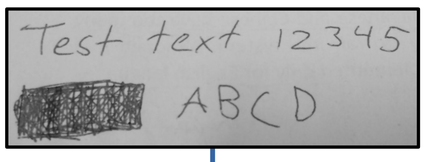
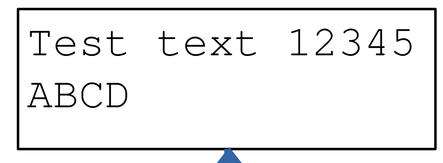
Preprocessing for Offline Handwriting Recognition

Perttu Pitkänen
Academic advisors: Masayuki Kawamata, Masahide Abe
Kawamata/Abe Laboratory
Department of Electronic Engineering



Optical Character Recognition (OCR) is the process of analyzing a picture of text and recognizing and extracting the characters to digital form. The task of handwriting recognition (HWR) concentrates on analyzing handwritten characters. Offline Handwriting recognition means that the input is static image file which is analyzed for text. Online handwriting recognition uses live writing data such as strokes to analyze the text while it is written.[1]

Handwriting recognition can be applied to many practical uses. Most importantly digitization of handwritten documents. Many well functioning OCR and HWR systems exist but the process of handwriting recognition is still undergoing development.



Typical handwriting recogntion process can be divided into three main phases [1]:

Preprocessing

At preprocessing stage the image is enhanced by applying varying filters, transforms and segmentation to it. For text recognition it is important to reduce noise from the image. After noise removal the image must be binarized without losing any of the textual information. Additionally some image skew can be detected and the image can be rotated according to the skew amount.

To distinquish text areas from other irrelevant objects in the image stroke analysis is considered. The goal is to find objects which have almost constant stroke width which usually is a characteristic feature of written text.

As a part of preprocessing it is also important to locate the area where the text is located. The goal is to identify the columns, paragraph, captions and other bodies of text. Afterwads the text must also be divided into lines and into words.

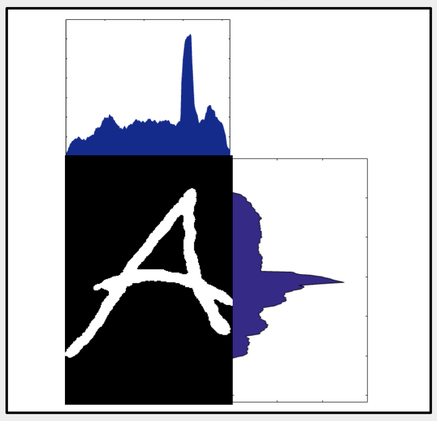
The preprocessing phase was chosen as the main focus for this research as it provides many interesting subjects of research especially in the field of image processing.

Feature Extraction

For classification stage some features must be extracted from previously found objects. Depending on preference individual characters or whole words can be used. The extracted features help to describe the object shape new inputs to different words or symbols. Some features that can be extracted include:

- Raw binary data
- Vertical and horizontal projection histograms
- Histogram of ordered gradients
- Topological features such as endpoints, loops and junctions
- Parameters of polynominals (curve fitting)

All of these methods provide measurable data of the object shape which then can be used to build feature vectors for classification phase.[2]



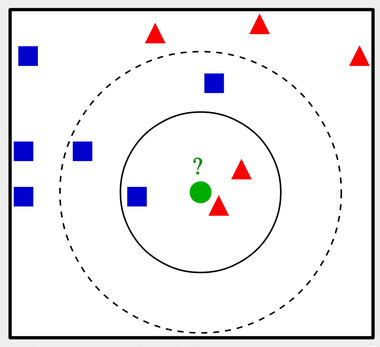
Example of feature extraction of the letter A. Vertical and horizontal projection histograms can be used as features

Classification

Lastly in the recognition process is the classification phase. Often during classification machine learning approaches are used.

Simple example for machine learning algorithm is the k-Nearest Neighbor algorithm. The algorithm searches for the closest match of test data in the feature space. The previous training data is distributed in the feature space and classified accordingly. Specified amount of the nearest neighbors of the new node are counted and compared. The class has the most representation within these points is the class of the new node. The k stands for the amount of neighboring data points that are compared to the test data, and it should be declared as an odd number to prevent a tie from happening between two classes.

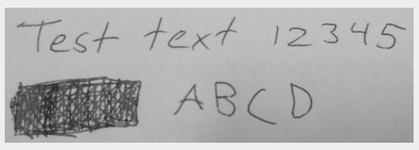
The feature space can be constructed from feature vectors aquired in the previous phase. More dimensions in feature space can result in better accuracy but also will make the execution time longer.



Vizualization on how k-nearest neighbours algorithm works. Blue squares and red triangles represent two groups of data. Green disc is the new data entry. Depending whether k value is 3 or 5 the new entry is clasified to the class which has the most representation within the group

Preprocessing Methods

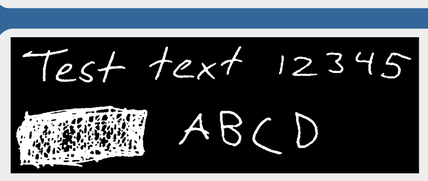
Input image



Noiseless image. Image size 2974x990.
Wiener filter size 10x10

Noise Removal

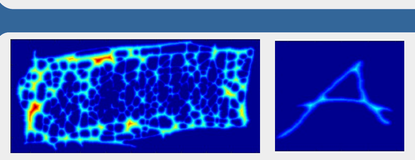
To remove noise adaptive Wiener filter is used. This filter can adapt to different amounts of noise. For areas which have more variation the filter can apply more smoothing and less smoothing to those areas without much variation. Size of the filter must be defined prior to the use. Larger filter sizes can result in excessive blur.



Binarized Image. W = 100, K = 0.1. Parameters were chosen with several tests.

Binarization

The goal of binarization phase is to acquire black and white i.e. binary image from the noiseless image. One binarization algorithm designed for this purpose is the Sauvola algorithm [3]. This algorithm was designed specifically text document binarization in mind and it is designed to take uneven lightning into account and can also differentiate between textual and non-textual regions applying different thresholding to them. The algorithm needs two parameters: The window size *W* and user defined parameter *K* “sensitivity”. Smaller *K* values cause more noise and larger values caused broken objects. Window size *W* had similar effect: larger window size caused more noise and smaller caused broken objects.

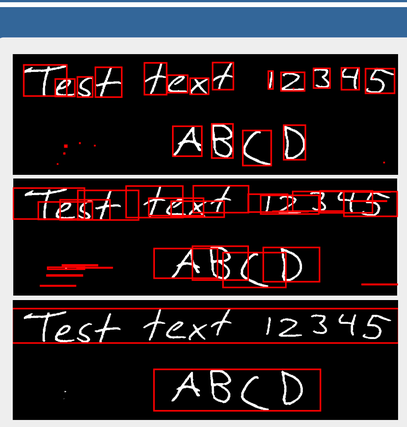


Visualization of stroke width variation. Chosen threshold 0.45. Dark blue represents thin stroke width and red represents large width.

Stroke Width Analysis

To differentiate between actual text and other irrelevant objects such as drawings, stroke width can be studied.[4] Handwritten characters usually have only little variation in stroke width but other objects can have larger variation. In these pictures the object width in each point is vizualized. For the drawing the stroke widht varies a lot compared to the written letter A which has only a little variation in the junction points.

According to the amount of variation some of the objects can be removed. This process requires a threshold value for the maximum amount of variation that is allowed. In this case the variation threshold was chosen to be 0.45 to filter the large drawing out.



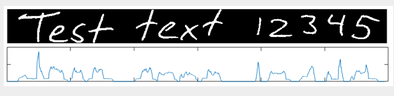
Row detection. Horizontal expansion 134 pixels.

Text Row Detection

For the next step is to try and detect the individual rows of text. At this point the developed system is only capable on handling horizontal text. To detect horizontal text rows the neihgboring objects should be found and grouped together. This can be done with following method: Firstly a bounding box is drawn around the individual objects. After that the boxes are expanded horizontally so that they overlap each other. All the overlapping boxes are then combined into bigger boxes which represent the rows of text.

The expansion amount can be adjusted depending on the text size and spacing. From the image can be seen that some noise still remains. These small objects are still expanded and can be seen as thin red boxes.

At this point some remaining small irrelevant regions should be removed. In this case the small areas can be removed by calculating the total area of all the boxes. The boxes that only take small fraction of the total area can be considered too small to contain text an can be discarded. This method is relative to the text size and only needs one argument to function which is the minimun allowed percentage.



Vertical projection histogram of first text row.

Word Separation

Next step in the process is to find individual words inside the previously found rows. One method could be to analyze the vertical projection histogram of given row. [4] The histogram gives information on the location of biggest area objects on the x-axis. The histogram has zero values for those areas without any white pixels. These areas can be considered spaces. These spaces occur also inside the words themselves if the characters doesn't overlap. To detect individual words the width of the spaces must be compared and the spaces that aren't wide enough will be ignored when dividing the row into words.

To feature extraction phase

Implementation

For quick development and tests the implementation of all these methods was done with MATLAB tools and programming language. The MATLAB image processing toolbox provided easy way to experiment with different image processing techniques.[6]

For some of the methods, such as the Sauvola algorithm, the implementation was aquired from open source repositories. MATLAB has also several inbuilt functions to help with the implementation process.

MATLAB being essentially a scripting language has its limitations in concern of execution speed. At this point of research the computation speed is not important and the main purpose is to prototype and experiment with different methods.

Evaluation

The functionality of these methods was tested with several input images with varying properties such as amount of shadows, text size, text color and character spacing. During the test the goal was to find which methods work best and with which parameters. The methods were compared by observing the quality of the output and the number of detected objects. At this point no reliable mathematical metric could be used to review the output and the output was analyzed only visually.

At this state the most important observation is that the system is highly dependant on parameters such as filter sizes or threshold values. These parameters do not work well if the text properties change significantly. During the tests the optimal threshold values were tested mainly for IAM Handwriting Database. The chosen values are:

- Wiener filter size 6x6
- Sauvola neighbourhood size *W* 100x100
- Sauvola threshold value *K* 0.6
- Stroke width threshold 0.65
- Horizontal expansion for row detection 70 px.

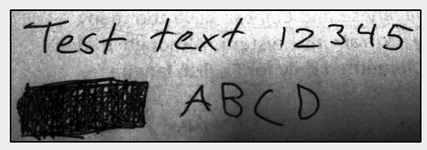
The methods are also prone to errors if the image contains severe skew or some area of the image is out of focus. For now the research will focus on high quality scans without excessive skew or blur.

For binarization other algorithms were also considered. The Sauvola algorithm was chosen as it resulted in reliable results with small effort when tested with images with noticeable shadow.

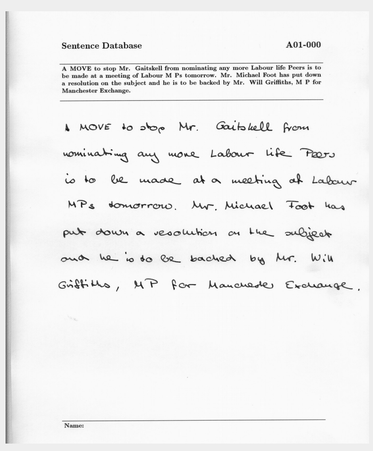
Failed preprocessing methods

During testing some methods were discarded. One method that failed to improve the preprocessing was histogram equalization which is used to enhance the image contrast. Usualy histogram equalization is used to enhance contrast and make objects more visible to human eye. However during the tests the histogram equalization made irrelevant noise such as paper texture more visible and made the overall process more difficult.

Also before the stroke analysis was implemented the irrelevant objects were tried to remove according to other properties such as their area, number of holes in the object or object eccentricity. These methods required the parameters to be picked carefully for each case compared to stroke width method which only requires one parameter.



Histogram equalization can cause more noise.



Example page from IAM Handwriting database.

Test Data

For demonstration and software development purposes small handwritten or digital images were used. (Such as used in demonstration above) These images were usually made to test some aspect of the process such as shadows in image or irrelevant objects.

For larger scale tests IAM Handwriting database will be used. The database contains 1539 pages of handwritten English text made by 657 different writers. Additionally the actual text and metadata is provided. The database entities were scanned with constant resolution 300dpi.[7]

The database entries lack any images or other figures which could make the recognition process more challenging. Some new kind of testing data might be needed afterwards.

Future work

As mentioned above the system is dependant of many parameters. The number of static constans will make automated handwriting recognition process slower and prone to errors. One of the future goals is to automate the parameter choosing process so little to none user input is required for the handwriting recognition. This can be done by running the recognition process first with initial parameter and afterwards adjusting the parameters and comparing the results.

To test all these methods in practice the feature extraction and classification phases should also be implemented for fully functional handwriting recognition system. At this point the system is only capable of the preprocessing. Some proposed methods for feature extraction would be the Histogram of Ordered Gradients and for classification the k-Nearest Neighbors algorithm.

After the whole recognition system is functional large scale tests must be conducted to evaluate the overall effectiveness of the system. The IAM Handwriting database is ideal for this purpose.

References

[1] M. Cheriet, N. Kharma, C. Liu, and C. Suen, *Character recognition systems: a guide for students and practitioners*. 2007.

[2] M. S. Dalal, “A Survey for Feature Extraction Methods in Handwritten Script Identification.”

[3] J. Sauvola and M. Pietikäinen, “Adaptive document image binarization,” *Pattern Recognit.*, vol. 33, no. 2, pp. 225–236, 2000.

[4] Y. Li and H. Lu, “Scene text detection via stroke width,” *Proc. - Int. Conf. Pattern Recognit.*, no. Icpr, pp. 681–684, 2012.

[5] N. Priyanka, S. Pal, and R. Mandal, “Line and Word Segmentation Approach for Printed Documents,” *Int. J. Comput. Appl. IJCA*, vol. RTIPPR, no. 1, pp. 30–36, 2010.

[6] I. The MathWorks, “Image Processing Toolbox.” [Online]. Available: http://se.mathworks.com/products/image/.

[7] U. V. Marti and H. Bunke, “The IAM-database: An English sentence database for offline handwriting recognition,” *Int. J. Doc. Anal. Recognit.*, vol. 5, no. 1, pp. 39–46, 2003.