Multi-script handwritten character recognition

Surinta, Olarik

MULTI-SCRIPT HANDWRITTEN CHARACTER RECOGNITION

Using Feature Descriptors and Machine Learning

OLARIK SURINTA

Multi-Script Handwritten Character Recognition
Using Feature Descriptors and Machine Learning

Olarik Surinta

Cover idea: Olarik Surinta, Umaporn Pupphachai, and Harmen de Weerd
Cover design: Pluis
Printed by: HAVEKA
ISBN: 978-90-367-9149-5 (printed) / 978-90-367-9149-6 (electronic)

This research was supported by University of Groningen, the Netherlands and Mahasarakham University, Thailand.

university of groningen

Multi-Script Handwritten Character Recognition

Using Feature Descriptors and Machine Learning

PhD thesis

to obtain the degree of PhD at the
University of Groningen
on the authority of the
Rector Magnificus Prof. E. Sterken
and in accordance with
the decision by the College of Deans.

This thesis will be defended in public on

Friday 23 September 2016 at 14.30 hours

by

Olarik Surinta
born on 1 October 1978
in Chiang Mai, Thailand

Supervisor
Prof.dr. L.R.B. Schomaker

Co-Supervisor
Dr. M.A. Wiering

Assessment Committee
Prof.dr. R.I. Ingold
Prof.dr. B.B. Chaudhuri
Prof.dr. E.A. Valentijn

To my wonderful *Surinta* family:
Yuchan, Orarak, and of course the generous *Phianmongkhol* family.

Dedicated to the loving memory of my father, *Prasert Surinta*.
1949 – 2005

CONTENTS

1 INTRODUCTION

Handwritten character recognition systems are challenging research areas in pattern recognition and artificial intelligence (Schomaker et al., 2007; Bunke and Riesen, 2011; Liwicki et al., 2011; Uchida et al., 2012). This methodology helps you to transform any type of documents (e.g., historical, medieval, inscription, palm leaf manuscript, book, newspaper, and even unrestricted document formats) to an intelligible format. Fig. 1 shows an example of a palm leaf manuscript. The manuscript is collected by the palm leaf manuscript preservation project in northeastern Thailand at Mahasarakham university (Chamchong et al., 2010). Thousands of palm leaf manuscripts are being digitized by this project. There are only a few manuscripts that have a complete transcription.

Interestingly, a million of handwritten texts are stored in the national archive of the Netherlands. There are about 600 kilometers of bookshelves full of various documents, such as government archives, letters, land councils, and financial administration (van der Zant et al., 2008). It is difficult to

Figure 1: An example of a palm leaf manuscript.

manually annotate all of these handwritten documents. Therefore, a semi-automatic system is necessary for labeling the words in the handwritten documents. To deal with the large digitized archives, the MONK system, which is a system for character recognition, has been developed. There are a number of benefits of using the MONK system: (1) it reduces the workload of historians (e.g. palaeography and medievalist) to manually annotate handwritten documents; (2) it retrieves word-images (van Oosten and Schomaker, 2014) from large archives within a second by fast processing servers; and (3) it operates 24/7 hours and is accessible as a search engine.

We can also use character recognition research to handle other cases which are useful for real-life situations. For example, this research can help blind or visually impaired people to read a text directly from any source, such as text messages, newspapers, and books. In this case a character recognizer converts a text image to a special content that can be recognized as text, and also forwards the output to the speech synthesizer for speaking out that text. As another example, we can create an image translator that can capture any kind of text images from different sources (e.g., billboard, leaflet, and even road sign) and translate it to other languages. This can help people to understand across languages.

Character recognition systems have been used in many business areas, for instance, bank cheque processing, mail sorting, and postal automation (Tang et al., 2004; Gaceb et al., 2008; Pal et al., 2010; Salimi and Giveki, 2013). It is very clear that people in the world benefit from this research area.

Furthermore, the character recognition systems are related to research areas such as writer identification and verification (Schomaker and Bulacu, 2004; Bulacu, 2007; Brink et al., 2012; He et al., 2015). Writer identification refers to the process of identifying the writer and the authenticity of the document. It is widely used in, for example, a justice court, forensic labs and for identifying an automated signature of banking applications.

This dissertation addresses the problem of handwritten character recognition. Recognizing an isolated handwritten character image under the condition of not very many handwritten character samples and various writing styles create challenging research problems.

The main goal of the research presented in this dissertation is to study *robust feature extraction techniques* and *machine learning techniques* for handwritten character recognition. Furthermore, we also focus on the line segmentation technique which is part of the document layout analysis.

1.1 Handwritten Character Recognition Systems

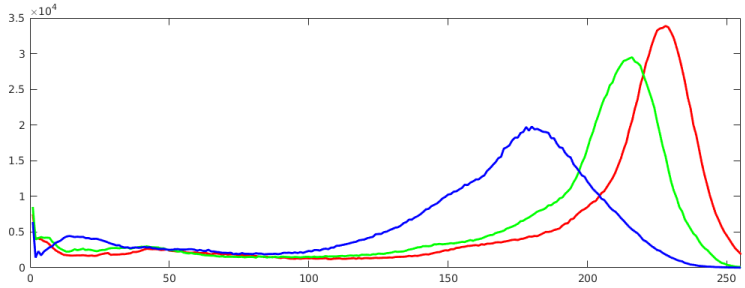
Three main components, consisting of pre-processing, segmentation, and character recognition, are essentially required for recognizing handwritten characters. These approaches are explained in the following sections.

1.1.1 Pre-processing

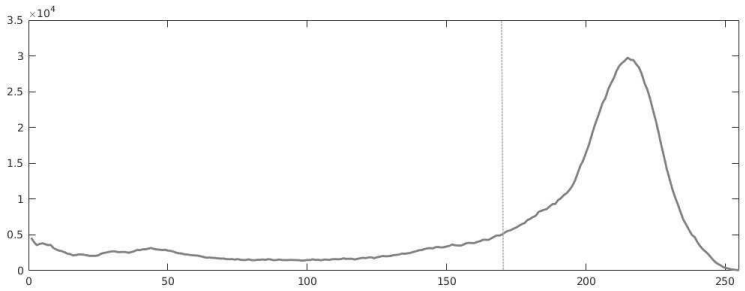
The aim of the pre-processing step is to produce data which are compatible with the handwritten character recognition systems. We mainly focus on the general operations consisting of image binarization, morphological operators, and image normalization.

The *image binarization* problem is to distinguish the foreground (ink or handwritten text) and the background (paper) without modifying the handwritten image (Solihin and Leedham, 1999; Plamondon and Srihari, 2000). The document image, which is generally stored in RGB (red, green, and blue) color space is converted to a grayscale image by calculating a weighted sum of the RGB intensity values of each pixel (Brink, 2011) as shown in Fig. 2.

When converting the grayscale image into a black and white (binary) image, an optimal threshold value is used to process an entire document image. Moreover, the histogram of the grayscale image which represents the distribution of the pixel intensity values of the grayscale image is constructed as shown in Fig. 2b. Thus, two main categories of thresholding algorithms; namely *global* and *local (adaptive)* have been proposed.



(a) Histogram of color image

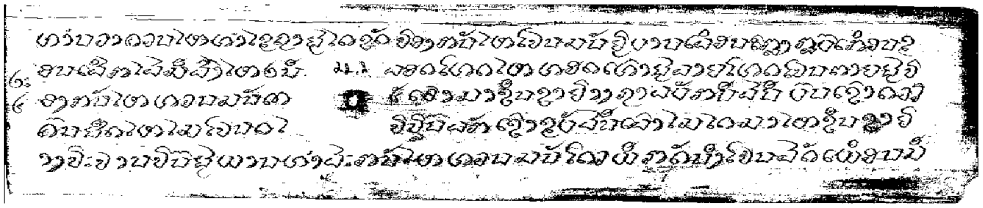


(b) Histogram of grayscale image

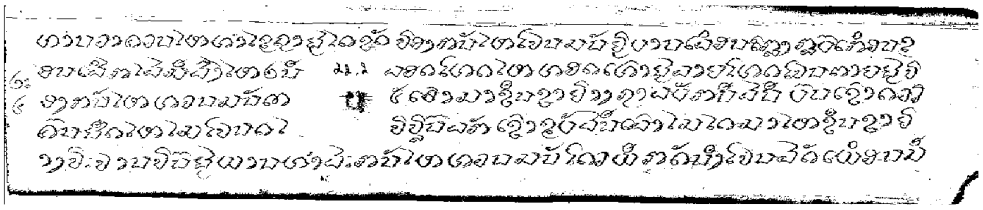
Figure 2: Illustration of the histogram of (a) color image and (b) grayscale image.

The *global* threshold (Otsu, 1979; Leedham et al., 2003) is chosen as the optimal threshold value which represents the entire document image. By this way the grayscale values of corresponding pixels are transformed into white or black. The corresponding pixel value is set to white when it is below the threshold value and set to black when it is above the threshold value. Fig. 3a shows the result of the global threshold after using Otsu's algorithm. Here, the threshold value is chosen as 175.

On the other hand, the *local* threshold creates different adapted values according to the information in the neighborhood area (Solihin and Leedham, 1999; Sauvola and Pietikäinen, 2000). Fig. 3b illustrates the result of the local threshold after using Sauvola's algorithm. Here, the neighborhood area is defined as 30×30 pixels.



(a) Otsu's algorithm when applied on the document from Fig. 1



(b) Sauvola's algorithm when applied on the document from Fig. 1

Figure 3: Result of the handwritten document image after using (a) Otsu's algorithm and (b) Sauvola's algorithm.

1.1.2 Segmentation

In the segmentation process we are concerned with the effect of the segmentation algorithms on the document image. The document image is partitioned into regions of interest, such as, lines, words, and also characters depending on the type of the application. A well-known basic technique called *projection profile* (Krishnamoorthy et al., 1993; Bukhari et al., 2011) which is based on the density of the ink on the document images, has been proposed and applied to all segmentation processes.

In the *line segmentation* process the horizontal projection profile is commonly used. The valleys in the histogram are considered to be the locations between the text lines. Fig. 4 shows the histogram of the horizontal profile of the palm leaf manuscript (see Fig. 3). We should note that one cannot

solve the segmentation problems only by using the horizontal projection profile, if the text lines in the document image are skewed or overlapping.

In the *word and character segmentation* process, the vertical projection profile has been proposed. We can apply this word segmentation method to some language scripts, such as Latin, since the natural handwriting style gives often a space between words and the handwriting of characters within a word is also connected. Fig. 5a shows a space between two words occurring in the natural handwriting style. As a result, this problem can be solved by using the vertical projection profile, see Fig. 5b.

On the contrary, in the Thai language (see Fig. 6), people ordinarily start writing the first word and keep writing until the end of the sentence without any space. Therefore, the basic word segmentation method cannot be applied to the Thai language. Nevertheless *character segmentation* can be directly applied to this language.

Furthermore, many machine learning techniques have been introduced to address the problems of document image segmentation. For example, for page segmentation of historical document images; convolutional auto-encoders (Chen et al., 2015), for text line extraction of historical manuscripts; dynamic multilayer perceptrons (Baechler et al., 2013), for word segmentation of handwritten documents; structural support vector machines (Ryu et al., 2015), and finally for character segmentation of cursive characters; Bayesian belief networks, support vector machines (Maragoudakis et al., 2002) and neural networks (Blumenstein, 2008), have been proposed.

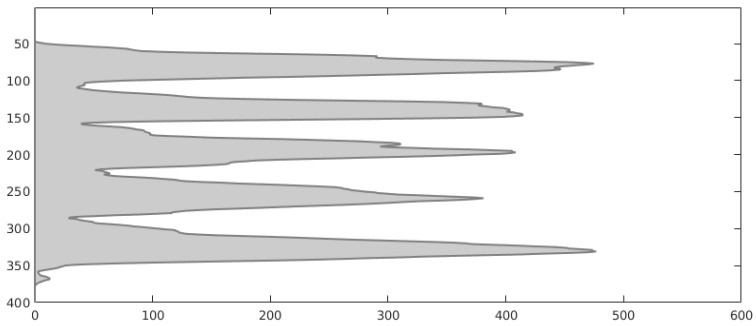
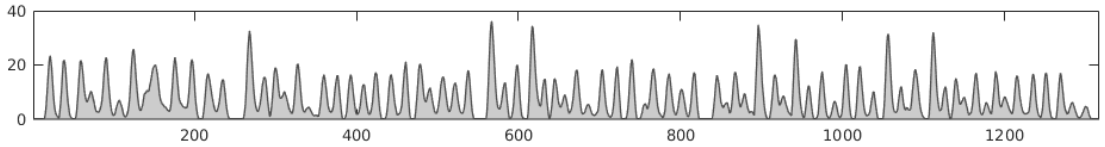


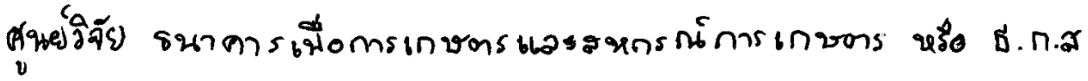
Figure 4: Illustration of the horizontal projection profile of the document image.

(a) Text line of Saint Gall manuscript

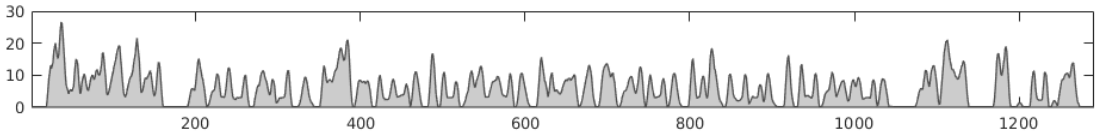


(b) Vertical projection histogram of Saint Gall manuscript

Figure 5: The result of the vertical projection profile of Saint Gall manuscript. (a) Saint Gall text. (b) The vertical projection profile of Saint Gall manuscript.



(a) Text line of Thai script



(b) Vertical projection histogram of Thai script

Figure 6: An example of the vertical projection profile according to the text line of Thai script. (a) Text line of Thai script. (b) The result of the vertical projection profile.

1.1.3 Character Recognition

Character recognition consists of two main components. These are feature extraction and machine learning techniques.

Feature extraction

Feature extraction is the process where important information, namely *feature vectors*, from the character images are created. It is one of many factors

which are used to increase the effectiveness of a recognition system. A popular and simple technique is the *pixel-based method* (Roy et al., 2004). It uses raw pixel intensities of the handwritten text without destructing the image that is itself represented as a feature vector. To make this method effective, however, a large amount of training data is required.

Many feature extraction techniques (which can also be called *feature descriptors*) have been proposed (Lowe, 2004; Dalal and Triggs, 2005) for the pattern recognition problems. In these techniques the gradient values extracted from local areas of the handwritten character image are used.

On the other hand, other feature learning methods, such as restricted Boltzmann machines, auto-encoders, and deep belief networks have been investigated in this area as well (Hinton et al., 2006; Coates et al., 2011b).

Machine learning

After feature extraction, a classifier is needed to determine the exact class of the characters. For this purpose, there are many classifiers which are based on machine learning techniques, for instance, support vector machines (SVM), artificial neural networks (ANN), deep belief networks (DBN), convolutional neural networks (CNN), hidden Markov models (HMM), and k-nearest neighbor (kNN) have been used in the character recognition process (Arica and Yarman-Vural, 2001; Hinton et al., 2006; Cireşan et al., 2011; Grafm̈uller and Beyerer, 2013; Schmidhuber, 2015).

1.2 Research Questions

Handwritten character recognition systems are definitely still an active area of research. Several challenges such as binarization of degraded document images, document layout analysis (e.g. segmenting the historical manuscripts into subsets, such as text line, background, and decoration graphic), text line segmentation (segmenting a text line into a single word), and handwritten character segmentation (e.g. segmenting overlapping or connected handwritten characters into isolated characters) have not been solved. Espe-

cially, the characteristic as well as the handwriting style of some languages (for example, Thai, Arabic, and Indian) give non-uniform spacing between two words and even intra-words. As a result, it is difficult to segment the exact word locations from a text line. Therefore, handwritten character recognition systems still have a long way to become almost perfect.

The major question that motivates the research in this dissertation is: *How can we improve the performance of the handwritten character recognition system?* This dissertation aims to contribute novel solutions to deal with the primary problems of handwritten character recognition. We start our investigation by focusing on the document layout analysis approach by extracting the textual information from the scanned document images. Moreover, we desire to understand which feature extraction techniques are most robust and precise for the recognition performance in various conditions. We address the following research questions:

RQ1 We focus on the task of segmenting the text areas from the handwritten document images. Here, we propose a novel line segmentation method for handwritten historical documents which are written in one column. The following questions come up; Is it possible to find potential candidates for starting points of lines separating upper and lower text areas of handwritten documents? What is the best technique that can separate two lines of handwritten text which are connected or overlapping?

RQ2 Very high accuracy is required for a character recognition system for being applied in real-world applications. However, there are difficult problems of the challenging handwritten scripts such as the similarities between the character sets and the similar structures between different characters. These factors negatively affect the performance of a handwritten character recognition system. What are the most robust local feature descriptors that can improve the recognition performance? Furthermore, when robust feature vectors are created, how do machine learning techniques perform for recognizing more complex handwritten characters? Can we assume that a simple machine learning technique such as the k-nearest neighbor algorithm provides high accuracy results?

RQ3 Several factors, including an insufficient amount of training samples and many different types of handwriting, can affect the performance of handwritten recognition systems. Thus, it is not a straightforward task to obtain high accuracies on difficult handwritten character datasets. How can we manage these issues? How do local feature descriptors and bag of visual words (BOW) perform for recognizing more complex handwritten characters? Furthermore, if we use the SVM algorithm for classification, which kernel type provides the best recognition accuracies on the challenging handwritten character datasets?

In order to answer all of these questions (**RQ1** to **RQ3**), Chapter 2 to Chapter 5 describe the research done in this dissertation. We will give concrete answers to these questions in Chapter 6.

1.3 Objectives of this Dissertation

There are two fundamental objectives at the basis of the research reported in this dissertation. We make these more explicit here.

1. *Line segmentation in handwritten documents:* the handwritten documents described in this dissertation are historical document images which contain several manuscripts, such as hagiography, medieval, historical, and contemporary manuscripts. These document image collections are written in a single column. Importantly, it contains a diversity of the common problems for line segmentation. Our first objective is to create a robust and effective line segmentation algorithm.

2. *Feature extraction / descriptors:* the handwritten character images presented in this dissertation are used as the input to the feature extraction techniques. We collect the handwritten characters from different language scripts, such as Thai, Bangla, Odia, and Latin scripts. These datasets are used in the recognition performance evaluation, because they contain several difficult problems and do not have a very large

number of handwritten character samples. In order to obtain an outstanding performance of the recognition system, we assume that feature extraction is the most important stage of a recognition algorithm, since it is able to capture the necessary information from the character images. Our second objective is to develop novel well performing feature extraction algorithms.

In our point of view, the present dissertation emphasizes effective techniques to solve the curious problems of handwritten character recognition, which include line segmentation and character recognition processes.

1.4 Contributions

The major contribution of the dissertation is a novel technique for feature extraction which is appropriate for isolated handwritten characters and represents both characters and digits from various handwritten scripts. We have performed experiments on five datasets containing MNIST (LeCun and Cortes, 1998), Latin (der Maaten, 2009), Thai (Surinta et al., 2012), Bangla (Bhowmik et al., 2009), and Odia (Mishra et al., 2013).

Additionally, we present a line segmentation method which is based on a well-known path-planning algorithm. This proposed algorithm is very beneficial when upper and lower text areas are overlapping or connected. We have performed experiments on the Saint Gall dataset (Fischer et al., 2011) and on several historical handwritten documents from the MONK line segmentation dataset (MLS) (Surinta et al., 2014). The specific contributions of the dissertation are as follows.

In Chapter 2, we propose a line segmentation method which is based on the well-known A^* path-planning algorithm (Nilsson, 1982). A number of cost functions has been designed to determine the optimal path which is separating upper and lower text regions. The proposed method is very useful when two subsequent lines of the handwritten text are overlapping or connected. The reason is that the combined cost function allows a separating path to pass through ink pixels or get close to text areas. This chapter is based on the following publication:

Surinta, O., Holtkamp, M., Karaaba, M.F., van Oosten, J.P., Schomaker, L.R.B., and Wiering, M.A. (2014). A^* path planning for line segmentation of handwritten documents. In Frontiers in Handwriting Recognition (ICFHR), The 14th International Conference on, pages 175-180.

Chapter 3 presents a method that achieves a high performance on the isolated handwritten Bangla digit dataset. We have compared four different techniques including the contour angular technique (CAT), the hotspot technique (HOT), the gray pixel-based method (GPB), and the black and white down scaled method (BWS). To obtain the highest recognition accuracy, the outputs of four classifiers are combined and then new data are classified by taking a vote of their predictions. This chapter is based on the following publications:

Surinta, O., Schomaker, L.R.B., and Wiering, M. A. (2012). Handwritten character classification using the hotspot feature extraction technique. In Pattern Recognition Applications and Methods (ICPRAM), The 1st International Conference on, pages 261-264. SciTePress.

Surinta, O., Schomaker, L.R.B., and Wiering, M.A. (2013). A comparison of feature extraction and pixel-based methods for recognizing handwritten Bangla digits. In Document Analysis and Recognition (ICDAR), The 12th International Conference on, pages 165-169. IEEE Computer Society.

In Chapter 4, we first provide a new standard Thai handwritten character dataset that is provided for the evaluation of classification and feature extraction techniques. It contains difficult problems such as complex structural characteristics of the characters and strong similarities between different character sets. We show that the local gradient feature descriptors, namely the scale invariant feature transform (SIFT) (Lowe, 2004) and the histogram of oriented gradients (HOG) (Lowe, 2004) obtain very high recognition performances when combined with the SVM (Vapnik, 1998) on the handwritten character datasets. This chapter is based on the following journal publication:

> Surinta, O., Karaaba, M.F., Schomaker, L.R.B., and Wiering, M.A. (2015). Recognition of handwritten characters using local gradient feature descriptors. Engineering Applications of Artificial Intelligence, 45:405-414.

Finally, Chapter 5 proposes the use of bag of visual words (BOW) (Csurka et al., 2004) for recognizing handwritten characters. There are some challenges in the handwritten character datasets, such as the writing styles, similar structures of different characters, background noise, and a lack of a large amount of handwritten character examples. In this approach, the intensity values in each local area are processed by using the HOG descriptor, called HOG-BOW. The HOG descriptor captures the gradient structure of the local shape and provides more robust features than with pixel intensities directly. Subsequently, a codebook is computed by using K-means clustering which is an unsupervised clustering algorithm. Eventually, to create the feature vectors, the soft-assignment scheme (Coates et al., 2011b) that computes the activity of each cluster is used. We have achieved very high recognition accuracies with the HOG-BOW method with L2 regularized support vector machines (L2-SVM) (Fan et al., 2008). This chapter is based on the following publication:

> Surinta, O., Karaaba, M.F., Mishra, T.K., Schomaker, L.R.B., and Wiering, M.A. (2015). Recognizing Handwritten Characters with Local Descriptors and Bags of Visual Words. In Engineering Applications of Neural Networks (EANN), 16th International Conference on, pages 255-264.

DOCUMENT LINE IDENTIFICATION 2

This chapter addresses the problem of document layout analysis of historical manuscripts. The term document layout analysis refers to partitioning the manuscript into a set of regions of interest. Transformation of the manuscript into meaningful structures is an essential step in the document layout analysis. To obtain textual information, a segmentation process (e.g., line, word, and character segmentation) is applied to the region of interest.

Consequently, this chapter describes the use of a novel A^* path-planning algorithm for performing line segmentation of handwritten documents. The novelty of the proposed approach lies in the use of a smart combination of simple soft cost functions that allows an artificial agent to compute paths separating the upper and lower text fields. The use of soft cost functions enables the agent to compute near-optimal separating paths even if the upper and lower text parts are overlapping in particular places. We have performed experiments on the Saint Gall and Monk line segmentation (MLS) datasets. The experimental results show that our proposed method performs very well on the Saint Gall dataset, and also demonstrate that our algorithm is able to cope well with the much more complicated MLS dataset.

Part of this chapter has previously been published in *Proceedings of the International Conference on Frontiers in Handwriting Recognition (ICFHR2014)*

Current search engines are very useful for people to search for information on the Internet. However, there is still a lot of information available on the Internet that cannot be used efficiently, because this information is contained in scanned document images that cannot be read or understood in an effective way by current search engine technology. We are especially interested in making handwritten documents accessible to people by recognizing the contents of these documents and making them searchable with a new generation of search engines tailored to handwritten documents.

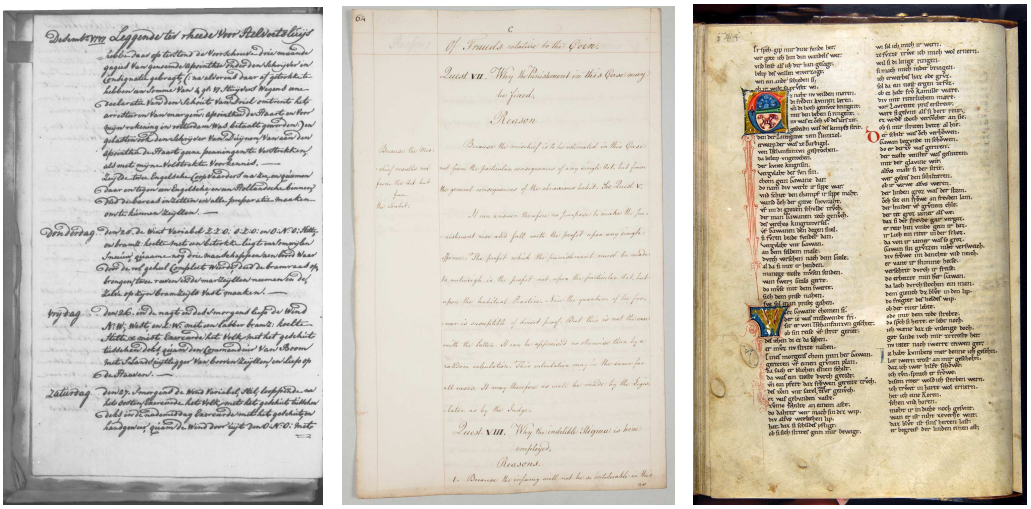
The *MONK* system is a historical manuscript recognition system, consisting of many techniques for searching for words in historical manuscript collections. The system consists of different handwriting recognition algorithms, which are trained by crowd sourcing techniques where volunteers can create ground-truth labels for words and lines that occur in the historical documents (Bulacu et al., 2007, 2009).

Extracting the textual information from the scanned document images is problematic and challenging (Gupta et al., 2006). The reason is that there are various types of documents which have different purposes and uses. Document layout analysis is an important process in character recognition systems which aims to address the problem of information extraction from various document images (Singh and Kumar, 2014). With this process the *regions of interests* in a document image are identified and categorized into categories, such as title, author, text, figure, and footer (Eickel, 1990; Ingold and Armangil, 1991) depending on the objective of the application. Fig. 7 shows some examples of historical document manuscripts.

2.1 Layout Analysis

In order to better understand the layout analysis, the basic idea starts by analyzing the document image and then generating the tree structure from the document image which is constructed by using the projection profile.

Krishnamoorthy et al. (1993) proposed the well-known X-Y tree structure algorithm for extracting the spatial structure of technical image documents



(a) MLS dataset (b) Bentham dataset (c) Parzival dataset

Figure 7: Some examples of the historical manuscripts: (a) manuscript from MLS dataset (Captain's logs, 1777), (b) manuscript from Bentham dataset and (c) manuscript from Parzival dataset (page 114).

into the tree structure. Ha et al. (1995) proposed the recursive X-Y cut algorithm which distinguishes a document image into a set of rectangular blocks. With this method the regions of interest are defined by using the profile analysis. Then the sub-region locations are assigned by local histograms. This method divides the document image recursively into a set of rectangular blocks.

Lee and Ryu (2001) proposed the parameter-free geometric method for page segmentation and region identification from the document image. To create a homogeneous region, the pyramidal quadtree structure is constructed and analyzed according to the top-down approach. For the top-level of the pyramid, the bounding boxes of connected components are extracted. Then, the periodicity of each region is estimated using the projection profile (both horizontal and vertical). In this stage, text regions can be distinguished from the other regions. However, dividing a region into small regions is required if the region is not an isolated periodical region.

On the other hand, the layout analysis can benefit from machine learning techniques such as support vector machines (SVM), multi-layer perceptrons (MLP), and dynamic multi-layer perceptrons (DMLP) (Baechler et al., 2013). The machine learning techniques classify the document images by learning different content areas from them and classify the content areas, such as text block, background, periphery, and decoration (Wei et al., 2014). Consequently, the content of the document image can be created according to feature extraction techniques, such as Gradient shape features (GSF), local binary patterns (LBP), and rotation invariant uniform (RIU) (Diem et al., 2011; Wei et al., 2014). In the layout analysis process, the document image is prepared for subsequent processing steps including line, word, and character segmentation. In this chapter, we propose a novel A* path-planning algorithm for line segmentation of handwritten documents as described in Section 2.2.

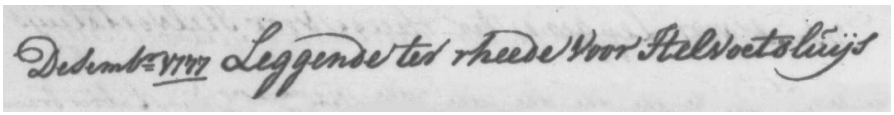
Word Segmentation

In word segmentation, usually the gap between two words is considered as a candidate word separator since the Latin language uses a space to separate words (see Fig. 8a). However, this rule cannot be applied to some other languages, such as Thai (see Fig. 9), Laos, and Bangla (or Bengali). Especially, the non-uniform spacing of words can appear in the modern Thai handwriting styles as shown in Fig. 8b. Furthermore, the challenge in word segmentation is that neighboring words may be connected.

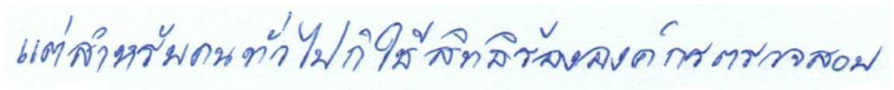
Importantly, various word segmentation techniques take the gaps between two connected component (CC) of words (Manmatha and Srimal, 1999; Louloudis et al., 2009; Simistira et al., 2011) into account.

For instance, Manmatha and Srimal (1999) proposed a blob analysis method for segmenting words on the George Washington dataset (George Washington, 1999). By this method, the filter operator performs a Laplacian of Gaussian (LoG) to find areas of edges in the document images and hence the blob extraction maps the document image and locates the word level.

Machine learning techniques have also been applied to word segmentation applications. For example (Louloudis et al., 2009) used unsupervised

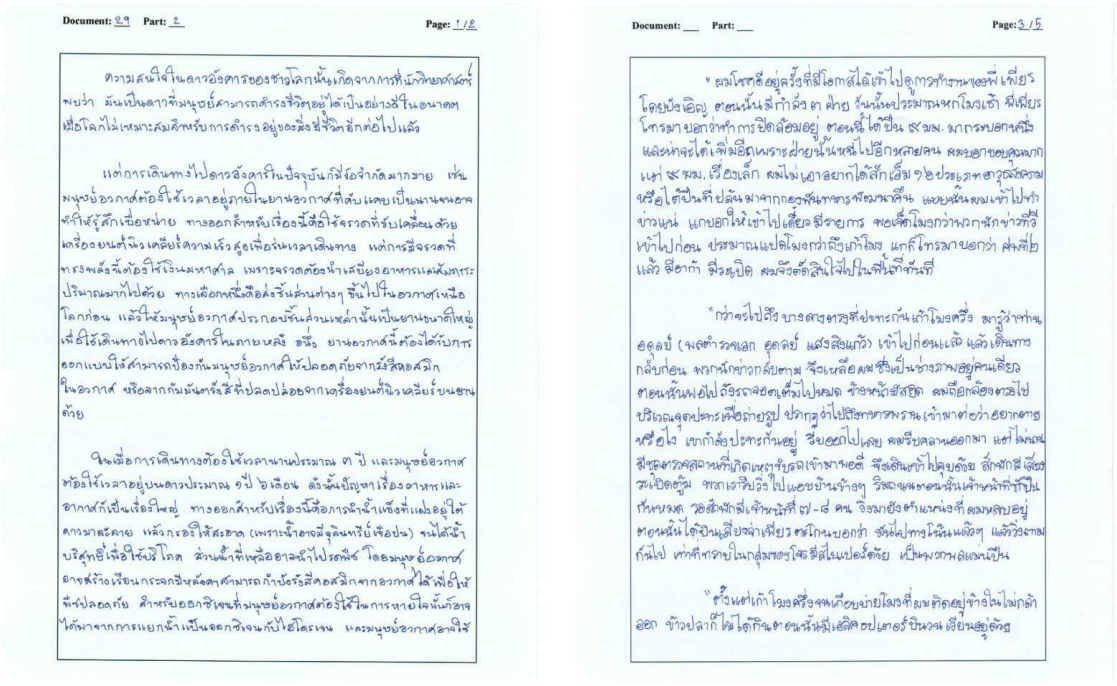


(a)



(b)

Figure 8: An illustration of the challenges in word segmentation. (a) Handwritten text from the MLS dataset (Captain's logs, 1777), (b) Thai handwritten text from Thai handwritten documents dataset.



(a) (b)

Figure 9: Illustration of the Thai handwritten documents dataset. In (a) and (b), the native writers studied in university and are aged from 20 to 23 years old.

learning to segment a sentence of a text line into words. The Euclidean gap metric which is represented as the minimum distance between adjacent overlapped components is created. The Gaussian mixtures method is used to divide the Euclidean gap metric data into two clusters (inter-class and intra-class) where a minimum distance model is associated for each cluster.

As a supervised learning technique, the gap metric which is represented for measuring the separability of CC is extracted and then given to the soft-margin SVM with a linear kernel as a classifier (Papavassiliou et al., 2010; Simistira et al., 2011).

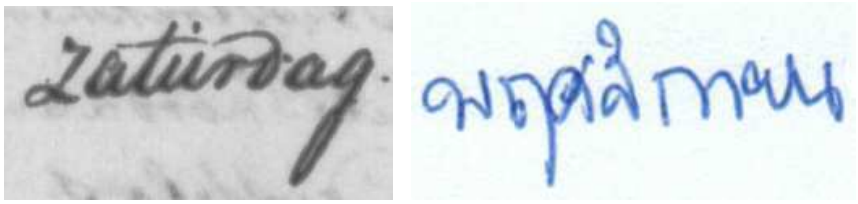
Character Segmentation

In character segmentation, the word image is divided into small segmented regions each of which probably contains an isolated character and then is detected by the CC technique (Naveena and Aradhya, 2012). Usually, the vertical projection profile, which is a traditional segmentation technique, is used (Xiu et al., 2006).

Due to the diversity of the handwriting styles, the vertical profile may not get an advantage for the Latin language, because the nature of the Latin handwriting is that neighbor characters are connected to each other as shown in Fig. 10.

Many holistic methods have been proposed for character segmentation. Arica and Yarman-Vural (2002) proposed the shortest-path algorithm which segments the word into characters, parts of the character, and even strokes. This method used a hidden Markov model (HMM) to recognize a word from each segmented partition.

On the other hand, Salvi et al. (2013) proposed a new global non-holistic method to segment characters from handwritten words by using an average longest path algorithm. The algorithm finds the candidate segmentation which is represented by the vertices of the word images. After that, to find the longest path between the vertices, each candidate segment is fed into SVM classifiers. The proper characters are then obtained from this process.



(a) Dutch word, “Saturday” (b) Thai word, “November”

Figure 10: Illustration of the challenges in character segmentation. (a) word from the MLS dataset (Captain’s logs, 1777), (b) Thai word from Thai handwritten documents dataset.

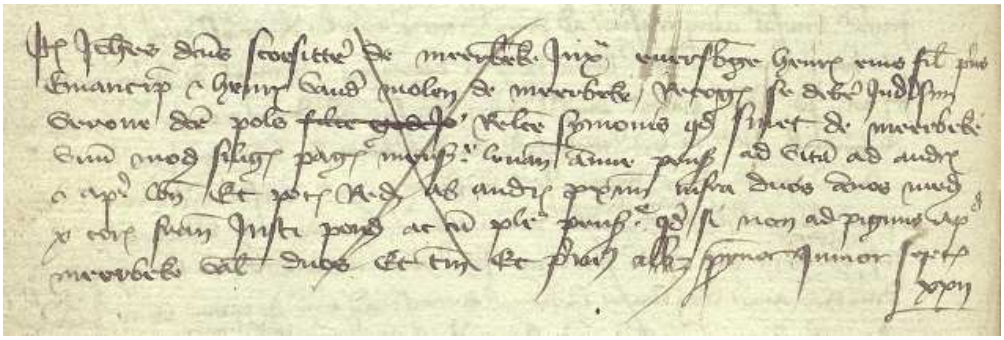
2.2 Line Segmentation

We will now describe a novel line segmentation algorithm for handwritten documents. Line segmentation (Likforman-Sulem et al., 2007) is one of the first techniques that needs to be applied to a document, before individual words or characters can be found and (parts of) the handwritten text can be automatically recognized. It should be noted that line segmentation is an intrinsically ill-posed problem that can only be solved using an interaction between classifiers and separation modules. Therefore, there is currently no method that can optimally deal with all difficulties such as curved lines and touching text lines, see Fig. 11, Fig. 13a and Fig. 17d for some complicated examples.

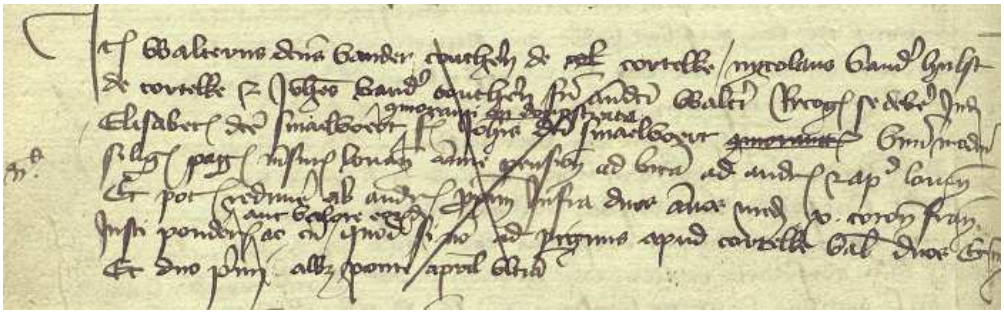
Related Work

The first step that is performed by a line segmentation algorithm is to find potential candidates for starting points of lines separating upper and lower text fields. Most often this step uses horizontal projection profiles, in which the amount of black ink is summed over the x-axis to obtain a profile indicating text areas having a lot or little to no black ink. Bulacu et al. (2007) proposed the smoothed horizontal projection profile to more robustly detect peaks and valleys in the binarized document image. In (Arivazhagan

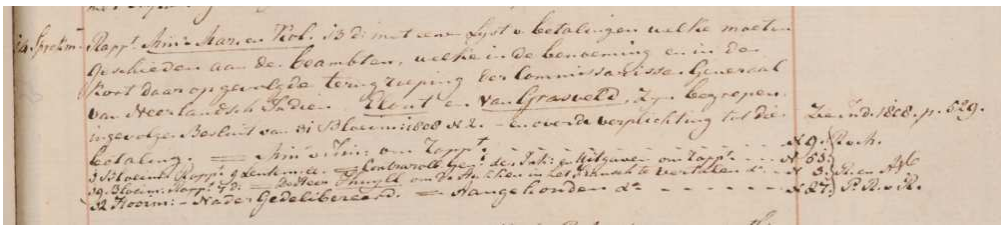
et al., 2007), the handwritten document is divided into chunks, and the smoothed projection profile in each chunk is calculated. Then, the valleys in the projection profile are considered as the starting state of the text lines. Also in (Chamchong and Fung, 2012), the baselines of the valleys are used to define the starting states for the separating lines.



(a) Aldermen's court, Leuven 1421



(b) Aldermen's court, Leuven 1421



(c) Cabinet of the King, KdK 1893

Figure 11: A complicated historical manuscript example from the MLS dataset.

After defining the starting states, various methods for finding the line separating upper and lower text ares have been proposed, we refer to (Likforman-Sulem et al., 2007) for a complete survey on this area. In (Bulacu et al., 2007), a droplet method that preserves the ink connectivity is developed. Beginning in the starting state, first an initial straight path is generated. Then, the document image is turned 90 degrees and an artificial water droplet is moved from the top to the bottom of the page. This droplet tries to move around the ink along the straight path with the aim to preserve the ascenders and descenders in the final segmentation. The experimental results on a part of a dataset named *"the cabinet of the Dutch queen"* containing 32,816 lines (31.6 per page) showed that 99.8% of the lines were correctly segmented.

Louloudis et al. (2009) proposed the use of the Hough transform to perform line segmentation. In this method, the average width and height of connected components in the whole document are computed and used for partitioning the text into sub-areas. The sub-areas are again partitioned into equally sized blocks. After that the ink gravity enter is computed in each block. Finally, the set of all gravity center points is processed by the linear Hough transform to find a straight separating line. This technique provides a line detection rate of 97.4% on the ICDAR 2007 handwriting segmentation dataset.

Saabni and El-Sana (2011) proposed the use of the seam carving method to perform language-independent text line extraction. This method finds the extreme points that indicate the layout of text lines by first generating an energy map using the signed distance transform. The minimum energy paths pass along the middle of the text lines. The region of the text line is estimated from a set of intersecting components. Finally, the components between two consecutive lines are extracted and associated to the closest text line.

Garz et al. (2012) proposed a binarization-free text line segmentation algorithm. First, parts-of-character interest points are located by means of the Difference of Gaussians (DoG) filter after which locations of local minima and maxima are found in the gray-scale image. These detected interest points represent the most significant locations of portions of the text. Then,

an energy map is computed around the located text points, and the seam carving technique is used to find a connected path with minimum cost that goes through low energy parts. This technique provides a hit rate of 0.9865 on 1,431 text lines of the Saint Gall dataset. The dataset is available from http://www.iam.unibe.ch/fki/databases.

Although these previous methods perform well when the text is structured, they still suffer from inaccurate line segmentations in case characters of subsequent lines are overlapping or touching. Therefore the aim of this research is to develop a robust method to deal with this overlapping or touching text-lines problem and to obtain accurate line segmentations for different kinds of manuscripts.

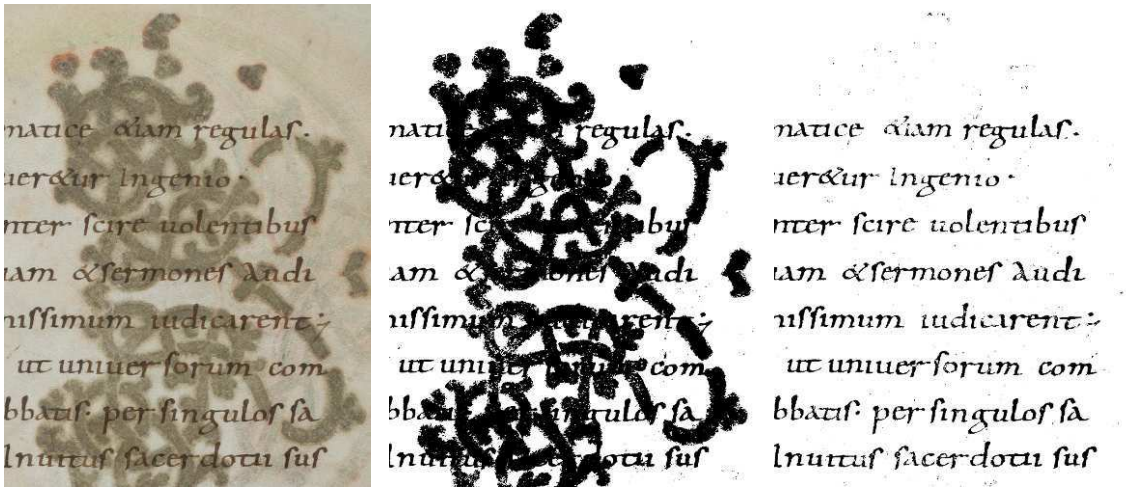
Contributions

We propose a line segmentation method based on the A^* path-planning algorithm (Nilsson, 1982). This well-known path-planning algorithm is combined with a number of cost functions to determine the optimal path separating upper and lower text areas. The cost functions have been designed in order to allow the separating path to go through text areas, although the path incurs a cost if it cuts ink pixels or gets close to them. This makes the proposed method very useful in case upper and lower text fields are overlapping or connected. The A^* path-planning algorithm has been widely used in the field of artificial intelligence, however, this chapter shows that this technique can also be very beneficial for line segmentation purposes. We have performed experiments on the Saint Gall dataset, and on several historical handwritten documents from the *MONK* line segmentation (MLS) dataset, which is a selection from the *MONK* collection. The MLS dataset is available from http://www.ai.rug.nl/~mrolarik/MLS/ for research purposes.

2.2.1 Text Line Localization

An important aspect of line segmentation is to automatically detect the locations of text lines. In our approach, text line localization is performed in

two steps: binarization and projection profile analysis. In the first step, the handwritten document images are binarized using a binarization technique. Such a technique takes into account the diversity of document images, texts, images, mixtures of texts, and images, line drawings, and noisy or degraded document images. Bulacu et al. (2007) and Surinta et al. (2013) use Otsu's algorithm, a global binarization technique, in their work. Otsu's algorithm uses one threshold value to process an entire document image. This algorithm is not performing well when the background of the document image is complicated as shown in Fig. 12b. On the other hand, Sauvola's algorithm (Sauvola and Pietikäinen, 2000) for local binarization copes effectively with complex backgrounds as shown in Fig. 12c. Because the contrast between handwriting and background is low (see Fig. 12a), the threshold value is calculated by the mean and standard deviation of the local neighborhood of the gray pixel values. This threshold value has to be calculated for each pixel (Singh et al., 2011; Shafait et al., 2008).



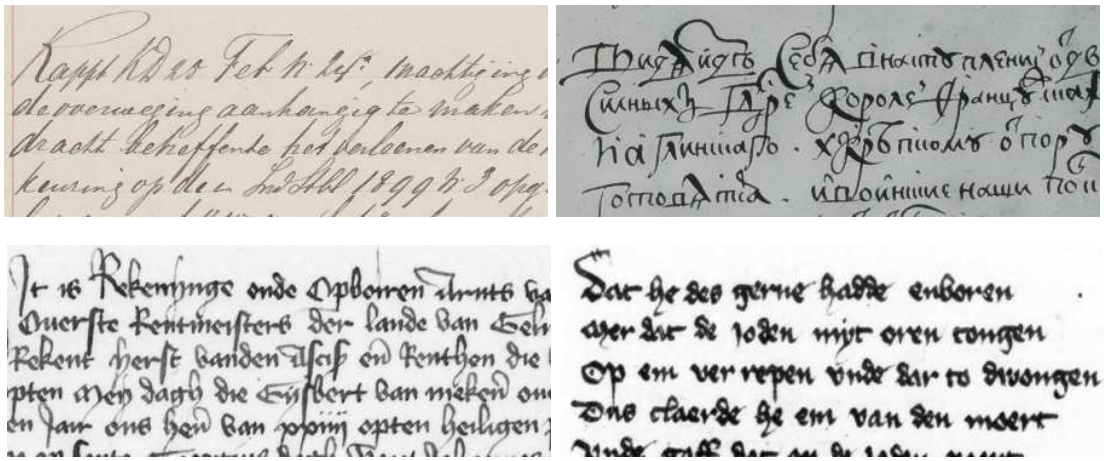
(a) Document image (b) Otsu's algorithm (c) Sauvola's algorithm

Figure 12: Results of the document image after using a binarization technique. (a) The original handwritten document image, (b) background noise is removed by Otsu's algorithm, and (c) the result of Sauvola's algorithm.

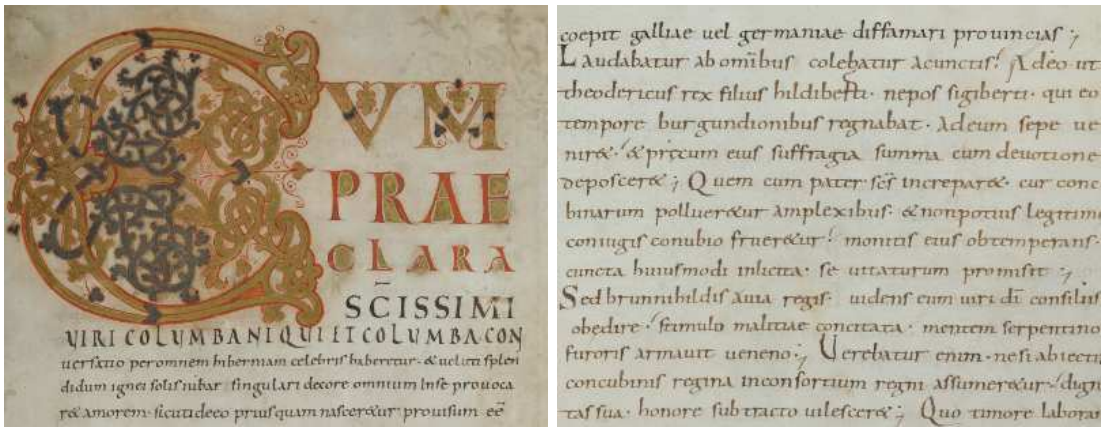
The experimental evaluation of the proposed method is done on two handwritten historical manuscript datasets, namely the MLS and the Saint Gall datasets (Fig. 13). The MLS dataset contains medieval, historical, and contemporary manuscripts, and has the purpose of testing line segmentation algorithms. The collection contains a wide variation of the common problems in handwriting recognition: lines with overlapping ascenders/descenders, slightly rotated scans, and curved base lines. The Saint Gall dataset is written in the 9^{th} century, uses Latin script and contains 60 pages. Each page is written in one column and some pages contain a graphic (Fischer et al., 2011; Garz et al., 2012).

In this research, we have used Sauvola's algorithm with a window size of 20×20 pixels (Fischer et al., 2011) to convert the typically 300 dpi gray document image to a binary document image, the result of which is shown in Fig. 12c. The structure of the characters is legible (Badekas and Papamarkos, 2007) when zooming into the pixel level, see Fig. 14. Unfortunately, Sauvola's algorithm was not able to delete all non-text graphics. For this reason, in case it failed, we manually removed the graphics. We will research automatic removal of all graphics and other issues related to layout analysis in future work.

The second step uses the concept of projection profile analysis (Ghosh et al., 2011) for finding the location of the text lines in the handwritten document image. The horizontal ink density histogram of the document image is computed by taking the sum of the black pixel values in the corresponding row, and then storing the values into a vector. Subsequently, the maxima are extracted from the vector. In our case, we consider the starting points for line segmentation to fall in between the local maxima of the ink density histogram. For dealing with noise, we make use of a *persistance* threshold, which allows to avoid false local maxima. Therefore, we only consider local maxima if the ink density histogram value at some local maximum is larger than $(\mu_h - \sigma_h)$, where μ_h is the average of the ink density histogram and σ_h is the standard deviation. Then these located local maxima represent the text lines, and the starting and end points for the subsequent line segmentation stage are set in the middle between two subsequent local maxima. In



(a) MLS dataset



(b) Saint Gall dataset

Figure 13: A variety of handwritten historical manuscript samples. (a) Four samples from the MLS dataset. (b) Two samples from the Saint Gall dataset, which is one dataset in the IAM historical document database (IAM-HistDB). Please note the problem of vertical overlap, especially in Figure (a) up-right.

this research, the starting state and the goal state is placed on the left and right of each line found with horizontal profile, respectively.

Figure 14: Result of Sauvola's algorithm when zooming to the pixel level.

2.2.2 The A^* Path-Planning Algorithm for Text Line Segmentation

Path planning has been applied to different applications in artificial intelligence, such as in robotic systems, route planners, and games. The aim of path planning is to compute the shortest path that allows the agent to reach its destination given its current position. Path-planning methods compute a path for a given environmental representation in the form of a map (Stentz, 1994). The environmental map can contain many obstacles, which the agent is not allowed to pass through.

The goal of the A^* path-planning algorithm is to minimize the sum of costs on the path between the starting state s_1 and the goal-state s_n. If we denote $s_1^a, s_2^a, \ldots s_n^a$ as the sequence of states traversed by path p^a, then the goal is to compute the optimal path p^* with the lowest total traveling cost:

$$p^* = \arg\min_{p^a} \sum_{i=1}^{n^a-1} C(s_i^a, s_{i+1}^a) \tag{1}$$

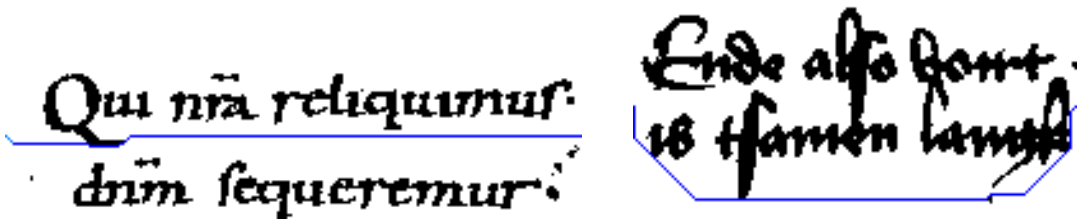
where $C(s_i, s_j)$ is the cost to go from state s_i to state s_j.

The A^* path-planning algorithm uses some heuristic function to speed up computing the optimal solution to reach the goal state. The algorithm combines the cost of the current path between the starting state and the current state with an admissible heuristic function that estimates the shortest possible cost from the current state to the goal state. The heuristic function uses the Euclidean distance to compute a lower bound on the expected cost to travel to the goal state from the current state.

The Problem with Unreachable Goal States

The standard A^* path-planning algorithm cannot address the problem of an unreachable goal state. For example, it cannot find a goal state when it is enclosed from all sides by obstacles. This is because in the standard algorithm the agent is not allowed to move through obstacles and then in this case it can never compute a path to reach the goal state.

We are interested in line segmentation of handwritten historical manuscripts. In our problem, black ink pixels are converted to obstacles and the start and goal states are determined based on the text line localization method explained in Section 2.2.1. When using the original A* path-planning algorithm for this problem, it is effective when the components of two lines do not overlap. On the other hand, the output of the algorithm is incorrect when the handwritten text of two subsequent lines is overlapping. Some results of the standard A^* path-planning algorithm are shown in Fig. 15a and Fig. 15b. The result in Fig. 15a is good and shows that for easy cases the standard algorithm can separate the lines well with an almost straight line. The result in Fig. 15b clearly shows the problem of the standard A* algorithm when the text parts of two lines are overlapping.



(a) (b)

Figure 15: Illustrations of the standard A^* path-planning algorithm. (a) The agent correctly separates two character lines. (b) The agent cannot divide the two touching text lines, because it cannot move through obstacles.

Our proposed A^* path-planning algorithm operates quite differently from the standard method. Most importantly, our A^* algorithm allows the agent to pass through obstacles. However, going through them incurs some cost and therefore it is better for the agent not to traverse these obstacles, if pos-

sible. Consequently, the problem of the touching text lines is solved at the algorithm level.

Cost Functions in our A Path-Planning Algorithm*

Our A* path-planning algorithm uses five cost functions. These cost functions are combined to compute the traveling cost from a state until the goal state is reached. We will now explain the used cost functions in our path-planning algorithm.

1. ***The Ink Distance Cost Functions*** $D(n)$ ***and*** $D'(n)$

   These two cost functions control the agent to stay more or less in between ink pixels above and below. They use the closest distance d of the agent to possible obstacles in the upward and downward directions. The ink distance cost function $D(n)$ for traveling through state n is:

$$D(n) = \frac{1}{1 + \min\left(d\left(n, n_{y_u}\right), d\left(n, n_{y_d}\right)\right)} \tag{2}$$

   where $d(n, n_{y_u})$ and $d(n, n_{y_d})$ are the distances between the state n and the closest obstacle (ink pixel) in the upward and downward direction, respectively. The distance is set to a maximum value if no obstacle is found in that direction. Note that the cost is highest (with a value of 1) if the agent passes an obstacle. Fig. 16 illustrates this cost function. We also make use of $D'(n)$ which attributes a much higher cost to getting close to pixel values compared to staying further away from the black pixels. $D'(n)$ is defined as follows:

$$D'(n) = \frac{1}{1 + \min\left(d\left(n, n_{y_u}\right), d\left(n, n_{y_d}\right)\right)^2} \tag{3}$$

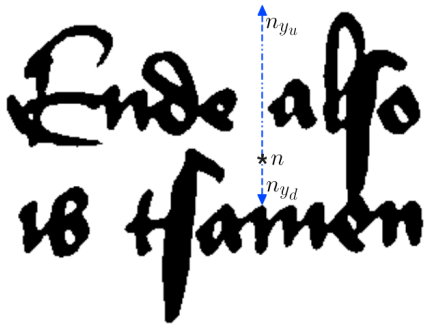


Figure 16: Illustration of the ink distance cost function. The minimum cost is computed from the distance between state n and the closest obstacles in the upward direction y_u and the downward direction y_d.

2. ***The Map-Obstacle Cost Function*** $M(n)$

   To enforce the agent from trying not to traverse obstacles in the map so that it does not cut through black ink, we created another cost function that gives a penalty only if the agent passes through an obstacle. In this cost function $M(n)$ returns 1, if the state n lies on a black pixel, and otherwise $M(n)$ returns 0.

3. ***The Vertical Cost Function*** $V(n)$

   The vertical cost function $V(n)$ is used so that the path does not deviate too much from the y-position of the starting and end position. This prevents the agent from going up or down an entire line. The cost function $V(n)$ is defined as:

$$V(n) = abs(n_y - n_y^{start})$$

   where n_y is the y-position of the current state and n_y^{start} is the y-position of the starting point.

4. ***The Neighbor Cost Function*** $N(s_i, s_j)$

   The neighbor cost $N(s_i, s_j)$ is used to compute the shortest path between the start and goal state, and is the same as also used in the

standard A* path-planning algorithm. When making a move to a new state, the cost $N(s_i, s_j)$ is 10 for a vertical, and horizontal step and the cost is 14 for diagonal directions. In each state the agent can make use of 8-directional movements.

The proposed A^* path-planning algorithm now uses the following combined cost-function $C(s_i, s_j)$:

$$\begin{aligned} C(s_i, s_j) &= c_d D(s_i) + c_{d2} D'(s_i) + c_m M(s_i) \\ &\quad + c_v V(s_i) + c_n N(s_i, s_j) \end{aligned} \tag{4}$$

The parameters c_d, c_{d2}, c_m, c_v, and c_n have been tuned using some images during preliminary experiments. This cost function is used to compute the path cost. After each move the state with the lowest value is used to expand its current path. Once the goal-state is to be expanded, the optimal path based on the used cost functions has been found.

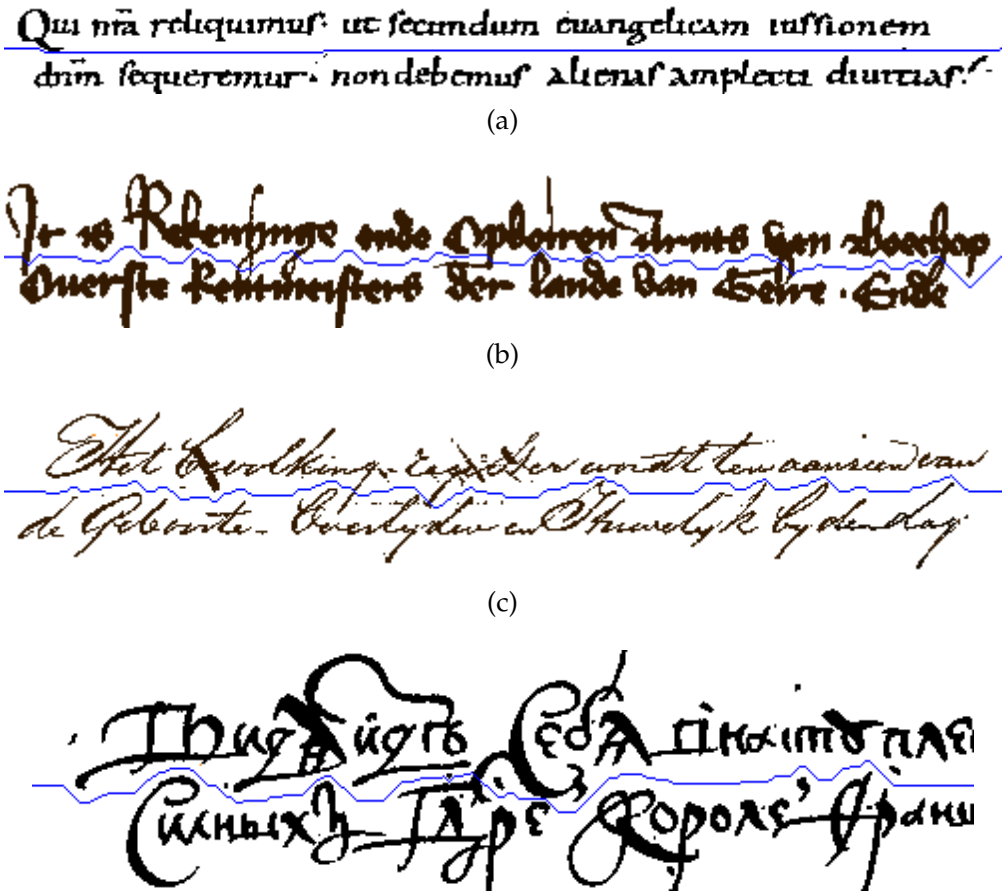
2.2.3 Experimental Evaluation

The whole algorithm now works as follows. In the first step the handwritten historical manuscripts are binarized using Sauvola's algorithm. In the second step the smoothed horizontal ink density histogram of the binary image is calculated. Then peaks of the horizontal ink density histogram are detected by the local maxima method. The starting state of each line is set between subsequent peaks. Finally, the A^* path-planning algorithm is applied.

Our line segmentation system has been applied to handwritten historical manuscripts from the MLS and the Saint Gall dataset. The values we used for the experiments on the Saint Gall dataset are: $c_d = 150$, $c_{d2} = 50$, $c_m = 50$, $c_v = 3$, and $c_n = 1$. The values we used for the experiments on the heterogeneous MLS dataset are: $c_d = 130$, $c_{d2} = 0$, $c_m = 50$, $c_v = 2.5$, and $c_n = 1$.

Some results of the A^* path-planning algorithm are shown in Fig. 17. The text lines are separated by the optimal path (i.e., the path with the

lowest cost). Some line segmentation results on whole document parts of our method are shown in Fig. 18 and Fig. 19.



(a)

(b)

(c)

(d)

Figure 17: Some results of our A* path-planning algorithm.

The ground-truth line segmentation for these datasets is acquired manually with the help of a tool developed for this task. This tool presents a scanned document in a web page, and allows the human user to mark handwritten text with the mouse pointer by selecting a vertical area (denoted by two y-values in the image) using JavaScript (see Fig. 20). After annotating these text lines, the page image is split as follows: for each line annotation the area above and below the line area are whitened in a copy of

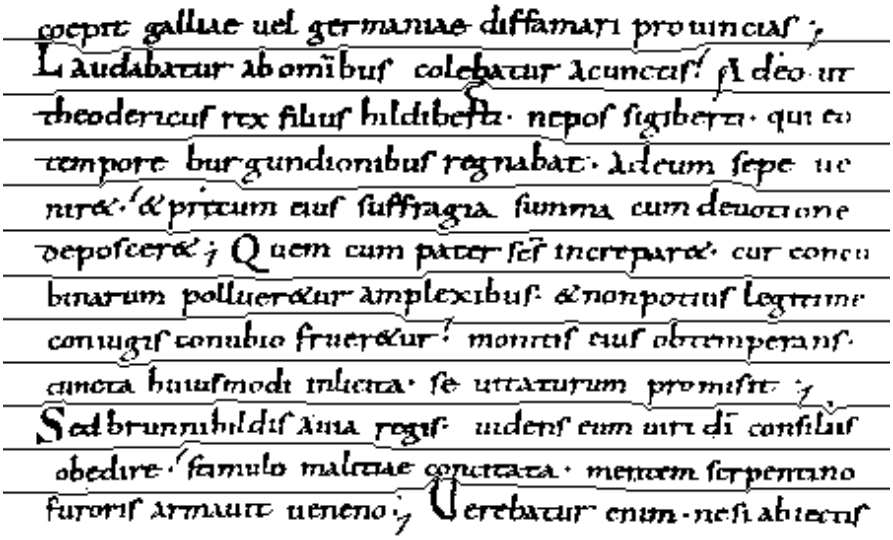


Figure 18: Line segmentation result on handwritten documents from the Saint Gall dataset.

the original image. This preserves the original image size. Descenders and ascenders from respectively the lines above and below are not removed by this process; they are removed manually by whitening them with a simple paint program (e.g., xpaint). This procedure yields the ground truth for the target line.

Both the input images and output images are losslessly compressed to prevent any influence from lossy compression artefacts. Each line annotation image is named as the original image, appended with a line number. This allows the matching of the ground-truth images with the output images of our A^* path-planning algorithm.

For evaluating the performance of the line segmentation algorithm, we will use the pixel-level hit rate and the line detection accuracy measure on the binary document image, as proposed by Li et al. (2008) and Garz et al. (2012).

After the line segmentation algorithm is finished, we will have M ground-truth lines and N detected lines. Now a matrix P of size $M \times N$ is computed, where P_{ij} is the number of shared black pixels between the i^{th} ground-truth

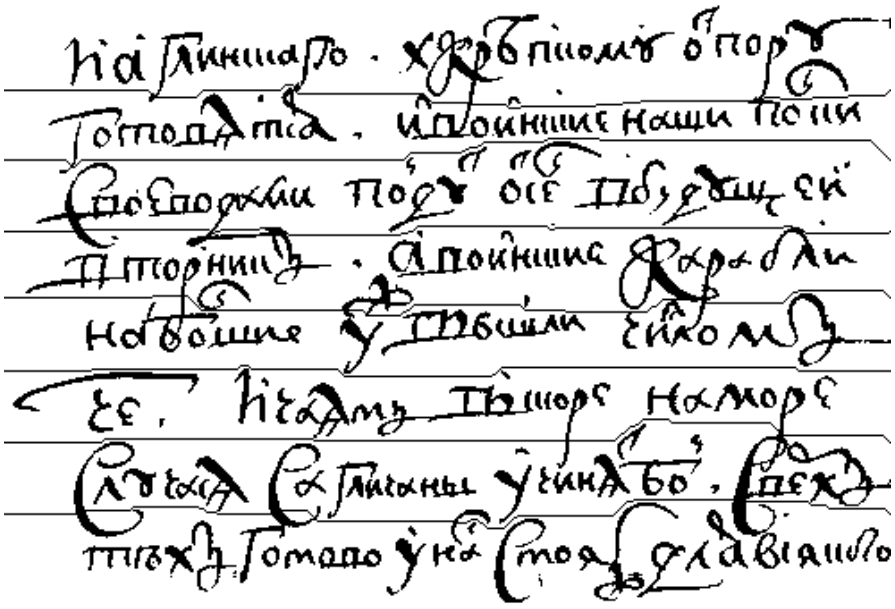


Figure 19: Line segmentation result on handwritten documents from the MLS dataset.

line and the j^{th} detected line (Garz et al., 2012; Baechler et al., 2013). The goodness of assigning particular ground-truth lines to particular detected lines is given by the total number of shared black pixels given this assignment. From all the possible assignments the optimal one is selected, resulting in the value $G(S_{max})$ indicating the total number of shared pixels for the optimal assignment S_{max}. The pixel-level hit rate is now defined as follows:

$$Hr = \frac{G(S_{max})}{|GT \cup R|} \tag{5}$$

where GT is the set of black pixels in the ground-truth line, and R is the total number of black pixels found by our algorithm including pixels which are not in the ground-truth.

We also use the text-line detection accuracy measure (Li et al., 2008; Baechler et al., 2013). A line i is correctly detected if $\frac{G_{ij}(S_{max})}{|GT_i|} \geqslant 0.9$ and

$\frac{G_{ij}(S_{max})}{|R_j|} \geqslant 0.9$. where GT_i is the set of black pixels in the ground-truth line i, R_j is the set of black pixels in the matching j-th line found by our algorithm and G_{ij} is the number of shared black pixels between line i of the ground-truth and the j-th detected line. This measure is affected by missing parts of the text-line and additional noise in the detected line.

We computed results on the Saint Gall dataset using a manuscript containing a total of 1,429 lines and on different manuscripts from the MLS dataset containing in total 995 lines. For the Saint Gall dataset, our method obtains a near-perfect pixel-level hit rate of 0.998 and a line-detection accuracy of 0.999. The computation time for the 1,429 lines is around 8 minutes. On the much more complicated MLS dataset our method obtains a pixel-level hit rate of 0.928 and a line-detection accuracy of 0.9. The computation time for this dataset for the 995 lines is around 26 minutes. Most errors on the MLS dataset are caused by our method for detecting starting points of lines, because this dataset contains some very short lines, with which our line detection algorithm cannot cope well. Other errors on the MLS dataset are caused by lines which are not horizontal, but slanted.

Table 1 presents the hit rate and the line accuracy for the Saint Gall dataset for several methods. In this table the performance of our method is compared with two systems including the method by Garz et al. (2012) and the method by Baechler et al. (2013). The results show that our method outperforms previous methods on this dataset.

Table 1: Line accuracy and hit rate of line segmentation on the Saint Gall dataset.

Method	Hit rate	Line accuracy
Baechler *et al.*	0.9600	0.9540
Garz *et al.*	0.9865	0.9797
Our method	0.9980	0.9990

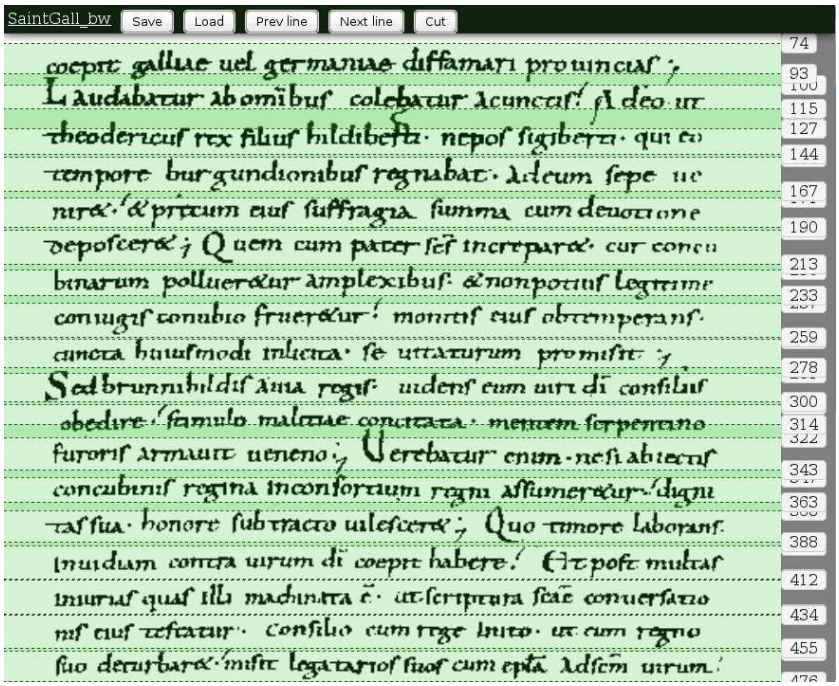


Figure 20: Illustration of the ground-truth tool. The text line area is marked by the human user.

2.3 Conclusion

We have presented the important process of the document layout analysis that segments the historical manuscript into a set of regions of interest and recognizes them as the textual information.

In this chapter, we focused specifically on line segmentation. An A^* path-planning algorithm is described that uses a combination of cost functions to control a line segmentation agent. The algorithm uses a binarized image obtained from a handwritten text image using Sauvola's algorithm. The start and end points of a line are detected using the smoothed horizontal ink density histogram. The algorithm uses different cost functions to stay as far away as possible from ink pixels and at the same time tries to compute the shortest path. The travel costs are always positive and increase until the

end of the line. Dynamic programming-like approach is used to efficiently compute the travel cost. The results on two historical line-segmentation datasets show that our A* path-planning algorithm successfully separates subsequent text lines, even when they partially overlap. The advantages of the proposed method are that our method is fairly simple to implement, quite fast, and robust for different kinds of handwritten documents. The disadvantage is that sometimes the method prefers to cut some text instead of going up with a curved line. This is a consequence of the trade-off between staying away from ink pixels and computing the shortest path. As stated in the introduction a synergy between bottom-up and top-down processing using text classification likelihoods can solve these problems in the future. Also Sauvola's algorithm needs to be applied iteratively using several window sizes in a real system.

Future Work

Layout analysis (Bulacu et al., 2007; Baechler et al., 2013) will be used to handle the document image before applying the line segmentation technique. We will use an object detection technique to detect non-text graphics in the handwritten images. Furthermore, we plan to use an energy function (Garz et al., 2012) that can be computed on gray images instead of the black and white images we used in this chapter. Given the current system, it would be interesting to see if the parameters can be automatically optimized using an adaptive framework. We will apply the A* path-planning algorithm to a handwritten Thai dataset, which we recently collected, and combine this with handwritten character recognition algorithms. Finally, we would like to compare our A* path-planning algorithm with seam carving technique (Saabni and El-Sana, 2011; Arvanitopoulos and Susstrunk, 2014; Zhang and Tan, 2014).

3 PIXEL-BASED AND FEATURE-BASED RECOGNITION

Feature extraction techniques can be important in character recognition, because they can enhance the efficacy of recognition in comparison to featureless or pixel-based approaches. Our *first* study in this chapter aims to investigate the feature extraction technique called the hotspot technique in order to use it for representing handwritten characters and digits. In the hotspot technique, the distance values between the closest black pixels and the hotspots in each direction are used as representation for a character. The hotspot technique is applied to three datasets including Thai handwritten characters (65 classes), Bangla digit (10 classes), and MNIST (10 classes). The hotspot technique consists of two parameters including the number of hotspots and the number of chain code directions. The datasets are then classified by the *k*-Nearest Neighbors algorithm using the Euclidean distance as function for computing distances between data points. In this study, the classification rates obtained from the hotspot, mask direction, and direction of chain code techniques are compared. The results revealed that the hotspot technique provides the largest average classification rates.

The *Second* study proposes another novel handwritten character recognition method for isolated handwritten characters. A feature is introduced

The research described in this chapter has previously been published in *Proceedings of the International Conference on Pattern Recognition Applications and Methods (ICPRAM2014)* and *Proceedings of the International Conference on Document Analysis and Recognition (ICDAR2013)*

for such patterns, the contour angular technique. It is compared to other methods, such as the hotspot feature, the gray-level normalized character image, and a basic low-resolution pixel-based method. One of the goals of this study is to explore performance differences between dedicated feature methods and the pixel-based methods. The four methods are combined with support vector machine classifiers and compared on a collection of handwritten Bangla digit images. The results show that the fast contour angular technique outperforms the other techniques when not very many training examples are used. The fast contour angular technique captures aspects of curvature of the handwritten image and results in much faster character classification than the gray pixel-based method. Still, this feature obtains a similar recognition compared to the gray pixel-based method when a large training set is used. In order to investigate further whether the different feature methods represent complementary aspects of shape, the effect of majority voting is explored. The results indicate that the majority voting method achieves the best recognition performance on this dataset.

3.1 Introduction

Feature extraction can play an important role in handwriting recognition. It is used for generating suitable feature vectors, and using them as representation of handwritten characters. The difference between handwritten characters and printed characters lies in the diversity of characters. In printed characters, the structural pattern of characters is always the same, therefore the main challenges are coping with different fonts, or scanning qualities. However, in handwritten characters, the pattern of characters is different, even for those of the same writer.

The main objective for using feature extraction is to reduce the data dimensionality by extracting the most important features from character images (Lauer et al., 2007). When the feature vector dimensionality is smaller, a set of features can be useful for representing the characteristics of characters. Moreover, feature extraction can play a significant factor for obtaining high accuracies in character recognition systems (Trier et al., 1996), especially if there is not a lot of training data available.

The main challenge in handwritten character classification is to deal with the enormous variety of handwriting styles as a result of different writers. Furthermore, some complex handwriting scripts comprise different styles for writing words. In some of them characters are written isolated from each other, (e.g., Thai, Laos, and Japanese), in some they are cursive, and in some the characters are connected (e.g., English, Indian, and Arabic). This challenge is recognized by many researchers (Cireşan et al., 2010; Meier et al., 2011; Song et al., 2011). The large variety of writing styles, writing persons, and the complicated features of the handwritten characters are very challenging for accurately classifying the characters.

A large number of studies investigated the problem of handwriting recognition based on the MNIST dataset (LeCun et al., 1998). The MNIST dataset was modified from the original NIST database. This dataset is nowadays used as a standard benchmark for testing machine learning techniques and pattern recognition methods (Cireşan et al., 2010; Meier et al., 2011). There are 60,000 handwritten digit images for training and 10,000 test images. The size of these handwritten digit images is normalized and the digits are cen-

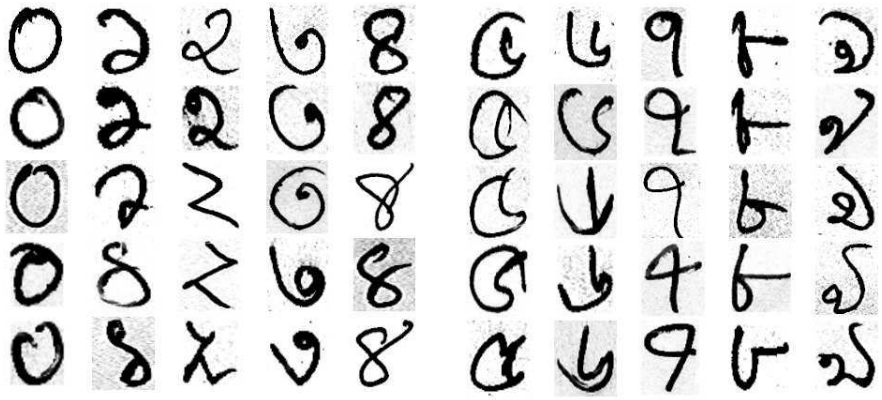
tered in a fixed-size image to fit into a 28 × 28 pixel space (LeCun et al., 1998; Meier et al., 2011). Furthermore, the handwritten images are completely separated from the background. Although this is a large dataset, it is very clear and researched extensively.

This chapter *first* of all aims to investigate a novel feature extractor for handwritten characters and other types of characters such as handwritten digits from different scripts. The main aim of this chapter is to propose a fast and easy to use feature extraction technique that obtains a good performance. This study focuses on isolated characters. Three datasets including Thai character, Bangla digit, and MNIST (Mixed National Institute of Standards and Technology database) were used to test our novel proposed feature extraction technique. The hotspot technique is used to determine the distance along a particular direction between the hotspots and the first black pixel of the object. The hotspots are distributed at fixed locations over the character images. This technique extracts some important information from the character images and is fairly robust to translation and rotation variances. The important parameters of this technique are the number of hotspots and the number of chain code directions.

Second, we focus on the recognition of handwritten *Bangla* (or Bengali) digits, which is the second most popular language in India and Bangladesh (Das et al., 2010). The dataset contains different kinds of background and a variety of pixel space resolutions. Various example digits of handwritten Bangla are shown in Fig. 21. It is clear, that when compared to the MNIST dataset, Bangla digits are more complicated and there is more style diversity (Bhowmik et al., 2009). For instance, the curly tails in Bangla characters makes the definition of a stable bounding box problematic.

Related Work

A feature extraction technique can play an important factor for getting high accuracies in handwritten character recognition systems (Surinta et al., 2012). Different feature extraction techniques have been used to train a recognizer for the handwritten Bangla digit dataset. However, also pixel-based methods that directly use the pixels have been explored. In one approach,



(a) Bangla digit 0-4 (b) Bangla digit 5-9

Figure 21: A variety of handwritten Bangla digit samples. (a) Set of numbers from 0 to 4 and (b) from 5 to 9. Note the large differences between the examples on the first and last rows and the variation in background intensity.

a pixel-based method, pixels are represented as data in high dimensional input space. Wen et al. (2007) and Roy et al. (2004) presented the original handwritten Bangla digit without computing any features and normalized the original image into a 16 $\times$ 16 and 28 $\times$ 28 pixel space, respectively. The number of handwritten digits used in their experiments were 16,000 and 10,677 records, respectively. Wen et al. (2007) used the support vector machine (SVM) classifier and obtained a recognition rate of 86.1%. Roy et al. (2004) used the multilayer perceptron (MLP) for the recognition and obtained a recognition rate of 92.1%

In the feature-based approach, an automatic feature extraction technique extracts unique information from the handwritten image. The number of resulting features from the feature extraction technique is often smaller than in the pixel-based method. Sanossian and Evans (1998) proposed a scanning technique for English characters. They used 64 $\times$ 64 pixels of binary images. The feature values are calculated by scanning through the image in horizontal, vertical, and inner (inside the character) directions of character images. Ferdinando (2003) used a vertical and two horizontal directions for

digits. The feature vectors from this technique are the positions of crossing points between each line.

An interesting approach is a direction technique consisting of 4 feature windows, and 4 neighbor masks. Firstly, Kawtrakul and Waewsawangwong (2000) used 4 feature windows including horizontal, vertical, left diagonal, and right diagonal directions. This technique found the contour of Thai character images. Subsequently, every feature window (3×3 pixels) is used to slide through all cells of character images. The feature vectors of this technique represent the number of perfectly matching windows in the feature window with the part of the character windows. Secondly, Pal et al. (2008) presented 4 direction neighbors. These directions are used to count a matching number of directions from contour images. Rajashekararadhya and Ranjan (2009) suggested the application of a feature extraction algorithm for Kannada script. In this study, the zone and projection distance metric technique was proposed. The distance values from four different directions including vertical downward direction, vertical upward direction, horizontal right direction, and horizontal left direction were calculated.

In addition, Wen et al. (2007) proposed the KPS technique. Their technique combined a Kirsch mask and principal component analysis (PCA).

The dimensionality reduction was used, because the Kirsch mask extracted 1,280 dimensional inputs. After PCA they used the SVM classifier and obtained an accuracy of 95.1% on the Bangla digit dataset. Basu et al. (2005) presented the shadow, the centroid and the longest-run feature extraction technique. It provided 76 dimensional inputs and 6,000 records were used, which were scaled to 32×32 pixel space. They used the MLP classifier and obtained an accuracy of 96.7% on the Bangla digit dataset.

Contributions

In this chapter we propose a novel method for feature extraction which is suitable for isolated handwritten characters called the hotspot technique. We have used three different handwritten datasets from different scripts (Thai character, Bangla digit and MNIST) to compare our novel feature ex-

traction technique to two state-of-the-art techniques. The results show that our novel method significantly outperforms the other two methods on two datasets containing handwritten digits. Only on the Thai character dataset containing 65 classes, one other technique achieves higher recognition accuracies. The average recognition rate over the three datasets is also highest for our novel technique, which demonstrates its effectiveness.

Second, we present a novel method that obtains state-of-the-art performance on the isolated handwritten Bangla digit dataset. We have compared four different techniques. The first feature is the contour angular technique (CAT) that computes the contour of the handwritten image using 8-directional codes, while counting the co-occurrences of angles along the ink trace. The second feature is the set of distance values that is computed between the hotspots and the black pixels of the handwritten image (Surinta et al., 2012). The third feature uses intensities of the pixel space from small blocks. The last feature is a gray pixel-based method that uses the whole handwritten image as the input (Das et al., 2010). Finally, we use a majority voting technique to combine the outputs of the four different classifiers and to obtain the highest recognition accuracy.

In order to create a feature vector, we study two kinds of techniques. In both, the relevant information is extracted from the handwritten character image and transformed into vector data (Dhandra et al., 2011). The two major techniques include image pixel-based methods and feature extraction techniques. These techniques are described in the following sections.

3.2 Image Pixel-Based Methods

3.2.1 Gray Pixel-Based Method (GPB)

The Gray pixel-based method uses the raw pixel intensities of the handwritten images to preserve the handwritten image without destructing subtle intensity gradients on the edges of the inktrace (Roy et al., 2004). The size

of the handwritten image is resized to the corresponding resolution, 784 (28 × 28) and 1,600 (40 × 40) feature values are computed.

3.2.2 Black and White Down Scaled Method (BWS)

This simple feature extraction technique is useful to compute a base performance. The black and white handwritten image is partitioned into 9 × 9 non-overlapping blocks. From each block the number of black pixels is computed, resulting in 81 features.

3.3 Feature Extraction Techniques

Feature extraction can play a significant factor for increasing the efficacy of recognition systems (Trier et al., 1996). It is a process that extracts the important information from the character images and transforms them into vector data. When a feature extraction technique is applied, the dimensionality of the resulting feature vector is often smaller in comparison with that of raw data (Lauer et al., 2007). Since the smaller feature vectors are afterwards used in a classification algorithm, with little training data they may suffer less from overfitting than pixel-based methods.

3.3.1 The Hotspot Technique (HOT)

The HOT is our first novel feature extraction method useful for representing the characters. The distance between black pixels and the hotspots in each direction is used to describe the whole object. In this technique, the size of the hotspot was defined as N × N. For example, the size of the hotspot can be 3 × 3. The distance between black pixels and the hotspots from the first to the last hotspot is calculated. The direction of the hotspots is defined by the chain code directions as shown in Fig. 22.

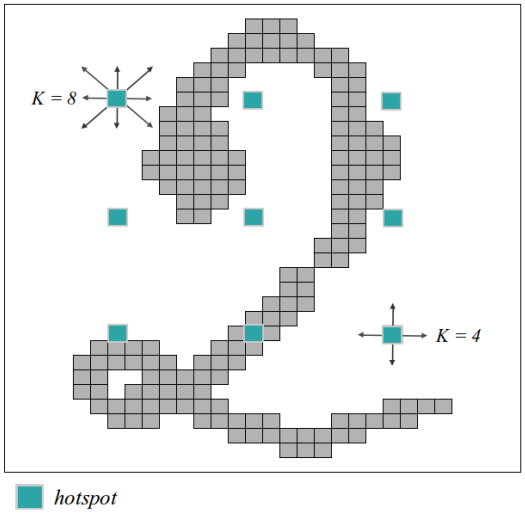


Figure 22: An example to illustrate the location and distribution of the hotspots. In this image, the use of 3×3 hotspots are shown. The chain code directions for identifying the distance, starting from the hotspot until the object in each direction was found. The directions from each hotspot are identified by using 4 directions and 8 directions chain code.

The hotspot feature vector P_s is defined as Equation (6);

$$P_s = \{(x_s, y_s), \{d_i\}, \{D_{si}\}\} \tag{6}$$

where (x_s, y_s) is the coordinate of the hotspot $d_i \epsilon \{0, 1, 2, ..., 7\}$ when the chain code direction is considered as 8-directional codes (see Fig. 27), and D_{si} is the distance between the hotspot and the first black pixel of the object found in the direction d_i. It is noted that, if there is no object pixel found then the distance D_{si} is set to d_{max}. The distance is measured by using the following equation (Eq. (7));

$$D_{si} = \begin{cases} \sqrt{(x_s - x_i)^2 + (y_s - y_i)^2} & \text{if } (x_i, y_i) \text{ exists,} \\ d_{max} & \text{else} \end{cases} \tag{7}$$

where (x_i, y_i) is the coordinate of the closest black pixel of the object in the specified direction. As feature vector we only consider 4 or 8 values of every hotspot and then concatenate it into some specific order to create a feature vector. The complete notation of feature vectors can be defined as follows;

$$f = \{D_{11}, \ldots, D_{1K}, \ldots, D_{L1}, \ldots, D_{LK}\}$$

where L is the number of hotspots and K is the number of directions. The feature vector size depends on the number of hotspots and the number of directions. For example, when 9 (3×3) hotspots and chain codes with 4 directions are used, the size of the resulting feature vector will be 36.

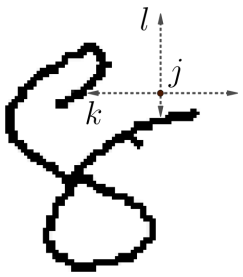


Figure 23: The feature values of the hotspot when considering the 4-directional codes. j is the coordinate of the hotspot (x_s, y_s), k is a distance value D_{si} when (x_i, y_i) exists and l is a distance value d_{max} when D_{si} does not exist.

A problem that we noticed when the 8 directions technique was applied is that some black pixels might not be determined, especially in diagonal directions. For example, the problem is that the hotspot goes through the black pixels but cannot find them. Therefore, the distance from that hotspot was determined as d_{max} (see Fig. 23). In Fig. 24 when the black pixel cannot be identified, the hotspot should be stopped at the distance with a value of 4 ($d_3 = 4$). In the present study, we use the mask conditional technique to solve this problem. When the mask condition technique was applied, all the black pixels in the contour of the character were found.

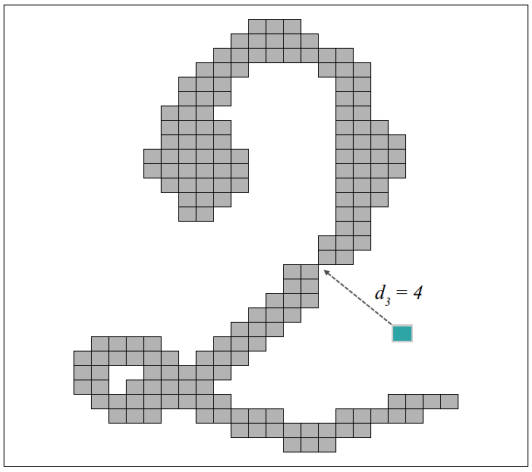
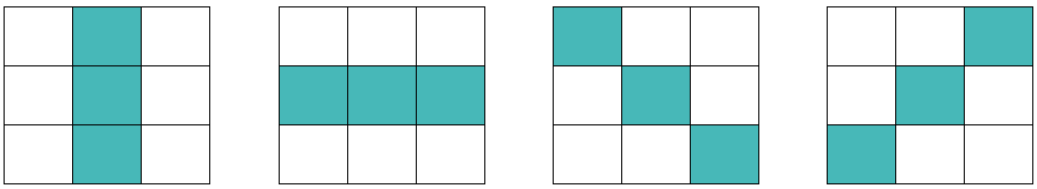


Figure 24: Illustration of the problem with 8 directions chain code. In this image, the black pixel could not be found in the left-up diagonal direction d_3. The mask conditional technique is used to solve the problem. As a result, the distance is defined as 4.

There are two parameters that influence this method including 1) number of hotspots and 2) number of chain code directions. The preliminary results demonstrated that the best setting uses 25 evenly spaced hotspots. The directions of the hotspots are defined by the 4-directional codes $d_i \in \{0, 2, 4, 6\}$, so that the hotspot technique provides 100 features.

3.3.2 Mask Direction Technique (MDT)

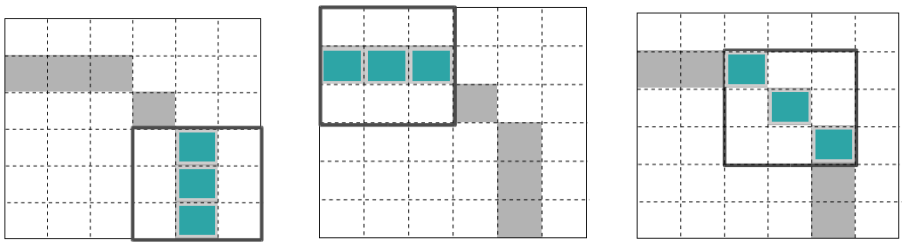
The MDT is also known as the direction feature (Blumenstein et al., 2003) and suitable for tracking the directions of the character image. In this technique, the character image is first peeled off from the original character image until the thin object is found. This is called thinning the character image. The mask size for the mask direction technique is 3×3 pixels (Kawtrakul and Waewsawangwong, 2000). The mask directions consist of vertical, horizontal, left-diagonal, and right-diagonal mask as shown in Fig. 25.



(a) Vertical mask (b) Horizontal mask (c) Left-diagonal mask (d) Right-diagonal mask

Figure 25: Illustration of the four mask directions. 25a vertical mask s_a, 25b horizontal mask s_b, 25c left-diagonal mask s_c and 25d right-diagonal mask s_d.

The MDT starts off by dividing the character image into small blocks. In the present study, each mask direction is used to slide through each pixel in the small block from the coordinate $(1, 1)$ to coordinate $(1, n)$ (from left to right) and search until the last coordinate (n, n) (top-down) is found. For example, some matched mask directions are shown in Fig. 26. The mask direction technique computes a fixed-size histogram containing the number of matched pixel-windows (mask directions) from whole small blocks.



(a) vertical direction (b) horizontal direction (c) diagonal direction

Figure 26: The matched masks in a particular direction 26a vertical direction, 26b horizontal direction and 26c diagonal direction.

Let us consider the set of distinct patterns $S = (s_a, s_b, s_c, s_d)$ where s_a, s_b, s_c, and s_d are the vertical, horizontal, left-diagonal, and right diagonal mask

as shown in Fig. 25 as a feature vector. We consider only the frequency of each pattern S within small blocks. The number of features obtained from mask direction were 64 features. These features were derived from dividing the character image into small blocks (16 blocks in total) and there were 4 mask directions.

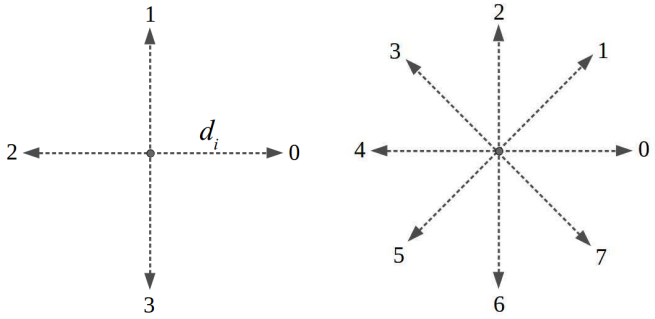
3.3.3 Direction of Chain Code Technique (DCC)

The DCC is an efficient technique in handwriting recognition (Bhowmik et al., 2007; Pal et al., 2008; Siddiqi and Vincent, 2009). We applied this technique to the present study according to the methods described by Siddiqi and Vincent (2009), although we adapted their technique to deal more efficiently with our datasets. They used the extraction technique for handwritten south India scripts, which include Kannada, Telugu, and Tamil. In their study, they use two feature sets, one consisting of 64 and the other of 400 feature values. The first feature set is used for high speed recognition. First, the stroke is extracted from the character image and divided into small blocks ($N \times N$). Consequently, four directions of the chain code are used to count the frequency of the directions from the contour image. The number of features obtained from the direction of chain code technique we used is 128. These features are derived from dividing the character image into small blocks (16 blocks in total) and there are 8 chain code directions.

The DCC is used to describe the directions of characters. Here, the thinned character was used to compute the chain code. The directions that will be computed in a character are shown in Fig. 27. The starting point of the direction in each block will first be identified. Interestingly, a wrong starting point will cause errors of the chain code as well as shown in Fig. 28. The procedures to indicate the pattern of the chain code should be similar.

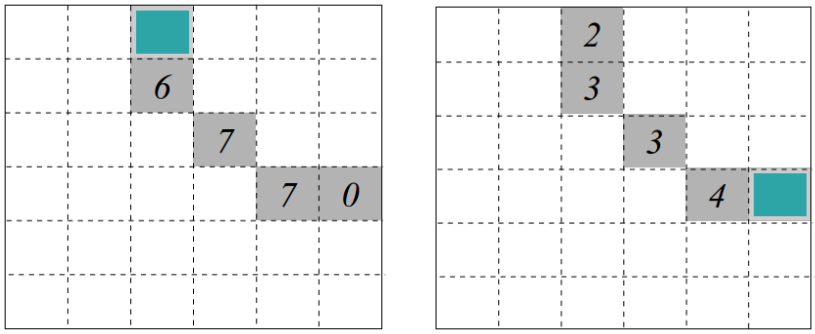
In Fig. 28a the starting point is (1,3), at this point the result of this chain code is $c = \{6, 7, 7, 0\}$, while in Fig. 28b the starting point is changed from (1,3) to (4,6). The chain code from this starting point is $c = \{4, 3, 3, 2\}$. The result is very different when the starting point is changed.

The algorithm we used for computing the directions of chain code representation is the following;



(a) 4 directions (b) 8 directions

Figure 27: Illustration the chain code directions. 27a chain code with 4 directions and 27b 8 directions.



(a) Starting point (1,3) (b) Starting point (4,6)

Figure 28: Illustration of the chain code with different starting points. 28a the starting point starts at coordinate (1,3), the chain code $c = \{6, 7, 7, 0\}$ and 28b coordinate (4,6), the chain code $c = \{4, 3, 3, 2\}$.

1. The character image is divided into small blocks ($N \times N$).
2. The starting point for each block is identified. The starting point is searched from the coordinate (1,1) and the black pixels along x-axis was found through the end of the x-axis $(1, n)$, which is the upper edge of the sub-character image block. Consequently, the image block

is searched through along the borders, so first along the upper edge, the right edge, the bottom edge and the left edge until finished at the last pixel. The first found black pixel is the starting point.

3. From the previous step, the starting point of the object is identified. Here, the 8 directions of chain code is used for identifying the direction of the neighbor pixel. The neighbor pixel cannot always be found by using the 4 directions procedure because the 4 directions of chain code do not consider the diagonal directions. After the direction of the object from the first starting point is found, the identified neighbor pixel is marked in order to avoid repeated identification. For identifying the next starting point, it is needed to be started off with the same starting point from the beginning. During this identification, when the marked pixels are found, the searching for the starting point is skipped.

4. The neighbor pixel of the object needs to be identified. The neighbor pixel is identified until the last one.

3.3.4 The Contour Angular Technique (CAT)

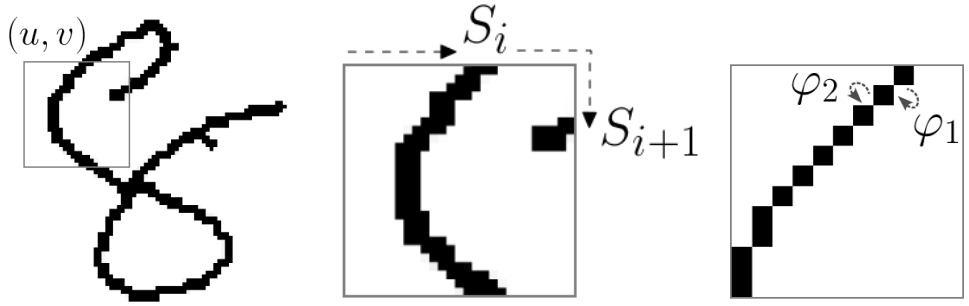
What is needed is a feature that can be computed conveniently and which captures more shape details than the angle along the contour of the ink trace. A proposed solution is to capture aspects of curvature. The approach is related to the usage of angular co-occurrence in the Hinge feature (Bulacu and Schomaker, 2007) in writer identification.

The CAT implementation is, unlike the Hinge, directed at the classification of characters and is a fast implementation of quantized angle co-occurrence computation. The technique consists of two stages.

The first stage

The method divides the handwritten character into 16 non-overlapping blocks and considers the contour of the handwritten image as 8-directional codes, see Fig. 27. This setting computes 128 features. *First*, the starting

point S_i for each block is identified. It is searched along the edges of image blocks, first along the upper, the right, the bottom until finished at the left edge. The first found black pixel is the starting point S_i (Fig. 29b). *Second*, 8-directional codes are used for identifying the contour of the neighbor pixels. Sometimes there are multiple neighbor pixels, for this reason a queue is used to arrange the neighbor pixels, which are all used to update the directional-code histogram.



(a) The handwritten image block (b) The starting point (c) The angular co-occurrence

Figure 29: The starting point S_i of the handwritten image block. (a) The handwritten image block (u, v) from the whole handwritten image. (b) The starting point S_i of the handwritten image is searched along the (u, v) borders. (c) The angular co-occurrence for which angles φ_1 and φ_2 are computed. Note that the rectangular block (u, v) shown in Fig. 29a is not shown in the appropriate location and size.

The second stage

The contour of the handwritten image is computed. According to the chain code directions as shown in Fig 27b, 8 directions of the chain code are used. The method considers the angles φ_1 and φ_2 (see Fig. 29c) that move from S_i until the last pixel in each block and then counts the co-occurring angles in a two-dimensional array indexed by φ_1 and φ_2. The result is the angular co-occurrence histogram with 64 (8×8 directions) elements.

The discrete angle co-occurrence histogram approximates the angular co-occurrence probability along the contours. By combining the outputs of both stages, the CAT feature extraction technique creates feature vectors of size 192.

3.4 Classification Methods

3.4.1 *k*-Nearest Neighbors (*k*NN)

The *k*NN algorithm is classified as an instance based learning algorithm (Martínez and Fuentes, 2003), which is suitable for large amounts of data. It is a well known non-parametric and simple algorithm. The *k*NN algorithm has been used in statistical estimation, scene recognition (Abdullah et al., 2010b) and also writer identification systems (Brink et al., 2012). In some previous studies, the *k*NN algorithm has been used in character recognition and a good recognition performance was obtained (Cordella et al., 2008). In (Kumar et al., 2011), the authors proposed two feature extraction techniques including diagonal and transition feature extraction, and then experimented on the Gurmakhi dataset. The recognition performance with this method is 94.12%. In (Rakesh Rathi and Jangid, 2012), the authors used feature mining algorithms to compute the feature vector and tested the method on the Devanagari vowels database. The KNN algorithm is used as a classification technique, and obtained the accuracy of 96.14%.

In *k*NN, the input vector is compared with training samples to compute the most similar *k* neighbors. The efficacy of the *k*NN algorithm depends on two key factors: a suitable distance function and the value of the parameter k (Hendrickx and van den Bosch, 2005). In this study, the Euclidean distance is selected as the function in order to calculate distance values from an input

vector **x** to each training sample **y**. The Euclidean distance is calculated by the following equation (Eq. (8));

$$d(x,y) = \sqrt{\sum_{i=1}^{N} \left(x^i - y^i\right)^2} \tag{8}$$

where N is the number of dimensions of x and y. Then distances between the input vector and the training samples are compared to identify the closest neighbors to the input vector. The parameter *k* is usually chosen as an odd number, e.g. if parameter $k = 3$, the three closest neighbors are considered in order to determine the class for a particular input vector. Let $Z = \{(y_i, c_i)\}$ be the set of M labelled training samples, where $y_i \in \mathbf{R}^N$ and $c_i \in C$ and C is the set of class labels present in the training samples. In the classification stage for an unknown sample x, first the distance $d(x, y_i)$ from x to each sample in Z is calculated according to Eq. (8). Let $D_k = \{d_1, d_2, ..., d_k\}$ be the set of k nearest distances for the input x, where $d_1 \leqslant d_2 \leqslant ... \leqslant d_k$. To classify an unknown sample x, the number of occurrences each class belongs to the input vectors in D_k is counted, and finally the most frequently occurring class is selected as the output of the classifier.

3.4.2 Support Vector Machine (SVM)

The support vector machine algorithm invented by Vapnik (1998) has been effectively applied to many pattern recognition problems.

Linear SVMs for Two-Class Problems

The SVM algorithm is very useful for two-class classification problems. The algorithm finds the optimal separating hyperplane, which has the maximum distance to the training points that are closest to the hyperplane.

The training points closest to the computed separating hyperplane are called support vectors. The original SVM is a linear binary classifier, which is useful for two-class classification problems. On the other hand, it does

not provide good separation for non-sparse complex data (e.g., image data). We will now shortly describe the workings of the SVM. Let D be a training dataset,

$$D = \{(x_i, y_i), 1 \leqslant i \leqslant M\} \tag{9}$$

where $x_i \in \mathbf{R}^N$ are input vectors and $y_i \in \{+1, -1\}$ is the binary label of pattern x_i. The optimal model from the set of hyperplanes in $\mathbf{R}^N$ is computed by the SVM optimization algorithm. The decision function is given by (Eq. (10)):

$$f(x) = \text{sign}(w^T x + b) \tag{10}$$

where w is the weight vector orthogonal to the hyperplane and b is the bias value. To compute the parameters w and b, the SVM algorithm minimizes the following cost function:

$$J(w, \xi) = \frac{1}{2} w^T w + C \sum_{i=1}^{n} \xi_i \tag{11}$$

subject to constraints:

$$w^T x_i + b \geqslant +1 - \xi_i \text{ for } y_i = +1$$

and

$$w^T x_i + b \leqslant -1 + \xi_i \text{ for } y_i = -1$$

where C controls a trade-off between training error and generalization, and $\xi_i \geqslant 0$ are slack variables which tolerate some errors, but which need to be minimized. While this soft margin method is useful to fit a model to a complex dataset, if used improperly, overfitting can occur.

The maximum margin splits the hyperplane with $w^T x + b = 0$. The splitting hyperplane obtains the largest distance to the closest positives

$w^T x + b = +1$ and negatives $w^T x + b = -1$. The linear kernel function is defined as follows:

$$K(x_i, x_j) = x_i^T x_j \tag{12}$$

Non-linear SVMs for Multi-Class Problems

The linear SVM algorithm has been extended to deal with non-linear multi-class classification problems by constructing and combining several binary classifiers (Hsu and Lin, 2002). Many non-linear kernel functions have been proposed. In this chapter we choose the radial basis function (RBF) kernel as a non-linear similarity function in the SVM classifier. The RBF kernel computes the following similarity value between two input vectors (Eq. (13)):

$$K(x_i, x_j) = \exp(-\gamma \left\| x_i - x_j \right\|^2) \tag{13}$$

where γ is a kernel parameter of the RBF kernel. Large values of the γ parameter may cause overfitting due to the increase of the number of support vectors.

For multi-class problems, we use the one-vs-all strategy. The problem consists of choosing the kernel function and tuning a variety of parameters (Rossi and Carvalho, 2008). In this method, the input vector is given to all SVM models which are constructed for each class. Then the class with the maximum discriminant output is selected from these models as the winning class. The idea here is that every model is constructed to discriminate between a class and the other classes (Liu et al., 2003).

The hyperparameters that need to be tuned in the SVM with the RBF kernel include the cost parameter (C) and the gamma parameter (γ). The C parameter has a significant effect on the decision boundary. It controls the width of the margin. The γ parameter directly affects overfitting. This causes large γ values to increase the number of support vectors (Rossi and Carvalho, 2008).

To deal with the large scale of the feature vectors, the L2-regularized SVM (L2-SVM) (Koshiba and Abe, 2003; Fan et al., 2008) has been proposed. It uses the sum of squares of ξ_i instead of the sum of the ξ_i as follows:

$$J(\mathbf{w}, \xi) = \frac{1}{2}\|\mathbf{w}\|^2 + \frac{C}{2}\sum_{i=1}^{M}\xi_i^2 \tag{14}$$

Model Selection

We used regular grid search for exploring the two dimensional parameter space. The logarithmic scale is chosen. The goal is to estimate the accuracy of the classifier for each point on the grid. In order to prevent over-fitting, cross validation is used. The training set is divided into k-folds, one fold is used as test set, and k-1 folds are used as training set. This is then repeated k times.

Normalization

It is very important to normalize the feature vector before applying SVMs. This is to avoid attributes of the feature vector in large numeric ranges. Therefore, the entire training and test dataset are scaled with the same normalization method. In our experiments, the features are scaled to the range [0, 1].

3.5 Data Collection and Preparation

The datasets used in the present study include Thai character, Bangla digit and MNIST (LeCun and Cortes, 1998). Fig. 30 shows some examples of handwritten characters. Each dataset consists of isolated characters. MNIST consists of 60,000 training examples and 10,000 test examples. MNIST is a handwritten digit dataset that has been widely used as benchmark for comparing feature extraction techniques (Lauer et al., 2007). In the present study, 10,000 records (10 classes) of the MNIST dataset were used. For the Bangla

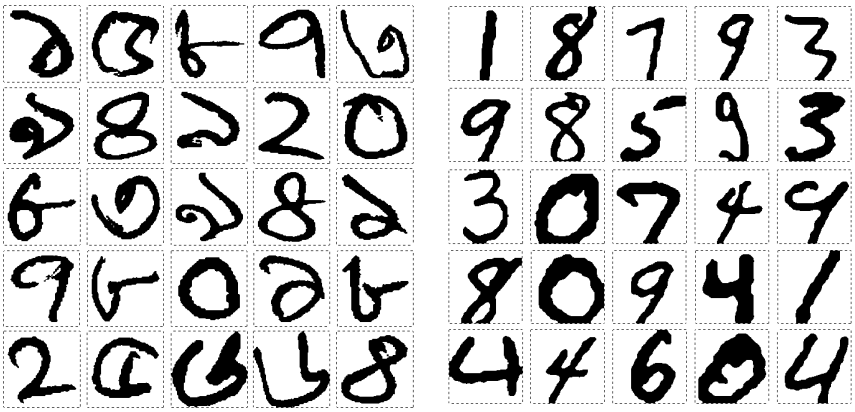
digit dataset[1], 10,920 records of the numbers 0 to 9 (10 classes) are used. The most distinctive aspect of the Bangla digit dataset is the large variety of handwriting styles (see Fig. 21). Because of the different handwriting styles, two of the numbers look similar as shown in Fig. 31. The Thai character dataset used in the present study includes 65 classes consisting of consonants, vowels and tones. There are 5,900 Thai character examples in this dataset. The Thai character dataset was collected from characters written by writers aged from 20 - 23 years old. Among this group of data, there are characters written by 7 female writers and 3 male writers.

In order to prepare the handwritten images for our feature extraction techniques, our system executes a number of preprocessing steps. First, the data in the handwritten Bangla digit dataset contain different kinds of backgrounds, some of which are clear but most are not clear and even quite noisy. The four background removal algorithms investigated in this study were those of Otsu, Niblack, Sauvola, and Wolf (Som et al., 2011; Wolf et al., 2002). The results of handwritten images after applying the binarization methods are shown in Fig. 32. In this experiment, Otsu's algorithm (Fig. 32b) was the best algorithm to transform a gray image into a binary image without noise.

The second problem with the handwritten Bangla digit dataset is that numbers were scanned into digital images at different resolutions. The Bicubic interpolation, which is an efficient normalization algorithm (Mahmud, 2005; Hamidi and Borji, 2010), was used to normalize the handwritten image to fit into a 28×28 and 40×40 pixel space, which yields images with quality from good to outstanding.

Finally, the last process in preparation of the dataset is thinning. This technique is used in order to create images which are uniformly thin, as the current dataset (Fig. 21) displays a large variety of ink thickness. The results from the thinning technique are shown in Fig. 33. Other researchers such as Pal et al. (2008), Bulacu and Schomaker (2007), and Liu and Suen (2009) have not applied thinning, as this procedure is not useful for their feature

[1]We are sincerely grateful to Dr. Tapan K. Bhowmik for providing the Bangla digit data used in the present study.

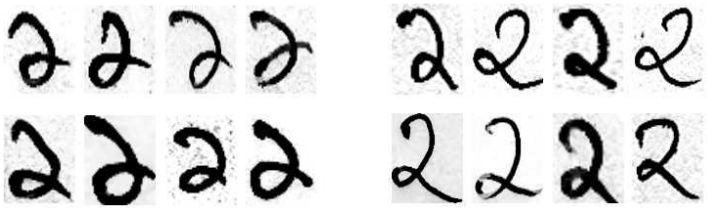


(a) Bangla digit dataset. (b) MNIST dataset.

(c) Thai character dataset.

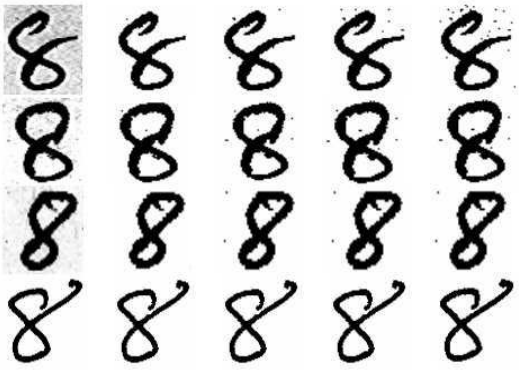
Figure 30: Some examples of handwritten character images used in the present study. (a) Bangla digit, (b) MNIST and (c) Thai character dataset.

extraction techniques. We will use thinning for our two feature extraction techniques, but not for the pixel-based methods.



(a) Bangla digit number 1 (b) Bangla digit number 2

Figure 31: Similarities between different handwritten Bangla digits. (a) The images of number 1, and (b) number 2.



(a) (b) (c) (d) (e)

Figure 32: Results of handwritten images after applying the binarization methods. (a) The handwritten image before removing background noise, (b) background noise is removed by Otsu's algorithm, (c) Wolf's algorithm, (d) Sauvola's algorithm, and (e) Niblack's algorithm.

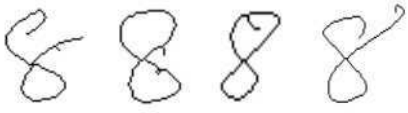


Figure 33: The handwritten images after applying the thinning technique.

3.6 Experimental Results

We evaluate the handwritten character recognition techniques on three handwritten datasets including Thai character, Bangla digit, and MNIST. The

datasets are composed of isolated handwritten scripts. The handwritten images are converted to binary and normalized to a fixed-size image. There are two pixel resolutions which are used in these experiments. In section 3.6.1, the handwritten images are normalized to 40×40 pixels and experiments with the kNN algorithm are described. In section 3.6.2, the images are normalized to 28×28 pixels and experiments with the SVM algorithm are described. In these experiments, we are using the recognition rate as our evaluation metric to mesure the performance of each feature extraction technique.

3.6.1 Experiments with the kNN Algorithm

We presented a novel method for feature extraction, called the hotspot technique. This novel technique is compared to other methods including mask direction and the direction of chain code technique. The feature vectors obtained from these techniques are classified by the kNN method. The outcome of the classification process is the classification rate. The classification process is an indicator for the efficacy of each technique. All different extractors were applied to three datasets including Thai character, Bangla digit, and MNIST. These three datasets were treated with the same methods so that the size of character images for all datasets are determined as 40×40 pixels.

The mask direction technique is also known as the direction feature (Blumenstein et al., 2003) and suitable for tracking the directions of the character image. The mask size for the MDT is 3×3 pixels (Kawtrakul and Waewsawangwong, 2000). The mask directions consist of horizontal mask, vertical mask, left-diagonal mask, and right-diagonal mask (Blumenstein et al., 2003). The number of features obtained from MDT was 64.

The DCC is an efficient technique in handwriting recognition (Bhowmik et al., 2007). We applied this technique to the present study according to the methods described by Pal et al. (2008), although we adapted their technique to deal more efficiently with our datasets by identifying the starting point of the direction in each block.

The datasets were divided into 10 subsets (90% training set and 10% test set). We randomly divided the data into a test and training set 10 different times. The value of k of the kNN classifier was optimized for each method and dataset.

Table 2: Comparison of data classification efficacy of feature extraction techniques by using kNN.

Dataset	Feature extraction technique		
	HOT	MDT	DCC
Thai character	83.3 ± 0.5	**88.0** ± 0.6	71.3 ± 0.7
MNIST	**89.9** ± 0.3	85.1 ± 0.3	83.5 ± 0.2
Bangla digit	**90.1** ± 0.4	87.6 ± 0.4	82.7 ± 0.4

The size of the feature vectors obtained from HOT, MDT, and DCC were 100, 64, and 128 dimensions, respectively, For the HOT we used $d_{max} = 20$ for the two datasets containing digits, and $d_{max} = 0$ for the Thai character dataset, which worked slightly better for an unknown reason than $d_{max} = 20$. Table 2 shows the comparison of classification efficacy of the different feature extraction techniques. It is found that the best feature extraction technique for classification is HOT, followed by MDT and DCC, respectively. The average classification rate obtained from HOT, MDT, and DCC are 87.8%, 86.9%, and 79.2%, respectively. Our new technique significantly outperforms the other feature extraction technique on the two datasets containing digits. The MDT outperforms our method on the Thai character dataset. The direction of chain code technique obtains the worst performance by far. This technique is more complicated and involves several subtleties which requires adapting it to different datasets. Much better results for MNIST have been reported in literature (above 99% accuracy), but in those studies more training patterns were used (60,000 compared to 10,000 in our study). This dataset has a very large number of examples and few classes, which makes pixel-based methods more effective. However, we believe that by more fine-tuning, using more examples and better classifiers,

Table 3: The Recognition Rates of the Different Methods on Bangla digit with an SVM classifier.

Training data	Recognition rate (%)			
	CAT	HOT	GPB	BWS
10%	**92.2** ± 0.9	88.0 ± 1.7	90.5 ± 1.3	88.1 ± 2.1
20%	**94.3** ± 1.2	90.3 ± 1.4	93.6 ± 1.5	91.0 ± 2.2
30%	**95.4** ± 0.9	91.8 ± 1.4	94.8 ± 1.2	93.5 ± 1.5
40%	**95.9** ± 1.0	92.7 ± 1.2	**95.9** ± 1.3	94.1 ± 0.9
90%	96.0 ± 0.8	92.7 ± 0.8	**96.4** ± 0.9	94.7 ± 1.1

and combining multiple feature extraction techniques, we are able to obtain similar performances.

3.6.2 Experiments with the SVM Algorithm

We have used 10-fold cross validation to evaluate the results of the handwritten Bangla digit recognition methods including the pixel-based methods. The best values of the C and the γ parameters, which were found by grid search, are chosen and used to train a model which is evaluated on the test set. The result of this process is the mean accuracy and the standard deviation (σ).

We used training set sizes of 10, 20, 30, 40, and 90%, respectively of 10,920 examples in total. The summary of results is given in Table 3. The recognition rate of GPB was quite low at 10% of the training set size for which it obtained 90.5% accuracy. The CAT feature obtained 92.2% accuracy when the dataset was decreased to 10%. The recognition rate of the CAT feature is significantly higher than the GPB pixel-based method for this small amount of training data. When all training examples are used, the best pixel-based method is slightly more powerful than the best feature extraction technique, although this difference is not statistically significant. The GPB obtains a

Table 4: The Recognition Rates of The Unweighted Majority Vote for Combining SVM Classifiers

Training data	Recognition rate (%)	
	Best classifier	UMV
10%	92.2 ± 0.9	**93.5** ± 1.0
20%	94.3 ± 1.2	**95.2** ± 1.0
30%	95.4 ± 0.9	**96.0** ± 0.8
40%	95.9 ± 1.0	**96.7** ± 0.9
90%	96.4 ± 0.9	**96.8** ± 0.6

high accuracy of 96.4%. The results of the feature and the pixel-based methods for recognizing the Bangla digits are also shown in Fig. 34.

Furthermore, we used the unweighted majority vote method (UMV) described by Moreno-Seco et al. (2006) to combine the outputs from the four different SVM classifiers. The number of votes for each class is counted, and the class with the majority is selected as the output of the ensemble. A random method is used to choose between classes when they obtained the same number of votes. The accuracy increased to 96.8% when UMV is used. The results of UMV are shown in Table 4 and in Fig. 34. In both result tables, the difference between the best method and the second best method is only statistically significant with 10% of the training examples.

We were also interested in the computation time of training the SVM classifier. In this experiment, the used desktop computer is: Intel(R) Pentium(R) D CPU 3.00GHz. The computation times of the different training schemes are shown in Fig. 35. CAT is 13.5 times faster than GPB. We also measured the operational time it costs on average to perform the necessary operations for each method (e.g., converting to 28×28, binarization, thinning, feature extraction, and classification with the SVM). The BWS method is the fastest one with 29 ms used to classify a handwritten character image. The CAT and HOT methods require 37 ms and 35 ms, respectively. Finally,

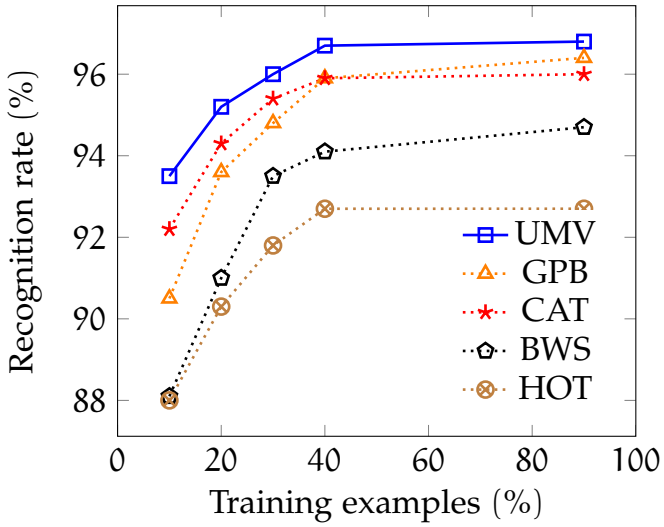


Figure 34: The recognition rates of feature extraction techniques and the majority vote for combining SVM classifiers on the handwritten Bangla digit dataset.

the slowest method is the GPB technique that consumes 73 ms to classify an image.

3.7 Conclusions

The present study proposed a new technique for feature extraction, named the hotspot technique. In this technique, the distance values between the closest black pixels and the hotspots in each direction are used as representation for a character. There are two key parameters to be taken into account; 1) number of hotspots and 2) number of chain code directions. The HOT was applied to digit datasets including MNIST and Bangla digit, and Thai characters.

For the two datasets with few classes, namely the handwritten digit datasets, Bangla and MNIST, in the first study the novel hotspot technique significantly outperforms the other methods. However, the MDT outper-

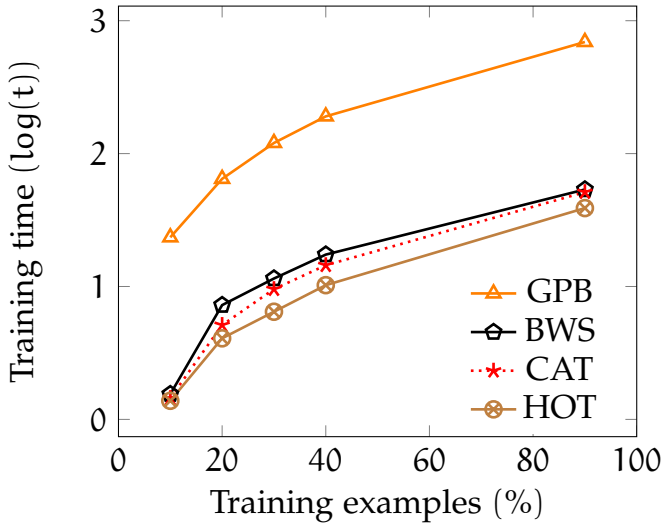


Figure 35: Results of the time (log(t)) needed for training the SVM classifier with the different methods for computing features, where t is in seconds.

forms the HOT on the Thai character dataset that has much more classes (65). Maybe the HOT needs more examples for this dataset, possibly because it is less robust to variances in the handwritten characters than the MDT. Our results on multiple script datasets confirmed that the highest average recognition rate is achieved when using the HOT technique.

Furthermore, we studied if feature extraction techniques can outperform pixel-based methods for handwritten character recognition. We described some advanced feature extraction techniques and evaluated the performance on the handwritten Bangla digit dataset. The techniques that were used include CAT, HOT, GPB, and BWS, and the system used a support vector machine as a classifier to yield high accuracies. The best feature extraction technique CAT outperforms the best pixel-based method when the training dataset is not very large. When the training dataset size increases, the best pixel-based method slightly outperforms this feature extraction technique. However, in terms of computation time the CAT feature extraction technique outperforms the GPB pixel-based method, because the latter uses all pixels of the handwritten image. Finally, The majority voting technique

increased the performance of the handwritten character recognition system. It obtained 96.8% accuracy with 90% of the training data.

Future Work

Several neural network architectures have obtained very high recognition rates on the MNIST dataset and we are interested in finding the utility of feature extraction techniques compared to the use of strong classifiers that immediately work on pixel representations. Furthermore, keypoint methods have not deserved a lot of attention in handwriting recognition, and we want to explore the use of adaptive keypoints to be more translation invariant and also use generative models to maximize the probability of generating the data.

On the other hand, we will concentrate on developing the novel feature extraction techniques that obtain high accuracies on the Thai handwritten datasets (character and digit), which is challenging because these characters also have curly extensions and shape variations, unlike the plain Arabic numerals in MNIST.

4 LOCAL GRADIENT FEATURE DESCRIPTORS

In this chapter we propose to use local gradient feature descriptors, namely the scale invariant feature transform keypoint descriptor and the histogram of oriented gradients, for handwritten character recognition. The local gradient feature descriptors are used to extract feature vectors from the handwritten images, which are then presented to a machine learning algorithm to do the actual classification. As classifiers, the k-nearest neighbor and the support vector machine algorithms are used. We have evaluated these feature descriptors and classifiers on three different language scripts, namely Thai, Bangla, and Latin, consisting of both handwritten characters and digits. The results show that the local gradient feature descriptors significantly outperform directly using pixel intensities from the images. When the proposed feature descriptors are combined with the support vector machine, very high accuracies are obtained on the Thai handwritten datasets (character and digit), the Latin handwritten datasets (character and digit), and the Bangla handwritten digit dataset.

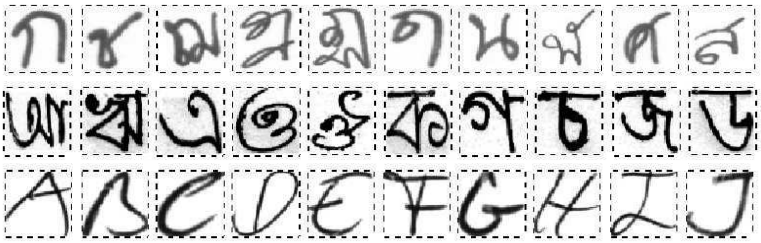
This chapter has previously been published in *Engineering Applications of Artificial Intelligence (2015)*

Handwritten character recognition systems have several important applications, such as zip-code recognition, writer identification for e.g. forensic research, searching in historical manuscripts and others. For such applications, the system should be able to recognize handwritten characters written on many different kinds of documents, such as contemporary or historical manuscripts. The aim is to let the system automatically extract and recognize the characters that are embedded in the manuscript. The quality of the manuscript is one of the factors that can improve the recognition accuracy (Gupta et al., 2011). It is essential to deal with the different problems that occur in the manuscripts, such as distortions in a character image and the background noise that can appear during the scanning process. The aim of our work is to develop new algorithms that can obtain a high recognition accuracy.

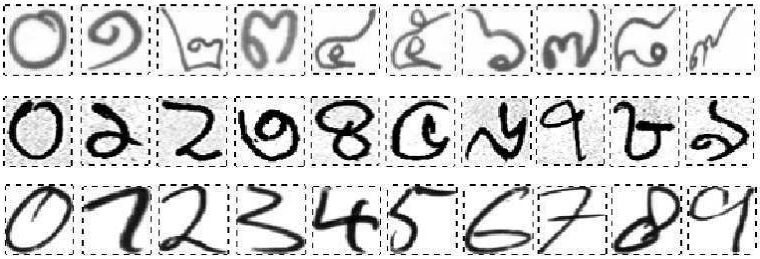
Obtaining high recognition accuracies on handwritten character datasets is a challenging problem, for which many different solutions have been proposed. Although on the standard MNIST dataset extremely high accuracies have been obtained (Meier et al., 2011), there are many other datasets consisting of less examples and which can be considered more difficult. These datasets are challenging due to different writing styles of the same characters, different writing persons (with differences in age, gender, and education), different writing devices, and difficulties due to background noise that appears from the printer (Surinta et al., 2012).

In this chapter we emphasize the importance of the recognition of complex handwritten Thai, Bangla, and Latin scripts, for which the handwritten characters and digits are highly varying due to different shapes, strokes, curls, and concavities (Mandal et al., 2011). Some samples of the handwritten characters are shown in Fig. 36. Note that the handwritten images shown in this chapter are resized to the same resolution for illustration purposes. Due to the high variety, the direct use of pixel intensities may not work very well, because there is sometimes little overlap between two handwritten images displaying the same character. Therefore, in this chapter we propose to use feature extraction techniques which are robust to local displacements, but still provide discriminative feature vectors as representation of the handwritten characters. The feature extraction methods that

we will make use of have also been extensively used for object recognition, namely a descriptor that is inspired by the scale invariant feature transform (SIFT) as proposed by Lowe (2004) and the histogram of oriented gradients (HOG) (Dalal and Triggs, 2005). This chapter shows that the use of these local gradient feature descriptors to extract features from handwritten characters and digits leads to a very well performing system. High recognition performances are obtained on the challenging handwritten datasets even with a simple classifier such as the k-nearest neighbor method, and very high recognition accuracies are obtained when using a support vector machine classifier.



(a)



(b)

Figure 36: Some examples of the Thai, Bangla and Latin handwritten scripts as shown in the first, second, and third row, respectively. Sample of (a) handwritten characters and (b) handwritten digits.

Related Work

In previous studies, the raw image (IMG) method, which directly copies the intensities of the pixels of the ink trace (Surinta et al., 2013), has often been used as the feature extraction method. It extracts a high dimensional feature vector that depends on the size of the input image.

In recent years, deep learning architectures (Hinton et al., 2006; Schmidhuber, 2015) have been effectively used for handwritten digit recognition. Most of the studies have focused on the benchmark MNIST dataset (LeCun and Cortes, 1998) and achieved high accuracies such as higher than 98 or 99 percent. The MNIST dataset consists of isolated handwritten digits with size of 28 $\times$ 28 pixel resolution and contains 60,000 training images and 10,000 test images. In (Hinton et al., 2006), a greedy training algorithm is proposed for constructing a multilayer network architecture which relies on the restricted Boltzmann machine, called deep belief networks (DBN). The performance obtained from the DBN with three hidden layers (500-500-2,000 hidden units) on the MNIST dataset was 98.75%. This accuracy is higher than obtained with a multi-layer perceptron and a support vector machine (SVM).

Furthermore, the convolutional neural network (CNN) (LeCun et al., 1998) is used as a feature extraction and classification technique, and the accuracy obtained is 99.47% (Jarrett et al., 2009). In another CNN-based method (Cireşan et al., 2011), the committee technique is proposed. Here multiple CNNs are combined in an ensemble, for which different CNNs are trained on different pixel resolutions of the images. The images in the dataset are rescaled from 28 $\times$ 28 (N $\times$ N) to N = 10, 12, 14, 16, 18, and 20 pixel resolutions. Then, 7-net committees are used. This method obtained the high accuracy of 99.73% on MNIST. However, a single CNN in their work is reported to take approximately 1 to 6 hours for training on a graphics processing unit (GPU) and the 7-net committees are seven times slower than a single CNN. The best technique for the MNIST dataset uses an ensemble of 35-net committees (Cireşan et al., 2012). This technique obtained the very high accuracy of 99.77%. Although such high recognition performances are sometimes achieved, these methods require large training sets and long training times to make the recognition system work well.

For handwritten Bangla digit recognition, Liu and Suen (2009) proposed to use the local gradient directions of local strokes, called the gradient direction histogram feature. The feature vectors are extracted from an image and then given to a classifier. The recognition performance of the best classifier is 99.40% on the the ISI Bangla numerals dataset (Chaudhuri, 2006) which contains 19,329 training images and 4,000 test images. Compared to the MNIST dataset, the ISI Bangla numerals dataset is more difficult due to background noise and more different types of handwriting.

In our research, we are interested in novel methods that obtain high recognition accuracies without the availability of many training examples, and which also do not require a huge amount of training time or high performance computing algorithms.

Contributions

This chapter first of all provides a new standard Thai handwritten character dataset for comparison of feature extraction techniques and methods. In this chapter we will make use of three complex datasets in total, namely Bangla, Thai, and Latin, for which very high recognition accuracies have not been obtained before. This is due to the difficult problems of the Thai and the Bangla handwritten scripts such as the complex structural characteristics of the characters, the similarities between the character sets (see Fig. 43a and Fig. 43b), the similar structures between different characters (see Fig. 39), and the background noise (see Fig. 43c). These factors negatively affect the performance of a handwritten character recognition system.

To address the problems mentioned, two local gradient feature descriptors that extract feature vectors from the challenging handwritten character images are proposed, namely the scale invariant feature transform keypoint descriptor (Lowe, 2004) and the histogram of oriented gradients (Dalal and Triggs, 2005). The feature descriptors compute feature vectors with image filters such as the Sobel filter and the Gaussian filter. Subsequently, the orientations within each region are calculated and weighted into an orientation histogram. Because these feature descriptors are invariant to small local displacements, the descriptors provide robust feature vectors.

These feature extraction methods extract features, which are then used as input for a classifier. In this chapter, we experimented with two different classifiers: a k-nearest neighbor classifier and a support vector machine, so that we can also compare performance differences between these machine learning methods. We evaluate the methods on the three handwritten character scripts: Thai, Bangla, and Latin, for which we use both the handwritten characters and the handwritten digits. To show the importance of using the proposed local gradient feature descriptors, we have compared them to a method that directly uses pixel intensities of the handwritten images (called the IMG method). The results show that the feature descriptors with the support vector machine obtain very high recognition performances on the datasets, whereas the use of the IMG method performs much worse.

4.1 Local Gradient Feature Descriptors

To study the effectiveness of local gradient feature descriptors for handwritten character recognition, we compare two existing feature extraction techniques, namely the histogram of oriented gradients and the scale invariant feature transform keypoint descriptor. Moreover, these local gradient feature descriptors are compared to the IMG method. The IMG method uses the raw pixel intensities of the handwritten images and is a simple and widely used method. In this study, the handwritten images are resized to two pixel resolutions: 28×28 and 36×36, so that for the IMG method 784 and 1,296 feature values are computed, respectively.

4.1.1 Histograms of Oriented Gradients (HOG)

The HOG descriptor was first introduced in (Dalal and Triggs, 2005) for detecting a human body in an image. It has become very successful in diverse domains such as face, pedestrian, and on-road vehicle detection (Déniz et al., 2011; Lee et al., 2013; Arróspide et al., 2013). The HOG descriptor is originally defined as the distribution of the local intensity gradients from

an image, which are computed from small connected regions (*"cells"*). We will now present the details of the HOG image descriptor.

The HOG feature vector is computed from the image using gradient detectors. In this chapter, each pixel is convolved with the simple convolution kernel as follows:

$$\begin{aligned} G_x &= I(x+1,y) - I(x-1,y) \\ G_y &= I(x,y+1) - I(x,y-1) \end{aligned} \tag{15}$$

where $I(x,y)$ is the pixel intensity at location x,y in the input image. G_x and G_y are the horizontal and vertical components of the gradients, respectively. In our experiments, the HOG descriptor is calculated over rectangular blocks (R-HOG) with non-overlapping blocks.

To ignore negative gradient directions, the range of gradient orientations are defined between 0° and 180° (Dalal and Triggs, 2005; Arróspide et al., 2013). The gradient magnitude M and the gradient orientation θ are calculated by:

$$\begin{aligned} M(x,y) &= \sqrt{G_x^2 + G_y^2} \\ \theta(x,y) &= \tan^{-1} \frac{G_y}{G_x} \end{aligned} \tag{16}$$

After this, histograms are computed from the occurrences of oriented gradients across large structures (*"blocks"*) of the image as shown in Fig. 37. The gradient orientations are stored into 9 orientation bins β.

The combination of the histograms from each block represents the feature descriptor. The feature vector size of the HOG descriptor depends on the selected numbers of blocks and bins. It has been shown that the performance of the HOG descriptor depends mostly on the number of blocks (Déniz et al., 2011).

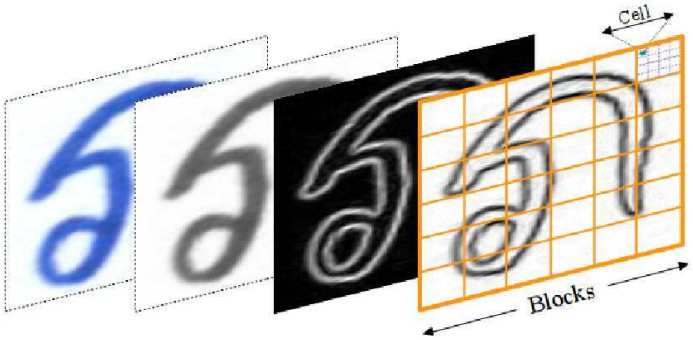


Figure 37: Example of the rectangular HOG descriptor. The third image shows the gradient magnitude image after applying the simple convolution kernel to the second image. The fourth image shows the partitioning of the image into 6×6 non-overlapping blocks. Here, each block provides a separate angular histogram for the gradient orientations, which are afterwards concatenated and normalized.

Finally, the feature descriptors are normalized by applying the L2 block normalization (Lee et al., 2013) as follows:

$$V'_k = \frac{V_k}{\sqrt{\|V_k\|^2 + \varepsilon}} \tag{17}$$

where V_k is the combined histogram from all block regions, ε is a small value close to zero and V'_k is the normalized HOG descriptor feature vector.

4.1.2 Scale Invariant Feature Transform Descriptor (siftD)

The Scale Invariant Feature Transform (SIFT) descriptor was described in (Lowe, 2004) and is quite similar to the HOG descriptor, but there are some important differences as well, which we will explain later. The siftD computes 128 dimensional feature vectors for each keypoint (Sun et al., 2014). The detected keypoints in the standard SIFT algorithm are computed so that they are invariant to different translations, scales, rotations and they

are also robust to other local geometric distortions. Additionally, for each keypoint a translation, scale and orientation value are computed. The SIFT method is widely used in object, scene and face recognition (Abdullah et al., 2009; Seo and Park, 2014).

To extract feature vectors from images, the standard SIFT algorithm detects keypoints that correspond to the local extrema of the Difference of Gaussians (DoG) function applied to the image with different scales. The problem of the standard SIFT algorithm is that in processing a character image, the number of detected keypoints will be variable. Therefore, the feature vectors are of variable size and different methods to handle this such as visual codebooks need to be used. However, the character images are well defined and well segmented. Therefore, in this study, we only use the 128-dimensional descriptor, at the given locations, e.g. the center of a character box (see Fig. 38). In order to determine whether this provides a sufficient resolution, additional experiments with more predefined keypoint centers will be performed, yielding higher-dimensional siftD feature vectors.

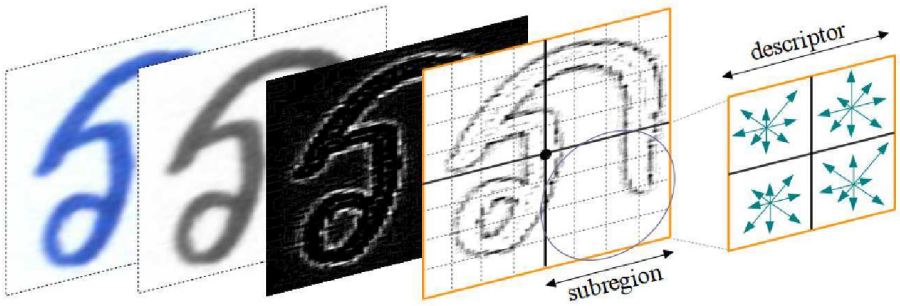


Figure 38: Overview of the SIFT descriptor. The illustration shows a character and one main region. The region is divided into 4×4 subregions (only 2×2 are shown). One subregion subsequently provides a descriptor which is represented as 8 orientation bins as shown on the right, yielding a 128-dimensional feature vector (dubbed siftD, here).

The siftD performs the following steps to extract the features from a handwritten character image. First, the input image is smoothed by a convolution with a variable-scale Gaussian kernel:

$$L(x, y, \sigma) = G(x, y, \sigma) * I(x, y) \tag{18}$$

where $I(x, y)$ is the pixel intensity at location x, y in the input image and $G(x, y, \sigma)$ is the Gaussian kernel. The parameter σ determines the width of the Gaussian kernel and is set to 0.8 in our experiments. Then, the horizontal and vertical components of the gradients G_x and G_y are computed according to Eq. 19. Afterwards, the magnitude $M(x, y)$ and orientation $\theta(x, y)$ for each Gaussian smoothed image pixel are computed according to Eq. 16.

$$\begin{aligned} G_x &= L(x+1, y, \sigma) - L(x-1, y, \sigma) \\ G_y &= L(x, y+1, \sigma) - L(x, y-1, \sigma) \end{aligned} \tag{19}$$

The main image is split into 4×4 subregions (blocks) and then for each block an orientation histogram is made. The orientation histogram uses 8 bins which cover 360 degrees, which results in 128 dimensions for the feature vector if one main region (consisting of 4×4 subregions) is used. The orientation histogram is weighted by gradient magnitudes and a Gaussian-weighted circular window (Lowe, 2004).

It should be noted that the proposed use of siftD is somewhat similar to the HOG method, which is also orientation based. However, there are still a number of differences:

1. The HOG descriptor uses absolute angles between 0 and 180 degrees and siftD uses all angles between 0 and 360 degrees.
2. In HOG all pixels are weighted equally in the rectangular blocks whereas in the siftD the influence of local gradients of different pixels is computed by weighting the distance of the pixel to the keypoint (center of the region).
3. siftD uses a Gaussian filter before extracting the gradient orientations and magnitudes.

4.2 Handwritten Character Datasets

We evaluate the different handwritten character recognition methods on three isolated handwritten script datasets belonging to three languages (Thai, Bangla, and Latin), all of which are composed of handwritten characters and digits. The original handwritten scripts in the datasets are not normalized to a fixed-size image and therefore are in numerous pixel space resolutions. Furthermore, we have manually collected a new Thai handwritten script dataset that contains 24,045 character images in total from various writers. The details of the Thai handwritten dataset are described in Section 4.2.1.

In order to prepare the handwritten character images, a few simple preprocessing steps are applied. The handwritten images are first converted from the color image format into a gray-scale image. Subsequently, the sample images are normalized into 28×28 and 36×36 pixel resolution with aspect ratios preserved. The experiments on the different pixel resolutions of the handwritten images are described in Section 4.3.

4.2.1 Thai Handwritten Dataset

The number of Thai consonants is not uniquely defined (Phokharatkul and Kimpan, 2002), because some characters are outdated. In this research, the Thai handwritten dataset is collected according to the standard Thai script consisting of 78 characters. On the other hand, Nopsuwanchai et al. (2006) presented a ThaiCAM database that contains a different Thai script of 87 characters. This script includes some extra obsolete characters and special symbols, which are not essential for writing. Generally, the writing style of several Thai characters is very similar, but there are some differences in some details such as head, loop, curl, and concavity as shown in Fig. 39. Some character recognition systems use local features to extract information of the characters (Phokharatkul et al., 2007). However, some important details can disappear because of the writing styles. Various writing styles of Thai handwritten characters are illustrated in Fig. 40.

The performances obtained with previous approaches have not reached very high recognition rates. Nopsuwanchai et al. (2006) proposed block-based principal component analysis and composite images and used a hidden Markov model (HMM) as a classification technique. They obtained 92.03% accuracy on the ThaiCAM database. Some hybrid techniques of heuristic rules and neural networks are employed in (Mitrpanont and Imprasert, 2011). The performance obtained from this approach on the Thai handwritten character dataset was 92.78%.

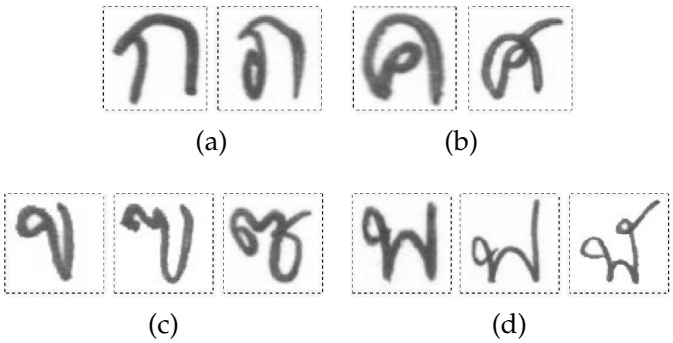
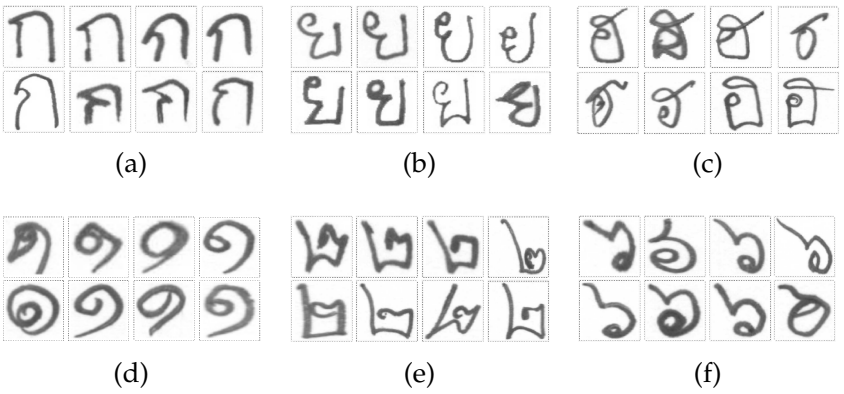


Figure 39: Illustration of the relation between different Thai characters. In (a) and (b), the second character is constructed by slightly changing the first (different) character. In (c) and (d) the third is created by a modification of the second character, which is a modification of the first character.

We collected a new Thai handwritten script dataset from 150 native writers who studied in the university and are aged from 20 to 23 years old. They used a 0.7 mm ink pen writing Thai scripts consisting of consonants, vowels, tones, and symbols on a prepared A4 form. The participants were allowed to write only the isolated Thai script on the form and at least 100 samples per character. We allowed writers to write in various styles without pressure. However, the character images obtained from this dataset generally have no background noise. Moreover, the forms were scanned at a resolution of 200 dots per inch and stored as color images. Finally, we have used an uncomplicated line and character segmentation method based on the horizontal and vertical projection profile (Surinta and Chamchong, 2008)

to separate and crop the isolated characters. The Thai handwritten dataset used in the experiments is a subset from a very large set. The dataset is available from http://www.ai.rug.nl/~mrolarik/THI/ for research purposes.



(a) (b) (c)

(d) (e) (f)

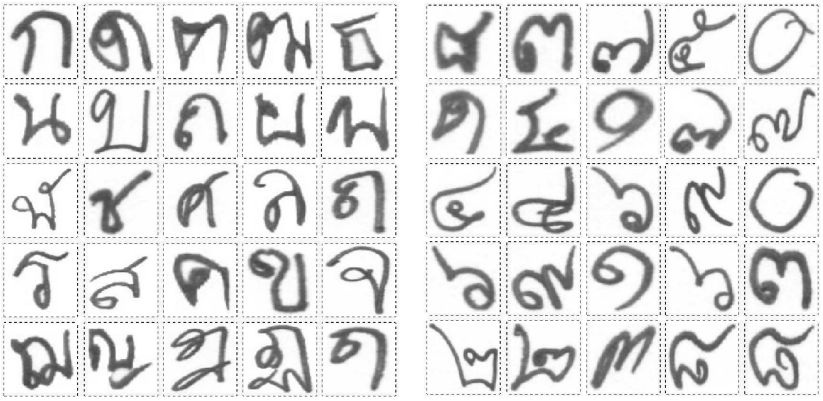
Figure 40: Illustration of the diversity in the writing styles of the Thai handwritten dataset. (a), (b) and (c) show samples of Thai handwritten characters and (d), (e) and (f) show samples of Thai handwritten digits.

Thai Handwritten Character Dataset (THI-C68)

This dataset consists of 13,130 training samples and 1,360 test samples randomly selected from the main dataset. Sample images of the dataset are illustrated in Fig. 41a. In this research, we have selected the standard Thai script of 68 Thai characters, which are composed of 44 consonants, 17 vowels, 4 tones, and 3 symbols.

Thai Handwritten Digit Dataset (THI-D10)

This dataset has 9,555 samples including 8,055 training samples and 1,500 test samples. The total number of samples in each class is larger than 900. Sample characters of this dataset are shown in Fig. 41b.

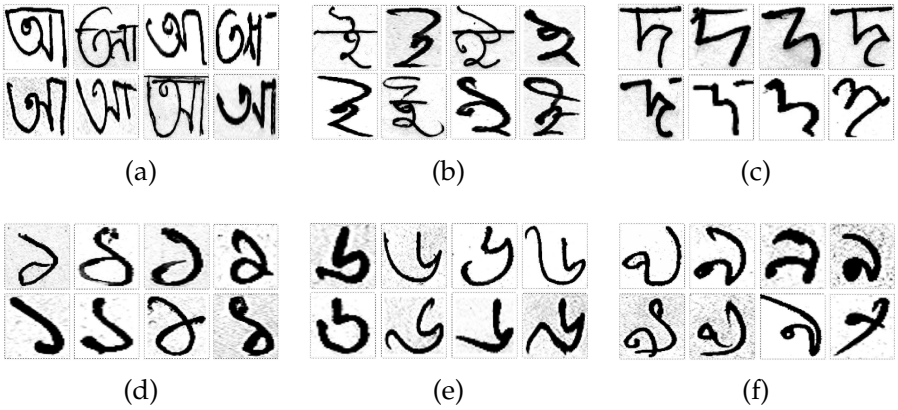


(a) THI-*C68* (b) THI-*D10*

Figure 41: Illustration of the Thai handwritten images. (a) Thai handwritten characters, and (b) Thai handwritten digits.

4.2.2 Bangla Handwritten Dataset

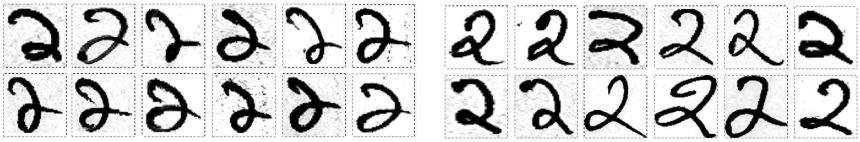
Bangla (or Bengali) is the second most popular language in India and Bangladesh and the fifth most used language around the globe (Pal et al., 2007). The Bangla basic script consists of 11 vowels and 39 consonants (Bhowmik et al., 2009; Das et al., 2010). This chapter deals with recognition of handwritten characters of 45 classes and handwritten digits of 10 classes from different writers. The Bangla handwritten dataset (Bhowmik et al., 2009) in this study has a large diversity of writing styles as shown in Fig. 42 and some characters are nearly identical with other characters. The dataset contains different kinds of backgrounds, some of which are clear, but most are quite noisy. Finally, it contains a variety of pixel space resolutions. Hence, it is much more challenging than the well-known MNIST dataset (LeCun and Cortes, 1998). Fig. 43a and Fig. 43b show the similarities between two different handwritten digits and Fig. 43c illustrates some examples of the noisy background.



(a) (b) (c)

(d) (e) (f)

Figure 42: The variety in writing styles of the Bangla handwritten dataset. (a), (b), and (c) Samples of Bangla characters. (d), (e), and (f) Some samples of Bangla digits.



(a) Bangla digits, number 1 (b) Bangla digits, number 2

(c) Noisy backgrounds

Figure 43: Illustration of the similarities between different Bangla handwritten digits and the noisy background which appears in the Bangla handwritten dataset. (a), (b) Similarities of Bangla digits between number 1 and number 2, respectively. (c) Bangla characters with various noisy backgrounds.

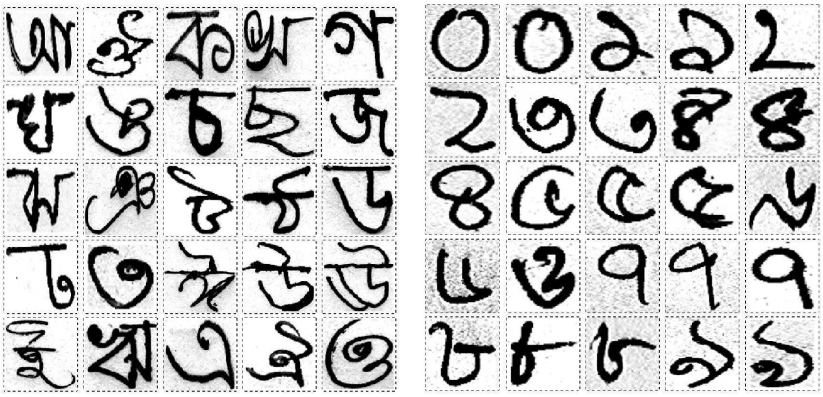
Bangla Handwritten Character Dataset (BANG-C45)

The Bangla basic character set includes 45 classes and contains 4,627 character images in the training set and 900 examples in the test set. In this dataset

the number of character images per class is around 100. Additionally, the characters in the dataset are in gray-scale format, and some of them have a noisy background. Some samples of the Bangla handwritten character dataset are shown in Fig. 44a.

Bangla Handwritten Digit Dataset (BANG-D10)

The set of Bangla digits consists of 9,161 instances in the training set and 1,500 instances in the test set. We randomly selected 150 character images per class as a test set. Some examples of this dataset are shown in Fig. 44b.



(a) BANG-*C45* (b) BANG-*D10*

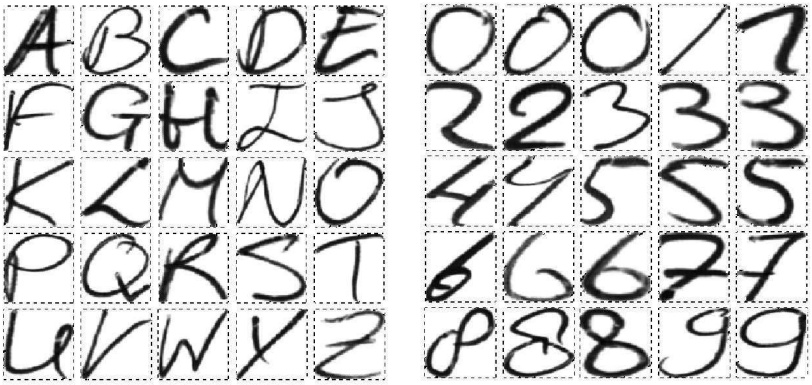
Figure 44: Some examples of the Bangla handwritten dataset. (a) Bangla handwritten characters, and (b) Bangla handwritten digits.

4.2.3 Latin Handwritten Dataset

The benchmark dataset for Latin handwritten character recognition is provided by der Maaten (2009). The original images were collected by Schomaker and Vuurpijl (2000) for forensic writer identification and was named the *Firemaker* dataset. The handwritten text was written in Dutch script by 251 writers. It has 40,133 handwritten images and consists of uppercase charac-

ters and digits. In this dataset the capital letters are collected except for the 'X' letter. The Latin handwritten characters, called **LATIN-*C25***, consist of 26,329 training samples and 11,287 test samples. The set of digits (**LATIN-*D10***), which has less character images, consists of 1,637 training samples and 880 test samples. Sample images of the Latin handwritten dataset are illustrated in Fig. 45.

An overview of the handwritten datasets is given in Table 5. The training set is used for 10-fold cross validation, splitting it according to the 9/1 rule. The test set is an independent hold-out set for an additional final evaluation.



(a) LATIN-*C25* (b) LATIN-*D10*

Figure 45: Some examples of the Latin handwritten dataset. (a) Latin handwritten characters, and (b) Latin handwritten digits.

4.3 Experimental Results

We have compared the IMG method which directly uses pixel intensities to the two local gradient feature descriptors, namely the HOG descriptor and the siftD. The datasets are composed of isolated handwritten characters and digits. The handwritten images are converted to gray-scale and normalized to a fixed-size image. There are two pixel space resolutions which are used in these experiments: 28×28 and 36×36 pixel spaces.

Table 5: Overview of the handwritten datasets.

Dataset	No. of Classes	Train	Test
THI-*C68*	68	13,130	1,360
THI-*D10*	10	8,055	1,500
BANG-*C45*	45	4,627	900
BANG-*D10*	10	9,161	1,500
LATIN-*C25*	25	26,329	11,287
LATIN-*D10*	10	1,637	880

In these experiments, we are using the recognition rate (accuracy) as our evaluation metric to measure the performance of each feature descriptor. For the experiments with *k*NN, the parameter *k* is selected from 1, 3, 5, and 7. For the SVM algorithm, grid-search with a logarithmic scale (Ben-Hur and Weston, 2010) is used to explore the two-dimensional parameter space of the SVM with the RBF kernel. Through grid-search the best combination of two parameters, C and γ, is then selected to create the model of the handwritten recognition systems. We use K-fold cross validation (cv) over the training set to prevent overfitting due to large γ and C parameter values. In this study, we use *K*-fold cross validation with $K = 10$ for both classifiers.

4.3.1 Experiments with the HOG Descriptor

We evaluated the performance of the HOG descriptor using several parameters. The parameters of the HOG descriptor include the block size $b_1 \times b_2$, the cell size $\eta_1 \times \eta_2$ and the number of orientation bins. The cell size parameters are defined as a square ($\eta_1 = \eta_2$). In our experiments we evaluate the use of 9 and 18 orientation bins. In our results, the orientation histogram with 9 bins slightly outperforms 18 bins. Furthermore, several block sizes are evaluated including $b = 3$, 4, 6, 7, and 9, respectively. The tested parameters of the HOG descriptor are illustrated in Table 6.

Table 6: Summary of the HOG descriptor parameters used in the experiments.

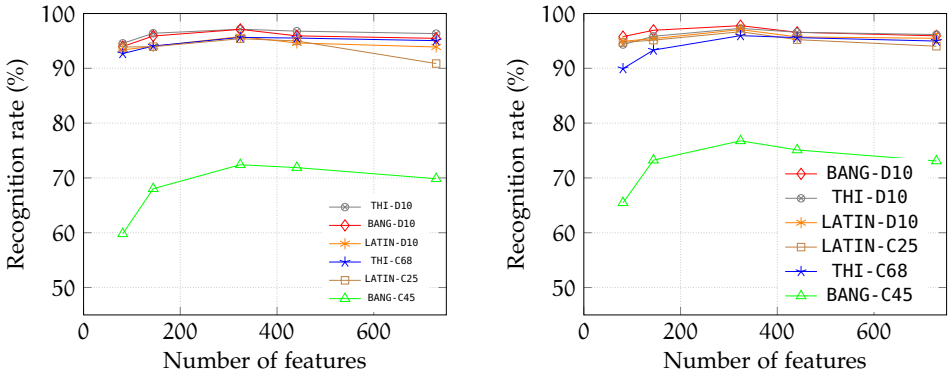
Block size b	Cell size η	Orientation bins β	Pixel space p	Feature dimension N
3×3	12×12	9	36×36	81
4×4	7×7	9	28×28	144
6 × 6	**6 × 6**	**9**	**36 × 36**	**324**
7×7	4×4	9	28×28	441
9×9	4×4	9	36×36	729

The experimental results of the HOG descriptor on the handwritten datasets using the different numbers of feature dimensions (N = 81, 144, 324, 441, and 729) are shown in Fig. 46. We evaluated the HOG descriptor on three handwritten character datasets including Thai (THI-*C68* and THI-*D10*), Bangla (BANG-*C45* and BANG-*D10*), and Latin (LATIN-*C25* and LATIN-*D10*). Fig. 46a shows the performance of the HOG descriptor using *k*NN with $k = 5$, and Fig. 46b using the SVM with the RBF kernel for which standard values are used in this experiment ($C = 1$ and $\gamma = 1/N$).

The results show that the HOG descriptor provides the highest recognition accuracies when the feature vector uses 324 dimensions. The performance is decreased slightly when the feature dimension is increased. In the following experiments, the HOG descriptor uses N = 324 features, given by blocks of size 6×6 in the handwritten character images of size 36×36 with orientation histograms consisting of 9 bins ($6 \times 6 \times 9 = 324$).

4.3.2 Experiments with the SIFT Keypoint Descriptor

For the SIFT keypoint descriptor experiments we evaluated the parameters of siftD. These are the pixel space, the number of keypoints, and the region size. The pixel space resolutions used in this experiments are 28×28 and 36×36 in pixel space. As mentioned before, in order to simplify the



(a) *k*NN (b) SVM

Figure 46: A comparison of the performance (%) of the HOG descriptor on the handwritten datasets using different numbers of features. The experiments use (a) *k*NN with $k = 5$ and (b) SVM with the RBF kernel. Here the RBF kernel parameters of the SVM algorithm are defined as $C = 1$ and $\gamma = 1/N$.

keypoint detection process, we divided the handwritten character image into small blocks (b × b blocks) and defined each center of a block as the keypoint. The evaluated siftD parameters are shown in Table 7.

In these experiments, the number of keypoints is an important factor for obtaining the highest recognition accuracy. We tried out several keypoint numbers: 1 (1×1), 4 (2×2), 9 (3×3), and 16 (4×4). The feature dimensionality is associated with the number of keypoints.

It is important to emphasize that a high dimensionality of the input vector can decrease the recognition performance. Furthermore, the high dimensionality can make the system slow and causes a lot of memory usage during the training process. The results are overall best with 1 keypoint, the only better result obtained with 4 keypoints of siftD ($128 \times 4 = 512$ features) is with the *k*NN classifier for the Bangla character dataset as shown in Fig. 47a and Fig. 47b. It is quite remarkable that the *k*NN and the SVM using siftD with one keypoint (128 features) performs so well as shown in Fig. 47.

Table 7: Summary of the siftD parameters used in the experiments.

Pixel space	Keypoints	Region size	Feature dimension
28×28	1 (1×1)	28×28	128
	4 (2×2)	14×14	512
	9 (3×3)	9×9	1,152
	16 (4×4)	7×7	2,048
36×36	**1(1 × 1)**	**36 × 36**	**128**
	4 (2×2)	18×18	512
	9 (3×3)	12×12	1,152
	16 (4×4)	9×9	2,048

4.3.3 Comparison of HOG and siftD to Pixel Intensities

We compared the IMG method to the local gradient feature descriptors on the challenging handwritten script datasets by using the *k*NN and the SVM as classifiers. The best feature descriptor parameter values from the previous experiments are selected. In contrast to the experiments before, here we have optimized the hyper-parameters of the classifiers. The best parameters found for these experiments are shown in Table 8.

The accuracy results of the *k*NN and the SVM with the RBF kernel are shown in Table 9 and Table 10. The accuracy results of *k*NN are shown in Table 9. The *k*NN algorithm is selected in this chapter, because we found it interesting to observe the performances obtained with a robust feature descriptor and a simple classifier. The performance of the *k*NN reaches above 95%, except on BANG-*C45* using both feature descriptors. The experiments show that while the HOG descriptor performs better on the Latin handwritten dataset with the *k*NN, siftD is more powerful than the HOG descriptor on the other datasets (see Table 9). Most importantly, however, is that the results obtained with the proposed local gradient feature descriptors are much better than those obtained with the direct use of pixel intensities.

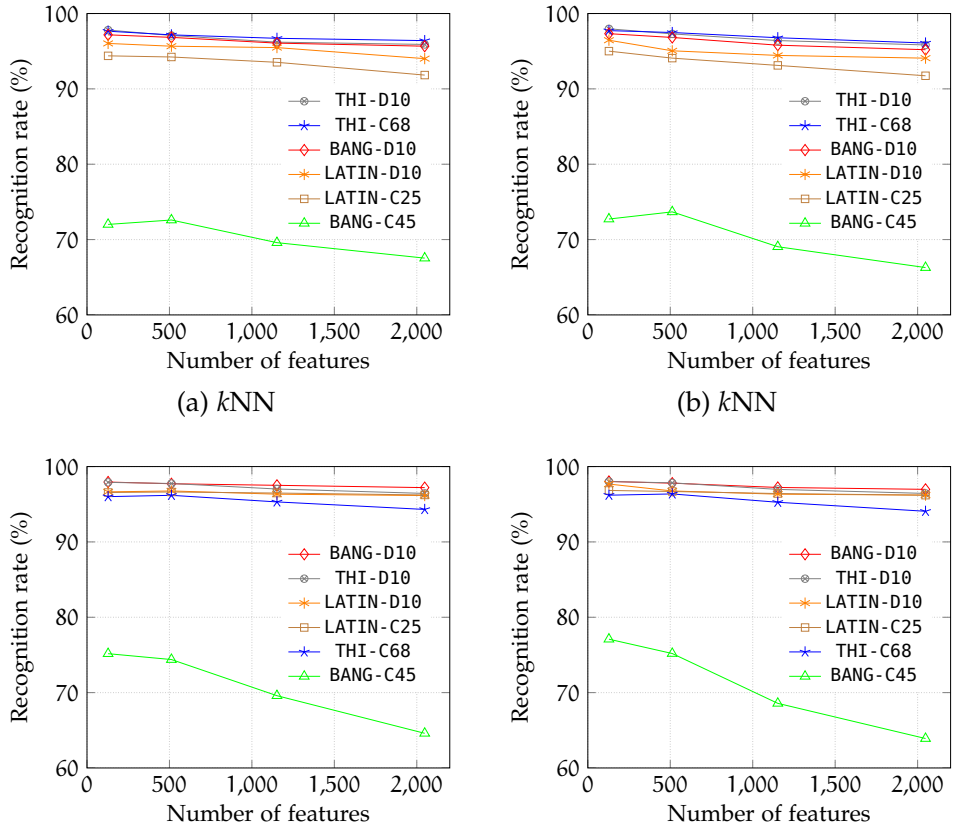


Figure 47: A comparison of the performance (%) of siftD on the handwritten character datasets using different numbers of features and pixel resolutions. The pixel resolutions of the handwritten images are (left column) 28×28 and (right column) 36×36 pixels. The experiments (a), (b) use kNN with $k = 3$ and (c), (d) SVM with the RBF kernel. The RBF kernel parameters of the SVM algorithm are defined as $C = 1$ and $\gamma = 1/N$.

We also compared these results with the kNN classifier to previous results obtained on the handwritten Bangla digit dataset. In (Surinta et al., 2013), the authors presented the unweighted majority voting method (UMV), which combines different SVM classifiers with the RBF kernel trained on different extracted features. This more complex ensemble method obtained 96.8%.

Table 8: The best training parameter values for the SVM with the RBF kernel and for the *k*NN algorithm.

Datasets	IMG		HOG		siftD		IMG	HOG	siftD
	SVM with RBF kernel						*k*NN		
	C	γ	C	γ	C	γ	*k*		
THI-*C68*	2^2	2^{-9}	2^2	2^{-6}	2^2	2^{-5}	1	3	3
THI-*D10*	2^2	2^{-9}	2^2	2^{-6}	2^2	2^{-5}	3	3	3
BANG-*C45*	2^2	2^{-10}	2^2	2^{-6}	2^4	2^{-7}	5	5	5
BANG-*D10*	2^1	2^{-9}	2^1	2^{-6}	2^2	2^{-5}	3	5	5
LATIN-*C25*	2^0	2^{-9}	2^2	2^{-7}	2^4	2^{-5}	1	5	3
LATIN-*D10*	2^0	2^{-9}	2^3	2^{-7}	2^4	2^{-6}	5	5	3

Our current results show that the HOG descriptor and siftD obtain 97.11% and 97.35%, respectively. Because the HOG descriptor and siftD with the *k*NN method provide higher accuracies than the more complex method used in (Surinta et al., 2013), these results demonstrate the effectiveness of the proposed local gradient feature descriptors.

We show the obtained results with the SVM classifier with the RBF kernel on the handwritten character datasets in Table 10. It can be seen from Table 10 that siftD is the best feature descriptor in our experiments on the three handwritten character datasets. The SVM with the RBF kernel outperforms the *k*NN with around 1% to 11% accuracy improvement. On the BANG-*C45* dataset, the SVM with the RBF kernel increased the recognition performance with about 11% compared to the *k*NN classifier.

To summarize the results: the siftD and HOG feature descriptors strongly outperform the direct use of pixel intensities. Even with much less features (siftD computes 128 features for all datasets except for BANG-*C45*), very good results are obtained. Furthermore, the results demonstrate that the SVM significantly outperforms the *k*NN classifier. Finally, we want to mention that the SVM with the siftD method obtains very high recogni-

Table 9: The accuracy (%) and the standard deviation of the *k*NN classifier obtained with cross validation and on separate test sets. The results are computed using three handwritten character datasets.

Datasets	IMG (%)		HOG (%)		siftD (%)	
	10-cv	Test	10-cv	Test	10-cv	Test
THI-*C68*	93.55 ± 0.46	82.87	95.83 ± 0.76	88.31	**97.73** ± 0.44	91.91
THI-*D10*	93.52 ± 0.76	86.47	97.23 ± 0.51	93.73	**97.97** ± 0.50	97.83
BANG-*C45*	53.17 ± 1.96	46.11	72.40 ± 1.90	69.00	**74.50** ± 1.71	69.67
BANG-*D10*	91.05 ± 0.53	89.87	97.11 ± 0.44	95.60	**97.35** ± 0.74	96.07
LATIN-*C25*	88.00 ± 0.96	90.54	**95.40** ± 0.51	95.17	95.01 ± 0.62	96.12
LATIN-*D10*	91.76 ± 1.82	96.25	**97.79** ± 1.20	95.11	96.46 ± 1.25	96.48

tion accuracies. On most datasets, cross validation accuracies around 99% are obtained. However, on the Bangla character dataset the performance of the HOG descriptor and siftD are much lower compared to the other datasets. This might be because of the image quality, the huge diversity in writing styles, the similarities between different characters, arbitrary used tail strokes which makes the definition of the bounding box around the characters harder, a high cursivity, and an insufficient number of handwritten training samples.

4.4 Discussion

In this chapter, we have demonstrated the effectiveness of local gradient feature descriptors for handwritten character recognition. The selected local gradient feature descriptors are siftD and HOG that extract the orientation histograms from the handwritten character gray-scale images. Only simple preprocessing schemes such as rescaling the image by preserving the aspect ratio and converting it from color to gray-scale were applied. We evaluated two machine learning techniques with the feature description methods on three different handwritten character datasets: Thai, Bangla, and Latin. The

Table 10: The SVM accuracy (%) and the standard deviation of handwritten character recognition experiments on the handwritten character datasets.

Datasets	IMG (%)		HOG (%)		siftD (%)	
	10-cv	Test	10-cv	Test	10-cv	Test
THI-*C68*	95.33 ± 0.08	90.59	98.42 ± 0.03	94.34	**98.93** ± 0.03	94.34
THI-*D10*	94.88 ± 0.09	88.53	98.58 ± 0.05	97.20	**99.07** ± 0.03	97.87
BANG-*C45*	63.25 ± 0.28	60.00	84.01 ± 0.33	82.78	**85.60** ± 0.18	85.00
BANG-*D10*	95.10 ± 0.09	94.87	98.73 ± 0.03	98.07	**98.91** ± 0.03	98.53
LATIN-*C25*	96.28 ± 0.03	95.94	97.79 ± 0.04	98.25	**98.23** ± 0.04	98.32
LATIN-*D10*	98.04 ± 0.12	96.36	98.10 ± 0.17	97.73	**98.58** ± 0.09	98.30

results show that the local gradient feature descriptors that convert the handwritten images to feature vectors are strong feature descriptors for handwritten character recognition problems. The siftD and the HOG descriptor give the best performances and significantly outperform the IMG method that directly uses pixel intensities (see Tables 9 and 10). Interestingly, siftD with only one keypoint (128 feature dimensions) outperforms the HOG descriptor (324 feature dimensions). The *k*NN and the SVM classifier have been compared, and the results show that the SVM significantly outperforms the *k*NN classifier.

Our present results outperform the previous results reported by other investigators. However, it is quite difficult to compare the results since all three datasets that we used in this chapter were not used in their works. In one related paper (Nopsuwanchai et al., 2006), their method obtained 92.03% on the ThaiCAM database. In another paper (Mitrpanont and Imprasert, 2011), their method obtained 92.78% on the Thai handwritten character dataset. Our best method obtains 94.34% on the test set and 98.93% with cross validation on the THI-*C68* dataset and 97.87% on the test set and 99.07% with cross validation on the THI-*D10* dataset. So our best method outperforms methods from previous works on the Thai handwritten character dataset.

Future work

We will focus on improvements of the handwritten character recognition performance on the Bangla character dataset. For this, we will study other feature descriptors which could even be more efficient and robust to the variation of the writing styles in this dataset and which can handle a small number of training examples.

5 LOCAL FEATURE DESCRIPTORS AND BAGS OF VISUAL WORDS

In this chapter we propose the use of several feature extraction methods, which have been shown before to perform well for object recognition, for recognizing handwritten characters. These methods are the histogram of oriented gradients (HOG), a bag of visual words using pixel intensity information (BOW), and a bag of visual words using extracted HOG features (HOG-BOW). These feature extraction algorithms are compared to other well-known techniques: principal component analysis, the discrete cosine transform, and the direct use of pixel intensities. The extracted features are given to three different types of support vector machines for classification, namely a linear SVM, an SVM with the RBF kernel, and a linear SVM using L2-regularization. We have evaluated the six different feature descriptors and three SVM classifiers on three different handwritten character datasets: Bangla, Odia, and MNIST. The results show that the HOG-BOW, BOW, and HOG method significantly outperform the other methods. The HOG-BOW method performs best with the L2-regularized SVM and obtains very high recognition accuracies on all three datasets.

This chapter has previously been published in *Proceedings of the International Conference on Engineering Applications of Neural Networks (EANN2015)*

We propose the use of several feature descriptors for handwritten character recognition. Obtaining high accuracies on handwritten character datasets can be difficult due to several factors such as background noise, many different types of handwriting, and an insufficient amount of training examples. Our motivation for this study is to obtain high recognition accuracies for different datasets even when there are not many examples in these datasets. There are currently many character recognition systems which have been tested on the standard benchmark MNIST dataset (LeCun and Cortes, 1998). MNIST consists of isolated handwritten digits with a size of 28×28 pixels and contains 60,000 training images and 10,000 test images. Compared to other handwritten datasets such as the Bangla and Odia character datasets, MNIST is simpler as it contains much more training examples, the diversity of handwritten digits is smaller in MNIST, and the number of digits is much smaller than the number of characters in the Odia and Bangla datasets. Therefore it is not surprising that since the construction of the MNIST dataset a lot of progress on the best test accuracy has been made.

Currently the best approaches for MNIST make use of deep neural network architectures (Schmidhuber, 2015). The deep belief network (DBN) (Hinton et al., 2006) has been investigated for MNIST in (Hinton et al., 2006), where different restricted Boltzmann machines (RBMs) are stacked on top of each other to construct a DBN architecture. Three hidden layers are used where the sizes of each layer are 500, 500, and 2,000 hidden units, respectively. The recognition performance with this method is 98.65% on the MNIST dataset.

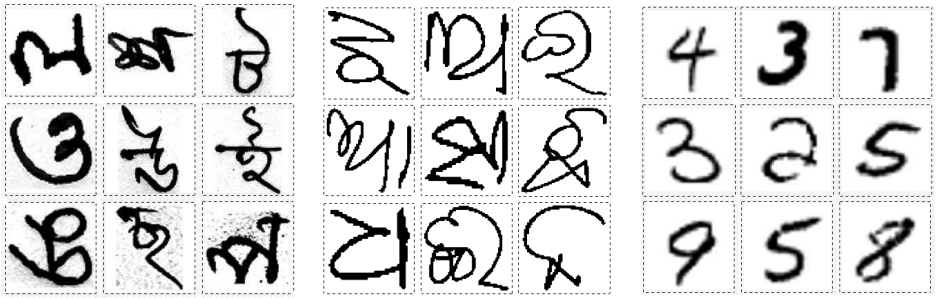
In (Meier et al., 2011), a committee of simple neural networks is proposed for the MNIST dataset, where three different committee types comprising majority, average and median committees are combined. Furthermore, deslanted training images are created by using principal component analysis (PCA) and the elastic deformations are used to create even more training examples. The trained 9-net committees obtained 99.61% accuracy on MNIST. This work has been extended in (Cireşan et al., 2012) where 35 convolutional neural networks are trained and combined using a committee. This approach has obtained an accuracy of on average 99.77%, which

is the best performance on MNIST so far. This technique, however, requires a lot of training data and also takes a huge amount of time for training for which the use of GPUs is mandatory.

Although currently many deep learning architectures are used in the computer vision and machine learning community, in (Coates et al., 2011b) an older method from computer vision, namely the bag of visual words approach (Csurka et al., 2004) was used on the CIFAR-10 dataset and obtained a high recognition accuracy of 79.6%. This simpler method requires much less parameter tuning and much less computational time for training the models compared to deep learning architectures. Also many other feature extraction techniques have been used for different image recognition problems, such as principal component analysis (PCA) (Deepu et al., 2004), restricted Boltzmann machines (Karaaba et al., 2014), and auto-encoders (Hinton and Zemel, 1994).

The cleanliness of MNIST and lack of variation may make MNIST a bad reference for selecting feature extractor techniques that are suitable for Asian scripts. Different feature extraction methods have been used for the Bangla and Odia handwritten character datasets. In (Hossain et al., 2011), the celled projection method is proposed. The recognition performance obtained on the Bangla digit dataset was 94.12%. In (Surinta et al., 2013), the image pixel-based method (IMG) which uses directly the intensities of pixels of the ink trace is used. The IMG is shown to be a quite powerful method (Surinta et al., 2013) when the training set size is increased and obtained a recognition accuracy of 96.4% on the Bangla digit dataset.

The use of local gradient feature descriptors has been investigated for three different handwritten datasets (Thai, Bangla, and Latin) in (Surinta et al., 2015b). High accuracies are obtained on the Thai handwritten datasets (character and digit), the Latin handwritten datasets (character and digit), and on the Bangla handwritten digit dataset, when the feature descriptors are combined with a support vector machine (SVM) (Vapnik, 1998).



(a) (b) (c)

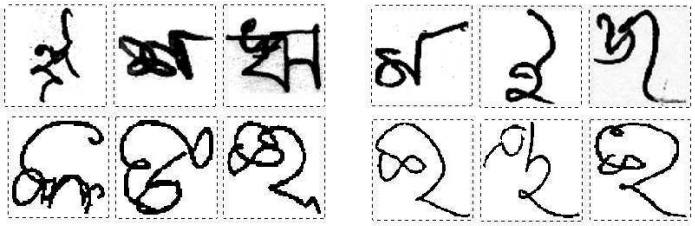
Figure 48: Illustration of the handwritten character datasets. (a) Some examples of Bangla and (b) Odia handwritten characters and (c) MNIST handwritten digits.

Contributions

In this chapter, we propose the use of histograms of oriented gradients (HOG), bags of visual words using pixel intensities (BOW), and bags of visual words using HOG (HOG-BOW) for recognizing handwritten characters. These methods are compared to the direct use of pixel intensities, the discrete cosine transform (DCT), and PCA on three datasets, namely Bangla, Odia, and MNIST, shown in Fig. 48. There are some challenges in the Bangla and Odia character datasets, such as the writing styles (e.g., heavy cursivity and arbitrary tail strokes), as shown in Fig. 49, similar structures of different characters, background noise, and a lack of a large amount of handwritten character samples. We have evaluated the six feature extraction techniques with three types of SVM (Vapnik, 1998) as a classifier, namely a linear SVM, an SVM with a radial basis function (RBF) kernel, and a linear SVM with L2-norm regularization (L2-SVM).

The results show that the HOG-BOW method obtains the highest accuracies on the three handwritten datasets. Also the HOG and BOW feature descriptors work much better than the more traditional techniques such as PCA, DCT, and the direct use of pixel intensities. The results also show a very high performance of HOG-BOW with the L2-SVM on the MNIST dataset. Its recognition performance is 99.43% without the use of elastic

distortions to increase the dataset, without the use of ensemble learning techniques, and without the need for a large amount of training time.



(a) (b)

Figure 49: Illustration of the Bangla and Odia handwritten characters, in the first and second row, respectively. Some examples of (a) heavy cursive and (b) arbitrary tail strokes writing styles.

It should be noted that the HOG descriptor that extracts feature vectors from the handwritten character images are described in Chapter 4. And the SVM classifier is described in Chapter 3.

5.1 Feature Extraction Methods

We study different kinds of feature extraction techniques to deal with the handwritten character datasets as described below.

5.1.1 Principal Component Analysis (PCA)

PCA is a well-known dimensionality reduction method, which extracts an effective number of uncorrelated variables (called 'principal components') from high-dimensional correlated variables (input data). In fact, a small number of principal components is sufficient to represent the actual data (Shi et al., 2003). Here, eigenvectors are computed from the training data which are used as a model which is applied on an image to compute the

feature vectors (Karaaba et al., 2014). After conducting preliminary experiments, we have selected 80 eigenvectors for this approach.

5.1.2 Discrete Cosine Transform (DCT)

The DCT technique transforms the data from a spatial domain (image) into frequency components using only a cosine function. We use 2D-DCT in our experiments since 2D-DCT is more suitable for 2D image data. Here, the highest coefficient values are stored in the upper left and the lowest valued coefficients are stored in the bottom right of the output array (Lawgali et al., 2011). The highest coefficient values are extracted in a zigzag form (Mishra et al., 2013) and then represented as feature vectors. In the experiment, 60 coefficient values were selected in the feature vectors.

5.1.3 Bag of Visual Words with Pixel Intensities (BOW)

The bag of visual words (Csurka et al., 2004) has been widely used in computer vision research. In this approach, local patches that contain local information of the image are extracted and used as a feature vector. Then, a codebook is constructed by using an unsupervised clustering algorithm. In (Abdullah et al., 2010a), some novel visual keyword descriptors for image categorization are proposed and it was shown that the soft assignment schemes outperform the more traditional hard assignment methods. In (Coates et al., 2011b), it was shown that the BOW method outperformed other feature learning methods such as RBMs and auto-encoders. In (Coates et al., 2011a), the method was applied to text detection and character recognition in scene images. We will now explain the BOW method consisting of patch extraction, codebook computation, and feature extraction. As an unsupervised learning based method, K-means clustering performs very fast compared to other unsupervised learning methods such as RBMs (Karaaba et al., 2014) and autoencoders (Hinton et al., 2006). The BOW method works as follows:

Extracting Patches from the Training Data

The sub-image patches X are extracted randomly from the unlabeled training images, $X = \{x_1, x_2, ..., x_N\}$ where $x_k \in \mathbf{R}^p$ and N is the number of random patches. The size of each patch is defined as a square with $(p = w \times w)$ pixels. In our experiments we used $w = 15$, meaning 15×15 pixel windows are used.

Construction of the Codebook

The codebook is constructed by clustering the vectors obtained by randomly selecting patches. Here, the codebook C is computed by using the *K*-means clustering method on pixel intensity information contained in each patch. Let $C = \{c_1, c_2, ..., c_K\}, c \in \mathbf{R}^p$ represent the codebook (Ye et al., 2012), where *K* is the number of centroids. In our experiments we used 400,000 randomly selected patches to compute the codebooks.

Feature Extraction

To create the feature vectors for training and testing images, the soft-assignment coding scheme from Coates et al. (2011b) is used. This approach uses the following equation to compute the activity of each cluster given a feature vector x from a patch:

$$i_k(x) = max\{0, \mu(s) - s_k\} \tag{20}$$

where $s_k = \|x - c_k\|_2$ and $\mu(s)$ is the mean of the elements of s (Coates et al., 2011b).

The image is split into four quadrants and the activities of each cluster for each patch in a quadrant are summed up. We use a sliding window on the train and test images to extract the patches. Because the stride is 1 pixel and the window size is 15×15 pixels, the method extracts 484 patches from each image to compute the cluster activations. The feature vector size is $K \times 4$ and because we use $K = 600$ clusters, the feature vectors for the BOW method have 2,400 dimensions.

5.1.4 Bag of Visual Words with HOG Features (HOG-BOW)

In the previous BOW method, the intensity values in each patch are extracted and used for clustering. With HOG-BOW, however, feature vectors from patches are computed by using the state-of-the-art HOG descriptor (Dalal and Triggs, 2005), and then these feature vectors are used to compute the codebook and the cluster activities. The HOG descriptor captures the gradient structure of the local shape and may provide more robust features. In this experiment, the best HOG parameters used 36 rectangular blocks and 9 orientation bins to compute feature vectors from each patch. As in BOW, HOG-BOW uses 4 quadrants and 600 centroids, yielding a 2,400 dimensional feature vector.

5.2 Handwritten Character Datasets and Pre-Processing

The handwritten character images in the Bangla and Odia datasets were scanned into digital images at different pixel resolutions. The details of the handwritten character datasets and pre-processing steps will now be described.

5.2.1 Bangla Character Dataset

The Bangla basic character consists of 11 vowels and 39 consonants (Bhowmik et al., 2009). In the experiment, the dataset includes 45 classes and contains 5,527 character images. The dataset is divided into training and test sets, containing 4,627 and 900 samples, respectively. Samples of the Bangla characters are shown in Fig 48a.

5.2.2 Odia Character Dataset

The Odia handwritten dataset was collected from 50 writers using a take-note device. This dataset consists of 47 classes, 4,042 training and 987 test samples. Some examples of the Odia characters are shown in Fig. 48b.

5.2.3 MNIST Dataset

The standard MNIST dataset (LeCun and Cortes, 1998) is a subset of the NIST dataset. The handwritten images were normalized to fit into 28×28 pixels. The anti-aliasing technique is used while normalizing the image. The handwritten images of the MNIST dataset contain gray levels. The dataset contains 60,000 handwritten training images and 10,000 handwritten test images, see Fig. 48c for some examples.

An overview of the handwritten datasets is given in Table 11.

Table 11: Overview of the handwritten character datasets

Dataset	Color format	No. of writers	No. of classes	Train	Test
Bangla character	Grayscale	Multi	45	4,627	900
Odia character	Binary	50	47	4,042	987
MNIST	Grayscale	250	10	60,000	10,000

5.2.4 Dataset Pre-processing

In order to prepare the handwritten character images from the Bangla and Odia datasets, a few pre-processing steps which include background removal, basic image morphological operations and image normalization are employed. First, the Bangla handwritten dataset contains different kinds of backgrounds and is stored in gray-scale images. On the other hand, the

Odia handwritten dataset is stored in binary image format as shown in Fig 48b. Hence, the background removal is applied only to the Bangla handwritten dataset. In this study, due to its simplicity and yet robustness feature, we selected Otsu's algorithm (Otsu, 1979) for removing background noise and making a binary image. Some examples of noisy background images are shown in Fig 50.

Figure 50: Illustration of the noisy background which appears in the Bangla handwritten dataset.

Next, a basic morphological dilation operation is applied to the binary handwritten images from the previous step. The dilation operation enlarges pixels to the boundaries of the handwritten image I by the structure element H given by $I \oplus H = \{(p + q) \,|\, p \in I, q \in H\}$. Here, the structure element uses size 3×1.

Finally, many researchers investigated the effect of scale differences for handwritten character recognition (Meier et al., 2011). In those studies, scales from 20×20 (minimum) to 40×40 (maximum) pixel resolutions are compared. In this study, we normalize the handwritten image into 36×36 pixels with the aspect ratio preserved. The pre-processing stage of a handwritten character image is shown in Fig. 51.

5.3 Experimental Results

We evaluated the feature extraction techniques on the Bangla, Odia and MNIST datasets by using three different SVM algorithms (Vapnik, 1998).

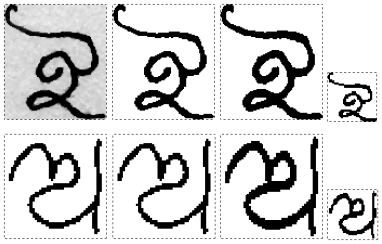


Figure 51: Illustration of the image pre-processing steps which are used in this experiments. The first row shows a Bangla character and the second row shows an Odia character. The original image of the handwritten characters (first column), the image after applying the background removal method (second column), the dilation images (third column) and the image normalization (fourth column).

We used an SVM with a linear kernel, an SVM with an RBF kernel, and a linear L2-regularized SVM (L2-SVM) (Fan et al., 2008). The results are shown in Table 12, Table 13, and Table 14. In each table we show the results of the feature extraction techniques with a different SVM. In all the tables, the results show that the HOG-BOW, the HOG and the BOW method significantly outperform the other methods. Furthermore, on all 9 experiments the HOG-BOW method performs best (highly significant differences according to a student's t-test are indicated in boldface). Importantly, some of the HOG-BOW results in Table 12 and Table 13 showed a higher accuracy on the test set than on the cross validation runs on the training set. This can be explained by the reason that to compute the accuracy on the final test set more training examples are used than when cross validation is used.

In terms of SVM algorithms, we can see different results. Here, the linear SVM obtains a worse performance compared to the SVM with the RBF kernel. The linear SVM seems, however, to better handle low dimensional input vectors, compared to the L2-SVM. The L2-SVM yields significantly better results if high dimensional feature vectors (Fan et al., 2008) are used such as with the HOG-BOW and the BOW method. In fact, the best results have been achieved with the L2-SVM with the HOG-BOW method. It is followed by the BOW and the HOG method, respectively. The HOG method

Table 12: Results of training (10-fold cross validation with the standard deviation) and testing recognition performances (%) of the feature descriptors when combined with the linear SVM.

Algorithms	Bangla dataset		Odia dataset		MNIST dataset	
	10-cv	Test	10-cv	Test	10-cv	Test
PCA	54.87 ± 0.20	53.67	56.57 ± 0.32	53.60	93.29 ± 0.02	92.69
DCT	59.33 ± 0.32	52.33	60.77 ± 0.40	54.81	92.51 ± 0.06	91.32
IMG	56.25 ± 0.22	54.33	56.12 ± 0.57	56.23	94.13 ± 0.05	94.58
BOW	77.96 ± 0.21	77.17	79.30 ± 0.34	78.01	98.71 ± 0.02	98.47
HOG	81.17 ± 0.30	80.11	79.86 ± 0.20	80.45	98.62 ± 0.01	99.11
HOG-BOW	**82.07 ± 0.24**	82.44	**81.74 ± 0.49**	82.43	**99.09 ± 0.03**	99.16

outperforms the BOW method when using the SVM with an RBF kernel (see Table 13). The feature vector size of each feature extraction technique is shown in Table 14.

5.4 Conclusion

In this chapter, we have demonstrated the effectiveness of different feature extraction techniques from computer vision for handwritten character recognition. We have shown that the HOG-BOW method combined with an L2-regularized SVM outperforms all other methods. The obtained accuracies with this method can be considered very high. On the MNIST dataset for example, HOG-BOW combined with the L2-regularized SVM obtains a recognition accuracy on the test set of 99.43% which is a state-of-the-art performance. The best method for MNIST (Cireşan et al., 2012) uses an ensemble of 35 convolutional neural networks and elastic deformations to increase the dataset and obtains around 99.77% accuracy. The proposed HOG-BOW method, however, is much faster, needs less training data and we have not yet evaluated its performance in an ensemble of different classifiers.

Future work

We want to research different ways to improve the HOG-BOW even more. We are interested in examining other soft assignment coding schemes to compute cluster activities and we also want to construct an ensemble method to obtain even higher accuracies.

Table 13: Results of training (10-fold cross validation with the standard deviation) and testing recognition performances (%) of the feature descriptors when combined with the SVM with the RBF kernel.

Algorithms	Bangla dataset		Odia dataset		MNIST dataset	
	10-cv	Test	10-cv	Test	10-cv	Test
IMG	63.25 ± 0.28	60.00	57.95 ± 0.42	60.28	96.95 ± 0.02	97.27
PCA	64.08 ± 0.30	61.11	60.57 ± 0.57	59.87	96.86 ± 0.02	96.64
DCT	70.18 ± 0.27	61.33	69.91 ± 0.34	63.63	98.18 ± 0.09	97.51
BOW	78.76 ± 0.38	77.17	81.29 ± 0.42	80.65	98.98 ± 0.01	98.97
HOG	83.11 ± 0.25	83.00	82.16 ± 0.27	83.38	99.13 ± 0.01	99.12
HOG-BOW	83.14 ± 0.18	83.33	**83.62 ± 0.17**	83.56	**99.30 ± 0.02**	99.35

Table 14: Results of recognition performances (%) of the methods when used with the L2-SVM.

Algorithms	Feature dimensionality	Handwritten character dataset		
		Test Bangla	Test Odia	Test MNIST
DCT	60	51.67	56.94	90.84
PCA	80	50.33	53.90	91.02
IMG	1,296	31.33	42.65	91.53
HOG	324	74.89	74.27	98.53
BOW	2,400	86.56	84.60	99.10
HOG-BOW	2,400	87.22	85.61	99.43

6 DISCUSSION

The objective of the research that we have described in this dissertation is getting a better understanding of features in order to provide accurate character recognition with reduced complexity and less training data. *Firstly*, in terms of document layout analysis, we have focused on a well-known path-planning algorithm as a line segmentation technique that is applied to several different handwritten document manuscripts. *Secondly*, robust feature extraction and machine learning techniques are proposed for recognizing handwritten characters.

In this dissertation, we have demonstrated that our algorithms (e.g. line segmentation and feature extraction techniques) are very efficient for improving a handwritten character recognition system. Furthermore, some state-of-the-art machine learning algorithms that can be used to improve the performance of handwritten character recognition are proposed. We will now briefly discuss the challenges of the character recognition systems and our findings.

In Chapter 2, due to the problems of overlapping and touching characters of subsequent lines, the well-known A^* path-planning algorithm cannot solve these kinds of problems. This original algorithm is effective only when the components of two lines do not overlap. For this reason, several simple soft cost functions are combined to enhance the performance of the A^* path-planning algorithm. The five simple cost functions which are combined with the path-planning algorithm are the ink distance, square ink distance, the map-obstacle, the vertical and the neighbor cost functions. These cost functions allow an artificial agent to calculate the best possible path that

separate the upper and lower text areas. We have performed experiments on two historical manuscripts, namely Saint Gall and Monk line segmentation (MLS) datasets.

The experimental results show that our A^* path-planning algorithm performs well on the Saint Gall dataset and also on the more complicated MLS dataset. In some cases the method was observed to cut some text instead of going up with a curved text line though.

In Chapter 3, we introduced two feature extraction techniques, namely *the hotspot technique (HOT)* and *the contour angular technique (CAT)*, which are computed based on thinned character images. We have evaluated these feature extraction techniques on three different handwritten scripts including Thai character, Bangla digit, and MNIST. As classifiers, the *k*-nearest neighbor (*k*NN) and the support vector machine (SVM) algorithms are used.

We separated our experiments into two parts according to the classifiers. These are experiments with *k*NN and SVM algorithms. For *k*NN, we presented the HOT technique to compare to the other feature extraction techniques including the mark direction and the direction of the chain code technique. In this experiment, the HOT technique significantly outperformed the other feature extraction techniques on the Bangla digit and MNIST datasets. However, the mark direction technique outperforms the HOT technique on the Thai character dataset.

Furthermore, we proposed the CAT technique for the isolated handwritten Bangla digit dataset. It was compared to other methods using the SVM as classifier, such as the HOT technique, the gray-level pixel-based method (GPB), and a basic low-resolution pixel-based method. We have learned that the CAT technique computes aspects of curvature of the handwritten image using 8-directional codes, while counting the co-occurrences of angles along the ink trace. The experiments showed that the CAT feature significantly outperformed the GPB method under the condition of a small amount of training data and much faster character classification than the pixel-based methods. Finally, the accuracy performance of the handwritten character recognition system is increased when the unweighted majority vote method (UMV) is used to combine the output from different SVM classifiers.

In Chapter 4, we improved the accuracy performance for handwritten character recognition by studying local gradient feature descriptors. We compared two existing feature descriptors, namely the histogram of oriented gradients (HOG) and the scale invariant feature transform (SIFT) descriptor. In addition, these two local gradient feature descriptors are compared to the well-known and widely used pixel-based method (IMG).

We should notice that although these two orientation based local feature descriptors somewhat resemble each other, there are still a number of differences. For example, the HOG descriptor uses simple convolution kernels such as the Sobel kernel and the SIFT descriptor uses a Gaussian filter before extracting the gradient orientations and magnitudes. Also, due to the gradient orientations, the HOG descriptor uses absolute angles between 0 and 180 degrees and the SIFT descriptor uses all angles between 0 and 360 degrees. The SIFT and the HOG descriptor gave the best performances and significantly outperformed the IMG method.

As for classifiers, the SVM algorithm performs better than the kNN algorithm. However, it is interesting that these robust local descriptors show a good performance even with the simple kNN algorithm.

In Chapter 5, there are some challenges in the handwritten character datasets to be handled, especially, in the Bangla and Odia datasets. With these datasets, we have to handle the diversity of writing styles (e.g., heavy cursivity and arbitrary tail strokes), similar structures of different characters, and cannot use a large amount of handwritten character samples. To address these problems, the bag of visual words with HOG features (HOG-BOW) was proposed. This is because, the state-of-the-art HOG descriptor captures the gradient structure of the local shape and provides more robust features than using pixel intensities.

In this approach, local patches of the unlabeled training images are randomly extracted and processed by using the HOG descriptor. Then, the K-means clustering algorithm is applied to construct a codebook. Consequently, the feature vectors for training and test images are extracted by using the soft-assignment coding scheme and given to the SVM algorithm for classification.

The L2-regularization (L2-SVM) algorithm yields significantly better results than other SVM models if high dimensional feature vectors are used. The experiments showed that the HOG-BOW method combined with the L2-SVM classifier obtained very high accuracy performances on the Bangla and Odia handwritten character datasets.

In this dissertation, the problem found during the experiment of the line segmentation is that the projection profile analysis which is used to locate the text line could not cope with the slant text line. Additionally, sometimes the method prefers to cut some characters instead of moving into the correct directions.

Furthermore, two local gradient feature descriptors (HOG and siftD) that extract feature vectors from the multi-script handwritten character datasets are used. As a classifier, the simple *k*NN algorithm and the SVM algorithm have been used. Surprisingly, the *K*NN algorithm obtains a good performance when combined with these robust local gradient descriptors. The results show that high accuracies are obtained when using the SVM algorithm as a classifier. However, these two feature descriptors combined with the SVM algorithm cannot manage with a small amount of training data.

However, the HOG-BOW method is proposed to handle some challenges in Bangla and Odia handwritten character datasets, such as a small number of character data, the diversity of writing styles, and similar structures of different characters. As a result, the HOG-BOW method combined with the L2-SVM classifier achieves high accuracy performances on both handwritten character datasets.

Two concrete approaches include line segmentation and character recognition are proposed for the multi-script handwritten character recognition system. However, the process of word and character segmentation is required to build a fully functional document recognition system. These are explained in the section of future work.

6.1 Answers to the Research Questions

According to Chapter 1, three research questions (**RQ**) concern the improvement of current character recognition systems. In this section, we briefly recall each question and provide some answers to them.

RQ1 We focus on the task of segmenting the text areas from the handwritten document images. Here, we propose a novel line segmentation method for handwritten historical documents which are written in one column. The following questions come up; Is it possible to find potential candidates for starting points of lines separating upper and lower text areas of handwritten documents? What is the best technique that can separate two lines of handwritten text which are connected or overlapping?

In answering the *RQ1*, we first detect the locations of text lines from the historical handwritten documents. Text line localization is performed in two steps: binarization (Otsu, 1979; Sauvola and Pietikäinen, 2000) and projection profile analysis (Ha et al., 1995; Marinai and Nesi, 1999). *Firstly*, the historical handwritten document images are binarized using a local binarization technique (Sauvola and Pietikäinen, 2000). With this step, the background noise is removed by the local binarization technique. *Secondly*, the horizontal projection profile analysis is carried out by taking the density of the black pixel values in the corresponding row. Then, the local maxima values are determined as the candidate starting points for line segmentation.

We propose a line segmentation technique which is based on the well-known A^* path-planning algorithm (Nilsson, 1982). our approach uses several simple soft cost functions for improving the path-planning algorithm. Consequently, the soft cost functions allow a separating path to get close to text areas and even pass through the connected text areas.

RQ2 Very high accuracy is required for a character recognition system for being applied in real-world applications. However, there are difficult problems of the challenging handwritten scripts such as the similarities between

the character sets and the similar structures between different characters. These factors negatively affect the performance of a handwritten character recognition system. What are the most robust local feature descriptors that can improve the recognition performance? Furthermore, when robust feature vectors are created, how do machine learning techniques perform for recognizing more complex handwritten characters? Can we assume that a simple machine learning technique such as the *k*NN algorithm provides high accuracy results?

The *RQ2* is mainly about obtaining a high accuracy in the case of the challenging handwritten scripts (Thai, Bangla, and Latin). To answer this research question, we propose to use two effective local gradient feature descriptors, namely the *HOG* and the scale invariant feature transform descriptor (*siftD*) to make a better character recognition system. These feature descriptors provide robust feature vectors which are invariant to small local displacements. After the feature descriptor process, the feature vectors are classified by the machine learning techniques. Then, we first start our investigation by focusing on the SVM algorithm (Vapnik, 1998). According to the experiments, however, the results show that siftD with the SVM classifier using the radial basis function (RBF) kernel outperforms the HOG descriptor. We want to mention that the siftD obtains very high recognition accuracies which makes it the best local feature descriptor for our experiments.

We also performed an experiment in which the feature vectors are classified by the *k*NN algorithm which is a simple non-parametric algorithm. The efficacy of the *k*NN algorithm depends on the distance function and the value of the parameter *k* (Hendrickx and van den Bosch, 2005). Consequently, by using both local feature descriptors the accuracy performance of the *k*NN is above 95% except on the Bangla handwritten character dataset. This is basically because of heavy cursive and arbitrary tail stroke writing styles, and a lack of a large amount of handwritten character samples.

In addition, our solution to these problems are using a bag of visual words method. We will answer the question in the *RQ3*.

RQ3 It is not a straightforward task to obtain high accuracies on challenging handwritten character datasets due to various factors. These factors include an insufficient amount of training samples and many different types of handwriting. How can we manage these issues? How do local feature descriptors and bag of visual words (BOW) perform for recognizing more complex handwritten characters? Furthermore, if we use the SVM algorithm for classification, which kernel type provides the best recognition accuracies on the challenging handwritten character datasets?

We have evaluated the feature descriptors on three different handwritten character datasets: Bangla, Odia, and MNIST. Especially, in Bangla and Odia datasets, we have to deal with a small amount of character images for training and need to handle various handwriting styles. To address this problem, we propose a new local descriptor, namely *HOG-BOW*. With the HOG-BOW descriptor, features of randomly local patches created from the training samples are computed by using the state-of-the-art HOG descriptor. We emphasize that increasing the number of random patches is one of many factors that can effect the performance of a handwritten character recognition system. Then, K-means clustering is applied to construct a codebook. Using the codebook, consequently, the feature vectors are extracted and given to the SVM algorithm for classification. The results show that the HOG-BOW method significantly outperforms the other state-of-the-art methods, such as HOG and BOW separately.

To answer the last question in *RQ3*, the feature vectors are given to three different types of SVMs including: a linear SVM, an SVM with the RBF kernel and a linear SVM using L2 regularization. In our comparison, the results clearly show that the HOG-BOW, BOW, and HOG feature descriptor methods significantly outperform the other methods. Eventually, the best performing method that obtains very high recognition accuracies on three datasets is the HOG-BOW combined with the L2-SVM classifier.

6.2 Future Work

The research presented in this dissertation motivates a diversity of future research projects, most of which are about improving the performance of handwritten character recognition by adding new methods or optimizing the existing methods.

Accordingly, further research is primarily focused on developing robust feature extraction techniques for the handwritten character recognition problem. In recent years, for instance, deep learning architectures (Bengio, 2009) are becoming more and more popular and have been applied to different research areas (Lee et al., 2007; Gregor and LeCun, 2010). In this approach, the learning algorithms (e.g. deep belief networks, auto-encoders, and convolutional neural networks) are proposed for learning features from very large and high-dimensional data. To evaluate the efficiency of this approach, these algorithms have been applied to character recognition and obtained high recognition performances on the benchmark MNIST dataset. These learning algorithms, however, require large training sets to make a recognition system work well. We plan to work on improving the HOG-BOW method to obtain higher accuracies as well as optimize the operator that computes cluster activities of each cluster. We are interested in combining the convolution neural network with the BOW method (CNN-BOW) as well.

In case of insufficient training data, we focussed on the recognition of Bangla and Odia handwritten character datasets. The reason is that there are some challenges; for example, a small amount of training data, different writing styles, and even similar structures of different characters. As for accuracy performance, however, the results were not high enough. There is still much work to be done to improve learning algorithms for these kinds of datasets. A promising way would be to use the multi-layer support vector machines (Wiering and Schomaker, 2014) together with HOG-BOW.

The process of the line segmentation is also important for the handwritten character recognition systems. To improve the accuracy of the line segmentation, some page segmentation techniques will be required to handle the document image before applying the line segmentation technique. In or-

der to correct a problem with line segmentation, character recognition and multiple hypothesis testing will be used concurrently. Additionally, there are some problems in word segmentation of the Thai language. This is because people usually write the sentence without any space between each word, while some people leave space between conjunctions or phase words. To optimize the Thai word segmentation, the process of lexicon and word recognition which divides text lines into meaningful words are required. All of these are thus seen as interesting directions for future work.

BIBLIOGRAPHY

Abdullah, A. B., Veltkamp, R. C., and Wiering, M. A. (2009). Spatial pyramids and two-layer stacking SVM classifiers for image categorization: A comparative study. In *Neural Networks (IJCNN), The international Joint Conference on*, pages 5–12.

Abdullah, A. B., Veltkamp, R. C., and Wiering, M. A. (2010a). Ensembles of novel visual keywords descriptors for image categorization. In *Control Automation Robotics Vision (ICARCV), The 11th International Conference on*, pages 1206–1211.

Abdullah, A. B., Veltkamp, R. C., and Wiering, M. A. (2010b). Fixed partitioning and salient points with MPEG-7 cluster correlograms for image categorization. *Pattern Recognition*, 43(3):650–662.

Arica, N. and Yarman-Vural, F. T. (2001). An overview of character recognition focused on off-line handwriting. *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, 31(2):216–233.

Arica, N. and Yarman-Vural, F. T. (2002). Optical character recognition for cursive handwriting. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 24(6):801–813.

Arivazhagan, M., Srinivasan, H., and Srihari, S. (2007). A statistical approach to line segmentation in handwritten documents. In *Document Recognition and Retrieval XIV (DRR), The 14th Conference on*, volume 6500, pages 65000T–65000T–11.

Arróspide, J., Salgado, L., and Camplani, M. (2013). Image-based on-road vehicle detection using cost-effective histograms of oriented gradients. *Journal of Visual Communication and Image Representation*, 24(7):1182–1190.

Arvanitopoulos, N. and Susstrunk, S. (2014). Seam carving for text line extraction on color and grayscale historical manuscripts. In *Frontiers in Handwriting Recognition (ICFHR), 14th International Conference on*, pages 726–731.

Badekas, E. and Papamarkos, N. (2007). Optimal combination of document binarization techniques using a self-organizing map neural network. *Engineering Applications of Artificial Intelligence*, 20(1):11–24.

Baechler, M., Liwicki, M., and Ingold, R. (2013). Text line extraction using DMLP classifiers for historical manuscripts. In *Document Analysis and Recognition (ICDAR), The 12th International Conference on*, pages 1029–1033.

Basu, S., Das, N., Sarkar, R., Kundu, M., Nasipuri, M., and Basu, D. K. (2005). An MLP based approach for recognition of handwritten Bangla numerals. In *Artificial Intelligence (IICAI), The 2nd Indian International Conference on*, pages 407–417.

Ben-Hur, A. and Weston, J. (2010). A user's guide to support vector machines. In Carugo, O. and Eisenhaber, F., editors, *Data Mining Techniques for the Life Sciences*, volume 609 of *Methods in Molecular Biology*, pages 223–239. Humana Press.

Bengio, Y. (2009). Learning deep architectures for AI. *Foundations and Trends® in Machine Learning*, 2(1):1–127.

Bhowmik, T. K., Ghanty, P., Roy, A., and Parui, S. (2009). SVM-based hierarchical architectures for handwritten Bangla character recognition. *International Journal on Document Analysis and Recognition (IJDAR)*, 12(2):97–108.

Bhowmik, T. K., Parui, S., Kar, M., and Roy, U. (2007). HMM parameter estimation with genetic algorithm for handwritten word recognition. In Ghosh, A., De, R., and Pal, S., editors, *Pattern Recognition and Machine Intelligence*, volume 4815 of *Lecture Notes in Computer Science*, pages 536–544. Springer Berlin / Heidelberg.

Blumenstein, M. (2008). Cursive character segmentation using neural network techniques. In Marinai, S. and Fujisawa, H., editors, *Machine Learning in Document Analysis and Recognition*, volume 90 of *Studies in Computational Intelligence*, pages 259–275. Springer Berlin Heidelberg.

Blumenstein, M., Verma, B., and Basli, H. (2003). A novel feature extraction technique for the recognition of segmented handwritten characters. In *Document Analysis and Recognition (ICDAR), The 7th International Conference on*, pages 137–141. IEEE Computer Society.

Brink, A. (2011). *Robust and applicable handwriting biometrics*. PhD thesis, Institute of Artificial Intelligence and Cognitive Engineering (ALICE), Groningen, The Netherlands.

Brink, A. A., Smit, J., Bulacu, M. L., and Schomaker, L. R. B. (2012). Writer identification using directional ink-trace width measurements. *Pattern Recognition*, 45(1):162–171.

Bukhari, S. S., Shafait, F., and Breuel, T. M. (2011). High performance layout analysis of Arabic and Urdu document images. In *Document Analysis and Recognition (ICDAR), The 11th International Conference on*, pages 1275–1279.

Bulacu, M., Brink, A., van der Zant, T., and Schomaker, L. R. B. (2009). Recognition of handwritten numerical fields in a large single-writer historical collection. In *Document Analysis and Recognition (ICDAR), The 10th International Conference on*, pages 808–812.

Bulacu, M., van Koert, R., Schomaker, L. R. B., and van der Zant, T. (2007). Layout analysis of handwritten historical documents for searching the archive of the cabinet of the Dutch queen. In *Document Analysis and Recognition (ICDAR), The 9th International Conference on*, volume 1, pages 357–361.

Bulacu, M. L. (2007). *Statistical Pattern Recognition for Automatic Writer Identification and Verification*. PhD thesis, Institute of Artificial Intelligence and Cognitive Engineering (ALICE), Groningen, The Netherlands.

Bulacu, M. L. and Schomaker, L. R. B. (2007). Text-independent writer identification and verification using textural and allographic features. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 29(4):701–717.

Bunke, H. and Riesen, K. (2011). Recent advances in graph-based pattern recognition with applications in document analysis. *Pattern Recognition*, 44(5):1057–1067.

Chamchong, R., Fung, C., and Wong, K. W. (2010). Comparing binarisation techniques for the processing of ancient manuscripts. In Nakatsu, R., Tosa, N., Naghdy, F., Wong, K. W., and Codognet, P., editors, *Cultural Computing*, volume 333 of *IFIP Advances in Information and Communication Technology*, pages 55–64. Springer Berlin Heidelberg.

Chamchong, R. and Fung, C. C. (2012). Text line extraction using adaptive partial projection for palm leaf manuscripts from Thailand. In *Frontiers in Handwriting Recognition (ICFHR), The 13th International Conference on*, pages 588–593.

Chaudhuri, B. B. (2006). A complete handwritten numeral database of Bangla - a major Indic script. In *Frontiers in Handwriting Recognition (IWFHR), The 10th International Workshop on*.

Chen, K., Seuret, M., Liwicki, M., Hennebert, J., and Ingold, R. (2015). Page segmentation of historical document images with convolutional autoencoders. In *Document Analysis and Recognition (ICDAR), The 13th International Conference on*, pages 1011–1015.

Cireşan, D. C., Meier, U., Gambardella, L. M., and Schmidhuber, J. (2010). Deep big simple neural nets excel on handwritten digit recognition. *Neural Computation*, 22(12):3207–3220.

Cireşan, D. C., Meier, U., Gambardella, L. M., and Schmidhuber, J. (2011). Convolutional neural network committees for handwritten character classification. In *Document Analysis and Recognition (ICDAR), the 11th International Conference on*, pages 1135–1139.

Cireşan, D. C., Meier, U., and Schmidhuber, J. (2012). Multi-column deep neural networks for image classification. In *Computer Vision and Pattern Recognition (CVPR), IEEE Conference on*, pages 3642–3649.

Coates, A., Carpenter, B., Case, C., Satheesh, S., Suresh, B., Wang, T., Wu, D., and Ng, A. (2011a). Text detection and character recognition in scene images with unsupervised feature learning. In *Document Analysis and Recognition (ICDAR), The International Conference on*, pages 440–445.

Coates, A., Lee, H., and Ng, A. Y. (2011b). An analysis of single-layer networks in unsupervised feature learning. In *Artificial Intelligence and Statistics (AISTATS), The International Conference on*, pages 215–223.

Cordella, L. P., Stefano, C. D., Fontanella, F., and Marrocco, C. (2008). A feature selection algorithm for handwritten character recognition. In *Pattern Recognition (ICPR), The 19th International Conference on*, pages 1–4.

Csurka, G., Dance, C. R., Fan, L., Willamowski, J., and Bray, C. (2004). Visual categorization with bags of keypoints. In *Computer Vision (ECCV), The 8th European Conference on*, pages 1–22.

Dalal, N. and Triggs, B. (2005). Histograms of oriented gradients for human detection. In *Computer Vision and Pattern Recognition (CVPR), IEEE Computer Society Conference on*, volume 1, pages 886–893.

Das, N., Das, B., Sarkar, R., Basu, S., Kunda, M., and Nasipuri, M. (2010). Handwritten Bangla basic and compound character recognition using MLP and SVM classifier. *Journal of Computing*, 2(2):109–115.

Deepu, V., Madhvanath, S., and Ramakrishnan, A. (2004). Principal component analysis for online handwritten character recognition. In *Pattern Recognition (ICPR), The 17th International Conference on*, volume 2, pages 327–330 Vol.2.

Déniz, O., Bueno, G., Salido, J., and la Torre, F. D. (2011). Face recognition using histograms of oriented gradients. *Pattern Recognition Letters*, 32(12):1598–1603.

der Maaten, L. V. (2009). A new benchmark dataset for handwritten character recognition. Technical Report TiCC TR 2009-002, Tilburg University.

Dhandra, B. V., Mukarambi, G., and Hangarge, M. (2011). Zone based features for handwritten and printed mixed Kannada digits recognition. In *VLSI, Communications and Instrumentation (ICVCI), The International Conference on*, volume 7, pages 5–9.

Diem, M., Kleber, F., and Sablatnig, R. (2011). Text classification and document layout analysis of paper fragments. In *Document Analysis and Recognition (ICDAR), The 11th International Conference on*, pages 854–858.

Eickel, J. (1990). Logical and layout structures of documents. *Computer Physics Communications*, 61(1-2):201–208.

Fan, R.-E., Chang, K.-W., Hsieh, C.-J., Wang, X.-R., and Lin, C.-J. (2008). LIBLINEAR: A library for large linear classification. *Machine Learning Research*, 9:1871–1874.

Ferdinando, H. (2003). Handwriting digit recognition with fuzzy logic. *Teknik Elektro*, 3(2):84–87.

Fischer, A., Frinken, V., Fornés, A., and Bunke, H. (2011). Transcription alignment of Latin manuscripts using hidden Markov models. In *Historical Document Imaging and Processing (HIP), The Workshop on*, pages 29–36.

Gaceb, D., Eglin, V., Lebourgeois, F., and Emptoz, H. (2008). Physical layout segmentation of mail application dedicated to automatic postal sorting system. In *Document Analysis Systems (DAS), The 8th IAPR International Workshop on*, pages 408–414.

Garz, A., Fischer, A., Sablatnig, R., and Bunke, H. (2012). Binarization-free text line segmentation for historical documents based on interest point clustering. In *Document Analysis Systems (DAS), The 10th IAPR International Workshop on*, pages 95–99.

George Washington (1999). George Washington papers at the library of congress. Accessed: 2015-11-02.

Ghosh, R., Bhattacharyya, D., Kim, T.-h., and Lee, G.-S. (2011). New algorithm for skewing detection of handwritten Bangla words. In Kim, T.-h., Adeli, H., Ramos, C., and Kang, B.-H., editors, *Signal Processing, Image Processing and Pattern Recognition*, volume 260 of *Communications in Computer and Information Science*, pages 153–159. Springer Berlin Heidelberg.

Grafm̈uller, M. and Beyerer, J. (2013). Performance improvement of character recognition in industrial applications using prior knowledge for more reliable segmentation. *Expert Systems with Applications*, 40(17):6955–6963.

Gregor, K. and LeCun, Y. (2010). Learning fast approximations of sparse coding. In *Machine Learning (ICML), 27th International Conference on*, pages 399–406.

Gupta, A., Srivastava, M., and Mahanta, C. (2011). Offline handwritten character recognition using neural network. In *Computer Applications and Industrial Electronics (ICCAIE), IEEE International Conference on*, pages 102–107.

Gupta, G., Niranjan, S., Shrivastava, A., and Sinha, R. (2006). Document layout analysis and classification and its application in OCR. In *Enterprise Distributed Object Computing Conference Workshops (EDOCW), The 10th IEEE International*, pages 58–58.

Ha, J., Haralick, R. M., and Phillips, I. T. (1995). Recursive x-y cut using bounding boxes of connected components. In *Document Analysis and Recognition (ICDAR), The 3rd International Conference on*, volume 2, pages 952–955 vol.2.

Hamidi, M. and Borji, A. (2010). Invariance analysis of modified C2 features: case study-handwritten digit recognition. *Machine Vision and Applications*, 21:969–979.

He, S., Wiering, M. A., and Schomaker, L. R. B. (2015). Junction detection in handwritten documents and its application to writer identification. *Pattern Recognition*, 48(12):4036–4048.

Hendrickx, I. and van den Bosch, A. (2005). Hybrid algorithms with instance-based classification. In Gama, J. ã., Camacho, R., Brazdil, P., Jorge, A., and Torgo, L., editors, *Machine Learning: ECML 2005*, volume 3720 of *Lecture Notes in Computer Science*, pages 158–169. Springer Berlin / Heidelberg.

Hinton, G. E., Osindero, S., and Teh, Y.-W. (2006). A fast learning algorithm for deep belief nets. *Neural Computation*, 18(7):1527–1554.

Hinton, G. E. and Zemel, R. S. (1994). Autoencoders, minimum description length and Helmholtz free energy. In Cowan, J., Tesauro, G., and Alspector, J., editors, *Advances in Neural Information Processing Systems 6*, pages 3–10. Morgan-Kaufmann.

Hossain, M. Z., Amin, M. A., and Yan, H. (2011). Rapid feature extraction for Bangla handwritten digit recognition. In *Machine Learning and Cybernetics (ICMLC), The International Conference on*, volume 4, pages 1832–1837.

Hsu, C.-W. and Lin, C.-J. (2002). A comparison of methods for multiclass support vector machines. *IEEE Transactions on Neural Networks*, 13(2):415–425.

Ingold, R. and Armangil, D. (1991). A top-down document analysis method for logical structure recognition. In *Document Analysis and Recognition (ICDAR), The 1st International Conference on*, pages 41–49.

Jarrett, K., Kavukcuoglu, K., Ranzato, M., and LeCun, Y. (2009). What is the best multi-stage architecture for object recognition? In *Computer Vision (ICCV), IEEE 12th International Conference on*, pages 2146–2153.

Karaaba, M. F., Schomaker, L. R. B., and Wiering, M. A. (2014). Machine learning for multi-view eye-pair detection. *Engineering Applications of Artificial Intelligence*, 33(0):69–79.

Kawtrakul, A. and Waewsawangwong, P. (2000). Multi-feature extraction for printed Thai character recognition. In *Natural Language Processing (SNLP), The 4th Symposium on*, pages 1–8.

Koshiba, Y. and Abe, S. (2003). Comparison of L1 and L2 support vector machines. In *Neural Networks (IJCNN), The International Joint Conference on*, volume 3, pages 2054–2059.

Krishnamoorthy, M., Nagy, G., Seth, S., and Viswanathan, M. (1993). Syntactic segmentation and labeling of digitized pages from technical journals. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 15(7):737–747.

Kumar, M., Jindal, M., and Sharma, R. (2011). K-nearest neighbor based offline handwritten Gurmukhi character recognition. In *Image Information Processing (ICIIP), The International Conference on*, pages 1–4.

Lauer, F., Suen, C. Y., and Bloch, G. (2007). A trainable feature extractor for handwritten digit recognition. *Pattern Recognition*, 40(6):1816–1824.

Lawgali, A., Bouridane, A., Angelova, M., and Ghassemlooy, Z. (2011). Handwritten Arabic character recognition: Which feature extraction method? *Advanced Science and Technology*, 34:1–8.

LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. (1998). Gradient based learning applied to document recognition. In *Proceedings of the IEEE*, volume 86(11), pages 2278–2324.

LeCun, Y. and Cortes, C. (1998). The MNIST database of handwritten digits.

Lee, H., Battle, A., Raina, R., and Ng, A. Y. (2007). Efficient sparse coding algorithms. In Schölkopf, B., Platt, J., and Hoffman, T., editors, *Advances in Neural Information Processing Systems 19*, pages 801–808. MIT Press.

Lee, S. E., Min, K., and Suh, T. (2013). Accelerating histograms of oriented gradients descriptor extraction for pedestrian recognition. *Computers & Electrical Engineering*, 39(4):1043–1048.

Lee, S.-W. and Ryu, D.-S. (2001). Parameter-free geometric document layout analysis. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 23(11):1240–1256.

Leedham, G., Yan, C., Takru, K., Tan, J. H. N., and Mian, L. (2003). Comparison of some thresholding algorithms for text/background segmentation in difficult document images. In *Document Analysis and Recognition (ICDAR), The 7th International Conference on*, pages 859–864.

Li, Y., Zheng, Y., Doermann, D., and Jaeger, S. (2008). Script-independent text line segmentation in freestyle handwritten documents. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 30(8):1313–1329.

Likforman-Sulem, L., Zahour, A., and Taconet, B. (2007). Text line segmentation of historical documents: a survey. *International Journal on Document Analysis and Recognition (IJDAR)*, 9(2-4):123–138.

Liu, C. L., Nakashima, K., Sako, H., and Fujisawa, H. (2003). Handwritten digit recognition: benchmarking of state-of-the-art techniques. *Pattern Recognition*, 36(10):2271–2285.

Liu, C. L. and Suen, C. Y. (2009). A new benchmark on the recognition of handwritten Bangla and Farsi numeral characters. *Pattern Recognition*, 42(12):3287–3295. New Frontiers in Handwriting Recognition.

Liwicki, M., Bunke, H., Pittman, J., and Knerr, S. (2011). Combining diverse systems for handwritten text line recognition. *Machine Vision and Applications*, 22(1):39–51.

Louloudis, G., Gatos, B. B., Pratikakis, I., and Halatsis, C. (2009). Text line and word segmentation of handwritten documents. *Pattern Recognition*, 42(12):3169–3183.

Lowe, D. G. (2004). Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110.

Mahmud, J. U. (2005). An intelligent feature analyzer for handwritten character recognition. In *Computational Intelligence for Modelling, Control and Automation (CIMCA), International Conference on Intelligent Agents, Web Technologies and Internet Commerce*, volume 2, pages 763–769.

Mandal, S., Sur, S., Dan, A., and Bhowmick, P. (2011). Handwritten Bangla character recognition in machine-printed forms using gradient information and Haar wavelet. In *Image Information Processing (ICIIP), The International Conference on*, pages 1–6.

Manmatha, R. and Srimal, N. (1999). Scale space technique for word segmentation in handwritten documents. In Nielsen, M., Johansen, P., Olsen, O., and Weickert, J., editors, *Scale-Space Theories in Computer Vision*, volume 1682 of *Lecture Notes in Computer Science*, pages 22–33. Springer Berlin Heidelberg.

Maragoudakis, M., Kavallieratou, E., and Fakotakis, N. (2002). Improving handwritten character segmentation by incorporating bayesian knowledge with support vector machines. In *Acoustics, Speech, and Signal Processing (ICASSP), IEEE International Conference on*, volume 4, pages IV–4174–IV–4174.

Marinai, S. and Nesi, P. (1999). Projection based segmentation of musical sheets. In *Document Analysis and Recognition (ICDAR), The 5th International Conference on*, pages 515–518.

Martínez, C. and Fuentes, O. (2003). Face recognition using unlabeled data. *Iberoamerican Journal of computer science research*, 7(2):123–129.

Meier, U., Cireşan, D. C., Gambardella, L. M., and Schmidhuber, J. (2011). Better digit recognition with a committee of simple neural nets. In *Document Analysis and Recognition (ICDAR), The 11th International Conference on*, pages 1250–254.

Mishra, T. K., Majhi, B., and Panda, S. (2013). A comparative analysis of image transformations for handwritten Odia numeral recognition. In *Advances in Computing, Communications and Informatics (ICACCI), The International Conference on*, pages 790–793.

Mitrpanont, J. L. and Imprasert, Y. (2011). Thai handwritten character recognition using heuristic rules hybrid with neural network. In *Computer Science and Software Engineering (JCSSE), The 8th International Joint Conference on*, pages 160–165.

Moreno-Seco, F., Iñesta, J., de León, P., and Micó, L. (2006). Comparison of classifier fusion methods for classification in pattern recognition tasks. In Yeung, D.-Y., Kwok, J., Fred, A., Roli, F., and de Ridder, D., editors, *Structural, Syntactic, and Statistical Pattern Recognition*, volume 4109 of *Lecture Notes in Computer Science*, pages 705–713. Springer Berlin / Heidelberg.

Naveena, C. and Aradhya, V. N. M. (2012). Handwritten character segmentation for Kannada scripts. In *Information and Communication Technologies (WICT), World Congress on*, pages 144–149.

Nilsson, N. J. (1982). *Principles of Artificial Intelligence*. Symbolic Computation / Artificial Intelligence. Springer.

Nopsuwanchai, R., Biem, A., and Clocksin, W. F. (2006). Maximization of mutual information for offline Thai handwriting recognition. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 28(8):1347–1351.

Otsu, N. (1979). A threshold selection method from gray-level histograms. *Systems, Man and Cybernetics, IEEE Transactions on*, 9(1):62–66.

Pal, U., Roy, R., and Kimura, F. (2010). Bangla and English city name recognition for Indian postal automation. In *Pattern Recognition (ICPR), The 20th International Conference on*, pages 1985–1988.

Pal, U., Sharma, N., Wakabayashi, T., and Kimura, F. (2008). Handwritten character recognition of popular south Indian scripts. In Doermann, D. and Jaeger, S., editors, *Arabic and Chinese Handwriting Recognition*, volume 4768 of *Lecture Notes in Computer Science*, pages 251–264. Springer Berlin / Heidelberg.

Pal, U., Wakabayashi, T., and Kimura, F. (2007). Handwritten Bangla compound character recognition using gradient feature. In *Information Technology (ICIT), The 10th International Conference on*, pages 208–213.

Papavassiliou, V., Stafylakis, T., Katsouros, V., and Carayannis, G. (2010). Handwritten document image segmentation into text lines and words. *Pattern Recognition*, 43(1):369–377.

Phokharatkul, P. and Kimpan, C. (2002). Handwritten Thai character recognition using Fourier descriptors and genetic neural networks. *Computational Intelligence*, 18(3):270–293.

Phokharatkul, P., Sankhuangaw, K., Somkuarnpanit, S., Phaiboon, S., and Kimpan, C. (2007). Off-line hand written Thai character recognition using ant-miner algorithm. *International Journal of Computer, Information, Systems and Control Engineering*, 1(8):2596–2601.

Plamondon, R. and Srihari, S. (2000). Online and off-line handwriting recognition: a comprehensive survey. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 22(1):63–84.

Rajashekararadhya, S. V. and Ranjan, P. V. (2009). Zone based feature extraction algorithm for handwritten numeral recognition of Kannada script. In *Advance Computing Conference (IACC), IEEE International*, pages 525–528.

Rakesh Rathi, Ravi Krishan Pandey, V. C. and Jangid, M. (2012). Offline handwritten Devanagari vowels recognition using KNN classifier. *International Journal of Computer Applications*, 49(23):11–16.

Rossi, A. L. D. and Carvalho, A. C. P. L. F. D. (2008). Bio-inspired optimization techniques for SVM parameter tuning. In *Neural Networks (SBRN), The 10th Brazilian Symposium on*, pages 57–62.

Roy, K., Vajda, S., Pal, U., and Chaudhuri, B. B. (2004). A system towards Indian postal automation. In *Frontiers in Handwriting Recognition (IWFHR), The 9th International Workshop on*, pages 580–585.

Ryu, J., Koo, H. I., and Cho, N. I. (2015). Word segmentation method for handwritten documents based on structured learning. *Signal Processing Letters, IEEE*, 22(8):1161–1165.

Saabni, R. and El-Sana, J. (2011). Language-independent text lines extraction using seam carving. In *Document Analysis and Recognition (ICDAR), The 11th International Conference on*, pages 563–568.

Salimi, H. and Giveki, D. (2013). Farsi/Arabic handwritten digit recognition based on ensemble of SVD classifiers and reliable multi-phase PSO combination rule. *International Journal on Document Analysis and Recognition (IJDAR)*, 16(4):371–386.

Salvi, D., Zhou, J., Waggoner, J., and Wang, S. (2013). Handwritten text segmentation using average longest path algorithm. In *Applications of Computer Vision (WACV), IEEE Workshop on*, pages 505–512.

Sanossian, H. Y. Y. and Evans, D. J. (1998). Efficient feature extraction technique for English characters. *Computer Mathematics*, 66(3-4):257–265.

Sauvola, J. and Pietikäinen, M. (2000). Adaptive document image binarization. *Pattern Recognition*, 33(2):225–236.

Schmidhuber, J. (2015). Deep learning in neural networks: An overview. *Neural Networks*, 61:85–117.

Schomaker, L. R. B. and Bulacu, M. (2004). Automatic writer identification using connected-component contours and edge-based features of uppercase Western script. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 26(6):787–798.

Schomaker, L. R. B., Franke, K., and Bulacu, M. (2007). Using codebooks of fragmented connected-component contours in forensic and historic writer identification. *Pattern Recognition Letters*, 28(6):719–727. Pattern Recognition in Cultural Heritage and Medical Applications.

Schomaker, L. R. B. and Vuurpijl, L. (2000). Forensic writer identification: A benchmark data set and a comparison of two systems. Technical report, University of Nijmegen.

Seo, J. and Park, H. (2014). Robust recognition of face with partial variations using local features and statistical learning. *Neurocomputing*, 129(0):41–48.

Shafait, F., Keysers, D., and Breuel, T. M. (2008). Efficient implementation of local adaptive thresholding techniques using integral images. In *Document Recognition and Retrieval XV (DRR), The 15th Conference on*, pages 681510–681510–6.

Shi, D., Damper, R. I., and Gunn, S. R. (2003). Offline handwritten Chinese character recognition by radical decomposition. *ACM Transactions on Asian Language Information Processing (TALIP)*, 2(1):27–48.

Siddiqi, I. and Vincent, N. (2009). A set of chain code based features for writer recognition. In *Document Analysis and Recognition (ICDAR), The 10th International Conference on*, pages 981–985.

Simistira, F., Papavassiliou, V., Stafylakis, T., and Katsouros, V. (2011). Enhancing handwritten word segmentation by employing local spatial features. In *Document Analysis and Recognition (ICDAR), The 11th International Conference on*, pages 1314–1318.

Singh, T. R., Roy, S., Singh, O. I., Sinam, T., and Singh, K. M. (2011). A new local adaptive thresholding technique in binarization. *Computer Science Issues*, 8:271–277.

Singh, V. and Kumar, B. (2014). Document layout analysis for Indian newspapers using contour based symbiotic approach. In *Computer Communication and Informatics (ICCCI), International Conference on*, pages 1–4.

Solihin, Y. and Leedham, C. (1999). Integral ratio: a new class of global thresholding techniques for handwriting images. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 21(8):761–768.

Som, H. M., Zain, J. M., and Ghazali, A. J. (2011). Application of threshold techniques for readability improvement of Jawi historical manuscript images. *Advanced Computing*, 2(2):60–69.

Song, W., Uchida., S., and Liwicki, M. (2011). Comparative study of part-based handwritten character recognition methods. In *Document Analysis and Recognition (ICDAR), The 11th International Conference on*, pages 814–818.

Stentz, A. (1994). Optimal and efficient path planning for partially-known environments. In *Robotics and Automation (ICRA), IEEE International Conference on*, pages 3310–3317.

Sun, Y., Zhao, L., Huang, S., Yan, L., and Dissanayake, G. (2014). L^2-SIFT: SIFT feature extraction and matching for large images in large-scale aerial photogrammetry. *ISPRS Journal of Photogrammetry and Remote Sensing*, 91(0):1–16.

Surinta, O. and Chamchong, R. (2008). Image segmentation of historical handwriting from palm leaf manuscripts. In Shi, Z., Mercier-Laurent, E., and Leake, D., editors, *Intelligent Information Processing IV*, volume 288 of *IFIP - The International Federation for Information Processing*, pages 182–189. Springer US.

Surinta, O., Holtkamp, M., Karaaba, M. F., van Oosten, J.-P., Schomaker, L. R. B., and Wiering, M. A. (2014). A* path planning for line segmentation of handwritten documents. In *Frontiers in Handwriting Recognition (ICFHR), The 14th International Conference on*, pages 175–180.

Surinta, O., Karaaba, M. F., Mishra, T. K., Schomaker, L. R. B., and Wiering, M. A. (2015a). Recognizing handwritten characters with local descriptors and bags of visual words. In *Engineering Applications of Neural Networks (EANN), 2015 16th International Conference on*, pages 255–264.

Surinta, O., Karaaba, M. F., Schomaker, L. R. B., and Wiering, M. A. (2015b). Recognition of handwritten characters using local gradient feature descriptors. *Engineering Applications of Artificial Intelligence*, 45:405–414.

Surinta, O., Schomaker, L. R. B., and Wiering, M. A. (2012). Handwritten character classification using the hotspot feature extraction technique. In *Pattern Recognition Applications and Methods (ICPRAM), The 1st International Conference on*, pages 261–264. SciTePress.

Surinta, O., Schomaker, L. R. B., and Wiering, M. A. (2013). A comparison of feature extraction and pixel-based methods for recognizing handwritten Bangla digits. In *Document Analysis and Recognition (ICDAR), The 12th International Conference on*, pages 165–169. IEEE Computer Society.

Tang, H., Augustin, E., Suen, C., Baret, O., and Cheriet, M. (2004). Recognition of unconstrained legal amounts handwritten on chinese bank checks.

In *Pattern Recognition (ICPR), The 17th International Conference on*, volume 2, pages 610–613 Vol.2.

Trier, Ø. D., Jain, A. K., and Taxt, T. (1996). Feature extraction methods for character recognition-a survey. *Pattern Recognition*, 29(4):641–662.

Uchida, S., Ishida, R., Yoshida, A., Cai, W., and Feng, Y. (2012). Character image patterns as big data. In *Frontiers in Handwriting Recognition (ICFHR), The 13th International Conference on*, pages 479–484.

van der Zant, T., Schomaker, L. R. B., and Haak, K. (2008). Handwritten-word spotting using biologically inspired features. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 30(11):1945–1957.

van Oosten, J.-P. and Schomaker, L. R. B. (2014). Separability versus prototypicality in handwritten word-image retrieval. *Pattern Recognition*, 47(3):1031–1038. Handwriting Recognition and other {PR} Applications.

Vapnik, V. N. (1998). *Statistical Learning Theory*. Wiley.

Wei, H., Chen, K., Ingold, R., and Liwicki, M. (2014). Hybrid feature selection for historical document layout analysis. In *Frontiers in Handwriting Recognition (ICFHR), The 14th International Conference on*, pages 87–92.

Wen, Y., Lu, Y., and Shi, P. (2007). Handwritten Bangla numeral recognition system and its application to postal automation. *Pattern Recognition*, 40(1):99–107.

Wiering, M. A. and Schomaker, L. R. B. (2014). Multi-layer support vector machines. In Suykens, J. A. K., Signoretto, M., and Argyriou, A., editors, *Regularization, Optimization, Kernels, and Support Vector Machines*, Machine Learning and Pattern Recognition, pages 3–21. CRC Press.

Wolf, C., Jolion, J.-M. M., and Chassaing, F. (2002). Text localization, enhancement and binarization in multimedia documents. In *Pattern Recognition (ICPR), The 16th International Conference on*, volume 2, pages 1037–1040.

Xiu, P., Peng, L., Ding, X., and Wang, H. (2006). Offline handwritten Arabic character segmentation with probabilistic model. In Bunke, H. and Spitz, A., editors, *Document Analysis Systems VII*, volume 3872 of *Lecture Notes in Computer Science*, pages 402–412. Springer Berlin Heidelberg.

Ye, P., Kumar, J., Kang, L., and Doermann, D. (2012). Unsupervised feature learning framework for no-reference image quality assessment. In *Computer Vision and Pattern Recognition (CVPR), IEEE Conference on*, pages 1098–1105.

Zhang, X. and Tan, C. L. (2014). Text line segmentation for handwritten documents using constrained seam carving. In *Frontiers in Handwriting Recognition (ICFHR), 14th International Conference on*, pages 98–103.

SUMMARY

The recognition of handwriting, in multiple scripts and from various historical periods is still a challenging problem at all levels of processing from intensity normalization to segmentation, feature extraction and classification. In this dissertation, several techniques are proposed to address the problems of segmentation and recognition.

Chapter 1 provides a brief general introduction to handwritten character recognition systems followed by the research questions. Additionally, the objectives of the dissertation and contributions are described.

Chapter 2 describes a new efficient A^* path-planning based line segmentation algorithm for historical document images. The goal of line segmentation is to find the best separating curvilinear paths that separate a target line from the surrounding neighbouring text lines. A pervasive problem is the fact that long strokes, ascenders and descenders from surrounding text may touch or cross the current target line. The proposed algorithm applies a cost-based search heuristic using the A^* algorithm to solve this problem. We have evaluated the line segmentation technique on handwritten historical images from the Saint Gall dataset and a specifically selected dataset from the MONK collections, namely the MONK line segmentation (MLS) dataset. These datasets are also available to the scientific community. The results indicate that our newly developed approach was effective in which various kinds of documents could be handled appropriately.

For handwritten character recognition several novel approaches have been proposed in this dissertation, described in chapters 3-5. There are some difficulties in handwritten character recognition which are typical to Asian scripts such as Bangla and Odia. We have studied two datasets containing challenges, such as the differences in writing style, similar shape structures that are found in different character classes, variable-sized and curly begin strokes and end strokes. Furthermore, the small size of available data sets for training and testing pose problems at the level of machine learning.

The objective was to propose a robust feature extraction technique which can solve these problems. Several handwritten character datasets have been used to evaluate feature extraction techniques on Thai, Bangla, Odia, Latin, and the well-known benchmark MNIST dataset.

To obtain the highest recognition performance, the robust feature vectors created by a feature extraction technique are classified by machine learning techniques. We found that even a simple machine learning technique such as the *k*-nearest neighbor (*k*NN) algorithm provides high accuracy results. On the other hand, very high recognition accuracies have been obtained when using a support vector machine (SVM) algorithm as a classifier.

In Chapter 3, we propose two feature extraction techniques, namely the hotspot technique (HOT) and the contour angular technique (CAT), which are computed based on thinned isolated character images. We compare these techniques with the pixel-based method. The results show that the CAT technique combined with an SVM classifier outperforms the other techniques when little training data is used. A higher recognition accuracy is achieved with an unweighted majority vote (UMV) method where the outputs of the different SVM classifiers are combined and the new data are classified subsequently by taking a vote of their predictions.

Chapter 4 describes the effective use of local gradient feature descriptors for handwritten character recognition. In this chapter, two local gradient feature descriptors, which are the histogram of oriented gradients (HOG) descriptor and the scale invariant feature transform descriptor (siftD) are compared. The orientation histograms from the handwritten character grayscale images are extracted and subsequently the robust feature vectors are provided. We have evaluated feature descriptors and classifiers on three complex handwritten datasets in total including Thai, Bangla, and Latin. The siftD and the HOG descriptors have shown the best performances when they are combined with the SVM algorithm as a classifier. Interestingly, even with the simple *k*NN algorithm, the local descriptors still show a good performance.

In Chapter 5, the bag of visual words (BOW) approach using the HOG descriptor, called HOG-BOW, is proposed for recognizing handwritten characters. In this approach, the intensity values that contain local information

of the handwritten character in each local area are processed by using the HOG descriptor. In this technique, firstly, the HOG descriptor captures the gradient structure of the local shape and provides robust features. Consequently, a codebook is constructed by using the K-means clustering method. Finally, the soft-assignment scheme computing the activity of each cluster is used to create the feature vectors. The results show that very high recognition accuracies on Bangla, Odia, and MNIST are achieved when using the HOG-BOW method combined with the L2 regularized SVM.

Chapter 6 comprises two main sections, consisting of the answers to the research questions and the future work. In this chapter, we briefly explain the proposed approaches and answered three main research questions in handwritten character recognition. In the future work, two main approaches are planned to be focused on. The first one is to improve feature extraction using a local feature learning technique together with the multi-layer support vector machines in order to obtain higher accuracies. The second is to focus on a segmentation technique which can be enhanced by applying a process of a lexicon in word recognition level.

SAMENVATTING

Het herkennen van handgeschreven teksten waarin meerdere handschriften voorkomen uit verschillende historische periodes is nog altijd een uitdagend probleem op alle niveau's van verwerking, van intensiteitsnormalisatie tot segmentatie, feature extractie en classificatie. In dit proefschrift worden verschillende technieken voorgesteld om de problemen omtrent segmentatie en herkenning aan te pakken.

Hoofdstuk 1 geeft een beknopte algemene introductie in schriftherkenningssystemen voor handgeschreven tekst, gevolgd door de onderzoeksvragen. Daarnaast worden de doelen en bijdragen van dit proefschrift beschreven.

Hoofdstuk 2 beschrijft een nieuw efficiënt regelscheiding algoritme gebaseerd op A^* path-planning voor historische documentafbeeldingen. Het doel van regelscheiding is om de best mogelijke paden te bepalen om een doelregel te scheiden van naastgelegen tekstregels. Een veelvoorkomend probleem is het feit dat lange pennenstreken, stoklijnen en staartlijnen van omliggende tekst de huidige doelregel kunnen raken of kruisen. Het voorgestelde algoritme past een kostengebaseerde zoekheuristiek toe die het A^* algoritme gebruikt om dit probleem op te lossen. We hebben deze regelscheidingstechniek geëvalueerd op afbeeldingen van historische documenten uit de Saint Gall dataset en een speciaal geselecteerde dataset uit de MONK verzameling, namelijk de MONK regelscheidingsdataset (MLS dataset). Deze datasets zijn ook beschikbaar voor de wetenschappelijke gemeenschap. De resultaten laten zien dat onze nieuw ontwikkelde methode een effectieve techniek is die voor vele verschillende soorten documenten kan worden gebruikt.

Voor handschriftherkenning worden in deze dissertatie een aantal nieuwe aanpakken voorgesteld, die zijn beschreven in hoofdstukken 3-5. Er zijn een aantal moeilijkheden met de herkenning van handgeschreven lettertekens, in het bijzonder voor Aziatische schriften zoals Bangla en Odia. We hebben

twee datasets onderzocht die uitdagingen bevatten zoals verschillende lettertekens met vergelijkbare structuren, variatie in de grootte, aanzet en einde van een pennenstreek, en variatie in de schrijfstijl. Daarnaast zorgde het kleine aantal beschikbare voorbeelden voor het trainen en testen voor problemen op het niveau van machine learning. Het doel was om een robuuste feature-extractietechniek te ontwikkelen die met deze problemen om kan gaan. De feature-extractietechnieken zijn geëvalueerd met behulp van datasets van handgeschreven lettertekens, waaronder de datasets Thai, Bang-la, Odia, Latin en de bekende MNIST benchmarkdataset.

Om de best mogelijke herkenning te verkrijgen, worden robuuste featurevectoren, die zijn verkregen uit feature-extractietechnieken, geclassificeerd met behulp van machine learning technieken. Zelfs simpele machine learning technieken zoals k-nearest neighbors (kNN) blijken een hoge mate van nauwkeurigheid te behalen. Aan de andere kant is de herkenning opvallend goed wanneer een support vector machine (SVM) algoritme gebruikt wordt om te classificeren.

In Hoofdstuk 3 stellen we twee feature-extractietechnieken voor, namelijk de hotspot-techniek (HOT) en de contour-angulartechniek (CAT), die beide berekend worden gebaseerd op uitgedunde geïsoleerde tekenafbeeldingen. We vergelijken deze technieken met de pixel-gebaseerde methode. Uit de resultaten blijkt dat de CAT-techniek gecombineerd met een SVM classifier beter presteert dan de andere technieken wanneer weinig trainingsdata wordt gebruikt. Een hoge herkenningsnauwkeurigheid wordt bereikt met een unweighted majority vote (UMV) methode waarin de output van de verschillende SVM-classifiers gecombineerd wordt en de meerderheid van hun voorspellingen gebruikt wordt om nieuwe data te classificeren.

Hoofdstuk 4 beschrijft effectief gebruik van local gradient feature descriptors voor handschriftherkenning. In dit hoofdstuk worden twee local gradient feature descriptors vergeleken, namelijk de histogram of oriented gradients (HOG) descriptor en de scale invariant feature transform descriptor (siftD). Deze extraheren de orientation histograms uit de afbeeldingen van handgeschreven tekens, en geven robuuste feature vectors terug. We hebben feature descriptors en classifiers geëvalueerd op drie complexe handgeschreven datasets, namelijk Thai, Bangla, en Latin. De siftD en HOG

descriptors geven de beste resultaten wanneer ze gecombineerd worden met het SVM algoritme als classifier. Opmerkelijk is dat de lokale descriptors zelfs met een simpel *k*NN algoritme een goed resultaat geven.

In Hoofdstuk 5 wordt voorgesteld de bag of visual words (BOW) benadering met gebruik van de HOG descriptor, de HOG-BOW, te gebruiken voor het herkennen van handgeschreven tekens. In deze benadering worden de intensiteitswaardes die lokale informatie bevatten over het handgeschreven letterteken in elk lokaal gebied verwerkt met behulp van een HOG descriptor. In deze techniek geeft de HOG descriptor eerst de gradiëntstructuur van de lokale vorm, en voorziet in robuuste features. Vervolgens wordt er een codebook geconstrueerd met behulp van de K-means clustering methode. Tenslotte wordt een soft-assignment scheme, dat de activiteit van elk cluster berekent, gebruikt om feature vectors te maken. Uit de resultaten blijkt dat de HOG-BOW methode gecombineerd met de L2 regularized SVM een zeer hoge herkenningsnauwkeurigheid bereikt op Bangla, Odia en MNIST.

Hoofdstuk 6 bestaat uit twee onderdelen, namelijk de antwoorden op de onderzoeksvragen en toekomstig onderzoek. In dit hoofdstuk bespreken we kort de voorgestelde aanpakken en de antwoorden op de drie voornaamste onderzoeksvragen van handgeschreven tekstherkenning. Het onderdeel toekomstig onderzoek benadrukt twee mogelijke richtingen. De eerste is om de feature extraction te verbeteren met behulp van een techniek om lokaal features aan te leren, gecombineerd met multi-layer support vector machines om een hogere nauwkeurigheid te behalen. De tweede richt zich op de segmentatietechniek, die kan worden verbeterd door een lexiconproces op woordherkenningsniveau toe te passen.

ACKNOWLEDGEMENTS

In September 2009, I received an important e-mail from Dr. Marco Wiering who later became my Ph.D. supervisor. The short message was written in that e-mail, "Prof. Lambert Schomaker agrees that you can come to do your Ph.D. here". For me, it was an absolute delight for having a great opportunity to pursue my study at the Institute of Artificial Intelligence and Cognitive Engineering (ALICE), the University of Groningen, the Netherlands.

Firstly, I would like to express my deep appreciation to Mahasarakham University, Thailand for granting the academic scholarship during my Ph.D. study.

I am truly grateful to my supervisor, Dr. Marco Wiering, who gave me a chance to work on the handwritten recognition project in the autonomous perceptive systems (APS) group. His expertise on machine learning, which he constantly shared with me, effectively contributed to the success of my Ph.D. project. He always knocked on my office's door and came up with a new idea, which made me stay focused and concentrated on my research. This really provided me more inspiration and also productiveness. From the first until the last day of my Ph.D. time, he was very friendly and always had a positive motivation. I have learned a lot of things from him. His advice and supervision literally helped me growing up more in the academic pathway.

Importantly, I would like to express my special gratitude to my promoter, Prof. Dr. Lambert Schomaker. His helpful suggestions were very beneficial to my research project. He shared his major proficiency on handwritten recognition with me every time when we discussed. His invaluable advice really helped enhancing the quality of my research publications. I also would like to express my appreciation to the reading committee, who gave useful suggestions and comments on my research dissertation.

During my stay as a Ph.D. student at the University of Groningen, I wish to thank the ALICE institute for providing me the financial support to participate in several excellent international conferences. Additionally, I also would like to express my gratitude to all my colleagues and friends here at the ALICE institute for offering me such a great working environment. Their kind help and support are sincerely appreciated. I definitely made a right decision to continue my higher education here, although I had to stay thousands of kilometers away from my home in Thailand.

I would like to acknowledge the present and former members of the APS group. We exchanged several research ideas, so many creative points of views came up after our discussions, which are very useful for my project. I also would like to thank Harmen, Trudy, and Charlotte who helped me to translate the summary of my dissertation into Dutch. I also want to thank all of my AI friends, although I cannot write down all your names in this dissertation.

I also want to say special thanks to my friend, Mahir Faik Karaaba, who was supervised by Dr. Wiering and Professor Schomaker as well. His work focused on the field of face recognition. He gave me some useful advice and ideas about my handwritten recognition project. We also worked effectively together on our projects. I do appreciate our good cooperation and friendship.

When I first arrived in Groningen in July 2010, I stayed in Meeuwerderweg, which was a very pleasant area for accommodation. In that house, I had two good housemates whom I will never forget. Andres who came from Ecuador and Liza who was my first Dutch friend. We always had a good time together. Thanks both of you for being my good friends and housemates.

It is important not to forget giving a big thank to the Kajuiter family (Jen, Jacob, Poom, and Jason), Umaporn Pupphachai and also Dollaya and her family. They gave me sincere smiles every time that we met. I love the way they are. Especially, Jen who always cooked very delicious food. Special thanks to my friend, Umaporn Pupphachai, who designed the gorgeous cover of this dissertation for me. I do really appreciate your kindness. I also

want to thank Thai people from all around Groningen for making me feel very welcome every time we had our meetings.

Finally but most importantly, I would like to express my gratitude to my family, especially my mom, Yuchan, and my younger brother Orarak, who always gave me unconditional love and support throughout my study. My thanks also go to my two aunts (Yupin and Apinya), my uncle in law (Suksawat) and my cousin (Apisit). Their visits to the Netherlands in 2012 and 2014 made me feel special and warm inside. I am thankful to everyone in my family for believing in me and for keeping pushing me forward.

Groningen, 23 January 2016
Olarik Surinta

AUTHOR PUBLICATIONS

Journal papers

- **Surinta, O.**, Karaaba, M.F., Schomaker, L.R.B., and Wiering, M.A. (2015). Recognition of Handwritten Characters using Local Gradient Feature Descriptors. Engineering Applications of Artificial Intelligence, 45: 405-414.
- Codreanu, V., Dröge, B., Williams, D., Yasar, B., Yang, P., Liu, B., Dong, F., **Surinta, O.**, Schomaker, L.R.B., Roerdink, J.B.T.M. and Wiering, M. A. (2014). Evaluating automatically parallelized versions of the Support Vector Machine. Concurrency and Computation: Practice and Experience, 1-21.

Peer-reviewed conference papers

- Karaaba, M.F., **Surinta, O.**, M.F., Schomaker, L.R.B., and Wiering, M.A. (2016). Robust Face Identification using Histogram of Oriented Gradients and Bag of Words. In Computer Vision, Imaging and Computer Graphics Theory and Applications (VISAPP), The 11th International Joint Conference on. (*accepted*)
- Karaaba, M.F., **Surinta, O.**, M.F., Schomaker, L.R.B., and Wiering, M.A. (2015). Robust Face Recognition by Computing Distances from Multiple Histograms of Oriented Gradients. In Computational Intelligence in Biometrics and Identity Management (IEEE CIBIM), IEEE Symposium on, pages 203-209.
- **Surinta, O.**, Karaaba, M.F., Mishar, T.K., Schomaker, L.R.B., and Wiering, M.A. (2015). Recognizing Handwritten Characters with Local Descriptors and Bags of Visual Words. In Engineering Applications of

Neural Networks (EANN), The 16th International Conference on, pages 255-264.

- Karaaba, M.F., **Surinta, O.**, Schomaker, L.R.B., and Wiering, M.A. (2015). In-Plane Rotational Alignment of Faces by Eye and Eye-Pair Detection. In Computer Vision, Imaging and Computer Graphics Theory and Applications (VISAPP), The 10th International Joint Conference on, pages 392-399.
- **Surinta, O.**, Holkamp, M., Karaaba, M.F., van Oosten, J.P., Schomaker, L.R.B., and Wiering, M.A. (2014). A* Path Planning for Line Segmentation of Handwritten Documents. In Frontiers in Handwriting Recognition (ICFHR), The 14th International Conference on, pages 175-180.
- **Surinta, O.**, Schomaker, L.R.B., and Wiering, M.A. (2013). A comparison of feature extraction and pixel-based methods for recognizing handwritten Bangla digits. In Document Analysis and Recognition (ICDAR), The 12th International Conference on, pages 165-169.
- **Surinta, O.**, Schomaker, L.R.B., and Wiering, M.A. (2012). Handwritten character classification using the hotspot feature extraction technique. In Pattern Recognition Applications and Methods (ICPRAM), The 1st International Conference on, pages 261-264.

INDEX