Optimization of Text Extraction on Law Enforcement Data

BSc Desmond Tjing Hien Cheung

Supervisors: Dominique Roest, Bas van Schaik, Gijs Smit

First Reader: Dr. Mark Hoogendoorn

Second Reader: Dr. Rikkert Hindriks

Amsterdam, January 2018

Vrije Universiteit Amsterdam
Faculty of Science
Business Analytics
De Boelelaan 1081a
1081HV

Eenheid Amsterdam
Dienst Regionale Recherche
Team Digitale Opsporing: TROI
Kabelweg 25
1014BA

Preface

I wrote this report to document the research that was conducted during my internship period at the Dutch Police. This internship was part of the course Master Project Business Analytics and concludes my Master's program in Business Analytics.
At TROI (Team Rendement Operationele Informatie) I researched the effectiveness of OCR (Optical Character Recognition) on visual data from confiscated electronic data carriers. Focus of the research was mainly on improving text extraction on law enforcement data. TROI is part of a TDO (Team Digitale Opsporing) which is a team that offers specialized knowledge and uncommon techniques within the Police organisation. More specifically, TROI aims to efficiently increase the quantity of variable and relevant information obtained from large data sources.

Firstly, I would like to thank Dominique for giving me this internship opportunity and showing me a small but great part of the Dutch law enforcement. Secondly, I would like to thank Bas, Gijs and all of TROI for their warm welcoming to the team and sublime guidance with their expertise throughout the entirety of my internship. Finally, I would like to thank Mark and Rikkert for their time, effort and supervision they have given me during my internship.

Desmond Cheung
Amsterdam, July 2018

Summary

Data is essential for a wide range of industrial branches. This can be observed in the private sector such as Facebook and Google, but also in the public sector such as law enforcement. The Dutch police is constantly innovating, trying to keep up with the exponential growth of data which makes it difficult to efficiently extract relevant information from case evidence. This efficiency can be achieved in computer vision technology, where we bestow human tasks upon machines. The task of reading is performed by a computer through optical character recognition. The quality of text recognition is mostly determined by image quality. Noise such as uneven lighting, skewness and complex backgrounds affect text recognition negatively. Visual data encountered by law enforcement is not only highly variable in relevance, but also in image quality. Our study attempts to improve text extraction from visual law enforcement data using various text recognition methodology. We studied conventional methodology based on morphological image properties as well as deep learning methodology, such as text detection with a fully convolutional neural network. We implemented three previous works of which the first uses various clustering algorithms to extract text, the second uses colour information to extract text and the third detects text using a fully convolutional network. We recognize text using open source OCR engine Tesseract. We optimized the parameters of the implemented pipelines and evaluated their performances in terms of text extraction using different performance measures. We showed that law enforcement data varies greatly in quality and thus requires a robust method of text extraction. We observe that this robustness is found in deep learning methodology, which generally outperforms conventional methodology. However, if the application of text recognition is on specific data such as chat messages or driver's licenses, then it's possible to tailor conventional methodology to achieve similar accuracy. We also show that skew detection need to be addressed to improve text recognition quality. This thesis gives new insights for improvement in law enforcement and is the first to compare conventional and deep learning methodology in text extraction.

Contents

1 Introduction IV

2 Literature review VII

3 Methods XI

- 3.1 Computer vision fundamentals . . . XII
- 3.2 Image segmentation . . . XII
  - 3.2.1 Thresholding . . . XIII
  - 3.2.2 Canny edge detection . . . XIV
- 3.3 Dilation . . . XVI
- 3.4 Connected Component Labeling . . . XVII
- 3.5 Text Line Extraction . . . XVIII
  - 3.5.1 Density Based Spatial Clustering of Applications with Noise . . . XVIII
  - 3.5.2 MST Angle based Clustering . . . XVIII
- 3.6 Text Detection with a Full Convolutional Neural Network . . . XIX
- 3.7 Text Recognition . . . XX
- 3.8 Skew Detection . . . XXI

4 Experimental Setup XXII

- 4.1 Description of the Data . . . XXII
- 4.2 Parameter Optimization . . . XXIII
- 4.3 Evaluation . . . XXIV

5 Results XXV

- 5.1 Pipeline 1 . . . XXV
- 5.2 Pipeline 2 . . . XXVI
- 5.3 Pipeline 3 . . . XXVII
- 5.4 Optimal pipelines evaluation . . . XXVIII

6 Conclusions and recommendations XXX

- 6.1 Conclusions . . . XXX
- 6.2 Recommendations . . . XXXI

Appendices XXXIV

Appendix A XXXV

Appendix B XXXVI

Chapter 1

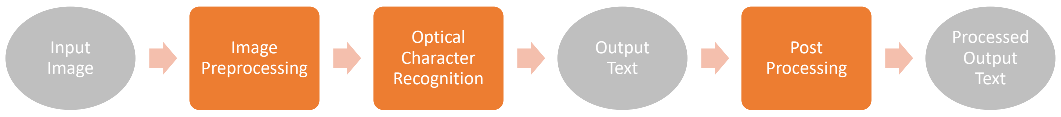
Introduction

Over the past decade there has been a dramatic increase in data. This increase is also noticeable in law enforcement. The Dutch police is collecting large quantities of data from confiscated electronic data carriers such as mobile phones, tablets and laptops. A laptop currently has an average hard disk capacity of 2 terabytes (Roest et al., 2014), which translates to storage for up to 620.000 images. Naturally, not all data is equally relevant in a police investigation and it is highly undesirable to extract relevant information from this data through manual investigation. A solution to this problem can be found in the field of computer vision. Computer vision is a component of AI which, just like many other fields in AI, aims to let computers perform human tasks (Robertson, 2013). One of these tasks is the visual understanding of text from visual data. Optical character recognition is the technology which translates text from visual data into machine-encoded text by means of pattern recognition. This allows for business process optimization of a detective investigating visual evidence, through automated text extraction. Other similar technologies which contribute to optimize efficiency of Dutch police investigations are object recognition, facial recognition and predictive analytics.

However, the digital visual evidence is not only notably variable in relevance but also in image quality. This variability in image quality makes pattern recognition increasingly difficult. Therefore, it will be harder to extract text in, for example, a photographed letter in bad lighting than a machine born PDF file of that same letter. Factors found to be influencing text extraction have been explored in several studies Farahmand et al., 2013. Examples of these factors are noisy backgrounds, skewness of text and image resolution.

Text recognition has been studied by many researchers using conventional as well as newer deep learning methodology. The studies using conventional methods propose different preprocessing techniques for various types of noise. Additionally, these studies research text extraction from specific types of documents, which lead to unambiguous assumptions to be made within their domain. Furthermore, most of these studies use born

Figure 1.1: General pipeline of end-to-end text recognition process



digital images as research data and thus contain relatively little noise. The studies using deep learning techniques are more robust and propose text detection with full convolutional neural networks from scene text images or text in the wild images. However, few writers have been able to draw on any systematic research into optimization of automated text extraction in a business process as described in our study. This paper attempts to answer the main question:

How can we improve the quality of text extraction on law enforcement data?

In order to answer this question, we try to answer the following subquestions:

- Is there some form of homogeneity in law enforcement data?
- Is the use of conventional methodology in text extraction viable in comparison to deep learning methodology?
- Does skew detection improve the accuracy of text recognition?

The main aim of this study is to optimize text extraction from digital visual evidence confiscated by the Dutch police by systematically evaluating existing conventional and deep learning methods for text extraction on real case data. The research data in this thesis are twofold: the visual data containing text and the ground truth of this visual data containing the textual elements of the images in the form of machine-encoded text. The visual data is drawn from three main sources: photographs from a PGP encrypted cellphone displaying various chat and email messages, a synthesized set using a driver's license and an open source text recognition dataset. The ground truths are manually transcribed texts from the these datasets, which are required to quantify our results.

A systematic evaluation to measure the various image processing techniques is conducted by creating several pipelines consisting of a number of consecutive phases, where each phase represents a processing technique which copes with one of the previously mentioned troublesome visual properties. After preprocessing, text recognition is performed using the open-source OCR engine Tesseract. This results in a number of output texts per image equal to the number of pipelines. These output texts are compared to the ground truth and the performance of the pipelines are determined with this comparison. Besides quantitative evaluation of these pipelines, qualitative methods were also used in this investigation.

Figure 1.1 shows that the output text after text recognition goes through post processing. Post processing of the raw text output of OCR on law enforcement data is not in the scope of this study. Furthermore, the thesis does not engage with text recognition of non-Latin based languages or handwritten texts. This is due to the fact that the ground truths are manually produced and thus require a native speaker of non-Latin based languages.

TROI is a small but significant team in the entire organization of the Dutch police (see appendix A), which aims to efficiently extract relevant information from large quantities of data. The findings of this study make an important contribution to optimize police investigation process. Furthermore, the importance and originality of this study is that it compares conventional methodology with newer deep learning methodology in the field of computer vision. Previous studies have made comparisons between conventional machine learning and deep learning in classification tasks ("Comparing Deep Learning and Conventional Machine Learning for Authorship Attribution and Text Generation"), and reviewed deep learning methodology for feature extraction in text mining (Liang et al., 2017), but no similar studies have been done for text extraction.

The overall structure of the study takes the form of six chapters, including this introduction. The second chapter gives a summary of the literature used in this research. Mainly three papers have been reviewed for the construction of the pipelines for text extraction. Furthermore, we reviewed literature on various alternatives for the individual steps of the pipelines. The third chapter gives an overview of the pipelines we used in this study and elaborate on the individual steps of the pipelines including the rationale behind each of them. The fourth chapter describes the experimental setup of this study and gives a description of the data. The fifth chapter shows the results of this study and briefly elaborates on these findings. The sixth chapter concludes and discusses the findings of this study.

Chapter 2

Literature review

Pattern recognition has made it possible for machines to perform human tasks. In case of optical character recognition, this is the task of reading (Eikvil, 1993). This is not reading of the same machine-encoded text which one types on a computer, but scanned documents, photographs of texts or just plain *non-editable* PDFs. Previous studies on text extraction system often have a similar pipeline as shown in figure 1.1, where the different steps in the pipelines can vary (Chaudhuri et al., 2017). One of the first steps of the pipeline is segmentation or preprocessing. Chaudhuri et al., 2017 and Eikvil, 1993 describe segmentation as the process of distinguishing objects of interest from other objects in the input image and preprocessing as noise removal. We mainly focus on these steps in this study using law enforcement data. Following steps in the pipeline are classification of individual characters and post processing the recognized text. Classification of characters is done using methods of pattern recognition (Hamad et al., 2016). Many techniques are studied for the task of optical character recognition, such as statistical tehcniques, support vector machine algorithms, neural networks and many more (Hamad et al., 2016). The simplest form of a post processing technique is the usage of a dictionary and correcting accordingly. However, there exists a probability of losing critical information with these techniques and thus do not include this step in our study. There are many fields in which optical character recognition is applied, namely: receipt imaging, legal industry, banking, health care and thus also law enforcement /(Hamad et al., 2016).

A considerable amount of literature has been published on OCR and more specifically, image preprocessing to improve OCR accuracy. Farahmand et al., 2013 identified various document image noises and proposes different methods to remove these noises. Examples of these noises are ruled line noise, marginal noise, clutter noise, stroke like pattern noise, background noise and salt and pepper noise. All these noises negatively affect accuracy of text recognition engines, because they make it difficult for OCR engines to distinguish relevant components from irrelevant components. Additionally, in this research the law enforcement data is highly variable, therefore different images contain different noise and thus require different preprocessing prior to OCR. In previous studies on automated text extraction, different conventional methods based on image morphology were used to extract text from images (Böschen et al., 2017)(Nagabhushan, 2010). More recent studies use deep learning methodology for text detection (Zhou et al., 2017). Our study aims to provide a comparative evaluation through a pipeline structure with the methodology and results from the previously mentioned studies on real life Dutch law enforcement data.

Böschen et al., 2017 compared various pipeline configurations for text extraction from scholarly figures. In order to measure the impact of individual steps of the pipeline, different methods were used in each step of the pipeline. Then each possible configuration of the pipeline was evaluated on text location detection and text recognition accuracy. The next paragraph summarizes each step of the best performing pipeline of Böschen et al., 2017.

In the first step of the pipeline by Böschen et al., 2017 a colour quantization is performed on the input image by first converting the image into a grayscale image. In the second step of the pipeline this grayscale image is converted to a binary image. Binarization is a frequently used technique in computer vision to extract the areas of interest by means of thresholding the pixels of an image (Chaki et al., 2014). Böschen et al., 2017 uses adaptive Otsu's method for binarization. Otsu's method determines the threshold for binarization with variances between and in grayscale levels of pixels. In case of Böschen et al., 2017 an adaptive version of Otsu's method was used, meaning that multiple thresholds were determined locally at different segments of an image rather than one threshold for the entire image. Additionally, several local adaptive thresholding techniques exist such as Niblack's method or Sauvola's method, of which the latter is an improvement of the former (Chaki et al., 2014). In the third step, Böschen et al., 2017 labels each segment of pixels by means of connected component labelling (CCL). CCL is an algorithm which gives the same label to pixels who are interconnected. This results in connected components which consist of text as well as non-text elements and their morphological properties (e.g., centroid coordinates, width and height of the components' bounding box, area of the component). In the fourth step of the pipeline, these morphological properties are used to cluster similar connected components with the density-based spatial clustering of applications with noise algorithm (DBSCAN). Böschen et al., 2017 uses this algorithm with the rationale that components with textual information share the same properties and that various components containing noise are suppressed by the algorithm. In the fifth step, Böschen et al., 2017 uses clustering to find individual text lines in the clusters produced by DBSCAN. The clustering algorithm used in this step is the minimum spanning tree angle-based clustering algorithm (Böschen et al., 2015). This algorithm uses components' centre of mass coordinates to determine the minimum spanning tree in the image. Then the histogram over the angles in the MST is calculated and edges with different orientation from the main orientation are removed. Böschen et al., 2017 makes two assumptions with the usage of the MST angle based clustering algorithm, namely that text components are most likely close together and that these components have similar orientation. Unfortunately, it is not possible to use these orientations for skew detection. In step 6 of the pipeline, Böschen et al., 2017 uses single string orientation detection (Chiang et al., 2015) as a method for skew detection. However, this method was not adopted in study. In the final step of the pipeline, Böschen et al., 2017 suggests two OCR engines; Tesseract and Ocropy. Even though slightly better results are obtained with Ocropy, this study uses Tesseract for text recognition. This is because Tesseract is open source and allows room for future research in training the underlying models. Additionally, Böschen et al., 2017 concludes no significant improvement in text recognition accuracy with various heuristic approaches for post processing. Even though Böschen et al., 2017 claims not to make any assumptions about the figures used in their study, they do make assumptions about the images. Namely, the images are born digital and thus relatively contain little to no noise in to data in our study.

Nagabhushan, 2010 proposes a multi-staged approach for text extraction from images with noisy backgrounds. The first stage of the approach detects potential text regions by means of edge detection. Canny edge detection is a widely used algorithm which can detect edges in noisy images while suppressing noise (Nagabhushan, 2010). Subsequently, Nagabhushan, 2010 labels the found regions with connected component labeling. These labeled components consist of text regions and non-text regions. These regions are distinguished by defining text regions as components with holes in them and non-text regions as components without holes. Therefore, these components without holes are removed.

In the second stage of the proposed approach, regions that are falsely assumed to be text regions from the previous stage are removed. Nagabhushan, 2010 suggests to examine the standard deviation of gray scale values in the connected component. This standard deviation is expected to be low in non-text regions (only background pixels) and high in text-regions (back- and foreground pixels). The third stage of the proposed approach extraction of foreground text is done by means of thresholding each remaining labeled text region. This threshold is determined per connected component based on the mean and standard deviation in gray scale levels of that region. As a result pixels containing text are converted to black (gray scale value 0) and others are white (gray scale value 255). However, some black pixels might comprise of false foreground pixels. The fourth stage detects these noisy regions by determining regions' ratio of black pixel density to region area. This ratio is compared to a threshold based on the average density of black pixels throughout the binary image and through this comparison it is determined if the region is considered noisy. The final stage reprocesses the noisy regions by redoing stage 1 (text region detection) and stage 3 (foreground text extraction) on these noisy segments (and thus creating multiple smaller segments). In their comprehensive study of text extraction in noisy images, Nagabhushan, 2010 uses data which consists of scans of various types of documents. However, the usage of a scanning machine for the data results in little negative impact on the image quality compared to the usage of a cellphone camera. Therefore, this study attempts to research the usage of their methodology on law enforcement data where image quality often is impacted.

Wu et al., 2008 suggests an algorithm for text line extraction based on morphology. Firstly, the proposed system extracts features of text lines. Therefore, the assumptions are made that text often has high contrast to background and that they generally have uniform horizontal orientation. Secondly, from the resulting regions after feature extraction Wu et al., 2008 selects potential text lines based on a set of rules. This rule-set consists of requirements the regions' morphology has to satisfy for it to be considered a potential text line. Finally, the proposed algorithm verifies the potential text lines more elaborately with criteria based on character sizes and the ratio of character width to height. Wu et al., 2008 tested the proposed algorithm with a dataset of 100 images with various backgrounds, lightings, fonts and text sizes. However, this dataset is similar to various text-in-the-wild image datasets which are open source available with many text extraction competitions. Therefore, the data used in the study by Wu et al., 2008 is not representative to law enforcement data.

Zhou et al., 2017 proposes an approach of text detection by means of a full convolutional neural network. With deep learning based methodology, such as a full convolutional neural network, these features for text extraction are learned by the model from the training data and thus does not make any assumptions about the data like with the conventional methods proposed in the previously mentioned studies. In the case of law enforcement data where the visual data is highly variable, it is intuitively desirable not to make any assumptions about the data. The single neural network used by Zhou et al., 2017 produces multiple geometries around predicted text locations and gives a confidence score for the text enclosing geometries. By means of thresholding and non-maximal suppression, results in a bounding box per predicted text instance in an image.

Another type of noise which can negatively impact the accuracy of optical character recognition is skewness. Therefore, the detection and correction of skewness in document are important issues which need to be addressed in the task of text extraction. Amin et al., 2000 proposes a method of skew detection by analyzing entire blocks of text in a document. The skew is determined for each text block with the Hough transform. Then the skew for the whole document is the most frequent detected angle among the skews of all text blocks. However, the Hough transform is computationally expensive. Therefore, there are studies which propose modified versions of the Hough transform (Kumar et al., 2012). Also most skew detection algorithms require multiple lines and words to use their spacing for calculation of the document's skew. Chiang et al., 2015 proposes the single-string orientation detection algorithm (SSOD), which uses various morphological operators and connected component analysis to determine the skew of individual strings.

For our study we used Tesseract for text recognition, which is an open-source OCR engine which recognizes text through a multi-step pipeline (Smith, 2007). Similarly to the pipelines of Böschen et al., 2017 and Nagabhushan, 2010, connected component analysis is performed on a binary image. Older versions of Tesseract nest the connected components into blobs, which are subsequently organized in text lines and their word spacing are analyzed. However, the latest version of Tesseract uses long-short term memory neural networks for finding text lines, which is a neural network often used deep learning applications which involve text such as: speech-to-text, text-to-image and image-to-text. The found text lines are split up into words and the Tesseract attempts to recognize each word with a classifier. This classifier uses pattern recognition to distinguish and classify the characters correctly. Modernization attempts were made to recognize words with the use of a convolutional neural network, but the results were disappointing (Smith, 2007).

Post processing algorithms such as Special Character Filtering per String proposed by Sas et al., 2013 are not included in this study. This algorithm deletes an entire text line if a repetition is found of special characters in that text line. In the case of law enforcement data, it is possible to encounter data which contains many special characters which contain valuable information. Examples are wallet addresses of cryptocurrency or various code language used on dark web forums. Another reason to not include post processing in this study, is the amount of slang contained by law enforcement data.

From the previously named literature, our study incorporates three works and systematically evaluates these works with law enforcement data. The three studies are that of Böschen et al., 2017, Nagabhushan, 2010 and Zhou et al., 2017. Böschen et al., 2017 researches text extraction from (born-digital) academic figures using various clustering algorithms. Nagabhushan, 2010 proposes a pipeline which extracts text from (born-digital) images with noisy backgrounds using colour information. Zhou et al., 2017 uses a fully convolutional neural network to predict the location and orientation of text regions in an image. In Chapter 3 we elaborate on the used methodology of these studies. Additionally, we propose a robust method for skew detection based on the previously described studies.

Chapter 3

Methods

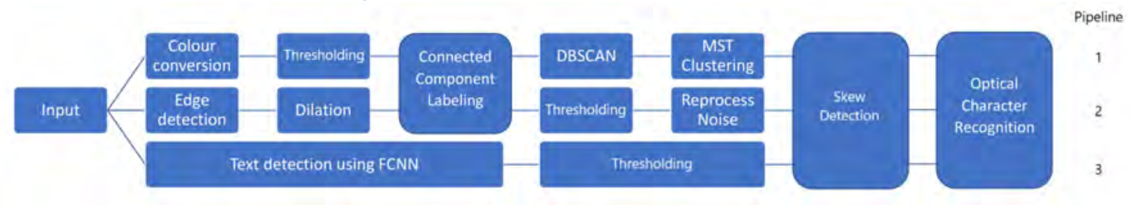
Three studies are used to identify optimizations of OCR on law enforcement data. These studies each have their own research scope and methodology which are elaborated on in Chapter 2. Our study uses quantitative analysis to optimize the performance of these studies' methodology on law enforcement data. The design of this analysis takes form of three general pipelines (see figure 3.1), each representing one of the studies described in Chapter 2. In this chapter we further elaborate on the techniques used in each pipeline. Then we give an overview of the data and discuss how this data was obtained. Finally, we go over the evaluation method of the pipelines with different configurations.

The first pipeline of figure 3.1 converts a color image to a gray scale image. This grayscale image is then converted to black and white using binarization/thresholding. Of this binary image the connected components are found by heuristically checking pixel connectivity. We calculated several morphological properties of these connected components for finding potential text lines. These candidate text lines are found by using the Density Based Spatial Clustering for Applications with Noise algorithm (DBSCAN). This algorithm clusters components who lie closely together in the image and which satisfy the minimum number of points per cluster. Components which do not meet this latter requirement are considered noise. Subsequently, we extract text lines from the candidate text lines with the Minimum Spanning Tree angle based clustering algorithm. This algorithm finds text lines based on the orientation of the components.

The second pipeline attempts to find the edges in the input image. These edges are dilated in vertical and horizontal direction. This results in a binary image with dilated white edges. These edges are labeled by means of connected component labeling. Subsequently, features are extracted and candidate text regions are produced. Based on variance in gray scale intensities, text regions are selected. These regions are thresholded to according to mean and standard deviation in gray scale values of that region. Additionally, the regions which are considered noisy are reprocessed (see Chapter 2).

The third pipeline detects text using a full convolutional neural network. This neural network predicts location and orientation of the input's text regions. These text regions are enclosed by a bounding box. Each of these predicted bounding boxes are thresholded

Figure 3.1: Text extraction pipelines

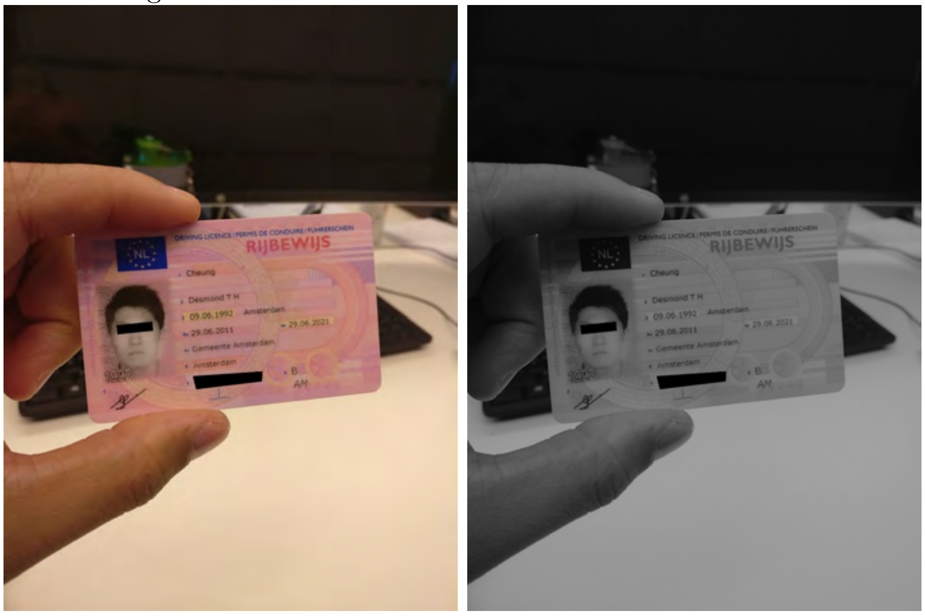


using Otsu's method.

3.1 Computer vision fundamentals

Computer vision starts with an input image I which consists of the most basic building blocks: pixels. We define $I(x, y)$ as the pixel's colour value of image I at position (x, y). For example, an image with a resolution of 500×500 contains 250,000 pixels. Each of these pixels are either represented in grayscale or in colour (see figure 3.2). A grayscale pixel only has one colour channel which can have a value between 0 and 255, where values near 0 are darker than values closer to 255. A colour pixel has three colour channels; one for red, one for green and one for blue. Each of these channels can also have a value between 0 and 255. A red colour pixel has value 255 in the red channel and 0 in the other two channels, a black colour pixel has value 0 in all three channels and a white colour pixel has value 255 in all three channels. We have just described the RGB colour space. Other colour spaces (such as sRGB, HSV, CMYK etc.) are not relevant for this study.

Figure 3.2: The original input image for the pipelines on the left and the gray scale conversion on the right.



Furthermore, resolution is of importance in computer vision, because the number of pixels of I is the amount of information the model receives as input. For text recognition, a minimum text size is required for reasonable accuracy. A lower case character x with a height below 10 pixels results in a very low recognition probability by the Tesseract engine (Smith, 2007). However, if the resolution is too large, more pixels need to be processed. This leads to longer computational times. For these two reasons, we scaled images in our study to a width of 1000 pixels with original aspect ratio.

3.2 Image segmentation

Image segmentation is the process of clustering pixels based on properties of the image of interest. Thus these clusters share the same image properties and are useful with tasks like object recognition and optical character recognition. Two well known image segmentation methods are explained in the subsections below, *Thresholding* and *Canny Edge detection.*

3.2.1 Thresholding

Optical character recognition by Tesseract is performed on binary images. Binarization is one of several ways of segmenting an image. This method segments an image by separating pixels into two classes: one class consisting of pixels representing the object and the other of background pixels. Each of these classes is assigned one of two gray values; often the foreground pixels have gray value 255 (white) and the background pixels have gray value 0 (black) (see figure 3.3). The assignment of one of the two gray values is determined by *thresholding* the pixel on interest based on its grayscale level. A binary image $B(x, y)$ of a grayscale image $I(x, y)$ with some threshold T can be defined as follows:

$$B(x,y) = \begin{cases} 255 & \text{if } I(x,y) \geq T \\ 0 & \text{otherwise} \end{cases}$$

How threshold T is calculated, determines the type of thresholding. Generally, thresholding can be divided into two types: global and local thresholding. Global thresholding applies one threshold to the entire image, while local thresholding calculates multiple thresholds for various parts of the image.

A well known global thresholding method is Otsu's method, which clusters pixels based on the variance in grayscale values of the image. Otsu's method assumes the image consists of two types of pixels foreground and background pixels. With P the image histogram, the type probabilities of various grayscale level pixels are estimated as follows:

$$q_1(t) = \sum_{j=0}^{t} P(j) \quad \text{and} \quad q_2(t) = \sum_{j=t+1}^{255} P(j)$$

Each pixel type's mean is given by:

$$\mu_1(t) = \sum_{j=0}^{t} \frac{j * P(j)}{q_1(t)} \quad \text{and} \quad \mu_2(t) = \sum_{j=t+1}^{255} \frac{j * P(j)}{q_2(t)}$$

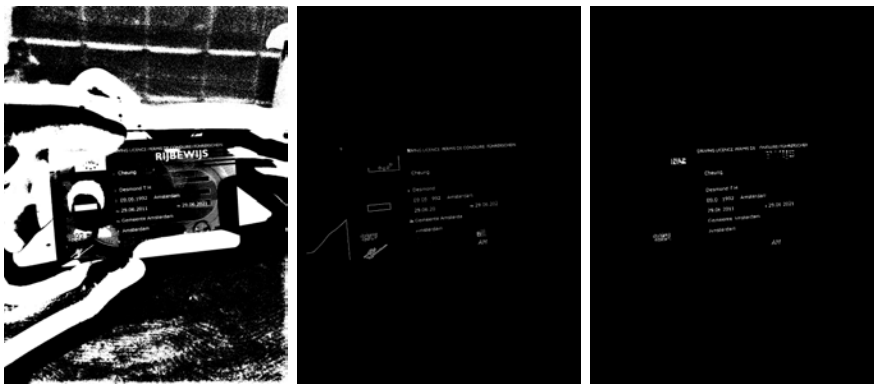
Then the optimal threshold t^* is determined by maximizing the variance in grayscale values between the two types of pixels (σ_b^2) (or minimizing the variance in gray scale values within the same pixel types (σ_w^2)).

$$\begin{aligned} \sigma^2 &= \sigma_b^2 + \sigma_w^2 \\ \sigma_w^2 &= q_1(t)\sigma_1^2(t) + q_2(t)\sigma_2^2(t) \\ \sigma_b^2 &= q_1(t)[1 - q_1(t)][\mu_1(t) - \mu_2(t)]^2 \end{aligned}$$

There also is also a local variant of Otsu's method. In the first pipeline of our study, we use local Otsu's method where the same previously mentioned variances are calculated locally by means of a kernel. This kernel is a rectangular window with dimension $b \times b$ which slides over the image and determines the threshold for the pixels inside the kernel. This adaptive/local thresholding was used in our study due to its better performance with documents under different lighting conditions, light texture, blurriness and low resolutions (see 2).

Where $m(B)$ is the mean in gray scale values of kernel B, $\delta(B)$ the standard deviation in gray scale levels of B, k a control parameter of value $[0.2, 0.5]$ and R the maximum standard

Figure 3.3: Binary images of the first, second and third pipeline from left to right respectively.



deviation in gray scale values. We evaluate the performance of both local thresholding methods.
In the second pipeline, we determine the threshold using the method of Nagabhushan, 2010 :

$$T(L) = m(L) - k * \delta(L)$$

Where L is connected component with label L. We determine the value of k with the average gray scale values of the foreground and background pixels in component L, $V_f(L)$ and $V_b(L)$ respectively. Then the value of k is determined by the author (Nagabhushan, 2010) as follows:

$$k = \begin{cases} 0.05 & \text{if } V_f(L) > V_b(L) \\ 0.4 & \text{otherwise} \end{cases}$$

In the third pipeline of the study, we evaluate Otsu's binarization method per predicted text region. We compare this variant of local Otsu's method with Sauvola's method. Sauvola determines the threshold of the kernel B as follows:

$$T(B) = m(B) * \left[1 + k\left(\frac{\delta(B)}{R} - 1\right)\right]$$

3.2.2 Canny edge detection

Another way of image segmentation is edge detection. One widely used edge detection algorithm is that of Canny. Canny edge detection is a multi-step algorithm which detects edges in an image while suppressing noise (see figure 3.4).

The first step of the algorithm is *smoothing* to remove noise. Smoothing in the Canny edge detection algorithm is done by applying Gaussian filter on the image. This filter is obtained by convolving each pixel of the image with a Gaussian kernel. The coefficients of this kernel are calculated using a two-dimensional Gaussian function:

$$G(x, y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}$$

The second step of Canny's edge detection *determines gradients* in the smoothed image. These gradients are of larger magnitude where transitions of the image are more obvious (i.e. the edges of an object in the image). The gradients are obtained by applying the

Sobel-operator which convolves each pixel of the smoothed image with the following kernels (one for each direction):

$$K_{G_x} = \begin{bmatrix} +1 & 0 & -1 \\ +2 & 0 & -2 \\ +1 & 0 & -1 \end{bmatrix} \quad K_{G_y} = \begin{bmatrix} +1 & +2 & +1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$$

The magnitude of the gradients are determined by applying some distance measure, such as Euclidean (eq. 3.1) or Manhattan (eq. 3.2), to the smoothed image's gradients of both directions:

$$|G| = \sqrt{G_x^2 + G_y^2} \tag{3.1}$$

$$|G| = |G_x| + |G_y| \tag{3.2}$$

The edges in an image are not specifically located by just the gradient magnitudes. Therefore, the direction of the edges are determined as follows:

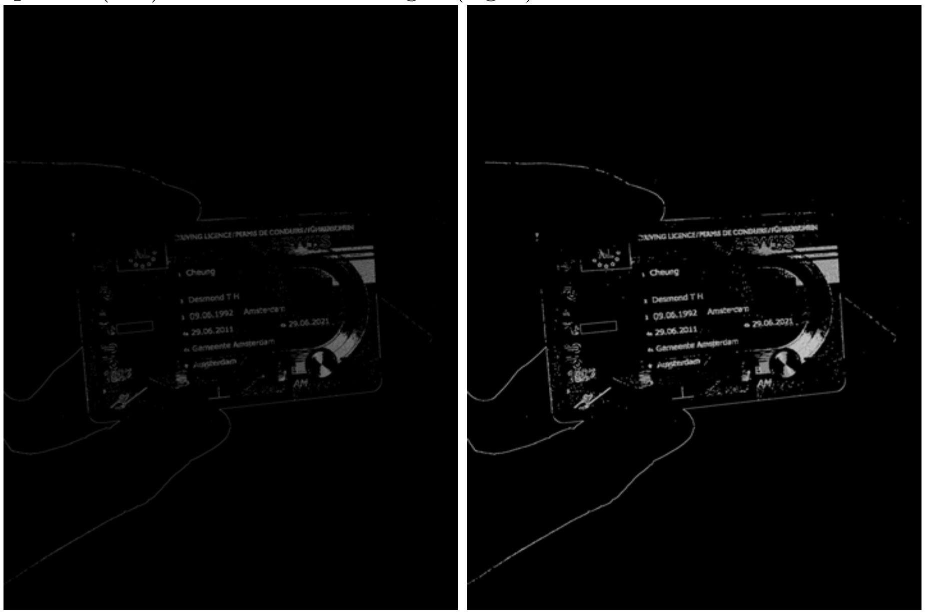
$$\theta = \arctan\left(\frac{|G_y|}{|G_x|}\right)$$

The third step of the algorithm *sharpens* the blurry edges found in the previous step. This is done by keeping the local maxima and discarding the rest in the image. Each pixel's magnitude is compared with (depending on their gradient's direction) the magnitude of neighboring pixels in an 8-connected neighborhood (see 3.4). Based on this comparison, the pixel is either kept or discarded. The resulting image contains pixels with their corresponding magnitudes.

The fourth step of the edge detection these pixels are suppressed by using double thresholding. Noise suppression is done as follows: pixels with magnitudes smaller than T_{low} are suppressed, pixels with magnitudes larger than T_{high} are converted white and pixels with magnitudes in between the two thresholds are made gray. We have found no literature on methods to determine the optimal values of the thresholds. Therefore in this study, the high threshold, T_{high}, is determined with Otsu's method. The low threshold,T_{low}, is determined by simply multiplying T_{high} with a ratio of $\frac{1}{3}$. Additionally, we experimented with two different deterministic values for T_{low} (85 and 127) while keeping T_{high} constant at 255.

The last step of Canny's edge detection discards the *weak* pixels (pixels between T_{low} and T_{high}) if they are not connected to a *strong* pixel (larger than T_{high}). This can be done by means of BLOB analysis with 8-connected neighborhood (see 3.4).

Figure 3.4: The binary image resulting from Canny's edge detection algorithm from the second pipeline (left) and the dilated edges (right).



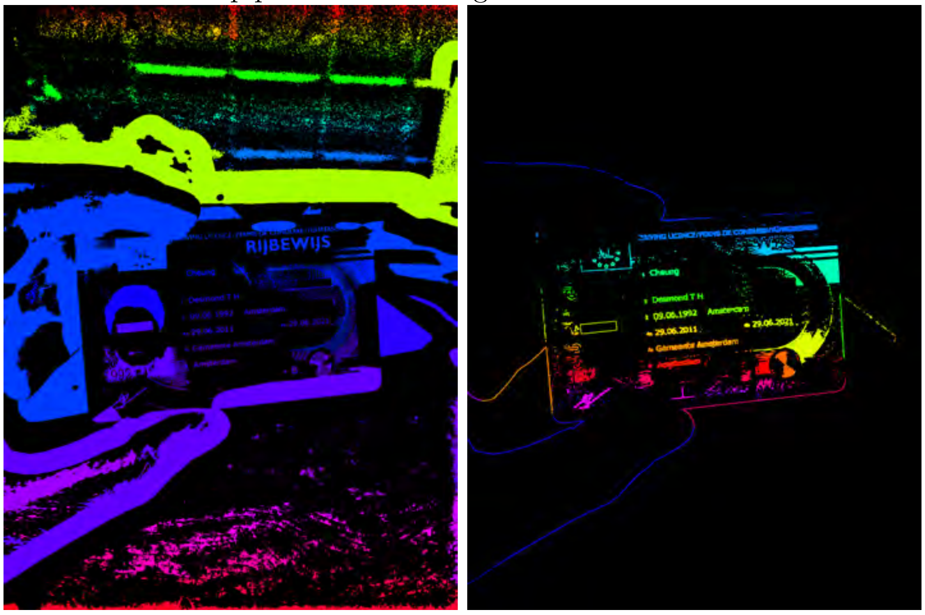
3.3 Dilation

After segmentation, we use *dilation* on the binary image in the second pipeline. Dilation is a mathematical operation which expands the foreground pixels in a binary image by means of a kernel. As a result, holes in dilated components become smaller. We remove components without holes after dilation in the second pipeline. This is due to the assumption that components with holes contain text, while components without holes are considered noise.
Mathematically we can define dilation as follows:

- Let X be the set of coordinates representing the binary image and K the set of coordinates of the kernel
- Then let Kx be the translation with the origin at x
- The dilation of X by K is then the set of points x so that the intersection of Kx with X is not empty.

Here the dimensions of K determine the expansion of foreground pixels in vertical and horizontal direction.

Figure 3.5: The connected components labeled visually with each label a different colour. Pipeline 1 on the left and pipeline 2 on the right.



3.4 Connected Component Labeling

For us to remove individual components based on their visual properties as described in the previous section, we have to be able to select the individual components. Therefore, we have to make the various components distinguishable. A very useful first step for making this distinction is *labeling connected components.* Connected component labeling (CCL) is an algorithm which assigns a label to each pixel equal to the labels of its connected neighboring pixels (or a new label when there are no connected neighbor pixels with a label). Figure 3.5 shows a visual representation of CCL used in the first two pipelines, where each color is a label. The algorithm can label based on 4-connectivity or 8-connectivity, where the latter considers diagonally connected pixels in addition to the neighboring pixels in vertical and horizontal directions. As a result we have a labeled image with each label corresponding to a connected component in the image (and the background having label 0). In the first pipeline, we compute some basic properties of these connected components such as area, center of mass, width and height. These properties are used for various clustering algorithms in this study, such as density based spatial clustering of applications with noise (see section 3.5.1) and minimum spanning tree angle based clustering (see 3.5.2). Furthermore, by labeling the components we can access component specific pixels for numerous operations.

In the second pipeline, we select potential text regions based on the variance in gray scale values per connected component. The rationale behind this is that regions with text have a higher contrast between text and background, while regions without text have lower contrast between their foreground and background pixels. Additionally, thresholding in the second pipeline is done based on the mean and standard deviation in gray scale intensities per connected component. The step before skew detection in the second pipeline, *Reprocess Noise'*, repeats the previously mentioned steps once.

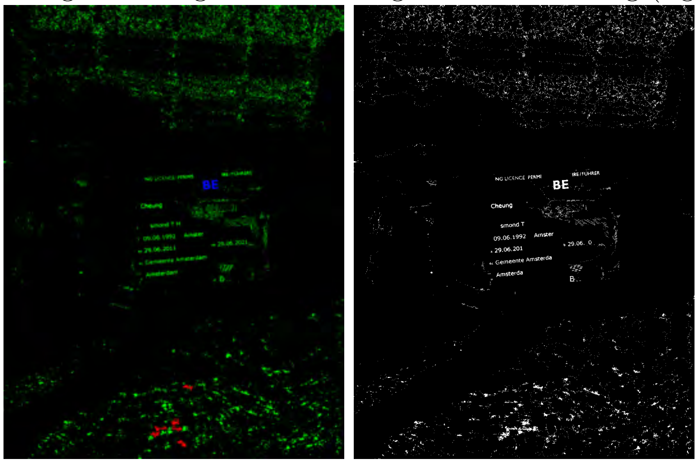
3.5 Text Line Extraction

In the first pipeline text lines are extracted in subsequent steps. Firstly, potential text regions are extracted by means of a clustering algorithm called *Density Based Spatial Clustering of Applications with Noise*, or DBSCAN. Secondly, individual text lines are extracted from these potential text regions by means of *Minimum Spanning Tree Angle based clustering.*

3.5.1 Density Based Spatial Clustering of Applications with Noise

After connected component labeling in the first pipeline, we use the area, width, height and coordinates of the center of mass of each component as input for density based spatial clustering of applications with noise (DBSCAN) to cluster the various image components in a 4 dimensional space. DBSCAN is an algorithm which clusters based on the density of the data points which need to be clustered. Text elements in an image are usually grouped closely together and share the same visual properties, while noise components are randomly located in the image and have various visual properties. Therefore, we use this clustering algorithm over other algorithms. The algorithm's parameters are ϵ, the maximum distance between points in a cluster, and the minimum number of points required to be considered a cluster. All points which are not clustered are most likely in low density areas and are considered as noise. Resulting clusters can compose of text as well as non-text elements and noise components are eliminated due to not satisfying the minimum number of points for it to be considered a cluster or they are too far away from other core points.

Figure 3.6: The created cluster by DBSCAN visually presented with each cluster a different colour (left). The image resulting after MST angle based clustering (right).



3.5.2 MST Angle based Clustering

Following DBSCAN, individual text lines were found using the minimum spanning tree angle based clustering algorithm. This algorithm is based on the rationale that elements belonging to the same text lines are closer together than other elements and that the edges between these elements have similar orientation.

The coordinates of the connected components' center of mass allows us to represent an image as a graph G. With this representation we construct the minimum spanning tree C. In this MST we calculate the angles between the edges in each vertex with more than one edge. The angles are determined by calculating the inverse cosine of the dot product between the two edges divided by the product of the edges' magnitudes:

$$\alpha = \cos^{-1}\left(\frac{\bar{e_1} \cdot \bar{e_2}}{|\bar{e_1}| \cdot |\bar{e_2}|}\right)$$

If a vertex is a leaf (vertex with degree 1), we set the angle at 180 degrees. As a result we have an angle matrix of $n \times n$, with n the number of vertices in C. We calculate a histogram over this matrix and remove all edges with angles deviating from the main orientation. Lastly, we discard the vertices with no edges. Translating the resulting set of vertices back to the image, results in an image with less non-text elements (according to the rationale). Below you can find pseudo code of the algorithm (see algorithm 1).

Algorithm 1: MST Angle based Clustering

```
   Data: Minimum Spanning Tree $C(V, E)$
   Result: $V(H) \subseteq V(C)$ with $\delta(H) > 0$
1  initialization;
2  foreach $v \in V(C)$ do
3      if $d(v) > 1$ then
4          foreach unique pair in $E(v)$ do
5              Calculate angle
6      else
7          Set angle at 180 deg
8  Calculate histogram of angles ;
9  Discard $e \in E(C)$ with angles deviating from main orientation;
10 Discard $v \in V(C)$ with $d(v) < 1$
```

3.6 Text Detection with a Full Convolutional Neural Network

At the base of the third pipeline is a full convolutional neural network, which is a neural network which is widely used for object classification. Based on the pixel values pf the input image, probabilities of each class are given for the object(s) of interest as output. Convolutional neural networks adopt their architecture from the visual cortex, where a set of cells look for specific features in the input the visual cortex receives. Similar to the set of neuronal cells looking for specific regions in a system, the FCNN has components with each their own specific tasks. These components are so called convolutional layers.

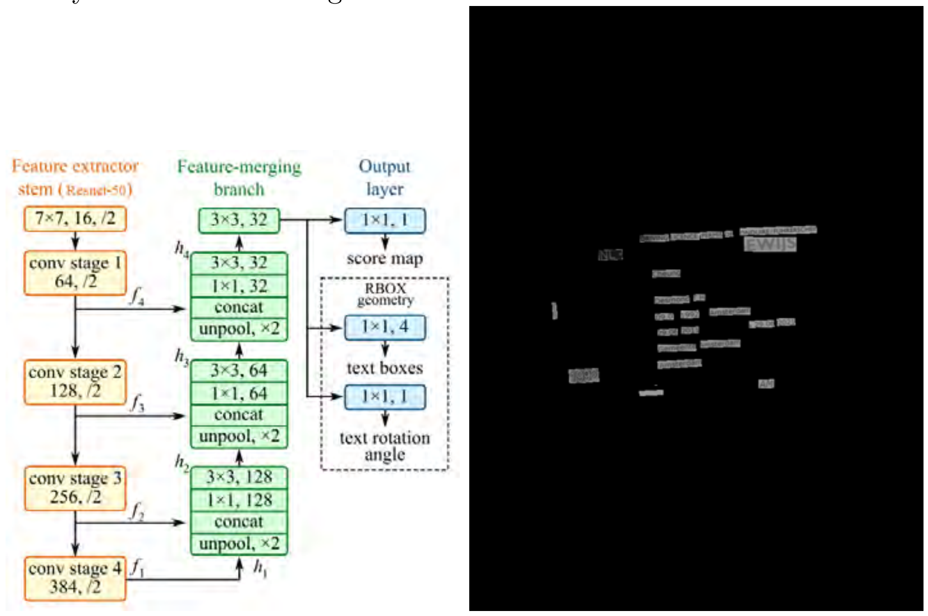
A convolutional layer is a kernel with certain dimensions consisting of weights which slides, or *convolves*, over the input image. Firstly, for every convolution the input image undergoes element-wise multiplication with the kernel. Secondly, the sum of these multiplications is calculated. Finally, resulting after all convolutions is a matrix with these sums which is called the *feature map.* Note that this feature map has smaller dimensions than the input image.

Next to convolutional layers, a FCNN consists also out of so-called *pooling layers.* These pooling layers are used to reduce the spatial dimensions of the input image's representations in the network (Singh Kathait et al., 2018). As a result the network has less information to work with, which leads to increased computational speed. Additionally, the reduction of spatial information results in less parameters and thus decreases the probability of over-fitting.

Each individual kernel can be seen as a feature identifier. These features are simple image characteristics such as edges, colours, curves and so on. The sum of the multiplications will be higher if a certain feature identifier extracts their feature. The weights of the feature identifiers are adjusted to classify objects properly by means of training with *backpropagation*. With backpropagation, the intial weights of the kernels are randomized. Then a training image goes through the neural network in the *forward pass*. After the forward pass, the output is determined and the error with the target value is calculated with the *loss function*. An example of a loss function is the mean squared error. Based on the loss, the weights are updated and backpropagation repeats again. This is done till the error is minimized and reaches a threshold where no significant improvement is achieved.

In our study we use a pre-trained model called ResNet50, which is a convolutional neural network which has already been trained on millions of images. For text detection, a large number of feature maps are merged with the existing model. These feature maps predict location and orientation of text regions. Figure 3.7 shows the an overview of the network architecture used and a sample output of the model.

Figure 3.7: An overview of the FCN used for text detection on the left and the predicted text regions by the FCN on the right.



3.7 Text Recognition

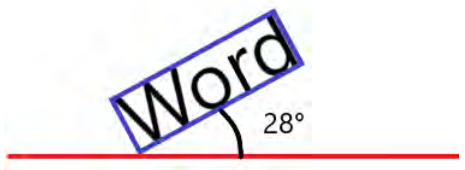
In this study, text recognition was performed using the open-source OCR engine Tesseract. The latest version of Tesseract firstly recognizes text lines with a Long Short-Term Memory neural network. LSTM neural networks are a type of recurrent neural network, which are often used for many different applications e.g. speech recognition, language modelling, image-to-text translation. Because our study makes use of Tesseract purely as text recognition module, we do not further elaborate on the full theory behind these networks. The model data for Latin-based languages supplied with the latest version of

Tesseract is trained on roughly 400,000 text lines with 4500 various fonts. Since our research scope does not include any font specific data, we chose not to (re-)train the LSTM neural network. From the recognized text lines, words are extracted by analyzing the spacing between text. This is a non-trivial task prior to word recognition (Smith, 2007). Successfully extracted words are then chopped into separate characters. Subsequently, these characters are recognized by a static classifier. An advantage of this static classifier is its reduced sensitivity to different fonts. However, this reduced sensitivity results in decreased ability to recognize text. Therefore, the output of the static classifier is used to train an adaptive classifier which has a higher sensitivity to various fonts and thus often used within each document (with fewer different fonts) to recognize text better. The two classifiers merely differ in character normalization prior to classification (Smith, 2007).

3.8 Skew Detection

We have implemented a self-developed robust method for skew detection based on image morphology at the end of the pipelines. This was done so that other noise components cant affect skew detection. The first step of the skew detection is connected component labeling on a copy of the segmented binary image. The second step removes very small and very large components. The rationale behind this is that the resulting image contains only words, which are used to determine the skew they are subject (Aradhya et al., 2006). The third step of the skew detection encloses the remaining components with a bounding boxes. A horizontal line is then connected with the most southern (lowest value of y) located corner of a bounding box. Then the angle is calculated from the horizontal line to the first side of the bounding box counterclockwise (see figure 3.8). The fifth step of the skew detection corrects the skew with the median of all calculated angles. We chose to take the median as corrected skew angle, because it is a robuster measure of central tendency than the mean.

Figure 3.8: A word enclosed with a bounding box (blue) and the angle it makes with the horizontal line (red)



Chapter 4

Experimental Setup

The first step in our experimental setup is data generation. In section 4.1 you can find the description of the data and how the data was obtained. The second step is the implementation of the pipelines described in chapter 3. We implemented three pipelines in total, of which two use conventional methodology for text extraction and one uses a full convolutional neural network for text detection. The third step of our setup is parameter optimization. We optimize the parameters (if possible) or use different methodology in each step of all pipelines. In section 4.2 we further elaborate on the different parameter configurations. The final step is the evaluation of the text extracted from the three pipelines with different parameter configurations. We explain the method to evaluate the extracted texts with the ground truth in section 4.3 and elaborate on the used performance measures. Furthermore, statistical analysis was performed on the obtained results to test significance.

4.1 Description of the Data

Unfortunately it is not an option to use only public datasets with text-in-the-wild images including ground truths, because this data has significantly different characteristics than law enforcement data. Law enforcement data often consists of various documents with different amounts of textual information. There is an excess of visual data extracted from law enforcement seized material. In order to asses the quality of text recognition on this data, we require the *ground truths* of these images. These ground truths contain all textual information of an image with complete precision. Prior to this study there were *no* ground truths available. Therefore, we generated ground truths for approximately 300 images. These images contain photographs of a PGP encrypted mobile phone. This phone's display shows various chat messages and email messages with varying amounts of text and noise, which thus forms a representative dataset suitable for our research.

Additionally, we synthetically created a dataset with photographs of a driver's license with various lighting, background, angles and overall image quality. This synthetic dataset is advantageous, because it requires only one ground truth for the entire set. We assessed the quality of the various pipelines on a public dataset(with similar size to that of the law enforcement data) and incorporated the results in this thesis even though the public dataset is not representative to law enforcement data.
Furthermore, we randomly split each of the datasets (PGP dataset and driver's license) in 70% training data and 30% test data. This was done to give generality to the produced results.

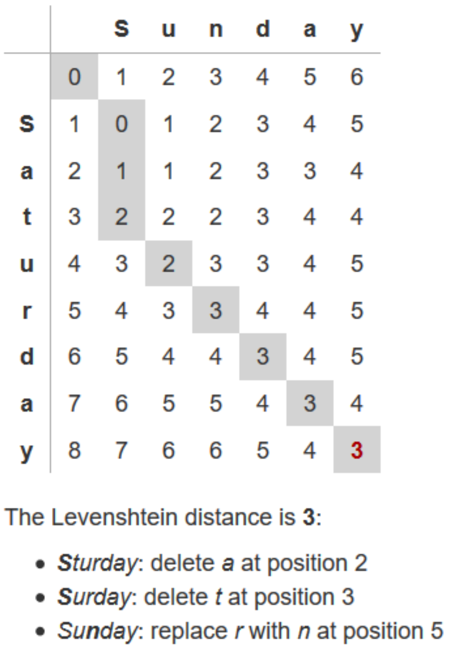
4.2 Parameter Optimization

After implementing the pipelines, we heuristically optimize the parameters of each pipeline. This heuristic approach of optimizing a parameter of an arbitrary pipeline, is done by varying the value of a parameter in certain range, while keeping other parameters in that pipeline constant. Thus, we assume that the individual parameters are independent from each other. This independence assumption of the parameters is due to otherwise large computational times. For the first pipeline, we optimize the two parameters of DBSCAN (ϵ and minPts) and test the performance of the pipeline with and without MST angle based clustering. The rationale behind the latter is that the algorithm assumes born digital images and thus might be affected by images rich in noise. For the second pipeline, we test between various thresholds for the Canny edge detection algorithm. There was no optimal method found in literature for determining the best thresholds for Canny edge detection and it appears to be that the optimal threshold is data dependent. Furthermore, we varied the size of the kernel used with the dilation operation. Additionally, we optimized the standard deviation that was used to remove false text regions based on gray scale intensities. For the third pipeline, there were no parameters to optimize. However, we evaluated two different thresholding methods, Otsu's and Sauvola's methods, for the binarization of the predicted text instances. Table 4.1 gives an overview of all parameter values that were used for each step of the pipelines.

Table 4.1: All the parameter values used for optimization of each step per pipeline.

Pipeline 1		**Pipeline 2**			**Pipeline 3**
DBSCAN	*MST Clustering*	*Canny Edge Detection*	*Dilation*	*σ of gray scale values*	*Thresholding*
ϵ: 0.2, 0.4, 0.5, 0.6, 0.8, 1.0	With or without	T_{low}: Otsu, 85, 127	Kernel dimensions: 2, 3, 4, 5	10,20,30,40,50	Otsu or Sauvola
minPts: 2, 3, 4, 5, 7					

Figure 4.1: An example of the Levenshtein distance between the words 'Saturday' and 'Sunday'



		S	u	n	d	a	y
	0	1	2	3	4	5	6
S	1	0	1	2	3	4	5
a	2	1	1	2	3	3	4
t	3	2	2	2	3	4	4
u	4	3	2	3	3	4	5
r	5	4	3	3	4	4	5
d	6	5	4	4	3	4	5
a	7	6	5	5	4	3	4
y	8	7	6	6	5	4	3

The Levenshtein distance is **3**:

- ***Sturday***: delete ***a*** at position 2
- ***Surday***: delete ***t*** at position 3
- ***Sunday***: replace ***r*** with ***n*** at position 5

4.3 Evaluation

In order to evaluate the various pipeline configurations and individual pipelines, we have to compare the extracted texts with the ground truths. Therefore, the Levenshtein's distance was used in this study evaluate the difference of two texts. The Levenshtein's distance is the minimal number of operations that is required to change one string into another. The operations which are taken into account are deletion, insertion and substitution. Figure 4.2 shows an example of how to determine the Levenshtein's distance to get from the word 'Sunday' to the word 'Saturday'. The previously mentioned operations can be done on character level and on word level. However, the aim of our study is to optimize text extraction in law enforcement data to make text in visual data searchable for a detective. Therefore, the performance measures used in our study are all on word level. The performance measures used in our study derive from the Levenshtein operations:

$$WER = \frac{\#deletions + \#insertions + \#substitutions}{\#words}$$

$$Precision = \frac{\#correct}{\#correct + \#substitutions + \#insertions}$$

$$Recall = \frac{\#correct}{\#correct + \#substitutions + \#deletions}$$

We use the harmonic mean of the latter two measures as leading performance measure to determine the optimal parameter configurations of the pipelines:

$$F - measure = \frac{2 * Precision * Recall}{Precision + Recall}$$

Finally, we perform a two-way ANOVA to determine weather there is a significant difference in performance of the various pipelines between the different datasets. For the two-way ANOVA there are three assumptions which need to be satisfied. Firstly, the independence assumption is most likely satisfied if we look at our research setup. Data generation most likely has not lead to dependent data samples, because we ensured heterogeneity in the data by synthetically adding noise. Secondly, we used the Shapiro-Wilks test to show we satisfy the normality assumption ($p > 0.05$). Finally, we tested homogeneity of variance of the data with Levene's test. The test was not significant and therefore satisfied the assumption of equal variances.

Chapter 5

Results

5.1 Pipeline 1

In table 5.1 below, when we look at the F-measures we see that a value of $\epsilon = 0.4$ produces the best results for the first two datasets. This might be due to similarities in noise between the two datasets. Both datasets are photographs made in resembling conditions of the same PGP encrypted cellphone. For the third dataset all values of ϵ perform almost equally well, except for $\epsilon = 0.2$ just like for the the PGP datasets. These findings might be due to the small sample size and thus insufficient heterogeneity in the datasets. This parameter, ϵ, is the maximum distance of a component to be considered a core point for a cluster. Thus, for $\epsilon = 0.2$ DBSCAN might be subject to heavy text loss due to text not grouped as dense to satisfy ϵ. And for values of ϵ that are too high, the clusters resulting from DBSCAN might include too much noise which else have been ommited. Therefore, for the best performing result over the data in this study we take $\epsilon = 0.4$ as optimal parameter value.

Table 5.1: Word-level benchmark averages of the first pipeline on training data with various values for ϵ, the first parameter of DBSCAN.

	PGP Chats				**PGP Emails**				**Synthesized set**			
	WER(%)	*Pr(%)*	*R(%)*	*F_1(%)*	*WER(%)*	*Pr(%)*	*R(%)*	*F_1(%)*	*WER(%)*	*Pr(%)*	*R(%)*	*F_1(%)*
0.2	117.20	4.40	4.67	4.45	88.23	6.20	6.35	6.21	119.46	3.61	2.64	2.72
0.4	106.36	**31.45**	45.41	**36.82**	83.00	**49.74**	**65.26**	**56.10**	118.67	**13.98**	11.09	11.18
0.5	108.82	30.68	45.69	36.37	86.51	45.16	64.89	52.89	117.96	11.55	11.35	10.58
0.6	104.46	30.92	**45.71**	36.55	**76.65**	43.93	64.24	51.80	**108.49**	11.26	11.20	10.20
0.8	**103.35**	29.71	45.23	35.49	77.29	43.61	63.89	51.43	109.55	13.25	12.25	10.97
1.0	104.87	28.36	44.07	34.14	77.77	42.04	63.88	50.36	114.67	12.111	**12.93**	**11.71**

Table 5.2 shows the best performance based on F-measures with a value of 4 for minimum number of points per cluster for the PGP datasets. Again for the synthesized dataset, the performance is uniformly distributed for all tested values for $minPts$. Therefore, we most likely can make the same conclusions as with the parameter ϵ and use $minPts = 4$ as optimal parameter value.

Table 5.3 shows that the exclusion of the minimum spanning tree angle based algorithm results in better text extraction. The reason for this could be that the algorithm was developed in a study (Böschen et al., 2015) using born digital images as data, while our data consist of analogously made photographs. Photographs are more likely subject to noise, such as lighting differences and skewness, than born digital images. This noise negatively impacts the clustering algorithm, due to the rationale behind the algorithm which assumes closely grouped components which share the same orientation to be of the same text line. A qualitative analysis of the algorithm (see figure 3.6) shows no decrease in noise and potential text loss. Therefore, we do not use the algorithm in our final evaluation.

Table 5.2: Word-level benchmark averages of the first pipeline on training data with various minimum samples per cluster, the second parameter of DBSCAN.

	PGP Chats				**PGP Emails**				**Synthesized set**			
	WER(%)	*Pr(%)*	*R(%)*	*F_1(%)*	*WER(%)*	*Pr(%)*	*R(%)*	*F_1(%)*	*WER(%)*	*Pr(%)*	*R(%)*	*F_1(%)*
2	117.20	26.68	44.54	34.54	88.23	44.05	64.62	52.03	119.46	11.60	**11.65**	10.48
3	106.36	31.29	**45.83**	36.85	83.00	46.54	65.51	54.04	118.67	11.43	10.75	10.06
4	105.51	31.31	45.78	**36.87**	76.81	**50.03**	**66.38**	**56.72**	115.88	11.95	9.43	9.32
5	104.46	**31.45**	45.41	36.82	**76.65**	49.74	65.26	56.10	**108.49**	**13.98**	11.09	**11.18**
7	**103.35**	31.02	44.08	36.09	77.29	47.5	61.58	53.33	109.55	12.13	10.07	10.32
10	104.87	29.09	40.67	33.62	77.77	44.67	55.88	49.37	114.67	10.83	9.62	9.46

Table 5.3: Word-level benchmark averages of the first pipeline on training data with and without minimum spanning tree angle based clustering.

	PGP Chats				**PGP Emails**				**Synthesized set**			
	WER(%)	*Pr(%)*	*R(%)*	*F_1(%)*	*WER(%)*	*Pr(%)*	*R(%)*	*F_1(%)*	*WER(%)*	*Pr(%)*	*R(%)*	*F_1(%)*
MST	121.58	16.23	22.52	18.7	120.5	20.47	26.73	22.95	118.29	5.38	5.09	4.77
No MST	**105.51**	**31.31**	**45.78**	**36.87**	**76.81**	**50.03**	**66.38**	**56.72**	**103.65**	**11.95**	**9.43**	**9.32**

5.2 Pipeline 2

In table 5.4, we can see that the empirically obtained value of 85 for the low threshold in the Canny edge detection algorithm is preferred over value 127 and the threshold determined using Otsu's method on the gray scale image. Especially with the synthesized dataset, determining the threshold with Otsu's method produces abysmal text extraction rates. It is likely that the double thresholding with Otsu's method does not suppress noise sufficiently and thus make text extraction significantly harder. These results show, like we found in our research, that there is no optimal method in determining these thresholds and that is is highly case dependent. Therefore, we use the empirically found value 85 for the low threshold (and 255 for the high threshold).

Table 5.4: Word-level benchmark averages of the second pipeline on training data with various thresholding methods of the Canny edge detection algorithm.

	PGP Chats				**PGP Emails**				**Synthesized set**			
	WER(%)	*Pr(%)*	*R(%)*	*F-score(%)*	*WER(%)*	*Pr(%)*	*R(%)*	*F-score(%)*	*WER(%)*	*Pr(%)*	*R(%)*	*F-score(%)*
Low T = 127	70.83	42.82	34.32	37.81	56.27	56.8	46.03	50.75	96.31	9.92	5.47	6.68
Low T = 85	**70.23**	44.09	**34.79**	**38.58**	**56.13**	**57.12**	**46.13**	**50.94**	**95.29**	**11.45**	**5.51**	**6.94**
Otsu	72.52	**46.52**	30.17	36.27	57.4	56.74	46.03	50.66	99.89	1.12	0.11	0.2

Based on the F-measures shown in table 5.5, we choose a kernel size of 3 in both directions for the dilation operation prior to connected component analysis. Dilation directly influences the following stage in the pipeline; connected component analysis. The components resulting from CCL influence the remaining pipeline stages. If dilation is performed with a too small kernel, then the connected components most likely contain too little information for filtering based on standard deviation of grayscale intensities. However, if the kernel used for dilation is too large then the connected components contain too many pixels for thresholding to be performed correctly. Therefore, we can conclude that dilation with a 3×3 kernel results in optimal connected components for text extraction with pipeline 2.

In table 5.6, the F-measures do not seem to differ much with various values of the standard deviation used for false text region removal for the PGP datasets. This means that each connected component has great contrast between fore- and background pixels with little noise, which results in little loss in text for every value for the standard de-

Table 5.5: Word-level benchmark averages of the second pipeline on training data for different kernel values of the dilation operation.

	PGP Chats				**PGP Emails**				**Synthesized set**			
	WER(%)	*Pr(%)*	*R(%)*	*F-score(%)*	*WER(%)*	*Pr(%)*	*R(%)*	*F-score(%)*	*WER(%)*	*Pr(%)*	*R(%)*	*F-score(%)*
2	101.33	22.38	28.53	24.98	82.94	35.67	44.85	39.64	103.8	7.2	4.83	5.3
3	76.09	34.4	**35.26**	**34.64**	66.55	43.81	**48.03**	45.7	94.72	12.97	**6.45**	**8.16**
4	**74.3**	**40.21**	29.3	33.61	**57.65**	**53.36**	46.12	**49.36**	96.27	12.31	5.73	7.2
5	81.65	35.27	18.96	24.34	64.28	52.97	38.6	44.46	**98.19**	**9.71**	4.71	5.92

viation in gray scale intensities. However, for the synthesized dataset best results are produced with a standard deviation of 20. The most likely reason for this is that text is not well distinguished from the background, which leads to loss of textual information per connected component. For example, if the colors of text and background are similar, then the difference in gray scale value would be smaller and thus lead to lower standard deviation per connected component. Causing the model incorrectly remove the connected component containing text. Therefore, to obtain optimal text extraction for all datasets we choose a standard deviation of 20 as optimal parameter setting.

Table 5.6: Word-level benchmark averages of the second pipeline on training data with various standard deviations in gray scale intensities used for removal of false text regions.

	PGP Chats				**PGP Emails**				**Synthesized set**			
	WER(%)	*Pr(%)*	*R(%)*	*F-score(%)*	*WER(%)*	*Pr(%)*	*R(%)*	*F-score(%)*	*WER(%)*	*Pr(%)*	*R(%)*	*F-score(%)*
10	74.22	40.3	28.89	33.36	57.65	53.36	46.12	49.36	98.99	8.75	6.33	6.82
20	74.3	40.21	29.3	33.61	57.65	53.36	46.12	49.36	**94.72**	12.97	**6.45**	**8.16**
30	73.84	40.68	29.51	33.91	**57.22**	55.3	**46.4**	50.32	96.76	**13.68**	3.28	4.92
40	**72.52**	46.52	**30.17**	**36.27**	57.4	**56.74**	46.03	**50.66**	99.21	6.66	0.79	1.39
50	81.5	**50.46**	18.78	27.03	60.04	51.61	44.62	47.74	100.0	0	0	0

5.3 Pipeline 3

From table 5.7 we can observe that Otsu's method outperforms Sauvola's method in terms of text extraction from law enforcement data. The predicted text regions from pipeline 3 eliminate a significant amount of noise, but create borders around these regions (see 3.3). Sauvola's method is great at handling lighting related noise, but is likely to be affected by these borders. This is due to the kernel sliding over the image and mostly containing large portions of background (see figure 3.7 and thus results in volatile thresholds for the whole image. However, the contrast of pixels containing text and background pixels is significantly increased for each predicted text region. Therefore, it is sufficient to apply global Otsu's method for each of these regions. Additionally, global thresholding requires less computation time than adaptive thresholding.

Table 5.7: Word-level benchmark averages of the third pipeline on training data with Otsu's method and Sauvola's method for thresholding.

	PGP Chats				**PGP Emails**				**Synthesized set**			
	WER(%)	*Pr(%)*	*R(%)*	*F-score(%)*	*WER(%)*	*Pr(%)*	*R(%)*	*F-score(%)*	*WER(%)*	*Pr(%)*	*R(%)*	*F-score(%)*
Otsu	**81.82**	**32.61**	**37.21**	**34.59**	**32.3**	**73.04**	**74.07**	**73.51**	**85.71**	**29.18**	**16.59**	**19.38**
Sauvola	89.26	22.48	24.05	23.09	49.15	56.4	60.47	58.32	89.22	19.91	14.59	16.24

5.4 Optimal pipelines evaluation

We can see from table 5.8 that overall best performance, based on F-score, is achieved by the third pipeline. With a two way ANOVA we show that the differences in performance between all pipelines are significant and the quality of text extraction is most likely dependent on the type of data. Furthermore, the p-value for interaction indicates that how the performance of text extraction changes for different datasets is dependent on the type of pipeline and vice versa. The ANOVA table can be found in Appendix B.

Table 5.8: Overview of the word-level benchmark averages of all pipelines on training data with the found optimal parameter configurations.

	PGP Chats				**PGP Emails**				**Synthesized set**			
	WER(%)	*Pr(%)*	*R(%)*	*F-score(%)*	*WER(%)*	*Pr(%)*	*R(%)*	*F-score(%)*	*WER(%)*	*Pr(%)*	*R(%)*	*F-score(%)*
Base	91.62	14.22	12.67	13.13	60.95	51.53	52.32	51.89	93.9	10.25	8.18	8.84
Pipeline 1	105.51	31.31	**45.78**	36.87	76.81	50.03	66.38	56.72	115.88	11.95	9.43	9.32
Pipeline 2	**70.23**	**44.09**	34.79	**38.58**	56.13	57.12	46.13	50.94	95.29	11.45	5.51	6.94
Pipeline 3	81.82	32.61	37.21	34.59	**32.3**	**73.04**	**74.07**	**73.51**	**85.71**	**29.18**	**16.59**	**19.38**

In table 5.9 we can see that skew detection probably does not improve text extraction from the PGP datasets, while text is most likely better extracted from the synthesized dataset with skew detection. This is due to the fact that the PGP datasets contain little skewness compared to the synthesized dataset. The robustness of the skew detection method causes the skew detection to wrongly attempt correct skew in the PGP datasets. Table B.2 in Appendix B tells us that skew detection can significantly improve text extraction ($p < 0.05$) dependent on the training dataset, but significant difference in extracted text can be found on the same dataset with or without skew detection ($p > 0.05$).

Table 5.9: Overview of the word-level benchmark averages of the third pipeline with and without skew detection on training data.

	PGP Chats				**PGP Emails**				**Synthesized set**			
	WER(%)	*Pr(%)*	*R(%)*	*F-score(%)*	*WER(%)*	*Pr(%)*	*R(%)*	*F-score(%)*	*WER(%)*	*Pr(%)*	*R(%)*	*F-score(%)*
Without skew detection	**81.82**	**32.61**	**37.21**	**34.59**	32.3	**73.04**	74.07	**73.51**	85.71	29.18	16.59	19.38
With skew detection	81.95	32.45	36.72	34.27	32.3	72.96	74.07	73.47	**84.01**	**27.0**	**18.97**	**21.65**

Table 5.10: Overview of the word-level benchmark averages of all pipelines on test data with the found optimal parameter configurations.

	PGP Chats				**PGP Emails**				**Synthesized set**				**Open source set**			
	WER(%)	*Pr(%)*	*R(%)*	*F-score(%)*	*WER(%)*	*Pr(%)*	*R(%)*	*F-score(%)*	*WER(%)*	*Pr(%)*	*R(%)*	*F-score(%)*	*WER(%)*	*Pr(%)*	*R(%)*	*F-score(%)*
Base	92.54	13.31	12.13	12.74	60.95	52.42	52.27	52.61	98.22	4.21	3.21	3.48	100	0	0	0
Pipeline 1	109.41	30.71	**45.17**	36.2	88.15	44.47	63.51	51.75	134.05	6.17	8.56	6.5	120.1	3.02	5.16	3,37
Pipeline 2	**73.69**	**43.32**	35.67	**38.67**	61.34	51.57	40.28	46.02	96.26	8.93	4.01	4.42	115.23	4.13	5.33	4,92
Pipeline 3	90.44	30.78	36.4	33.11	**34.31**	**70.77**	**73.24**	**71.95**	**88.51**	**19.44**	**11.76**	**14.09**	**99.22**	**5.87**	**6.24**	**6,09**

Tables 5.10 and 5.11 show that the test data follows the results of the train data, except for the PGP chat dataset. This might be due to the small data sample in this study. However, the differences between F-scores are little. Therefore, it is likely that these results can be reproduced with other data. Additionally, the two way ANOVAs lead to the same conclusions to be made about the significance of the test results as with the train results (see tables B.3 and B.4).

Table 5.11: Overview of the word-level benchmark averages of the third pipeline with and without skew detection on test data.

	PGP Chats				**PGP Emails**				**Synthesized set**				**Open source set**			
	WER(%)	*Pr(%)*	*R(%)*	*F-score(%)*	*WER(%)*	*Pr(%)*	*R(%)*	*F-score(%)*	*WER(%)*	*Pr(%)*	*R(%)*	*F-score(%)*	*WER(%)*	*Pr(%)*	*R(%)*	*F-score(%)*
Without skew detection	90.44	**30.78**	**36.4**	**33.11**	34.31	**70.77**	**73.24**	**71.95**	88.51	19.44	11.76	14.09	**99.22**	**5.87**	**6.24**	**6,09**
With skew detection	**89.91**	30.53	35.35	32.5	**34.43**	70.74	72.41	71.52	**85.39**	**23.88**	**15.78**	**18.75**	101.56	4.2	5.87	5.55

In conclusion, we find that pipeline 3 performs significantly better for the PGP email dataset and for the driver license dataset. For the PGP chat dataset, it seems that pipeline 2 is preferred over the other pipelines by small margins. However, pipeline 3 (only 1 parameter) is more robust than the other two pipelines (3 parameters). Therefore, our preference goes to pipeline 3 for all datasets. Additionally, the heuristic method for parameter optimization used in our study could lead to different results. However, this most likely increases the probability on overfitting on our data, especially with the small sample size.

Chapter 6

Conclusions and recommendations

6.1 Conclusions

The main goal of the current study was to examine optimizations of text extraction in law enforcement data. This was done by studying law enforcement data and using methodology from earlier studies about text extraction. We show in our study that there simultaneously exists homogeneity in the content of police data and also a significant amount of heterogeneity in the quality of the police data which directly impacts the quality of text extraction. We attempted to improve text extraction from law enforcement data, but were limited to a small sample size. Our work can be applied to multiple practices in police investigation. For instance, if the goal is to have optimal automated text extraction on the entire visual database collected by the Dutch police, then no assumptions can be made on image morphology. However, if there exists a certain scope in an investigation to optimize text extraction from , for example, screenshots from chat applications, then it is possible to define a certain template per screenshot and extract text more specifically per data type.

We compared two previous studies based on conventional methodology for text extraction and one previous study which detected text using a fully convolutional neural network. With our research data, we showed that deep learning methodology surpasses the performance of both conventional studies. This is most likely do to the fact that conventional methodology requires certain assumptions to be made about the data for feature extraction, while a full convolutional neural network learns these features during the training phase. With this finding, we confirm again that there is no homogeneity in image quality in law enforcement data. We also experimented text recognition without segmentation (OCR with base Tesseract) and showed that preprocessing is required to eliminate noise affecting text recognition accuracy.

Our research has shown that our proposed robust skew detection algorithm improves text recognition accuracy in skewed law enforcement data. We will also extend future research into skew detection with deep learning methodology. There is abundant more room for further progress in text extraction from law enforcement data using deep learning methodology.

This project is the first comprehensive assessment of text extraction on law enforcement data, using conventional and deep learning methodology. This new understanding should help improve automated text extraction from law enforcement data and thus police investigation procedures. Limitations of this study are the small sample size of the data, the usage of only Latin based lingual data and exclusion of handwritten data. Notwithstanding the relatively limited sample, this work offers valuable insights in the possibilities of automated text extraction in law enforcement data.

6.2 Recommendations

The findings of this study provide insights for future research in text extraction using deep learning methods. This methodology does not only have to be applied on text segmentation, but can also be used for text recognition, skew detection and even post processing. A reasonable approach to tackle the latter could be to develop a language model for slang encountered in law enforcement data.

Continued efforts are needed to make multi-lingual data more accessible to include non Latin based languages into text extraction. This can be achieved by means of a translator or by narrowly cooperating with foreign colleague law enforcement with ongoing research in similar fields.

Further research might explore the possibilities of handwritten text extraction. Potential solutions for handwritten text recognition also lie in deep learning methodology. The challenge with handwritten data is also the availability of ground truth data.

Taken together, these findings support strong recommendations to further investigate and experiment into deep learning methodology with text extraction. Additionally, if practical applications have been envisioned on specific data, conventional methodology can also provide improvement. This could be researched by categorizing various data types, such as chats, emails, bank transfer, street signs and many more.

Bibliography

Amin, Adnan and S Fischer (2000). "A Document Skew Detection Method Using the Hough Transform". In: *Pattern Analysis & Applications* 3.3, pp. 243–253. ISSN: 1433-7541. DOI: 10.1007/s100440070009. URL: http://dx.doi.org/10.1007/s100440070009.

Aradhya, V N Manjunath, G Hemantha Kumar, and P Shivakumara (2006). "{S}kew {D}etection {T}echnique for {B}inary {D}ocument {I}mages based on {H}ough {T}ransform". In: *{I}nt. {J}ournal of {I}nformation {T}echnology* 3.1, pp. 194–200. ISSN: 1305-2403. URL: https://pdfs.semanticscholar.org/3708/48c90fde1bdc4b0fa287acefece0da2eba34.pdf.

Böschen, Falk and Ansgar Scherp (2015). "Multi-oriented Text Extraction from Information Graphics". In: *Proceedings of the 2015 ACM Symposium on Document Engineering - DocEng '15*, pp. 35–38. DOI: 10.1145/2682571.2797092. URL: http://dl.acm.org/citation.cfm?doid=2682571.2797092.

— (2017). "A Comparison of Approaches for Automated Text Extraction from Scholarly Figures". In: 10133, pp. 15–27. ISSN: 00200255. DOI: 10.1007/978-3-319-51814-5. URL: http://link.springer.com/10.1007/978-3-319-51814-5.

Chaki, Nabendu, Soharab Hossain, and Shaikh Khalid Saeed (2014). *Studies in Computational Intelligence 560 Exploring Image Binarization Techniques.* Springer. URL: https://link-springer-com.vu-nl.idm.oclc.org/content/pdf/10.1007{\%}2F978-81-322-1907-1.pdf.

Chaudhuri, Arindam et al. (2017). *Optical Character Recognition Systems for Different Languages with Soft Computing.* Vol. 352, pp. 9–42. ISBN: 978-3-319-50251-9. DOI: 10.1007/978-3-319-50252-6. URL: http://link.springer.com/10.1007/978-3-319-50252-6.

Chiang, Yao-Yi and Craig A. Knoblock (2015). "Recognizing text in raster maps". In: *GeoInformatica* 19.1, pp. 1–27. ISSN: 1384-6175. DOI: 10.1007/s10707-014-0203-9. URL: http://link.springer.com/10.1007/s10707-014-0203-9.

Eikvil, Line (1993). "Optical Character Recognition". In: December.

Farahmand, Atena, Abdolhossein Sarrafzadeh, and Jamshid Shanbehzadeh (2013). "Document Image Noises and Removal Methods". In: *Proceedings of the Internatioanl MultiConference of Engineer and Computer Scientists* I. ISSN: 20780958. URL: http://www.iaeng.org/publication/IMECS2013/IMECS2013{_}pp436-440.pdf.

Hamad, Karez Abdulwahhab and Mehmet Kaya (2016). "A Detailed Analysis of Optical Character Recognition Technology". In: *International Journal of Applied Mathematics, Electronics and Computers.*

Kumar, Deepak and Dalwinder Singh (2012). "Modified Approach of Hough Transform for Skew Detection and Correction in Documented Images". In: *International Journal of Research in Computer Science* 2.3, pp. 37–40. ISSN: 22498257. DOI: 10.7815/ijorcs.23.2012.027. URL: http://www.ijorcs.org/manuscript/id/27/deepak-kumar/modified-approach-of-hough-transform-for-skew-detection-and-correction-in-documented-images.

Liang, Hong et al. (2017). "Text feature extraction based on deep learning: a review". In: *Eurasip Journal on Wireless Communications and Networking* 2017.1, pp. 1–12. ISSN: 16871499. DOI: 10.1186/s13638-017-0993-1.

Luppescu, Gregory and Francisco Romero. "Comparing Deep Learning and Conventional Machine Learning for Authorship Attribution and Text Generation". In: (), pp. 1–9. URL: `https://web.stanford.edu/class/cs224n/reports/2759272.pdf`.

Nagabhushan, P. (2010). "Text Extraction in Complex Color Document Images for Enhanced Readability". In: *Intelligent Information Management* 02.02, pp. 120–133. ISSN: 2150-8194. DOI: `10.4236/iim.2010.22015`. URL: `http://www.scirp.org/journal/iim`.

Robertson, Andrew (2013). "OPTICAL CHARACTER RECOGNITION A Study of Success and Failure in Innovation". In: *Management Decision* 9.3, pp. 213–223.

Roest, Dominique and Thinka Bethlem (2014). "Meer rendement uit digitaal beslag". In: *Tijdschrift voor de Politie* 76.1, pp. 34–38. URL: `https://www.politieacademie.nl/kennisenonderzoek/kennis/mediatheek/pdf/89259.pdf`.

Sas, Jerzy and Andrzej Zolnierek (2013). "Three-Stage method of text region extraction from diagram raster images". In: *Advances in Intelligent Systems and Computing*. Vol. 226. Springer, Heidelberg, pp. 527–538. ISBN: 9783319009681. DOI: `10.1007/978-3-319-00969-8_52`. URL: `http://link.springer.com/10.1007/978-3-319-00969-8{\_}52`.

Singh Kathait, Shailendra and Shubhrita Tiwari (2018). "Application of Image Processing and Convolution Networks in Intelligent Character Recognition for Digitized Forms Processing". In: *International Journal of Computer Applications* 179.20, pp. 975–8887. URL: `http://www.ijcaonline.org/archives/volume179/number20/kathait-2018-ijca-915460.pdf`.

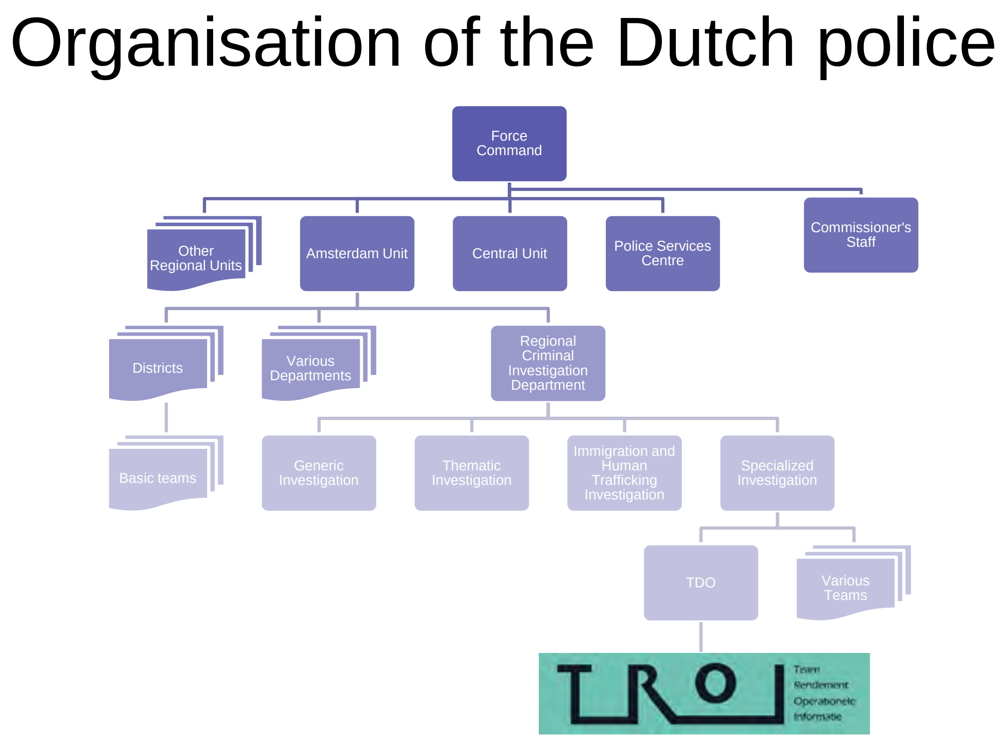
Smith, Ray (2007). "An overview of the tesseract OCR engine". In: *Proceedings of the International Conference on Document Analysis and Recognition, ICDAR* 2, pp. 629–633. ISSN: 15205363. DOI: `10.1109/ICDAR.2007.4376991`. arXiv: `arXiv:1011.1669v3`. URL: `https://static.googleusercontent.com/media/research.google.com/en//pubs/archive/33418.pdf`.

Wu, Jui Chen, Jun Wei Hsieh, and Yung Sheng Chen (2008). "Morphology-based text line extraction". In: *Machine Vision and Applications* 19.3, pp. 195–207. ISSN: 09328092. DOI: `10.1007/s00138-007-0092-0`. URL: `https://link-springer-com.vu-nl.idm.oclc.org/content/pdf/10.1007{\%}2Fs00138-007-0092-0.pdf`.

Zhou, Xinyu et al. (2017). "EAST: An efficient and accurate scene text detector". In: *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017* 2017-Janua, pp. 2642–2651. ISSN: 1063-6919. DOI: `10.1109/CVPR.2017.283`. arXiv: `1704.03155`. URL: `http://arxiv.org/abs/1704.03155`.

Appendices

Appendix A


Organisation of the Dutch police
Force Command
Other Regional Units
Amsterdam Unit
Central Unit
Police Services Centre
Commissioner's Staff
Districts
Various Departments
Regional Criminal Investigation Department
Basic teams
Generic Investigation
Thematic Investigation
Immigration and Human Trafficking Investigation
Specialized Investigation
TDO
Various Teams
TROI
Team Rendement Operationele Informatie

Appendix B

Table B.1: Two-way ANOVA table of the average F-scores of the different pipelines and train datasets

Source	**SS**	**df**	**MS**	**F**	**Prob>F**
Pipeline	5.445	4	1.36139	112.28	1.014e-79
Dataset	40.890	2	20.4456	1686.28	0
Interaction	3.091	8	0.386	31.86	5.1358e-45
Error	12.731	1050	0.012		
Total	62.158	1064			

Table B.2: Two-way ANOVA table of the F-scores of the different train datasets with and without skew detection

Source	**SS**	**df**	**MS**	**F**	**Prob>F**
Pipeline	0.006	1	0.0066	0.473	0.491
Dataset	20.538	2	10.269	818.899	1.168e-145
Interaction	0.0181	2	0.009	0.721	0.486
Error	5.267	420	0.013		
Total	25.829	425			

Table B.3: Two-way ANOVA table of the average F-scores of the different pipelines and test datasets

Source	**SS**	**df**	**MS**	**F**	**Prob>F**
Pipeline	0.719	4	0.179	25.728	1.198e-19
Dataset	30.276	3	10.092	1443.156	1.782e-268
Interaction	1.683	12	0.140	20.0571	7.806e-37
Error	4.055	580	0.007		
Total	36.734	599			

Table B.4: Two-way ANOVA table of the F-scores of the different test datasets with and without skew detection

Source	**SS**	**df**	**MS**	**F**	**Prob>F**
Pipeline	0.002	1	0.002	0.365	0.545
Dataset	14.917	3	4.972	629.764	3.732e-111
Interaction	0.021	3	0.007	0.907	0.438
Error	1.831	232	0.0079		
Total	16.774	239			