a measure of total polygonal image area taken up by each connected component; height, width, aspect ratio distribution of writing in an image bbox, as a measure of character or word size characteristic of each scribe and blobLoG and blobDoG, as statistical measures of the diametric extent of connectivity of constant intensity pixels which are

functions of the length of cursive lines. Using these features, linear and non-linear principal component analyses and unsupervised scikit-fuzzy c-means soft-clustering algorithm we could successfully cluster three visually distinguishable writing into three distinct scribes. In effect this study offers a tool to separate handwriting in an offline document featuring mixtures of scribes. It remains to be seen whether our method can be used as a universal tool for recognizing handwriting in all languages, and in all document formats.

Methods:

The Data: C-A 35 Document

The scanned 19th century C-A 35 manuscript is available at the archive.org with unique identifier 168036072_80_17 ([24]). It is a Los Angeles department of state document written in Spanish language. The 445 pages long document features volumes 5-8 that were written by hand and shows evidence of 3 visually distinguishable handwriting styles, hereafter aliased as "hands".

The PDF of scanned C-A 35 document was subjected to pdf2image command line tool to batch-convert pages to 200 dpi PNG images ([25]). Detection of lines and word bounding boxes (bboxes) was done using command line tesseract tool ([26]). The bboxes, found in hOCR files, were used as references to crop portions of the PNG images that were subjected to various feature extraction algorithms. The hands with similar size, meaning and look of characters differ in their extent of cursive flow. Our feature extraction algorithms are designed to extract many physical measures of the extent of cursive flow characteristic of each hand. In addition we extracted stroke width, angular components as measures of slants, loops and flares, height and width of characters and words.

All the feature extracting modules required some preprocessing of the input image. The line and word images were filtered using unsharp-mask, converted to grayscale and inverted to create black background images with white writing before extracting corner angle and stroke width. Height, width measurements using OpenCV and convex area, orientation measurements using SciPy.ndimage.label required conversion to grayscale image with white background and black/gray writing. Convex area and orientation extraction additionally required unsharp-mask filtering of the raw image. Extraction of connected component, blobLoG and blobDoG required conversion to grayscale and inversion of the image to create black background image. Stroke width and connected component extraction required additional preprocessing: thresholding of the inverted image using OTSU filter, binary image creation and skeletonizing.

The following features have been extracted from the hOCR line and word bboxes of the C-A 35 document:

Corner Angle ([27, 28])

The skimage.feature method corner_peaks was used to calculate corner peaks as highest intensity pixels found in the corners of an image bbox. These corner peaks were compared to the highest intensity center pixel to calculate corner orientation using method corner_orientations. The corner orientation, by definition, is the angle (-180 to +180 degrees) of the vector from the corner

coordinate to the intensity centroid found in the local neighborhood, calculated using first order central moment.

Stroke Width ([29, 30, 31, 32])

The stroke width is defined as the perpendicular distance of the two edges of a pen stroke. We used distance_transform_edt of SciPy.ndimage to calculate the distance transformed image. Every pixel of the transformed image gave the distance value from its nearest 0 valued pixel. A Boolean mask was created from the skeletonized image that stored True values for center pixels of strokes and False values for everything else. Using this Boolean mask, the distance values for the centers were extracted. Stroke width was calculated as twice the distance of the center pixel from its closest 0 valued pixel.

Convex Area and Orientation ([33, 34, 35])

skimage.measure provides regionprops method that returns extensive statistics about an image that is segmented into labeled areas. We used SciPy.ndimage.label function to group pixels into labels where all pixels with common label have same integer value and connect to each other either horizontally, vertically or diagonally. For each labeled area regionprops extracts the value of convex area, which is the area of a polygon that fits to a particular labeled area. This gives a finite measurement of the total area of individual connected components that is a function of the length of cursive stroke used by the writer. This measurement is similar to the "connected components" feature we discuss later in this section. Orientation of the labeled area is calculated as the angle between the 0th axis and the major axis of the ellipse that has the same second order moment as the labeled area. The orientation ranges between -90 and +90 degrees.

Height, Width, Aspect ratio ([36, 37])
OpenCV ([38]) provides the cv2 module that is efficient in segmenting images of hands into bboxes of lines, words and characters containing similar values grouped in contours ([35, 36, 37]). We used it to calculate height and width of segmented bboxes of a given image. Height is a good measure of the size of individual characters. We chose height of bboxes that are less or equal to 100 pixels in width, to make sure that we sampled heights of characters or graphemes. Widths of bboxes are representative of character widths, and pen strokes used to write words in a cursive flow or extent of a connected line. We used all widths and heights to calculate aspect ratio (width/height) which is a well-recognized characteristic of individual hands.

Connected Component ([39])

Connected components were calculated using skeletonized images as blobs with connectivity = 2, meaning diagonally connected components were included alongside horizontally and vertically connected components. skimage.measure.label method was used to calculate labels for all blobs. Using those blob labels, skimage.measure.regionprops calculated statistics of all segmented regions. The areas of all the regions were stored as the connected components. The difference between features "convex area" and "connected component" is that connected components were calculated using skeletonized images which were segmented and grouped into labels using skimage.measure.label, whereas convex area features were extracted from raw grayscale images that were segmented using SciPy.ndimage.label. We used two different

methods of segmentation, taken from SciPy and scikit-image libraries, to make sure that we included all cursive components, picked by two independent segmentation algorithms.

BlobLoG ([40, 41]) and BlobDoG ([42])

Blobs are regions in an image in which some properties are nearly constant compared to the neighboring regions. Computer vision's blob detection methods can thus be used to identify cursive writing, in which the connected characters would have same stroke width, forward or backward leaning orientation and pixel intensity, properties that can be used to group them in a single blob. Most common blob detectors are based on Laplacian of Gaussian method (LoG). In this method the image is convolved using a Gaussian kernel for a certain scale to generate a scale-space representation, which is then subjected to a Laplacian operator. The Laplacian is the sum of second order partial derivative of the Gaussian function for each independent variable. In Difference of Gaussian (DoG) approach, the difference of two Gaussian kernel convolutions is computed. LoG can be interpreted as the limiting value of DoG and hence is not too different in their methods of measurements. We extracted diameters of both blobLoG and blobDoG for all hands ([43]), to increase the probability of detecting the true extent of a cursive piece of writing.

Standardscaling of data and principal, independent and kernel principal component analyses

One main problem with extracting features from bboxes is that the data in neighboring bboxes are strongly correlated and so do the extracted features. To reduce association between features of neighboring bboxes we used principal (PCA), independent (ICA) and kernel principal (KPCA) component analyses, that project the data into orthogonal spaces and transform the data to maximize the variance in their numeric values. The features require to be scaled suitably for the PCA analyses. Of all the scaling methods provided in sklearn.preprocessing module, StandardScaler ([44]) and MinMaxScaler ([45]) worked the best. We chose StandardScaler for all scaling purposes. StandardScaler centers the data around zero mean and unit variance. The standard scaled value of a feature x is calculated as:

$z = (x - a) / s$ Eq. 1

where a is the mean and s the standard deviation.

The mean and standard deviation for each column are calculated from all samples provided in the dataset and are stored in a one dimensional array with size equal to total number of features. This array is referred to while transforming the data into scaled data.

PCA ([46]) does linear dimensionality reduction by using singular value decomposition (SVD) of the data to project to a lower dimensional space. It centers a multivariate dataset and projects it into sequential orthogonal components with decreasing variance. In scikit-learn PCA works like a transformer, which can learn custom number of components and project the input data onto them. ICA ([47]) can be used to distribute multivariate data in subcomponents that show maximal independence. It works well in separating superimposed signals. KernelPCA ([48]) is the non-linear extension of PCA analyses which achieves dimensionality reduction by use of kernels. We have used kernel = cosine and kernel = rbf (rbf: radial basis function), as they can

work well on angular data and data with implicit periodicity, which we expect to see in features relating to cursive writing.

Scikit-Fuzzy c-means soft Clustering Method ([49, 50])

Scikit-fuzzy is a Python module originally written by SciPy community and is available on Anaconda platform for Windows 10.0 where we ran all of our feature extraction and data analyses algorithms. Fuzzy logic stems from the idea that there isn't any absolute truth, just a partial truth. This logic is implemented in fuzzy c-means clustering algorithm. In this algorithm each data point is assigned membership between 0 to 100 percent, into different neighboring fuzzy clusters, defined by their own calculated mean (c-mean). The Euclidean distance matrix generated by skfuzzy.cluster.cmeans method is used to calculate membership of a data in various clusters. Fuzzy partition coefficient (FPC) is a good index for determination of the "goodness of fit". FPC is defined as the distance between the fuzzy centers. The closer the FPC value to 1, the better the fuzzy c-means clustering is and the more separated the clusters are. We used FPC values as a reference to accept or reject the algorithm. Fuzzy clusters with FPC values of 0.7 or higher for 2-center fuzzy clustering and 0.6 or higher for 3-center fuzzy clustering were accepted as good fit for our dataset.

Results:

Extracting the features and generating data containing numeric feature rows

The hOCR files containing bbox coordinates of lines and words and their corresponding PNG images of C-A 35 pages were used to extract 10 features discussed in methods. The pixels at the edges of each page were excluded by cropping the images by 5%. This was done to reduce edge related error in feature extraction. In hOCR files, each line bbox is immediately followed by their corresponding set of word bboxes. Those word bboxes were generated from the image within line bbox coordinate and hence they were a subset of the preceding line bbox. Features were extracted from both line and word bboxes.

Each row of data with numeric features contained 10 data points for each feature, making the total column number 100. The image bboxes that produced less than 5 values for any of the 10 features were excluded, since no reliable statistics can be generated using them. For hOCR image bboxes producing 5 or more values for a single feature, mean and standard deviation were calculated using the values and were fed to a normalized number generator. The normalized number generator used normal distribution to generate random samples with values in the range (mean ± 2 × standard deviation) for each feature. These random samples were used to make the total number of data points per feature equal to a multiple of 10 using the following formulas where N is the total number of extracted feature values, p is the quotient of N and n is the remainder; m is the required number of random samples generated by normalized number generator:

$n = N - p \times 10$ Eq. 2

where $N \geq 5$; $0 \leq p \leq N/10$;

$1 \leq n \leq 9$ when $N \geq 10$, $5 \leq n \leq N$ when $N < 10$

$n + m = 10$ Eq. 3

where $1 \leq m \leq 10 - n$; $(N + m) \bmod 10 = 0$; N, n, m, p are integers

In Eq. 2 N is the total number of original feature values falling in the range (mean ± 2 × standard deviation). This simple Gaussian filtering is required to exclude outliers with huge variance. The line bboxes usually produced more than 5 values per feature, whereas word bboxes produced much fewer feature values, often less than 5, causing rejection of several word bboxes. In case of line bboxes as many rows were generated as possible by serially concatenating 10 values per feature using this formula:

$S_1 = [X1,1, ... X1,10]$, $[X2,1 ... X2,10]$, ... , $[X10,1 ... X10,10]$ for 1st row

OR

$Sr = [Xi,j]$ Eq. 4

Where $1 \leq i \leq 10$; $((r - 1) \times 10 + 1) \leq j \leq r \times 10$;

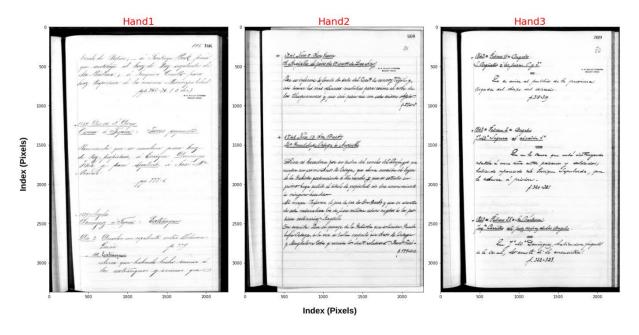$1 \leq r \leq (N + m) / 10$; i, j, r are integers

Figure 1: *The C-A 35 document features 3 visually distinguishable handwriting styles that we call Hand1, Hand2 and Hand3, serialized in order of their appearance in the document. This figure shows representative pages containing each of these handwriting styles.*

In Eq. 4 S_r is the rth row, $X_{i,j}$ is the feature value in ((i-1) x 10+1)th to (i x 10)th columns for ith feature. j is the index of $X_{i,j}$ within the collection of (N + m) values for ith feature and has range ((r − 1) x 10 + 1) to (r x 10) for the rth row. For each line bbox a loop was iterated to generate r rows of data. The rows were then vertically appended to the dataset. In case of word bboxes only one row was generated, and excess feature values for all features were ignored. This was done to minimize duplication of data, as word bboxes are subsets of preceding line bbox, whose features had already been extracted and included in the preceding rows.

Figure 1 shows the examples of 3 distinct hands found in the C-A 35 document. They are named Hand1, Hand2 and Hand3, based on their order of appearance in the C-A 35 document. When we followed C-A 35 document page by page, we found a continuous flow of similar looking writing, presumably by a single scribe. There are two title pages across which handwriting shifted from one writer to another: pages 124 and 260. They had completely different character styling, causing their exclusion. They signal real hand shifts in the document. Hand1 spans pages 2 – 123, Hand2 spans pages 125-259 and Hand3 spans pages 261 – 445. Hand1 has big lettering and long descending lines associated with letters like 'p' or 'g' that contain downward shafts. The capital letters also show characteristic flares at their start. Hand1 writing is sparse and cursive, generating 7619 feature rows from first 90 pages averaging at 84.7 rows per page. Hand2 writing is much denser, with somewhat loopy cursive flow and characteristic flares that are distinctly different compared to other two hands. 13601 feature rows were extracted from first 90 pages of Hand2, averaging at 151.1 rows per page. Hand3 is characterized by thicker pen stroke, sparse writing and large fluctuations in the height of individual characters. 7695 feature rows were extracted from first 90 pages of Hand3 averaging at 85.5 feature rows per page. Among all the 3 hands, Hand1 appears to have the longest cursive lines.

The features described in the methods are designed to extract spatial information about both graphemes and cursive words. Visualization in Figure 2 presents the graphical description of the features analyzed. Figure 2A shows a cropped image taken from Hand3 example in Figure 1. The original scanned document shows black writing on white background. The top 2 text lines are not underlined. The 4 bounding lines of the text, the blue dashed lines, help to distinguish capital letters from small letters. Capital letters found in between 1st and 3rd bounding lines differ in height and width compared to the small letters that are found in between 1st and 3rd, 2nd and 3rd and 2nd and 4th bounding lines. The bottom 2 text lines of Figure 2A contain underlines and digits in addition to alphabets. They require definitions of 5 bounding lines. The mutual distances between these 5 bounding lines in the bottom of figure 2A are evidently different than the top 4 bounding lines. The digits in the underlined texts are found in between 2nd and 4th bounding lines, capital letters are found in between 1st and 4th bounding lines and small letters are found in between 1st and 4th, 3rd and 4th and 3rd and 5th bounding lines. These differences add to the variability in the height feature as well as features like aspect ratio, convex area and connected component that are dependent on the height of a labeled area on the image. Scikit-image, SciPy and cv2 Python libraries segment input images into graphs containing single characters, groups of characters and single words. The height feature represents height of all graphs with width 1-100 pixels. Therefore heights are more representative of single characters. Widths of all segmented graphs were included as features; hence width represents the horizontal
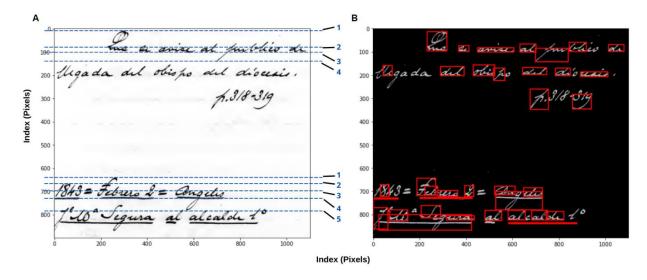
Figure 2: *This figure presents the graphical description of the handwriting features used to cluster handwriting with the help of fuzzy logic. **A**. 1100 X 900 pixels cropped image bbox is taken from page 309, Hand3, shown in figure 1. The text lines toward the top of the image are not underlined. The 4 bounding lines (blue dashed lines) of the writing contain capital letters between 1st and 3rd bounding lines; small letters are found within 1st and 3rd, 2nd and 3rd or 3rd and 4th bounding lines. The text lines in the bottom of the image are underlined and contain digits, causing the distances between bounding lines change. We require 5 bounding lines to define the capital letters, small letters and digits. Capital letters are found within 1st and 4th bounding lines, small letters are found within 1st and 4th, 3rd and 4th or 3rd and 5th bounding lines, whereas digits are found between 2nd and 4th bounding lines. **B**. Measurement of connected component: the image here is the same image shown in **A**, after segmentation by skimage.measure.label. The red rectangles are the bounding boxes of each segmented region. Segmentation uses skeletonized, inverted, grayscale image as reference, and hence pixels with intensity below threshold are not grouped as segmented regions. The bright pixels within one region have similar intensity values and are touching each other horizontally, vertically or diagonally. The areas of the regions are extracted as connected components.*

extent of all segments. blobLoG and blobDoG are composite features, reporting diametric measures of single characters, as well as cursive components along horizontal axis. Similarly aspect ratio, corner angle, orientation, stroke width and convex area are composite features representing both single characters and words. These feature extraction processes require fewer preprocessing steps and report numerous values for a text line. Among all features, connected component extraction involves most rigorous image preprocessing (see methods) and typically produces fewer values than width or convex area. The red rectangular boxes in figure 2B mark all the segmented regions, of the original image bbox shown in figure 2A, whose areas were extracted as connected components. The height and width of the red bboxes were calculated using skeletonized image as reference. Hence words or graphemes written with light pen strokes went below threshold and were not demarked with a red rectangular bbox in figure 2B, as they were not assigned a label. hOCR bboxes from which 5 or more connected components could be

extracted were considered suitable for all features extraction. Width, aspect ratio, convex area, connected component, blobLoG and blobDoG are direct measures of cursivity and are hence classified here as cursive components. In our datasets features were concatenated in this order: 1. connected component, 2. orientation, 3. height, 4. blobLoG, 5. width, 6. convex area, 7. corner angle, 8. stroke width, 9. aspect ratio, 10. blobDoG. The features representing cursive components were interspersed with features that represent non-cursive components.

We noted that in the C-A 35 document first hand shift happens at page 124 and second at page 260. They are referred here as Handshift1 and Handshift2. We used our feature extraction algorithms to extract features from 2-3 pages at either side of the hand shift points. 65 feature rows for each of Hand1 and Hand2 were picked at Handshift1 and 117 feature rows for each of Hand2 and Hand3 were picked at Handshift2. The title pages (124, 260) were omitted while extracting features. The following section compares hands at the two hand shift points.

The features for each hand overlap in feature plane at the hand shift points

The cursive writing is extremely correlated in nature. Stronger correlation is expected to be seen between neighboring word bboxes than neighboring line bboxes. Our data has very large contribution from line bboxes and fewer contributions from word bboxes. When we plotted first 3 PCA reduced components, generated from 100 original feature columns, the hands formed distinct, yet overlapping clusters in three dimensions. Hand1 and Hand2 data around Handshift1, and Hand2 and Hand3 data around Handshift2 are plotted in Figures 3A and 3B. The hands are color coded as Hand1: blue, Hand2: red and Hand3: green. Figures 3C and 3D show plots of the first PCA component against the chronological indices at the two hand shifts. We saw contrasting patterns of distribution at either side of the hand shift points, which are located at the middle of each dataset. Indices lower than the hand shift and indices higher than the hand shift are occupied by data corresponding to two different hands, and hence they are colored differently. Figures 3E and 3F show plots of first PCA component against second PCA component. The hands overlap in these plots, like how they do in Figures 3A and 3B, however the 2-center scikit fuzzy c-means clustering analyses generated 2 clusters (shown in blue and orange) with distinct cluster centers (red squares) and FPC scores greater than 0.7. The fuzzy memberships of each hand from either side of Handshift1 and Handshift2 are indicated in the legends of Figures 3E and 3F. Evidently each hand has membership in both the fuzzy clusters, although their percent distribution varies significantly, proving that scikit-fuzzy can successfully distinguish between handwriting styles by segregating the PCA components differentially among fuzzy clusters. We also tried hard clustering of hand shifts data using Birch, MeanShift, Agglomerative, Gaussian Mixture and k-means algorithms, where one cluster would be presumably assigned to just one hand ([51, 52]). Those algorithms produced inconsistent clustering results, indicating that they were not good enough to cluster our data containing lots of overlaps between the hands, although they produced good results in other studies ([22, 23]).
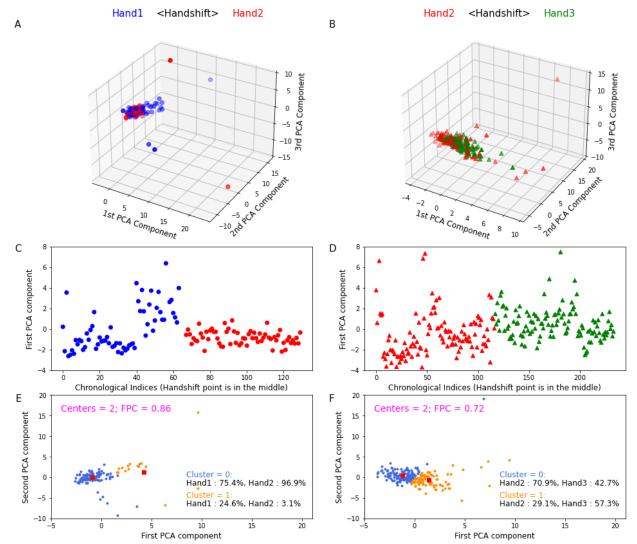
Figure 3: *This figure presents analyses of writing around two existing hand shift points in the C-A 35 document. The handwriting shifts from Hand1 to Hand2 at page 124 and from Hand2 to Hand3 at page 160. 65 feature rows per hand were taken around Hand1<Handshift>Hand2 for analyses in **A**, **C**, **E**. 117 feature rows per hand were taken around Hand2<Handshift>Hand3 for analyses in **B**, **D**, **F**. **A** and **B**, 3-dimensional representations of 3 PCA components generated from data around Hand1<Handshift>Hand2 and Hand2<Handshift>Hand3. The hands are color coded as Hand1: blue, Hand2: red, Hand3: green. The features are distinct for each hand although overlapping in their distribution pattern. **C** and **D**, the distribution pattern of first PCA component at either side of the two hand shift points. Spread of data points on the left side of hand shift, with smaller chronological indices than the hand shift point located in the middle, is distinctly different compared to the spread of data points found at the right side of the hand shift point, at higher indices. **E** and **F**, results of 2-center scikit-fuzzy clustering algorithm generated using first 2 PCA components of the data around both hand shift points. Scikit-fuzzy differentially distributes the hands from each side of the hand shift points, as indicated by their membership percentages shown in the legend.*

Testing efficacy of scikit fuzzy in terms of distinguishing hands around dummy hand shifts

We have tried scikit fuzzy clustering on a big dataset created by serially concatenating first 7600 rows each of Hand1, Hand2 and Hand3, with artificial or dummy hand shifts happening at the junctions of hands. Both 2-center and 3-center fuzzy c-means clustering were successful in differentially distributing the 2 component PCA or ICA reduced data in separate clusters, resulting in good separation of cluster centers with high FPC scores (dataset 1, table 1). 2-center clustering showed better separation of the hands with membership percentages differing by 5% or more. For example, 2 component PCA reduction of dataset 1 from table 1 showed the following membership distribution in the 2-center fuzzy clusters: 0th cluster: {Hand1: 55.8%, Hand2: 95.4%, Hand3: 44.3%}, 1st cluster: {Hand1: 44.2%, Hand2: 4.6%, Hand3: 55.7%}. Evidently, the 3 different hands have unique membership percentages in the 2 clusters that can be used as a signature for each hand. We were unable to try KernelPCA on such huge dataset. KernelPCA algorithm is much slower compared to PCA or ICA, as it performs a non-linear decomposition of features by projecting the features suitably at a lower or higher dimension than the data itself. It could be used successfully for smaller datasets.

The real world handwriting datasets are often quite small, with only a few pages per hand available for feature extraction. Keeping that in mind we created datasets by concatenating data for each hand, picked around pages 124 and 260, the real hand shift points in C-A 35 document. We vertically concatenated equal (datasets 2, 3) or unequal (dataset 4) number of rows for each hand in three different combinations (table 1). 2-center and 3-center scikit fuzzy could differentially distribute the hands in separate clusters for each data combination, as judged by the FPC scores being always greater than 0.7 for 2-center and always greater than 0.6 for 3-center fuzzy clustering (table 1), with PCA reduction of data helping the clustering the most.

To further test the efficacy of scikit-fuzzy in detecting mixture of hands in a given document; we designed dummy hand shift points by putting 2 random pages at either side of a hand shift. We picked 50 consecutive rows for each hand from random locations of the collections containing Hand1, Hand2 and Hand3 data. We called these 50 consecutive rows as 1 page. 2 random pages of one hand, 100 rows of features in total (left side), were concatenated to other 2 random pages (right side) of either same hand (self<Handshift>self) or a different hand (self<Handshift>other). Hand shift point was defined as the junction of the page 2 of left side and page 1 of right side. 2-5 centers fuzzy clustering algorithms were tested for their efficacy in distributing hands among clusters for both self<Handshift>self and self<Handshift>other combinations, as shown in the distribution pattern of the FPC values in Figure 4A. In this plot, summary of 25 random combinations of each of Hand1<Handshift>Hand1 and Hand1<Handshift>Hand2 were compared using first 2 components of PCA, ICA or KernelPCA decomposed data. We saw similar plots for all other self<Handshift>self and self<Handshift>other combinations (not shown). 2-center fuzzy clustering works best for differentially distributing hands in different clusters when tried on 2 PCA components for each dataset, as the mean FPC score is the highest for PCA among all decomposers.

Serial No.	Datasets	No. of Centers / Fuzzy Partition Coefficients (FPC)			
		2 component PCA	2 component ICA	2 component KernelPCA (kernel = Cosine)	2 component KernelPCA (kernel = rbf)
1	7600 rows of each: Hand1+Hand2+Hand3	2-center FPC = 0.82 3-center FPC = 0.75	2-center FPC = 0.83 3-center FPC = 0.76	-	-
2	65 rows each of Hand1 and Hand2 around Handshift1+ 65 rows of Hand3 next to Handshift2	2-center FPC = 0.82 3-center FPC = 0.76	2-center FPC = 0.81 3-center FPC = 0.80	2-center FPC = 0.77 3-center FPC = 0.66	2-center FPC = 0.82 3-center FPC = 0.69
3	65 rows of Hand1 next to Handshift1+ 65 rows each of Hand2 and Hand3 around Handshift2	2-center FPC = 0.77 3-center FPC = 0.73	2-center FPC = 0.76 3-center FPC = 0.70	2-center FPC = 0.74 3-center FPC = 0.66	2-center FPC = 0.77 3-center FPC = 0.68
4	Handshift1 + Handshift2: 65 rows of Hand1, 320 rows of Hand2, 171 rows of Hand3	2-center FPC = 0.82 3-center FPC = 0.81	2-center FPC = 0.81 3-center FPC = 0.80	2-center FPC = 0.77 3-center FPC = 0.64	2-center FPC = 0.82 3-center FPC = 0.70

Table 1: *Four datasets containing various samples of the 3 hands from C-A 35 document were subjected to 2-center and 3-center scikit-fuzzy c-means clustering. The Fuzzy Partition Coefficient is a good measure of the effectiveness of fuzzy distribution of data among distinct clusters. This table shows that in all cases 2-center c-means clustering works better than 3-center clustering.*

Higher number of cursive features gives better success rate in detecting real hand shifts

Ideally if two hands across a hand shift point belong to a single writer as it is in self<Handshift>self combinations, the fuzzy cluster memberships should have been identical for both left and right sides. For the statistical measure of membership differences, we set a tolerance score of 5%. If the membership percentages in the 0th cluster of 2-center fuzzy c-means

clusters differed by more than 5% for the left and right side hands of a dummy hand shift point, we called two sides to be "different" and assigned a Boolean difference value to True, designated by non-equality operator != in figures 4B to 4G, assuming that the left side and right side belong to different scribes. Using this paradigm, we calculated the percentage of times left side and right side of a hand shift were different in this way:
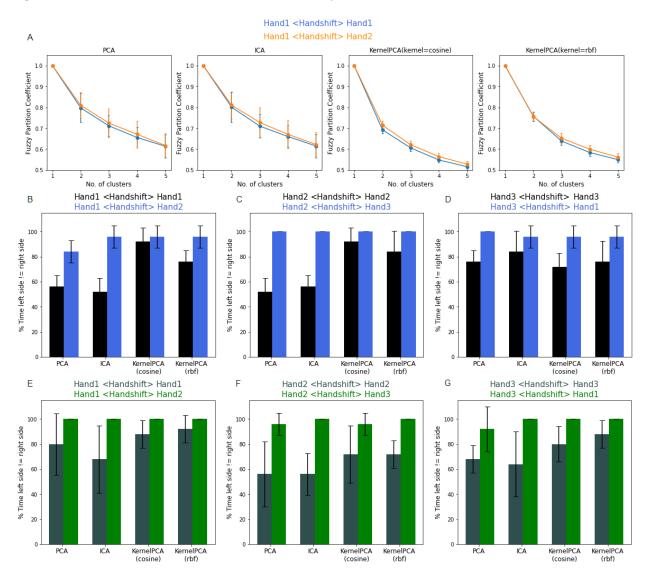

Figure 4: *This figure presents the variation in the fuzzy partition coefficient and fuzzy membership of the hands seen for 2 to 5-center scikit-fuzzy clustering using data around dummy hand shift points. Dummy hand shift points were created either by concatenating 4 random pages of the same hand (self<Handshift>self) or 2 random pages each of two different hands (self<Handshift >other). **A**. The plots combine 25 iterations of picking 4 random pages and concatenating them to create 25 different sets of self<Handshift>self and self<Handshift >other dummy hand shifts, where hand shift point is situated at the junction of pages 2 and 3. We used PCA, ICA, KernelPCA (kernel=cosine) or KernelPCA (kernel=rbf) decomposers to reduce 100 feature columns into 2 components for all dummy hand shifts. Data around all*

dummy hand shift points were clustered best by 2-center scikit-fuzzy c-means algorithm. Four adjacent plots show mean and standard deviation of 25 FPC scores for 2-5 center scikit-fuzzy clustering, in case of each decomposer, as the titles suggest. Hand1<Handshift>Hand1 in blue is compared with Hand1<Handshift>Hand2 in orange. 2-center scikit-fuzzy works best on PCA components, as the mean FPC scores suggest. Similar result was obtained for other self<Handshift>self and self<Handshift >other combinations (not shown here). ***B, C, D,*** *These plots present analyses based on data presented in Figures* ***3E*** *and* ***3F*** *which show that 2-center scikit-fuzzy model can differentially cluster the hands from either side of a real hand shift point. We tested whether the membership percentages of the hand at self<Handshift>self dummy hand shift points are similar or different when comparing data from the left side of the hand shift to that taken from the right side. To be different the memberships in the cluster 0 of 2-center scikit fuzzy model should differ by more than 5%. Plots present the mean and standard deviation of percentages of time left and right side memberships differ by more than 5% in 25 iterations. As the histograms suggest the left and right sides for self<Handshift >other are more likely different in their memberships than self<Handshift>self, for all combinations of self and other, with PCA bringing out the most difference among hands.* ***E, F, G,*** *These plots present the same analyses done for* ***B, C, D,*** *but for 6 selected features (60 columns of features). The selected features were orientation, height, width, corner angle, aspect ratio and blobDoG. These features are more related to graphemes and show most difference in their absolute values when compared across different hands. Plots present the mean and standard deviation of percentages of time left and right sides' membership in the cluster 0 of 2-center scikit-fuzzy differed by more than 5%. The data was summarized using 25 iterations. As the histograms suggest the left and right sides for self<Handshift>other are more likely different in their memberships than self<Handshift>self, for all combinations of self and other, with PCA and ICA bringing out the most difference.*

1. We evaluated 5 random dummy hand shifts using 2-center scikit fuzzy clustering of 2 PCA, ICA or KernelPCA components and calculated the Boolean fuzzy membership differences between left and right sides using preset tolerance of 5%.
2. Sum of all 5 Boolean values was divided by 5 and then multiplied by 100 to calculate percentages.
3. 5 such percentages (calculated from 25 total dummy hand shifts) were used to calculate mean and standard deviation.

Data for both self<Handshift>self and self<Handshift>other combinations of hands, generated using above 3 steps are plotted for Hand1, Hand2, Hand3 in Figures 4B-4D. The plot titles indicate the hand shift combinations tested. It is clear from the histograms that in almost 100 percent of times scikit fuzzy could assign different membership percentages to two visually different hands found at the opposite sides of the hand shift no matter the type of decomposers used (blue bars in Figure 4B-4D). When data for the same hand was placed at the opposite side of dummy hand shifts scikit-fuzzy assigned differential membership percentages to left and right side at about 60% or less cases when 2 components of PCA or ICA reduced Hand1 or Hand2 data were used (black bars in Figures 4B and 4C). Hand3 features seem to show more variability compared to Hand1 and Hand2, as scikit fuzzy assigned different membership values for

different random pages of Hand3 70-80% of time (black bars in Figure 4D). Hand3 features seem to have higher non-linear components, as KernelPCA with a cosine kernel could extract the similarities among Hand3 pages more often than PCA (Figure 4D). Similar intra-writer variabilities in features were observed in Bengali handwriting recognition ([53]). In an attempt to reduce the number of features and increase the difference between self<Handshift>self and self<Handshift>other combinations, with respect to the same hands showing similar fuzzy cluster membership, and distinct hands showing differential fuzzy cluster memberships, we reduced the number of selected features from 10 to 6, by keeping most single character related features and reducing cursive components. We concatenated the selected features in this order: orientation, height, width, corner angle, aspect ratio and blobDoG. Connected component, convex area, blobLoG and stroke widths were omitted, as their absolute values were not too different when compared across 3 hands. This reduced the number of cursive components in feature selected dataset from 6 to 3 and increased the weights of angular features. 2-centers scikit-fuzzy classified the 2 PCA components of feature selected data with an average FPC score of 0.7 or higher. That did not benefit us in our quest to make scikit fuzzy to assign nearly identical membership percentages to same hands found across dummy hand shifts. As the plots in figures 4E-4G suggest, the self<Handshift>self combinations (grey bars) show difference in left and right sides more often in case of Hand1 and Hand2 (Figure 4E and 4F), compared to original data with 10 features (black bars in Figure 4B and 4C). We conclude that cursive features contribute more to the differences in handwriting styles in our samples of handwriting, and all 10 features are required to distinguish handwriting.

Discussion:

Our study intends to detail on 10 handwriting features relating to both graphemes and graphs of words that are written with cursive strokes, demonstrate their effectiveness in numerically defining 3 distinct scribes found in a scanned historical document and show that we can use fuzzy logic to cluster them differentially into more than one cluster. In effect this study offers an unsupervised clustering tool to separate handwriting found in offline documents featuring mixtures of scribes.

In this study we present a way of extracting features using open source image processing tools provided in scikit-learn, scikit-image, SciPy and OpenCV Python libraries accessible on Anaconda platform. We extracted some numeric features related mostly to single characters, like height, and some features that were dependent on both single as well as group of characters joined by cursive lines, like width, aspect ratio, diameter of blobs covering one or many characters, area of one or a group of characters, , corner angle, orientation, and stroke width. We realized that combining many different measures of cursive flow helped us in extracting the essential characteristics of a scribe. Tesseract's hOCR tool was used to detect reference bounding boxes in the scanned images of handwriting. Hence our feature extraction methods relied heavily on hOCR grouping of lines. Most of the Python libraries that we used in our work segmented the hOCR images further, in order to extract spatial measurements related to characters, words and lines. Five or more cursive components were found in most of the hOCR lines, helping us getting a reasonable distribution of them. None of our analyses use any information related to the language properties and lexicons. This makes our methods of treating the data language independent.

One of our primary goals was to propose a way to cluster scribes in an unsupervised manner. Many past works presented well designed feature extraction algorithms that facilitated supervised classification of labeled handwriting datasets with up to 97% accuracy ([21]). Supervised classification would not work for forensic document analytics where none of the handwriting is provided with any label. Our work intended to address that issue by providing a suitable unsupervised method for clustering unknown handwriting found in offline documents. We have shown that unsupervised scikit-fuzzy soft clustering technique can be successfully used in combination with PCA analyses of 10 numeric features, to differentiate between scribes found in C-A 35 document. Fuzzy logic suggests that every hand may belong to more than one cluster. The membership of a data point into a particular cluster depends on the distance of the data point from the cluster center in the feature space. A data point is assigned to the cluster whose cluster center is the closest to it. A hand may have many feature rows that would fall into separate clusters depending on how far they are in the feature space compared to various cluster centers. Each hand is therefore assigned memberships in more than one cluster, based on the distribution of its features rows among the clusters. We calculated cluster memberships of each hand to show that each hand had unique memberships in fuzzy clusters. We have also showed that 10 features are optimum to bring out the most difference between hands, while making sure that writing by a single scribe are assigned similar cluster memberships by scikit-fuzzy c-means clustering algorithm. Machine learning algorithms provided in scikit-learn suite had been previously used for supervised handwriting recognition purposes ([54]). Our work is one of the first that used scikit's soft clustering algorithms for unsupervised clustering of handwriting.

Our results suggest that KernelPCA decomposition does not work well in bringing out the differences in Hand1 and Hand2 despite the nonlinearity in the data. This was surprising for us as we expected KernelPCA to perform better, because of intrinsic nonlinearity in the data and also because of other works that reported better classification scores for kernelPCA ([10]). This could be due to the fact that angular features don't fluctuate in their numeric values as much as the cursive features. Often the features despite being unique may not strongly differ in their numerical aspects. The cursive features vary greatly in value, with a strong linear trend, compared to stroke width or height of characters. However this does not have to hold true for all handwriting. As we demonstrated in figure 4D, Hand3 features had stronger influence of non-linear components, since KernelPCA worked better compared to PCA. We suggest a comparison of PCA, ICA and KernelPCA components for each handwriting using self<Handshift>self dummy hand shift clustering to come up with the best reducer for each handwriting style before comparing and contrasting with other scribes. Self<Handshift>self can be created using random number of rows of known writer and a known writer can be compared to another unknown writer using many randomized self<Handshift>other combinations of rows. 2-center scikit-fuzzy analyses and tests described in figure 4 can become an essential tool to detect real hand shifts and estimate the number of scribes in a document containing mixture of handwriting, especially benefiting forensic data analytics.

References:

[1] R. Austin Hicklin , Linda Eisenhart, Nicole Richetelli, Meredith D. Miller, Peter Belcastro, Ted M. Burkes, Connie L. Parks, Michael A. Smitmeanshifth, JoAnn Buscaglia, Eugene M. Peters, Rebecca Schwartz Perlman, Jocelyn V. Abonamah, and

Brian A. Eckenrod. Accuracy and reliability of forensic handwriting comparisons. *Proceedings of National Academy of Sciences (USA)* 119(32): e2119944119, 2022.

[2] Dimitris Arabadjis, Constantin Papaodysseus and Athanasios Rafail Mamatsis. Identification of the Writer of Historical Documents via Geometric Modeling of the Handwriting. In *ICSIPA* pages 180-184, IEEE 2021

[3] Ravneet Kaur. Handwriting Recognition of Gurmukhi Script: A Survey of Online and Offline Techniques. *International Journal of Computer Trends and Technology (IJCTT)*, 49(1): 2231-2803, 2017.

[4] Luiz G. Hafemann, Robert Sabourin and Luiz S. Oliveira. Characterizing and evaluating adversarial examples for Offline Handwritten Signature Verification. *Transactions on Information Forensics and Security*, 4(8): 21b-2166, IEEE 2019.

[5] R. Castrillon, A. Acien, J.R. Orozco-Arroyave, A. Morales, J.F. Vargas, R. Vera-Rodrıguez, J. Fierrez, J. Ortega-Garcia and A. Villegas. Characterization of the Handwriting Skills as a Biomarker for Parkinson's disease. IEEE 2019.

[6] Moises Diaz, Momina Moetesum, Imran Siddiqi and Gennaro Vessio. Sequence-based dynamic handwriting analysis for Parkinson's disease detection with one-dimensional convolutions and BiGRUs. *Expert Systems with Applications*. 168: https://doi.org/10.1016/j.eswa.2020.114405, 2021.

[7] Kim Stitzer. Examples of Handwriting Styles, https://www.drawyourworld.com/blog/examples-of-handwriting-styles.html.

[8] Lucy. What are the 12 characteristics of handwriting? https://www.thepencompany.com/blog/handwriting/what-are-the-12-characteristics-of-handwriting.

[9] Veronica Aubin, Marco Mora and Matilde Santos-Penas. Off-line Writer Verification based on Simple Graphemes. *Pattern Recognition*, S0031-3203(18): 30077-3, 2018.

[10] Andreas Fischer, Volkmar Frinken, and Horst Bunke. Hidden Markov Models for Off-Line Cursive Handwriting Recognition. *Handbook of Statistics*, 31: 421-442, 2013.

[11] Ameur Bensefia and Thierry Paquet. Writer verification based on a single handwriting word samples. *EURASIP Journal on Image and Video Processing*, 2016: 1-9, 2016.

[12] A. A. Brink, J. Smit, M. L. Bulacu and L. R. B. Schomaker. Writer identification using directional ink-trace width measurements. *Pattern Recognition, 45: 162-171, 2012.*

[13]   Shafqat Khan and Muhammad Rashid and Faraz Javaid. A high performance processor architecture for multimedia applications. *Comput. Electr. Eng.*, 66: 14-29, 2017.

[14]   Sheng He and Lambert Schomaker. Writer identification using curvature-free features. *Pattern Recognition,* 63: 451–464, 2017.

[15]   Wenhai Wang, Yirui Wu, Palaiahnakote Shivakumara and Tong Lu. Cloud of Line Distribution and Random Forest Based Text Detection from Natural/Video Scene Images. In book *MultiMedia Modeling* pages 48-60, 2018.

[16]   Vincent Christlein, David Bernecker, Florian Hönig, Andreas K. Maier and Elli Angelopoulou. Writer identification using gmm supervectors and exemplar-svms. *Pattern Recognition* 63: 258–267, 2017.

[17]   Tobias Grüning, Gundram Leifert, Tobias Strauß, Johannes Michael and Roger Labahn. A two-stage method for text line detection in historical documents. *International Journal on Document Analysis and Recognition (IJDAR)*, 22: 285–302, 2019.

[18]   Felipe Petroski Such, Dheeraj Peri, Frank Brockler, Hutkowski Paul and Raymond Ptucha. Fully Convolutional Networks for Handwriting Recognition. *16th International Conference on Frontiers in Handwriting Recognition (ICFHR)*, 86-91, 2018.

[19]   Bhunia Ankan Kumar, Aishik Konwer, Ayan Kumar Bhunia, Abir Bhowmick, Partha P. Roy, and Umapada Pal. Script identification in natural scene image and video frames using an attention based convolutional-LSTM network. *Pattern Recognition*. 85: 172–184, 2019.

[20]   Naz Saeeda, Saad Bin Ahmed, Riaz Ahmad, and Muhammad Imran Razzak. Zoning features and 2DLSTM for Urdu text-line recognition. *Procedia Comput. Sci.* 96: 16–22, 2016.

[21]   Sebastian Eskenazi, Petra Gomez-Kramer and Jean-Marc Ogier. A comprehensive survey of mostly textual document segmentation algorithms since 2008. *Pattern Recognition*, 64: 1– 14, 2017.

[22]   Louis Vuurpijl and Lanmbert Schomaker. Coarse writing-style clustering based on simple stroke-related features. *Progress in Handwriting Recognition*, 37–44, 1996.

[23]   Karyna Korovai and Oleksandr Marchenko. Handwriting Styles Clustering: Feature Selection and Feature Space Analysis based on Online Input. *Third International Conference on Data Stream Mining & Processing (DSMP)*, 195-200, IEEE 2020.

[24]   Thomas Savage. Department of State Papers: Los Angeles. https://archive.org/details/168036072_80_17.

[25] PDF to Image command line conversion. Pdf2image documentation at
      https://www.pdftron.com/documentation/cli/guides/pdf2image.

[26] Tesseract documentation at https://tesseract-ocr.github.io.

[27] Paul L. Rosin. Measuring corner properties. *Computer Vision and Image
      Understanding*, 73(2): 291-307, 1999.

[28] Scikit-image documentation. https://scikit-
      image.org/docs/dev/api/skimage.feature.html.

[29] Azriel Rosenfeld and John Pfaltz. Distance Functions in Digital Pictures, *Pattern
      Recognition*, 1: 33 – 61, 1968.

[30] Anil Jain. Fundamentals of Digital Image Processing. Chapter 9 Prentice-Hall,
      1989.

[31] Distance Transform. https://homepages.inf.ed.ac.uk/rbf/HIPR2/distance.htm.

[32] SciPy documentation.
      https://docs.scipy.org/doc/scipy/reference/generated/scipy.ndimage.distance_tran
      sform_edt.html.

[33] Wilhelm Burger and Mark Burge. Principles of Digital Image Processing: Core
      Algorithms. Springer-Verlag, London, 2009.

[34] B. Jähne. Digital Image Processing. Springer-Verlag, Berlin-Heidelberg, 2005.

[35] Scikit-image documentation: Measure region properties. https://scikit-
      image.org/docs/dev/auto_examples/segmentation/plot_regionprops.html#sphx-
      glr-auto-examples-segmentation-plot-regionprops-py.

[36] Tracy, John C. Plane Surveying: A Text-Book and Pocket Manual. J. Wiley &
      Sons, 1907.

[37] Davis, John C. Statistics and data analysis in geology. Wiley 1986.

[38] OpenCV documentation: What are Contours?
      https://docs.opencv.org/3.4/d4/d73/tutorial_py_contours_begin.html .

[39] Kensheng Wu, Ekow Otoo and Arie Shoshani. Optimizing connected component
      labeling algorithms. Paper LBNL-56864, 2005, Lawrence Berkeley National
      Laboratory (University of California), http://repositories.cdlib.org/lbnl/LBNL-
      56864.

[40] Tony Lindeberg. Scale Selection Properties of Generalized Scale-Space Interest
      Point Detectors. *Journal of Mathematical Imaging and Vision*, 46(2): 177-210,
      2013.

[41]   Tony Lindeberg. Image matching using generalized scale-space interest points. *Journal of Mathematical Imaging and Vision*, 52(1): 3-36, 2015.

[42]   Tony Lindeberg. Scale invariant feature transform. *Scholarpedia*, 7(5):10491, 2012.

[43]   Blob Detection: scikit-image documentation https://scikit-image.org/docs/stable/auto_examples/features_detection/plot_blob.html.

[44]   StandardScaler Documentation: https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.StandardScaler.html.

[45]   MinMaxScaler Documentation: https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.MinMaxScaler.html#sklearn.preprocessing.MinMaxScaler.

[46]   Arthur Szlam, Yuval Kluger and Mark Tygert An implementation of a randomized algorithm for principal component analysis. *ACM TOMS*, 43(3): 28:1-28:14, 2016.

[47]   Matrix decomposition documentation: https://scikit-learn.org/stable/modules/generated/sklearn.decomposition.PCA.html.

[48]   Bernhard Schölkopf, Alexander Smola, and Klaus-Robert Müller. Kernel principal component analysis. *International conference on artificial neural networks*, Springer, Berlin, Heidelberg, 1997.

[49]   Timothy J Ross. Fuzzy Logic With Engineering Applications. 3rd edition Wiley, pages 352-353, 2010.

[50]   Scikit Fuzzy c-means clustering: https://pythonhosted.org/scikit-fuzzy/auto_examples/plot_cmeans.html.

[51]   Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot and Édouard Duchesnay. *JMLR*. 12(85):2825–2830, 2011.

[52]   Unsupervised clustering using scikit-learn: https://scikit-learn.org/stable/modules/classes.html#module-sklearn.cluster.

[53]   C. Adak, B. B. Chaudhuri and M. Blumenstein. An Empirical Study on Writer Identification and Verification From Intra-Variable Individual Handwriting. *IEEE Access*, 7: 24738-24758, 2019.

[54]   Joel Odd and Emil Theologou. Utilize OCR text to extract receipt data and classify receipts with common Machine Learning algorithms. Linköping University, Department of Computer and Information Science Bachelor thesis. 2018. http://liu.diva-portal.org/smash/get/diva2:1215460/FULLTEXT01.pdf