Document Image Binarization

ITRI617

Prof G. Drevin

Computer Science and Information Systems

Potchefstroom Campus NWU

Johan Venter

October 13, 2022

Contents

1 Artefact

The development of this artefact was inspired by the method used for document image binarization by [SLT12] that is discussed later in this paper. This artefact takes the form of a program written in python that makes use of functionality from three main libraries namely numpy [Har+20], scikit-image [Wal+14] and scipy [Vir+20] in conjunction with custom developed methods.

A document image is provided as input where it is converted to a grayscale image for processing. It is then passed through a series of steps that each modify it in some way. The process is comprised of four main steps. The images used for testing and demonstration are open source, provided by DIBCO 2016 Handwritten Document Dataset.

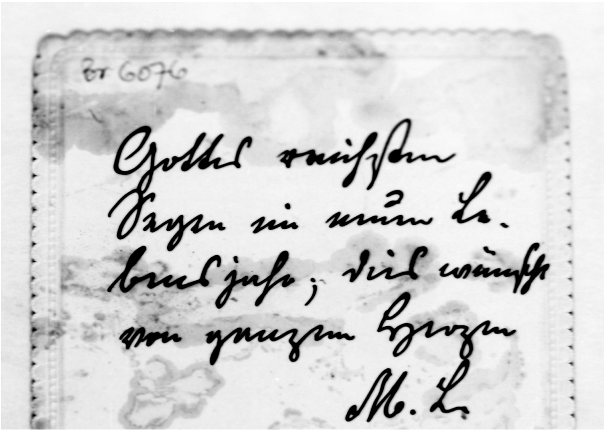

Figure 1: Original Document Image

1.1 Denoising

The first step is to denoise the input document image. Both low and high freqency noise are prevalent in most images. The low freqency noise or coarse noise is filtered by applying the wavelet denoising filter available in the scikit-image library. This filter uses the standard deviation of the intensity values of the image as an input parameter as demonstrated in Figure 2.

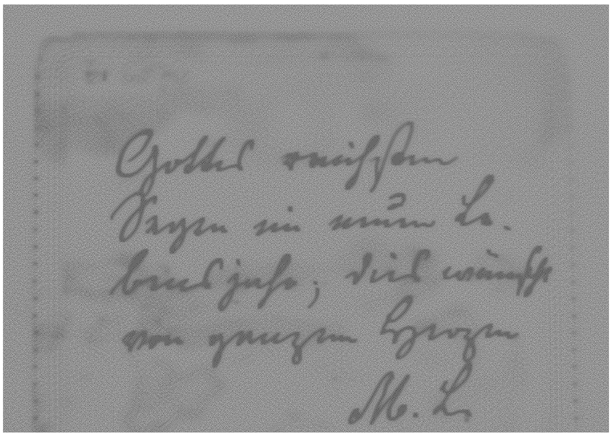

Figure 2: Noise removed by wavelet filter

This is followed by applying a custom adaptive wiener filter that uses a 3x3 gaussian kernel for low frequency gaussian noise removal (Figure 3). At this point the image can still have gray level intensity values covering a wide range of values (it is not binarized)

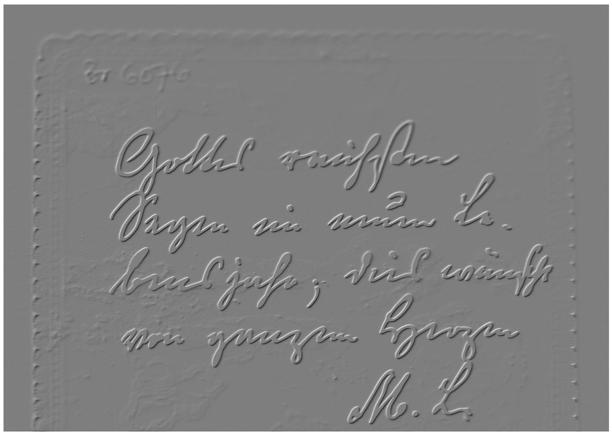

Figure 3: Noise removed by wavelet filter

1.2 Thresholding

Otsu's thresholding method is used to convert the denoised image into a bi-level image (binary image / binarized image).

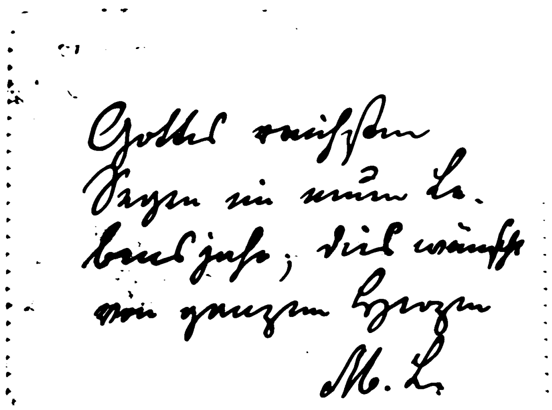

Figure 4: Otsu thresholded image

1.3 Text Stroke Width Estimate

An estimation of text stroke width will be utilised in the final step. An effective method of doing this as introduced by [mut20] consists of two main steps, namely performing a distance transform and skeletonizing the image.

1.3.1 Distance Transform

The distance transform uses the thresholded image and assigns each pixel a value according to its euclidean distance to the closest white pixel (background pixel), thereby creating an image with the brightest pixels in the center of the text.

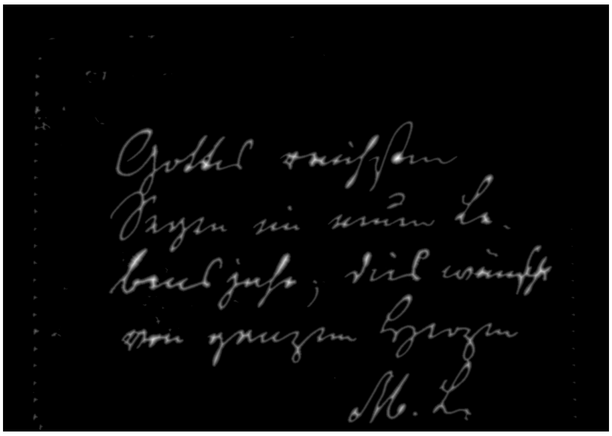

Figure 5: Distance transform

1.3.2 Skeletonizing

Skeletonizing an image consists of making multiple passes over an image and detecting the edge pixels. These edge pixels are then removed unless they break the connectivity of the identified object [Wal+14].

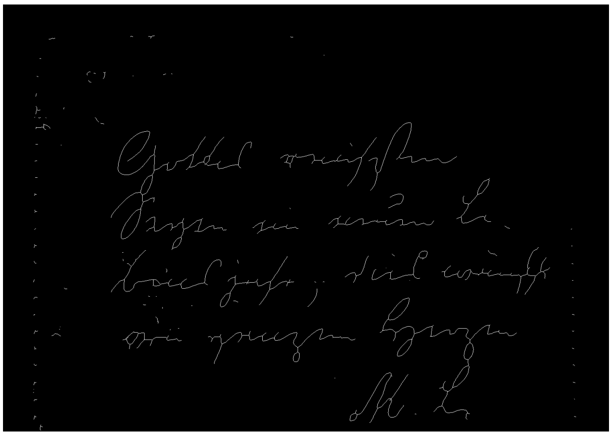

Figure 6: Skeletonizing transform

The bright pixels in the skeletonized image are then used as indexes for the selection of pixels in the distance transformed image. The selected pixel values in the distance transformed image are summed, then averaged to obtain a quantity half of the actual average text stroke width.

1.4 Median Filter

Finally, A median filter that uses the calculated stroke width as a parameter passes over the image to remove artefacts on the image smaller than the text stroke to produce the final result.

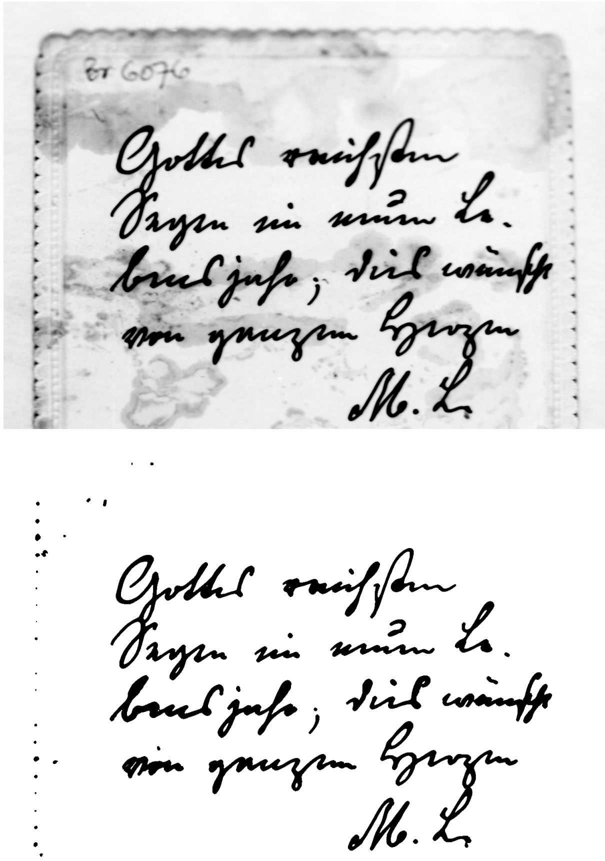

Figure 7: Original vs Final Binarized Image

2 Development Lifecycle

The general lifecycle of the development of this project involved the iterative research and testing of multiple existing strategies.

2.1 Research

The first step in the lifecycle was the research for this project. The research needed to be directed towards gaining an understanding of the problem, the related fields and key terms and jargon related to the problems encountered.

2.2 Existing Solutions

Once a firm understanding of the field was established, existing solutions were researched and evaluated. This also required an understanding of the surrounding technologies and fundamentals of the methods used.

2.3 Development

Candidate methods, ideas and technoloies were identified and appropriate open source libraries were leveraged where needed. Several methods were implemented, tested and relinquished.

2.4 Iterative Approach

Although the main lifecycle of the development of the project was sequential, each step happened iteratively.

3 Development of the Artefact

As a guide for methods relating specifically to document image binarization, two main sources were utilised [SLT12] [GPP06]. For general image processing principles and practices numpy [Har+20], scikit-image [Wal+14] and scipy [Vir+20] were invaluable.

3.1 Denoising

3.1.1 Wiener Filter

The Wiener filter is optimal by the mean squared error measure, since it is defined by it. This property along with popular use and consistency of results by [Nat13], [GPP06] makes the wiener filter as well as wavelet filters the obvious choice as candidates for denoising the document images.

[GPP06] describes the use of an adaptive wiener filter that makes use of the local properties in an image to reduce noise using the following formula:

$$I(x,y) = \mu + \frac{(\sigma^2 - v^2)(I_{source} - \mu)}{\sigma^2}$$

where μ is the local mean, σ^2 the variance in a 3 x 3 window and v^2 is the average variance.

The selected algorithm is a variation on the original Wiener filter, called the Wiener-Hunt filter that transforms the image into the freqency domain first by using the fourier transform

$$\hat{x} = F^{\dagger}(|\Lambda_H|^2 + \lambda|\Lambda_D|^2)\Lambda_H^{\dagger}Fy$$

with F and $F^{\dagger}$ the Fourier and inverse Fourier transforms respectively, Λ_H the Fourier transform of the transfer function and λ a damping constant as described by [Wal+14].

3.1.2 Wavelet Filter

The wavelet transform decomposes the image into a collection of wavelets. A wavelet is a wave-like function, that has a finite 'energy' and symmetric area around the x-axis. The transform can be interpreted as the convolution of a set of wavelets over the image that will output a signal proportional to the similarity between the wavelets and the image.

$$\int_{-\infty}^{\infty} f(x,y) \cdot g(x,y)\, dx$$

Other Denoising methods were considered such as total variation denoising and bilateral denoising.

3.2 Thresholding

References

[GPP06] Basilios Gatos, Ioannis Pratikakis, and Stavros J Perantonis. "Adaptive degraded document image binarization". In: *Pattern recognition* 39.3 (2006), pp. 317–327.

[Har+20] Charles R. Harris et al. "Array programming with NumPy". In: *Nature* 585.7825 (Sept. 2020), pp. 357–362. DOI: 10.1038/s41586-020-2649-2. URL: https://doi.org/10.1038/s41586-020-2649-2.

[mut20] muthukrishnan. *stroke width identification algorithm for text like regions*. Sept. 2020. URL: https://muthu.co/stroke-width-identification-algorithm-for-text-like-regions/.

[Nat13] Asoke Nath. "Image Denoising Algorithms: A Comparative Study of Different Filtration Approaches Used in Image Restoration". In: *2013 International Conference on Communication Systems and Network Technologies*. 2013, pp. 157–163. DOI: 10.1109/CSNT.2013.43.

[SLT12] Bolan Su, Shijian Lu, and Chew Lim Tan. "Robust document image binarization technique for degraded document images". In: *IEEE transactions on image processing* 22.4 (2012), pp. 1408–1417.

[Vir+20] Pauli Virtanen et al. "SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python". In: *Nature Methods* 17 (2020), pp. 261–272. DOI: 10.1038/s41592-019-0686-2.

[Wal+14] Stéfan van der Walt et al. "scikit-image: image processing in Python". In: *PeerJ* 2 (June 2014), e453. ISSN: 2167-8359. DOI: 10.7717/peerj.453. URL: https://doi.org/10.7717/peerj.453.