Department of Computer Science - University of Fribourg, Switzerland

RECOGNITION OF ULTRA LOW RESOLUTION, ANTI-ALIASED TEXT WITH SMALL FONT SIZES

THESIS

Submitted to the the Faculty of Science, University of Fribourg (Switzerland)

To obtain the degree of Doctor Scientarium Informaticarum

Farshideh Einsele-Aazami

from

Switzerland and Iran

Thesis N° 1610
Imprimerie St Paul, Fribourg
2008

Accepted by the Faculty of Science of the University of Fribourg
following the proposal of:

- Prof. Ulrich Ultes-Nitsche, University of Fribourg (Jury President)
- Prof. Rolf Ingold, University of Fribourg (Thesis Director)
- Prof. Horst Bunke, University of Bern, Switzerland (External Expert)
- Prof. Roger Hersch, Federal Institute of Technology Lausanne (EPFL), Switzerland (External Expert)
- Dr Jean Hennebert, University of Fribourg (Expert)

Fribourg, July 3rd, 2008

Faculty Dean	Thesis Director
Prof. Titus Jenny	Prof. Rolf Ingold

Acknowledgements

This thesis was conducted in Computer Science Department of the University of Fribourg and was founded by Swiss National Science Foundation (SNF), University of Fribourg and Hasler Stiftung. This thesis is the achievement of 5 years of research in the area of document analysis and image processing.

Firts of all, I would like to express my sincere appreciation to Prof. Rolf Ingold, for accepting me in his research group DIVA, for his belief on me, for his valuable guidance and insight, encouragment and reliance throughout this research.

I am also greatly indebted to Dr Jean Hennebert for supervising me especially with his valuable knowledge on the hidden Markov models and besides for his patient and guidance in various domains of research and scientific writing.

I also would like to thank the Jury President Prof. Ulrich Nitsche for accepting to preside my thesis defense and also the official referees of this thesis: Prof. Hurst Bunke and Prof. Roger Hersch for the honor they gave me and for their excellent expertise and precious advices to improve this thesis and for the wonderful discussions we had during the thesis defense process.

I am also thankful for the support and friendly atmosphere in the entire DIVA group that made me work with pleasure for my research. Especially I want to thank my colleague Andreas Humm firstly for his useful programming and LaTex tips and secondly for being a good friend and not let me feel down in the crucial periods of this thesis. As well, I express my gratitude to the secretary team of Department of Informatics (DIUF) and especially to Mrs Eliane Schaller for her optimistic and kind way.

Lastly but most importantly, I thank my wonderful husband Alex for his love and patience and moral support he gave me during every phase of this thesis. I am also very lucky to have such lovely kids as Cyrus and Semira that always heard about the progress of Mami's work with great interest.

Abstract

In this thesis, we address the problem of recognizing text at ultra low resolution (between 72-100 dpi) that is anti-aliased and has font sizes between 7-12 points. Such text is frequently found in web images and is an important carrier of information indexing and retrieval on the web. *Anti-aliasing* is applied to smooth out edges and diagonals, thus making text more legible on computer screens. However, the anti-aliased characters are mostly connected and can not be segmented with common segmentation methods used in classical document analysis. Moreover, anti-aliasing produces contextual noise at the left and right sides of adjacent characters causing drastic deterioration in recognizing such characters, even if they have been segmented.

The classical OCR systems are not capable to recognize such text, as they are basically designed for bilevel text with a resolution of more than 150 dpi, typically obtained using flat scanners. Several research works report on algorithms for detection and recognition of text embedded in web images. However, these works are often directed towards the implementation of front-end image processing methods in order to enhance the image quality and pass it through classical OCRs.

The main contribution of the work presented here is about a novel approach to segment and recognize anti-aliased text at ultra low resolution (ULR) with small font sizes. The *dedicated recognizer*, presented in this thesis, can be directly applied on the detected and extracted anti-aliased text with no needs for using of cost-intensive pre-processing methods to enhance image quality to be recognized with classical OCRs.

Several studies on single characters and words are presented in this report. We used synthetical character and words for all our studies. The first study aimed at choosing an adequate feature extraction methodology for anti-aliased low-resolution characters. The features that were selected are based on the computation of central moments which are translation invariant and convey sufficient spatial information from character shapes at ultra low resolution. The second study aimed at understanding the mutual influence of the connected characters in a word. In both studies, the recognition of characters is performed by computing an estimation of the probability density functions of central moment features using Gaussian based estimators. We could reach good Character Recognition Rates (CRR) of 99.93% for isolated characters. These results have shown that central moments are adequately discriminative. The CRRs for contextual characters were less accurate (98.45%) due to the contextual noise between adjacent

characters. On the other hand, these experiments have shown that we need an intrinsic model to simultaneously segment and recognize connected characters in ULR anti-aliased words with small font sizes.

Therefore, hidden Markov models (HMMs) using a sliding window feature extraction have been chosen to build dedicated word recognizers. In addition, the insights of the previous studies have been beneficial to the design of the word recognizers. Furthermore, each word is modeled using the character models which are constrained by minimum and mixmum width constraints. The contextual noise is modeled as an inter-character state inserted between character models. Further optimizations of the HMM implementation have also been implemented to further increase the Word Recognition Rate (WRR), such as using mixture of Gaussian models instead of mono Gaussian models to estimate the emission probabilities.

We have implemented two dedicated recognizers. The first one is a dictionary-based recognizer which can recognize up to 60'000 words. The second one is an open vocabulary recognizer that can recognize any arbitrary word in any language written in Latin alphabet. We performed experiments in a mono-font context and achieved a WRR of up to 97.56% for the dictionary based recognizer. We achieved an overall WRR of 97.43% for the open vocabulary recognizer. Furthermore, we assessed the performance of the open vocabulary recognizer in a multi-font context more specifically as a sans serif and a serif recognizer. We achieved an overall WRR of 96.94% for both cases.

Zusammenfassung

In dieser Dissertation wird das Problem der Texterkennung in Anti-Aliased Bildern mit sehr tiefer Auflösung von 72 bis 100 dpi und bei Schriftgrössen zwischen 7 bis 12 Punkten behandelt. Solche Texte kommen häufig in Bildern im Web vor. Da die darin enthaltenen textlichen Informationen von grosser semantischer Bedeutung sein können, verbessert eine ausreichend gute Erkennung die Qualität der Indexierung und erhöht auch die Gewinnung von Informationen bei automatischen inhaltsbasierten Suchsystemen.

Anti-Aliasing ist eine Technik, die zur Verbesserung der Lesbarkeit bei einer limitierten Anzahl von Pixeln verwendet wird. Dabei werden die Pixel in Graustufen dargestellt und ermöglichen damit eine bessere Erkennbarkeit, speziell bei Kanten und Diagonalen. Anti-Aliasing führt aber auch dazu, dass ein Teil der einzelnen Buchstaben auf jeweils beiden Seiten miteinander verschmilzt. Diese Verschmelzung führt dazu, dass die einzelnen Buchstaben nicht mehr mit den bekannten Segmentierungsmethoden der klassischen Dokumentanalyse aufschlüsselbar sind. Zusätzlich erzeugt diese Technik ein so genanntes "kontextliches Rauschen", sowohl auf der linken wie auch auf der rechten Seite der einzelnen Buchstaben. Dieses Rauschen führt zu einer deutlichen Verschlechterung der Schrifterkennungsrate selbst dann, wenn die Buchstaben eines jeden Wortes vorgängig segmentiert worden sind. Die klassischen OCR-Systeme sind nicht in der Lage, solche Texte ausreichend zu erkennen, da sie grundsätzlich zum Erkennen von Scans aus gedrucktem Text mit einer minimalen Auflösung von 150 dpi entworfen worden sind. In der Literatur findet man so auch zahlreiche Artikel, die von der Extraktion und Erkennung von Texten in Web-Bildern handeln. Diese Arbeiten konzentrieren sich aber fast ausschliesslich auf die Textextraktion und die Erhöhung der Auflösung des extrahierten Textbildes mittels Anwendung diverser Preprocessing Techniken. Die somit erhaltenen Textbilder werden dann anschliessend mittels eines kommerziellen OCR Softwares erkannt.

Die hier vorgestellte Arbeit verwendet einen neuartigen Ansatz, sowohl für die Segmentierung wie auch für die Erkennung von Anti-Aliased Texten in Bildern mit sehr niedriger Auflösung auch bei kleinen Schriften. Die hier eingesetzten Methoden werden direkt auf den extrahierten Anti-Aliased Text angewendet und verwenden somit auch keine Preprocessing Methoden zur Verbesserung der Bildqualität und Erhöhung deren Auflösung.

Es werden zuerst Experimente mit einzelnen Buchstaben vorgestellt. Eine erste Studie diente zur Ermittlung einer geeigneten Methode zur Extraktion der Merkmale von isolierten

Buchstaben. Die so ermittelten Eigenschaften waren die zentralen Momente erster und zweiter Ordnung. In einer zweiten Studie wurden die gegenseitigen kontextlichen Einflüsse von benachbarten Buchstaben untersucht. Die Verteilung der Merkmale in den beiden ersten Studien erfolgte dabei unter der Annahme einer einfachen Gauss'schen Verteilung (Mono-Gauss). Damit konnte bei isolierten Buchstaben eine bemerkenswert gute Erkennungsrate von 99.93% erreicht werden. Bei Buchstaben, die aus dem Kontext eines Wortes herausgelöst worden sind, betrug die Erkennungsquote immer noch 98.45%. Diese Abnahme ist in erster Linie auf den Einfluss des "kontextlichen Rauschens" zurckzuführen.

Beide Studien haben die Notwendigkeit nach dem Einsatz einer statistischen Methode aufgezeigt, die in der Lage sein sollte, verbundene Buchstaben in einem Wort gleichzeitig zu segmentieren und zu erkennen. Eine Methode, die sich dazu als sehr geeignet erweisen sollte, ist die Hidden Markov Modelle (HMM). Um das Problem der Segmentierung zu umgehen, wurde das unbekannte Wortbild in mehrere kleine Fenster aufgeteilt (mittels der sogenannten Sliding Windows Technik) und es wurden dann die Eigenschaften jedes einzelnen Fensters berechnet.

Im Weiteren konnten auch die Erkenntnisse der ersten Studien bei der Entwicklung eines Erkennungssystems für ganze Wörter einfliessen. So konnte zum Beispiel ein zusätzlicher Buchstabe zur Darstellung des "kontextlichen Rauschens" eingeführt werden oder es wurden Multi-Gauss'sche Funktionen anstelle der einfachen Gauss'schen Verteilung zur Darstellung der Distribution der Merkmale eingesetzt. Zur weiteren Verbesserung der Wort-Erkennungsrate wurden die Zustände mit einer minimalen und maximalen Vorkommenslänge versehen.

Es wurden zwei verschiedene Erkennungssysteme für isolierte Wörter implementiert. Das Erste basiert auf einem Wörterbuch und ist in der Lage, bis zu 60'000 Wörter zu erkennen. Dem Zweiten liegt kein Wörterbuch zugrunde und kann daher auf jedes beliebige Wort in jeder Sprache angewendet werden, solange nur der Text in lateinischen Buchstaben geschrieben ist. Es wurden auch Experimente zur Messung der Performances beider Erkennungssysteme durchgeführt. Diese Experimente wurden vorerst in einem "Mono-Font" Kontext ausgeführt.

Danach wurden die zu diesen typographischen Schrifteigenschaften passenden und optimierten Charakter-Modelle dem Erkennungssystem zugeführt. Damit und mit dem Einsatz des Viterbi-Algorithmus wurde das richtige Wort erkannt. Mit einer solchen Konfiguration wurde eine generelle Wort-Erkennungsrate bei 48 Fonts von 97.56% für das Wörterbuch-basierte Erkennungssystem und von 97.43% für das Wörterbuch-unabhängige Erkennungssystem erreicht. Zusätzlich wurden die Möglichkeit der Gruppierung der Serif und Sans Serif Fonts zur Bildung eines Serif oder Sans Serif Erkennungssystems für das Wörterbuch-unabhängige System überprüft. Es konnte eine generelle Wort-Erkennungsrate von 96.94% in beiden Fällen erreicht werden.

Contents

Chapter 1

Introduction

Optical Character Recognition (OCR) is the process of converting text embedded in a digital image into text or word processing files that can be easily edited and stored. OCR technology has made significant impact on the way information is stored, shared and edited. Prior to OCR, to turn a book into a word processing file, each page would have to be typed word for word. OCR has made considerable advances in recent years thanks to the progress in the recognition methodologies and also to the availability of better and inexpensive scanners. As a drawback, OCR technology has been tuned to perform well in the context of high quality scanned documents and still does not perform well in recognizing text in images with a resolution less than 150 dpi. In addition, OCRs often fail to recognize anti-aliased text with small font sizes, as they are generally designed to recognize bilevel text in high resolution images. In this chapter, we first provide an overview of the OCR domain and then focus on our thesis work from the perspective of data acquisition, application and used technologies.

1.1 OCR overview

When looking to the OCR domain from a broad perspective, we can distinguish three different categories as follows:

1.1.1 Data acquisition

Data acquisition is the way the input image is produced. We can think of various ways to produce the input image digitally:

- *Scanner*:
  The text is lying on a sheet of paper and its image is fed to a dedicated scanning device which typically is of high resolution.

- *Digital steady or video camera*:
  Text is in a scene such as store signs, road signs, billboards, etc.

- *Superimposed text in digital video camera*:
  Text is artificially produced and added to the scene like subtitles, headlines of news programs or sport news.

- *Rendered text using some image processing tools like Adobe Photoshop or Macromedia Fireworks*:
  Text is synthetic with special effects applied to it like hinting, anti-aliasing etc.

- *Screen-captured text using screen snapshot software on top of other word processing tools*:
  Text is altered by rendering options of word processing software.

1.1.2 Applications

Detection and recognition of text in digital images can be useful for different application domains such as below:

- *Document processing*:
  The objective here is to convert scanned documents, text embedded in web images, video images and digital cameras into a word processing document.

- *Web indexing*:
  The text contained in a web image is of high semantical value. Therefore its detection and recognition makes indexation of web pages more efficient.

- *Content based image retrieval and indexing of video images*:
  Indexation and retrieval of the archived videos that contain movies, news and sport programs is an important application. In addition to this, the explosive growth of recent video applications like YouTube demands for even more efficient systems to index and retrieve their textual context.

- *Automated reading*:
  This can be used to help the visually impaired or blind to read the embedded text in digital images. A great deal of applications for automated narrating of documents for people such as car drivers also exists.

- *Screenshot OCR*:
  Recent word processing softwares pose an option to construct screenshots from portions of a document. This option is frequently used as a fast and uncomplicated way to share interesting parts of a document amongst people. Currently some of the available commercial OCRs are capable of detecting and recognizing such text if the snapshot is sufficiently zoomed-in.

- *Object/Product recognition*:
  Getting more information about an interesting object or product. For example, we may

be interested to find out more details or the best catalogue price for a bottle of wine from its picture that was captured with a digital camera.

- *License plate recognition*:
  Identification of cars' license plates for various police forces and as a method for electronic toll collection on pay-per-use roads. Monitoring traffic activity like adherence to a red light at an intersection, etc..

1.1.3 Technologies

Text that is embedded in digital images using multiple text acquisition technologies as previously described, needs to be *detected* and *recognized* for each of the different applications that we have listed above. A complete OCR system has to be able to work on scanned documents and to detect and recognize text in digital images. However, most of the commercial OCR systems are designed to detect and recognize text from bilevel images having a resolution of at least 150 dpi. Most of the work found in literature about *text detection and recognition* of digital images is focused on developing methods to extract and detect the embedded text in such images. That is, methods to localize text and to separate it from its background, binarize it to have a bilevel text image, and preparing it so that it can be fed to classical Optical Character Recognition (OCR) engines. Few works of developing a dedicated recognizer that can be applied directly following extraction and detection of text have been reported. In chapter 2, we provide a survey about methods and algorithms that are reported in the literature for text detection in digital images. Text recognition can be applied either on isolated characters, words, text lines or an entire page.

1.2 Context

The World Wide Web is the world's largest publicly available information provider nowadays. Text embedded in web pages contains valuable information that can be used for information retrieval and indexing. The success of the World Wide Web has made information accessible at low costs around the globe. The images as a powerful communication medium [1] have been rapidly integrated into web pages since digital devices are available at reasonable costs which allows the images to be produced and published in large scales. Consequently, web images are now a substantial part of web pages and often contain textual information with high semantical value that can be used for information retrieval and indexing [2].

Search engine crawlers constantly scan web pages for information indexing. However, such crawlers are usually programmed to only seek out the HTML plain text. The Internet Engineering Task Force (IETF) has made a recommendation to utilize the $< Alt >$ tag to describe the content of an image and overcome this drawback. However, web designers often disregard this recommendation and thus web engine crawlers are not able to retrieve and index such crucial

textual information.

Various works report on methods to index the web images. These approaches are either content-based or text-based. The content-based approaches use the image shape, color and texture for search and indexing. They usually compare description of features of a target image with the images contained in their database [3], [4] and [5]. Content-based indexing is firstly computationally cost intensive when used in a large database and secondly it needs a draft of the searched image to query the database which is not simple nor always available. Text-based indexing analyzes the HTML text associated with these images [6]. Textual description of the image content is stored in the database. Such description can be for ex. textual information contained in the $< Alt >$ tag or in alternative optional fields, image titles or surrounding text in a HTML page. When the user enters a keyword description of his searched information, this keyword is then compared to the description of the stored images in the database. Text-based indexing is obviously computationally less cost intensive because it needs a textual keyword description to search for contextual information.

However, both described approaches are still insufficient to provide good retrieval quality since the content-based approach does not use textual information embedded in images or in the surrounding HTML text and the text-based approach uses textual annotations in HTML text that is added manually. Textual human annotations are unfortunately mostly poorly provided in HTML pages.

1.3 Contribution

In this thesis, we propose a novel approach aiming at developing a *dedicated recognizer* that can be directly applied to anti-aliased text embedded in web images. Such a *dedicated recognizer* can be employed instead of cost intensive pre-processing algorithms to enhance the image quality. The main advantage of the *proposed dedicated recognizer* is that it is developed taking into account the specificities of anti-aliased, ultra low resolution text in small font sizes. Our motivation for developing such recognizer has been to gain higher recognition accuracy which can significantly contribute to the domain of information retrieval and indexing of web pages. Another practical application we can think of is to be able to *cut* and *paste* text from web images into different word processing applications. Ultimately, the developed recognizer could be used for other purposes such as automatic reading to aid visually impaired in reading text embedded in images contained in a web page.

We have conducted two preliminary studies aiming at understanding the specificities of such text and at extracting features that are strongly discriminative. To show the proof of our concept we have applied the chosen features on isolated characters with ultra low resolution (ULR) that are anti-aliased at small point sizes between 5-12 points. We could confirm that the selected features were fairly discriminative. The next challenge we faced was the character segmentation. As shown in Fig. 1.1 the characters in such words are mostly connected, thus

the well-known segmentation methods in classical document analysis like connected component analysis or vertical and horizontal projection profiles cannot be applied to isolate characters in the words.

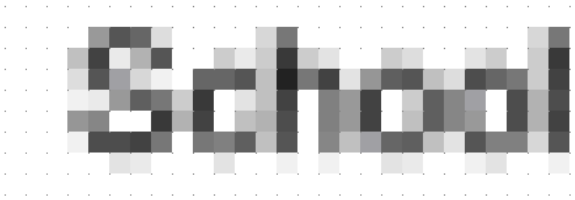


Figure 1.1: A magnified image of word "School" at ULR, anti-aliased with 9 point size

Therefore, we have chosen hidden Markov models (HMMs) that are able to simultaneously segment and recognize the characters. In the literature we can find studies that deal with similar segmentation challenges as ours. Such problem domains, for example, are cursive handwriting recognition that has to simultaneously segment and recognize connected characters or automatic speech recognition which faces the segmentation and recognition of connected speech phonemes. Most approaches in these two research fields make use of HMMs to segment and recognize handwritten text or voice signals. HMMs are sophisticated and flexible statistical models that are used in a wide variety of signal and data analysis applications such as biological sequence analysis, study of protein molecules and financial problems allowing to sort through the random "noise" of the financial markets.

1.4 Thesis Outline

The thesis is structured as follows:

Chapter 2: State of the art

Introduces the related work for text detection and recognition in digital images.

Chapter 3: Low resolution text rendering

Provides an overview of the techniques used in digital typography to represent text on low resolution screens and ends up with the specificities and challenges faced with the ultra low resolution, anti-aliased texts at small font sizes.

Chapter 4: Algorithmic fundamentals

Provides an overview of the state of the art techniques used in pattern recognition and specifically in the document analysis. We delve into the details of the selected features and classification methods we have used for training and testing.

Chapter 5: Study of ULR single characters

This chapter reports the following:

- Investigate if the selected features have a normal distribution
- Investigate the discriminative power of the selected features
- Investigate the classification performance of Bayes decision theorem

Chapter 6: Recognition of ULR single words using HMMs

This chapter reports on various recognition systems that we have implemented for recognizing ULR single words. The recognition systems are based on hidden Markov models as an intrinsic model to simultaneously segment and recognize connected characters in an ULR word. Basically, we have implemented two different recognition systems:

- A dictionary-based recognizer
- An open vocabulary recognizer

In addition, in this chapter we report on the improvements we have made to our recognizers. These improvements are:

- Inter-character model that is inserted as an extra character between the characters
- Different HMM-topologies
- Automatic training using HMMs
- Optimization of Viterbi algorithm for a dictionary-based word recognizer
- Determining an optimal training set for an open vocabulary word recognizer

Chapter 2

State of the art

There is no doubt about it: The rapid growth of images containing text in multimedia environments demands for efficient systems for information indexing and retrieval. As the text contained in such images is of high semantical value, many studies have been conducted on detection and recognition of such embedded text. Jain et al. [7] state that text detection is interchangeably used with text localization and text extraction. However, for the sake of clarity, we can divide both text detection and recognition into different areas as shown in Fig. 2.1.

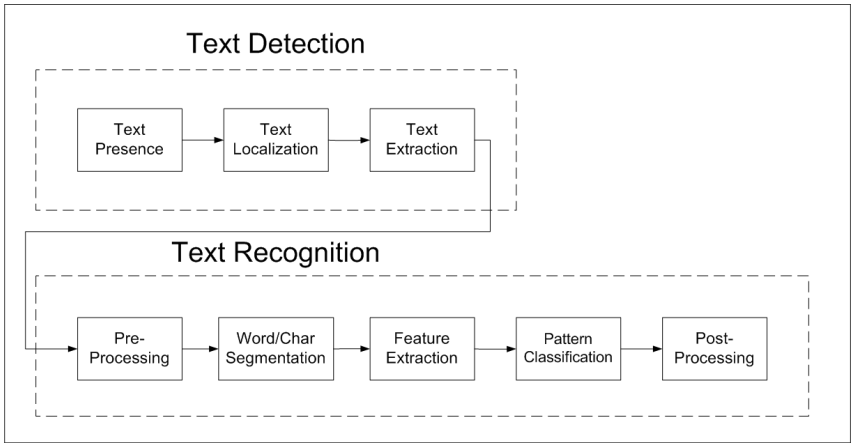


Figure 2.1: Text detection and recognition

Therefore text detection includes following stages:

- *Text presence determination*: Determining the precise presence of the text in a given image
- *Text localization*: Locating the text in the image and generating bounding boxes around it.
- *Text extraction*: Segmentation or separation of the text from the background

Text Recognition can be divided into five stages:

- *Pre-processing*: Binarizing, normalizing and enhancing the image
- *Word / Character segmentation*: Segmentation of words in a text line and characters in a word
- *Feature extraction*: Transforming the data set into discriminative features in order to extract the relevant information from the input data
- *Pattern classification*: Using classification algorithms to assign an unknown pattern to a certain class
- *Post-processing(Language-based Modeling)*: Improving the recognition rate by assigning the knowledge-based probability of character or word sequences to the recognition system, making some assumptions such as spoken language, vocabulary used, etc.

From a broad perspective, we can divide digital images into four major applications:

1. Web images
2. Video images
3. Images captured by digital cameras
4. Screen-shot images

Each of these applications has its own specificities that differentiates them. Consequently we can find in the literature various studies of *text detection and recognition* in each specific application. In the next section, we provide a survey of the related works for *text detection and recognition* from these four types of digital images. Liang, Li and Doermann in [8] [9] provide another survey that lists the works of two first types, i.e. web and video images.

A further analysis of the existing literature shows that the quantity of such studies dealing with *text detection* of digital images predominate the works devoted to *text recognition* of such images. To be precise, the general procedure of the majority of the published works is to concentrate on pre-processing methods to improve the quality of the detected images and enhance their resolution to be suitable (> 150 dpi) to be recognized with classical OCRs. Consequently, we can find few works that developed *dedicated recognizers* designed for text embedded in digital images.

Acknowledging this, we decided to focus our research on the development of a targeted recognizer that is specifically trained for the recognition of such text. We were specially interested in developing a system for recognition of text in web images *that is anti-aliased with ultra low resolution (between 70-96 dpi) and has small point sizes (between 6-12 points). While we focus on text on anti-aliased text in web images, we believe that the principles of our approach could also be generalized to the recognition of other kind of texts, like shadowed text etc., embedded in web images and even further to text embedded in images from video and those shot using a digital camera.*

2.1 Text detection and recognition in web images

Typical images from the web are banners, headers, illustrations, buttons etc.. Web images are in many ways different from real scene images and other document images. First, the text embedded in web images is in visual distinction from its background to attract quickly the attention of web site visitor. Second they are of ultra low resolution (72-100 dpi) and have various artifacts due to the color quantization and lossy compression that speeds up the download. Third such text is often of small font size (between 6-12 points). Finally because few pixels are available to represent the shape, anti-aliasing filters are used to help smoothing the transitions between edges and diagonals of the characters. On the other hand, such text shows a few advantages as it is unskewed and does not contain noise often seen in scanned documents like blurring and speckles and additional noisy pixels. Fig. 2.2 shows two web images containing textual information. The image at the top is the original image and the image below is its enlarged version.

Figure 2.2: Examples of web images; top: image in original size, bottom: magnified image

However, classical OCRs often fail to recognize such text as they are designed to work with mostly bilevel images having a resolution up to 150 dpi and font sizes bigger than 10 pts. We have verified this fact on a commercial OCR and report of the obtained results later in this chapter.
Nevertheless, several studies are conducted on the problem of detecting and recognizing text embedded in web images. In the following we give a survey of the works in the literature in this research field. We are not aware of works that might have been performed under industrial confidentiality codes and therefore not accessible in the scientific literature. Additionally, table 2.1 shows an overview of the described works.

Kanugo et al. in [18] investigate the fraction of images on the Web which contain text and also the percentage of image text that does not appear in the corresponding HTML file. They

Table 2.1: Overview of the works for extraction and recognition of text in web images

Author	Year	OCR type	Test Database	Detection Rate	Recognition Rate	Detection Method
Antonacopoulos [10]	2000	-	42	70.05%	-	Color space analysis
Antonacopoulos [11]	2002	-	?	80-95%	-	Combined connected component & Fuzzy segmentation
Antonacopoulos [12]	2004	-	115	51-75%	-	Split-Merge
Lopresti [13]	1996	dedicated	50	-	8 character classes: 69.7%	Surface Fitting
Lopresti [14]	1997	dedicated	50	-	8 character classes: 89.3%	N-tuple
Lopresti [15]	1997	-	262	47%	-	Color clustering & Connected components
Perantonis [16]	2003	FineReader	650	85.58%	61.58%	Text Area identification
Perantonis [17]	2004	FineReader	1'100	80.45%	64.07%	Text area location

statistically sample the images contained in web pages that were retrieved by a search engine for specific queries. Then they try to find the fraction of sampled images that contain text. This fraction is used to estimate the probability of finding text in a web image. Additionally they count the number of image words that do not appear in the corresponding HTML files. They randomly chose 265 images containing text from 862 web pages that contained total 18'161 images. They have built a ground truth by manually entering the image text file into a corresponding text file. They show in their evaluation tests that 42% of the images in the sample contain text. In addition to this, they show that 59% of the images containing text have at least one word that does not appear in HTML text. Thy have also performed preliminary experiments with other queries and have obtained similar results. Their finding shows two important facts: first the indexation of text in images is crucial, as words in the images are not always included in the surrounding HTML text and second they raise the question of if the image content of web pages is increasing with time. The authors state that their quantitative estimates concur with the subjective hypothesis that the amount of text in web images is increasing.

In [2] Antonacoupolous et al. conduct a study to assess the impact and consequences of text contained in images. They report on the fact that of the total number of words visible in a web page, 17% are in image form. Of the words in image form, 76% do not appear elsewhere in the encoded text. In addition to this, the recommended *Alt* tag for textual description in images are wrong, incomplete or does not exist in 56% of the cases.

Furthermore in [10] the authors propose their first approach to text extraction from WWW images. They approximate in their method the human perception characteristics for the identification of character regions. Their method is based on a combination of Hue-Luminance-Saturation(HLS) and a wavelength-luminance representations to analyze the differences in chromacity and luminance. Then, they group together regions with more than 25% similarity and finally use a connected components analysis by taking into account the size, aspect ratio and

density of individual components to examine whether a component is a character or not. They extract 124 images from web with varying degrees of their for- and background complexity for their training set and 42 images with the same complexity variation for their test set. They visually inspect if the characters are correctly identified or partly identified or are missing. They find out that their method works well for monochrome text and background as 89% of characters have been correctly identified. Moreover 72% of characters contained in a monochrome text over multicolored background were correctly identified. The character recognition rates decreases to 66% for multicolored text over monochrome background and to 56% for multicolored text over multicolored background. However, in their approach the recognition of the extracted text is still an open issue. Later in [11] they describe a new method for the extraction of text from web images. Pixels of similar colors are merged into components and a fuzzy inference mechanism is devised to group components into larger character-like regions. They report of a character extraction rate that varies between 80-95% regarding the image background and foreground complexity. Finally in [12] the authors propose a complete method to extract characters from non-uniform color and in more complex situations. They use a split-and-merge segmentation algorithm by using the Hue-Light-Saturation representation of color. They achieve character recognition rates (CRR) up to 74.24%. Besides their text extraction method achieves 53.4% precision and 88% recall.

Lopresti and Zhou in [13] explore some possible web applications for existing document analysis techniques and report that classical image processing and pattern recognition techniques can not be applied to different web applications without modification. They further report in their work that most web pages contain at least a fraction of their text in image format and besides for most pages in average about 18% of the image text doesn't appear elsewhere. Additionally they report about a method for text detection by quantizing the color space into a small number of color classes by clustering colors into groups. After quantization of color space, they separate the correct foreground color classes that represent the text. However, this method can not be applied on text with varying colors and images with gradually changing colors. At recognition step, they present a method that models a character shape by a polynomial surface function. Further they report of preliminary experimental results showing that their recognition method is effective to distinguish the majority of character classes. However, their method can not distinguish between character classes with similar shapes like 'c' and 'e'. Besides the polynomial surface representation method is computationally cost intensive.
In [15] the same authors report on an approach based on color clustering for text extraction followed by a connected component analysis for text detection. Their approach assumes that the foreground color is roughly uniform for a given character. Pixels are assigned to the character classes closest to their original colors. Then, they find out connected components belonging to each color class. Their algorithm classifies connected components into text-like and non-text-like classes. Finally they apply additional tests to eliminate spurious components as short words or non-text lines. The evaluation test is done on 262 images chosen from web. To

assess performance, they manually transcribe the text from web image and compare it with the extracted characters. They have found out that in 1/5 of the samples the detection algorithm has extracted more than 90% of the text from input. In this paper, they report an average charceter detection rate of 47%.
Additionally, in [14] they describe a further method for recognizing text embedded in web images. This method is based on n-tuple technique that works better than polynomial surface fitting method. To measure the effectiveness of their second method, they use a practical searching algorithm to find n-tuples that are 'shift' invariant which makes them independent of sample variations. They achieve higher CRR by increasing the number of n-tuples. When setting the number of n-tuples to 20, they obtain an average CRR of 90.2%.

Pretagonis et al. in [16] present a novel web image processing algorithm for text area identification that helps commercial OCR engines to improve their web image recognition accuracy. Their algorithm converts first the web image color into gray scale to consider the transitions of brightness that are perceived by human eyes. Second, they use an edge extraction technique that helps to extract all objects from the web image. Finally they use a conditional dilation technique to iteratively choose the text objects among all objects. Then, they pass the extracted text through a commercial OCR. They test their algorithm on a test corpus of 680 images and obtain promising detection rates up to 83.5% and CRR up to 61.5%. In [17] the authors have improved their algorithm. They define a threshold to distinguish between the brightness transitions from the text body to its background. For the recognition task they use a classical OCR engine that is fully integrated with their text extraction method. They report results on a test corpus of 1'100 images in 4 different European languages that have been extracted from laptop offers and job offers. They have created a ground truth set with the annotations of the text areas. Their performance evaluation method is based on counting the number of matches between the text areas detected by the algorithm and the text in the ground truth. They report of 80.45% detection rate. They perform a quantitative evaluation of the performance of the text extraction and preprocessing tool in combination with a commercial OCR engine. and can improve the recognition rate of commercial OCR by 20%. The authors argue in that work that the main reason why they do not achieve higher recognition rates is that they use a classical OCR engine to work with ultra low resolution images. They additionally proclaim that their future work would be devoted on integrating a specific *low resolution OCR engine*.

2.2 Text detection and recognition in video images

Digital videos play an important role in entertainment, education and other multimedia applications. In addition to this, new video applications like YouTube has been literally exploding in terms of quantity of data. As it is indeed nowadays extremely easy to produce a "private"' video and make it publicly available for example on YouTube. Therefore, Content-based indexation and retrieval of video images is an important application that needs automatic systems to

efficiently and properly search and index large video databases. We can find extensive studies in the literature that propose various text detection methods for specific applications like page segmentation, license plate recognition etc. Text in video images is often in ultra low resolution that depends on the poor quality of used screen device poor quality but has the advantage of not containing noise of scanned documents like additional noisy pixels.

Text can be found in video images in two different categories:

- Scene text that is embedded in natural scenes like shop names, street names, text on a T-shirt, street signs. Fig. 2.3 shows some possibilities of scene text in video images.
- Superimposed text that is artificially added to the video frames like in news videos, video commercials, sport videos and others as shown in Fig. 2.4. Such text has the additional advantage of being mostly unskewed as such text is added in rectangular frames to the video images

Various works are published dealing with detection and recognition of both text types in video images. Below we give a survey of the works for both types.

2.2.1 Scene text

Scene text appears mostly without intention in video scenes like in a scene of shopping mall with many shop names or on a surface like on a T-shirt. Such text tends to be very difficult to detect and recognize. Such text is embedded in a 3D scene and therefore can be rotated, tilted, slanted, partially hidden, partially shadowed, or be upon different surfaces.

Figure 2.3: Some examples of scene texts

Below we give a short survey of works devoted for detection and recognition of text in scene images. Additionally, table 2.2 shows an overview of these studies.

Jun Ohya et al. in [19] describe a method for character recognition in scene images. They have employed this method to read variously formatted characters with unknown size, font and gray-level under uncontrolled lighting. While the authors report reliable recognition results, their approach is restricted to text characters that are almost upright, monochrome and not connected in order to facilitate the text detection. In addition to this, in the presented work they have focused on still images rather than video streams. Their system is trained to extract isolated characters from scene images of road signs, sign boards of shops, etc.. The scene characters have

Table 2.2: Overview of the works for extraction and recognition of text in natural scenes in video images

Author	Year	Used OCR	Test Database	Detection Rate	Recognition Rate	Detection Method
Ohya [19]	1994	?	100	93.4%	66.5%	Local thresholding
Wu [20]	1999	-	650	93.5%	-	Text area identification
Shim [20]	2000	-	5'000	98.77%	-	Extracting candidtae text regions & text characteristics verification
Zhang [21]	2002	dedicated	2'000	?	85.7%	Local intensity normalization
Chen [22]	2001	commercial	4'000 frames	-	13.4%	Edge based

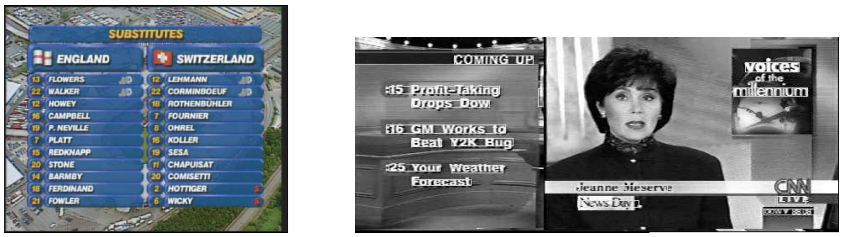
various sizes, gray-levels and formats under uncontrolled lighting conditions. They have tested on 100 images and obtained a character extracting rate of 71.1% and a character non-rejecting rate of 93.4%. They also report of an overall CRR of 66.5%. They do not mention the method they have used for recognizing characters but mention that the reported recognition rate is lower than the rate they obtained from current OCRs. But since they are working with scene images that contain different font sizes and uncontrolled lighting conditions, they consider their results as relaible.

Shim et al. in [20] propose a robust and effective text extraction system. This system can handle complex image backgrounds, deal with different font sizes and font colors that appear in a scene text such as normal and inverse video. They first extract candidate text regions and pass it through a verification process by removing the candidate region that is less than 12 pixels width or 4 pixels height. They apply their method to more than 5'000 frames. They then prepare a ground truth for each frame to assess their system's performance. They measure a detection rate of 98.77%. Finally, they do not report of recognition rates for the detected text but mention that their future works consists of developing OCR techniques specialized for video fonts.

J. Zhang et al. in [21] introduce a reliable approach for recognition of Chinese characters captured from a TV camera. Instead of using binary information as most OCR systems do, they extract features from image intensity directly. They normalize the image intensity and employ Gabor transforms for feature extraction and finally use LDA (Linear Discriminant Analysis) for features classification. They train their system on 6 kinds of different Chinese fonts of total 3'755 characters. For their test set they use 1'630 randomly selected characters from their sign library which contains more than 8'000 characters from more than 2'000 images. They achieve an overall CRR of 85.78% for their test set and up to 99.81% for their training set. As their system is tested on character images with different fonts, various lighting conditions, rotation and even affine transform, they consider their results as fairly satisfying.

Another interesting application for text detection and recognition in natural scenes is automatic recognition of vehicle license plates. One can find a large number of related works for license plate recognition like [23], [24], [25], [26] and many others. However, most of these works make assumptions that are only applicable to license plate recognition, i.e. they assume for example that the background of characters/numbers is mainly monochrome and that their location is restricted. Therefore, these works are out of our scope and we do not give a detailed survey here.

2.2.2 Superimposed text



(a) Sport video (b) News video

Figure 2.4: Example of video images containing superimposed text

One can find in the literature other synonyms for *superimposed text* like *caption text* or *artificial text*. However, superimposed text that is artificially produced and laid over the scene in a post-processing stage like for example embedded captions in TV programs or commercials or newscasts like on CNN or BBC, is often an important carrier of information and suitable for information indexing and retrieval. Superimposed text usually appears in clusters and lies horizontally and therefore does not have perspective deformations. In addition, superimposed text usually moves in an uniform way and unlike scene text does not have an arbitrary motion; moreover such text is usually monochromed but it can also have shadows for effective visualization.

Most of the published methods to detect and recognize superimposed text embedded in video images can be classified into four categories:

- *Connected-component-based*
  Connected-component-based methods detect text regions by analyzing the geometrical arrangement of edges or homogeneous color or grayscale values belonging to text. Connected component-based methods are efficient methods to extract text when text does not contain connected characters and background is monochrome and simple.

- *Texture-based*
  Such methods are based on distinct textual properties of text in video images. Texture based methods use usually specific knowledge of textural properties of the text region prior

to classification. These textural properties can be for example the contrast between text and background and the horizontal intensity variation of text that is different from its background.

- *Edge-based*
  The edge-based methods seek for features that are independent of contrast, font-color, font-size and language. Text string in videos has sharp transitions of color and luminance, i.e. edges, against its local background. Therefore edge-based features can be more reliable than font-color features as they contain two important properties: edge strength and edge density.

- *Hybrid methods*
  Hybrid methods combine two or more of the methods described above.

The majority of the related works to detect and recognize superimposed text in video images is concentrated on developing algorithms to localize and detect the embedded text. As a consequent of this, most of the researchers have applied various pre-processing techniques such as binarizing and up-sampling to enhance the image quality in order to employ commercial OCR engines. Few works report of developing a dedicated recognizer that is specifically trained to recognize the detected and extracted text. Table 2.3 shows an overview about works on detection and recognition of superimposed text in video images. Below we provide also a short survey of these works. A more detailed survey is provided in [7] by K. Jung et al..

Connected component-based

R. Lienhart and F.Suber in [27] describe a character segmentation method that operates on uncompressed frames and makes use of intra and inter-frame features of text appearances in digital videos. They employ a Split-and-Merge algorithm that performs segmentation by merging the regions with an average color together. They test their algorithm on 8 video samples and obtain a character segmentation rate ranging from 86% to 100%. They also test their own OCR software using a feature vector classification. They do not report on the obtained recognition rate in this work but mention that their software has to be further improved to reach the accuracy rate of commercial OCR packages. In [28] they test their segmentation algorithm on thirty video samples from title sequences of feature films, newscasts and commercials and obtain a character segmentation rate between 96% and 99%. They additionally report of a character recognition rate around 80%. The authors mention that the reason for the fairly low recognition rate originates from the narrowness of their current implementation of OCR software that is firstly trained only with 12 fonts and secondly does not deal with connected characters. However, They state that the recognition performance is still good enough for simple indexing of caption text. The authors have improved their algorithm in [29] so that individual characters have not different colors after text segmentation but have been binarized to enable a standard

Table 2.3: Overview of the works for extraction and recognition of superimposed text in video images

Author	Year	Used OCR	Test Database	Detection Rate	Recognition Rate	Detection Method
Lienhart [27]	1996	dedicated	8 video samples	86-100%	?	Connected components
Lienhart [28]	1996	dedicated	30 video samples	96-99%	80%	Connected components
Lienhart [29]	1998	dedictaed	30 video samples	-	46-76%	Connected components
Smith [30]	1995	-	63 video samples	63%	- ,	Texture based
Zhong [31]		-	8 video s samples	99%	-	Texture based
Shin [32]	2000	-	2500 keyframes	94.1%	-	Texture based
Lienhart [33]	1999	commercial	?	79.6%	69.9%	Texture based
Xi [34]	2001	commercial	40 minutes video	94.7%	67.5%	Texturebased
Sato [35]	1998	dedicated	210 minutes video	-	83.5%	Edge based
Chen [36]	2001	commercial	4'000 frames	-	82.6%	Edge based
Chen [22]	2000	dedicated	30 minutes video	97%	97%	Hybrid
Zhang [37]	2002	dedicated	3 sport videos	97%	95%	Hybrid
Shim [20]	1998	-	12 video samples	98.2-100%	-	Hybrid

OCR software to recognize them. The conventional OCR they incorporate into their system is Recognita V3.0 for Windows 95. They obtain a character recognition rate between 47% and 76%.

Michael A. Smith and Takeo Kanade in [30] propose a method to automatically create video browsing data that incorporates content specific video information. After detecting the text, they do not deal with its segmentation and recognition. They characterize text as a "horizontal rectangular structure of clustered sharp edges"' and use this feature to extract text. They test their method on 63 images and obtain an overall detection rate of 63%. As they do not explicitly mention how they asses the detection performance of their method, we assume that this has been done by trained operators. Additionally they do not use a classical OCR or a dedicated OCR for recognizing the detected text.

Texture-based

Y. Zhong et al. in [31] report on a texture-based caption text localization method which operates directly in the DCT domain for MPEG video sequences or JPEG images. The DCT coefficients in JPEG images or MPPEG video, which capture the directionality and periodicity of local image blocks are used as texture measures to identify text regions. Each unit block in the

compressed images is classified as either text or nontext based on local horizontal and vertical intensity variations. Additionally, they use postprocessing procedures including morphological operations and connected component analysis to refine the detected text. However, they use in their algorithm only the luminance and not the color information to locate text. They evaluate on 2'360 frames from 8 different video sequences. They report that 99% of the preseneted caption text is localized but they get still 1.58% of noncaption areas that are falsely accepted. The False rejects happens firstly when the font size or the gap between the characters is too big so that no strong texture is present and secondly when the contrast between foreground and background is too weak so that texture energy is not sufficiently high. Furthermore, the authors do not report of recognition results in their work.

C.S. Shine et al. in [32] describe a method that uses a small window to scan the image and classify the pixel located in the center of window as text or non-text using a SVM. To facilitate the detection of various text sizes, they use a pyramid of images generated from the original image by gradually reducing the resolution at each level. The classified images are then post-processed to generate the bounding boxes of text in the image. Then, they increase the resolution of the images to their original scale. They test their method on 2'500 key frames with a size of 320x240 manually selected Korean news archives and 200 commercials. 500 of these frames were used in the training time and the rest were used at testing time. They have manually tagged a set of rectangles representing text for each frame and compared the output of the classifier with those tags. The proposed method can detect 94.5% of the text regions with the false-detection rate of 4.2%. The authors report that the errors are primarily because of the low resolution. They plan to use a larger training set or incorporating domain-specific knowledge to overcome the false-detection rates. The authors do not report of a recognition step.

R. Linehart et al. in [33] propose a novel approach for localizing and segmenting text in complex images and videos. Text lines are identified using a multi-resolution feed-forward network that have been trained to detect text at a fixed scale and position. Localized text is then scaled to a fixed height of 100 pixels and segmented to a binary image with black characters and white background. Such text is then ready to be recognized with a classical OCR system. They have tested their system on a large dataset of complex video images and achieved a character segmentation rate of 79.6% and a character recognition rate of 69.9% .

J. Xi et al. in [34] report on a new system for text information extraction from news videos. They use a texture-based detection method by using edge maps as the texture features of text lines. They then use projection profiles to extract the text line. To increase the accuracy of text block identification, they use some rules to filter out non-text blocks. Additionally, thea have enlarged text blocks resolution and binarized the image to pass it through an existing OCR module. They have collected several CNN news videos with a total length of 40 minutes. They manually annotate each video frame so that the positions and the ASCII strings of each text line in this frame is stored into a ground truth file. Finally, they perform their detection

algorithm on this data and get a detection rate of 94.7% when comparing detected text with the annotations of the ground truth data. They also report of a CRR of 67.5% by passing the detected text through a commercial OCR software.

Edge-based

T. Sato et al. in [35] describe a VideoOCR technique that uses an interpolation filter to overcome the problem of low resolution characters and complex backgrounds of text embedded in news archives. They use edge properties of characters to detect text regions. They combine sub-pixel interpolation on individual frames and multi-frame integration across time to increase the resolution and reduce background variability. They further use 15x3, 3x7, 9x7 and 9x7 filters to detect vertical, horizontal, left diagonal and right diagonal elements that present a character shape. Then, they add the positive values at the same position among the filtered images to integrate four directional line elements. Additionally, they use a thresholding at a fixed value to produce a binary image which is used to determine positions of characters and recognize characters. They apply a simple segmenentation technique by using a vertical projection profile. The peaks indicates character boundaries. To avoid oversegmentation, they use a recognition-based character segmentation using a conventional pattern matching algorithm to recognize the characters. They evaluate their method using CNN Headline News programs. They use seven 30-minute programs that includes a total number of 2'370 characters. They obtain a CRR of 83.5%. Additionally, they implement a simple OCR program that consists of binarization of an image by straightforward thresholding at a fixed value and character extraction by horizontal and vertical projections of the binary image and matching by correlation. The test data is passed through the implemented OCR and delivers a character recognition rate of 46.5%. Therefore, the VideoOCR technique is twice more efficient as method of the simple OCR. Although the authors gain a reliable character recognition accuracy by their VideoOCR, the word recognition rate of 48.3% is still insufficient. By incorporating the differences between recognition results with words in a dictionary and selecting a word having the least difference, they can increase the word recognition results to 65.2%.

Hybrid

D. chen et al. in [36] present a two-step method for text detection. They propose a localization/verification scheme that quickly extracts text blocks in images with a low rejection rate. This localization process allows the authors to further extract individual text-lines and normalize the size of the text. Then, they perform precise verification in a set of feature spaces that are invariant to gray-scale changes. A post-processing step as a new gray-scale consistency algorithm is proposed to improve segmentation results. The text localization step is evaluated on a half hour video from a Belgian news program. The performance of the text localization step is measured in terms of rejection rate and precision. They get no rejected regions, whereas

the precision rate is 55.4%. Then, they test their text verification algorithm on a database consisting of still images and half an hour of video recorded from TV. The verification scheme of the authors has removed 7'225 regions of the 7'537 false alarms and gives a 0.24% rejection rate and a 97% precision rate. The authors have implemented a multiple hypotheses recognition scheme and report of a character recognition rate of 97% and a 93% word recognition rate. The authors state that the performance of their proposed system is good enough to be used in a video annotation and indexing system.

D. Zhang et al. in [37] propose domain-specific video algorithms for extraction and recognition of superimposed text in digital video. Their system uses DCT coefficients and motion vectors as features for localization of superimposed text. Additionally, they employ long-term temporal consistency to enhance the localization performance. For character segmentation, they combine unsupervised cluster-based classification and local minima searching of projection profile to obtain more accurate segmentation. For text recognition, they use Zernike moments. By exploring domain-knowledge and a statistical transition graph, they finally enhance the recognition rate by inferring constraints about domain-specific knowledge, such as ball counts and game score of baseball videos. They test three baseball videos and one NBA basketball video as experimental subjects. To perform evaluation tests on their data base, they produce groundtruth keyframes and obtain only 3% misses and 21.9% false alarms. In addition to this, they perform character recognition on the detected keyframes. The recognition algorithm is not precisely described but it is obvious that the authors use a dedicated recognizer presumably a pattern matching algorithm. They achieve character recognition rates of 92%. After applying some domain knowledge about strike-ball image sequences, they can improve the character recognition rate to 95%. Their system is a mono-font system and can not handle yet font variations but they plan to improve it to be a multi-font system. The authors claim that though their system is primarily implemented and tested for baseball videos, it can be generalized and used in other video text extracting applications such as news and films.

In addition to the above works, one can find in the literature few works that describe algorithms that deal both with scene text and superimposed text. Below we list some works dealing with both text types in video sequences.

Shim et al. in [20] present a text extraction system that is robust enough to handle scene text as well as superimposed text. Moreover, their algorithm can deal with different font sizes, font styles and font appearances such as normal and inverse video. Besides, their algorithm can deal with noise and artifacts introduced by block-based MPEG encoding of digital video. They test their text extraction system on 12 video streams whose play-durations varies from 10 seconds to 1 minute. The total number of tested frames are 5'000. Their text extraction algorithm has a character miss rate between 0-2.68%. The authors do not mention recognition rates in their work but state that their future work is directed towards handling multi-colored text and developing a dedicated OCR specialized for video fonts.

In [22] Chen et al. locate the sub-structures of the text in video images using edge-detection

and estimate the orientation and scale of each sub-structure using a family of filters. Besides they select three scale ranges and enhance the contrast of these sub-structures as character strokes in each scale range individually to improve the performance of both text detection and segmentation. In order to use a standard OCR system, the detected text region is normalized and then directly binarized to segment text and background. They use TypeReader OCR package and report of character recognition rates of 7.4%-13.4% for scene text and of 36.1%-82.6% for superimposed text. This experiment shows clearly that the recognition of scene text in comparison to the superimposed text is indeed more difficult.

2.3 Text detection and recognition in images captured from digital cameras

With the fast popularization of digital camcorders and mobile phones or PDAs with integrated digital cameras, the necessity of camera based character recognition has been immensely increased. One could think of different applications in this field like tourist guide system, navigation system, meeting archiving systems, wearable cameras for visually impaired persons, documenting ancient books that are too sensitive to touch, etc. One could additionally say that though these are important applications, but the size of their market is limited. However, an important application is probably the indexation of personelle or publicly available pictures on the web. A Gartner group survey [38] says that 48 percent of cellular phones will incorporate cameras by 2006 and 81 percent will have them by 2010. Consequently, people use more often their digital cameras to capture text instead of using bulky scanners. Therefore, the amount of camera-captured documents is growing constantly and this automatically can boost and accelerate the research activities of camera-based document analysis.

Despite the fact that the field of traditional scanner-based document analysis has been extensively researched during the last 30 years [39], their techniques can not be applied directly on camera-based images. This is because camera-based images have additional sources of noise due to their perspective distortions, blur, lightening conditions, low resolutions and low brightness contrast. Notwithstanding, the constant growth of resolution of mobile cameras (recently min. 2 Megapixel) is a motivating factor making them enough versatile tools to be integrated in future document analysis systems.

Various researchers have been recently working on the field camera-based document analysis. These works are either focused on a specific application like sign translation, systems for visually impaired persons, document archiving, etc. or on a specific technique like image mosaicing to better zoom and capture pieces of a full document, capturing documents under predefined angle of a digital camera, working with bigger fonts but having arbitrary camera angles, etc. Most of the proposed systems concentrate after text detection on techniques for pre-processing the captured image like perspective normalization and warping, image enhancement methods like interpolation, deblurring, adaptive thresholding, etc.. Then, the obtained high quality images are

binarized and ready to be recognized with commercial OCRs. A few works have tried to develop a dedicated recognizer that takes into account as much as possible the specificities of such images and need as few as possible image pre-processing techniques. Most of the researchers argue that the reason here fore is that the classical OCRs have reached a high level of recognition maturity so that it is more advantageous to enhance the image quality for passing through commercial OCRs. On the other hand, the same authors claim mostly in the conclusion of their studies that there is still a substantial need for a dedicated OCR to increase the recognition accuracy of the detected text from images shot by digital cameras.

Table 2.4: Overview of the works for extraction and recognition of text captured from digital cameras

Author	Year	Used OCR	Test Database	Detection Rate	Recognition Rate	Detection Method
Zandifar [40]	2002	commercial	?	-	98%	Hybrid method
Ezaki [41]	2004	-	504 images: mostly chars $>$ 30 pixels height	Precision: 0.56 Recall: 0.7	-	Connected components
Ezaki [42]	2005	-	75 images: only chars $<$ 30 pixels height	Precision: 0.62 Recall: 0.64	-	Fischer's discriminant rate
Doermann [43]	2006	-	?	?	?	Image mosaicing technique
Uchida [44]	2006	-	?	per line: 84%	-	Character cross ratio
Omachi [45]	2006	commercial	?	?	95.5%	Embedding inf. in char patterns
Bae [46]	2006	commercial	6'756 characters	?	99.19%	hybrid method
Ohya [19]	1994	dedicated	100 text images	93.4%	66.5%	Local thresholding
Zhang [21]	2002	dedicated	8'000 chars	-	99.7%	Gabor transforms & LDA
line Gao [47]	2001	-	?	93%	-	Adap. color modeling & searching algorithm

In [40] Zandifar et al. propose a camera based system that speaks the textual information for visually impaired people. The system consists of a computer, a digital video camera, an audio interface and a classical commercial OCR. The camera captures text from the scene with full control of focus and zoom. They apply different operations on captured images such as image mosaicing to enhance OCR results, auto-focusing to gain the best focusing positions, auto-zooming to reach OCR font-size. They have developed a system conssitsing of a digital camera, a computer and loudspeakers. The system scans document images in the environment of the visually impaired and converts them to speech. A commercial OCR is integrated into their system. The authors report of obtained recognition rates of bigger than 98%. However they do not mention if such rate is character or word based.

Ezaki et. al. in [41] describe a system design for a camera-based reading system that extracts text information from natural scenes images. In their study the authors find out that the system

effectiveness is strongly dependent on character size. This would be specially the case for the natural scenes acquired by a visually impaired person. Text extraction of their system is based on connected component method. First, their system tries to find images containing text areas with small characters. Then it zooms into the found text area to retake higher resolutions images necessary for character recognition. To evaluate their algorithm, they use the dataset available in ICDAR 2003 Robust Reading Competition. This dataset contains 504 realistic images with textual content. They report of a precision rate up to 0.56 and a recall rate up to 0.7. They have improved their system in [42] to cope with more complex backgrounds and to have more accuracy when extracting smaller font sizes. The proposed system tries to find out the image areas with small characters to zoom them into the found areas to enlarge the resolution that is necessary to recognize text by the classical OCR systems. A new text extraction system is introduced that uses Fisher's Discriminant Rate (FDR) to decide whether an image area should be binarized using local or global thresholds. If the text area is detected in the initial image, the camera zooms-in to obtain more detailed images of each candidate text area. The higher resolution characters are then recognized and read out to the blind person via a voice synthesizer. For evaluating the performance of the proposed system, they use the same database as the first study (ICDAR 2003 Robust Reading Competition). To more precisely assess system performance that is especially improved for small characters, they choose from this database 75 images containing characters smaller than 30 pixels in height. They achieve a Precision rate up to 0.62 and a Recall rate up to 0.64. For character recognition, they use a commercial OCR, a dedicated character recognizer is not implemented yet.

In [48] Clarck et al. describe a novel automatic text reading system using an active camera for OCR. The authors use for their system a low-resolution, stationary video camera with pan, tilt and zoom control to actively generate a high resolution representation of a previously located text region and determine how to best partition the region into a set of tiled segments. These segments are finally mosaicied together using a simplified cross correlation. The final mosaic is then fed into an OCR engine. They have tested their algorithm on two text images. The first one delivers 90% word accuracy and the second one delivers 83% word accuracy. The same authors in [49] propose a method for recovery of paragraphs of text under full perspective transformation in a single image. They use the projection profiles from hypothesized vanishing points in order to recover accurately the lines of text. After estimating the horizontal and vertical vanishing points of the text planes, they transform the text to a so called *'fronto-parallel'* view and apply image enhancement methods to increase the resolution to be suitable for classical OCRs. The authors state that their algorithm performs well for a wide range of paragraphs, provided each paragraph has at least three or more full lines. The author do not report either of explicit evaluation performance of their system or of recognition results when using a commercial OCR like in [48]. They intend to improve an automatic recognition system in their future works.

In [43] Doermann et al. present an image mosaicing technique for camera-captured images to reconstruct a full page image. Their image registration method can align document images with

as little as 10% overlap and severe perspective distortions. The number of mosaics in their test varies from four to eight. In the overlapping area, they apply a sharpness blending across the border and within. They also propose an image blending method that is optimized for document images, which addresses the inconsistent lighting, 'ghost' image and varying sharpness problems. Then, they have applied their method in full A4 page document mosaicing experiments. The authors state that in all cases the registration is accurate but they do not mention an explicit performance of their system. In addition to this, the authors do not address the recognition task in their work.

In [44] Uchida et al. present a method that uses the cross ratio of each character class. Each character image is printed with a pattern that comprises five strips and the cross ratio is then calculated from these strips. As cross ratio is projective invariant, the class information can be extracted correctly regardless of camera angle. They have evaluated their method on a limited set of 167 character images captured by digital camera from various angles. The experimental results shows a line detection rate of 80.4% . They do not conduct a recognition process.

In [45] Omachi et al. propose a method of embedding information in a character pattern so that the class of the character can be identified. Their algorithm should be robust against geometric distortions that is usually the case for camera-captured images. They assign two colors to each character pattern like adding shadow or using different color for the character profile line. They use character patterns of 500 pixel height and generate shadow by moving the character pattern right by ten pixels and below by ten pixels. Then the area ratio of the shadow and the character pattern is calculated. To assess performance of the proposed system, they use twelve word patterns that were used for traffic signs generated with their character patterns. Each word is photographed in sunlight by a digital camera with three camera angles. They use 4 font families and achieve an overall CRR of 95.5%.

In [46] Bae et al. describe a camera based character recognition system, which is implemented for mobile devices such as PDA and cellular phones with color cameras. The authors have firstly developed a camera based character recognition system for PC that includes techniques such as image enhancement, local adaptive binarization and noise reduction in order to effectively extract character regions. Then, they apply vertical projection profiles to segment the the lines into words and further to characters. This method works well for the Korean text for two reasons: first the Korean characters are almost four times bigger in size than Latin characters, second the Korean characters are not connected and though vertical projection profiles are able to clearly discriminate the character boundaries. Finally they convert the PC based OCR system to an embedded system for cellular phones. They evaluate their system using camera document images of the ETRI database. They have trained MLPs with 11260 characters and tested with 6'756 characters. They report reliable character an overall character recognition rate of 99.19%.

In [19] Ohya et al. propose a flexible new method for character recognition for scene images. Their method can deal with a variety of illumination conditions and without a-priori knowledge of the size, font, position, gray level and format of the characters. Their method uses a local

thresholding that does not need a training process for image segmentation. The differences of gray-level values between adjacent regions is used for character segmentation. A character recognition process calculates the similarity between a pattern candidate and each category in a dictionary. They use a relaxation operation to boost the recognition and remove the ambiguities and contradictions between similar patterns. They measure the performance of their algorithm on 100 images in which characters are printed on road sign, license number plates of cars, sign board of shops, etc.. They achieve a character detection rate of 93.4% and and a character recognition rate of up to 66.5% with their relaxation approach. The authors state that the recognition rate is lower than a commercial OCR that they currently use, but since the dictionary used in the experiments does not include all the fonts like the current-use-OCR, the results are considered appropriate. However, their system assumes that the characters are almost upright, not textured, having the same gray level in different images and not connected.

In [21] Zhang et al. present a robust approach for text embedded in natural scenes. The authors use local intensity normalization for image enhancement and Gabor transforms for feature extraction. In order to reduce dimensions of feature vectors to make them computationally less cost intensive, they utilize system LDA. They have evaluated the proposed system on a Chinese sign recognition task. Their system includes 6 different chinese fonts each containing more than 3'000 characters for training set. For test sets they used 2'000 chinese sign images containing more than 8'000 characters under various perspective and lightening conditions. By combining two wavelength Gabor features they achieve reliable character recognition rates up to 99.7%. Their system is now integrated into an automatic chinese sign translation system.

Gao and young in [47] describe a system for sign detection, recognition and translation from natural scenes that are shot for example by international tourists with digital cameras. They propose an automatic text detection method based on an adaptive color modeling and searching algorithm. Color modeling is optimized by using EM algorithm under the constraint of text layout relations for a specific language. Their text detection model works fairly well with an overall detection performance of 93%. However, the authors do not explicitly state recognition performance of their proposed system.

2.4 Screen-shot images

2.4.1 Recognition of screen-shot images with commercial OCRs

Since recently there exists softwares that can recognize text in images as a screen-shot using for ex. Adobe PDF or Word 2007. We fed such a commercial OCR, i.e. AbbyFineReader Intelligent OCR, with screen shot images of different zooms and different backgrounds. We found out that this OCR has three basic limitations:

- **Text with complex layout**
  Fig. 2.5 shows a website with a complex layout structure. The selected OCR is unable to

segment all the contained images correctly as the website contains images with complex backgrounds like the scene pictures having different colors and shapes. The selected OCR cannot accurately segment such images with complex backgrounds. Consequently, the OCR is not able to properly recognize the text in the non-segmented areas. However, this limitation is out of our scope, as we did not deal with text layout analysis in this dissertation.

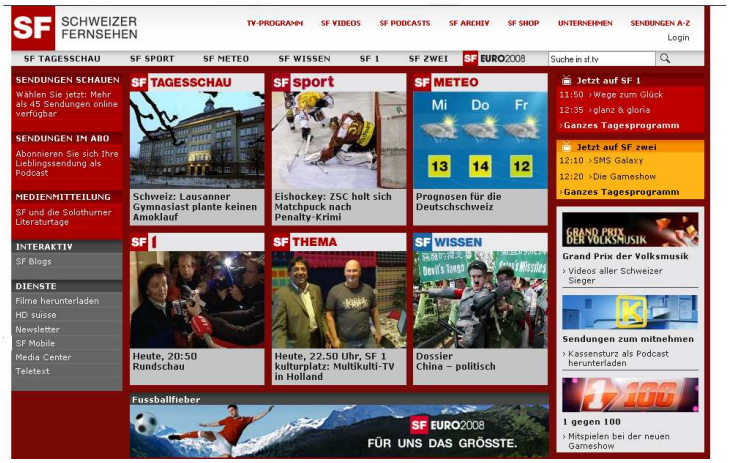


Figure 2.5: Website with complex layout

- **Text with complex backgrounds**
  Fig. 2.6 shows examples of buttons containing text. Such buttons are used frequently in web sites. We have chosen these images, because the buttons have a large variety of backgrounds. Some backgrounds are monochrome like the third button in the top row. Some are designed with uniform color like the forth button in the top row. Others have gradient colors like the second and third button in the second top row, etc. Furthermore, text is rendered both in black or white. Fig. 2.7 shows the recognition results of the corresponding buttons. The buttons with gradient backgrounds are not-recognized. Besides text in button images with gray or white backgrounds or backgrounds with an uniform color or monochrome backgrounds are mostly well recognized. In addition, the selected OCR can recognize both black and white rendered text, presumably it simply inverts the pixels of the segmented white colored text to black and then recognizes it.

- **Text with small font sizes, i.e. < 10 poins**
  Another limitation that such OCRs show is that they can either fail to, or only partly recognize the screen-rendered text with small font sizes ($10pts$). Fig. 2.8 shows some examples we tested. These examples were chosen from our synthetic single word database that we produced for our work. The details of this database can be read in chapter 3. The fonts are *serif (Georgia, Times New Roman)* and *sans serif (Verdana)*. In addition, we tested two basically different font styles: *roman (plain)* and *italics* as we speculated that

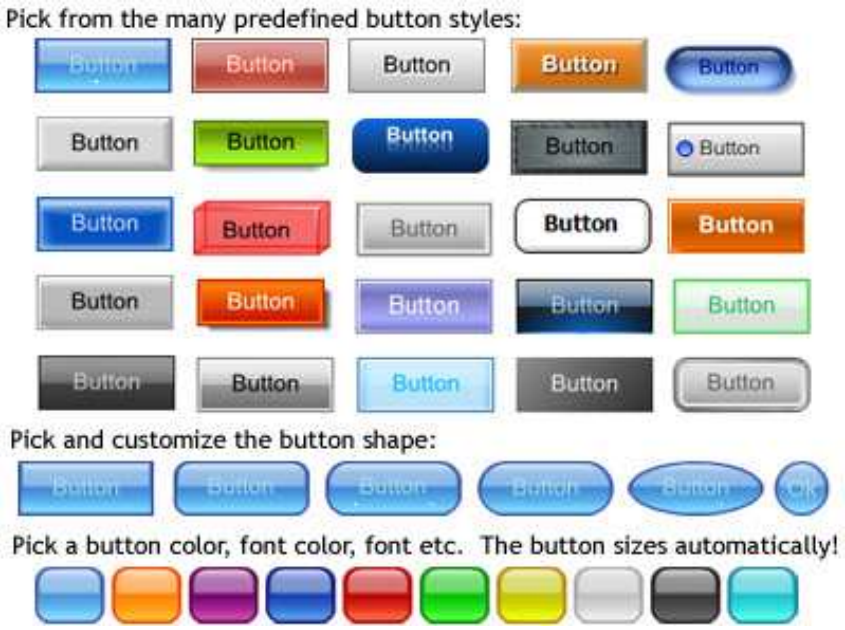


Figure 2.6: Buttons containing text with different backgrounds

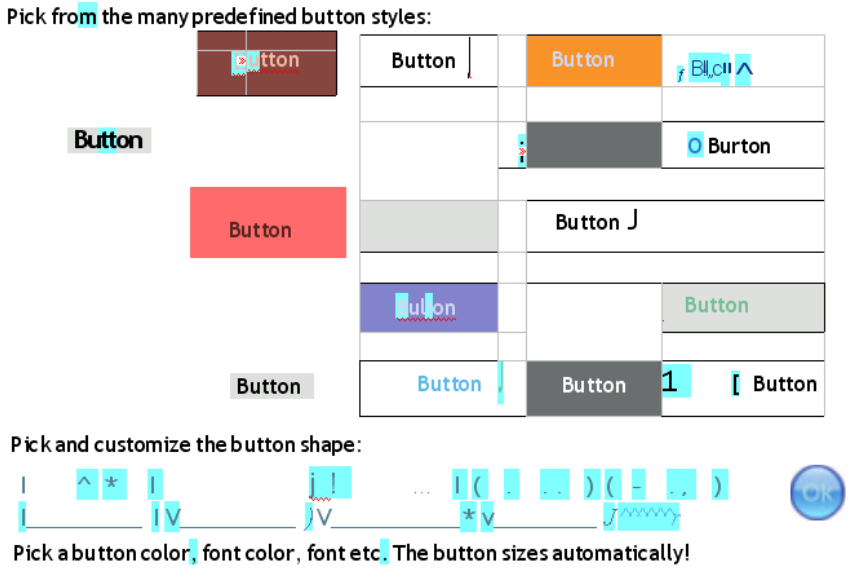


Figure 2.7: Recognized text in different buttons

recognition of *italics* text in small font sizes could be an extra element of difficulty for the selected OCR. The results show that the selected OCR has major difficulties recognizing screen shot (screen-rendered) words with small font sizes. In addition, recognition of *italics* style seems to be even worse than *roman* style.

Font family	Font group	Font style	Point size	Word (original size)	Word (close-up size)	Recognized text
Georgia	Serif	Plain	9 pts.	band		Inn::!
Georgia	Serif	Plain	8 pts.	fright		*tri/ht*
Verdana	Sans serif	Italics	9 pts.	*band*		-
Verdana	Sans serif	Italics	8 pts.	*pack*		-
Times New Roman	Serif	Plain	9 pts.	band		band

Figure 2.8: Examples of limitations of the selected OCR to recognize ULR, anti-aliased words with small points sizes

2.4.2 Works in detection and recognition of screen-shot images

An interesting approach for detection and recognition of screen-shot images is the recent published works of Wachenfeld et al. [50] [51] [52] [53]. The authors report in their works on a recognition system for *screen-rendered text*. The kind of the *screen rendered text* which they use are text in screen shot images that has font sizes < 10 points. Such a definition of their work make them a parallel approach to ours and though we give below a short survey of the database and the recognition system they produced and implemented. As noted above, the terminology the authors use in their published works for such text is the *screen-rendered text*, whereas in our work we use in general the term *ultra low resolution text*.

Due to the fact that no standard databases exist for ultra low resolution, anti-aliased text at small font sizes, the authors build two freely available databases [54] that they describe in [53]. The first database is the so called *Screen-Char* database, which holds currently 28'080 screen-shot images of single characters. They use this database for training and test of classifiers for screen-rendered characters. The second database is the the so called *Screen-Word* database and contains two data sets: a collection of 400 *embedded words* and a collection of 2'400 *isolated words*. This database is used to train and test their recognition system. The embedded words

have been obtained from several existing documents and represent real-world data. The isolated words are synthetically generated for specific combinations of font type, font style, font size and other rendering conditions such as sampling grid. They have chosen 20 words for building the synthetical isolated word database. The first 10 words are words containing all letters in Latin alphabet both for lower and upper case letters and the other 10 words have been chosen randomly while taking into consideration that some character pairs like *rn* and *cl* leading to confusions with *m* and *d* were included in random words. The proposed recognition system of wachenfeld et al. is shown in Fig. 2.9.

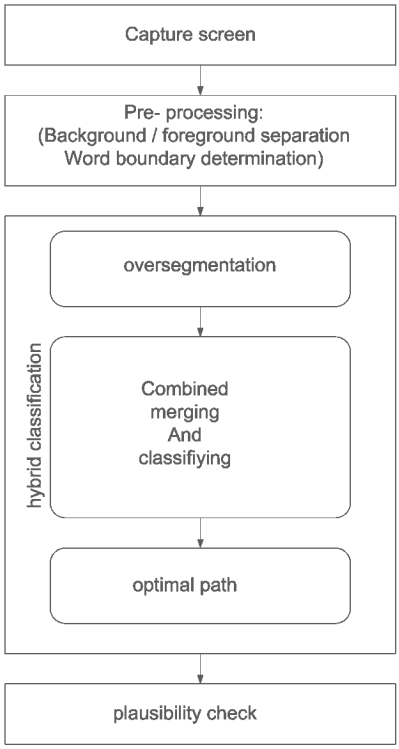


Figure 2.9: Recognition system Wachenfeld et al.

Their approach consists of 6 steps:

- **Pre-processing**
  After capturing and loading a screen-shot image, they first use a threshold to separate background from foreground. Next, they determine the word boundaries by processing components of connected foreground pixels. They finally group components close together and remove those which cannot be characters due to their form or size.

- **Segmentation**
  The authors use a soft segmentation, where the choice between multiple segmentation

candidates is based on recognition. They first split the foreground pixels of the word boundary region into a sequence of smaller units. This results in an over-segmentation. For an image $w \times h$ they compute w segmentation paths of height h which minimizes a cost function using a dynamic programming algorithm. The paths are laid over a graph and the $n-1$ paths are selected iteratively using a threshold w_{min}.

- **Feature extraction**
  In this step, the authors segment the images into several 5 x 5 blocks. They compute the average gray value of each block using sub-pixel precision. They then get a 25 dimensional feature vector for each block. The authors state that many features like contour based approaches have difficulties with self-touching or broken characters at small font sizes. Additionally they find that typographic grayscale features as proposed in [55] are not suitable. Further, the authors state that the Hu's seven moment invariants and Fourier descriptors need a much higher spatial resolution of 64 x 64 or 128 x 128 to work properly.

- **Classification**
  For the classification task the authors use 48 character classes from 52 lower and upper case letters, i.e. the authors merge eight pairs of very similar classes (o/O, c/C, s/S, v/V, w/W, x/X, z/Z) and add two additional classes. They then compute a plausibility as the average Euclidean distance between the corresponding feature vector and class c_k. The classification result for a single segment is a list of classes ordered by a plausibility test.

- **Optimal n-best path: Hypothesis graph**
  A hypothesis graph is built in parallel during the over-segmentation step. A complete graph would consist of the resulting sub-components and all possible merging of neighborhood sub-components. To avoid huge graphs they have limited merging by aspect ratio conditions. For each segment in the graph they compute the plausibility that the segment belongs to the class c_K. This is done for all classes. The concatenation of classifications along all paths results in word candidates. For each word candidate they compute a plausibility as a product of the classification plausibilities of its segments, weighted by their segment's width in pixels w_p. The result is a list of word candidates ordered by plausibility.

- **Post-processing: Plausibility check**
  This post-processing task uses knowledge about typography to modify or remove the order of word candidates. They check here whether the classification result for a segment is conform with its height or position. The result of this step is an improved word candidate list.

Further, the authors perform evaluation tests to show the performance of their recognition system. They do not use dictionaries or language models for their tests. The only constraints that is employed in their system is based on typographical knowledge at character level. They perform two different evaluation tests. The first experiments test the performance of the isolated

characters. They use 20'080 ultra low resolution isolated characters for training. For test, they use a test subset of 15'808 plain characters and gain a character recognition rate of 98.91%. In addition, they test their system on 400 embedded words. The authors report that in 347 cases the correct word is in the first place and in 383 cases the correct word is under the first ten word candidates.

2.5 Thesis motivation

To summarize, the majority of the works we have introduced for detection and recognition of text in digital images report mostly of significant results at *text detection* level. Furthermore, these works use a commercial OCR for recognition of the extracted text, as they claim that such softwares have reached a high accuracy above 99.9%. However, the commercial OCRs have a major limitation, as they are designed to recognize bilevel text that has a minimum resolution of 150 dpi. In this dissertation, we have dealt with the problem of recognition of anti-aliased text with small font sizes such as those frequently embedded in web images. Such text has a poor quality between 72-100 dpi(computer screen resolution) and therefore has to be enhanced before feeding to commercial OCRs. We could think of another approach that employs a *dedicated recognizer* for such text that would not use methods such as normalization, binarization and others.

However, to the best of our knowledge and with the exception of the recent works of Wachenfeld et al., we could not find any research work aiming at developing and using a *dedicated recognizer* for text at ultra low resolution that is anti-aliased and is rendered at small point sizes. Such a dedicated recognizer has the clear advantage to be directly applicable on the detected text without using cost-intensive image pre-processing methods. We believe that such a recognizer can indeed deliver better recognition results and provide a substantial contribution to a more accurate retrieval and indexation of textual information encapsulated in the web pages. Therefore, we have studied the recognition task for isolated characters and isolated words and have built different single word recognizers that were based either on a real-word dictionary or were even not dependent to a vocabulary and even language. Further, we believe that we can use the principles of the presented system for the recognition of text embedded in other digital images. However, our system needs to be modified in some certain aspects to be applicable for recognition of text in video images and those images shot from mobile cameras.

Chapter 3

Low resolution text rendering

3.1 Digital typography

Typography is at least as old as the printing system in the history of human civilization. There are basically two different models regarding possible interactions of the typographical development. Some historians view the development of this technique in "the Far East" (China) as separate from that occuring in mid-15th century in Europe, while others view them as connected. Digital typography is a field that overlaps two others: a) classical or letterpress typography and b) computer science [56]. A detailed explanation of the complex field of typography and in our particular case digital typography is beyond the scope of the presented work. In this chapter this chapter we present some descriptions on the topics relating to our work about recognition of anti-aliased text at ultra low resolution with small point sizes.

Typography is the art and techniques of type design, modifying type glyphs, and arranging type. Type glyphs (characters) are created and modified using a variety of illustration techniques. The arrangement of type is the selection of typefaces, point size, line length, leading (line spacing) and letter spacing. A font is a set of glyphs (images) representing the characters from a particular character set in a particular typeface. A glyph is the actual artistic representation of an abstract glyph, in some typographic style that may be drawn on the screen or paper. From the mid-1980s, as digital typography has grown, type designers have adopted several printed font formats as computer fonts. Computer fonts ("digital font") are stored in a computer file containing letter forms . Principally there are two different requirements that typography has to fulfill. These are the *readability* and the *legibility* of the font. As these two terms are mostly confused together, we give a short description of them to be able to make a clear distinction between these two terms:

- *Legibility*
  Legibility is that characteristic of the typeface that allows the eye to distinguish one character from the other. In some fonts, the actual shapes of some letters cause the typeface to have a depreciated legibility. For instance, a lower case *'i'* next to another

straight, upright character like an *'l'* or *'t'* makes these characters illegible in certain font families. Therefore, legibility is built in to the font by the font designer.

- *Readability*
  Readability is the relative ease with which a typeface can be read when characters are arranged in words, sentences, and paragraphs. It concerns the difficulty of the language itself, not its appearance. Factors that affect readability include sentence and word length, and the frequency of uncommon words. Readability is therefore more important for printed text, whereas the more relevant factor that matters for text on low resolution screens especially when embedded in images, is its legibility, as such text is generally not in long sentences and paragraphs.

Some fonts, such as Verdana, are designed primarily for use on computer screens and we assume that more effort has been done on their *legibility* than *readability*.

Digital fonts store the image of each character either as a bitmap or by geometrical description of their outline also called a vector font. The main advantage of outline fonts is their scalability. A font family is typically a group of related fonts which vary only in size, scale, weight, orientation, width, etc, but not design. For example, Arial is a font family, whereas Arial Roman, Arial Italic and Arial Bold are individual fonts making up the Arial font family. Font families can be divided into two different groups:

- Serif fonts
  Serif fonts have some decorative embellishments at the end of their strokes. Common examples for such fonts are Times and Georgia. Serif fonts are the most used fonts in printed material like books, newspapers and magazines. Fig. 3.1 shows a sentence written with the serif font 'Georgia'.

The quick brown fox jumps over the lazy dog

Figure 3.1: An example of a text line written by serif font 'Georgia'

- Sans serif fonts
  Sans serif fonts are free of serifs as the name suggests. The low contrast and absence of serifs makes most sans serif fonts harder to follow for general reading. They are fine for a sentence, passable for a paragraph, but are not well suited to be used in the text of a book. Sans serif fonts are commonly used for display typography that demands legibility above high readability. For example most web pages use modern sans serif fonts, because it is commonly believed that, in contrast to the case for printed material, sans serif fonts are easier than serif fonts to read on the ultra low resolution computer screen. For instance, since Verdana was first shipped with Internet Explorer in 1996, it has become one of the most widely used fonts in web sites round the globe. Fig. 3.2 shows a sentence written with the sans serif font 'Verdana'.

The quick brown fox jumps over the lazy dog

Figure 3.2: An example of a text line written by sans serif font 'Verdana'

Additionally digital fonts contain glyph descriptions, font metrics information that are used for text composition.

3.2 Font design

The early approach of creating digitized characters was to draw them by hand and then digitize it manually point by point using an appropriate software such as Ikarus [56]. Another approach is to simulate the hand-drawn designs with the mathematical specification of pen trajectories and shapes by using Metafont [57]. Characters can also be digitized by scanning the drawing and using an interactive outline editor like Fontlab's TypeTool, FontLab Studio [58], Fontographer [59], etc..

Applications using digital fonts and their rasterizers (font engines), appear in Microsoft and Apple Computer operating systems, Adobe Systems products, desktop publishing applications and other applications from various companies. Fonts can be designed either as bitmap fonts or outline fonts.

3.2.1 Bitmap fonts

Bitmap fonts store each letter, number, symbol for every point size, style and resolution with a unique bitmap. The bitmaps are a bunch of rows and columns of pixels that are on or off. For example, a font that has three sizes, and any combination of bold and italic, needs 12 complete sets of images. Bitmap fonts have the advantage that they are fast and simple to render but they generally show the disadvantage of having a visually poor quality than scalable outline fonts. Fig. 3.3 shows a bitmap font written on different point sizes on the computer screen. Bitmap fonts were used in the early days of computer fonts and are mostly replaced by outline fonts.

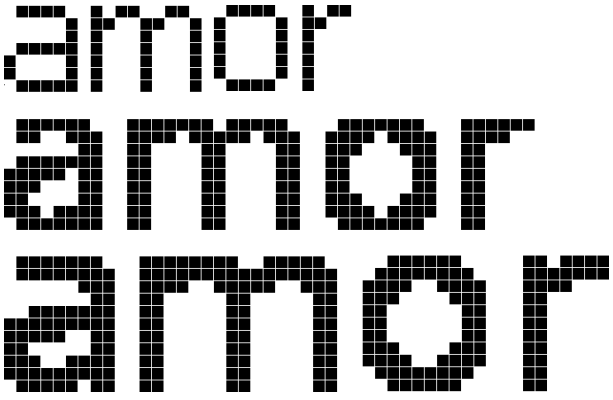


Figure 3.3: Bitmap font in different font sizes

3.2.2 Outline fonts

When an outline font is used to be rendered on the computer screen in a specific size by an application, a ratserization routine uses an already stored character's outline, usually at big resolution, and renders it to the desired size and then creates a bitmap of the letter on the fly. It takes nearly 20 milliseconds to render the font. The advantage of outline fonts is that they save space on the disk by creating an outline for each letter rather than calling up a discrete bitmap from memory. The disadvantage of outline fonts are the rounding errors that happen while the resulting bitmap has to be fitted into the computer grid. To overcome this drawback, some font engines use techniques like *hinting* and *anti-aliasing* to reach an optimal *legibility* of that character on the computer screen. Fig. 3.4 shows the outline of character **'M'** in different sizes and transformations.

(a) rotated (b) small (c) big (d) mirrored

Figure 3.4: An example of scalability of outline fonts: Character **'M'** in different transformations and scales

Common outline font formats are *PostScript Type 1 and 3* [60], *TrueType* and *OpenType* [61] and *METAFONT* [57]. TrueType (TT) and PostScript Type 1 (PS1) are both *Multi-platform* outline font standards for which the technical specifications are openly available. *Multi-platform* means that both font types are usable on multiple sorts of computer systems. Postscript fonts were introduced by Adobe at early 1980 and were the first computer fonts that have been taken seriously by the printing industry. They have been used in Macintosh operating system. The fast popularity of PostScript fonts was due to the fact that Macintosh operating system provided any application with the ability to use any fonts installed on the system. Doing so, more software developer were encouraged to work on Macintosh operating system as at that times fonts were costly and software developers had to pay extra charges on font designer companies to be able to integrate them in their software. Facing this successful approach of Macintosh, Microsoft has launched the same strategy and has made True Type accessible with the windows operating system.

- **Postscript Type 1 and Type 3**
  As already said, Postscript Type 1 and Type 3 were developed by Adobe for professional digital typesetting. The glyphs in PostScript are outline fonts described with cubic Bezier curves. Type 1 fonts were restricted to a subset of the PostScript language that has

been originally developed as a language to process graphics. Type 1 and Type 3 fonts use additionally hints that are not publicly available and though can not be copied or imitated.

- **True Type and Open Type**
  Apple was irritated that Adobe licensed PostScript to printer manufacturers who undercut Apple's own LaserWriter. So, Apple and Microsoft agreed a cross-licensing and product development deal, the fruits of which would be available to both parties: Microsoft would bring a PostScript-style graphics engine to the table (TrueImage), while Apple would create a font system even better than Adobe's. The outcome of this cross-license was the font TrueType developed in late 1980s as a competitor to Adobe's Type 1 fonts used in PostScript. Therefore True Type is a joint-work of both companies and has been used in their both operating systems. The first difference between TrueType and PostScript fonts is their use of different sorts of mathematics to describe their curves. The primary advantage of TrueType over PostScript1 fonts is the fact that TrueType allows better hinting. TrueType hints can do all that PostScript can, and almost anything else, as defined by the very flexible instructions. Hinting is described later in this chapter.

- **METAFONT**
  METAFONT [57] is a programming language used to define vector fonts. It is also the name of the interpreter that executes METAFONT code, converting the vector fonts into bitmap fonts that can be included in PostScript documents. *METAFONT* uses a different sort of glyph description. Thus, rather than describing the outline of the glyph directly, a *METAFONT* file describes the curve traveled by the center of the pen, and the pen's shape is allowed to vary as the pen moves. The result is much like an outline font, but with slightly softened corners defined by the pen shape. LaTeX has initially used *METAFONT* fonts. Not too long ago (before Adobe Type Manager (ATM) was available for the Macintosh and Microsoft Windows), fonts were too expensive to be generally available. In those days, a font building tool like *METAFONT* was essential if TeX was going to have a number of typefaces and a large number of mathematical symbols. Today, the role of *METAFONT* is diminishing. Many people choose to use PostScript fonts almost entirely. In fact, with PostScript alternatives to the Computer Modern Math fonts now available, it's possible to use TEX without using *METAFONT* fonts at all.

Hinting

Hinting is a technique to fit the character outline into a given grid. Fitting an unmodified outline into a grid of an output device especially on computer screens, causes some severe legibility problems. Due to the fact that a pixel is the smallest visual unit on a computer screen, it is possible that part of a character's outline for a given grid alignment could only fit in a fraction of a pixel. By a monochrome screen a character shap has a bilevel representation, i.e. a pixel is either on (black) or off (white). In such a case deciding whether a pixel to be black

or white is very crucial. This decision is even more sensitive for the small font sizes (6-12 pts.), where few pixels are available to represent a character shape. A modification of the character's outline is often of great help to increase the characters's legibility. This modification is called *hinting* or *grid-fitting*. Hinting includes many mathematical instructions stored in a font file that allow the distortion of a character outline at a specific size. Consequently the original outline remains undistorted as distortion is an a case-by-case base, i.e. only in specific font sizes. Hersch et al. have developed techniques for the grid-fitting of character outlines to a given rasterization grid [62] [63] [64]. In [65] a model-based matching and hinting of fonts is described. This model uses a table of applicable hints for automatic hint generation that are added to character's outline description. Additionally, the authors report on first building higher-order character structural parts like serifs, stems, bowls and junctions and second mapping them into the outline descriptors to design more advanced fonts. In [66] the authors introduce an universal auto-hinting system for typographic shapes as the existing hinting systems have been often conceived to match particular families of fonts such as Latin, Kanjai, Arabic etc.. The authors report of a method for the automatic recognition of character structure elements. They then use the gained knowledge about location of the stems and analysis of the outline parts between stems for producing automatically appropriate grid constraint rules (hints). Fig. 3.5 shows an unmodified outline of character **'M'** and its hinted version.

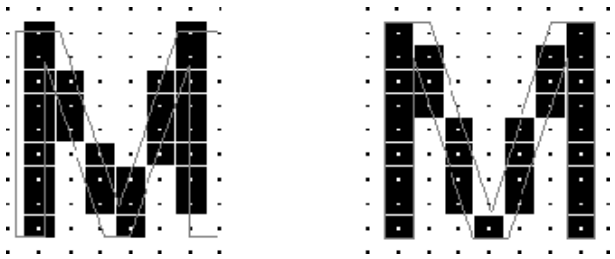
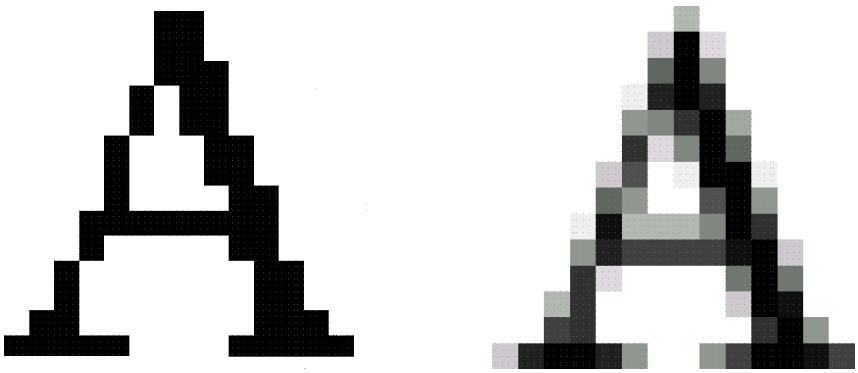


Figure 3.5: 'M' unhinted / hinted

Anti-aliasing

Using color screens has the advantage to use 255 gray scale colors. Anti-aliasing is a technique that uses these gray level colors in order to smooth the sharp edges and diagonals of a character by a bilevel (black and white) representation of a character shape. As a consequence of this, if the character includes a vertical line which should be one pixel wide but falls exactly between two pixels, it will appear on screen as a two-pixel-wide gray line. This blurriness is a trade off of clarity for accuracy. The text on the screen will appear sharper but wider than it is when using a printing device and looking at it on the paper. Fig. 3.6 illustrates bilevel versus anti-aliased representation of character **'A'**.



(a) **'A'** Bilevel (b) **'A'** Anti-alised

Figure 3.6: Character 'A' in bilevel and anti-aliased representation

Hinting of anti-aliased text

Bilevel characters are rendered on low-resolution displays by applying hinting techniques as described above. Ultra low-resolution anti-aliased characters are normally created from high-resolution instances by filtering and resampling. However, for anti-aliased character rendering, hinting techniques are also often applied in order to create sharp anti-aliased character images. Such an approach is for example presented in [67]. The proposed method relies on the set of visual rules that type designers have derived from many years of manual design. The authors call their method *perceptually tuned generation of anti-aliased fonts*, because it is based on the designer and reader's perception. Additionally, their method promotes accurate weight and phase control to ensure that the character elements have a minimum width even at a small size and without having shape variability in different grid alignments. Ultimately, it computes the most visually spacing of adjacent anti-aliased characters using a model based on the way the human vision system perceives space between characters. The authors claim that this spacing is a crucial part of their work because correct spacing leads to improved font quality. As a proof of this concept, they generate and space anti-aliased characters for a size of 12 points (see Fig. 3.7) and compare them with filtered but not hinted anti-aliased characters (see Fig. 3.8). The authors expect that their method can generally enhance the display capability of ultra low resolution computer screens.

Figure 3.7: A perceptually tuned anti-aliased text

We assume that such hinting techniques are incorporated into the Adobe Acrobat reader software. Fig. 3.9 shows a magnified screen-shot image of hinted anti-aliased text extracted from Adobe Acrobat reader.

Figure 3.8: A traditinally filtered anti-aliased text

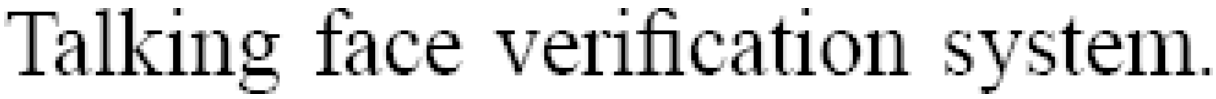
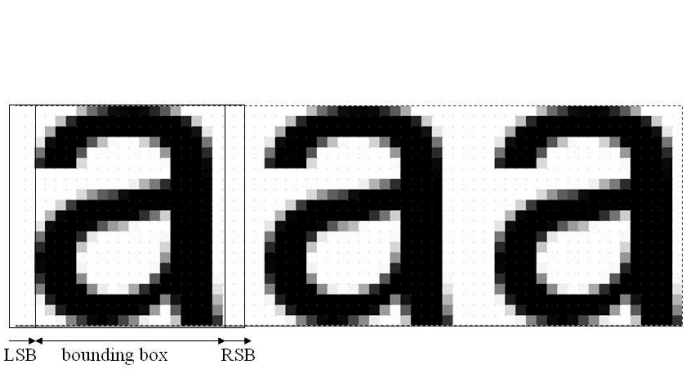


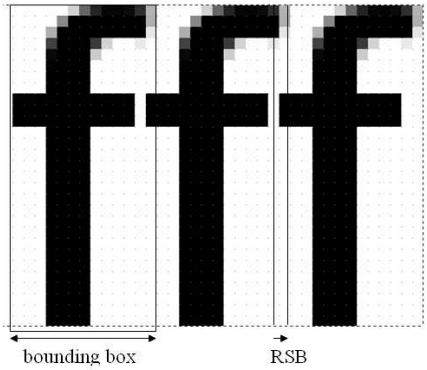
Figure 3.9: An enlarged screen-shot from Adobe Acrobat Reader

3.2.3 Font metrics

As stated earlier, each character from a particular font in digital typography is stored in a file containing its shape as a bitmap font or an outline font. This file contains additional information to adjust for example the proper character's interspacing between the precedent and following characters. These additional information is called font metrics. Font metrics components that are often used in digital typography are advance, bounding box (bbox), and left and right side bearings. The advance of a character is the distance from the character's origin to the origin of the next character. The bounding box of a character is the smallest rectangle that completely contains the visible portion of the character. The left-side bearing (LSB) is the distance from the character's origin to the left of its bounds rectangle. If LSB is negative, part of the character is drawn to the left of its origin. The right-side bearing (RSB) is the distance from the right side of the bounds rectangle to the next character's origin (the origin plus the advance). If RSB is negative, part of the character is drawn to the right of the next character's origin. Fig. 3.10 demonstrates these components for characters **'a'** and **'f'**.



(a) Metrics of **'a'**



(b) Metrics of **'f'**

Figure 3.10: (a) Font metrics for **'a'** with positive RSB (b) Font metric for **'f'** with negative RSB

3.2.4 Character spacing

As stated earlier, inter-character spacing is stored together with font shape in font file. There are two kinds of possible inter-character spacing:

- **Proportional spacing**
  Proportional-spaced fonts adjust the inter-character space based on the shape of the individual characters. The width of a character is varied based on its shape. Adjusting inter-character spacing is actually a function of kerning. Kerning is an algorithm that defines the space of specific character pairs rather than those characters alone. This method increases the legibility at computer screens. For instance, the letter 'A' and the letter 'V' are typically stored in each font as a kerning pair where they will be spaced differently when appearing next to each other. Kerning is a feature of many typesetting systems and computer programs, but will not be further discussed here. An example of a proportional-spaced font is **'Verdana'**. Fig. 3.11 shows a word written using this font at size of 40 points.

Figure 3.11: An example of a proportional font 'Verdana' for word 'Wrapper'

- **Monospace spacing**
  In this method the space between the characters does not vary. An example of such font is **'Courier'**. Fig. 3.12 shows an example of a word written using this font at size of 40 points. Virtually all typewriters of a few decades ago used Courier. More modern printing devices including some electronic typewriters are nowadays capable of producing proportional spaced fonts.

Figure 3.12: An example of a proportional font 'Courier' for word 'Wrapper'

3.3 Specificities of ULR text

As stated earlier, the aim of our work has beeb to develope a recognition approach for anti-aliased, ultra low resolution (ULR) text, i.e. text having a resolution between 72-100 dpi and small font sizes between 6-12 points. We described in the previous section the general techniques that are used to generate such text. In this section we describe specificities of this kind of text, i.e.the previously introduced techniques like hinting and anti-aliasing are employed. However,

the aim of this work has not been to develop a complete OCR system that is usually designed to analyze a full page containing paragraphs, phrases, linking signs etc. Thus we chose as our start point to study the specificities of such text a text line.

3.3.1 Segmentation of words in an ULR sentence

Figure 3.13: A text line containing ULR words in original size

Figure 3.14: An enlarged version of a text line containing ULR words

As previously described, wenn text is rendered at ultra low resolution, most of the font engines use anti-aliasing. As noted in last section, anti-aliasing is a useful technique when few pixels are available to make the sharp edges and diagonals of characters look smoother and more legible. Fig. 3.13 shows a sentence written with the font family *'Verdana'*, font style *roman (plain)* at 9 point size in its original size. Fig. 3.14 shows its enlarged version. As can be seen, the words in such a sentence usually have a distance of one or more white pixels from each other. This distance is suitable to apply well-known segmentation-first algorithms of classical document analysis to segment such a text line into words.

3.3.2 Segmentation of characters in an ULR word

Figure 3.15: Segmentation problem by ULR Words

Apparently, by looking at ULR words, we can observe that, especially for font sizes $\leq$ 9 points, the characters can not be segmented prior to recognition. This specificity is a challenging issue for recognition of such words. Because the majority of segmentation methods that are used in classical document analysis need a minimum distance of one pixel between the adjacent characters. Therefore, we can not use for example connected components analysis for segmenting the characters in context of a word that is rendered at ULR. Fig. 3.15 shows an enlarged version

of word *'brown'*. As can be seen the bounding boxes of characters can not be separated. One approach to segment characters could be to simultaneously segment and recognize characters. One could also think of using one of the few existing segmentation-first algorithms like the one introduced in the work of Wachenfeld et al. [51]. However, in our approach we decided to apply a statistical methods that can simultaneously segment and recognize such characters. We go further into details in chapters 4 and 6.

3.3.3 Variability due to Grid alignment

Figure 3.16: Character **'o'** in different grid alignments

Most of the image processing applications, utilized by web designers, like Adobe Photoshop, Macromedia Fireworks etc. use anti-aliasing to render ULR-text. However, this technique has the effect that the images of the same character are not identical. That is, the anti-aliased characters have different gray values dependent to the spatial grid alignment. Fig. 3.16 demonstrates such variability for character *'o'* in different words like *'brown'*, *'fox'* and *'over'* that are placed in different grid alignments due to their occurrence in the text line shown in fig. 3.14.

3.3.4 Adjacent characters

As seen in Fig. 3.10, each character in a low resolution screen is rendered within a bounding box. We can define three possibilities regarding the mutual influences of bounding boxes of adjacent characters in a word rendered at ultra low resolution as below:

- *The bounding boxes are separated, i.e. they have a distance of at least one pixel.* This is usually the case when characters are rendered with a font size > 10 points. Fig. 3.17 demonstrates this case for the characters *'w'* and *'n'* in the word *'brown'* rendered using font family *'Verdana'*, *roman (plain)* style at point size 13.

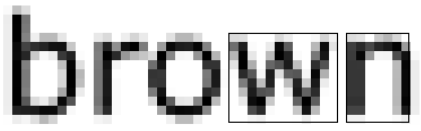


Figure 3.17: Characters *'w'* and *'n'* with separate bounding boxes

- *The bounding boxes are touching, i.e. there is no distance between them.* This is usually the case when characters are rendered with a font size between 8-10 points. Fig. 3.18 shows this case for characters *'b'* and *'r'* in word *'brown'* using font family *'Verdana'*, *roman (plain)* style at point size 10.

Figure 3.18: Characters *'b'* and *'r'* with touching bounding boxes

- *The bounding boxes are overlapping, i.e. one or more pixels of both bounding boxes merge together at their left and right borders.*

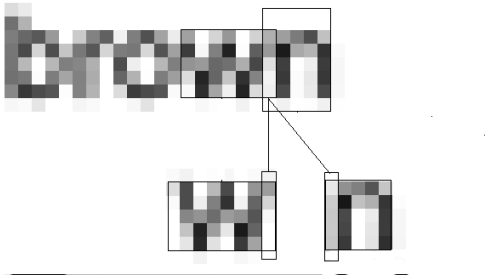


Figure 3.19: Characters *'w'* and *'n'* with merged bounding boxes

This happens mostly when characters are rendered with font sizes < 10 points. Fig. 3.19 shows the word *'brown'* rendered with font family *'Verdana'*, *roman (plain)* style at point size 9. As can be seen the rightmost pixel column of bounding box of *'w'* is merged with the leftmost pixel column of bounding box of *'n'*.

3.4 A simulation method to gain ULR database

3.4.1 ULR characters

At the time of our investigations no standard database of ULR characters was publicly available. Therefore we decided to build our own database for training and test. According to previous section, the ULR characters have to be of a resolution between 72-100 dpi, anti-aliased and multi-aligned with small font sizes (between 6-12 pts). To generate representative samples, we produced binary images of character classes (52 classes, lower- and upper-case letters) with a resolution k times larger and down-sampled them by the same factor using different image processing tools. The shape variability is achieved by varying the precise portion of the binary shape relatively to the re-sampling grid. This technique allows to produce k^2 different samples.

Fig. 3.20 illustrates this method with a re-sampling factor of $k = 5$. The binary images of character *'w'* at big font size (= 45 points) have the same shape as can be seen at the left hand side of fig. 3.20. Therefore this technique allows us to obtain $k^2 = 5^2 = 25$ different shapes as can be seen at right hand side of Fig. 3.20.

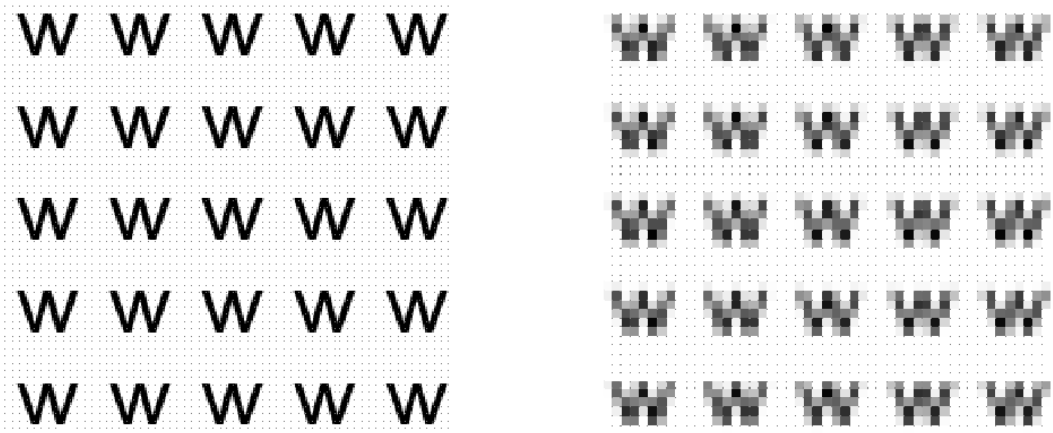


Figure 3.20: Character 'w' at high resolution (left side) and ultra low resolution (right side)

Figure 3.21: Example of binary images of word *brown* (45 pts) in different grid alignments

Figure 3.22: Example of down-sampled images of word *brown* (9 pts) in different grid alignments

3.4.2 ULR words

The same method can be applied to gain ULR words with different shapes. We can produce binary images of such words that have the same shape in big font size as shown in Fig. 3.21. When the image dimension is re-sampled according to down-sampling factor k, we gain words with different shapes by placing them in different coordinates of sampling grid as illustrated in Fig. 3.22.

Chapter 4

Algorithmic fundamentals

Machines can certainly beat humans in repetitive tasks like calculation of mathematical instructions or brute force search algorithms. Perhaps the most famous example of machine's capability for brute force algorithms has been demonstrated by the world chess championship in 1997 [68] when a machine named 'deep blue' has outperformed the Russian Garry Kasparov, the reigning world chess champion. However, the issue of building machines in order to recognize patterns is still the most challenging field in computer science. Machines have usually limited ability to perceive their environment. On the other hand, humans have a highly developed perception that happens below the conscious level of humans and obviously the complex mechanism of such a perception has been a matter of investigation in the past 40 years. In the recent years, researchers have made significant improvements to develop algorithms and theories to make machines recognize patterns. Consequently we now have machines that are able to recognize patterns better than humans in a specific problem area like face [69] and character recognition [70] and other recognition tasks. Machines can be connected to very performant sensors that sometimes beat the ability of humans. However, the most important difference between machines' and humans' perception is that human can mix easily different modalities and higher level of contextual information from one recognition task to another while machines lack this ability and can be used only for a limited and well-defined recognition task .

Although pattern recognition is therefore dependent of special characteristics of the problem, there exists a commonly used abstract classification model for pattern recognition. This model has three components:

- *a transducer* that prepares the input data for machine processing,
- *a feature extractor* that gathers relevant information from the input data
- *a classifier* that uses and compares the information given by the feature extractor to assign the input data to one of the finite possible classes for that specific problem.

In this chapter, we describe the fundamentals of these components that we have chosen and used for our recognition system. The first component of our recognition system, i.e. the transducer is

the set of image preparation tools that one put together to cope with the limitations of computer screen and specially with its limited resolution of 72-96 dpi. Other physical characteristics of the computer screen are out of scope. Therefore we give in the following the fundamentals of the selected feature extractor and the used classification algorithms.

4.1 Feature extraction

The borders between the feature extraction and classification task are arbitrary. The perfect feature extractor has the capacity to deliver such a good representation that a simple classifier is able to categorize the classes. On the other hand, a powerful classifier does not necessarily need the use of an ideal feature extractor. However, feature extraction is related always to the specific problem domain. Generally speaking, features have to be discriminative enough and invariant to the irrelevant transformations of the input data. The transformations of the input data is dependent from the problem domain. Generally speaking, there exist three kinds of transformations: a) horizontal and vertical translation b) rotation c) scale. Some other transformations can happen in other problem domains like isomorphic transformation when taking a picture from digital cameras in a 3D-scene, or artificial horizontal spacing that are invented by the web designer to make text in web images more legible and dominant.

One can find in the related works of classical document analysis a wide variety of suggested features. A complete list of these features is out of the scope of this work. However, we introduce here some frequently used features in the classical document image analysis.

- **Geometrical features**
  Geometrical features are features like perimeter, area, maximum and minimum distances from a boundary to the center of mass, number of holes, Euler number, compactness, etc.. However, these features are commonly used for the pre-classification of objects into characters or graphics. More details about these features can be found in [71].

- **Aspect ratios**
  These features are extracted from shape sizes in x- and y- directions. They are computed from building the ratio of width and height of the bounding box of a particular image. They can significantly contribute to the final classification of characters as can be seen in [72]. They are generally less used for word recognition tasks.

- **Horizontal and vertical projection profiles of black pixels**
  Projection is defined as an operation that maps mostly a binary image into a one-dimensional array called a histogram or projection profile. The values of the histogram are the sums of the black pixels along a particular direction. Two types of histograms are defined. They are at 0-degrees (horizontal projection histogram) and 90-degrees (vertical projection histogram) with respect to the horizontal axis y. A horizontal projection histogram h(x) of a

binary image (x,y) is the sum of black pixels projected onto the vertical axis x. A vertical projection histogram v(y) of a binary image (x,y) is the sum of black pixels projected onto the horizontal axis y. These features are very efficient in the classical document image analysis and therefore have been used in most of OCR engines for for segmentation and classification of characters of text in images scanned from printed material like books, newspapers or a sheet of paper. In addition, they have been used as reliable features for a font recognition system [73] that has been developed in our group. We assume that the reason for such a good reliability is that printed text has enough pixels available for characters that are mostly bi-level. In our case, where few pixels are available and the pixels representing anti-aliasing filters are applied to character shapes that let their pixel have 255 different gray values, we came to the conclusion that we better find other features that would be able to discriminate such character shapes.

- **Gabor filters**
  The Gabor Filters have received considerable attention because the characteristics of certain cells in the visual cortex of some mammals can be approximated by these filters. In addition these filters have been shown to possess optimal localization properties in both spatial and frequency domain and thus are well suited for texture segmentation problems [74] and [75]. Gabor filters have been used in many applications, such as texture segmentation, target detection, fractal dimension management, edge detection, retina identification, image coding and image representation. Gabor filters have been also sporadically used for document analysis. However, since such filters are rather general, we have decided to concentrate on features which could be more specific and therefore better suited for our purpose.

- **Fourier descriptors**
  Fourier descriptors are frequently used in classical document analysis mostly in the preprocessing step, as they can be made translation, scale and rotation invariant [76] and [77]. Therefore the contour of a known image can be recognized irrespectively of its size, position and orientation. Fourier descriptors use only the boundary of the object and can be used to capture the gross essence of a boundary and though are very useful for the shape analysis. These features have the disadvantage that they can not deal with disjoint shapes where single closed boundary may not be available.

- **Hough Transform**
  Hough transform are used mostly to identify positions of arbitrary shapes, most commonly circles or ellipses and even non-parametric curves [71], [78] and [79]. The Hough transform is a technique which can be used to isolate features of a particular shape within an image. The main advantage of the Hough transform technique is that it is tolerant of gaps in feature boundary descriptions and is relatively unaffected by image noise. This technique is particularly useful for computing a global description of a feature, given (possibly noisy)

local measurements. The motivating idea behind the Hough technique for line detection is that each input measurement (e.g. coordinate point) indicates its contribution to a globally consistent solution (e.g. the physical line which gave rise to that image point).

- **Zernike moments**
  The Zernike polynomials were first proposed in 1934 by Zernike [80]. They are a set of complex polynomials which form a complete orthogonal set over the unit circle $x^2+y^2=1$ in polar coordinates (r,θ). Their moment formulation is noise resilient and does not obtain information redundancy. Zernike moments are used mostly for image reconstruction purposes. The disadvantages of Zernike moments are that they are only rotation invariant, i.e. the image has to be first normalized in order to be scale and transformation invariant.

- **Central moments**
  Central moments have the substantial advantage that they do not just use the boundary information of a shape but also all the pixels of it. It can also be applied to gray scale images and is insensitive to the distribution of the gray values in that shape. Therefore, they convey the area information of the shape, which is a valuable feature extraction method, when the shape contains only few pixels.

  Generally speaking, given a 2D input data $f(x,y)$, a set of moment features based is defined via

$$m_{pq} = \sum_x \sum_y x^p y^q f(x,y) \qquad p,q = 0,1,2,.. \tag{4.1}$$

  where $f(x,y)$ are gray values of a pixel at x and y coordinates.

  The central moments are related to the center of mass or center of gravity. The coordinates of center of the mass are $\hat{x}$ and $\hat{y}$ and are defined via $\hat{x} = \frac{m_{10}}{m_{00}}$ and $\hat{y} = \frac{m_{01}}{m_{00}}$

  Therefore, the central moments can be defined via

$$\mu_{pq} = \sum_x \sum_y (x-\hat{x})^p (y-\hat{y})^q f(x,y) \qquad p,q = 0,1,2,.. \tag{4.2}$$

  For detailed description of central moments see also [77], [76] and [81].

4.2 Classification methods

Methods that are used in pattern classification can be divided into three groups: syntactic and structural and statistical pattern recognition [82]. Below we give a short description of these three groups. The first two groups can be linked together.

- **Syntactic and structural pattern recognition**
  Syntactic pattern recognition take the view that a pattern is composed of simpler subpatterns that can also be built from even smaller subpatterns. Application areas for syntactical pattern recognition can be natural language processing, line drawing analysis and 3-D 'blocks-word' description. The structural pattern recognition uses symbolic data structures, such as strings or trees and graphs for the representation of individual patterns. In this case, the recognition problem turns into a pattern-matching problem. These approaches can be used in applications like music notes interpretation or maps interpretation that are made of strong hierarchical structures. It is worth asking if this pattern recognition approach is still relevant. On the one hand one can admit that the pattern representation capability of tractable formal grammars is limited, mostly because of parsing difficulties. On the other hand, the syntactical Pattern recognition using representations such as trees and attributed graphs are gaining popularity in certain areas. However, some recent studies show that combining the syntactical and statistical pattern recognition delivers reliable pattern representation capabilities [83].

- **Statistical pattern recognition**
  Statistical pattern recognition is based on statistical characterisations of patterns, assuming that the patterns are generated by a probabilistic system. The generation in itself can be deterministic but with a deteroriation process which is itself stochastic that present characteristics that are similar to a stochastic process. Statistical pattern recognition are widely used in the classical character recognition tasks with significant results. Therefore, we decided to employ them for our approach. We have basically used two statistical methods which we describe below.

4.2.1 Fundamentals of Bayesian decision theory

Bayes decision theory is a statistical approach that is used successfully since decades for many applications of pattern recognition [84]. This theory makes the assumption that the decision problem is posed in probabilistic terms and that all of the relevant probability values are known. Let $\omega_1, ..., \omega_c$ be the finite set of c classes and feature vector x be a d-component vector. The distribution of x conditional on ω_j being the true class is called class-conditional probability density function $p(x|\omega_j)$ or also the *likelihood*. If we suppose that we have some prior knowledge of how likely we are to get the class ω_j and we call this prior knowledge the *a priori* probability $P(\omega_j)$, then the joint probability that a pattern is in category ω_j and has the feature vector x can be written as follows:

$$p(\omega_j, x) = P(\omega_j|x)p(x) = p(x|\omega_j)P(\omega_j). \tag{4.3}$$

By rearranging this we get the so called *Bayes formula*:

$$P(\omega_j|x) = \frac{p(x|\omega_j)P(\omega_j)}{p(x)} \tag{4.4}$$

where

$$p(x) = \sum_{j=1}^{c} p(x|\omega_j)P(\omega_j) \tag{4.5}$$

$$P(\omega_j|x) = \frac{p(x|\omega_j)P(\omega_j)}{\sum_{j=1}^{c} p(x|\omega_j)P(\omega_j)} \tag{4.6}$$

Bayes formula can be expressed in informal English like:

$$posterior = \frac{likelihood \times prior}{evidence} \tag{4.7}$$

For the sake of simplicity, we can further assume that all of our classes are equiprobable, which means that they all have the same *prior* probability. In this case our formula can be simplified as follows:

$$P(\omega_j|x) = \frac{p(x|\omega_j)P(\omega_j)}{(\sum_{j=1}^{c} p(x|\omega_j))P(\omega_j)} \tag{4.8}$$

Further, the sum of all the probabilities of all classes is a constant and does not influence the decision which class is more probable to represent the unknown pattern. Therefore, the structure of a Bayes classifier is primarily determined by the conditional density or likelihood of ω_j with respect to feature vector x, i.e. $p(x|\omega_j)$. The density function that has received more attention, is the *multivariate normal or Gaussian density.* The reason of such attractivity could be the analytical tractability of *Gaussian density.* Besides Gaussian densities are corresponding to many natural stochastic phenomena. Therefore, we use this function as the state-conditional probability function in our case. Supposing to have a feature vector with d components, the general multivariate normal density can be written as:

$$p(x|\omega_j) = \frac{1}{(2\pi)^{\frac{d}{2}}|\Sigma|^{\frac{1}{2}}} exp[-\frac{1}{2}(x-\mu)^t\Sigma^{-1}(x-\mu)] \tag{4.9}$$

Where x is a d x 1 feature vector and μ is a d x 1 mean vector and Σ is a d x d covariance matrix. These values have been obtained during the training process and are known values. The system we have designed will then calculate feature vectors from the unseen image of the isolated characters and will then calculate all the conditional state probability functions for all classes ω_j. The class with the maximum probability value will be the winner class and the unseen image will then be assigned to this class. The maximum probability value is also called *maximal likelihood.*

As maximal likelihood values can become very small and computationally less tractable, we have decided to work on the log-domain that delivers sum values rather than product and division values and is though of more practical use avoiding very small values in the dominator

of a fraction leading the problem to be numerically unsolvable. The maximal likelihood in the log domain is:

$$g(x|\omega_j) = -\frac{1}{2}(x-\mu)^t\Sigma^{-1}(x-\mu) - \frac{d}{2}log2\pi - \frac{1}{2}log|\Sigma| \tag{4.10}$$

4.3 Fundamentals of hidden Markov models

Bayes desicion theory is well suited when making a single decision like recognizing single characters in our case. We may have to make a sequence of decisions while the sequences are influenced directly by their proceeding sequences. Therefore, we could be interested in modeling sequences. Such sequences are usually representing a temporal process (like speech or stock values). They can also be representing a spatial process like in handwriting, DNA or screen rendered characters connected by anti-aliasing process like in this work. Hidden Markov models can be used for obtaining such sequence of decisions as a powerful statistical model. Hidden Markov models are called after the Russian mathematician Andrei Andrejewitsch Markov that lived from 1856 till 1922 in Russia. The complete theory has been firstly introduced some decades later by Baum and Welsch in 1960. However, this was written by mathematician and was rather difficult to implement and understand. Later in 1970s Viterbi, a telecommunication engineer graduated from MIT and USC, has developed a decoding algorithm based on dynamic programming for the signal processors. Viterbi algorithm was a break through for the theory of hidden Markov models and made it usable for a wide variety of practical applications in wireless communications, automatic speech processing, bioinformatics and for some specific areas of document analysis like cursive handwriting recognition.

Basically, a hidden Markov model (HMM) is a statistical model in which the system being modeled is assumed to be a Markov process with unknown parameters, and the challenge is to determine the hidden parameters from the observable parameters. Therefore, in a hidden Markov model, the state is not directly visible, but variables influenced by the state are visible. Hence the name hidden Markov models. Transitions among the states are governed by a set of probabilities called transition probabilities. In a particular state an outcome or observation can be generated, according to the associated probability distribution.

4.3.1 Elements of hidden Markov models

In the following we give an overview about the elements of the HMMs. The complete theory is explained in [85] and [86]. We have made the assumption in our model that the next state is dependent only upon the current state, such model is called *first order hidden Markov model*. First order HMMs are also much easier to implement than the higher order HMMs. We used this model, as for our case, the characters in a single word influence only their adjacent characters. We call this influence *contextual noise*. Such noise does not spread itself, according to our observations, to some other characters after or before a specific character.

In order to define a HMM completely, the following elements are needed.

- *The number of states of the model, N.* The states are denoted as $S = \{S_1, S_2, ..., S_N\}$ and the state at time t as q_t. Generally the states are interconnected in such a way that any state can be reached from any other state, as it is the case for the *ergodic model*; however many other interconnections between states are also of interest. The interconnections define the topology of the HMM.

- *The number of distinct observation symbols per state, M.* We denote the individual symbols as $O = \{o_1, o_2, ..., o_M\}$

- *The state transition probabilities* $A = \{a_{ij}\}$.

$$a_{ij} = p\{q_{t+1} = j | q_t = i\}, \quad 1 \leq i, j \leq N, \tag{4.11}$$

  where q_t denotes the current state. For the special case where any state can reach any other state in a single step, $a_{ij} \geq 0$ for all i,j. For other types of HMMs, we could have $a_{ij} = 0$ for one or more (i,j) pairs.

  Transition probabilities have to satisfy the normal stochastic constraints,

$$\sum_{j=1}^{N} a_{ij} = 1, \quad 1 \leq i \leq N \tag{4.12}$$

- *An output (emission) probability in each of the states,* $B = \{b_j(k)\}$.

  There are two general types of HMM split according to the form of their output (emission) distribution. If the output (emission) distribution is based on discrete elements then the models are called discrete HMMs. Alternatively if the output (emission) distribution is continuous, i.e. if the feature vectors are of any real numbers as is in our case, they are referred to as continuous HMMs. We have used Gaussian distribution, such as multivariate mono-Gaussian distribution (see previous section) or Gaussian mixture models (GMMs). GMMs are used, when the distribution tends to have more than one local maximums. GMMs are introduced later in this chapter. For the GMMs, the emission probability of being in each state can be written as:

$$b_j(k) = \sum_{m=1}^{M} c_m N(o_m; \mu_m, \Sigma_m) \tag{4.13}$$

- The initial state distribution, $\pi = \{\pi_i\}$. where

$$\pi_i = p\{q_1 = i\}, 1 \leq i \leq N \tag{4.14}$$

Therefore a complete specification of a HMM requires specification of two model parameters (N, M), specification of observation symbols and the specification of three probability measures A, B, π. For the sake of simplicity, we can use the compact notation

$$\lambda = (A, B, \Pi) \tag{4.15}$$

Fig. 4.1 illustrates an example of a HMM model for 3 states and 4 possible observations with the corresponding transition and emission probabilities [87].

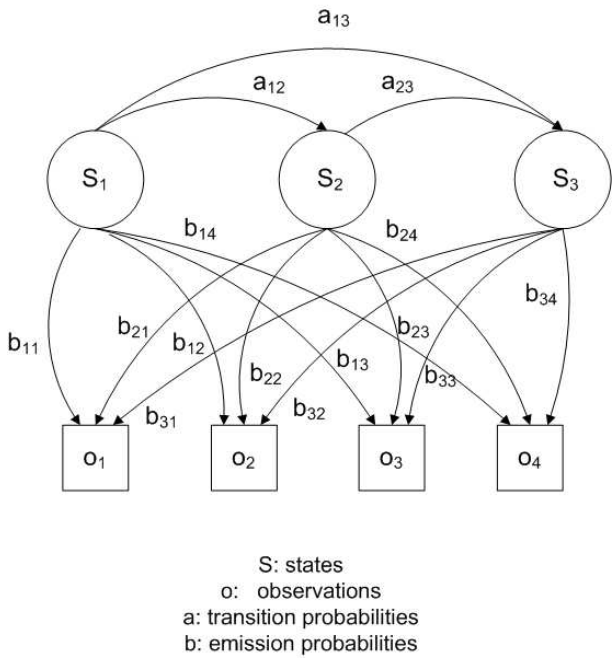


Figure 4.1: An example of a HMM model

4.3.2 Topologies

The possible topologies of HMMs are:

- **Ergodic or fully connected models**
  In such models every state could be reached from every other state. Fig. 4.2 demonstares a 4 state ergodic model.

- **Left-right or bakis model**
  Such models assume that states proceed from left to right. Fig. 4.3 shows a 4-state left-right model. In this model the states can skip to the two further states to the right. One could think of many other different types of skips. However, the topology is very much dependent to the nature of the problem domain. We do not make use of skip between

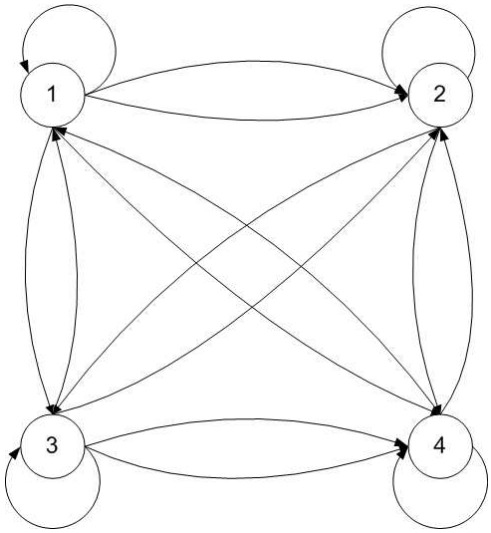


Figure 4.2: A 4 ergodic HMM-topology

characters in our recognition approach. We rather assume that states are interconnected either to themselves or to their proceeding state like for ex. in fig. 4.4.

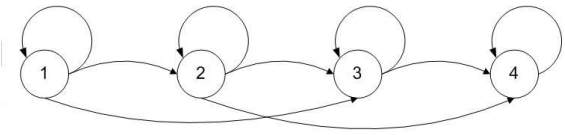


Figure 4.3: A 4 state second order left-right HMM-topology

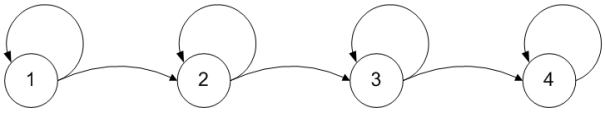


Figure 4.4: A 4 state first order left-right HMM-topology

- **Modified left-right model**
  The left-right model shown above is the simplest topology one can think of. This model, which we call in our work '*simple left-right* is without incorporation of a-priori knowledge about the behavior of states, i.e. knowledge about minimum and maximum number of times a state can repeat. This is an advantage as the system is a general and not knowledge based, but on the other hand the recognition accuracies can be lower than systems including knowledge. Therefore, we have evaluated some modifications of left-right system that we describe below:
  - **Left right with minimum width constraint**

This topology is a modified left right topology that considers minimum width constraints of the characters. When we assume that the states are characters in our case, with this topology we oblige the Viterbi to spend a minimum amount of time in the same state. This is meaningful because some characters like 'w' and 'm' are wider than the others. This topology has the disavantange that we have to use some a-priori-knowledge like font metric information. This implies that we first must be aware of the font family of the words prior to its recognition. On the other hand, as it is always the case by machine learning, we gain more recognition accuracy. Fig. 6.17 shows such a topology modeling a single character.

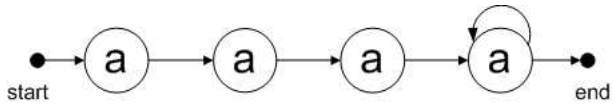


Figure 4.5: A character topology based on minimum width constraint

– **Left-right with minimum and maximum width constraint**
In this topology, we go even further than minimum width constraints and assume additionally a maximum width constraint for each character. This means that a single character has to be visited by the Viterbi algorithm a minimum and a maximum amount of observation sequences. This topology is knowledge-based. We will show in chapter 6 that we can infer this knowledge automatically during HMM-training. This makes such a topology independent of an a priori font knowledge. Fig. 4.6 shows this topology used to model a single character.

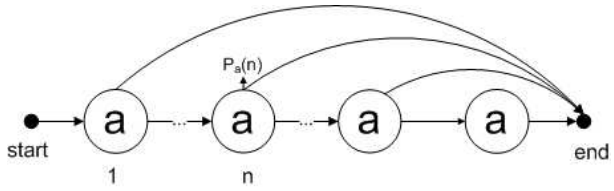


Figure 4.6: A character topology based on minimum and maximum width constraint

4.3.3 Gaussian mixture models to model emission probability

The unimodal multivariate Gaussian distribution that has been previously discussed in this chapter, has some significant limitations when it comes to model a data set with a distribution that has more than one local maximum. In such cases a linear superposition of multiple Gaussians gives a better characterization of the data set. Such a linear superposition is named in the literature as mixture distributions. If we assume that the distribution is a Gaussian normal distribution, then we can call it a Gaussian mixture model (GMM). Data sets with more than one local maximas can therefore more accurately be modeled by using a sufficient number of

Gaussian distributions each representing with their local means, weights and variances. The superposition of K Gaussian densities can be written as

$$b_{q_j}(o_j) = \sum_{k=1}^{K} \pi_k N(o_j|\mu_k, \Sigma_k) 1 \leq j \leq T \tag{4.16}$$

and is called a mixture of *Gaussians* for a given observation being. Each Gaussian density $N(x|\mu_k, \Sigma_k)$ is called a component of the mixture having its own μ_k and Σ_k. Therefore the form of the Gaussian mixture distribution is determined by the parameters $\pi = \{\pi_1, \pi_2, ..., \pi_K\}$, $\mu = \{\mu_1, \mu_2, ..., \mu_K\}$ and $\Sigma = \{\Sigma_1, \Sigma_2, ..., \Sigma_K\}$. We could call the above equation also the likelihood to have vector $X = \{x_1, x_2, ..., x_T\}$ as data set.

For small values we better consider the likelihood in the log domain given by

$$lnb(O|\pi, \mu, \Sigma) = \sum_{j=1}^{T} ln\{\sum_{k=1}^{K} \pi_k N(x_j|\mu_k, \Sigma_k)\} \tag{4.17}$$

We have therefore a more complex equation than a single Gaussian, due to the asummption over k inside of the logarithm. To find a mixmal likelihood for this equation, we no longer can consider a closed-form analytical solution. An elegant and powerful method to find the maximum likelihood for GMMs is the *expectation maximization (EM)* framework. This method is discussed in the next section.

4.3.4 Expectation maximization(EM) for estimation of GMMs

Suppose that we have a set $\mathcal{D}$ with N independent samples $o_1...o_T$ according to the probability as $b(O|q_j)$. Suppose that we assume that $b(O|q_j)$ is a normal density with mean μ_j and covariance matrix Σ_j. We can assume that a parameter verctor θ_j consists of the components of μ_j and Σ_j. To show the dependence of $p(o|q_j)$ on θ_j, we write $p(o|q_j)$ as $p(o|q_j, \theta_j)$. The problem is now to use the information provided by the training samples to estimate the unknown parameters for the vector θ. We define the likelihood of θ with respect to the set of samples $o_1...o_T$ as $p(\mathcal{D}|\theta)$ written

$$p(\mathcal{D}|\theta) = \prod_{k=1}^{T} p(x_k|\theta) \tag{4.18}$$

We can finally define $l(\theta)$ as the log-likelihood function as follows:

$$l(\theta) \equiv lnp(\mathcal{D}|\theta) \tag{4.19}$$

By writing a formal equation as the argument θ that maximizes the log-likelihood, that is,

$$\theta = \arg\ \max_{\theta} l(\theta) \tag{4.20}$$

The basic idea of the EM algorithm is to iteratively estimate the likelihood given the data that is present. That is to find a set of model parameters θ that maximizes the log-likelihood $l(\theta)$. The search of best model starts with some initial value of the model parameters , θ^0. Then in the Mth step, the optimal θ^M is found.

In our implementation, to better model the data and to find out the optimal distribution of different Gaussians, we use a splitting technique, i.e. we define a *splitting sequence set* like split sequence $= \{1, 2, 4, ..., 64\}$ that defines the constituent splitting steps till the target number of Gaussian components. Therefore, in our model the initial values come from a one-clump Gaussian distribution. The next step consists of both expectation and maximization processes. In the expectation step, we begin to split according to the ascending order of the splitting sequence set. To better model the data, we split the Gaussians with the highest weight into two Gaussians and calculate the new means, covariance matrices and weights of the new multi Gaussian distribution. In the maximization step we maximize the log likelihood of the previously calculated distribution until a stopping criteria is reached. This two processes are repeated until the target number of Gaussian components is reached. We could write it in pseudo-code like this:

- `split number = 1`
- `calculate the initial Gaussian values for split number=1`
- `set the initial weight = 1`
- `repeat`
  - `take the next splitting number`
    - `if split number=2`
      - `then split the mono-Gaussian into 2 Gaussians`
      - `calculate the new weight, Gaussian parameters for each Gaussian`
      - `perform expectation`
      - `perform maximization`
    - `else`
      - `take the Gaussian with the highest weight`
      - `split this Gaussian into 2 Gaussians and calculate weight and Gaussian parameters for each Gaussian`
      - `perform expectation`
      - `perform maximization`
- `until split number = target number of Gaussian components`

4.3.5 The evaluation problem of the HMMs

We can look at the evaluation problem as one of scoring how well a given model matches a given observation sequence. For example when we want to choose among several competing models, the model which best matches the observations. In other words, we are likely to calculate the probability of the observation sequence $o_1, ..., o_T$ given the model λ, i.e. $P(O|\lambda)$. Let us consider every possible state sequence of length T(the number of observations)

$$Q = q_1 q_2 ... q_N \tag{4.21}$$

Thus the probability of the observation sequence O for the specific state sequence Q is

$$P(O|Q, \lambda) = \prod_{t=1}^{T} P(o_t|q_t, \lambda) \tag{4.22}$$

Under the assumption that the observations are statistical independent, we get

$$P(O|Q, \lambda) = b_{q_1}(o_1)...b_{q_T}(o_T) \tag{4.23}$$

Besides, the probability of state sequence Q can be written as

$$P(Q|\lambda) = \pi_{q_1} a_{q_1 q_2} a_{q_2 q_3} ... a_{q_{T-1} q_T} \tag{4.24}$$

The joint probability of O and Q, i.e. the probability that O and Q occur simultaneously, can be calculated by the product of each probability, i.e.,

$$P(O, Q|\lambda) = P(O|Q, \lambda)p(Q, \lambda) \tag{4.25}$$

We can therefore get the probability of O, given the model λ by summing the joint probability over all possible state sequences Q:

$$P(O|\lambda) = \sum_{alleQ} P(O|Q, \lambda)p(Q, \lambda) = \sum_{q_1 q_2 ... q_T} \pi_{q_1} b_{q_1}(o_1) a_{q_1 q_2} b_{q_2}(o_2) .... a_{q_{T-1} q_T} b_{q_T}(o_T) \tag{4.26}$$

This equation is computationally not feasible, as the number of multiplications and additions even for small values are proportional to N^T, e.g. for $N = 3$ (states) and $T = 70$ (observations), we have 3^{70} computations, which is rather inefficient. Thus the number of terms in the summation grows exponentially with the length of the chain. Therefore different methods have been developed to make the evaluation task computationally more cost efficient. The most famous methods are Baum-Welch-algorithm known as forward-backward algorithm and Viterbi algorithm. The most significant difference between Baum-Welch algorithm and Viterbi is that Viterbi allows additionally a backtracking giving the best state Q^* by computing $P(O, Q^*|\lambda)$

that maximizes the value of Q^*.

We have chosen the Viterbi algorithm for our work because of its fast convergence comparing to the other methods. We give in the following a short and comprehensive description of this algorithm.

4.3.6 Viterbi algorithm: How to find the 'correct' state sequence?

In many applications of the HMMs it is often of interest to find the most probable sequence of hidden states for a given observation sequence. This can be solved efficiently using the Viterbi algorithm. As we already mentioned, the number of possible paths through the lattice grows exponentially with the length of the chain. The advantage of Viterbi algorithm is that it searches this space of paths efficiently to find the most probable path with a computational cost that grows only linearly with the length of the chain. To find the best state sequence $q = \{q_1 q_2 ... q_T\}$ for a given observation sequence $O = \{o_1 o_2 ... o_T\}$, we need to define the probability

$$\delta_t(i) = P\left[q_1 q_2 ... q_{t-1}, q_t = i, o_1 o_2 ... o_t | \lambda\right] \tag{4.27}$$

where $\delta_t(i)$ is the highest probability along a single path, at time t, for the first t observations and ends in state i. Therefore in order to obtain the distribution at time $t+1$, we have to maximize the product of $\delta_t(i)$ and the corresponding probability to be in state i at time $t+1$ as follows

$$\delta_{t+1}(j) = \left[max_i \delta_t(i) a_{ij}\right] . b_j(o_{t+1}) \tag{4.28}$$

In order to retrieve the most probable state sequence, we need to keep track of the argument that maximized the equation 4.28 for each t and j. We introduce the term $\psi_t(j)$ for as maximal probable state. The Viterbi algorithm is a recursive process defined as:

1. Initialization

$$\delta_1(t) = \pi_i b_i(o_1), \quad 1 \leq i \leq N \tag{4.29}$$

$$\psi_1(t) = 0 \tag{4.30}$$

2. Recursion

$$\delta_t(j) = max\left[\delta_{t-1}(i) a_i j\right] b_j(o_t), \quad 2 \leq t \leq T, \quad 1 \leq j \leq N \tag{4.31}$$

$$\Phi_t(j) = argmax_{1 \leq i \leq N}\left[\delta_{t-1}(i) a_i j\right] b_j(o_t), \quad 2 \leq t \leq T, \quad 1 \leq j \leq N \tag{4.32}$$

3. Termination

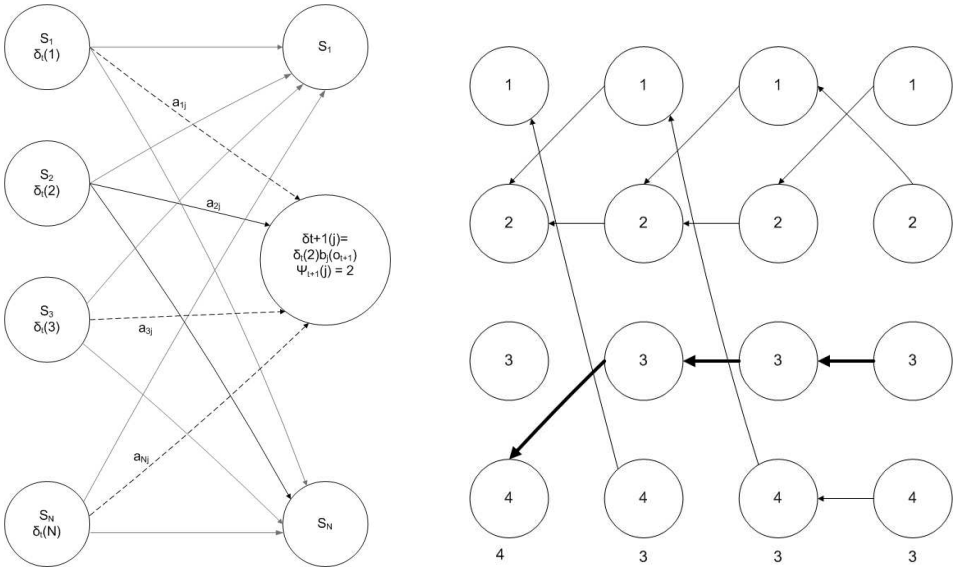
$$P^* = max_{1 \leq i \leq N} [\delta_t(i)] \tag{4.33}$$

$$q_T^* = argmax_{1 \leq i \leq N} [\delta_T(i)] \tag{4.34}$$

4. Path (state sequence) backtracking

$$q_t^* = \psi_{t+1}(q_{t+1}^*) \tag{4.35}$$

Fig. 4.7 demonstrates the recursion and backtracking step of the Viterbi algorithm.



(a) Recursion step (b) Backtracking step

Figure 4.7: (a) The recursion step of Viterbi (b) The backtracking step of Viterbi

Generally looking, Viterbi makes a dramatic computational cost saving as follows: Consider a particular time step n and a particular state k at that time step. There will be many possible paths on the corresponding node at the lattice diagram. If we only retain the particular path with the highest probability and keep track of the previous states. Suppose that the number of states would be K at time n, then at time $n+1$ we would have K^2 possible paths to consider, but we need only to obtain K of these corresponding to the best path for each state at time $n+1$. When reaching the final step N, we will discover which state corresponds to the overall most probable path. As there is an unique path coming into that state, we can trace the path back to N-1 to discover the state occupied at this time, and continue this back-tracking through the lattice to the state n=1. A detailed explanation of Viterbi algorithm can be found in [85].

4.3.7 N-best decoder: modified Viterbi algorithm

The previously described Viterbi algorithm results to the best path. It can likely happen that there are existing competing paths that have very close δ_{end}. Comparing the best path with the ground truth can deliver some mismatches. A reasonable solution to avoid such mismatches in the recognition process is to keep a certain amount of paths that are above a certain threshold till the last observation and have n several paths by backtracking. These competing paths can pass through a second discrimination process, that defines a certain criteria to be fulfilled. The second step could be performed either manually like for example comparing the labeled best paths and use human judgement or some lexical or grammatical constraints or language models for a more accurate text recognition. Fig. 4.8 shows a left-right HMM based on a closed dictionary consisting of 2 words: {cat, cut}, i.e. the system contains 6 states {c, a, u, t, o, m}.

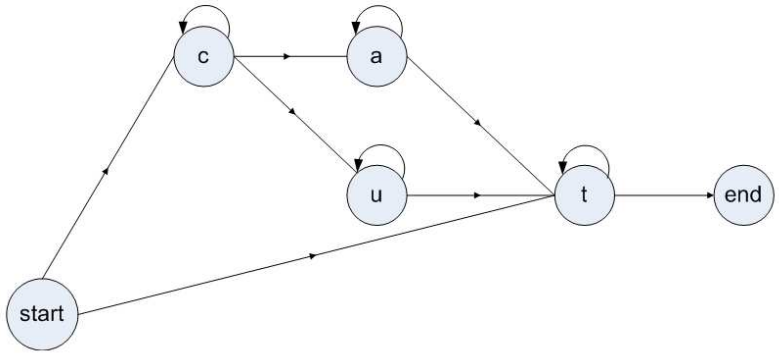


Figure 4.8: A two word dictionary left-right HMM

Fig. 4.9 shows the trellis diagram for this HMM when using modified Viterbi when the number of best paths are two. As we can see, two words: {cat,cut} are best candidates to represent the genuine word 'cat'.

4.3.8 Pruning

The Viterbi algorithm calculates in each step all possible δs for all states at time t, chooses the maximum of these δs at time t per state and finally takes the maximum of the competing δs as the best choice in that observation sequence. In case of large HMMs like for example parallel word-HMMs built from a large dictionary, there could be a vast number of possible transitions between the different states that brings the system at its memory and processor limits. For such cases a special algorithm called *pruning* is developed that takes a decision at such sequence to remove all the paths until now, when their δ is below a certain threshold. pruning has the advantage that it accelerates the computation time but on the other hand it could remove the genuine path as this could not be maximum at a certain time or observation sequence.

The pruning algorithm we use for our experiments consists of following values:

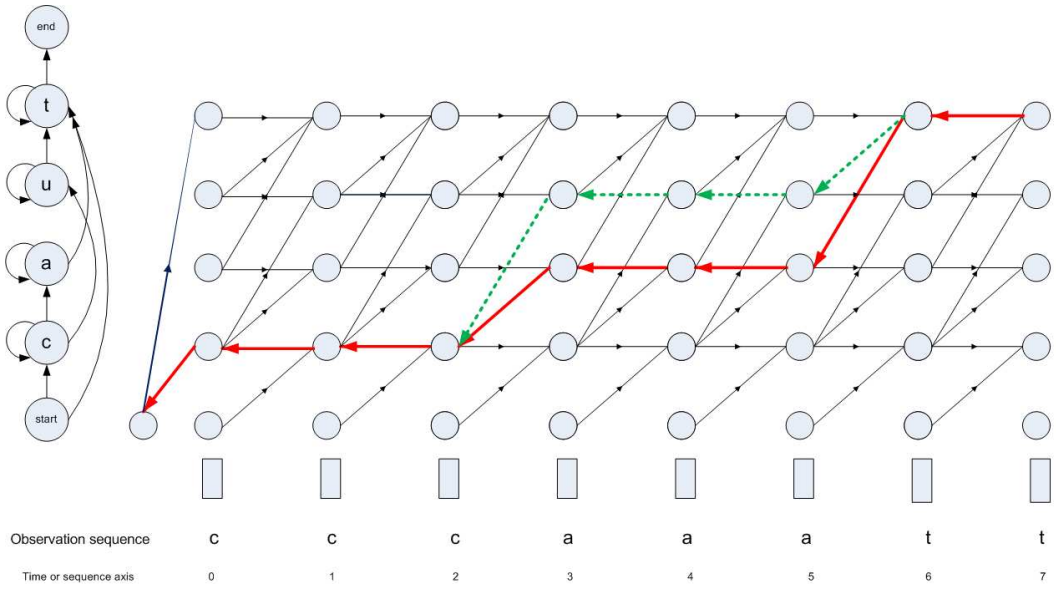


Figure 4.9: Modified Viterbi for 2 best paths

- **Pruning coefficient**: A positive constant value smaller than 1.
- **Pruning threshold**: A threshold with initial value smaller than 0.1.
- **Maximum number of paths**: Defines the maximum number of paths for each time step or observation sequence
- **N-best coefficient**: The maximum number of best paths

The **Pruning threshold** is therefore defined as:

$$\text{Pruning threshold} = \delta_{min} + (\delta_{max} - \delta_{min}) * \text{pruning coefficient} \tag{4.36}$$

4.3.9 Training with HMMs

One of the most powerful features of hidden Markov models is its ability to perform an automatic training. That is, it determines a reliable set of transition probabilities as a_{ij} and emission probabilities as b_{jk} from a set of training samples. There are different methods for applying the automatic HMM training. However, we are interested in our work to build a single word recognizer. Each word is made up of different characters for those we need to have a reliable set of transition and emission probabilities. We have chosen an iterative approach with the following steps:

- **Linear segmentation**
  This step has the goal of obtaining an initial segmentation of the words into characters. After this initial segmentation, we can calculate the initial transition and emission probabilities that we can use in our iteration process. As one can see from fig 4.10, a single

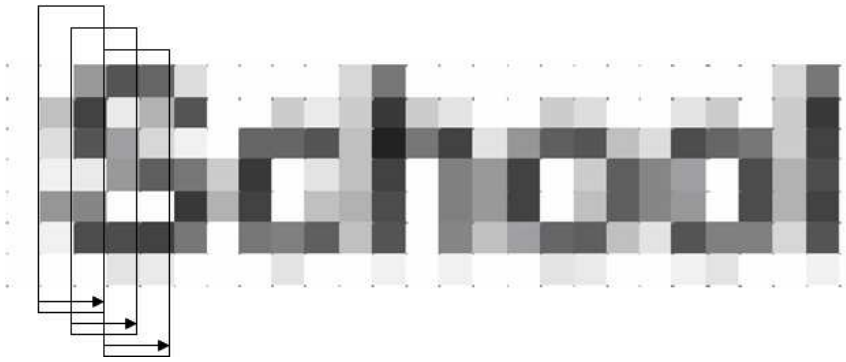


Figure 4.10: Sliding window technique for word 'school' at ultra low resolution and 9 point size

ULR word image gives no evidence about the segmentation of the characters. Therefore, we have decided to lay a sliding window that is 2 pixels wide and is shifted by one pixel to the right at the top of each word as shown in Fig. 4.10. We then compute the chosen features for each sliding window. The chosen features for our several studies have been central moments as will be explained in chapters 5 and 6. Conequently for a word of $width = w$ pixels, we obtain a set of feature vectors X containing $w_X = w - 1$ feature vectors, i.e. $X = \{X_1, X_2, ..., X_{w_X}\}$. The only infered constraint in this step is the ground truth information about number of characters in a specific word. By dividing the number of feature vectors by the number of the existing characters in a word, we can assign an equal number of feature vectors to each corresponding character. This step ends, when the feature vectors of the sliding windows of all words in trainning set have been claculated and assigned to each character class. Finally, the initail emission (output) and transition probabilities for each character will be calculated. Generally the transition probabilities are calculated as below:

$$a_{ij} = \frac{\#\quad of\quad transitions\quad i \rightarrow j}{total\quad \#\quad of\quad transitions\quad i} \tag{4.37}$$

where i,j are in our case the Latin letters

- **Iterative reestimation**
  In this step, we first determine the new emission probabilities based on the EM algorithm assuming a mixture of Gaussian distribution (GMM). The number of mixture components is the subject of investigation at training time. Next we define the character interconnections or with the other words the word topology according to our choice and let the Viterbi algorithm label the correct character sequences. This labeling of the observation sequences allows us to assign a more accurate set of feature vectors to each character class. Additionally, the knowlege about the assigned number of feature vectors to each character, let us to estimate new transition probabilities according to the formula 4.37. Finally we calculate the total probability of the previously labeled that were obtained with the

Viterbi algorithm.

- **Termination**
  The reestimation step is finished when the difference of the total probabilities of the previous and the present iteration is below a give threshold, i.e. we have obtained a satisfied convergence, or the number of iteration steps is bigger than a certain amount. By reaching the stopping criteria, we obtain the following parameters of our implemented hidden Markov model:

  1. Optimized transition probabilities for all characters
  2. Optimized emission probabilities for all characters
  3. The correct number of observation sequences for each character. This number can be also named as character's alignment. This number could be logically different in context of different words. However, knowlege about such alignment can assist us to better model the character states at testing time.

4.4 OCR systems using HMMs

The hidden Markov models (HMMs) have several powerful capabilities like simultaneous segmentation and recognition, automatic training on non-segmented data and also language-independent training and recognition. One could think of using HMMs in classical, scanner based OCR systems. The approaches to recognize printed and scanned text with HMM have begun early after 1990. The first efforts report limited success like in [88]. Such works show clearly that approaches like Bayesian multivariate density formula or pattern matching deliver better results, where characters are well-formed and separated from their neighbors. As stated earlier, the disadvantage of such methods is a segmentation-first dependency and consequently they prove to have massive difficulties, when it comes to the recognition of blurred and connected characters. Therefore, in the area of recognition of printed text, HMMs have been used for special cases when segmentation and recognition needs to be performed at the same time. This can be for example poorly printed text that consists of connected and degraded characters forming a word. Bose et al. introduce in [89] the promising results using HMMs for such condition. Since their system has been one of the earliest works in this area, the authors have used a combined structural and statistical approach. Their system is based on a sub-character segmentation by a structural analysis to identify the dominant strokes for the segmentation process. These strokes have then been used to obtain the number of states in their HMM topology. Therefore, the HMM topology they have used is a left-right topology by which each character has different number of states based on the pre-segmentation process. At training time, they have generated a set of approximately 550 non-overlapping training characters for one font at several blur models using a specific noise model and have kept other noise parameters constant. However, estimation of

transition and emission probabilities at training time has been performed with automatic HMM training. At recognition time, they have used Viterbi scoring to find the character string that best matches the given connected and degraded string. They have achieved reliable results for recognizing words that have been modeled using the described character topology.

In [90] the authors have developed an omnifont open-vocabulary OCR system that is principally capable of recognizing printed text from most of the world's languages. Their system depends on the estimation of character models, as well as a lexicon and a grammar, from training data. Both for training and test they perform some preprocessing on the scanned data like deskewing the page and locating lines of text. They then divide the line into a sequence of overlapping frames. Each Frame is a narrow vertical strip whose width is a small fraction of the line and they normalize the height of the frame to have scale invariancy, i.e. minimizing the independency of the font size. Then they apply a feature extraction for training and test by selecting script-independent and simple features like intensity, vertical and horizontal derivative of intensity and local slope and correlation. They obtain a 80 dimensional feature vector. For computational reasons, they divide their 80 dimensional feature vector into eight separate subvectors of 10 features each, that are modeled to be conditionally independent so that they can perform the probability density as a product of eight probabilities. This probability density is modeled with a GMM with 64 components. They model each character by a 14 states left-right second order HMM. However, they do not use the possibility to have a different number of states for each character, i.e. more states for wider character like 'w' and less states for narrower characters like 'i' and 'l'. They have also developed a mathematical formula to find out the proper amount of training data each for plain and italic data so that the system could be assumed to behave well for a mixed style recognition task. The system also considers statistical language models at character and word level, i.e. character bigrams, character trigrams and word bigrams. While injecting character bigram and trigram constraints into their system makes their system an open-vocabulary system, i.e. unlimited vocabulary can be recognized, injecting word bigram constraints make the system dependent of a closed vocabulary. Finally they report of a hybrid system that performs character-based recognition with some high level constraints set by a word lexicon and a unigram language model at word level. With the hybrid system they obtain recognition rates, which were closed to the closed vocabulary system that contained word bigram constraints. They report of a fairly good character error rate for the hybrid system that is 1.1% for English scripts and 3.3% for arabic scripts.

Velagaupudi in [91] reports on the advantage of using HMMs to boost accuracy in OCR systems by using the Viterbi error correction for a limited vocabulary. He uses a pre-segmented image data of a vocabulary of 6877 handwritten English words. He uses additionally different classifier like neural networks, support vector machines and multi layer pereceptrons. He has shown that for the limited vocabulary cases, Viterbi error correction boosts the accuracy about 10% in average.

4.5 Cursive handwriting recognition using HMMs

In the cursive handwriting the only a priori segmentable entities are whole words and not single characters. Therefore, traditional *'segmentation first'* approaches can not be used for cursive handwriting recognition. Hence, HMMs can be used as an intrinsic classification task as they are capable to segment implicitly during the recognition.

4.5.1 On-line handwriting

The development of mobile terminals like PDAs, electronic notepads, electronic books and mobile phones creates an increasing need for new types of mediums like electronic pens. Electronic pens are likely because first the mobile terminals are mostly too small for keyboards and second some people prefer pen input interface that corresponds to their handwriting habit . A recognition system for such mobile devices requires to be firstly adapted to work with relatively low level memory and processor resources and secondly be easily customizable to allow users to define their own writing style, signs, abbreviations etc. Most of the existing systems lack still these properties as for example user have to enter isolated characters and still with pre-defined shapes to make the recognition system to deal with their handwriting. Hidden Markov models have intrinsic properties that make them attractive for handwriting recognition where characters are connected in the words and segmentation need to be done simultanouesly with the recognition. Specifically, on-line handwriting has some common issues with the problem of continuous speech recognition. Because it can be viewed as a signal (x, y, coordinates) over time. Besides the shape of a handwritten character depends on its neighbors. On the other hand, handwriting has also some differences with speech. For example the dots and crosses involved in the characters 'i' and 'j' and 'x' and 't' are added after the whole word is written and therefore the characters are not fully recognized until the whole word is written. In the following we introduce some works in this field, which have used HMMs.

Hu and Brown in [92] introduce a HMM based on-line handwriting system that incorporates HMMs into a complex stochastic language model. The pattern elements of the handwriting model are subcharacter stroke types modeled with HMM's. They also introduce new invariant features called *similitude* transformation that is a combination of translation, rotation and scaling and are invariant with respect to these three factors of geometric distortion. Their test and training model is a first order HMM connecting subcharacters as strokes to build a character model. They have tested their system on almost 4000 unconstrained word samples from 18 writers covering a 32 word vocabulary containing confusing word pairs like *cut* and *out* and achieved an overall word recognition rate of 94%.

Stanter et al. in [93] describe an on-line cursive handwriting recognition based on HMMs. They use the HMM based continuous speech recognition system called BYBLOS for their purpose. The whole system is unmodified except for using another feature vectors than the ones for automatic speech recognition. The system is based on a left-right second order HMM in which

each character has 7 states. Additional HMMs are used to model contextual effects between adjacent characters. A statistical bigram grammar to relate pairs of words has been also created for a larger set of sentences from ATIS corpus. Experiments have shown that both context and grammar are powerful tools to improve the recognition rates. Motivated by these results, they have embarked on evaluating a larger vocabulary consisting of 25000 words and 6 writers including non-native writers. They have achieved an average of 4.2% word error rate and 1.4% character error rates.

Tokuno et al. in [94] report on a context-dependent substroke model for on-line handwriting recognition that uses substroke HMMs for Kanjai and Hiragana characters in Japanese. The authors have used HMM-substrokes to reduce the huge amount of memory and enormous computation time. As there are more than 6000 characters in those Japanese writing systems, using this system allows them to model each character as a concatenation of only 25 substroke HMMs. They have achieved a CRR of 92% for cursive Kanjai and 98% for Hiragana handwritings.

Funada et al. in [95] prpose an on-line handwriting recognition algorithm that is based on an HMM Self-Organizing Map(SOM) density tying in order to reduce memory for Japanese handwriting. SOM density tying has reduced the dictionary size to 1/7 of the original size. The CRR without SOM has been 91.34%, while using SOM in the recognition system has slightly reduced the CRR to 90.45%.

Bellagarda et al. in [96] introduce a possible solution to on-line handwriting recognition in the domain of small portable platforms. Such platforms have limited resources and this makes considering a continuous parameterization of hidden Markov models not practical. Therefore, the authors propose a discrete parameterization for HMMs using allographic methods. The discrete labels come from an alphabet of sub-characters, elementary handwriting units. To decrease the CPU usage they use a clustering phase followed by a pruning phase. They achieve over all speakers a large decrease of CPU time for experiments, while the average error rates remains essentially unchanged. This reduction varies from a factor of 14 for writer indepenendent(WI) to a factor of 20 for writer dependent(WD) experiments. They show that discrete parameter HMMs work well on small platforms with limited resources.

Biadsy et al. in [97] introduce the first HMM-based solution to on-line Arabic handwriting recognition. On-line handwriting recognition of Arabic script is a challenging task as it is both cursive and unconstrained. Arabic script has some similarities to Roman script, because it uses spaces and punctuation markers to separate words. However, Arabic script is more challenging than Roman script to recognize as arabic script has more dots above or under the letters. Besides eliminating, adding or moving a dot or more would produce a completely new word other than the one was intended. Arabic script has more variations of *delayed strokes* attached to a letter body which creates a new word than those in Roman script. Since in the Arabic script each letter has 2 to 4 shapes, the authors have decided to use each letter shape as a character. The underlying HMM is a discrete Left-to-right model first order using empirically chosen number of states for each letter shape. In addition, to enhance the word recognition, the authors have

added more words constraints into the system by embedding each letter shape model in a word-part dictionary network. They achieve reliable results for a test corpus of 6000 word parts written by ten writers down to 94.4% for a word part dictionary of 40'000 words. One must alsoconsider the fact that the authors could not compare their results, as no Arabic on-line handwriting recognition system previously existed.

Okumara et al. in [98] propose a new HMM implementation for on-line handwriting recognition that uses not only pen-direction feature but also pen-coordinate feature to describe accurately the shape variation of on-line Japanese characters. By their proposed HMM, a pen-coordinate feature is observed at an inetr-state transition and a pen-direction feature is observed at an intra-state transition, i.e. self-transition. Thus, each state specifies the starting position and the direction of a line segment by its incoming inter-state transition and its self-transition. Additionally, they have implemented another HMM framework for multi stroke characters by embedding the proposed HMM into a stroke-order free character recognition framework called *cube search* [99]. *Cube search* is a technique for determining the optimal stroke-to-stroke matching between two K-stroke on-line chinese character patterns. The authors could improve significantly the recognition results by using the additional pen-coordinate feature both for single stroke and for multi stroke Chinese characters.

Kosmala et al. in [100] present a novel method for the introduction of context dependent hidden Markov models for on-line handwriting recognition. They use an advanced tree-based state clustering that is more advantageous than the data driven clustering especially for large vocabulary of 200'000 words. The advantage of tree-based clustering is to map unseen n-graphs onto seen n-graphs by analyzing class memberships of the context graphemes. To give an example 23'000 unseen words in their test set can be synthesized by mapping them onto 2'000 seen trigraphs. The mapping process is carried out by means of questions, which concern a set of HMM states S. The state set S is collected on the root of the tree. Applying question q splits the set S into different subsets. This process is carried out until all leave nodes are reached or until a further splitting yields no significant increase of log-likelihood of the training data. The clustering is done on trigraphs. The tree-based clustering of trigraphs allows to incorporate context dependencies into the system that leads to better recognition rates comparing to context independent HMM character models. They achieve with the explained trigraph approach the highest accuracy of 90.6% and a relative error reduction of 36%.

Humm et al. in [101] propose a novel user authentication system using combined acquisition of online signature and speech modalities. The system is based both on GMMs and HMMs, whereas the authors show in this paper that GMMs can be advantageously replaced by HMMs by considering the optimal number of states for HMM models and also using the maximum a posteriori adaptation (MAP) for training of model parameters. They further report on a significantly better performance when both modalities are used, i.e. on-line handwritten signature and speech, than each modality used alone.

4.5.2 Off-line handwriting

Off-line handwriting can be divided into three areas: *Handwriting interpretation* that is the task of determining the meaning of a body of handwriting like letter addresses, *Handwriting recognition* is about determining the meaning of a handwriting text and *Handwriting identification and/or verification* is determined to authenticate (identify/verify) the author of a special text for example genuine signatures. Off-line handwriting is an unique task for every individual like DNA, finger prints, iris, etc. As computerization is becoming more prominent these days, handwriting recognition is gaining more importance in different fields like signature authentication in the banks, recognition of ZIP codes and addresses on letters or recognizing the handwritten legal amount on personal checks etc. The task of off-line handwriting recognition is more difficult than on-line handwriting recognition due to the fact that off-line handwriting recognition can not utilize the information inherent in the writing trace like time and pressure of the pen because only binary images of handwritten words are available and hence it is not an easy task to reconstruct the writing movement from the image without knowing the correct meaning of the character sequence.

Mohamed et al. in [102] describe a lexicon-based, handwritten word recognition system that combines segmentation-free and segmentation based techniques. The segmentation-free technique applies lexicon information and character level information provided by continuous density HMMs. The segmentation-based method uses a segmentation technique as a combination of connected components and a further splitting module that results to a set of primitive segments. This set of primitives is passed through a dynamic programming matching. The dynamic programming matching delivers the best match between the primitives and characters in a string representing the word image. Each method delivers reliable recognition rates down to 78%. However, the combination of these two techniques improves significantly the recognition rate down to 89%.

Chen et al. in [103] propose a complete scheme for totally unconstrained handwritten word recognition based on HMMs. They use a modified Viterbi algorithm to improve the performance by choosing the L-best paths, instead of a single best path and a two step re-estimation strategy to check which of the L-best paths satisfies a given criteria. They use a variety of features like moment features, geometrical and topological features, pixel distribution features and also reference line features to capture information among the segments. The authors have developed a comprehensive system for estimation of model statistics. For example they estimate the emission probabilities by using the K-nearest neighbor and Vector Quantization (VQ) classifier. At the recognition time, for each iteration of the modified Viterbi algorithm, an optimal state sequence is obtained. If its corresponding letter sequence exists in the dictionary, the modified Viterbi algorithm stops and this word is said to be the result of *Direct recognition*. On the other hand, if within 20 iterations the modified Viterbi algorithm, still can not find a word existing in the dictionary as a perfect match, a *Hypothesis Generation* scheme will be applied

to. The hypothesis is ranked by matching all the 20 optimal state sequences to every word in the dictionary with equal costs for deletion, insertion and substitution.

Vinciarelli et al. in [104] introduce an off-line recognition system for a large vocabulary of unconstrained handwritten texts that uses HMMs. By assuming that the written text is in English, they develop a statistical language model that helps improving the recognition system. Their statistical language model makes use of the dependency of the words to the previous written words. They call this dependency the statistical language model (unigrams(no word dependency is considered), bigrams (dependency of adjacent words) and threegrams(dependency of two previous and following words)) and calculate the transition probabilities for a large lexical (10'000 to 50'000 words) of unigrams, bigrams and trigrams to train their HMM based model. The model topology is left-right HMM assuming that all the letters in the Latin alphabet have the same number of states and Gaussains per state. They obtain the required number of states and Gaussian components through a validation test that determines the best score among different systems having different states $10 \leq S \leq 14$ and $10 \leq g \leq 15$. The words are modeled as letter sequences having the number of states gained through the described validation test. They use various offline-handwriting databases for their evaluation test and get different results showing that the recognition improvement using statistical language models is database dependent. Some databases like Reuters seem to be more aligned than the others and in these databases they obtain no further improvement using trigrams over baseline of twograms, whereas in the IAM database using trigrams transition probabilities to train the system improves system's performance.

Zimmermann et al. in [105] investigate the impact of bigram and trigram language model on the performance of a HMM based off-line handwritten recognition system. They use N-gram language models that provide a simple approximation for sentence probabilities based on the relative frequencies of word sequences of length n. They set in their experiments N=2 (resp. 3). They calculate the frequencies of bigrams and trigrams in a relative big training data. Additionally they add an arbitrary amount of (grammatically correct) random sentenced to the data set and again calculate the bigram and trigram frequencies. Their results show that the trigram language models outperform significantly the bigram language models. This finding is in contrast to [104] where no improvement of the system using a trigram language model over bigram language model has been achieved. The authors explain the reason of this mismatch due to the limited number of words in [104] and a not reliable alignment of linguistic resources in the experimental set ups of [104] in comparison to their experiments. The authors also find out that using the random sentences does not bring any improvement over the baseline system. The authors mention the reason here for due to the using a general-purpose treebank grammar rather than a well-aligned small and task-specific random sentences used by [104].

Marti et al. in [106] present the improvements they have achieved by using statistical language model in their HMM-based cursive handwriting recognition system. They apply firstly some preprocessing like line separation, skew and slant correction and finding the baseline.

They extract nine geometrical features from sliding windows of one column width and the image's height m, moving from left to right over each text line. In their system, a 14 states HMM model is build for each character. They choose a left to right first order HMM topology for the character models. At training time, the initialization of the model has been done by using the Viterbi alignment for segmenting the training observations and recomputing model parameters and reestimating of transition and emission probabilities is done by applying the Baum-Welch algorithm. At recognition time, the character models are concatenated to words and words to the sentences leading to a recognition network. This network does not include any contextual knowledge at character level. They use Viterbi algorithm to find the best path in their model. Their system automatically segment the lines into words and words into characters. This HMM has been improved using a language model incorporating linguistic information beyond the world level. Their system have included constraints of word-unigrams and word-bigrams that were automatically obtained from a large text corpus. They show that the incorporation of language models clearly enhance system's performance.

The task of handwritten writer identification and verification is addressed by Schlapbach et al. in [107]. They use a HMM based handwriting recognition system that consists of N different HMMs for N different writers. All the HMMs have the same architecture but differ from each other as each HMM has its own transition and emission probabilities. For the identification task, a text line of unknown identity is presented to each HMM based recognizer. Each recognizer delivers a recognized text line with a recognition score. The system claims the person to be the writer for whom there has been obtained the highest score among all the HMM recognizers. They achieve an overall writer identification rate in 96.56% of all cases. Additionally they use a confidence measure as an error rejection that is calculated using the log-likelihood scores of first N ranks of the underlying HMM recognizer. By changing the number of N they have achieved different error rejection rates that have improved the identification rate. For the writer verification task, they decide whether a text line with a text line identity was written in fact by an author or not. For this, they have a set of text lines each belonging to a person and a set of forgeries as malicious attempts af impostors trying to access an authorizing system. They test their system for different values of N and achieve an equal error rate of 2.5%. Similarly to the writer identification results, they observe that increasing the number of N to calculate the error rejection rate has an impact on the performance of their system.

For the task of off-line writer identification, in [108] Schlapbach et al. have compared the previously described HMM system [107] with a system using Gaussian mixture models (GMMs). Both systems are evaluated on two test sets. The first set is a unskillfully forged set containing in total 8'100 text lines from 100 clients and 20 impostors. The second test is a skillfully forged set containing 238 text lines from 20 clients and 20 impostors. They have studied three different confidence measures for their evaluation. These are based on the raw log-likelihood score, a cohort model and a world model approach. The authors have found out that GMM system works better than HMM system on confidence measure based on the raw log-likelihood

score. This is not surprising, because the HMM system is made of several states and needs both trained transition and emission probabilities, whereas GMM system consists of one state and needs only the trained emission probabilities. Consequently by applying world model and cohort model the authors get better results for their HMM system than GMM system, that is what can be expected, as HMMs model the words and characters much precisely than GMMs and take also the character orders in a word and word orders in a sentence into consideration by using a left-right character and word topology. The lowest error rate they get is with their HMM system using the cohort model as confidence measure for both test sets, i.e. skilled and unskilled forgeries.

Chapter 5

Study of ULR single characters

As stated earlier, the ULR text has some specificities that makes its recognition a difficult task. In this chapter, we present a study about anti-aliased single characters that are of ultra low resolution and with small font sizes. Single characters are either isolated or reside in the context of a word. In the latter case we call such characters *contextual characters*.

In this chapter we first introduce our selected features and report on the tests that we have employed to investigate their distribution. Second, we will describe our implemented single character identification system. To assess performance of our identification system we have conducted evaluation tests for the identification of both *single isolated characters* and *single contextual characters*. We will present results, error analysis and conclusions for both cases. The chapter ends with our final conclusion about the conducted study.

5.1 Feature extraction

5.1.1 Selected features

In chapter 4, we introduced some common features used in classical pattern recognition. Most of the introduced features work properly for printed bilevel text that has a minimum resolution of 150 dpi. Therefore, we sought for features that could deal with an average height of between $6 - 12$ pixels and would take advantage of the additional information contained in gray values of anti-aliased pixels.

We have chosen *central moments* for three reasons:

- Central moments can convey satisfactory spatial information about the whole geometric shape of an object.
- Central moments are translation invariant. We expect the shift-invariance property to hold also in sub-pixel domain.

- Central moments are computationally more cost efficient than for example Fourier descriptors or Zernike moments.

Therefore we decided to study them more deeply for our dissertation.

The general formula of central moments is already introduced in chapter 4. Here we introduce the central moments we used as features for our work. The central moments are related to the center of gravity. The coordinates of the center of gravity are $\hat{x}$ and $\hat{y}$. The mathematical definition of $\hat{x}$ and $\hat{y}$ is described in chapter 4. Central moments can be defined by

$$\mu_{pq} = \sum_x \sum_y (x - \hat{x})^p (y - \hat{y})^q f(x, y) \qquad p, q = 0, 1, 2, .. \tag{5.1}$$

where $f(x, y)$ are gray values of a pixel at x and y coordinates.

In chapter 4, we have defined the moment features as follows:

$$m_{pq} = \sum_x \sum_y x^p y^q f(x, y) \qquad p, q = 0, 1, 2, .. \tag{5.2}$$

The central moments we use are μ_{00}, μ_{11}, μ_{20}, μ_{02}, μ_{21}, μ_{12}, μ_{22}. One can verify that the central moments μ_{10} und μ_{01} are zero.

When using the definition of moment features, our selected central moments can be written as follows:

$$\mu_{00} = m_{00} \tag{5.3}$$
$$\mu_{11} = m_{11} - \hat{x} m_{01} = m_{11} - \hat{y} m_{10} \tag{5.4}$$
$$\mu_{20} = m_{20} - \hat{x} m_{10} \tag{5.5}$$
$$\mu_{02} = m_{02} - \hat{y} m_{01} \tag{5.6}$$
$$\mu_{21} = m_{21} - 2\hat{x} m_{11} - \hat{y} m_{20} + 2\hat{x}^2 m_{01} \tag{5.7}$$
$$\mu_{12} = m_{12} - 2\hat{y} m_{11} - \hat{x} m_{02} + 2\hat{y}^2 m_{10} \tag{5.8}$$

5.1.2 Verification of marginal features' normal distribution: χ^2 test

As stated in chapter 4, we decided to use methods of statistical pattern recognition for our work. Further we chose to use normal multivariate density functions for estimating the likelihood of an unknown character image to belong to one of the 52 classes. The parameters of multivariate density functions are obtained assuming that extracted features have a normal distribution. In this section we report on the investigation we employed to verify the normal distribution of selected features. We used χ^2 test for this purpose.

χ^2 test is a statistical test for testing the null hypothesis that the distribution of a discrete random variable coincides with a given distribution. Null hypothesis is an assertion before the

analysis that has yet proven . χ^2 test is one of the most popular goodness-of-fit tests. As in all significance tests, a sufficiently large sample size is necessary. There is no accepted cutoff. Some set the minimum sample size at 50, while others would allow as few as 20. The hypothesized distribution is specified in advance, so that the number of observations that are expected to appear, can be calculated without reference to the observed values. As already described in chapter 3, we produce samples by down-sampling high resolution character images. For this test, we have chosen a down-sampling factor of 10. This allows us to have 100 different patterns for each class that is judged enough for the χ^2 test. Following steps need to be performed for the χ^2 test:

- *Specification of the null hypothesis* H_0, i.e. with which standard value P(in %) the zero hypothesis shall be accepted. P is called the *significance value*. In practical environments this standard value P is mostly 95% or 99%. With 100 samples in our case we have decided for $P = 95\%$ which can be considered as an acceptable significance level for the distribution.

- *Building the classes.* When the significance value P is 95%, we can build 6 classes to distribute 100 samples. We define the classes as follows:

  - 1.class: $[\mu - 3.\sigma^2\ ,\ \mu - 2.\sigma^2[$
  - 2.class: $[\mu - 2.\sigma^2\ ,\ \mu - 1.\sigma^2[$
  - 3.class: $[\mu - 1.\sigma^2\ ,\ \mu[$
  - 4.class: $[\mu\ ,\ \mu + 1.\sigma^2[$
  - 5.class: $[\mu + 1.\sigma^2\ ,\ \mu + 2.\sigma^2[$
  - 6.class: $[\mu + 2.\sigma^2\ ,\ \mu + 3.\sigma^2]$

  where μ is the mean and σ^2 the variance of the distribution

- *Calculation of the theoretical frequency in % for each class*

- *Assigning frequency limits for each class*

- *Calculation of the expectation value E for each class* as follows:

$$E = \frac{\text{theoretical frequency}.N}{100} \tag{5.9}$$

  where N = number of samples
  In order to have enough samples in a class, we consider only classes with a $E \geq 3\%$. Classes that do not fulfill the previous condition are merged with adjacent class

- *Calculating B as the observed frequency value of each class*

- *Determining the degree of freedom* $f = k - 1$ *where* $k =$ *number of classes*
- *Calculation of the* χ^2 as follows:

$$\chi^2 = \sum_i \frac{(E_i - B_i)^2}{E_i} \tag{5.10}$$

- According to a standard look-up-χ^2-table, if the calculated total χ^2 value is higher than the expected χ^2 value in the table for the specific number of degree of freedom, then the null hypothesis is rejected.

We produced two data bases each containing 100 samples for each of the 52 upper and lower case character classes for two different rendering methods. The chosen font was Arial in plain style with 9 points. The methods we used, were Adobe Photoshop and a method contained in the Java library [109]. Next we calculated the features for each class and investigated their normal distribution using the χ^2 test.

Fig. 5.1 shows the results of χ^2 test for Adobe Photoshop and for Java rendering method. The features extracted from Adobe Photshop have a normal distribution in 93.13% of cases. Those extracted from Java show a far less normal distribution, i.e. in 31.87% of the cases. Therefore features related to Adobe Photoshop have mostly a normal distribution whereas those extracted from Java rendering method are most likely not distributed normally. The reason herefor could be that the variability of samples produced with the Photoshop method are much greater than of samples produced with Java method. Fig. 5.2 and Fig. 5.3 show 15 different samples produced with Photoshop and Java method. The calculated standard deviation of Photoshop patterns is between 5-10 times bigger than the one for Java patterns. We assume that the patterns produced with Java patterns are probably hinted and therefore have a low standard deviation as the shapes of hinted anti-alised patterns show a marginal spatial variability as demonstrated in chap 3.

5.2 System description

This chapter describes the single character identification experiment. We have set up a character identification system as illustrated in Fig. 5.4. With this system, each test corresponds to an identification task that aims at determining what character is the most probable for a given ULR character image as input. The system computes the features of an unknown character image. This feature vector is then fed into M parallel models with M equal to the number of different characters to recognize. The system is supposed to be applicable both for isolated and contextual characters. For the isolated characters, we considered a feature vector consisting of 7 central moments and used 52 character models. For the recognition of contextual characters

Character	m00	m11	m20	m02	m21	m12	m22
A	✓	✓	✓	✓	✓	✓	✓
B	✓	✓	✓	✓	✓	✓	✓
C	✓	✓	✓	✓	✓	✓	✓
D	✓	✓	×	✓	✓	✓	✓
E	✓	✓	✓	✓	✓	✓	✓
F	✓	✓	✓	✓	✓	✓	✓
G	✓	✓	✓	✓	✓	✓	✓
H	✓	✓	×	✓	✓	✓	✓
I	✓	✓	✓	✓	×	×	✓
J	✓	✓	✓	✓	✓	✓	✓
K	✓	✓	✓	✓	✓	✓	✓
L	✓	✓	✓	✓	✓	×	×
M	×	✓	✓	✓	✓	✓	✓
N	✓	✓	×	✓	✓	✓	✓
O	✓	✓	×	✓	✓	✓	✓
P	✓	✓	✓	✓	✓	✓	✓
Q	✓	✓	✓	✓	✓	✓	✓
R	✓	✓	✓	✓	✓	✓	✓
S	✓	✓	✓	✓	✓	✓	✓
T	✓	×	✓	✓	✓	✓	✓
U	✓	✓	×	✓	✓	✓	✓
V	✓	✓	✓	✓	✓	✓	✓
W	✓	✓	✓	✓	✓	✓	✓
X	✓	✓	✓	✓	✓	✓	×
Y	✓	✓	✓	✓	✓	✓	✓
Z	✓	✓	✓	✓	✓	✓	✓
a	✓	✓	✓	✓	✓	✓	✓
b	✓	✓	×	✓	✓	✓	✓
c	✓	✓	✓	✓	✓	✓	✓
d	✓	✓	✓	✓	✓	✓	✓
e	✓	✓	✓	✓	✓	✓	✓
f	✓	✓	✓	✓	✓	✓	✓
g	✓	✓	✓	✓	✓	✓	✓
h	✓	✓	✓	✓	✓	×	✓
i	✓	✓	✓	✓	×	✓	✓
j	✓	✓	✓	✓	✓	✓	✓
k	✓	✓	✓	✓	✓	✓	✓
l	✓	✓	✓	✓	✓	✓	✓
m	✓	✓	×	✓	✓	✓	×
n	✓	✓	✓	✓	✓	✓	✓
o	✓	✓	×	✓	✓	✓	✓
p	✓	✓	×	✓	✓	✓	✓
q	✓	✓	✓	✓	✓	✓	✓
r	✓	✓	✓	✓	✓	✓	✓
s	✓	✓	✓	✓	✓	✓	✓
t	✓	✓	✓	✓	✓	✓	✓
u	✓	✓	✓	✓	✓	✓	✓
v	✓	×	✓	✓	✓	✓	✓
w	✓	✓	✓	✓	✓	✓	✓
x	✓	✓	×	×	×	×	×
y	✓	✓	✓	✓	✓	✓	✓
z	✓	✓	✓	✓	✓	✓	✓

(a) Results of χ^2 test for Photoshop features

Character	m00	m11	m20	m02	m21	m12	m22
A	✓	×	×	×	✓	×	×
B	×	✓	×	×	✓	✓	×
C	✓	×	×	×	×	×	×
D	×	×	×	×	×	×	×
E	×	×	×	×	✓	✓	×
F	×	✓	×	×	×	✓	×
G	✓	✓	×	×	×	×	✓
H	×	×	×	×	✓	×	✓
I	×	×	×	×	×	×	×
J	×	✓	×	×	×	×	×
K	✓	✓	×	×	×	✓	×
L	×	✓	×	×	✓	✓	×
M	✓	×	×	×	×	×	✓
N	×	×	×	×	×	✓	✓
O	✓	×	×	×	×	✓	×
P	✓	×	×	×	×	×	×
Q	✓	✓	×	×	✓	✓	✓
R	✓	✓	×	×	✓	✓	×
S	×	✓	×	×	✓	✓	×
T	×	×	×	×	×	×	×
U	×	✓	×	×	×	×	×
V	✓	×	✓	×	×	×	×
W	×	×	×	×	×	×	✓
X	✓	✓	×	×	×	✓	×
Y	✓	✓	×	×	×	✓	×
Z	✓	✓	×	×	✓	×	×
a	✓	✓	×	×	✓	✓	✓
b	×	×	×	×	✓	✓	×
c	✓	×	×	×	✓	×	✓
d	✓	×	×	×	✓	×	×
e	✓	×	×	×	✓	×	×
f	✓	×	×	×	✓	×	×
g	✓	×	×	×	✓	✓	✓
h	✓	✓	×	×	✓	✓	×
i	✓	×	×	×	×	×	×
j	✓	×	×	×	×	×	×
k	✓	✓	×	×	✓	✓	×
l	×	×	×	×	×	×	×
m	✓	×	×	×	×	✓	×
n	✓	×	×	×	×	✓	×
o	✓	✓	×	×	×	×	✓
p	✓	×	×	×	✓	✓	×
q	✓	×	×	×	×	×	×
r	✓	✓	×	×	✓	×	×
s	✓	×	×	×	✓	✓	✓
t	×	×	×	×	×	✓	×
u	✓	×	×	×	×	×	×
v	✓	✓	×	×	×	✓	×
w	✓	×	×	×	×	×	×
x	✓	✓	×	×	✓	×	×
y	✓	×	×	×	✓	×	✓
z	✓	✓	×	×	×	×	×

(b) Results of χ^2 test for Java features

Figure 5.1: Results of normal distribution test for extracted features of 52 character classes

a feature vector consisting of 8 components and a set of 26 character models were considered. Using this identification system we were able to, first, assess the discriminative power of the selected features, and secondly, to study the impact of rendering algorithms by calculating the resulting chracter recognition rates(CRR). We were additionally interested to assess the performance of the system for both isolated and contextual characters. The probability density function of feature vector $x = \{x_1, .., x_D\}$ given the character class c_i was estimated using a multivariate mono-Gaussian function as follows:

$$p(x|c_i) = N(x, \mu_i, \Sigma_i) \tag{5.11}$$

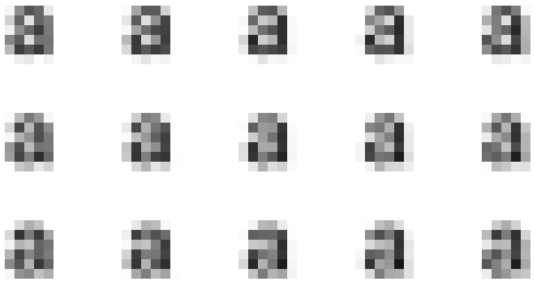


Figure 5.2: 15 samples of character 'a' downsampled with Photoshop

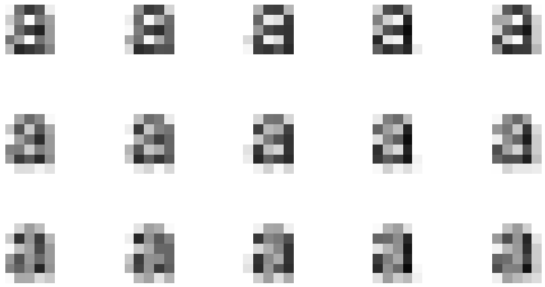


Figure 5.3: 15 samples of character 'a' downsampled with Java

in which $i = 1...M$ is the index of the character class and the Gaussian density N is parameterized by a mean $D \times 1$ vector μ_i and a $D \times D$ covariance matrix Σ_i, where D is the number of features.

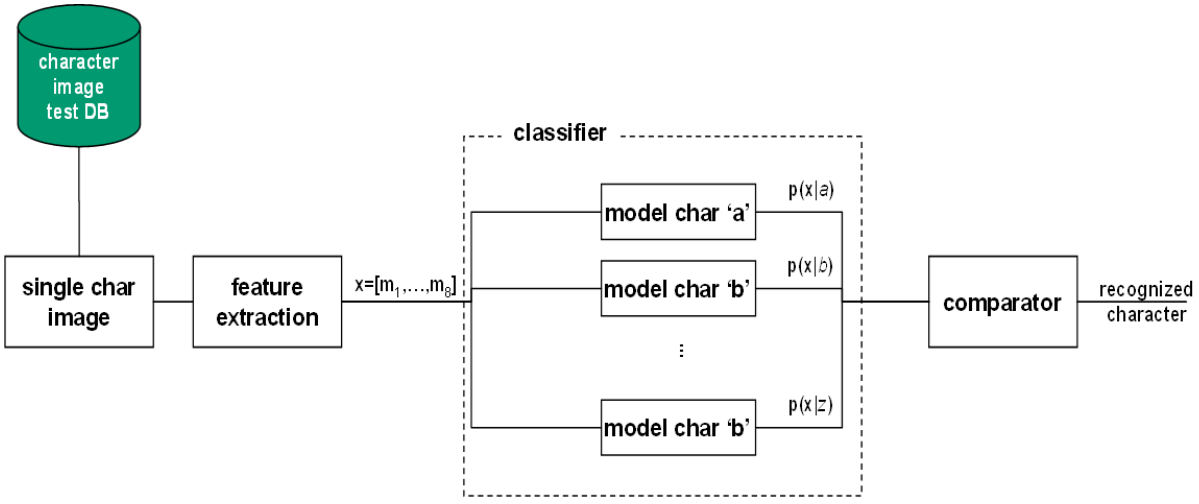


Figure 5.4: Schematic view of the single character identification system

Classification can then be performed in the comparator block of Fig. 5.4 with

$$I^* = \text{argmax} \quad p(x|c_i) \tag{5.12}$$

where I^* is the index of the winner character class. In this last equation, we make the assumption that a priori probabilities of character classes are all equal, which is not always the case in practice.

5.3 Experiments on single isolated characters

5.3.1 Introduction

This section aims at demonstrating the of central moments as the chosen features to discriminate isolated, anti-aliased characters at ultra low resolution and small font sizes. We performed three different evaluation tests on single isolated characters. The first two experiments were conducted in a so called *mono-font* context assuming that the font family was supposed to be known and the system could thus choose the appropriate training set for the character recognition. Such an approach is justified when combining judiciously font recognition using an Optical Font Recognition (OFR) tool that is, for example developed in our group, with character recognition.

- *General stuy*:
  These series of experiments aimed at gaining character recognition rates in the cases of varying font sizes, font styles and font families. These experiments were performed using Photoshop rendering method. We chose Photoshop as the rendering method for our general study. We verified that the distribution of the extracted features from characters rendered by this method, is normal in most of the cases. Aside from this fact, the features extracted from characters rendered by Photoshop show a much greater variability than extracted features of patterns rendered by the Java method.

- *Impact of rendering method*:
  This set of experiments had the objective of investigating the impact of two different methods that used anti-aliasing filter for character rendering (Photoshop & Java).

- *Multifont experiment*:
  In the third evaluation series, we tested the system performance of a so called *multi-font* character identification system, where the corresponding training set would contain characters of different font families. While the font size and style were assumed to be known, such a system has the advantage that character images of an unknown character can be recognized without a priori knowledge of its font family.

5.3.2 General study

Our character identification system is described in fig. 5.4. It is fed with a data base containing synthetical isolated characters. For this experiment we have chosen 196 different fonts by combining the following:

- 4 sans-serif font families: Arial, Comic sans MS, Verdana, Tahoma
- 3 serif font families: Times New Roman, Courier New, Century
- 4 font styles: roman, italics, bold, bold & italics
- 7 font sizes: 3-9 points

To gain a synthetical data base containing single isolated characters, we produced 25 different patterns of each 52 classes in big font size and down-sampled them with a factor of $k = 5$ by using our ULR text simulation method described at the end of chapter 3. Therefore, each character class in our database was presented in 4'900 different shapes. Hence, our database included 254'800 isolated characters.
Table. 5.5 shows the absolute number of errors (#) and the percentage of errors (%) when recognizing single isolated characters for different font families, sizes and styles. Each set of experiments contains 1'300 samples. Table 5.6 shows good results for font sizes between 4-9 points. On the other hand, the error rates drastically increases for font size of 3 points. Furthermore, we noticed that a significant number of the errors are due to the confusion between 'i' & 'l' especially for the sans-serif fonts. Table 5.7 shows the number of errors and error rates when such confusion were not taken into account. Fig. 5.8 shows the overall error rate in this case. When we compare table 5.5 and 5.7, we can conclude that the sans serif font *Arial* shows the highest overall number of such confusions, whereas other sans-serif fonts like *Comic sans serif*, *Tahoma* and *Verdana* show this when point size is below 4 points. Generally speaking, one can say that fonts like Verdana, Tahoma, Comic sans Serif that have been designed especially for computer screens, have a higher *legibility* also for our computerized method. Another surprising conclusion of the above experiments is the fact that the machine is able to recognize patterns at very small font sizes, i.e. less than 4 points, whereas the human vision system fails to recognize such small patterns. This fact shows again the outstanding capability of machines to outperform human perception for a particular task as we stated it at the beginning of chapter 4.

The overall CRR results are summarized at table 5.1.

Table 5.2 shows the overall recognition rates for different font styles. We see that the recognition of italic fonts are slightly better than roman fonts. In addition, bold sytle increases the recognition rates both for roman and italic fonts.

Font Size		9		8		7		6		5		4		3	
		#	%	#	%	#	%	#	%	#	%	#	%	#	%
Font Type	Font Style														
	Plain	28	2.15	31	2.38	27	2.08	39	3	34	2.62	0	0.00	157	12.08
	Bold	25	1.92	25	1.92	25	1.92	38	2.92	25	1.92	15	1.15	75	5.77
	Italics	27	2.08	17	1.31	21	1.62	0	0	37	2.85	25	1.92	127	9.77
Arial	Bold+Italics	44	3.38	16	1.23	25	1.92	18	1.38	25	1.92	10	0.77	77	5.92
	Plain	0	0	0	0	0	0	0	0	1	0.08	4	0.31	89	6.85
	Bold	0	0	0	0	0	0	0	0	0	0	5	0.38	54	4.15
	Italics	0	0	0	0	0	0	0	0	0	0	1	0.08	87	6.69
Comic	Bold+Italics	0	0	0	0	0	0	0	0	0	0	0	0.00	60	4.62
	Plain	0	0	0	0	0	0	5	0.38	3	0.23	0	0.00	141	10.85
	Bold	0	0	13	1	0	0	0	0	0	0	0	0.00	116	8.92
	Italics	0	0	0	0	0	0	0	0	4	0.31	0	0.00	138	10.62
Tahoma	Bold+Italics	0	0	0	0	0	0	1	0.08	0	0	0	0.00	66	5.08
	Plain	0	0	0	0	0	0	5	0.38	0	0	2	0.15	100	7.69
	Bold	11	0.85	9	0.69	0	0	0	0	0	0	13	1.00	69	5.31
	Italics	0	0	0	0	0	0	0	0	0	0	0	0.00	86	6.62
Verdana	Bold+Italics	0	0	0	0	0	0	0	0	0	0	0	0.00	55	4.23
	Plain	0	0	0	0	0	0	0	0	1	0.08	2	0.15	261	20.08
	Bold	0	0	0	0	0	0	0	0	0	0	0	0.00	149	11.46
	Italics	0	0	0	0	0	0	0	0	1	0.08	4	0.31	232	17.85
Century	Bold+Italics	0	0	0	0	0	0	0	0	0	0	3	0.23	118	9.08
	Plain	0	0	0	0	1	0.08	6	0.46	11	0.85	14	1.08	115	8.85
	Bold	0	0	0	0	0	0	0	0	0	0	4	0.31	37	2.85
	Italics	0	0	0	0	0	0	12	0.92	2	0.15	18	1.38	79	6.08
Courier	Bold+Italics	0	0	0	0	0	0	0	0	0	0	0	0.00	36	2.77
	Plain	3	0.23	0	0	0	0	0	0	6	0.46	24	1.85	110	8.46
	Bold	0	0	0	0	0	0	0	0	3	0.23	4	0.31	51	3.92
	Italics	4	0.31	0	0	0	0	1	0.08	3	0.23	15	1.15	76	5.85
Times	Bold+Italics	0	0	0	0	0	0	0	0	0	0	9	0.69	24	1.85
Error in %			**0.39**		**0.3**		**0.27**		**0.34**		**0.43**		**0.473**		**7.651**

Figure 5.5: Error table of character identification of different fonts

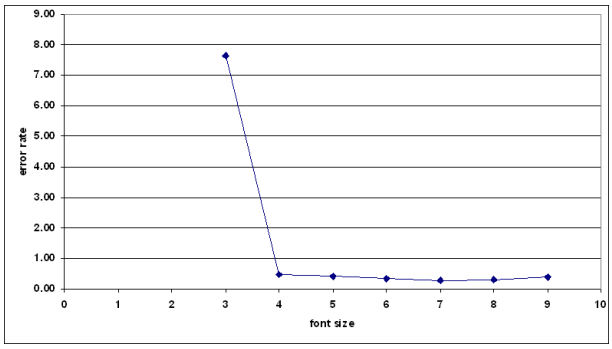


Figure 5.6: Error rate in % for different font sizes

Table 5.1: Overall recognition rates of isolated characters by different font sizes

Font size	9	8	7	6	5	4	3
	99.93%	99.92%	99.99%	99.82%	99.84%	99.66%	92.59%

Table 5.2: Recognition rates of isolated characters for different font styles

	Roman	Italics	Mean
Normal	99.51%	99.71%	99.61%
Bold	99.72%	99.93%	99.83%
Mean	99.62%	99.82%	

5.3.3 Influence of rendering method

In order to investigate the influence of the rendering method on the recognition rates of isolated characters, we performed various experiments using two rendering methods:

Font Size		9		8		7		6		5		4		3	
		#	%	#	%	#	%	#	%	#	%	#	%	#	%
Font Type	Font Style														
	Plain	9	0.69	7	0.54	2	0.15	19	1.46	10	0.77	0	0	137	10.54
	Bold	0	0	0	0	0	0	0	0	13	1	0	0	52	4.00
	Italics	0	0	0	0	0	0	14	1.08	0	0	0	0	104	8.00
Arial	Bold+Italics	0	0	0	0	0	0	1	0.08	0	0	0	0	55	4.23
	Plain	0	0	0	0	0	0	0	0	1	0.08	2	0.154	89	6.85
	Bold	0	0	0	0	0	0	0	0	0	0	0	0	54	4.15
	Italics	0	0	0	0	0	0	0	0	1	0.08	4	0.308	87	6.69
Comic	Bold+Italics	0	0	0	0	0	0	0	0	0	0	3	0.231	60	4.62
	Plain	3	0.23	0	0	0	0	0	0	6	0.46	24	1.846	141	10.85
	Bold	0	0	0	0	0	0	0	0	3	0.23	4	0.308	116	8.92
	Italics	4	0.31	0	0	0	0	1	0.08	3	0.23	15	1.154	138	10.62
Tahoma	Bold+Italics	0	0	0	0	0	0	0	0	0	0	9	0.692	66	5.08
	Plain	0	0	0	0	0	0	0	0	1	0.08	4	0.308	100	7.69
	Bold	0	0	0	0	0	0	0	0	0	0	5	0.385	69	5.31
	Italics	0	0	0	0	0	0	0	0	0	0	1	0.077	86	6.62
Verdana	Bold+Italics	0	0	0	0	0	0	0	0	0	0	0	0	55	4.23
	Plain	0	0	0	0	1	0.08	6	0.46	11	0.85	14	1.077	261	20.08
	Bold	0	0	0	0	0	0	0	0	0	0	4	0.308	149	11.46
	Italics	0	0	0	0	0	0	12	0.92	2	0.15	18	1.385	232	17.85
Century	Bold+Italics	0	0	0	0	0	0	0	0	0	0	0	0	118	9.08
	Plain	0	0	0	0	0	0	5	0.38	3	0.23	0	0	115	8.85
	Bold	0	0	13	1	0	0	0	0	0	0	0	0	37	2.85
	Italics	0	0	0	0	0	0	0	0	4	0.31	0	0	79	6.08
Courier	Bold+Italics	0	0	0	0	0	0	1	0.08	0	0	0	0	36	2.77
	Plain	0	0	0	0	0	0	5	0.38	0	0	2	0.154	110	8.46
	Bold	11	0.85	9	0.69	0	0	0	0	0	0	13	1	51	3.92
	Italics	0	0	0	0	0	0	0	0	0	0	0	0	76	5.85
Times	Bold+Italics	0	0	0	0	0	0	0	0	0	0	0	0	24	1.85
Error in %			**0.07**		**0.08**		**0.01**		**0.18**		**0.16**		**0.34**		**7.41**

Figure 5.7: Error table of character identification of different fonts without confusion between 'l' & 'i'

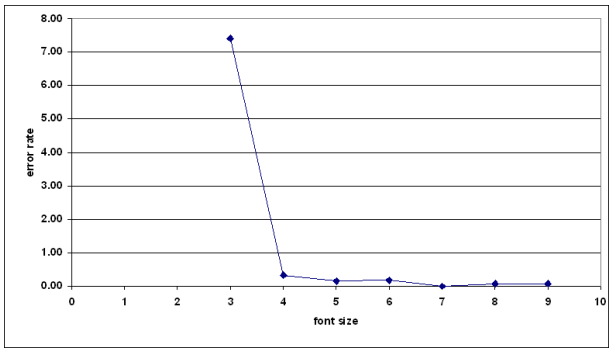


Figure 5.8: Error rate in % for different font sizes without confusion between 'l' & 'i'

Photoshop and Java. We produced two different databases containing isolated characters from different fonts and rendering methods. The evaluation tests were performed using the following combinations:

- 4 sans serif fonts: Arial, Comic sans MS, Tahoma, Verdana
- 3 serif fonts: Century, Courier, Times new Roman
- 1 point size: 8 points
- 4 different font styles: plain, bold, italics, bold & italics

We have performed three sets of experiments as follows:

– *Standard experiment*

In these tests we aimed at evaluating the CRRs when using the same rendering method as training and test set. Table 5.4 shows the results for Photoshop method and table 5.3 for Java method. We gained a CRR of 97% with Photoshop method and of 98% with Java method. The higher CRR for Java method is possibly a result of the fact that the variability of Java patterns is considerably lower than that of Photoshop patterns as previously described in section 5.1.2.

Table 5.3: Influence of rendering method; training set: Java, test set: Java

	plain		bold		italics		bold + italics		Mean
	#	%	#	%	#	%	#	%	%
Arial	25	98.08	25	98.08	21	98.38	0	> 99.9	98.63
Comic	0	> 99.9	0	> 99.9	0	> 99.9	0	> 99.9	> 99.9
Tahoma	0	> 99.9	13	99	0	> 99.9	0	> 99.9	> 99.9
Verdana	0	> 99.9	9	99.31	0	> 99.9	0	> 99.9	> 99.9
Century	0	> 99.9	0	> 99.9	0	> 99.9	2	99.85	99.96
Courier	0	> 99.9	0	> 99.9	0	> 99.9	0	> 99.9	> 99.9
Times	0	> 99.9	0	> 99.9	0	> 99.9	0	> 99.9	> 99.9
Mean in %	-	99.73	-	99.73	-	99.77	-	99.98	99.8

Table 5.4: Influence of rendering method; training set: Photoshop, test set: Photoshop

	plain		bold		italics		bold + italics		Mean
	#	%	#	%	#	%	#	%	%
Arial	31	97.62	25	98.08	17	98.69	16	98.77	98.29
Comic	0	> 99.9	0	> 99.9	0	> 99.9	0	> 99.9	> 99.9
Tahoma	0	> 99.9	13	99	0	> 99.9	0	> 99.9	99.75
Verdana	0	> 99.9	9	99.31	0	> 99.9	0	> 99.9	99.83
Century	0	> 99.9	0	> 99.9	0	> 99.9	0	> 99.9	> 99.9
Courier	0	> 99.9	0	> 99.9	0	> 99.9	0	> 99.9	> 99.9
Times	0	> 99.9	0	> 99.9	0	> 99.9	0	> 99.9	> 99.9
Mean in %	-	99.66	-	99.48	-	99.81	-	99.82	99.7

– *Influence of different models*

These series of experiments had the objective of evaluating the CRRs when one rendering method is used as training set and the other as test set. Table 5.5 shows the unusable results,i.e. an overall CRR of 36.38%, when Java method is used for character modeling to recognize the Photoshop patterns. On the other hand, using Photoshop patterns to recognize Java patterns, delivers an overall CRR of 93.69% (see table 5.6). A comparison of these two results indicates that trained Photoshop patterns are able to recognize Java patterns with good accuracy, whereas trained Java patterns show an unsatisfactory result to recognize Photoshop patterns. Once again, the reason could be that the trained Java patterns show a small variability and thus fail to recognize the Photoshop patterns as previously described in section 5.1.2.

Table 5.5: Influence of rendering method; training set: Java, test set: Photoshop

	plain		bold		italics		bold + italics		Mean
	#	%	#	%	#	%	#	%	%
Arial	946	27.23	771	40.69	981	27.54	600	53.85	36.58
Comic	968	25.54	523	59.77	823	36.69	501	61.46	45.87
Tahoma	984	24.31	677	47.92	870	33.08	455	65	42.58
Verdana	977	24.85	617	52.52	897	31	397	69.46	44.46
Century	988	24	875	32.69	954	26.62	776	40.31	30.9
Courier	1012	22.15	980	24.62	1010	22.31	989	23.92	23.25
Times	950	26.92	896	31.08	923	29	816	37.23	31.06
Mean in %	-	25	-	41.33	-	29.03	-	50.18	36.38

Table 5.6: Influence of rendering method; training set: Photoshop, test set: Java

	plain		bold		italics		bold + italics		Mean
	#	%	#	%	#	%	#	%	%
Arial	80	93.85	57	95.62	226	79.54	58	95.54	91.13
Comic	22	98.31	41	96.85	54	95.85	63	95.15	96.54
Tahoma	41	96.85	5	99.62	119	90.85	50	96.15	95.87
Verdana	19	98.54	6	99.54	83	93.62	26	98	97.42
Century	118	90.92	63	95.15	157	87.92	47	96.38	92.6
Courier	185	85.77	22	98.31	272	79.08	36	97.23	90.1
Times	80	93.85	72	94.46	196	84.92	59	95.46	92.17
Mean in%	-	94.01	-	97.08	-	87.4	-	96.27	93.69

– *Influence of mixed models*

When we mix both sets of patterns and use them as training sets, we can observe excellent CRRs for both test sets: 99.7% for Java patterns (see table 5.7) and 99.59% for Photoshop patterns (see table 5.8).

Table 5.7: Influence of rendering method; training set: Photoshop + Java, test set: Java

	plain		bold		italics		bold + italics		Mean
	#	%	#	%	#	%	#	%	%
Arial	31	97.62	25	98.08	23	98.23	22	98.31	98.06
Comic	0	> 99.9	0	> 99.9	0	> 99.9	0	> 99.9	> 99.9
Tahoma	3	99.77	2	99.85	0	> 99.9	0	> 99.9	99.9
Verdana	0	> 99.9	0	> 99.9	0	> 99.9	0	> 99.9	> 99.9
Century	0	> 99.9	0	> 99.9	0	> 99.9	2	99.85	99.96
Courier	0	> 99.9	0	> 99.9	0	> 99.9	0	> 99.9	99.98
Times	0	> 99.9	0	> 99.9	0	> 99.9	0	> 99.9	> 99.9
Mean in %	-	99.63	-	99.7	-	99.74	-	99.74	99.7

More generally, we can state that the single isolated character classifier can be trained to make it capable of recognizing characters independently to the rendering algorithm.

Table 5.8: Influence of rendering method; training set: Photoshop + Java, test set: Photoshop

	plain		bold		italics		bold + italics		Mean
	#	%	#	%	#	%	#	%	%
Arial	34	97.38	36	97.23	26	98	24	98.15	97.69
Comic	0	100	0	> 99	0	> 99	0	> 99.9	> 99.9
Tahoma	1	99.92	13	99	0	> 99.9	0	> 99.9	99.73
Verdana	0	> 99.9	7	99.46	3	99.77	0	> 99.9	99.81
Century	0	> 99.9	0	> 99.9	0	> 99.9	0	> 99.9	> 99.9
Courier	1	99.92	0	> 99.9	2	99.85	0	> 99.9	99.94
Times	1	99.92	0	> 99.9	0	> 99.9	0	> 99.9	99.98
Mean in %	-	99.59	-	99.38	-	99.66	-	99.74	99.59

5.3.4 Multi-font experiment

We achieved fairly reliable CRRs in the two previous evaluation tests. Both tests were performed using a *mono-font* character identification system. We were interested to assess the performance of the system for a *muti-font* classifier, as such a classifier is more of practical use than a *mono-font* classifier. For this test series we used the following configurations:

- 6 fonts:
  - 3 sans-serif fonts: Arial, Tahoma, Verdana
  - 3 serif fonts: Century, Georgia, Garamond
- 1 point size: 8 points
- 4 font styles: plain, bold, italics, bold + italics
- training sets containing:
  - all selected 6 fonts
  - 3 sans serif fonts
  - 3 serif fonts
- 1 rendering method: Photoshop

We have evaluted two series of experiments as follows:

- The objective of these tests was to gain CRRs when the character models were trained with all 6 fonts for point size 8 but each time for one font style. Using 4 font styles, we gained 4 different character models as training sets. We chose the following strategy for our tests:
  We mixed the patterns of all 6 fonts (serif, sans serif) for each style, and then randomly generated 1/3 of them as training sets and 2/3 of them for test. Table 5.9 shows the results for testing with Photoshop patterns. These results were rather not convincing.

Table 5.9: Multi-font exp.; Training set: all 6 fonts

	plain		bold		italics		bold + italics		Mean
	#	%	#	%	#	%	#	%	%
Arial	135	89.62	58	95.54	152	88.31	134	89.62	90.79
Tahoma	146	88.77	> 99.9	92.31	137	89.46	108	91.69	90.56
Verdana	162	87.54	102	92.15	162	87.54	94	92.77	90
Century	122	90.62	108	91.69	228	82.46	199	64.69	87.37
Garamond	186	85.69	156	88	225	82.69	370	71.54	81.98
Georgia	142	89.08	78	94	186	85.69	114	91.23	90
Mean in %	-	88.55	-	92.28	-	86.03	-	86.94	88.45

- The previous experiments did not deliver satisfactory results when all fonts were trained to model the characters. Therefore we decided to group sans serif and serif fonts together and train two character models each with 4 different styles. We chose the same test strategy as the previous experiments and observed much better CRR than in the former experiments. Table 5.11 shows the CRRs for serif fonts when recognized with a serif classifier. Table 5.10 illustrates the CRRs for sans serif fonts recognized with the sans serif classifier. We gain an overall CRR of 98.99% for the sans serif classifier and of 97.53% for the serif classifier.

Table 5.10: Multi-font exp.; Training set: 3 sans serif fonts

	plain		bold		italics		bold + italics		Mean
	#	%	#	%	#	%	#	%	%
Arial	60	95.38	48	96.31	41	96.85	48	97.23	96.44
Tahoma	46	96.46	41	96.85	8	99.38	12	99.08	97.94
Verdana	36	97.23	54	95.85	4	99.69	0	> 99.9	98.19
Mean in %	-	96.39	-	96.33	-	98.64	-	98.77	97.53

Table 5.11: Multi-font exp.; Training set: 3 serif fonts

	plain		bold		italics		bold + italics		Mean
	#	%	#	%	#	%	#	%	%
Century	61	95.31	3	99.77	81	93.77	42	96.77	96.4
Garamond	80	93.85	10	99.23	125	90.38	115	91.15	93.65
Georgia	52	96	8	99.38	50	96.15	17	98.69	97.56
Mean in %	-	95.05	-	99.46	-	93.44	-	95.54	95.87

The overall CRR for both group of classifiers can be seen at table 5.12. We can see from table 5.12 that a classifier that is trained with serif fonts or sans serif fonts delivers better results than a classifier that is trained with all selected fonts. Therefore, in order to achieve better CRR, we still would need the a priori knowledge that the font is serif or sans serif.

Table 5.12: Overall recognition rates of isolated charcters using the multi-font classifier

	plain	bold	italics	bold + italics
6 fonts	86.87%	92.14%	83.35%	85.96%
sans serif	97.6%	97.45%	98.92%	99.06%
serif	97.01%	99.58%	95.92%	97.19%

5.3.5 Conclusion

This section presented the results of evaluation tests on single isolated characters that do not contain contextual noise. The objective for choosing such training and test environment was to determine the discrimination power of the selected features by means of estimation of mono-Gaussian multivariate density functions to model the feature's distribution.
It has been shown that central moments are fairly discriminative. We can conclude that they are able to convey sufficient information from character shapes in ultra low resolution, anti-aliased character images. Additionally, it could be shown that character discrimination performs well for font sizes down to 5 points. In addition, it was shown that classifiers can be trained independently of the applied anti-alaising rendering methods. Finally the possibility of clustering the serif and sans serif fonts as classifiers still delivered reliable CRR. The gained study shows that if the characters are *isolated*, they can be recognized very accurately. In the next section, we report on additional evaluation tests performed on contextual characters that were extracted from words.

5.4 Experiments on contextual characters

The previous evaluation tests showed that central moments are able to discriminate single isolated characters by the means of estimation of features' distribution with mono-Gaussian multivariate density functions. Another outcome of the previous study was that recognition of uppercase letters was more accurate than of lower case letters. This is due to the fact that upper case letters have a larger and more distinguishable shapes than the lower case letters. For example a confusion between 'i' & 'l' is more likely than between 'I' and 'L'. there are also other pairs of lower case letters that exhibit the same form of confusion for ex. confusion between 't' and 'f' vs. 'T' & 'F' etc. Thus, for the study of contextual characters, we decided to perform evaluation tests only on the lower case character samples. In addition, for these experiments, we added the *relative baseline* as a new feature, since we observed errors resulting from confusions between characters that were similar in shape but their stems were above or below the baseline. This was for example the case for example between *'p'* & *'b'* or *'q'* & *'d'* etc..

We have evaluated two groups of experiments for the recognition of contextual characters: 1)mono-font recognition and 2)multi-font recognition. As we needed a database for running our experiments, we developed a method to gain contextual characters from words. Below, we will describe our segmentation method first, followed by an explanation of how we built our training

set. Finally we will report about the results of our evaluation tests.

5.4.1 A method to obtain contextual characters

In order to build a database containing contextual characters, we first produced a synthetic database of 520 English words chosen from a real word dictionary containing 120'000 words. Then we used our method described in chapter 3 to segment words into contextual characters. Additionally, the occurrences of each character in our database was proportional to its frequency of a real world dictionary.

Extracting contextual characters

To obtain *contextual characters*, we had to overcome the segmentation problem in a ULR-word. Fig. 5.9 shows the word *bank* from our ULR-word database. As it can be seen, the characters in this word are connected. We developed a segmentation method based on a-priori knoweldge of font metrics information to isolate the contextual characters. To use font metrics information, the character string of each word had to be known.

Figure 5.9: The ULR image of the word *bank* in point size 9

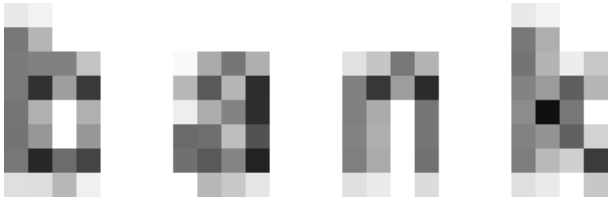


Figure 5.10: The segmented characters of word *bank* in point size 9

Our algorithm used the following information:

- *the reference point for each character: p*; where p is defined as follows:
  $p = x_i - LSB$, where x_i is the lower-leftmost-x-coordinate of character's bounding box and LSB is the character's left side bearing (see chapter 3 for more details)
- *The width of bounding box for each character: w*
- *A shift value s to skip the noisy zone between the adjacent characters and adjust the rounding error that is applied to the down-sampled image due to its transformation to sub-pixel domain*

Our algorithm can be written in pseudo-code as below:

- `Get the big font size:` n
- `Get the bounding box for each character in big font size:` r
- `Get the lower leftmost x coordinate of each bounding box in big font size:` x_{old}
- `Get the width of the bounding box in big font size:` w
- `Add some additional pixels (=` s`) to the width to catch up with rounding correction when rendering at small point size`

$$x_{\text{new}} = x_{old} + s \tag{5.13}$$

$$w_{\text{new}} = w_{old} + s \tag{5.14}$$

- `Define down-sampling factor of` k `to obtain ULR-patterns in a small size of` $\frac{n}{k}$ `points.`
- `calculate the reference point and width of the bounding box for the resampled pattern as follows:`

$$x_{small} = \frac{x_{\text{new}}}{k} \tag{5.15}$$

$$w_{small} = \frac{x + w_{\text{new}}}{k} - x_{small} \tag{5.16}$$

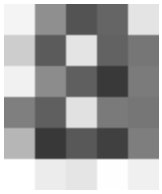
Fig. 5.10 shows the segmented characters $\{b, a, n, k\}$ from the word *bank* segmented with our method.

Building training sets

To have a training set with sufficient sample variability, we have decided to build a set containing the following samples:

- The segmented character images from our word data base by using our segmentation method
- The images of the single isolated characters
- Images of single isolated characters cut on their left and right sides, see also fig. 5.4.1 so that their new width was:

$$width_{\text{new}} = width_{original} - 1 \tag{5.17}$$



(a) Character 'a' original

(b) Character 'a' cut one pixel column at right side

(c) Character 'a' cut one pixel column at left side

Figure 5.11: Three versions of 'a' contained in the training set of contextual characters

5.4.2 Mono-font experiment

We used the identification system shown at Fig. 5.4. We conducted our experiments as follows: First the unknown ULR word was segmented into single contextual characters using our previously described segmentation method. Next, under the assumption that the segmented characters were unknown, we passed them through our character identification system. Then, the system calculated the maximum likelihood of unknown characters for each known class and determined the most probable character class. Our tests were performed using the following font combinations:

- 3 sans serif fonts: Arial, Tahoma and Verdana
- 3 serif fonts: Century, Georgia and Times
- 5 font sizes: 8-12 points
- 4 font styles: plain, bold, italics, bold + italics

The above combination led to 120 different fonts.

Table 5.12 shows the gained results in the form of number and percentage of errors for all the selected fonts. We can observe from the results that *bold* style increases the recognition rate both for *plain (roman)* and *italics* fonts as is illustrated in Fig. 5.13. Furthermore, the results of the *italics* fonts are worse than the ones of *roman* fonts. The reason being that we used rectangular segmentation windows, which are probably not appropriate for *italics* fonts. Improvements should be obtained with slanted windows. Furthermore, table 5.13 shows the overall recognition rates of selected fonts for different font sizes.

Table 5.13: Mono-font exp.: Overall recognition rates of contextual characters for different font sizes

Font size	12	11	10	9	8
	99.03%	98.81%	98.14%	97.34%	96.12%

	Point size	plain		bold		italics		bold + italics	
		#	%	#	%	#	%	#	%
Century	**8**	181	8.16	80	3.61	260	11.72	30	1.35
	9	125	5.64	67	3.02	193	8.70	15	0.68
	10	86	3.88	79	3.56	201	9.06	21	0.95
	11	5	0.23	2	0.09	55	2.48	13	0.59
	12	2	0.09	28	1.26	81	3.65	6	0.27
Mean			**3.60**		**2.31**		**7.12**		**0.77**
Georgia	**8**	29	1.31	42	1.89	76	3.43	144	6.49
	9	40	1.80	30	1.35	54	2.43	22	0.99
	10	10	0.45	11	0.50	30	1.35	12	0.54
	11	16	0.72	14	0.63	41	1.85	16	0.72
	12	7	0.32	6	0.27	29	1.31	30	1.35
Mean			**0.92**		**0.93**		**2.07**		**2.02**
Times	**8**	160	7.21	64	2.89	305	13.75	175	7.89
	9	125	5.64	9	0.41	196	8.84	80	3.61
	10	58	2.61	1	0.05	96	4.33	63	2.84
	11	13	0.59	6	0.27	115	5.18	38	1.71
	12	5	0.23	6	0.27	62	2.80	43	1.94
Mean			**3.26**		**0.78**		**6.98**		**3.60**
Arial	**8**	75	3.38	42	1.89	67	3.02	72	3.25
	9	43	1.94	54	2.43	78	3.52	111	5.00
	10	29	1.31	12	0.54	75	3.38	30	1.35
	11	25	1.13	6	0.27	20	0.90	6	0.27
	12	41	1.85	14	0.63	9	0.41	1	0.05
Mean			**1.92**		**1.15**		**2.25**		**1.98**
Tahoma	**8**	33	1.49	23	1.04	73	3.29	39	1.76
	9	34	1.53	22	0.99	21	0.95	25	1.13
	10	4	0.18	22	0.99	25	1.13	17	0.77
	11	1	0.05	46	2.07	16	0.72	9	0.41
	12	52	2.34	13	0.59	7	0.32	1	0.05
Mean			**1.12**		**1.14**		**1.28**		**0.82**
Verdana	**8**	29	1.31	6	0.27	37	1.67	22	0.99
	9	23	1.04	30	1.35	11	0.50	8	0.36
	10	97	4.37	10	0.45	2	0.09	1	0.05
	11	102	4.60	60	2.71	8	0.36	3	0.14
	12	43	1.94	27	1.22	3	0.14	0	0.00
Mean			**2.65**		**1.20**		**0.55**		**0.31**

Figure 5.12: Mono-font exp.: Number and percentage of errors of contextual characters for different fonts

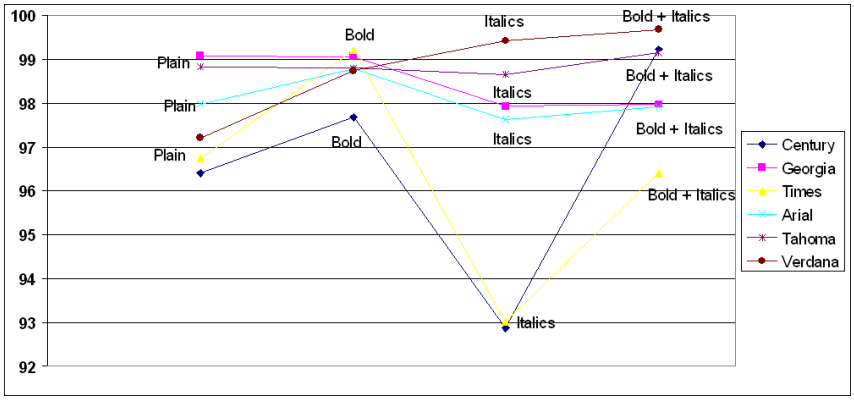
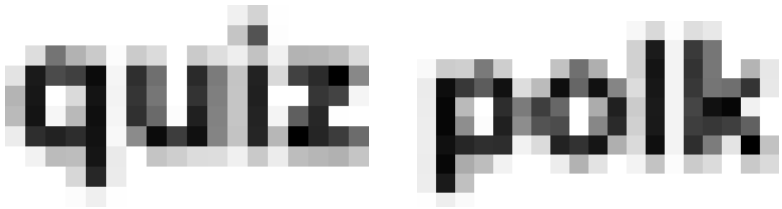


Figure 5.13: Mono-font exp.: Recognition rates of contextual characters for different fonts

We analyzed the errors and observed that a considerable number of mis-recognition errors belonged to two main groups:

1. Confusion between *'i'* and *'l'*. Fig. 5.14 shows words *'quiz'* and *'polk'* that contain the

letters *'i'* and *'l'*. As can be seen, only one or two pixels have a different gray value, whereas the majority of the pixels of both letters have similar gray values. Such confusion led to a considerable amount of mis-recognition especially by sans serif fonts. This error can also be seen by serif fonts at small point sizes where the decorative 'serif' is represented by only one pixel.



(a) The word *quiz* (b) The word *polk*

Figure 5.14: Example of misconfusion between i and l in two ULR-words

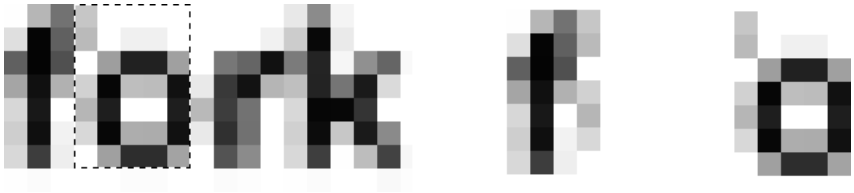


Figure 5.15: Example of misrecognition when 'f' is the precedence letter

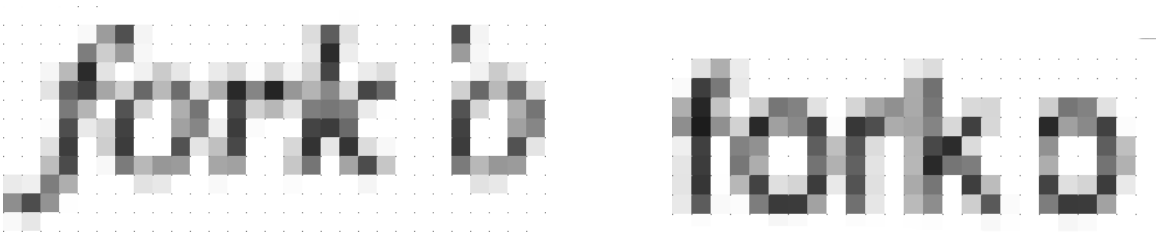
2. Mis-recognition when the precedence character is a 'f'. The character 'f' has a negative RSB (right side bearing), i.e. the upper most left side of its bounding box overlaps with the bounding box of the adjacent character at its right side. This adds one or more additional *'noisy'* pixel at the upper left side of the bounding box of the adjacent character. When segmenting the characters with our segmentation method, it so happens that these *'noisy'* pixels lead to confusion errors. Fig. 5.15 illustrates this for the word *fork*. The character *'f'* is the preceding letter of the character *'o'* which deteroriates the shape of *'o'* and leads to a mis-recognition of *'o'* as the character *'b'*.

Omitting these errors led to higher CRR, as can be seen in table 5.16. The improvement to recognition rates is between 2-3%.

Fig. 5.18 shows the results for different font styles. As already seen in Fig. 5.13 the recognition of *'italics'* fonts were worse than *'roman'* fonts. However, the results for *'italics'* fonts after correction of previously described errors, that are shown in Fig. 5.18, were better than *'roman'* fonts. The reason can be explained as follows: first the majority of errors found for the *'italics'* fonts were when *'f'* was the precedence letter in a word. Second *'italics'* fonts are more slanted and thus have a wider bounding box. As a result more pixels from both adjacent characters are in the overlapped zone as Fig. 5.17 illustrates.

	Point size	plain		bold		italics		bold + italics	
		#	%	#	%	#	%	#	%
Century	**8**	89	4.01	64	2.89	73	3.29	11	0.50
	9	83	3.74	65	2.93	8	0.36	7	0.32
	10	34	1.53	79	3.56	0	0.00	13	0.59
	11	5	0.23	2	0.09	30	1.35	1	0.05
	12	0	0.00	25	1.13	16	0.72	1	0.05
Mean			**1.90**		**2.12**		**1.15**		**0.30**
Georgia	**8**	3	0.14	7	0.32	5	0.23	17	0.77
	9	24	1.08	4	0.18	12	0.54	4	0.18
	10	6	0.27	10	0.45	5	0.23	3	0.14
	11	11	0.50	5	0.23	8	0.36	1	0.05
	12	4	0.18	0	0.00	8	0.36	5	0.23
Mean			**0.43**		**0.23**		**0.34**		**0.27**
Times	**8**	44	1.98	16	0.72	22	0.99	12	0.54
	9	23	1.04	1	0.05	11	0.50	12	0.54
	10	10	0.45	1	0.05	5	0.23	10	0.45
	11	4	0.18	5	0.23	18	0.81	4	0.18
	12	3	0.14	5	0.23	6	0.27	11	0.50
Mean			**0.76**		**0.25**		**0.56**		**0.44**
Arial	**8**	67	3.02	40	1.80	16	0.72	9	0.41
	9	40	1.80	54	2.43	1	0.05	3	0.14
	10	27	1.22	11	0.50	0	0.00	0	0.00
	11	25	1.13	6	0.27	11	0.50	3	0.14
	12	41	1.85	12	0.54	1	0.05	0	0.00
Mean			**1.80**		**1.11**		**0.26**		**0.14**
Tahoma	**8**	19	0.86	0	0.00	11	0.50	8	0.36
	9	15	0.68	13	0.59	8	0.36	2	0.09
	10	11	0.50	0	0.00	27	1.22	43	1.94
	11	26	1.17	3	0.14	18	0.81	6	0.27
	12	6	0.27	50	2.25	0	0.00	4	0.18
Mean			**0.69**		**0.60**		**0.58**		**0.57**
Verdana	**8**	28	1.26	6	0.27	14	0.63	0	0.00
	9	23	1.04	30	1.35	8	0.36	0	0.00
	10	96	4.33	8	0.36	0	0.00	0	0.00
	11	102	4.60	55	2.48	4	0.18	0	0.00
	12	43	1.94	27	1.22	0	0.00	0	0.00
Mean			**2.63**		**1.14**		**0.23**		**0.00**

Figure 5.16: Mono-font exp.: Number and percentage of errors of contextual characters after error correction for different fonts



(a) italics style (b) Roman style

Figure 5.17: Segmentation of *'f'* in word *'fork'*

According to the above observations, the font *Verdana* has specific characteristics as it has the least number of errors whereas other fonts show higher recognition rates when the previously described errors were not considered. Table 5.16 demonstrates this fact. Therefore, we can conclude that the font *Verdana*, amongst the selected fonts, is the most legible font for our computerized character recognition algorithm.

Table 5.14 shows the overall recognition rates of all selected fonts for different font sizes after error correction.

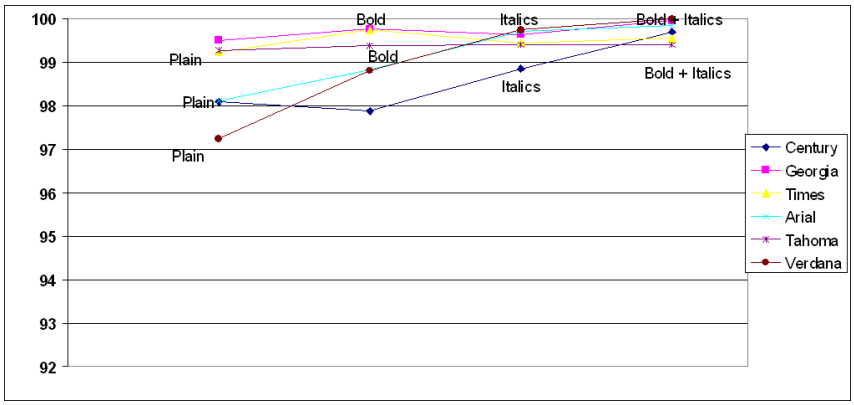


Figure 5.18: Mono-font exp.: Corrected recognition rates of contextual characters for different fonts

Table 5.14: Mono-font exp.: Overall recognitin rates of contextual characters for different font sizes

Font size	12	11	10	9	8
	99.17%	98.79%	98.73%	98.25%	97.26%

5.4.3 Multi-font experiment

Another interesting question for the identification of contextual characters was to assess the performance of our identification system as a *multi-font* recognizer. According to the results of the experiments performed on single isolated characters, we decided to build classifiers that rely on the a-priori knowledge whether the character to be recognized is serif or sans serif. Therefore we needed to feed the system with two training sets: one with serif and another with sans serif characters and the task was to recognize an unknown serif or sans serif character image. Our tests were performed using the following font combinations:

- 3 sans serif fonts: Arial, Tahoma and Verdana
- 3 serif fonts: Century, Georgia and Times
- 5 font sizes: 8-12 points
- 4 font styles: plain, bold, italics, bold + italics

	Point size	plain		bold		italics		bold + italics	
		#	%	#	%	#	%	#	%
Arial	**8**	197	9.30	67	3.16	234	11.05	161	7.60
	9	207	9.77	71	3.35	71	3.35	240	11.33
	10	145	6.85	57	2.69	83	3.92	58	2.74
	11	32	1.51	37	1.75	41	1.94	34	1.61
	12	139	6.56	29	1.37	26	1.23	14	0.66
Mean			**6.80**		**2.46**		**4.30**		**4.79**
Tahoma	**8**	181	8.55	92	4.34	308	14.54	139	6.56
	9	148	6.99	15	0.71	100	4.72	93	4.39
	10	28	1.32	20	0.94	100	4.72	112	5.29
	11	12	0.57	58	2.74	29	1.37	56	2.64
	12	11	0.52	29	1.37	100	4.72	47	2.22
Mean			**3.59**		**2.02**		**6.02**		**4.22**
Verdana	**8**	140	6.61	67	3.16	345	16.29	8	0.38
	9	113	5.34	14	0.66	85	4.01	9	0.42
	10	8	0.38	20	0.94	116	5.48	10	0.47
	11	3	0.14	50	2.36	52	2.46	11	0.52
	12	2	0.09	41	1.94	49	2.31	12	0.57
Mean			**2.51**		**1.81**		**6.11**		**0.47**

Figure 5.19: Multi-font exp.: Number and percentage of errors of contextual characters using a sans serif classifier

	Point size	plain		bold		italics		bold + italics	
		#	%	#	%	#	%	#	%
Century	**8**	172	8.12	102	4.82	158	7.46	196	9.25
	9	156	7.37	106	5.00	173	8.17	51	2.41
	10	122	5.76	16	0.76	176	8.31	59	2.79
	11	29	1.37	2	0.09	146	6.89	65	3.07
	12	61	2.88	29	1.37	175	8.26	31	1.46
Mean			**5.10**		**2.41**		**7.82**		**3.80**
Georgia	**8**	0	0.00	125	5.90	130	6.14	117	5.52
	9	41	1.94	37	1.75	133	6.28	79	3.73
	10	38	1.79	41	1.94	109	5.15	49	2.31
	11	18	0.85	24	1.13	105	4.96	44	2.08
	12	4	0.19	32	1.51	68	3.21	0	0.00
Mean			**0.95**		**2.45**		**5.15**		**2.73**
Times	**8**	155	7.32	77	3.64	247	11.66	219	10.34
	9	101	4.77	204	9.63	169	7.98	174	8.22
	10	17	0.80	74	3.49	162	7.65	230	10.86
	11	20	0.94	48	2.27	140	6.61	137	6.47
	12	40	1.89	66	3.12	143	6.75	137	6.47
Mean			**3.14**		**4.43**		**8.13**		**8.47**

Figure 5.20: Multi-font exp.: Number and percentage of errors of contextual characters using a serif classifier

Table 5.19 shows the number and percentage of identification errors when our system used as a sans-serif recognizer. Table 5.20 shows the results of using the system as a serif recognizer.

We grouped the results for both classifiers according to different font style and point sizes. The results are listed in table 5.15 to table 5.18. When looking at sans serif classifier results in table 5.15 and table 5.16, we can see that font *Verdana* shows the highest overall recognition rates amongst sans serif fonts. We can additionally see from serif classifier results shown in table 5.17 and table 5.18 that the font *Georgia* has the highest overall recognition rates amongst serif fonts.

We studied more deeply the improvement history of legible fonts developed by Microsoft inc. [110]. Concerning sans serif fonts, we noticed that *Tahoma* has also been designed for computer screens but *Verdana* has been designed later as an extension of *Tahoma* to be more legible. The most important extension of *Verdana* is that it has extra space between characters so they don't *touch* each other (in our terminology: not connected). The bold style is quite bold, ensuring that there is a difference between bold and roman, yet the bold characters never fill-in, even at small sizes (we can still read it at 4 point, at least under Windows). Additionally, special care has been taken with letters like 'I', 'l', 'i' and 'J' so that they aren't confused as follows: The lowercase 'i' is slightly shorter than the lowercase 'l', which also makes them more distinct. Letter combinations such as 'fi' and 'fl' and 'ff' are designed so that they clearly do not *touch* each other and therefore are more legible. These special considerations are in accordance with our error analysis that we described above. As previously stated, our experiments showed that *Georgia* as a serif font is more legible amongst other selected serif fonts for our machine learning algorithm. The reason could be that this font has been designed to be a serif font with fairly clear characters at 8-12 points. Additionally, its x-height is larger than *Times*, but not as large as *Verdana*'s. Furthermore, its italics is fluid, and its bold is ultra-bold similar to bold of *Verdana*.

Table 5.15: Multi-font exp.: Recognition rates of contextual characters using a sans serif classifier for different font styles

	Plain	Bold	italics	Bold + italics
Arial	93.2	97.54	95.7	95.21
Tahoma	96.41	97.98	93.98	95.78
Verdana	97.49	98.19	93.89	99.53

Table 5.16: Multi-font exp.: Recognition rates of contextual characters using a sans serif classifier for different font sizes

	Arial	Tahoma	Verdana
8	92.22	91.50	93.39
9	93.05	95.8	97.39
10	95.95	96.93	98.18
11	98.3	98.17	98.63
12	97.54	97.79	98.77

5.4.4 Conclusion

The evaluation tests regarding the recognition of contextual characters indicated that:

1. Central moments are fairly discriminative features

Table 5.17: Multi-font exp.: Recognition rates of contextual characters using a serif classifier for different font styles

	Plain	Bold	italics	Bold + italics
Century	94.9	97.59	92.18	90.75
Georgia	99.05	97.55	94.85	97.27
Times	96.86	95.57	91.87	91.53

Table 5.18: Multi-font exp.: Recognition rates of contextual characters using a serif classifier for different font sizes

	Century	Georgia	Times
8	92.59	95.61	91.76
9	94.26	96.58	92.35
10	95.6	97.2	94.3
11	97.14	97.75	95.93
12	96.51	98.77	95.44

2. Recognition accuracy of single isolated characters is higher (above 99.9%) than contextual characters (98.45%).

3. Contextual noise between adjacent characters deteriorates the recognition accuracy. Therefore, we decided to define an extra character to represent contextual noise. We named this extra character the *inter-character*.

4. Multivariate mono-Gaussian density functions are insufficient to represent the distribution of features of contextual characters. Thus, we decided to use mixtures of Gaussian models (GMMs) to estimate the probability density functions of our features.

5. The segmentation of single words can not be solved using the classical segmentation methods in pattern analysis. Therefore, to develop an automatic single word recognizer, we can choose one of the following approaches:

   - Either we develop methods to pre-segment the characters in an ULR-word and then recognize them using a classical segmentation-first-classifier
   - Or we would make use of a classification method that is able to simultaneously segment and recognize connected characters in an ULR word

5.5 Discussion and further conclusions

In this chapter, we presented a study of single anti-aliased characters with a resolution of 72-100 dpi with font sizes between 3-12 points. We reported on the performance of the selected

feature extraction method and showed that it works very accurately for isolated characters and delivers fairly good results for contextual characters. Therefore, we can integrate this method in a recognition system for words. Additionally, we provided a thorough study on the mutual influence of adjacent characters at their left and right borders. We defined this mutual influence as *contextual noise*. In the experiments of contextual characters we showed that we were able to get good recognition results when we skipped this *noisy zone*.

Moreover, we can state that the system used to obtain the results in this chapter is not realistic as it intrinsically implies that the segmentation of characters inside the word is known. A direct observation of the ULR-word "bank" in this chapter can convince us that most of the well-known pre-segmentation methods used in classical OCR systems [39] are not applicable to segment connected characters in such a word. As already argued in the previous section, a possible way to resolve this problem would be to develop specific segmentation methods for such characters and use, for instance, Bayesian classifier for the classification task. However, we strived for a statistical method capable of segmenting and recognizing connected characters at the same time. Hidden Markov models (HMMs) are highly capable of resolving this task. Hence, we opted to build a word recognizer using HMMs. In addition, HMMs are also capable of performing automatic training when no a-priori knowledge of font metrics and character segmentation is needed. Furthermore, HHMs are also well suited to include linguistic knowledge such as probabilities of character and word sequences, which will allow increased recognition rates of ULR words. Next chapter reports on using HMMs to build a word recognizer that is able to deal with the specificities of the ULR, antialiased word images, with font sizes between 6-12 points.

Chapter 6

Recognition of ULR single words using HMMs

6.1 Introduction

The experiments in the prior chapter have shown that we have to design a dedictaed recognizer based on a statistical approach capable of dealing with the recognition problem of connected characters in an ULR word. As hidden Markov models have shown outstanding ability to simultaneously segment and recognize for example the connected characters in handwriting recognition or connected speech phonemes in speech recognition, we opted to build our *single ULR word recognizer* using HMMs.
important outcome of the experiments of the previous chapter has been that the central moments as selected features are reliable enough to discriminate the different character classes and can be hence integrated in our *single word recognizer*. Central moments also present the property of being translation invariant. This property is interesting in the case of recognition of isolated words using HMMS where a sliding window must be used to extract the features. In addition, the comparison of the experiments with contextual characters with the ones using isolated characters has shown that the contextual noise at both end sides of the characters significantly deteroriates the recognition results. Knowing this, we aimed at building HMM topologies for word models that have been composed from the characters and additionally incorporate the specificities of contextual noise due to the close proximity of adjacent characters. Therefore, we opted to build a new extra character model between the adjacent characters. We named it *inter-character* and developed several methods to infer it from single words that we will explain in this chapter.
We have built several HMM-based recognizers each with different objectives. The first objective was to measure how the performance of the recognition system would evolve when using a HMM word topology that would contain the *inter-character* model. The second objective was to investigate the performance of the recognizer using different HMM topologies. Another

interesting aspect was to enlarge the database both for training and test in order to find out the system limitations. We also investigated the possibility of fully automated training, i.e. using a training procedure without the need of manually segmenting a bootstrap set to initialize the values of the HMM parameters. Furtermore we were interested to know the system's performance when the dedictated recognizer was relying on a large dictionary. Since the experiments of prior chapter cleary have shown the limitations of mono-Gaussian probability density function to model the feature's distribution, we estimated the feature's distribution for our word recognizer with GMMs. Therefore, we were interested to find a balanced trade off between the number of Gaussian components and the size of training set. Obviously as number of Gaussian components increases, more training data is needed to reasonably model the mixture of Gaussian models. Finally we were interested on designing an open vocabulary recognizer that would recognize any arbitrary word written in Latin Alphabet in any language. Our motivation for such an approach was twofold: building a multi-lingual system and removing the limitations in terms of vocabulary size. This final system also proved itself to be very efficient in terms of cpu and memory footprint as all linguistic constraints were removed from the HMM topology. In the following we describe the experimental set ups and the results obtained when investigating the above objectives.

6.2 Fundamental experiments

In this stage we have aimed at measuring the impact of the following components on the system's performance

- Impact of injecting *inter-character* into the word models
- Impact of different HMM topologies
- Impact of using a full covariance matrix vs. a diagonal covariance matrix for the estimation of states' emission probabilities, under the assumption that features would follow a mono-Guassian normal distribution

6.2.1 System description

As is described below, we have built our recognizer using a sliding-window feature extraction feeding a HMM based classifier. This combination has the clear advantage that the segmentation and recognition problem can be solved simultaneously. However, for this first set of experiments, the training was employed using already segmented contextual characters. In this stage, we used the automatic HMM-training to obtain the optimal values of emission probabilities for already segmented characters. In other words, we did not infer the character segmentation at training time, as we supposed that characters were pre-segmented using our knowledge-based segmentation method as described in chapter 5.

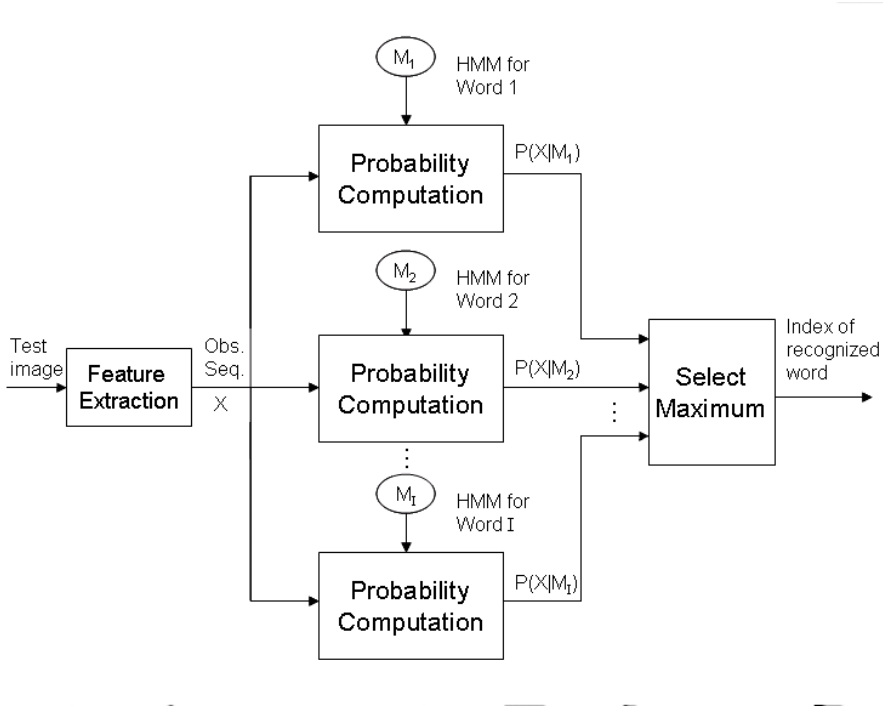


Figure 6.1: Single word Recognizer

The chosen features for training and test were the first and second order central moments plus an additional feature computed from the difference between the baseline and the y coordinate of the gravity center of each analysis window. This last feature is actually optimistically computed as the baseline is here assumed to be correctly estimated. The font we used to produce our training and test set is *Verdana* due to its high legibility as shown in the previous chapter.

As already described, at training time we laid a sliding window on top of each character and claculated the chosen features for each sliding window. These feature sets were used to obtain the optimal emission probability using the EM-algorithm described in chapter 4.

At testing time the sliding window was employed at the top of the whole unknown word. Our System is illustrated in Fig. 6.1. Generally speaking, the image of an unknown word was fed to the system, where the recognizer compared the competing probabilities of the unknown image being each word in the system dictionary. The word model having the greatest probability value among all word models in the dictionary was then selected as the recognized word.

The recognition system was built of different components that are explained below:

The inter-character model

The experiments on contextual characters showed the importance to model the contextual noise between connected characters as an extra character in ULR words. We named this extra character *inter-character* and labeled it in our report as *'#'*. In this stage, we used the font metrics information about the width of the bounding box of each character to separate the characters

from the inter-characters in training database. For this reason we developed a simple algorithm written in pseudo-code as follows:

- `Get the next character from the word`
- `Determine the width of bounding box of the character =` w
- `Determine two different widths for the character:` $w_1 = w$ `and` $w_2 = w - 1$
- `Cut two 2-pixel rectangles between the adjacent characters. The leftmost` x `coordinate of the rectangles are calculated as:`
  - $x_1 = w_1$
  - $x_2 = w_2$

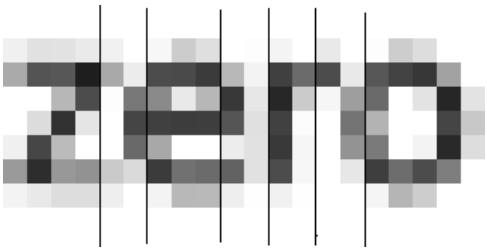


Figure 6.2: Example of extracting *inter-character* from a single ULR-word

Training configuration

We used a training database containing 750 ULR words (the same words as the experiment on contextual characters). As a by-product of applying the described method to gain inter-character, we could isolate the contextual characters. We laid a sliding window at the top of each character as shown in Fig. 6.8 and extracted the corresponding features. The ULR words were rendered with Adobe Photoshop. As shown in chapter 5, the corresponding extracted features follow a normal distribution. Therefore we calculated the mean vector and covariance matrix for each class that were used for the estimation of character states' emission probabilities.

Test configuration

We used two different topologies for recognition of unknown words. We also used two different inter-character models. Below we give a description of the chosen topologies:

- **Inter-character topology** The availability of the *inter-character* is dependent to the font family and especially to the font size. The smaller is the size the less probable it is that we can cut a 2-pixel window between the adjacent characters. However, we can distinguish the following cases:

1. Inter-character is *not* or *once* available, i.e. inter-character is modeled using 0 or 1 sliding windows as shown in fig. 6.3. This is usually the case using small font sizes(< 9 pts).

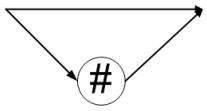


Figure 6.3: *Inter-character* topology for small font sizes

2. inter-character is *once* or *twice* available, i.e. 1 or 2 sliding windows model the inter-character as shown in fig. 6.4. This is usually the case using font sizes between 9-12 pts.

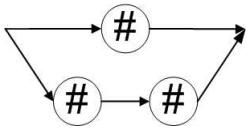


Figure 6.4: *Inter-character* topology for font sizes between 9-12 pts

- **Chosen topologies**

  – *Simple Left-Right*

  This topology was chosen for its simplicity and was, for us, the first configuration to investigate. Fig. 6.5 and Fig. 6.6 show the modified simple left-right topology which were introduced in chapter 4 using *inter-character* models.

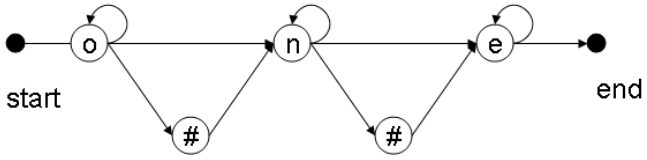


Figure 6.5: Example of simple left-right topology using 0/1 inter-characters

  – *Left-Right minimum width*

  At testing time we analyzed some misrecognized results. Doing this, we inspected some state sequences obtained on mis-recognized words. An example of state sequence for word 'two' obtained with simple left-right topology is reproduced below:

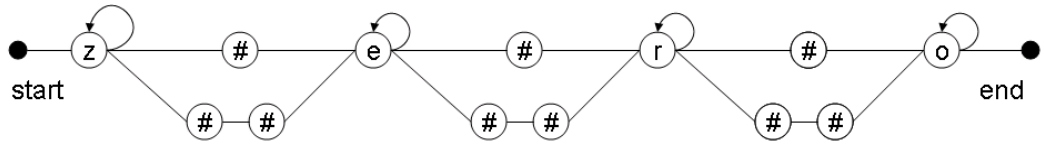


Figure 6.6: Example of simple left-right topology using 1/2 inter-characters

* Genuine word= two
* Recognized word = one
* Best state sequence for word one: o o o o o o o o o o n n e e e e
* Best state sequence for word two: t t t t w w w w o o o o o o o o
* Ground truth state sequence for word two: t t t t w w w w w w w o o o o o

What is happening can be explained as follows. By nature, the HMM is trying to maximize its likelihood. It will then associate few observations to states that gives low emission probabilities while it will spend more observations in states giving high emission probabilities. The state sequence for the winning word 'one' is clearly spending only two observations in character model 'n'. Character 'n' is of course longer than 2 pixels (actually 3 pixels considering the width of the analysis window) in our images. In order to avoid phenomena as the one described above, one can introduce so-called minimum width topologies. This topology is simply obtained by repeating a state a given number of time obliging the Viterbi algorithm to spend a minimal amount of observations in the same category of character while the last character state has the possibility to be repeated for an unlimited amount of time. Fig. 6.7 illustrates such a topology for word 'two'. In our configuration, the minimum width values have been obtained from the a priori known font metric information. We have to underline that such minimum width values are dependent to a given font, leading to a system tuned for a specific font.

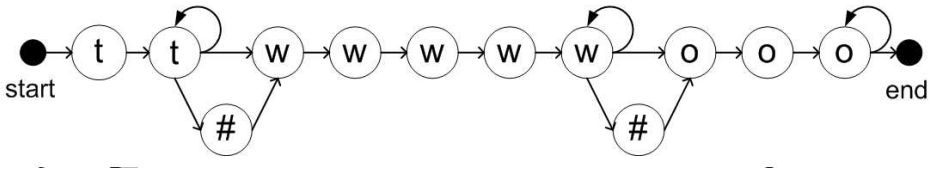


Figure 6.7: Example of minimal width topology using 0/1 inter-characters

The data base that was used as test set contained the same 500 ULR words as the experiments on contextual characters. We chose this limited size of dictionary for these first series of experiments to be able to answer the different questions we raised at the beginning of this chapter. Doing this, we aimed at being able to compare the recognition rates from the HMM-based system with the ones of the previous study on contextual characters. If the new system would be able to segment and recognize the words simultanouesly and deliver satisfactory recognition results,

then we would have the confirmation of having chosen the appropriate classification algorithm for our purpose.
As explained in previous chapters, the characters in an ULR-word are connected and therefore can not be isolated. Thus we decided to lay a sliding window at the top of each unknown word image that is 2 pixel wide shifting one pixel to the right of the word as shown in Fig. 6.8. The features were then extracted for each window and were fed to the recognition system. Words were built using the chosen HMM-topologies at Fig. 6.5 and Fig. 6.7. The recognition system used then the Viterbi algorithm to find out the most probable word among all the competing words existing in the system dictionary.

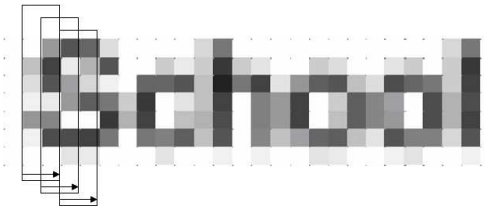


Figure 6.8: Example of sliding window with width = 2 pixels and shift= 1 pixel

6.2.2 Evaluation tests

According to the topologies introduced above, different evaluation tests were performed. We also experimented with two implementations of the Gaussian probability density functions used to estimate the emission probabilities. The first one is using the regular *full* covariance matrix. The second one used a simplified *diagonal* covariance matrix, making the extra assumption that the components of the feature vectors are de-correlated. While this assumption is potentially too restrictive, it allows a much faster computation of the emission probabilities, where the computation of the inverse of the covariance matrix is involved. We have chosen a dictionary with a limited size of 500 words for these evaluation test in order to be able to see the impact of the inter-character, different HMM-topologies, size of training set and choice of covariance matrix on the recognition rate of test words without having extra errors that would come from the impact of a large dictionary size. Notwithstanding, we report in the next section about the impact of dictionary size on recognition rates. The results of our tests are presented at table 6.1.

Table 6.1: WRR for font Verdana, plain, 9 points; training size: 750 words

without inter-character	diag. covariance	full covariance
Simple L-R-HMM	52%	56%
L-R-Min-Dur-HMM	60%	79%
with inter-character	diag. covariance	full covariance
Simple L-R-HMM	$> 99\%$	$> 99\%$
L-R-Min-Dur-HMM	$> 99\%$	$> 99\%$

6.2.3 Conclusion

Looking at the results presented at table 6.1, we can make the following conclusions:

- As expected, we see clearly an improvement coming from the introduction of minimum width topology.
- Using full covariance matrices leads to slightly better results than with diagonal covariance matrices. Therefore, for the sake of a fast computation, we will use in our further experiments the diagonal covariance matrix.
- The introduction of the inter-character model into the HMM topologies has increased the recognition rates about 30% which is a significant improvement.

These encouraging results indicate clearly that HMMs are strongly suitable to simultaneously segment and recognize ultra low resolution words, provided that the right topologies and models are used. We should however underline at this stage that, while the recognition rates are encouraging ($\geq 99\%$), the vocabulary size is here quite limited (500 words).

6.3 Experiments on a large dictionary

The previous evaluation tests showed that using HMMs for the segmentation and recognition of test words shows reliable results. We could see clearly that inserting *inter-character* between the adjacent characters in a word model and also using *minimal width topology* significantly increases the word recognition rates. Additionally we could show that employing *diagonal covariance matrix* leads still to reliable results and accelerates the computational process. Notwithstanding, the related training procedure was performed using the a priori knowledge of font metrics information and therefore the character segmentation was nearly ideal and thus the system did not need to be fed with a large amount of training data. In this section we are introducing evaluation tests that are based on automatic HMM-training as explained at chapter 4. The advantage of applying automatic HMM-training is that the system works independently from a certain a-priori knowledge about font metrics. On the other hand the algorithm we introduced at chapter 4 uses *linear segmentation* as the initial segmentation of data to then iteratively infer the optimal alignment of the characters in an ULR-word. Such method has the disadvantage that it requires to train on more data in order to smooth out the imperfections of the linear segmentation. For this reason, we increased the number of words in the training set. In addition, the dictionary size of the HMM-recognizer of the previous tests was increased in order to verify the system's limitations. Furthermore, as the system's complexity was increased, in terms of number of to be recognized words, we attempted to introduce richer probability density function estimators based on a weighted sum of Gaussian densities and employed Gaussian Mixture models (GMMs). Finally we tested system's performance for both a *sans serif* and a *serif* font used

as training and test sets. We have to note that our the *HMM-based ULR single word recognizer* is still a mono-font recognizer, i.e. the recognizer assumes that the font of the unknown word image is a priori known. Such an approach is justified, as we can make use of an Optical Font Recognition (OFR) tool for ULR words developed, for example, in our group. To sum up, we aimed at measuring the impact of the following components on system's performance:

- Performing an automatic HMM-training using linear segmentation as bootstrap alignment
- Increasing the system's dictionary size up to 60'000 words
- Increasing the size of the training set up to 6'000 words
- Using Gaussian mixture models (GMMs) to estimate the emission probabilities
- Using serif versus sans serif font as training and test sets

6.3.1 System description

As illustrated in the block diagram of Fig. 6.9, our recognizer system consists of two processes, training and recognition. Both processes are based on HMMs where states are associated to characters. In this manner, any word can be modeled by a HMM where the corresponding states are simply connected together.

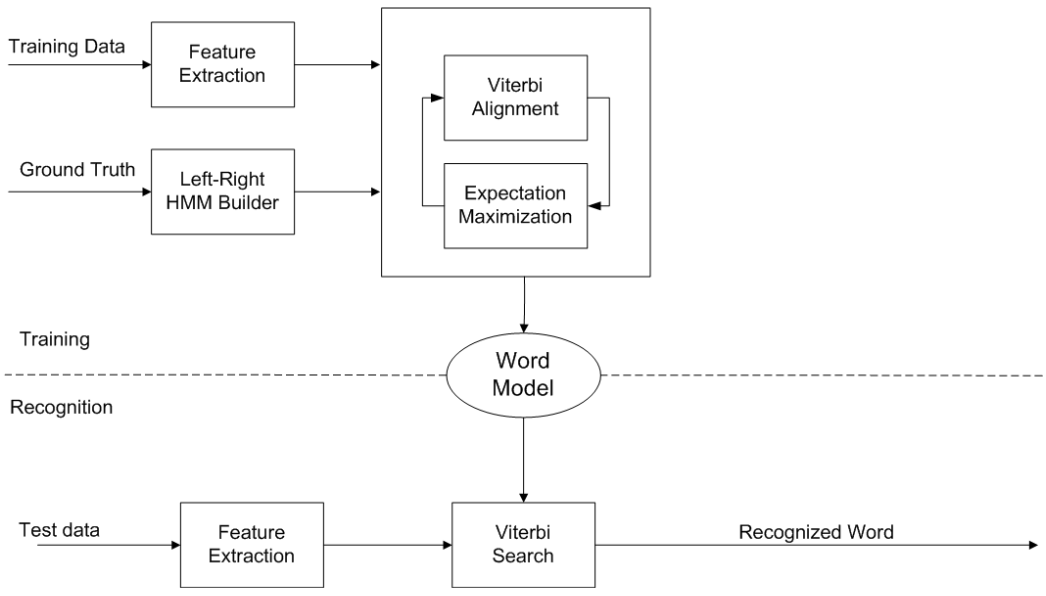


Figure 6.9: Block diagram of the dictionary-based word recognizer

Training configuration

The training method was performed directly on words using an expectation maximization (EM) iterative process as explained in chapter 4. The likelihood-function we selected is based on Gaussian mixture models (GMMs). Additionally, as we could show in the last section that the impact of using full covariance matrix vs. diagonal covariance matrix is negligible, we made the assumption that the components of the feature vector were uncorrelated. Therefore, we were able to use the diagonal covariance matrix. Such an assumption made the recognizer computationally considerably faster than using the full covariance matrix. Fig. 6.10 shows the topology of an HMM recomposed at training time for the word *'cat'*. In this topology we assumed that a fix number of *'#'*(inter-characters) like, for example, 1 or 2 times are inserted between the adjacent characters.

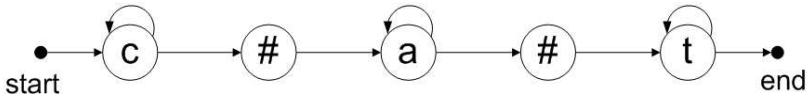


Figure 6.10: Word topology for training

Test configuration

At recognition time a large HMM was built by taking into account the vocabulary defined in a dictionary and according to a tree-like structure that was built to optimize the memory usage. In order to reduce the computation time, a pruning procedure that is explained previously in chapter 4, was applied to discard the less probable paths along the dynamic programming computation. As we were testing our system on unseen vocabulary, the HMMs for each word were composed using the states trained on each character. For estimating the optimal alignment at testing time, we used the Viterbi criterion as explained in detail in chapter 4, stating that instead of considering all potential paths through the HMM, only the best path is taken into account, i.e. the path that maximizes the product of emission and transition probabilities. As stated before, the characters were associated to states in our word models. The number of states was defined by the number of characters composing the word and the choice of the HMM topology to model each character. We chose both the simple-left-right and the minimum width topology with *inter-character* injection such as presented at Fig. 6.5 and Fig. 6.7 to build the large HMM at testing time and compared the resulting system performance. The minimum width values for each character class that were used in the minimum width topology were still inferred from the a priori font metric information.

An algorithm for memory optimization

The dictionary-based HMM would work theoretically as presented in Fig. 6.1, i.e. we had to build the word topology for all the competing words in the dictionary and calculate the probability of the unknown image for each word to determine the index of the word that would have the highest probability. Such an algorithm works well when using a relatively small dictionary, i.e. $\leq$ 3000 words, but the augmentation of the dictionary size to some 10'000 words, increases drastically the computation time and the system turns to be inefficient. Hence, we developed an algorithm that optimized the resulting HMM by merging states that were in common from the start state. In other words, the resulting HMM was like a tree structure where the root would be the starting state and the sons of the root would be the first states of the different HMMs with no duplications of these states. We can write the algorithm in pseudo-code as follows:

- `Repeat`
  - `Build the HMM of the n-th word`
  - `Build the HMM of the n+1-th word`
  - `Build a new empty HMM as merge of both HMM`
  - `Find all the father states in both HMMs that have the same label and equal self-loop and transition probabilities`
  - `Add one of the father states to the merged-HMM`
  - `Modify the new transitions to the merged father state and add them to the merged-HMM`
  - `Add all the transitions from father states to the merged-HMM`
  - `Add all the son states to the merged-HMM`
  - `Delete old transitions that have been modified`
  - `Delete the second father state that has not been added to the merged-HMM`
- `Until n=size of dictionary`

To give an example, let us suppose that we have a big dictionary consisting a large quantity of words including the words *but* and *bat*, the father states *'b'* from words *'but'* and *'bat'* are merged together as shown in Fig. 6.11. Further optimization of the tree topology could be obtained by performing similar merging from the end state, but were not implemented here. Optimization from the root state was enough in terms of cpu and memory footprint reduction.

6.3.2 Evaluation tests

The training set that we used for these evaluation tests contained 6'000 ULR words. Using automatic HMM-training procedure enabled us to segment words into characters in order to

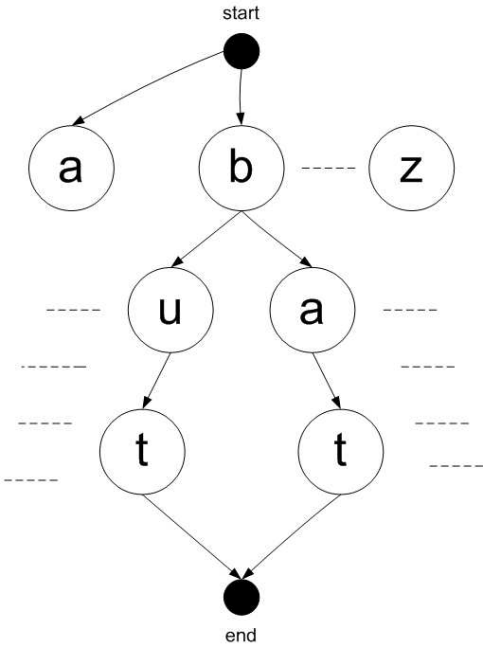


Figure 6.11: Optimization algorithm

estimate the mean vector and diagonal covariance matrix to feed the recognition system. We performed different evaluation tests that we can group as follows:

- **Impact of the number of iteration to estimate the emission probabilities**
  The first question for an automatic training was to tune optimally the components that were used in the process. As already described at chapter 4 we repeated the iteration step until a certain number of iterations at which the previous probabilities were converging to a constant value. Fig. 6.12 shows that after 10 iterations this convergence is reached. When continuing the process until 20 iterations, we see clearly that the probability values are swinging around the value of 10 iterations.

- **Impact of number of Gaussian components**
  We were interested to investigate the impact of number of Gaussian components on the word recognition rates (WRR) for different fonts. Therefore performed various experiments for a *serif* font and a *sans serif* font in roman (plain) style and 9 point size by which we increased the number of Gaussian components. The results are shown in table 6.2. We see clearly in this table that increasing the number of Gaussian components from 1 to 2 drastically increases WRR. Going from 2 to 4 components does not lead to much bigger improvements. Additionally, serif font system reaches less accuracy when comparing to sans serif font. This again is probably due to the fact that serif fonts have more complicated character shapes including small flourishes. We can also observe that the serif font requires more complex models than sans serif font, i.e. using a mixture of 4 Gaussian components

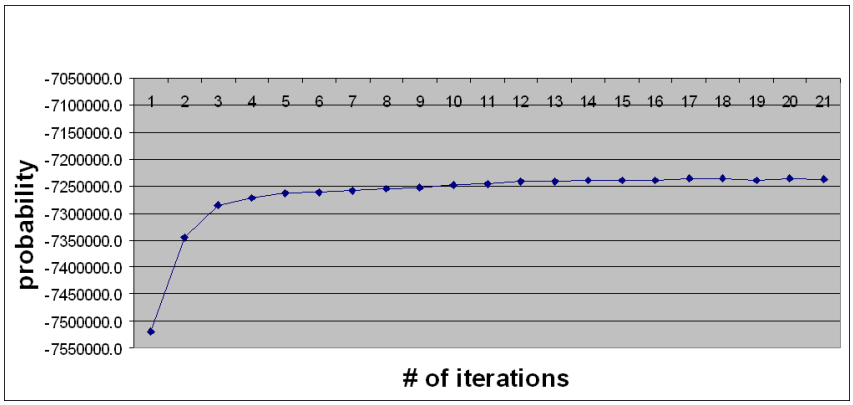


Figure 6.12: Estimation of optimal number of iterations for automatic HMM-training for Verdana, Plain, 9pts

is more accurate for serif font while a mixture of 2 Gaussians seems to be optimal for sans serif font.

Table 6.2: Influence of the number of Gaussian components on WRR of a 3'000 word system for two fonts, plain, 9 points

# Gaussians	Sans Serif	Serif
1	52.6%	16.6%
2	99.5%	97.5%
4	99.0%	98.9%

- **Impact of vocabulary size and HMM-topology**
  We analyzed here the influence of the vocabulary size and of the different HMM-topologies described above. Results were obtained using the 2 Gaussian components-system and are shown in 6.3. We see that the accuracy is slightly decreasing when increasing the vocabulary size. However, the system was still leading to good accuracy even for large vocabulary up to 60'000 words. As already seen from the results of the previous evaluation tests, the minimum width topology delivers in general better results. Additionally, we can observe that similar to the first set of experiments, the serif font is more difficult to recognize than the sans serif font. This is probably also the reason that most chosen fonts for screen-rendered text are *sans serif* fonts due to their overall better *legibility* comparing to the *serif* fonts. Once again, this is an interesting observation that the implemented automatic recognizer has similar tendencies to the human vision system as already described in chapter 3 and is able to better recognize legible sans serif fonts than readable serif fonts.

Table 6.3: WRR for different vocabulary size and HMM topologies and for two fonts: **sans serif** and (serif), plain, 9 points, 2 Gaussian components

Sans serif	3'000 words	12'000 words	60'000 words
Simple left-right	99.5%	98.0%	96.0%
Minimum width	99.8%	99.4%	97.7%
Serif	3'000 words	12'000 words	60'000 words
Simple left-right	97.5%	92.4%	90.3%
Minimum width	98.6%	96.5%	93.4%

6.3.3 Error analysis

We also performed an error analysis that is summarized in Table 6.4 for the serif font. Many errors were related to confusions between characters that are globally similar such as for example '`t`' and '`i`' or '`o`' and '`e`'. Increasing the number of Gaussian components could recover some of these mistakes by allowing more precise modeling. This is clearly illustrated for the word '`aloi`' and '`wenda`'. Besides, a large number of errors were related to characters that shared local similarities such as, for example, the vertical strokes of letters '`m`' and '`n`'. This also led to insertion errors as in the case of '`d`' recognized as '`cl`'. The introduction of minimum width topology allowed to recover some of these mistakes. In some examples, this is the combination of minimum width and model complexity that allowed to remove the mis-recognition, like for the word '`shah`' in Table 6.4.

Table 6.4: Typical examples of mis-recognition for sans serif font, 12'000 test words

Genuine word	Recognized word			
	simple, 2-Gauss	simple, 4-Gauss	min dur, 2-Gauss	min dur, 4-Gauss
aloi	alot	aloi	alot	aloi
wenda	wonda	wenda	wonda	wenda
moller	noller	noller	moller	moller
dumm	clum	clum	clum	dumm
shah	shale	shale	shale	shah

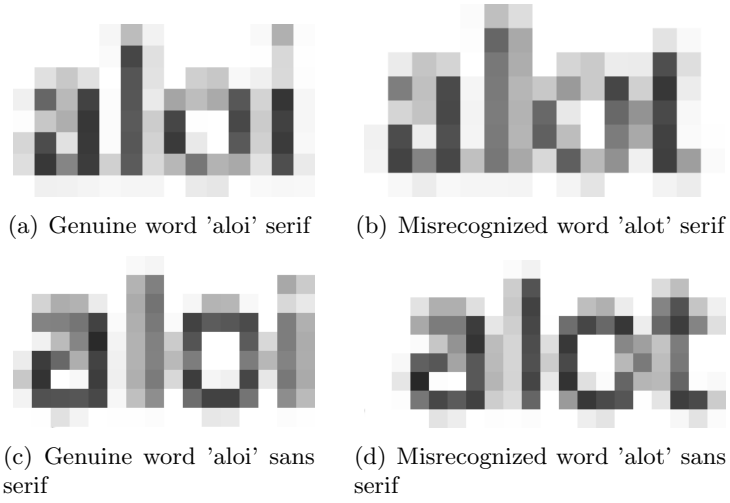
We can see the image of the confused words in two exapmles as below:

- *Confusion between 'aloi' and 'alot'*
  The first line of Fig. 6.13 shows the genuine word *'aloi'* and the mis-recognized word *'alot'* for serif font. The second line shows these two words for sans serif font. As we can see the characters *'i'* and *'t'* are more likely to be confused by the serif font than by the sans serif font. The character *'t'* is wider and therefore can be less confused with the character *'i'* by the sans serif font, whereas they have the same width by the serif font.

- *Confusion between 'wenda' and 'wonda'*
  The shape of character *'e'* in word *'wenda'* and *'o'* in word *'wonda'* in Fig. 6.14 for the

serif font, are pretty similar to each other, for example they both have some white pixels representing a hole in their middle, whereas for the sans serif font, these characters are different as *'e'* has no white pixels comparing to *'o'*.

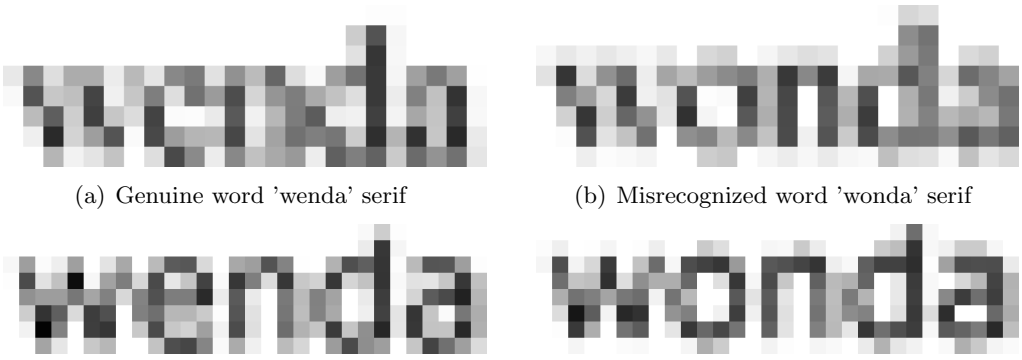
The above error analysis confirms again the higher *legibility* of screen-rendered ULR sans serif fonts over the serif ones.



(a) Genuine word 'aloi' serif (b) Misrecognized word 'alot' serif

(c) Genuine word 'aloi' sans serif (d) Misrecognized word 'alot' sans serif

Figure 6.13: Confusion between *'aloi'* and *'alot'*



(a) Genuine word 'wenda' serif (b) Misrecognized word 'wonda' serif

(c) Genuine word 'wenda' sans serif (d) Misrecognized word 'wonda' sans erif

Figure 6.14: Confusion between *'wenda'* and *'wonda'*

6.3.4 Conclusion

The above experiments have shown that:

- The recognizer is able to perform automatic HMM-training that does not need the a priori knowledge of font metrics to segment characters in an ULR word. Instead the training process finds iteratively the best character alignment within a word

- The recognizer is able to reliably recognize up to 60'000 words that is fairly close to the size of a real world dictionary
- The system's performance increases by using the Gaussian Mixture models (GMMs) to estimate the emission probabilities
- The recognizer can be trained both for serif and sans serif words. However, serif fonts are leading to a slight degradation of the accuracy probably due to their lower *legibility* comparing to sans serif fonts. This is probably the reason why serif fonts are mostly used for printed text as they have higher *readability* than sans serif fonts.

6.4 Experiments on an open-vocabulary recognizer

With the previous set of experiments we used a HMM-based single ULR word recognizer that was able to solve the character segmentation and word recognition at the same time. In addition, the system was able to perform automatic training and we additionally showed that using GMMs instead of mono-Gaussian density function to model the features' distribution, considerably increased the WRR specially when the recognizer contained a large vocabulary. In this approach, one left-right HMM was built for each word where characters were associated to one or more HMM states. A large HMM was finally built considering the vocabulary taken from a dictionary of up to 60'000 words, making each word-level HMMs competing against each other. While giving reliable results, this approach had two drawbacks. First, the recognition was limited to the words available in the dictionary and would have typically configured as mono-lingual. Some potential inputs were, indeed, not available in the dictionary, such as inflected forms or proper names, and therefore could not be recognized. Second, the memory and CPU usage was still pretty high even when performing optimization algorithms. A porting of this system on low-end devices such as PDAs would have been difficult to realize. To overcome these drawbacks, we built a word recognizer based on ergodic topology for the HMM, where all character models were connected to each other. With such a system, the vocabulary size is potentially unlimited while keeping low the usage of system resources. We were specifically interested to investigate the following:

- Inferring the appropriate alignment of inter-character from the automatic training
- Inferring the minimum and maximum width values from the training procedure to replace the previously used knowledge based approach
- Evaluating system performance using minimum and maximum width values
- Finding the optimal number of Gaussian components for training of GMMs
- Evaluating the system's performance when changing font sizes from 6 to 10 points for different fonts

- Testing the robustness of the system for various fonts
- Building a serif and a sans serif recognizer and evaluating their performance

6.4.1 System description

As illustrated on Fig. 6.15, our recognizer consisted of two parts: training and recognition. The training part was the same as for the dictionary-based recogfnizer and aimed at computing character models by recomposing word-level HMMs based on simple left-right topology iteratively analysing a large training set of word images. During training, there was no minimum and maximum width model used in this approach. At recognition time, we used an Ergodic HMM topology for word modeling where each character model was connected to each other. At test time, we tested different character topologies with minimum and maximum width constraints. The values of such constraint were inferred either from font metrics information or from automatic HMM-training. The latter case had the advantage that it did not need the a priori knowledge of font metrics. Below we give more details about the different blocks composing our system.

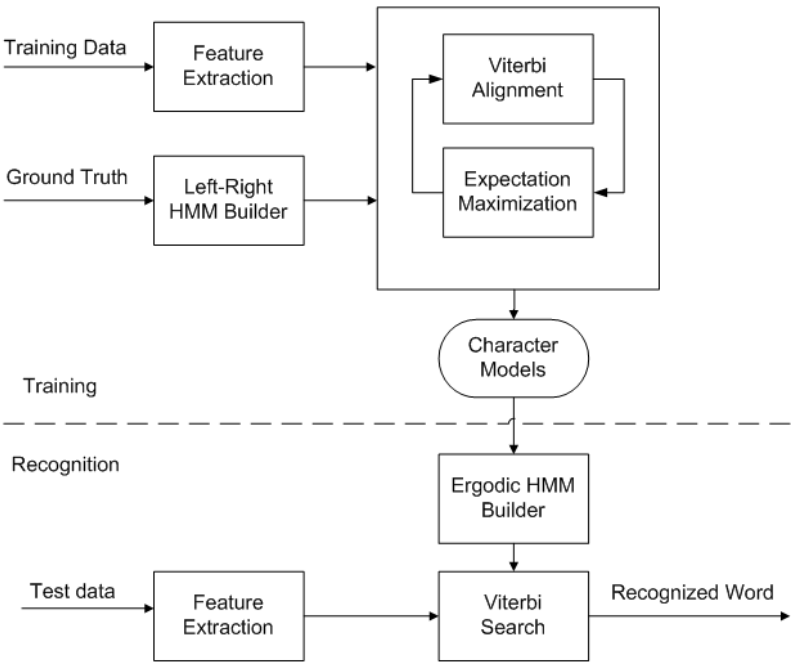


Figure 6.15: Block diagram of open vocabulary word recognizer

- **Training configuration**

  The training method was performed directly on words for which a simple left-right HMM was recomposed by gluing together the corresponding character sub-models. At training time, a character was modeled with one state where a self-loop transition allowed to remain

in this model as long as the sliding window was on top of the character. Fig. 6.16 shows the topology of an HMM recomposed at training time for the word 'cat' as already introduced in chapter 4. In this topology we let the *'#'*(inter-character) to be treated as any other character, i.e. we let Viterbi to remain an unlimited number of times in inter-character and state and determine its optimum alignments (minimum and maximum constraints.

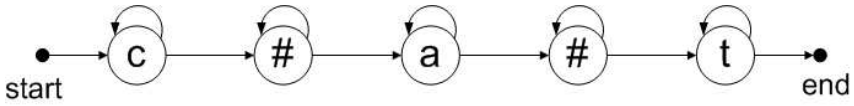


Figure 6.16: Word topology for training

- **Test configuration**

  At testing time, we are proposing here to build an HMM with an ergodic topology at character level. Each character sub-model can take different topology as introduced in chapter 4, i.e. with or without minimum or maximum width constraints. In our previous experiments, we used two different HMM topologies and observed that a topology using minimum width constraints delivered better results than the simple left-right topology. As the simple left-right topology basically impeaches the decoding procedure to leave too early a state giving low local scores. However, the minimum width values were inferred from the font metrics of each character. For these evaluation tests, we are additionally introducing an extension of this topology that is illustrated in chapter 4. The values of each transitions leading to the end of the model are corresponding to the probabilities $p_i(n)$ of observing at least n feature vectors in a given character model i. These values are computed during training by inspecting the Viterbi forced alignment on each word. This new character model, while introducing similar minimum width constraints as in model Fig. 6.17, introduces also a maximum width constraint, expressing the fact that characters have limited width. For a given test image, we used the Viterbi criterion to determine the best path in this ergodic topology. This path actually defines the recognized sequence of characters composing the word. The Viterbi decoder was also configured to prune out the less probable paths along the recognition process to keep the memory and cpu usage in reasonable ranges.

  In other related recognition domains where the vocabulary of the input is potentially very large, ergodic topologies have also been proposed. We can refer to [111] where an ergodic HMM system is presented to recognize handwritten street names, to [112] for automatic language identification and to [113] for speaker verification.

6.4.2 Experimental results

We performed several evaluation tests that we group as follows:

Experiments with character models based on minimum width constraints

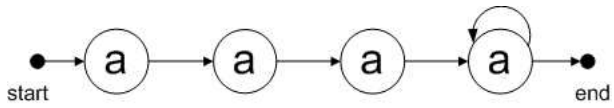


Figure 6.17: A character topology based on minimum width constraints

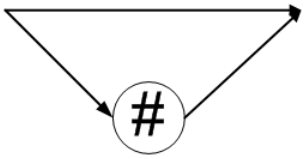


Figure 6.18: Inter-character model

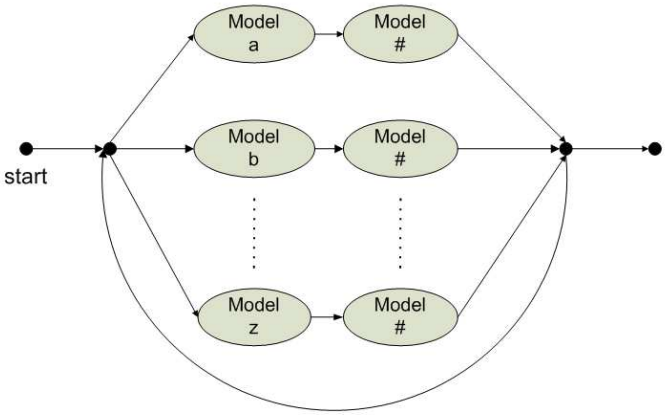


Figure 6.19: Ergodic HMM with character model based on minimum width constraints

We built an ergodic system as shown in Fig 6.19 where the characters are modeled as Fig. 6.17. As can be seen on this Figure, the characters are all connected to inter-character, that itself is modeled like in Fig. 6.18, and accessible in parallel and a transition is looping back to all characters. In this Figure, the states represented by black dots are non-emitting states classically used to glue sub-HMMs together. We employed this method on font *Verdana* in different point sizes and for different number of Gaussian components. The results are shown at table 6.5. We can conclude therefore the following points:

- Increasing number of Gaussian components up to 64 increases drastically as shown in Fig. 6.20 the WRR. We noticed that a GMM with 64 Gaussian components is a good trade-off between reaching a good WRR and the size of the available training data.

Table 6.5: WRR(%) for open vocabulary recognizer with mininum width constraints

# of Gaussians	Font Size				
	6 pts	9 pts	11 pts	12 pts	Mean
2	48.98	29.78	23.03	17.11	29.72
4	61.07	55.81	29.9	39.95	46.68
8	73.15	62.24	51.74	53.35	60.12
16	79.65	75.47	74.78	67.00	74.22
32	84.97	88.93	89.51	82.78	86.54
64	91.02	93.63	93.85	90.25	92.19

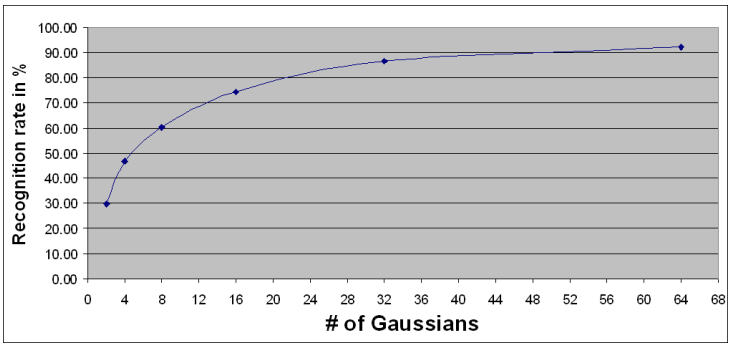


Figure 6.20: Increasing recognition rates by increasing the number of Gaussian components

- The WRRs for point size 12 are lower than those for point size 11 that has to be further investigated.

Experiments with character models based on minimum and maximum width constraints

The last experiments showed that big font sizes like 12 points show worse WRR than smaller font sizes like 9 and 11 points. We assumed that this is an imperfection of our model as we forced the Viterbi whether to skip the inter-character or to stay one time in it as shown in Fig. 6.18, which is apparently not correct for big font sizes. Probably with big font sizes, the contextual noise between the adjacent characters can be represented by more than 2 pixels. This forced us to use a new character model as illustrated in Fig. 6.21 with minimum and maximum constraints that are inferred during the training process. Our ergodic topology is illustrated on Fig. 6.22. As can be seen on this Figure, the characters are all accessible in parallel and a transition is looping back to all characters from the inter-character model '#'. Furthermore, the inter-character is modeled with minimum and maximum constraints as any other character. We assumed that this model is more general and applicable both for small and big font sizes, as the transitions between the states reflect the 'real' probability of repeating each character and sequence. Besides inserting the appropriate inter-character model for each point size reflects a

better alignment of inter-character for each specific font than for previous experiments. Finally, the word model lets either insertion of inter-character for big font sizes or skip the inter-character for smaller font sizes.

In Fig. 6.22, the states represented by black dots are again non-emitting states that glue sub-HMMs together. The transition probabilities going from submodel '#' back to each character is actually corresponding to the case of equiprobable character sequences, for any pair of characters.

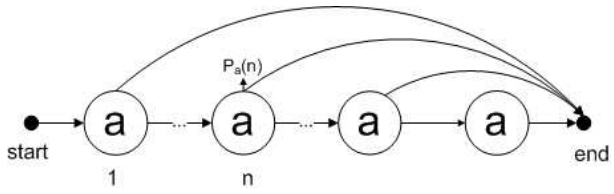


Figure 6.21: Character model with min-max width constraint

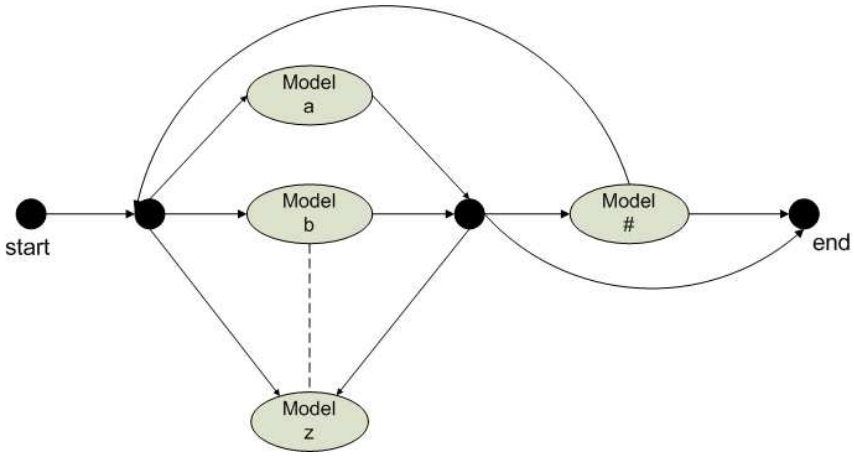


Figure 6.22: Ergodic HMM for character models with minimum and maximum constraints

We experimented with font *Verdana* using different font sizes and GMMs with 64 components and could get better WRR for point size 12 as can be seen at table 6.6. On the other hand the WRRs of other font sizes were not significantly improved. We applied an error analysis for the btained results that is shown in table 6.7.

Table 6.6: WRR (%) for open vocabulary recognizer with min-max width constraints

	Font Size					
	7 pts	8 pts	9 pts	10 pts	11 pts	12 pts
sans serif	91.60	92.08	93.01	93.63	93.77	97.76

We see in table 6.7 confusions between 'q' and 'd', 'b' and 'p', 'h' and 'n', 'n' and 'r', 'n' and 'h'. Our hypothesis is here that the training set does not contain enough training data for some

Table 6.7: Error analysis for open vocabulary recognizer using min-max width constraints

Genuine word	Recognized word for Font Size					
	12pts	11 pts	10 pts	9 pts	8 pts	7 pts
aldercy	alqlercy	aldercy	aldercy	aldercy	aldercy	alqercy
dakar	qlakar	dakar	qlakar	dakar	qlakar	qdakar
dawn	qlawn	dawn	qlawn	dawn	qlawmh	qdawn
deluded	qleluqled	deluded	qleluded	deluded	qleluded	qdeluqed
enrique	enridue	enrique	enridue	enrique	enridque	enridue
weapons	weapons	weapons	weapons	weapons	weapons	weapbons
bausch	bausch	bausch	bausch	bausch	bausch	bpausch
stann	stann	starhn	starhn	stann	stann	stann
tsang	tsang	tsarhg	tsarhg	tsang	tsang	tsang
stann	stann	starhn	starhn	stann	stann	stann
exiting	exiting	exitirhg	exitirhg	exiting	exiting	exiting

letters that are frequently confused (for example 'q').

Experiments with a *balanced* training set

We enlarged the size of training set that contained more words including the confused characters and we obtained better intermediary WRRs. But we still had other confusions, which motivated us to produce a balanced-training set containing a balanced quantity of each character combined with each other character at its left and right sides. That is, we decided to use a training set containing character trigrams of all possible combinations. Therefore, the training set contained $26 * 26 * 26 = 17'576$ trigrams. Since we gained better WRR with the character models with minimum and maximum width constraints, we decided to run a new series of experiments based on this character model and word models based on ergodic topology as in Fig. 6.22 and use the balanced-trigrams as training set. We were also interested to gain results for both a serif and a sans serif font. The results are listed in table 6.8. In addition, we tested different font sizes and font styles for the two selected fonts. We can see in this table that the WRRs for sans serif font is indeed considerably improved when comparing to the results from table 6.6. The improvement of WRR is about 1.5%(for 12 pts) and 6%(for 8,9 pts). When comparing the WRRs of serif and sans serif fonts, we see that sans serif font has higher WRRs. This again confirms the finding of the previous chapter that sans serif fonts are more legible than serif fonts for our computerized recognition algorithms.

Mono-font experiments

To assess performance and roboustness of the open vocabulary recognizer that used as training set the balanced trigrams and minimum-maximum constraint topology for modeling characters

Table 6.8: WRR (%) for open vocabulary recognizer for a serif and a sans serif font with different font styles and sizes using min-max width constraints and balanced training set

Font family	Font style	Font Size				
		7 pts	8 pts	9 pts	10 pts	12 pts
sans serif	plain	96.98	96.21	98.84	98.89	99.28
	bold	97.09	97.25	98.34	99.09	99.48
	italics	95.68	96.84	97.5	97.97	98.95
	bold+italics	96.81	98.22	98.67	98.89	99.36
serif	plain	93.88	96.10	97.81	98.48	99.47
	bold	94.83	95.09	98.34	98.14	99.96
	italics	95.51	96.12	96.84	97.17	99.28
	bold+italics	96.59	96.64	97.44	98.12	98.81

in a word, we performed experiments with various serif and sans serif fonts. These experiments were conducted in a mono-font context. As already stated, we supposed that we could recognize the font in a preprocessing step using an existing OFR for ULR words. We used the following font combinations to build 48 fonts for building training and test sets:

- 3 sans serif fonts: Arial, Tahoma, Verdana
- 3 serif fonts: Times New Roman, Garamond, Georgia
- 4 different font styles: plain, bold, italics, bold + italics
- 2 different font sizes: 9pts, 11pts

The results are shown in table 6.9. We see from this table that the sans serif font Verdana has the best overall WRR. Furthermore, the serif font Georgia has the best WRR amongst serif fonts. These observations are in accordance with the mono-font experiments performed on single characters presented in the previous chapter.

Multi-font experiments

We were interested to assess the performance of our open vocabulary recognizer as serif or sans serif classifier. Once again, we can assume here using an available OFR that determines whether the to be recognized word is a sans serif or a serif font and the task it to recognize the serif fonts with the serif recognizer and the sans serif fonts with the sans serif recognizer. We used the same serif and sans serif fonts as for the mono-font experiment described in this section for building training and test sets. Table 6.10 shows the gained results for the sans serif recognizer and table 6.11 the results for the serif recognizer. As expected, the mono-font recognizer has a better overall WRR. In addition, the overall WRR of the sans serif recognizer is better than the

Table 6.9: WRR (%) for open vocabulary recognizer with min-max width constraints using the balanced training set for different fonts

Font family	Fontsize	Font Style				
		plain	bold	italics	bold+italics	Mean
Arial	9pts	96.89	97.87	96.78	97.4	97.24
	11pts	97.6	98.9	97.1	98.23	97.96
Tahoma	9pts	97.03	98.53	97.12	97.98	97.67
	11pts	98.89	99.01	97.67	98.56	98.53
Verdana	9pts	98.84	99.27	97.5	98.81	98.61
	11pts	99.28	99.48	97.95	99.17	98.97
Times	9pts	94.09	95.14	93.23	94.56	94.26
	11pts	96.45	97.65	95.34	95.1	96.14
Garamond	9pts	95.76	96.53	95.12	96.35	96.57
	11pts	96.8	97.64	96.18	97.49	96.76
Georgia	9pts	97.81	98.34	96.84	97.44	97.61
	11pts	99.47	99.94	97.28	98.81	98.88
	Mean	97.41	98.19	96.63	97.49	97.67

one for serif recognizer. Furthermore, these results show the same tendencies with those gained from multi-font experiments presented in previous chapter.

Table 6.10: WRR (%) of open vocbulary system as a sans serif recognizer with min-max width constraints using the balanced training set

Font family	Fontsize	Font Style				
		plain	bold	italics	bold+italics	Mean
Arial	9pts	96.13	97.3	95.89	96.8	96.53
	11pts	97.23	98.53	96.95	98.01	97.98
Tahoma	9pts	96.87	97.79	96.97	97.43	97.27
	11pts	98.45	98.86	97.37	97.89	98.14
Verdana	9pts	98.24	98.75	97.2	98.1	98.07
	11pts	99.04	99.35	97.76	98.9	98.75
Mean		97.66	98.43	97.02	97.86	97.74

6.4.3 Conclusion

The above experiments showed that our open vocabulary recognizer performs well for the recognition of any arbitrary ULR single words. We could increase the recognition rates and remove the system's imperfections using the following improvements:

- Finding an optimal number of Gaussian components to represent the feature's distribution

Table 6.11: WRR (%) of open vocbulary system as a serif recognizer with min-max width constraints using the balanced training set

Font family	Fontsize	Font Style				
		plain	bold	italics	bold+italics	Mean
Times	9pts	93.28	94.14	92.23	94.01	93.42
	11pts	96.24	97.21	94.88	94.87	95.8
Garamond	9pts	95.03	95.93	94.78	95.93	95.42
	11pts	96.23	97.33	95.97	97.16	96.67
Georgia	9pts	97.24	97.85	96.12	97.17	97.1
	11pts	99.01	99.32	97.01	98.38	98.47
Mean		96.18	96.99	95.17	96.25	96.15

- Using a more accurate character model with minimum and maximum width constraints that were inferred during automatic HMM-training.
- Using a balanced training set, in which character trigrams were built to represent all the possible adjacencies at their left and right borders

6.5 Conclusions

Table 6.12: Dictionary based word recognizer vs. open vocabulary word recognizer

Criterion	Dictionary based	Open vocabulary
Rec rates	better	good
# of Gaussians	4 components	64 components
Training set size	medium $<$ 8000 words	large $>$ 19000 words
Language independence	typically no	yes if Latin chars are used
Flexibility	limited to dict.	high, no word limitation
Portability for low end devices	low – medium	medium – high

In this chapter, we presented two dedicated recognizers for anti-aliased words at ultra low resolution with font sizes between 7-12 points. Both recognizers use the central moments as feature extraction method and mixtures of Gaussian models to represent the features' distribution. Table 6.12 shows a comparison of both for different criteria.

As can be seen in table 6.12 the open vocabulary recognizer is more of practical use, as it is portable to low end devices. In addition, the of open vocabulary recognizer can deal with an unlimited dictionary size and is also not dependent to a specific language. The only limitation with the open vocabulary recognizer is that it needs as input words written in Latin alphabet. We believe but that it is also able to recognize words coming from other similar alphabetical systems

like for example Cyrillic or Greece alphabet simply by train the corresponding character models. Furthermore, we believe that we could use the principles of our system to develop dedicated recognizers for Arabic, Chinese and other alphabets. On the other hand, the dictionary based recognizer delivers more accurate results using less Gaussian components for estimating the probability density functions of the chosen features and also needs a relatively smaller training set. This is at the price of higher cpu, higher memory footprint and limited vocabulary size.

Chapter 7

Conclusions and future works

In this thesis, we have studied the problem of text recognition in Ultra Low Resolution (ULR) images. Such text images are mostly produced using an image processing application like Adobe Photoshop, Macromedia Fireworks, Corel Draw, Corel Photopaint etc.. Once text images have been generated using one of these tools, they are often inserted into HTML pages to compose menus or other clickable labels that appear in banners, buttons, advertisements etc.. The length of such text is usually not more than a couple of words or even one single word in case of embedded in buttons. When text contained in such images has a font size below 10 points, anti-aliasing is usually applied in order to smooth edges and diagonals. Recognition of such text in these types of images has been the target of our research.

In this dissertation, we have assumed that words can be accurately detected and segmented using classical algorithms used in document analysis. In other words, we have made the assumption that our system receives as input an image of a single word. To achieve our goals, first we have conducted a preliminary study on isolated characters. We then have used the outcomes of this study to better build a system for the recognition of isolated words. In our research work, we have decided to use methods of statistical pattern analysis. Statistical pattern analysis is based on two important components: 1)a feature extraction and 2)a statistical classification method. As features, we have used the first and second order central moments. In addition, we have used Bayes decision using the mono-Gaussian density functions for the task of identifying single characters and hidden Markov models (HMMs) combined with mixtures of Gaussian models for the recognition of a single word.

7.1 Study of single characters

In this study we aimed at testing the discriminative power of the selected features. Our classification system used the principle of maximum likelihood to choose the most probable character class for the test image, assuming equal class prior probabilities. We also assumed that the distribution of extracted features of character images would be corresponding to a mono-Gaussian

multivariate density function. We verified this assumption using a normal distribution goodness-of-fit test. The gained results have shown that our selected features are fairly discriminative and can be used as a part of our single word recognizers. Furthermore we have tested the implemented single character identification system on 192 fonts and gained highly motivating character recognition rates of up to 99.99 in a mono-font context. We have also investigated the possibility of recognizing text for two different rendering methods and could show that the system can be implemented to recognize single characters independently of the anti-aliasing rendering methods.In addition, we could also test a multi-font single character identifier by training the character models with all the selected fonts, but for one size and one style, and the task has been to recognize any single character belonging to one of these fonts. We have obtained recognition results up to 95% for this task. We could group serif and sans serif fonts together and train serif and sans serif character models to gain a serif and a sans serif classifier. We could obtain a better CRR up to 99% for both classifiers. Additionally, we have obtained better overall CRRs for sans serif fonts than for serif fonts as the decorative 'serifs' cause more contextual noise at character boundaries and decrease the recognition accuracy for serif fonts. Generally, we have observed that we can make machine to better learn recognizing sans serif fonts than serif fonts. It is interesting to observe that machins show similar perception tendencies as human vision system for this task. Indeed, we can observe an increasing use of sans serif fonts like Verdana and Tahoma in browsers or other computer screen text visualization software. These fonts have also been specifically designed to be more legible for human eyes on computer screens than the serif fonts. Serif fonts are mostly used for printed material where readability takes priority over legibility.

We continued our study with an analysis of the impacts of anti-aliasing on adjacent characters. This has resulted to a thorough and meaningful understanding of the contextual noise between the connected adjacent characters for different font families and font styles with font sizes between 3-10 points. We have concluded that such contextual noise has been a major diminishing factor for the recognition of contextual characters. By developing a knowledge-based segmentation method we could discard such noisy zones and obtain higher recognition results. However, such knowledge-based segmentation was completely artificial as we had to assume the a priori knowledge of the character strings of a particular word. Therefore, we have decided to consider such contextual noise as an extra character enabling us to segment it during training and recognition process.

Finally the experiments of this study have clearly shown the limitations of the mono-Gaussian density function to model the features of contextual characters. We have drawn the conclusion that the common approach in document analysis to segment characters prior to recognition is not applicable in our case. We sought thus for a statistical method capable of simultaneously segment and recognize the connected characters in single words.

7.2 Recognition of single words

In this study, we have investigated the issue of recognizing single words. The targeted words, as previously stated, assumed to be generated with ultra low resolution, rendered with anti-aliasing filters and with small font sizes. Under such conditions, the characters in the words are usually connected and the adjacent characters contain contextual noise between each other. Thus, the features extracted for a given character are impacted due to this noise. Therefore we opted to use hidden Markov models (HMMs) for the recognition task. Using HMMs, we could model words with characters as states with a left-right topology and insert the inter-character as an extra character model between them. HMMs can additionally perform an automatic training that results in gaining optimal state emission and transition probabilities for each character state without making any a priori assumption about the segmentation of words into characters. Furthermore, we have shown that models linked to the minimum and maximum width constraints of characters can be easily included in the HMM topologies, increasing further the performances.

We could show that inserting the inter-character model between the adjacent characters significantly improves the recognition results. We also have tested different character interconnection possibilities(topologies) using knowledge-based character models with minimum width constraints and moreover using minimum and maximum width constraints that have been inferred during the automatic HMM training. Further, we used mixtures of Gaussian models (GMMs) for modeling the distribution of our features. We can underline two justifications for such a choice: first the system's complexity has been increased as we no longer dealt with single characters but with single words. Secondly the experiments dealing with single contextual characters have shown the insufficiency of the mono-Gaussian density functions to model the feature's distribution.

We have basically built two single word recognizers. The first one is a dictionary-based recognizer that relies on the recognition of test words that are chosen from a large dictionary. The second one is a language-independent, open vocabulary recognizer that can deal with any arbitrary word written in any language in Latin alphabet.

We have built a large HMM containing of several thousands of words and used a tree-based optimization algorithm to reduce the system cpu and memory usage. The maximum number of words contained in our dictionary-based system was 60'000 words. We could gain fairly good word recognition rates (WRR) of up to 99.9% with a mixture of 4 Gaussian components to model the features' density functions. Although the results were highly promising, the system has shown a major weakness: such a dictionary-based system is less suitable to be ported on low-end devices like PDAs and mobile cameras and therefore not practical to be used for different possible applications for such devices. Furthermore, the system was limited to the recognition of the 60'000 words, automatically discarding all unknown inflections or proper names.

Thus we opted to build an open vocabulary recognizer that has relied on character models using a HMM ergodic topology. This made it possible to build each arbitrary word by connecting

one character state to another character state. Such a recognizer is more portable as it uses only HMMs of approximately 100 characters and special signs to be ported on low-end devices. On the other hand, such an open vocabulary system is far more complex than the dictionary-based system and experiments have shown that we have to use a mixture of 64 Gaussian components to gain WRRs above 96%. Another complexity of such a system is that it needs a much larger training set to cover all the potential details of the distribution of features. We have conducted several experiments with frequently confused characters and could finally build a training set that could deliver WRRs that have been comparable with the ones gained from the dictionary-based system, i.e. up to 99.9%.

Finally, we have tested the robustness of our system using 48 different fonts and could gain good WRRs of up to 99.54% in a mono-font context. We then have mixed the serif and sans-serif training tests together building a serif and sans serif recognizer and could still reach still WRRs of up to 99%.

7.3 Future perspectives

We have presented two single word recognizers for ultra low resolution words that are anti-aliased and have small font sizes between 6-12 points. We could show that such a system is highly capable to gain more accurate recognition accuracies than, for example a current commercial OCR proclaimed to recognize screen-shot text images as shown in chapter 2. Our system can generally be used in a mono-font context, such an approach is justified when combing our system with an existing Optical Font Recognizer(OFR). Our goal for developing such a system was primarily to recognize anti-aliased text with a maximum height of 10 pixels such as those frequently found in web images. Going further, we believe that the principles of the presented system could be generalized to recognize other text images that are captured with other low resolution devices such as digital and video cameras. Obviously some components of the system such as injecting of inter-character to model the noise in our case has to be modified and the system has to be adapted to the new sources of noise in digital cameras like 3D-perspective deteriorations, different illuminations and zooming. In addition, the presented recognizers are not independent of the font size, i.e. they have to be further improved to be scale invariant. Potential improvements in this direction could be additional scale invariant moment features like 7 Hu's moment invariants [114]. In addition, we could think of using the rotation invariant moments like Zernike moments [80] as features to make the recognizers rotation invariant making them independent of the different font styles like *roman* which has more rectangular shapes and *italics* with more slanted and angled shapes.

Moreover, potential future works could go in the direction of including linguistic constraints in the system architecture. One possibility would be to measure character trigram frequencies from a dictionary and to use these values for the transition probabilities between character sub-models. A generalization of this approach could also be performed computing n-gram character

sequences. Another possibility could be to keep the n-best recognition hypothesis as output of the ergodic model and to prune out the hypothesis that are improbable looking in a dictionary. Future work could also consider using real-life images extracted from the web in the evaluation framework developped in this thesis.

List of Figures

List of Tables

Bibliography

[1] D. Bathurst, R. Bathurst, and D. Davis. *The Telling Image: The Changing Balance between Pictures and Words in a Technological Age*. Clarendon Press, 1990.

[2] A. Antonacopoulos, D. Karatzas, and J.O. Lopetz. Accessing textual information embedded in internet images. In *Proc. of Electronic Imaging,Internet Imaging II*, San Jose, California, USA, 2001.

[3] S. Santini. Multimodal search in collections of images and text. In *Journal of Electronic Imaging, Volume 11, Issue 4*, pages 455–468, 2002.

[4] S. Scarloff, T. Taycher, and M.L. Cascia. Imagerover: A content-based image browser for the world wide web. In *Proc. of IEEE Workshop on Content-Based Access of Image and Video Librairies (CBAIVL97)*, pages 2–9, 1997.

[5] A.B. Benietz, M.Beigi, and S.F.Chang. A content-based meta search engine for images. In *SPIE Proc. of Storage and Retrieval for Image and Video databases*, 1997.

[6] Z. Gong, L. Hou, and C. W. Cheang. Web image indexing by using associated texts. In *Knowledge and information systems*, volume 10, No. 2, pages 243–264, Faculty of Science and Technology, University of Macau, MACAO, 2006.

[7] K. Jung, K.I. Kim, and K. Jain. Text information extraction in images and video: a survey. In *Journal of Pattern Recognition*, volume 37, pages 977 – 997, 2004.

[8] J. Liang, D. Doermann, and H. Li. Camera-based analysis of text and documents: a survey. In *International Journal On Document Analysis and Recognition*, volume 7, pages 84–104, 2005.

[9] D. Doermann, J. Liang, and H. Li. Progress in camera-based document image analysis. In *Proc. of Seventh International Conference on Document Analysis and Recognition (ICDAR03)*, volume 1, page 606, 2003.

[10] A. Antonacopoulos and D. Karatzas. An anthropocentric appraoch to text extraction from www images. In *Proc. of the 4th IAPR Workshop on Document Analysis Systems (DAS00)*, pages 515–526, Rio de Janiro, Brazil, 2000.

[11] A. Antonacopoulos and D. Karatzas. Text extraction from web images based on human perception and fuzzy interface. In *Docuemnt Analysis Systems V*, Princeton, NY, USA, 2002.

[12] A. Antonacopoulos and D. Karatzas. Text extraction from web images based on a split-and-merge segmentation method using color perception. In *Proc. of the 17th International Conference on Pattern Recognition (ICPR04)*, Cambridge, UK, 2004.

[13] D. Lopresti and J. Zhou. Document analysis and the world wide web. In *Proc. of IAPR Workshop on Document Analysis Systems*, pages 651–669, PA, 1996.

[14] D. Lopresti and J. Zhou. Ocr for world wide web images. In *Proc. of SPIE on Document Recognition IV*, pages 58–66, 1997.

[15] D. Lopresti and J. Zhou. Extracting text from www images. In *Proc. of the 4th International Conference of Document Analysis and Recognition (ICDAR97)*, pages 248–252, 1997.

[16] S.J. Perantonis, B. Gatos, and V. Maragos. A novel web image processing algorithm for text area identification that helps commercial ocr engines to improve their web recognition accuracy. In *Proc. of the second International Workshop on Web Document Analysis*, Edinburgh, United Kingdom, 2003.

[17] S.J. Perantonis, B. Gatos, V. Maragos, V. Karkaletsis, and G. Petasis. Text area identification in web images. In *Lecture Notes in Artificial Intelligence n. 3025 Springer Verlag*, pages 82–92, Samos, Greece, 2004.

[18] T. Kanunngo and C. Ha Lee. Using html metadata to find relevant images on the web. In *Proc. of Internet Computing*, volume II, pages 842–848, Las Vegas, USA, 2001.

[19] J. Ohya, A. Shio, and S. Akamatsu. Recognizing characters in scene images. In *IEEE Transactions on Pattern Analysis and Machine Intelligence*, volume 16, No. 7, pages 214–220, 1994.

[20] J. C. Shim, C. Dorai, and R. Bolle. Automatic text extraction from video for content-based annotation and retrieval. In *Proceedings of the 14th International Conference on Pattern Recognition (ICPR98)*, volume 1, page 618, 1998.

[21] J. Zhang, X. Chen, A. Hannemannn, J. Yang, and A. Waibel. A robust approach for recognition of text embedded in natural scenes. In *Proceedings of 16th International Conference on Pattern Recognition*, volume 3, pages 204–207, 2002.

[22] D. Chen, K. Shearer, and H.Bourlard. Text enhancement with assymetric filter for video ocr. In *Proc. of 11th International Conference on Image Analysis and Processing*, pages 192–197, 2001.

[23] S. N. H. Sheikh Abdullah, M. Khalid, R. Yusef, and K. Omar. License plate recognition using multi-cluster and multilayer neural networks. In *Proc. of Information and Communication Technologies (ICTTA06)*, volume 49, pages 1818–1823, 2006.

[24] T. Naito, T. Tsukada, K. Yamada, and K. Kozuka nd S. Yamamoto. Robust license-plate recognition method for passing vehicles under outside environment. In *IEEE transactions on Vehicle Technology*, volume 49, pages 2309–2319, 2000.

[25] S. L. Chang, L. S. Chen, Y. C. Chung, and S. W. Chen. Automatic license plate recognition. In *IEEE transactions on Intelligent Transportation Systems*, volume 5, pages 42–53, 2004.

[26] C. A. Rahman, W. Badaway, and A. Radmanesh. A real time vehicle's license plate recognition system. In *Proceedings of IEEE Conference on Advanced Video and Signal Based Surveillance*, pages 163–166, 2003.

[27] R. Lienhart and S.Frank. Automatic text recognition in digital videos. In *Proc. SPIE 2666*, pages 180–188, Boston, USA, 1996.

[28] R. Lienhart. Automatic text recognition for video indexing. In *Proc. of the ACM Multimedia*, pages 11–20, Boston, USA, 1996.

[29] R. Lienhart and W.Effelsberg. Automatic text segmentation and recognition for video indexing. In *Proc. SPIE 2666*, pages 180–188, 1998.

[30] Micael A. Smith and Takeo Kanade. Video skimming for quick browsing based on audio and image characterization. In *Technical Repost CMU-CS-95-186*, Carneige Melon University, 1995.

[31] Y. Zhong, H. J. Zhang, and A.K. Jain. Automatic captoion localization in compressed video. In *IEEE Transactions on Pattern Analysis and Machine Intelligence*, volume 22, Issue 4, pages 385–392, 2000.

[32] C.S. Shin, K.I. Kim, M.H. Park, and H.J. Kim. Support vector machine based text detection in digital video. In *Proceedings of the 2000 IEEE Signal Processing Society Workshop*, volume 22, Issue 4, pages 634–641, 2000.

[33] R.Lienhart and A.Wernicke. Localizing and segmenting text in images and videos. In *IEEE Transactions of Circuits and Systems for Video Technology*, volume 12, No. 4, 2002.

[34] J. Xi, X. Hua, X. Chen, L. Wenyin, and H. Zhang. A video text detectiona and extraction system. In *proceedings of IEEE International Conference on Multimedia and Expo*, pages 873–876, 2001.

[35] T. Sato, T. Kanade, E. Hughes, and M.Smith. Video ocr for digital news archive. *Proc. of the 1998 International Workshop on Content-Based Access of Image and Video Databases (CAIVD98)*, page 52, 1998.

[36] D. Chen, J. Odobez, and H. Bourlard. Text detection and recognition in images and video frames. In *Journal of the Pattern Recognition Scociety*, volume 37, pages 595–608, 2004.

[37] D. Zhang, R. Rajendran, and S. Chang. General and domain-specific techniques for detecting and recognizing superimposed text in video. In *Proceedings of IEEE International Conference on Image Processing*, volume 1, pages I–593 – I–596, 2002.

[38] Gartner group 2006 press release. http://www.gartner.com/it/page.jsp?id=498310.

[39] G. Nagy. Twenty years of document image analysis in pami. In *IEEE Transactions on Pattern Analysis and Machine Intelligence*, volume 22, pages 38–62, 2000.

[40] A. Zandifar, R. Duraiswami, A. Chahine, and L.Davis. A video based interface to textual information for the visually impaired. In *Proc. of fourth IEEE International Conference on Multimodal Interfaces*, pages 325–330, 2002.

[41] N. Ezaki, M. Bulacu, and L.Shoemaker. Text-detection from natural scene images: Towards a system for visually impaired persons. In *Proc. of the 17thh International Conference on Pattern Recognition (ICPR04)*, pages 1051–4651, 2004.

[42] N. Ezaki, K. Kiyota, B. T. Minh, M. Bulacu, and L. Schomaker. Improved text-detection methods for a camera-based text reading system for blind persons. In *Proc. of the Eighth International Conference on Document Analysis and Recognition (ICDAR05)*, pages 257–261, 2005.

[43] J. Liang, D. Menthon, and D. Doermann. Camera-based document image mosaicing. In *Proc. of the 18th International Conference on Pattern Recognition (ICPR06)*, pages 476–479, 2006.

[44] S. Uchida, M. Iwamura, S. Omachi, and K. Kise. Ocr fonts revisited for camera-based character recognition. In *Proc. of the 18th International Conference on Pattern Recognition (ICPR06)*, pages 1134–1137, 2006.

[45] S. Omachi, M. Iwamura, S. Uchida, and K. Kise. Affine invariant information embedment for accurate camera-based character recognition. In *Proc. of the 18th International Conference on Pattern Recognition (ICPR06)*, pages 1098–1101, 2006.

[46] K.S. Bae, K. K. Kim, Y. G. Chung, and W. P. Wu. Character recognition system for cellular phone with camera. In *Proc. of 29th Annual International Computer Software and Applications Conference (COMPSAC 2005)*, volume 2, pages 539–544, 2005.

[47] J. Gao and J. Yang. An adaptive algorithm for text detection from natural scenes. In *Proc. of the 2001 IEEE Conference on Computer Vision and Pattern Recognition (CVPR01)*, volume 2, page 84, 2001.

[48] M. Mirmehdi, P. Clark, and J. Lam. Extracting low resolution text with an active camera for ocr. In *Proc. of the IX Spanish Symposium on Pattern Recognition and Image Processing*, pages 43–48, 2001.

[49] P. Clark and M. Mirmehdi. Estimating the orientation and recovery of text planes in a single image. In *Proc. of British Machine Vision Conference (BMVC01)*, 2001.

[50] S. Wachenfeld, S. Fleischer, and X. Jiang. A multiple classifier approach for the recognition of screen-rendered text. In *Proc. of International Conference of Pattern Recognition (ICPR06)*, pages 1086–1089, 2006.

[51] S. Wachenfeld, H. U. Klein, and X. Jiang. Segmentation of very low resolution screen-rendered text. In *Proc. of the 9th International Conference on Document Analysis and Recognition (ICDAR07)*, volume 2, pages 1153–1157, 2007.

[52] S. Wachenfeld, H.U. Klein, and X. Jiang. Recognition of screen-rendered text. In *Proc. of International Conference of Pattern Recognition (ICPR06)*, pages 1086–1089, 2006.

[53] S. Wachenfeld, H. U. Klein, and X. Jiang. Annotated databases for the recognition of screen-rendered text. In *Proc. of the 9th International Conference on Document Analysis and Recognition (ICDAR07)*, volume 1, pages 272–276, 2007.

[54] S. Wachenfeld, H. U. Klein, and X. Jiang. The official screen-char and screen-word database website, 2006.

[55] L. Wang and T.Pavlidis. Direct gray-scale extraction of features for character recognition. In *IEEE Trans. Pattern Analysis and Machine Intelligence*, volume 15, pages 1053–1067, 1993.

[56] J. Andre and R.D. Hersch. Teaching digital typography. In *Electronic Publishing, Artistic Imaging and Digital Typography*, volume 5(2), pages 79–89, 1992.

[57] D. E. Knuth. *Digital Typography.* CSLI Pulications, 1999.

[58] Fontlab typographic tools. http://www.fontlab.com/index.php?option= com_content& task=blogcategory&id=80&Itemid=39/.

[59] Fontlab typographic tools. http://www.fontlab.com/index.php?option=com_content& task=blogcategory&id=77&Itemid=104/.

[60] Adobe postscript 3. http://www.adobe.com/products/postscript/, 2007.

[61] Fontlab and opentype specifications: overview. http://www.microsoft.com/typography/SpecificationsOverview.mspx, 2004.

[62] R.D. Hersch. Character generation and grid constaints. In *Proc. of SIGGRAPH87, ACM Computer Graphics*, volume 21, pages 243–252, 1987.

[63] P. Karow. Digital formats for typefaces. In *URW Verlag*, 1987.

[64] P. Karow. Apple computer, truetypespec. In *The TrueType font format specification*, 1990.

[65] R.D. Hersch and C. Betrisey. Model-based matching and hinting of fonts. In *Proc. of ACM Computer Graphics*, volume 25(4), pages 71–80, 1991.

[66] J. Herz and R.D. Hersch. Towards an universal auto-hinting system for typographic shapes. In *Electronic Publishing*, volume 7(4), pages 251–260, 1994.

[67] R.D. Hersch, C. Bétrisey, Justin Bur, and André Gürtler. Perceptually tuned generation of grayscale fonts. In *IEEE Computer Graphics and Applications*, volume 1, pages 537–540, 1996.

[68] Deep blue (chess computer). http://en.wikipedia.org/wiki/IBM_Deep_Blue.

[69] Computers outperform humans at recognizing faces in recent tests. http://www.technologyreview.com/Infotech/18796/?a=f.

[70] K. Chellapilla, K. Larson, P. Simard, and M. Czerwinski. Computers beat humans at single character recognition in reading based human interaction proofs (hips). http://research.microsoft.com/displayArticle.aspx?id=1293.

[71] D.H. Ballard and C.M. Brown. *Computer Vision*. Prentice Hall, 1982.

[72] T.M. Ha and H. Bunke. Handwritten numeral recognition by perturbation method. In *Proc. of the Fourth Int. Workshop on Frontiers of Handwriting Recognition*, volume 13, pages 97–106, 1994.

[73] A. Zramdini and R. Ingold. Optical font recognition using typographical features. In *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, volume 2, No. 8, pages 877–882, 1998.

[74] O. Pichler, A. Teuner, and B.J. Hosticka. A comparison of texture feature-extraction using adaptive gabor filtering, pyramidal and tree-structured wavelet transforms. In *Pattern Recognition*, volume 29, pages 733–742, 1996.

[75] J.M.H. du Buf and P. Heitkemper. Texture features based on gabor phase. volume 23, pages 227–244, 1991.

[76] B. Jähne. *Practical Handbook on Image Processing for Scientific Applications.* CRC Press, 1997.

[77] R.C. Gonzalez and R.E. Woods. *Digital image processing.* Addison Wesley, 1992.

[78] P.D. Picton. Hough transform references. In *Int. Journal of Pattern Recognition and Artificial Intelligence*, volume 1, pages 413–425, 1987.

[79] J. Illingworth and J. Kittler. A survey of hough transform. In *Computer Vision, Graphics and Image Processing*, volume 44, pages 87–116, 1988.

[80] F. Zernike. Beugungstheorie des schneidenverfahrens und seiner verbesserten form, der phasenkontrastmethode (diffraction theory of the cut procedure and its improved form, the phase contrast method). In *Physica*, volume 1, pages 689–704, 1934.

[81] H.Bunke and P. S P.Wang. *Handbook of Character Recognition and Document Image Analysis.* World Scientific, 1997.

[82] R.Schallkoff. *Pattern recognition: statistical, structural and neural approaches.* John Wisley & Sons, 1992.

[83] M. Basu, H. Bunke, and A. Del Bimbo. Guest editor's introduction to the special section on synctactic and structural pattern recognition. In *IEEE Transactions an Pattern Analysis and Machine Intelligence*, volume 27, no.7, pages 1009–2005, 2005.

[84] R.O. Duda and P.E. Hart. *Pattern classification and scene analysis.* John Wisley & Sons, 1973.

[85] L. Rabiner and B.H. Juang. *Fundamentals Of Speech Recognition.* Prentice Hall, 1993.

[86] C. M. Bishop. *Pattern Recognition and Machine Learning.* Springer Science+Business Media LLC, 2006.

[87] Hidden markov model from wikipedia, the free encyclopedia. http://en.wikipedia.org/wiki/hidden_Markov_model.

[88] R. Shinghal and G.T. Touissant. Experiments in text recognition with the modified viterbi algorithm. In *IEEE Transaction of Pattern Analysis and Machine Intelligence (PAMI)*, volume 1, pages 184–192, 1979.

[89] C.B. Bose and S.S. Kuo. Connected and degraded text recognition using hidden markov models. In *Journal of Pattern Recognition*, volume 27, pages 1345–1363, 1994.

[90] Z. Lu, I. Bazzi, A. Kornai, and J. Makhul. A robust, language-independent ocr system. In *Proc. 27th AIPR Workshop: Advances in Computer-Assisted Recognition SPIE*, 1999.

[91] P. Velagapudi. Using hmms to boost accuracy in optical character recognition. In *Proc. of SPIE, 27th AIPR Workshop: Advances in Computer-Assisted Recognition*, volume 3584, pages 96–104, 1999.

[92] J. Hu, M. Brown, and W. Turin. Hmm-based on-line handwriting recognition. In *IEEE Transactions on Pattern Analysis ans Machine Intelligence*, volume 18, pages 1039–1045, 1996.

[93] T. Starner, J. Makhul, R. Schwarz, and G. Chou. On-line cursive handwriting recognition using speech recognition methods. In *Proc. of IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP07)*, volume v, pages 125–128, 2007.

[94] J. Tokuno, N. Inami, S. Masuda, M. Nakai, H. Shimodaira, and S. Sagayama. Context-dependent substroke model for hmm-based on-line handwriting recognition. In *Proc. of Eighth International Workshop on Frontiers in Handwriting Recognition (IWFHR02)*, page 78, 2002.

[95] A. Funada, D. Muramatsu, and T. Matsumoto. The reduction of memory and the improvement of recognition rate for hmm on-line handwriting recognition. In *Proc. of Ninth International Workshop on Frontiers in Handwriting Recognition (IWFHR04)*, pages 383–388, 2004.

[96] E. J. Bellegarda, J. R. Bellagarda, D. Nahamoo, and K. Nathan. On-line cursive handwriting recognition using speech recognition methods. In *Proc. of IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP95)*, volume 4, pages 2631 – 2634, 1995.

[97] F. Biasdy, J. El-Sana, and N. Habash. Online arabic handwriting recognition using hidden markov models. In *Proc. of Tenth International Workshop on Frontiers in Handwriting Recognition*, 2006.

[98] D. Okumura, S. Uchida, and H. Sakoe. An hmm implementation for on-line handwriting recognition based on pen-coordinate feature and pen-direction feature. In *Proc. of Eigth International Conference on Document Analysis and Recognition (ICDAR05)*, volume 1, pages 26–30, 2005.

[99] J. P. Shin and H. Sakoe. Stroke correspondance search for stroke-order and stroke-number free on-line character recognition – multilayer cube search –. In *IECE Transactions*, volume J82-D-II, no.2, pages 69–81, 2004.

[100] A. Kosmala, D. Willett, and G. Rigoll. Advanced state clustering for very large vocabulary hmm-based on-line handwriting recognition. In *Proc. of Fifth International Conference on Document Analysis and Recognition (ICDAR99)*, pages 442–445, 1999.

[101] A. Humm, J. Hennebert, and R. Ingold. Hidden markov models for spoken signature verification. In *proc. of IEEE Conference on Biometrics: Theory, Applications and Systems (BTAS07)*, pages 1–6, 2007.

[102] M. Mohamed and P. Ghader. Handwritten word recognition using segmentation-free hidden markov modeling and segmentation-based dynamic programming techniques. In *IEEE Transcations on Pattern Analysis and Machine Intelligence*, volume 18, pages 548–556, 1996.

[103] M. Chen, A. Kundu, and J. Zhou. Offline handwritten word recognition using a hidden markov model type stochastic network. In *IEEE Transcations on Pattern Analysis and Machine Intelligence*, volume 16, pages 481–496, 1994.

[104] A. Vinciarelli, S. Bengio, and H. Bunke. Offline recognition of unconstrained handwritten texts using hmms and statistical language models. In *Transcations of PAMI, IEEE*, pages 709–720, 2003.

[105] M. Zimmermann and H.Bunke. N-gram language models for offline handwritten text recognition. In *Proc. of 9th Int. Workshop on Frontiers in Handwriting Recognition (IWFHR04)*, pages 203–208, 2004.

[106] U. V. Marti and H. Bunke. Using a statistical language model to improve the performance of an hmm-based cursive handwriting recognition system. In *Journal of Pattern Recognition and Art. Intelligence*, volume 15, pages 65–90, 2000.

[107] A. Schlapbach and H. Bunke. Using hmm-based recognizers for writer identification and verification. In *Proc. of 9th Int. Workshop on Frontiers in Handwriting Recognition (IWFHR04)*, pages 167–172, 2004.

[108] A. Schlapbach and H. Bunke. Off-line writer verification: a comparison of a hidden markov model (hmm) and a gaussian mixture model (gmm) based system. In *Proc. of 10th Int. Workshop on Frontiers in Handwriting Recognition (IWFHR06)*, pages 275–280, 2006.

[109] Java2 se1.4. http://java.sun.com/j2se/1.4.2/docs/api/java/awt/Image.html /getScaledInstance.

[110] Microsoft typography. http://www.microsoft.com/typography/default.mspx.

[111] M. A. El-Yacoubi, M. Gilloux, and J. M. Bertille. A statistical approach for phrase location and recognition within a text line: an application to street name recognition. In *IEEE Transactions on Pattern Analysis and Machine Intelligence*, volume 24, No. 2, 2002.

[112] S. A. Santoshkumar and V. Ramasubramanian. Automatic langauage identification using ergodic hmm. In *Proc. of ICASSP05*, pages 455–468, 2005.

[113] Y. Miyazawa, J. I. Takami, S. Sagayama, and S. Matsunaga. An all-phoneme ergodic hmm for unsupervised speaker verification. In *Proc. of Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP94)*, pages I–249 – I–252, 1994.

[114] M. K. Hu. Visual pattern recognition by moment invariants. In *IRE Trans. on Information Theory*, pages 179–187, 1962.

Curriculum Vite

Education

2003-2008: PhD Student at Department of Informatics, University of Fribourg, Switzerland

1989-1990: Master in Industrial Management at Institut of BWI, Federal Institute of Technology Zurich (ETHZ), Switzerland

1983-1989: Master in Electrical Engineering, Spezialization: Telecommunications at ETHZ, Switzerland

1979-1981: System Engineering at Melli University, Tehran, Iran

1972-1979: Physics-Mathematics gymnasium, Tehran, Iran

Professional Experiences

2000-2006: Lecturer in Telecommunications and Computer Science at Universities of Applied Sciences, Switzerland

1999-2000: Key Account Manager for Telecommunication Networks at Commcare AG, Zurich, Switzerland

1996-1999: Project Manager for internatinal customers at Swisscom AG, Zurich, Switzerland

1993-1996: Maternal leave and birth of Cyrus Einsele (1993) and Semira Einsele (1995)

1990-1993: Systems Engineer and Product Manager for EDI at IBM AG, Zurich, Switzerland

Languages

German: fluent

English: fluent

French: intermediare

Persian: mother tongue

Research Interests

Image processing and document analysis

Camera-based document analysis

Digital typography

Information indexing and retrieval

Publications

- F. Einsele, R. Ingold, J. Hennebert, *A Language-Independent, Open-Vocabulary System for Recognition of Ultra Low Resolution Word Images.* In Proc. of the 2008 ACM symposium on Applied computing, pages 429-433, Fortaleza, Ceara, Brazil, 2008

- F. Einsele, R. Ingold, J. Hennebert, *A HMM-based Approach to Recognize Ultra Low Resolution Anti-Aliased Words.* In Lecture Notes in Computer Science, volume 4815, pages 511-518, Springer Berlin, Heidelberg,, 2007

- F. Einsele, R. Ingold, J. Hennebert, *Recognition of Ultra Low Resolution Word Images Using HMMs.* In Advanced in Soft Computing, volume 45, pages 429-436, Springer Berlin, Heidelberg, 2008

- F. Einsele, J. Hennebert, R. Ingold, *Towards Identification of Very Low Resolution, Anti-Aliased Characters.* In proc. of IEEE International Symposium on Signal Processing and its Applications (ISSPA07), Sharjah, United Arab Emirates, 2007

- F. Einsele, R. Ingold, *A Study of the Variability of Very Low Resolution Characters and the Feasibility of their Discrimination Using Geometrical Features* In Proc. of 4th World Enformatica Congress, International Conference on Pattern Recognition and Computer Vision (WEC05), pages 213-217 Istanbul, Turkey, 2005