Extracting Textual Information from Images and Videos for Automatic Content-Based Annotation and Retrieval

Dissertation

zur Erlangung des akademischen Grades
Doktor der Naturwissenschaften (Dr. rer. nat)

dem
Fachbereich Mathematik und Informatik der
Philipps-Universität Marburg
vorgelegt von

Julinda Gllavata

aus Durrës, Albanien

Marburg/Lahn, 2007

Vom Fachbereich Mathematik und Informatik der
Philipps-Universität Marburg als Dissertation am
29. Januar 2007 angenommen.

Erstgutachter: Prof. Dr. Bernd Freisleben, Philipps-Universität Marburg

Zweitgutachter: Prof. Dr. Manfred Sommer, Philipps-Universität Marburg

Tag der mündlichen Prüfung: 5. Februar 2007

Dedication

To my lovely parents Esi and Meti.
Prindërve të mi të dashur Esit dhe Metit.

Acknowledgements

First of all, I would like to express my gratitude to my advisor Prof. Dr. Bernd Freisleben for his invaluable support, guidance and encouragement throughout this research project. Without the helpful discussions, suggestions and insights, this work would not have been completed.

I would like to thank Prof. Dr. Manfred Sommer for accepting to act as the second reviewer of this thesis.

The research work presented in this thesis was financially supported by the Deutsche Forschungsgemeinschaft (SFB/FK 615, Teilprojekt MT) and by Deutscher Akademischer Austausch Dienst (DAAD, Stability Pact for South Eastern Europe). The support of these two institutions is gratefully acknowledged. In this context, I wish to thank Prof. Dr. Mira Mezini and Dr. Jochen Münch for their support.

I want to thank all the people at the Department of Mathematics and Computer Science of the University of Marburg for supporting me during my time here. In particular, I would like to thank my colleague Ralph Ewerth for the helpful discussions, joint collaborative research and for carefully reading parts of this thesis. I also would like to thank our secretary Mechthild Keßler for offering me her help when needed.

Many thanks also go to all my friends, who were so enthusiastic in their support during the preparation of this thesis and for cheering me up during much-needed study breaks.

Lovely thanks go also to Artan Gllavata for proof-reading part of this thesis.

I wish to express sincere appreciation and love to Ermir, for his patience, understanding and support during all these years. I would also like to thank him for the collaborative discussions that we had and for spending time for proof-reading this thesis.

Last, but not least, I am indebted to my whole family, especially my father Meti and my mother Esi. Their unwavering faith in me gave me the confidence to pursue my ambitions, and I will forever be thankful for all they have done.

The following sentence is written for them in Albanian. Ju falenderoj nga zemra të gjithëve për mbështetjen e madhe që më keni dhënë.

Abstract

One way to utilize semantic knowledge for annotating databases of digital images and videos is to use the textual information which is present. Usually, it provides important information about the content and is a very good entity for queries based on keywords. In this context, the extraction of scene and artificial text from images and videos is an important research problem, with the aim of achieving automatic content-based retrieval and summarization of the visual information. The process of text extraction includes several steps:

- *Text detection* is aimed at identifying image parts containing text.
- *Text localization* merges text regions which belong to the same text candidate and determines the exact text positions.
- *Text tracking* tracks the localized text over successive frames in a video.
- *Text segmentation and binarization* include the separation of the localized text from the image background. The output of this step is a binary image where black text characters appear on a white background.
- *Character recognition* performs optical character recognition (OCR) on the binarized image and converts the binarized image to ASCII text.

In this thesis, a robust system for automatically extracting text appearing in images and videos with complex background is presented. Different algorithms are proposed addressing solutions to different steps of the text extraction process mentioned above. The system can operate on JPEG images and MPEG-1 videos. The tracking of the text appearing in videos is also addressed and a novel algorithm is presented. Individual and comparative experimental results demonstrate the performance of the proposed algorithms for the main processing steps: text detection, localization and segmentation, and in particular, their combination.

Text in images or videos can appear in different scripts, such as Latin, Ideographic, Arabic, etc. The identification of the used script can help in improving the segmentation results and in increasing the accuracy of OCR by choosing the appropriate algorithms. Thus, a novel technique for script recognition in complex images is presented.

Content-based media retrieval has received a lot of attention during the last years and query by example is the most used methodology. In this context, it may be of interest to search for images of video frames where a text *visually similar* with the input text image appears. Thus, a novel technique that deals with the holistic comparison of text images is proposed. Recently, relevance feedback methods have attracted researchers due to the possibility they offer to interact with the user to increase the performance of a content-based image retrieval (CBIR) system. However, due to the increasing number of images and the need of the user to explore the media before taking a decision, the employment of techniques to visualize or browse a collection of images is becoming important. Consequently, several visualization/browsing methods are proposed to facilitate the interactive exploratory analysis of large image data sets and assist the user during the semantic search.

Zusammenfassung

Die in digitalen Bildern und Videos vorhandene textuelle Information bietet eine hervorragende Möglichkeit, um semantisches Wissen in den Prozess der Indexierung von Bild- und Videodatenbeständen einfließen zu lassen. Die Verbindung dieser Information mit dem Inhalt der digitalen Medien ermöglicht wortbasierte Abfragen, die diese textuelle Information ausnutzen. Deshalb ist die Textextraktion aus Bildern und Videos im Rahmen von automatischen inhaltsbasierten Suchsystemen von großer Bedeutung.

Die Textextraktion aus Bildern und Videos besteht aus folgenden Schritten.

- *Die Textdetektion* definiert den Prozess der Identifizierung der Regionen in Bildern, in denen Text erscheint.
- *Die Textlokalisierung* baut auf der Textdetektion auf und verschmilzt die gleichem Text zugehörigen Regionen zwecks Bestimmung der exakten Textposition.
- *Die Textverfolgung* in Videos realisiert die Verfolgung von zuvor lokalisiertem Text über mehrere aufeinander folgende Einzelbilder hinweg.
- *Die Textsegmentierung und Textbinärisierung* ist der Prozess der Trennung der Textpixel und Hintergrundpixel. Die Ausgabe dieses Schritts ist ein binäres Bild, in dem die Zeichen schwarz auf einem weißen Hintergrund erscheinen.
- *Die Zeichenerkennung* verfolgt das Ziel der Extraktion von ASCII-Text aus einem binären Bild mittels optischer Zeichenerkennung.

Diese Arbeit stellt ein robustes System für die automatische Extraktion von Text in Bildern und Videos vor. Verschiedene Algorithmen werden für jedes der oben genannten Probleme präsentiert. Das System kann sowohl mit JPEG Bildern als auch mit MPEG-1 Videos arbeiten. Die experimentellen

Ergebnisse dokumentieren die Güte der einzelnen Schritte und deren Kombination.

Da Text in Bildern in unterschiedlichen Schriften (z. B. ideographische Schrift oder lateinische Schrift) erscheinen kann, ermöglicht die vorherige Erkennung der Schrift eine bessere Textsegmentierung oder Texterkennung. Für diesen Zweck wird eine Methode zur Schrifterkennung in Bildern mit komplexem Hintergrund vorgestellt.

Des Weiteren ist eine neue Methode entwickelt worden, um den holistischen Vergleich zwischen Textbildern zu ermöglichen. Im Rahmen der inhaltsbasierten Suche sind solche Ansätze von Interesse, um die Suche nach Bildern mit ähnlichen Textvorkommen zu vereinfachen. Außerdem gewinnt die Suche anhand von Beispielen im Rahmen von inhaltsbasierter Suche zunehmend an Bedeutung. Seit Kurzem sind Relevanz-Feedback-Verfahren in den Blickpunkt des Interesses gerückt, da sie Benutzern die Möglichkeit bieten, mit dem System zu interagieren. Darüber hinaus wächst der Bedarf für Methoden zur Visualisierung und Exploration (Browsing) von Bilddatenbeständen, begründet durch deren zunehmende Größe und dem daraus resultierenden Benutzerinteresse, schnell und einfach diese großen Bestände durchsuchen zu können. Daher werden neue Methoden vorgeschlagen, die den Benutzer während dieses semantischen Suchprozesses unterstützen.

Contents

List of Figures

List of Tables

List of Algorithms

Chapter 1

Introduction

When something can be read without effort,
great effort has gone into its writing.
- Enrique Jardiel Poncela -

1.1 Text in Images and Videos

With the growing number of digital multimedia libraries, the need to efficiently index, browse and retrieve multimedia information is increased. Several approaches have been developed for indexing, querying and retrieving multimedia information. One possibility is to use the textual information embedded in the multimedia data. This offers important information for multimedia data understanding and is a very good entity for keyword-based queries.

In this context, text extraction has gained a lot of attention in the last years in several applications. Jung et al. [JKJ04] distinguish between four main classes of applications: (I) page segmentation (which include newspaper headlines extraction [TL04, JZ96, TLS96]); (II) address block location [YJM97]; (III) license plate location [CH97, KC01]; and (IV) content-based image/video indexing [ZGST94, SDB98]. Although a lot of work has been done, it is still not easy to design domain independent text extraction systems. This is because there are so many possible sources of variation when extracting text with different fonts, size, color, orientation and alignment, which could possibly be embedded in shaded or complex backgrounds. These variations make the problem of text extraction very challenging.

In Figures 1.1, 1.2, 1.3 and 1.4 images extracted from different domains are shown to illustrate different properties of text in the respective domains.

The images shown in Figure 1.1 are known as document images, and many approaches [JZ96, TLS96] have been proposed to treat the problem of text extraction in the case of this type of images. Although the images shown in Figure 1.2 are similar to those in Figure 1.1, the methods used for document image analysis cannot be directly applied to them. Accordingly, Jung et al. [JKJ04] have distinguished this category of images as multi-color document images. Perroud et al. [PSBH01] and Hase et al. [HYT+04] have proposed methods that deal with the localization of text in multi-color document images.

The other category of images consists of videos and scene images, which are illustrated in Figures 1.3 and 1.4, respectively. The text occurring in this category of images will be the focus of this thesis. Compared to the texts in document images, the texts that appear in scene images or videos come in a much smaller quantity. However, they usually carry important information about the content of the media. Visual texts often impart knowledge about anchors' names, place locations, brands of products, scores of a match, date and time when an event took place. All this information is crucial for understanding and indexing scene images and videos.

Current research in the field of text extraction in scene images/videos focuses on developing appropriate algorithms to extract different types of text from complex images or videos with the purpose of enabling the automatic content based indexing of images/videos using the results obtained from the text extraction algorithms. The whole text extraction process is illustrated in Figure 1.5.

1.1.1 Artificial and Scene Text

Text appearing in images and videos is usually categorized into two main groups according to the source where it comes from:

- *artificial text* (referred also as *caption text* or *superimposed text*)
- *scene text* (referred also as *graphics text*)

Artificial text (see Figure 1.4) is laid over the image at a later stage (e.g. the name of someone during an interview, the name of the speaker, etc.). In contrast to artificial text, scene text (in Figure 1.3) is part of the scene (e.g. the name of a product during a commercial break, etc.) and is usually more difficult to extract due to different alignment, size, font type, light condition, complex background and distortion. Moreover, it is often affected by variations in scene and camera parameters, such as illumination, focus, motion, etc.

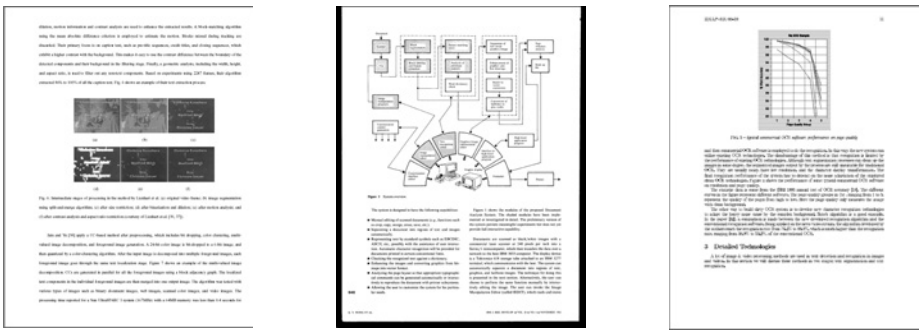


Figure 1.1: Different samples of document images

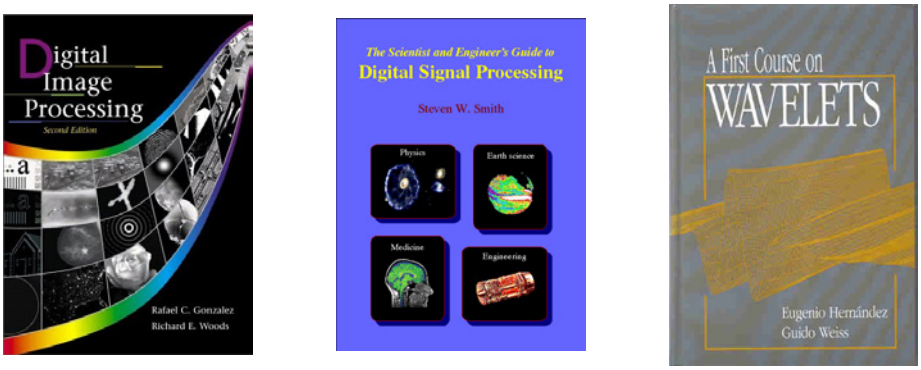


Figure 1.2: Different samples of multi-colored document images

1.1.2 Characteristics of Text

Text in images can exhibit many variations with respect to the following properties (as listed in [JKJ04]):

1. **Geometrical Properties**:

   - *Size*: The size of the text can vary in different ranges. Depending on the language, an English text requires at least 8-pixels-high font size, while a Chinese text requires at least a 20-pixels-high font size, due to the large number of strokes [NC05].
   - *Alignment/Orientation*: As previously mentioned, scene text can be aligned in arbitrary directions and can have perspective distortions. Artificial text is usually horizontally aligned, although sometimes due to special effects, it can appear as a non-planar text.

Figure 1.3: Different samples of complex images with scene text

Figure 1.4: Different samples of complex images with artificial text

- *Inter-Character Distance*: Characters within a word usually have an uniform distance between them. However, an Ideographic script may show a different inter-character distance compared to a Latin script.
- *Aspect Ratio*: Characters of Latin languages show different aspect ratios compared to those of Chinese.
- *Stroke Density*: The stroke density of English characters is approximately uniform, whereas that of Chinese characters may vary significantly (from 1 to more than 20).
- *Stroke Statistics*: English characters consist mainly of vertical and horizontal strokes, whereas Chinese characters have four directional strokes.

2. **Color Relationship**:

   - *Color*: Text strings can have different colors, but the characters belonging to an artificial text usually have a homogenous color (monochrome). However, scene text can contain characters with different colors (polychrome) or sometimes characters with the same color, but with very different illuminations, as illustrated in Figure 1.3.

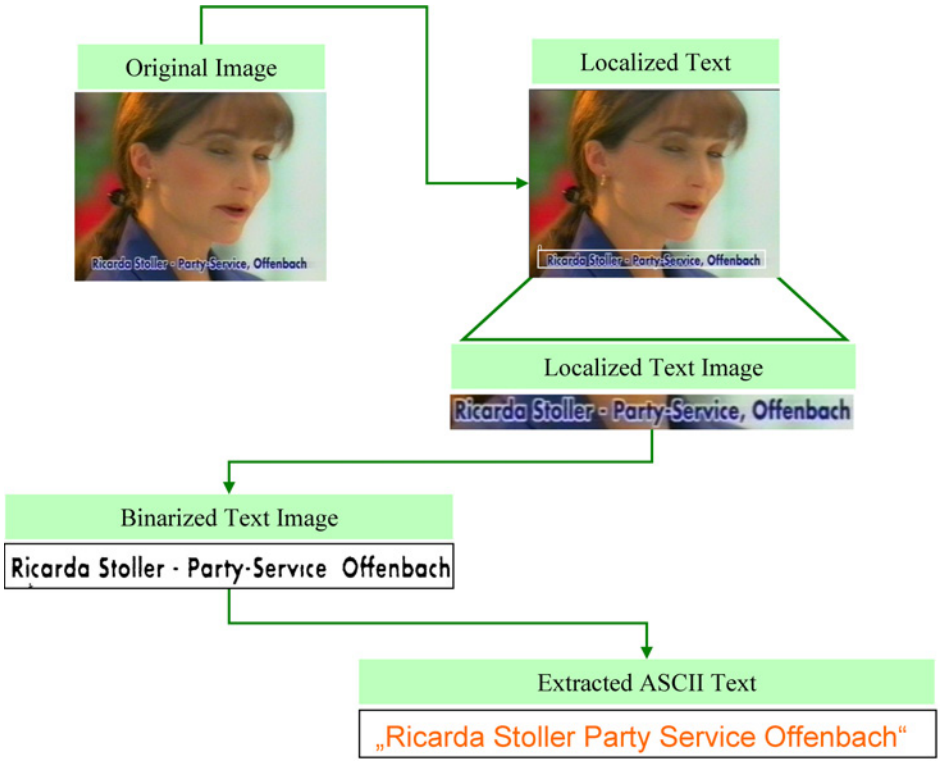


Figure 1.5: Visualization of text extraction process

- *Text Polarity*: This property describes whether the text has a dark color compared to that of the background or vice versa. There are two types of polarities namely positive and negative. When the text color is light and the background color is dark, the text polarity is described as positive and as negative in the reversed scenario.

3. **Motion**: Text appearing in a video is present in more than one frame, and this is associated with or without movement of the text. Artificial text usually moves linearly in a horizontal or vertical way, whereas scene text can move also in a non-rigid way.

4. **Edge/Contrast**: The contrast that the text shows against the background (in order to be easily readable by the human) results in strong edges at the boundaries of text and the background.

5. **Compression**: Most of the digital images and videos are stored in a compressed format. This property can be used to increase the speed of the text extraction methods and is usually used by text tracking algorithms.

6. **Background**: The complexity of the background where the text appears in can vary from simple to considerably difficult. The background can be composed of different colors, textures and other objects may be embedded in it as well.

Table 1.1 summarizes the explained properties ([JKJ04]) of text in video/scene images.

1.2 Content-Based Indexing and Retrieval

Content-based image indexing refers to the process of attaching labels and signatures to images based on their content and extracted features, respectively, whereas content-based image retrieval (CBIR) is aimed at the efficient retrieval of relevant images from large image databases, based on the attached labels. Features can be categorized into two main groups: *perceptual (low-level)* and *semantic (high-level)* features. Perceptual features include attributes such as color, intensity, shape, texture, and their temporal changes, whereas semantic features describe the present objects (such as vehicles, text, faces, etc.), events, and their relations (more details are given in Chapter 2).

Compared to low-level features and to other high-level features, the text provides clearer and more obvious information about the content of specific media. It is a powerful source of information about the content of an image. Artificial text in news videos usually provides information about the name of related people e.g. anchors, location, subject of discussion, date and time when an event has occurred. Artificial text often provides an abstract of the program and is often not available in other media like the audio "track". Titles and credits displayed at the beginning or end of movies provide information like names of actors, producers, contributors, etc. Captions in sport programs often contain the names of the teams, players, scores, etc. Text information can also be found in the scene, e.g. the players' numbers and names, the name of the team, brand names, location and commercials. Displayed maps, figures and tables in videos contain text about locations, temperatures, and certificate items. Titles, logos, and names of programs displayed in videos are important for the annotation with respect to program types and names. Moreover, more and more web sites choose text pictures to improve the design of their web presentations.

In this context, text appearing in images in general and artificial text in particular is of special interest due to two reasons:

- Text is very useful for describing the content of an image.
- The successful extraction of text enables automatic text based annotation and subsequent keyword-based searching or content oriented

processing.

Table 1.1: Different properties of text in images and videos

Property		Variants or sub-classes
Geometry	Size	Regularity in size of text
	Alignment	Horizontal/vertical
		Straight line with skew (implies vertical direction)
		Curves
		3D perspective distortion
	Inter-Character distance	Aggregation of characters with uniform distance
	Aspect Ratio	
	Strokes	Different stroke density and statistics
Colors relationship	Color	Gray
		Color (monochrome, polychrome)
	Text Polarity	Text color is dark or light
Motion		Static
		Linear movement
		2D rigid constrained movement
		3D rigid constrained movement
		Free movement
Edge		Strong contrast (edges) at text boundaries
Compression		Un-compressed image JPEG, MPEG-compressed image
Background		The background can be different

1.3 The MEDIANA Project

The work presented in this thesis is done in the framework of a large media research project entitled "Media Upheavals", which is currently being conducted at the University of Siegen, University of Marburg, University of Dortmund and Fraunhofer Gesellschaft St. Augustin in Germany. The project is funded by the Deutsche Forschungsgemeinschaft (SFB/FK615).

The project is divided into thirteen subprojects in which a total of about 80 people perform research or administrative tasks. It is aimed at investigating the foundations and the structural aspects of the comprehensive media changes and their impact on the appearances and changes of media culture and the development of media aesthetics at the beginning of the 20^{th} century and in the transition to the 21^{st} century. In particular, the impact of media technology changes, such as the introduction of film and cinema around 1900 and the Internet and World Wide Web towards the end of last century, are considered. For example, the subproject "Industrialization of Perception" investigates the impact of movies on the social perception context of people in the years between 1895 and 1914 in Germany. Only for this particular project, a total of 15.000 films from this time are still available, and a comprehensive random sample set will be exposed to scientific film analysis. This project seeks to study the hypothesis that the fact that industrialization made "things happen faster", can be observed in the evolution of film during the period of years 1895-1914.

The goal of our subproject named MT ("Methoden und Werkzeuge zur rechnergestützten medienwissenschaftlichen Analyse") is to provide a high-performance video content analysis system to support the above subproject and other subprojects which require semantic analysis of the films. The software workbench Mediana [LFG+02, EGG+03, EF06a] is currently under development to provide media scientists with the possibility to automatically analyze the content of the films. Mediana includes several algorithms for video content analysis, including shot boundary detection [EF03, EF04b, EF04a, EBK+05, EF06c], text detection, localization [GEF03a, GEF03b, GEF04b, GEF04a, GF06] and text segmentation [GESF04, GF05a], camera motion estimation [ESTF04], face detection and recognition [VJ01, EF06b, EMF06], and audio processing i.e. speaker classification [SF06]. In addition, database tools for the organization of heterogeneous data sources are also included.

This thesis is focused on the different algorithms developed to deal with the challenging problem of text detection, localization and segmentation in images and videos. In addition, algorithms for analyzing different properties of text, such as determining the script of the present text and matching visually two different texts, will also be introduced. Finally, a novel image browsing technique for facilitating content based image retrieval will be presented.

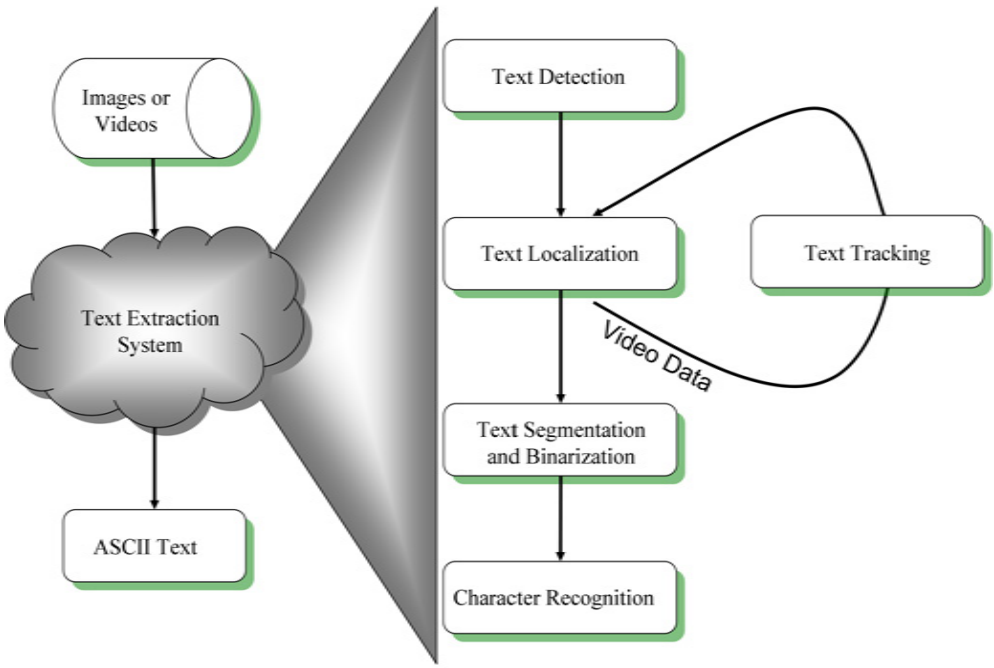


Figure 1.6: The architecture of a text extraction system

1.4 Problem Statement

This section discusses the definition of different problems as addressed by this thesis. The input of the discussed problems can be a colored or gray-scale image or a video stream, respectively. The scene content is unconstrained and may be both indoor or outdoor, under arbitrary lighting or contrast conditions. The object of all discussed problems is the text that appears in input images or videos, in many different languages or scripts.

1.4.1 Text Extraction

Text extraction (TE) is a complex problem which includes many other difficult sub-problems, namely text detection, text localization, text tracking, text segmentation and binarization, and character recognition. The architecture of a text extraction system is shown in Figure 1.6. All these sub-problems will be explained shortly in the following paragraphs and in more detail in the Chapters 3, 5 and 6.

1.4.1.1 Text Detection

Text detection (TD) takes as input a raw image. Its aim is to decide if there is any text present in the image and if so, to give as output the parts of image

containing text. The TD problem is addressed in Chapter 3.

1.4.1.2 Text Localization

Text localization (TL) takes as input the output of the TD step and merges text regions which belong to the same text candidate. Its final task is to determine the exact coordinates of the text position. Three different solutions to this problem are discussed in Chapter 3.

1.4.1.3 Text Tracking

Text often spans over thirty or even hundreds of frames in a digital video. In such cases, text tracking (TT) algorithms are used to exploit the temporal occurrence of text over a sequence of video frames. This process is necessary to reduce the processing time, by not applying the TDL algorithms to every frame separately. A solution to this problem is discussed in Chapter 5.

1.4.1.4 Text Segmentation and Binarization

After the text has been localized, the text segmentation and binarization (TSB) step deals with the separation of the text pixels from the background pixels. The output of this step is a binary image where black text characters appear on a white background. This process, which is covered in Chapter 6, is very important in order to achieve accurate results during subsequent character recognition tasks.

1.4.1.5 Character Recognition

Character recognition (CR) is the final step of the text extraction process. It performs optical character recognition (OCR) on the binarized text image and converts the binarized text image into the corresponding ASCII text. The recognized ASCII text is a good entity to index the image/video using the present textual information. The recognition step is usually language dependent, and it is assumed that the language is known beforehand. Although this is an important step towards an automatic image/video indexing process, it is beyond the scope of this thesis. In this thesis, it is assumed that the binarized text will be recognized by a standard OCR package. This problem has been mentioned here only for the sake of completeness.

1.4.2 Script Recognition

The text that appears in images or videos can be of a different script, such as Latin, Ideographic, etc. Languages such as English, German, Albanian, Ital-

ian, French etc., are classified under Latin script, whereas Chinese, Japanese and Korean languages are known as an Ideographic script.

The process of script recognition (SR) deals with the problem of identifying the category of the script that a localized text belongs to, and is the subject of Chapter 7. The identification of the used script can help in improving the segmentation results and in increasing the accuracy of the OCR by choosing the appropriate algorithms.

1.4.3 Text Matching

In the content image/video retrieval domain, it may be interesting to have the possibility to search for images of video frames where a text *visually similar* with the input text image appears. In our case, *visually similar* means that the two text images look similar to each other to a certain degree, which does not necessary imply that the same text appears in them.

Given two different text images, a text matching (TM) algorithm tries to evaluate the visual similarity between these two text images. This problem is covered in Chapter 8.

1.4.4 Semantic Browsing of Images

Content-based media retrieval has received a lot of attention during the last years and query by example is the most used methodology. Recently, relevance feedback methods have attracted researchers due to the possibility they offer to interact with the user in order to increase the performance of a CBIR system. However, due to the increasing number of images and the need of the user to explore the media before taking a decision, the employment of techniques to visualize or browse a collection of images is becoming important.

In this context, browsing methods with the aim to facilitate the interactive exploratory analysis of image data sets and assist the user during semantic search are discussed in Chapter 9.

1.5 Contributions of this Thesis

This thesis investigates methods for building an efficient system for extracting text embedded in images and videos. The proposed methods do not make any assumptions about the color, position, language, size and font of text. In addition to text extraction methods, algorithms are proposed to analyze certain patterns of the localized text. The contributions of this thesis are as follows:

1. Three novel methods for text detection and localization are proposed. The first two methods localize the embedded text through the analysis of the horizontal projection profile of the pre-calculated edge image. The first one is a fixed threshold based method, whereas the second uses an adaptive threshold of the profile to localize the text. The third method is a texture based method, which means that a set of texture features is used to distinguish between possible text and non-text areas. Text detection is carried out using an (unsupervised) clustering method (avoiding in this way the need for training data) to classify image blocks into three groups: "text", "complex background" and "simple background". Then, an adaptive projection-based method is used again to find the exact position of the text. A simple text verification process is also proposed.

2. The problem of text segmentation in complex images is also addressed, and two new unsupervised text segmentation methods are introduced. Both of the methods use the color components and different texture features to separate the text pixels from the background pixels. The first method, after determining the possible color of the text, uses a k-means clustering method to classify the data into two clusters: "text" and "non-text". The second method employs an adaptive number of clusters for different text images combined with a fuzzy clustering algorithm, improving in this way the text segmentation results significantly. A voting algorithm based on various textual characteristics is also presented to select the text cluster.

3. A novel fuzzy ensemble based text detection method is proposed which integrates temporal information of text in video at the raw (grayscale or RGB color space) image level. The presence of static text in multiple successive frames (in the same position) is exploited by applying a fuzzy ensemble method.

4. A new algorithm to track the temporal occurrence of the moving text within a group of pictures is proposed. The MPEG motion vector information extracted from the current frame is employed in order to predict the position of text in the next frame.

5. The recognition of the script of the localized text is necessary to choose the appropriate text segmentation method and the OCR engine. Thus, a supervised method is proposed to recognize the script after the text has been localized. A machine learning approach, namely k-nearest neighbour is used to classify the script into Latin or Ideographic script, based on a set of low level features.

6. A new method to visually compare pairs of text images is proposed. First, the salient points in each of the images are estimated using a slightly modified version of the Harris/Stephens [HS88] corner detector. Second, the best mapping is determined by applying the Scott and Longuet-Higgins (SLH) [SLH91] method or the sequential correspondence finding (SCF) algorithm. The SLH algorithm models the correspondence between the set of salient points as an affine transformation and uses the properties of singular value decomposition to find the corresponding pairs, while SCF is based on spatial geometrical constraints and different distance measurements to define the right mapping. Then, the dissimilarity between two given images is defined relying on the established correspondence.

7. Finally, a novel technique called Image Proximity Matrix (IPM) is proposed to visualize the relationships in a collection of images. This IPM view facilitates the interactive exploratory analysis of image data sets and assists the user during the semantic search. The core of the technique consists of displaying the proximity matrix of pair wise images in a tabular form in analogy to the reorderable matrix. Furthermore, two dimension reduction methods are included, namely Multidimensional Scaling and Sammon Mapping to complement the IPM view.

In the framework of the research done in this thesis, the following papers have been published:

1. J. Gllavata, R. Ewerth and B. Freisleben. A Robust Algorithm for Text Detection in Images Proceedings of the International Symposium on Image and Signal Processing and Analysis, Rome, Italy, pages 611-616. IEEE Press, 2003.

2. J. Gllavata, R. Ewerth and B. Freisleben. Finding Text in Images via Local Thresholding, Proceedings of the International Symposium on Signal Processing and Information Technology, Darmstadt, Germany, pages 539-542. IEEE Press, 2003.

3. J. Gllavata, R. Ewerth, T. Stefi and B. Freisleben. Unsupervised Text Segmentation Using Color and Wavelet Features, Proceedings of the International Conference on Image and Video Retrieval, Dublin, Ireland, pages 216-224. Springer Verlag, 2004.

4. J. Gllavata, R. Ewerth and B. Freisleben. Text Detection in Images Based on Unsupervised Classification of High-Frequency Wavelet Coefficients, Proceedings of the International Conference on Pattern Recognition, Cambridge, UK, pages 425-428. IEEE Press, 2004.

5. J. Gllavata, R. Ewerth and B. Freisleben. Tracking Text in MPEG Videos, Proceedings of the Annual ACM International Conference on Multimedia, New York City, USA, pages 240-243. ACM Press, 2004.

6. J. Gllavata, R. Ewerth and B. Freisleben. A Text Detection, Localization and Segmentation System for OCR in Images, Proceedings of the International Symposium on Multimedia Software Engineering, Miami, USA, pages 310-317. IEEE Press, 2004.

7. J. Gllavata and B. Freisleben. Adaptive Fuzzy Text Segmentation in Images with Complex Backgrounds using Color and Texture, Proceedings of the International Conference on Computer Analysis of Images and Patterns, Paris, France, pages 756-765. Springer Verlag, 2005.

8. J. Gllavata and B. Freisleben. Script Recognition in Images with Complex Backgrounds, Proceedings of the International Symposium on Signal Processing and Information Technology, Athens, Greece, pages 589-594. IEEE Press, 2005.

9. J. Gllavata and B. Freisleben. Combination of Crisp and Fuzzy Clustering Methods for Text Extraction in Complex Images, Proceedings of the International Conference on Application of Fuzzy Systems and Soft Computing, Siegen, Germany, pages 148-157. b-Quadrat Verlag, 2006.

10. E. Qeli, J. Gllavata and B. Freisleben. Customizable Detection of Changes for XML Documents Using XPath Expressions, Proceedings of ACM Symposium on Document Engineering (DocEng'06), Amsterdam, The Netherlands, pages 88-90. ACM Press, 2006.

11. J. Gllavata, E. Qeli and B. Freisleben. Detecting Text in Videos Using Fuzzy Clustering Ensembles, Proceedings of the IEEE International Symposium on Multimedia, San Diego, USA, pages 283-290. IEEE Press, 2006.

12. J. Gllavata, E. Qeli and B. Freisleben. Holistic Comparison of Text Images for Content-Based Retrieval, Proceedings of the IEEE International Symposium on Multimedia, San Diego, USA, pages 299-306. IEEE Press, 2006.

1.6 Organization of this Thesis

The remainder of this thesis is organized into nine chapters.

Chapter 2 introduces the fundamentals of digital image processing and related techniques. Furthermore, the principles of machine learning algorithms that will be used in this thesis are also described.

Chapter 3 discusses the state of the art of text detection/localization work, and provides details of the proposed text detection and localization approaches. The evaluation of the methods concludes the chapter.

Chapter 4 presents a new method to detect text in videos by exploiting temporal information in a video through the employment of a fuzzy cluster ensemble.

Chapter 5 describes a novel algorithm to track text over a video. Then, experiments conducted for a set of video sequences where moving text appears are presented.

Chapter 6 discusses different text segmentation methods introduced during the last years. Then, the proposed algorithms for text segmentation and binarization and their evaluation follows.

Chapter 7 presents a supervised method that enables the recognition of the script on a complex background and its evaluation.

Chapter 8 focuses on the problem of comparing two different text images using holistic techniques. In addition, experimental results with a set of binarized text images are discussed.

Chapter 9 describes the usefulness of applying browsing techniques to support content based image retrieval. A set of browsing techniques that enable the easy navigation through image collections is presented.

Chapter 10 concludes the thesis with a summary of the achievements and limitations of the proposed solutions and proposes some future research directions.

Chapter 2

Theoretical Background

Those who wish to succeed, must ask
the right preliminary questions.
- Aristotle -

2.1 Introduction

This chapter introduces several image processing algorithms which play an important role in image analysis in general and in text extraction from images in particular. In addition, different classification methods will also be introduced.

In Section 2.2, the terms image, digital image and image processing are defined. Sections 2.3, 2.4, 2.5, 2.6 and 2.7 briefly discuss the principal techniques used in image processing. Section 2.8 gives the basics of MPEG encoding. Finally, Section 2.9 introduces several classification methods which will be applied to solve different problems covered in this thesis.

2.2 Digital Image Processing

2.2.1 Images

Images are pictures: a way of recording and presenting information visually [Eff00]. Pictures are important to humans because of their effectiveness to capture and exchange visual information. In Figure 2.1, a picture of the

Figure 2.1: A photograph of the old city of Durrës

beautiful old city of Durrës (Albania) is shown. Through the use of pictures, the need for a lengthy verbal description is often avoided. This emphasizes the point that humans are primarily visual creatures, which means that vision is the most advanced of our senses. Thus, it is not surprising that images play the most important role in human perception and the brain is adept at visual data processing.

Photography is an imaging technique that records information similar to how our human visual system receives it. Both human vision and photography require a light source to illuminate a scene. The information about the objects in the scene is recorded as variations in the intensity and color of the detected light (see Figure 2.2). Due to the fact that the light tends to move in straight lines, the geometric properties of the objects in a scene are preserved. However, although a scene typically is three-dimensional, the image of that scene is always two-dimensional. Humans are limited to the visual band of the electromagnetic spectrum (in Figure 2.2 called "light"), while imaging machines cover almost the entire electromagnetic spectrum, ranging from gamma to radio waves. They can operate on images generated by sources which humans may not associate with images such as ultrasound, electron microscopy, and computer-generated images.

2.2.2 Digital Images

Usually, an image is defined as a two-dimensional function $f(x, y)$, where x and y are the plane coordinates. For monochrome images, the value $(f(x, y))$ of f at a given location (x, y) shows the intensity of the light detected at that point. In the case of color images, $f(x, y)$ is a vector of numerical data.

An image is called a *digital image*, when x, y and the values of f are all finite. Thus, a *digital image* consists of a finite number of elements or pixels, each of which has a particular location and value. The transformation

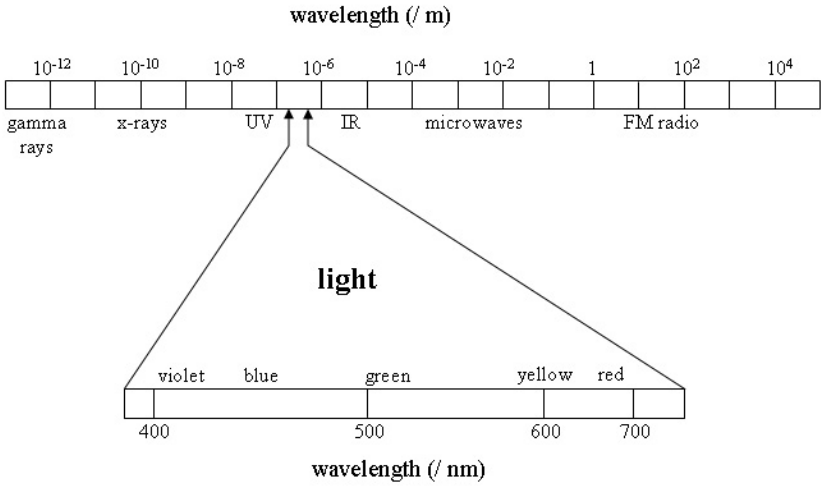


Figure 2.2: The spectrum of electromagnetic radiation (taken from [Eff00])

of the continuous function $f(x, y)$ into a finite function with discrete values is accomplished by the processes of sampling and quantization. These two processes usually are carried out by a piece of hardware known as analogue to digital converter.

2.2.3 Digital Image Processing

According to Efford [Eff00], *image processing* is a general term for the wide range of techniques that exist in manipulating and modifying *images* in various ways, whereas under the term *digital image processing* operations performed on *digital images* using computers are included. For Gonzales and Woods [GW01], the field of *digital image processing* refers to processing *digital images* by means of a digital computer.

There is no general agreement among authors regarding where image processing stops and other related areas, such as image analysis and computer vision, start. However, it is important to consider three types of computerized processes in this continuum: low-, mid-, and high-level processes [GW01]. Low-level processes involve primitive operations, which are characterized by the fact that both input and output are images. These processes include image processing operations such as to reduce noise, increase contrast, etc. Mid-level processing on images involve tasks such as segmentation, description of the segmented objects, and their classification. Similar to low-level processing, the inputs of mid-level processing are images, whereas their outputs are different attributes extracted from those images (e.g., edges,

contours, and the identity of individual objects). High-level processes consist in "understanding" of an ensemble of recognized objects, as in image analysis, and, at the far end of the continuum, performing the cognitive functions normally associated with vision.

Based on the preceding comments, Gonzales and Woods [GW01] conclude that the digital image processing field contains processes whose inputs and outputs are images and, in addition, processes that extract attributes from images, up to and including the recognition of individual objects.

2.3 Edge Detection in the Spatial Domain

Edge detection is the most useful approach for detecting discontinuities in the grey level or color of an image. Efford [Eff00] defines edges as locations in an image where there is a sudden variation in the grey level or color of pixels. Intuitively, an edge can be thought of as a set of connected pixels that lie on the boundary between two regions. The contours of potentially interesting scene elements (solid objects, surface marking, text strings, etc.) all generate intensity or color edges. However, in contrast to a region boundary which is a global property of the region, an edge is a local concept. Edge enhancement and detection operations are obvious steps to undergo when attempting to locate and recognize the scene elements. However, it must be pointed out that the problem of locating and recognizing scene objects is not trivial, because they usually are occluded in the background and noise, which can also generate strong edges.

The edge detection techniques that are discussed in this section are performed directly on the pixels of the image. Hence, the term *spatial domain* is used to distinguish this type of techniques from the others that take place in the *frequency domain* (see Section 2.4). These techniques are carried out through the convolution of a kernel (mask or template) with the image. The mechanism of convolution is illustrated in Figure 2.3. At each point (x, y), the response of the convolution at that point is given by the sum of products of the kernel coefficients and the image pixels in the area spanned by the convolution mask (edge detection mask). For the 3×3 mask shown in Figure 2.3, the response R of the convolution with the mask at a point (x, y) in the image is:

$$\begin{aligned} R = {} & w(-1,-1)f(x-1,y-1) + w(-1,0)f(x-1,y) + ...+ \\ & w(0,0)f(x,y) + ... + w(1,0)f(x+1,y) + w(1,1)f(x+1,y+1) \end{aligned}$$

The mask is centered at point (x, y) when the computation of the sum of products takes place. In general, the convolution of an image f of size $M \times N$

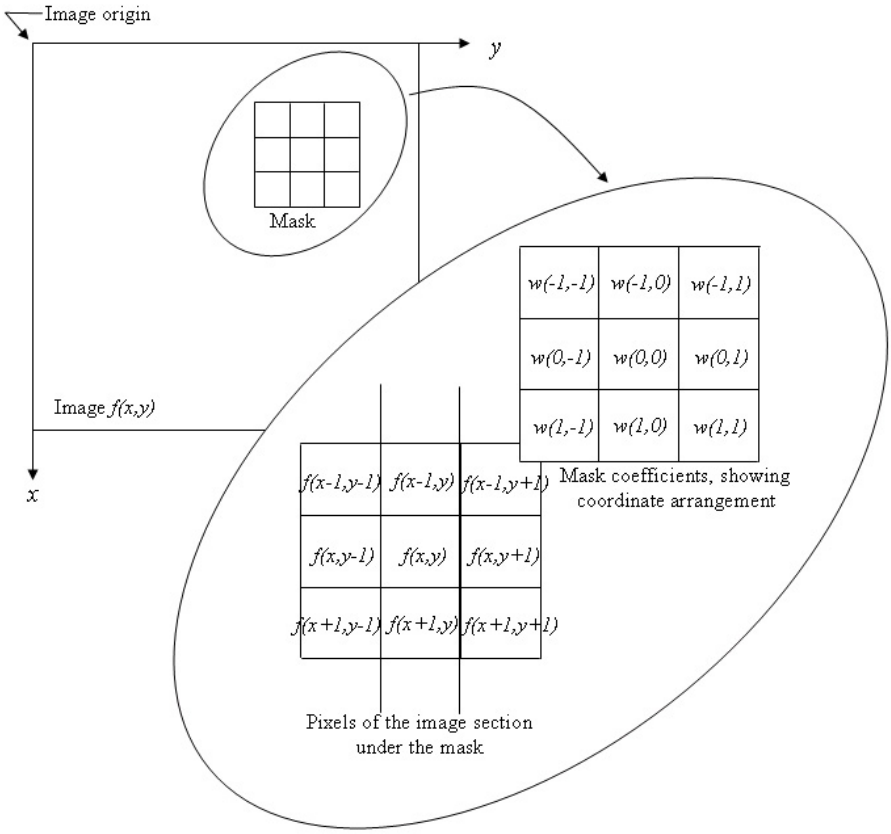


Figure 2.3: The mechanism of spatial convolution. The magnified drawing shows a 3×3 mask and the part of the image under it; the part of the image is shown displaced out from the mask for ease of readability (taken from [GW01])

with a mask w of size $m \times n$ is given by the equation:

$$g(x,y) = \sum_{i=-a}^{a} \sum_{j=-b}^{b} w(i,j) f(x+i, y+j), \; \forall \; (x,y) \in f \tag{2.1}$$

where $a = (m-1)/2$ and $b = (n-1)/2$.

In the following two sections, the first- and second-order edge detection methods will be shortly reviewed.

2.3.1 First-Order Edge Detection Operators

First-order derivatives in image processing are implemented using the magnitude of the gradient. The gradient of an image f at a location (x, y) is

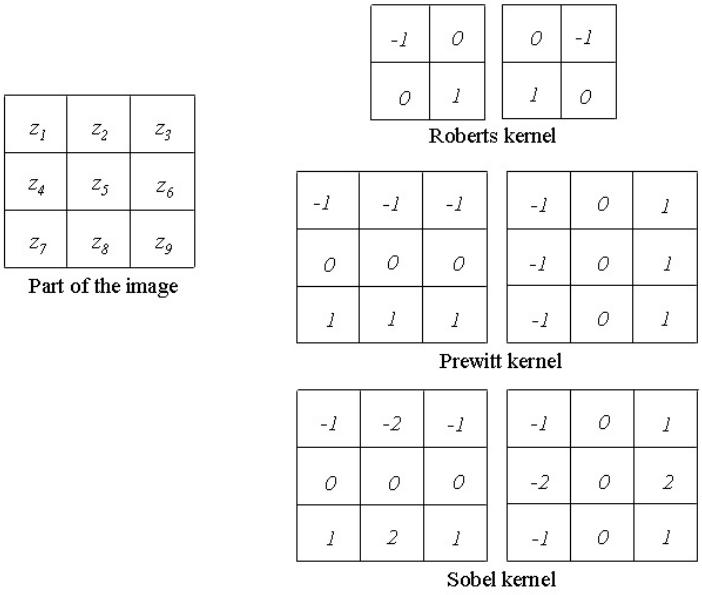


Figure 2.4: The various masks that might be used to compute the gradient of an image

defined as the two-dimensional vector:

$$\nabla \mathrm{f} = \begin{bmatrix} G_x \\ G_y \end{bmatrix} = \begin{bmatrix} \partial f/\partial x \\ \partial f/\partial y \end{bmatrix}$$

This vector is aligned along the direction of the edge. Gradient magnitude and direction are given by:

$$\nabla f = \mathrm{mag}(\nabla \mathrm{f}) = \sqrt{G_x^2 + G_y^2} \tag{2.2a}$$

$$\theta = \tan^{-1}(\frac{G_y}{G_x}) \tag{2.2b}$$

where the angle θ is measured with respect to the x-axis. Due to the fact that the square-root operation in Equation 2.2a is computationally expensive, the gradient magnitude is approximated with:

$$\nabla f = \mathrm{mag}(\nabla \mathrm{f}) = |G_x| + |G_y| \tag{2.3}$$

Computation of the gradient of an image is obtained through the computation of the partial derivatives $\partial f/\partial x$ and $\partial f/\partial y$ at every pixel location. One of the simplest way to compute a first-order partial derivative at a point z_1 of a given image is to use the Roberts cross-gradient operators [Eff00] (convolving the image with the Roberts mask shown in Figure 2.4):

$$G_x = (z_5 - z_1) \tag{2.4a}$$

$$G_y = (z_4 - z_2) \tag{2.4b}$$

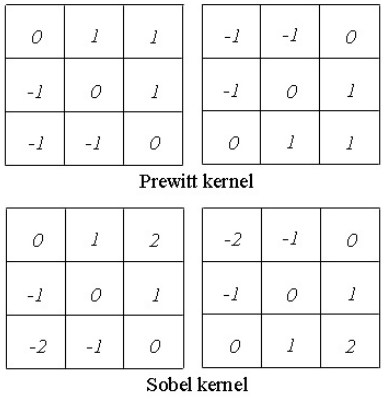


Figure 2.5: Prewitt and Sobel masks for detecting diagonal edges

where z_1, z_2, z_4 and z_5 are the pixels' values at the resprective locations. In order to slightly smooth the edges caused by the present noise, the gradients can be computed over a 3×3 neighborhood, which allows us to implement the gradient as a pair of convolution operations. The Prewitt and Sobel masks [Eff00] shown in Figure 2.4 can be used as convolution kernels in order to calculate the gradient in x and y direction.

It is possible to modify these masks, so that they have their strongest response along the diagonal directions. In Figure 2.5, two additional Prewitt and Sobel masks for detecting edges in the diagonal directions are shown.

2.3.2 Second-Order Edge Detection Operators

Gradient operation is an effective edge detector when the pixel gray levels (colors) change over space very rapidly. But when the gray levels change slowly from dark to bright, the gradient operation will produce a very wide edge. In this case, it is helpful to consider using the Laplace operation. The Laplacian of an image f is a second order derivative defined as:

$$\nabla^2 f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2} \tag{2.5}$$

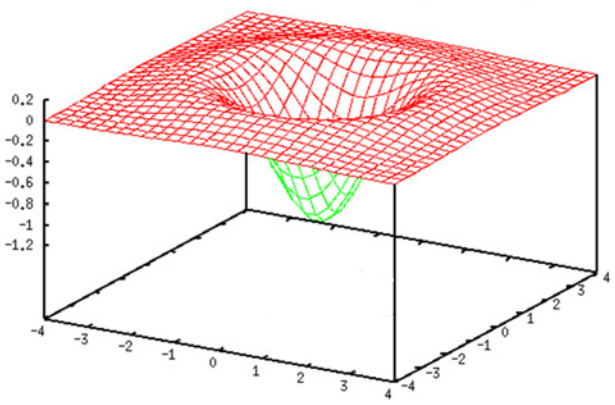
The second order derivative of the wide edge will have a zero crossing in the middle of the edge. Therefore, the location of the edge can be obtained by detecting the zero-crossings of the second order derivative of the image. Digital approximation of the Laplacian are shown in Figure 2.6. Since the Laplace operator may detect edges as well as noise (isolated, out-of-range), it may be desirable to smooth the image first by convolution with a Gaussian filter and then use a Laplacian to detect the edges. At the end, the localization of the edges is done by finding zero crossings. A radially-symetric, two-

0	-1	0
-1	4	-1
0	-1	0

-1	-1	-1
-1	8	-1
-1	-1	-1

Laplacian masks

Figure 2.6: Laplacian masks



0	0	-1	0	0
0	-1	-2	-1	0
-1	-2	16	-2	-1
0	-1	-2	-1	0
0	0	-1	0	0

Figure 2.7: Laplacian of a Gaussian (LOG) 3-D plot and its approximated mask

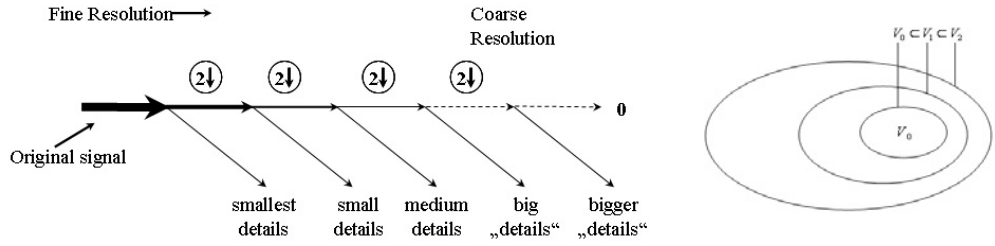
dimensional Gaussian is given by:

$$h(r) = \exp(\frac{-r}{2\sigma^2}) \text{ where } r = x^2 + y^2 \tag{2.6}$$

σ is the standard deviation of the distribution. The Laplacian of h, which is equivalent of the second derivative of h with respect to r, is:

$$\nabla^2 h = -(\frac{r^2 - \sigma^2}{\sigma^4}) \exp(\frac{-r}{2\sigma^2}) \tag{2.7}$$

This operator is often referred to as the Laplacian of Gaussian (LOG) filter. Figure 2.7 shows a 3-D plot and a 5×5 mask that approximates $\nabla^2 h$.



(a) (b)

Figure 2.8: (a) The general idea of multiresolution analysis; (b) The nested function spaces spanned by a scaling function

2.4 Multiresolution Analysis and Wavelet Transforms

2.4.1 Multiresolution Analysis

The word multiresolution refers to the simultaneous presence of different resolutions, whereas multiresolution analysis (MRA) refers to a technique that analyzes a signal (image) in multiple frequency bands (resolutions). The general idea of multiresolution analysis is illustrated in Figure 2.8(a), whereas the formal definition of MRA is provided in the following paragraph.

Let L denote the vector space of measurable, square-integrable functions. Multiresolution analysis of $L^2(R)$ is formally defined as a nested sequence of closed subspaces $\{V_j\}_{j\in Z}$ of $L^2(R)$ (see Figure 2.8(b)) that satisfies the following conditions:

- $... \subset V_{-1} \subset V_0 \subset V_1 \subset ... \subset L^2(R)$
- $\bigcap_j V_j = \{0\}$, and $\overline{\bigcup_j V_j} = L^2(R)$
- $f(t) \in V_j \Leftrightarrow f(2t) \in V_{j+1}$
- $f(t) \in V_0 \Rightarrow f(t-k) \in V_0$
- There exists a function $\phi(t)$ called scaling function, such that $\{\phi(t-k)\}$ is an orthonormal basis of V_0:

$$V_0 = \{f \in L^2(R) : f(t) = \sum_{k\in Z} \alpha_k \phi(t-k)\} \tag{2.8}$$

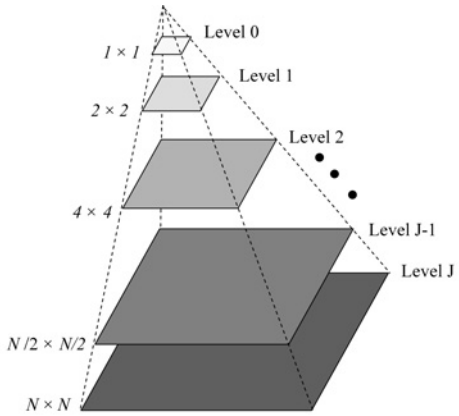


Figure 2.9: The image pyramid structure used for multiresolution analysis

Two existing approaches for multiresolution analysis are image pyramids and sub-band coding.

2.4.1.1 Image Pyramids

The simplest structure to represent images at different resolutions is the *image pyramid* introduced by Burt and Adelson [BA83]. An image pyramid is a collection of images in decreasing resolution, arranged in the shape of a pyramid. The structure of an image pyramid is illustrated in Figure 2.9. The base of the pyramid contains the high-resolution image, whereas the apex contains a low-resolution approximation. For an image of dimensions $N \times N$, the pyramid will have maximally $J = \log_2 N$ levels. In practice, the pyramid is truncated and only a necessary number of resolution levels is used. Different interpolation filters such as Laplacian, etc. are proposed for increasing/decreasing the resolution of a given image.

2.4.1.2 Sub-band Coding

Sub-band coding is a technique which relates also to multiresolution analysis. This technique consists of decomposing the signal/image into a set of independent sub-bands (channels) in order to treat the signal-image/sub-bands individually for different purposes. This decomposition of signal into different sub-bands is usually done using a collection of filters called a filter bank. A filter is usually a linear transformation that transforms an input signal to another signal. The sub-bands can be down sampled without loss of information, because the bandwidth of resulting sub-bands is smaller than that of the original signal. Reconstruction of the original signal is accomplished without errors by upsampling, filtering, and summing the individual sub-bands. Figure 2.10 shows the principal components of a two-band sub-band

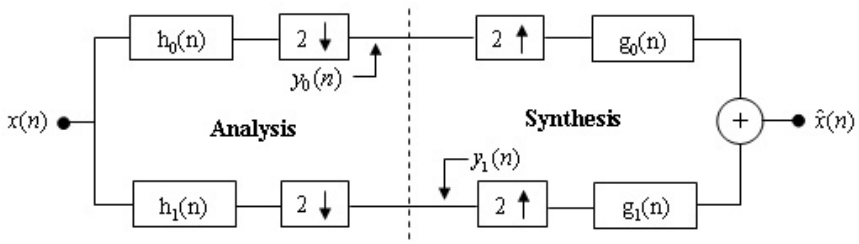


Figure 2.10: A two-band filter bank for 1D sub-band decomposition and reconstruction. $h_0(n)$ and $h_1(n)$ are the analysis filters, a low-pass and a high-pass filter respectively, $g_0(n)$ and $g_1(n)$ are the synthesis filters

decomposition and reconstruction system.

2.4.2 Wavelet Transforms

2.4.2.1 Continuous Wavelet Transform

The continuous wavelet transform (CWT) of a function $f(t) \in Z$ is a decomposition of $f(t)$ into a set of kernel functions $h_{a,b}(t)$ called wavelets as shown in Equation 2.9:

$$X_W(a,b) = \int_{-\infty}^{\infty} f(t) h^*_{a,b}(t) dt \tag{2.9}$$

where $*$ denotes the complex conjugate. However, most wavelets are real values and they are obtained from a single prototype wavelet (mother wavelet) $h(t)$ by means of dilation/contraction (scale) and translation:

$$h_{a,b}(t) = \frac{1}{\sqrt{a}} h(\frac{t-b}{a}) \tag{2.10}$$

where $a \in R^+$ is the scale factor, b is the translation factor and $\sqrt{a}$ is the energy normalizing factor. By substituting Equation (2.10) into (2.9), we will have the CWT of $f(t)$ defined as:

$$X_W(a,b) = \frac{1}{\sqrt{a}} \int_{-\infty}^{\infty} f(t) h^*(\frac{t-b}{a}) dt \tag{2.11}$$

The CWT analyzes the function by convolving it with wavelets. Different frequencies are detected by scaling the wavelets using the scaling variable a.

2.4.2.2 Discrete Wavelet Transform

CWT is not well-suited for discrete computation. In theory, there are infinitely many scales and infinitely many translations to compute in order to

achieve perfect reconstruction of signal. In practice, both have to be quantized, which leads to redundancy and errors in reconstruction, thus making it difficult to use CWT in practical applications. The discrete wavelet transform (DWT) overcomes the quantization problems and allows fast (linear time complexity) computation of the transform for digitized signals. DWT also allows perfect signal reconstruction.

The DWT in contrast to CWT is evaluated at discrete scales and translations. The discrete scale is denoted as $a = a_0^i$, where $i \in Z$ and $a_0 > 1$ is a fixed dilation step, whereas the discrete translation factor is denoted as $b = kb_0a_0^i$, where $k \in Z$. The notation of the wavelets is changed from now on from $h(t)$ to $\psi(t)$ in order to distinguish the discrete wavelet transform from the continuous wavelet transform. Based on the above definitions and Equation 2.10, the discrete wavelets are expressed as:

$$\psi_{i,k}(t) = \frac{1}{\sqrt{a_0^i}} h(\frac{t - kb_0a_0^i}{a_0^i}) = \frac{1}{\sqrt{a_0^i}} \psi(\frac{t}{a_0^i} - kb_0) \tag{2.12}$$

The DWT with the scaling factor of $a_0 = 2$ is effective for computer implementation. DWT is also a form of multiresolution analysis and is closely related to filter banks, pyramid algorithms in image processing, quadrature mirror filters in speech processing and fractals.

2.4.2.3 Filters and Wavelet Transform

Filters. Wavelets are basis functions in the function space. All the functions of the basis are derived from a single function $\{\psi_{jk}(t) = 2^{j/2}\psi(2^jt - k)\}$, called the basic wavelet or mother wavelet $\psi(t)$, by means of translations and dilations. In the multiresolution analysis framework, the wavelet bases are derived from the scaling function $\phi(t)$, using the following equation:

$$\psi(t) = \sqrt{2} \sum_k g_k \phi(2t - k) \tag{2.13}$$

where the scaling function ϕ may be further decomposed as a linear combination of the scaling functions at a higher resolution level:

$$\phi(t) = \sum_k h_k \phi(2t - k) \tag{2.14}$$

The coefficients g_k and h_k are called interscale coefficients and are used in the wavelet decomposition as discrete high-pass and low-pass filters, respectively.

Wavelet Transform. Representing a function $f(t)$ in terms of a wavelet basis is accomplished by doing a wavelet transform. The approximation $f_J(t)$

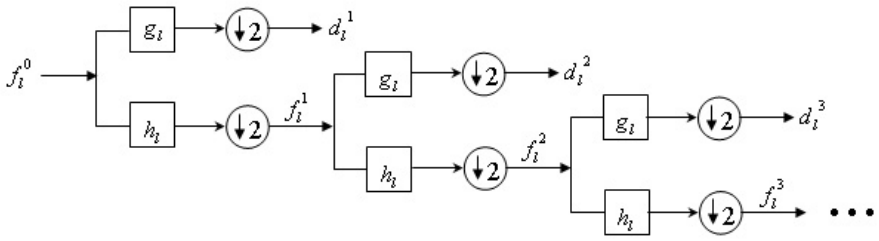


Figure 2.11: Illustration of the one-dimensional wavelet transform

of a function $f(t)$ can be written in terms of the wavelet basis $\{\psi_{jk}(t) = 2^{j/2}\psi(2^j t - k)\}$ and scale basis $\{\phi_{j_0 k}(t) = 2^{j_0/2}\phi(2^{j_0} t - k)\}$ as:

$$f_J(t) = \sum_k f_k^J \phi_{Jk}(t) = \sum_{l=j_0}^{J-1} \sum_k d_k^j \psi_{lk}(t) + \sum_k f_{k_0}^j \phi_{j_0 k}(t) \tag{2.15}$$

where the coefficients f_k^j and d_k^j are given by the inner products:

$$f_k^j = \int_{-\infty}^{\infty} f(t)\phi_{jk}(t)dt \text{ and } d_k^j = \int_{-\infty}^{\infty} f(t)\psi_{jk}(t)dt \tag{2.16}$$

The coefficients $\{f_k^j\}$ represent the smooth part or the trend of the function $f(t)$ at the resolution level j and $\{d_k^j\}$ represent the details of the function. The wavelet transform iterates the decomposition of the smooth part into a smooth part and details while leaving the details intact. Thus, $\{f_0^J, f_1^J, ..., f_{N-1}^J\}$ is transformed into:

$$\{\{f_{00}^j, f_{10}^j, ...\}, \{d_{00}^j, d_{10}^j, ...\}, \{d_0^{j_0+1}, d_1^{j_0+1}, ...\}, ..., \{d_0^{J-1}, d_1^{J-1}, ..., d_{N/2-1}^{J-1}\}\} \tag{2.17}$$

using the following formulas:

$$f_k^{j-1} = \sum_l h_{l-2k} f_l^j \text{ and } d_k^{j-1} = \sum_l g_{l-2k} f_l^j \tag{2.18}$$

The inverse wavelet transform does the inverse using:

$$f_k^j = \sum_l h_{k-2l} f_l^{j-1} + \sum_l g_{k-2l} d_l^{j-1} \tag{2.19}$$

Figure 2.11 gives an illustration of the one-dimensional wavelet transform.

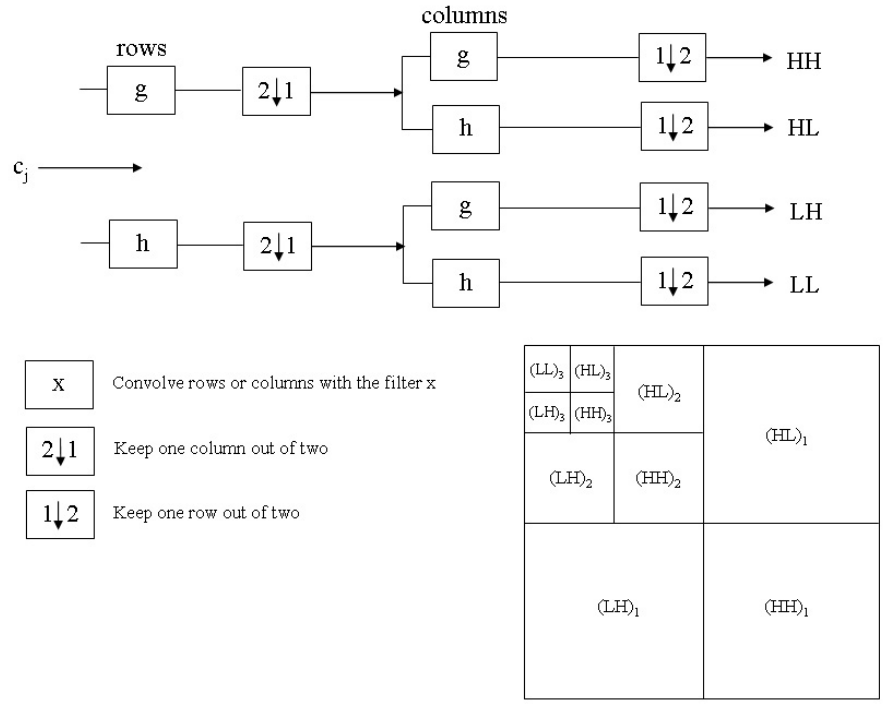


Figure 2.12: Illustration of 2D wavelet decomposition with quadrature mirror low-pass and high-pass filters h_l and g_l; the presentation of 2D wavelet decomposition

2.4.2.4 Two-Dimensional Wavelet Transform

The wavelet transform can be easily extended to the two-dimensional case for image processing applications. The wavelet transform of a two-dimensional image $f(x, y)$ is:

$$X_W(a_x, a_y; u, v) = \frac{1}{\sqrt{a_x a_y}} \int \int f(x, y)\psi(\frac{x-u}{a_x}; \frac{y-v}{a_y})dxdy \qquad (2.20)$$

This function can be reduced to two variables u and v, when the scale factors $a_x = a_y = a$.

The schematic of 2D wavelet decomposition using low-pass and high-pass filters h_l and g_l is illustrated in Figure 2.12. At each resolution, the pair of 1D filters h_l and g_l first are applied to each row of the image, resulting in a horizontal approximation image and a horizontal detail image. Then, the pair of 1D filters is applied to each column of the two horizontally filtered images. The sampling by 2 is applied after each filtering. On the right of Figure 2.12, the two-dimensional wavelet decomposition is presented.

2.5 Image Segmentation

Image segmentation deals with the subdivision of the image into distinct areas, which constitute the present objects or features of interest in the image. Segmentation can be also regarded as a process of grouping together pixels that have similar attributes [Eff00]. Segmentation of complex, nontrivial images is one of the most difficult tasks in image processing. The success or failure of subsequent image analysis procedures depends directly on the accuracy of the segmentation process. Segmentation techniques have found numerous applications in different areas, such as:

- Industrial inspection, which consists of analyzing images of products in order to determine the presence or absence of specific anomalies;
- Optical character recognition;
- Tracking of objects in a sequence of images;
- Classification of terrains visible in satellite images;
- Detection and measurement of bone, tissue, etc., in medical images, which is very useful for diagnosing different diseases.

According to Gonzales and Woods [GW01] image segmentation techniques generally are based on one of two basic properties of intensities values: discontinuity or similarity. The first category consists of approaches which partition an image based on abrupt changes in intensity, such as edges in an image, whereas the second category consists of approaches that group together pixels that are similar according to a set of predefined criteria. Methods such as thresholding, region growing, region splitting and merging belong to this category.

Efford [Eff00] classifies segmentation methods into two groups: contextual or non-contextual. Non-contextual methods ignore the relationships that exist between features in an image. A non-contextual approach simply subdivides an image into regions on the basis of some global attribute, such as grey level. Threshold-based segmentation is a typical non-contextual method. Contextual methods, on the other side, make use of the relationships between image features. Thus, a contextual approach might group together pixels that have not only similar grey levels, but are also close to each other. Region growing methods are the simplest contextual techniques.

2.6 Morphological Image Processing

Morphology is defined as the branch of biology that deals with the form and structure of organisms (animals and plants) with no consideration of their

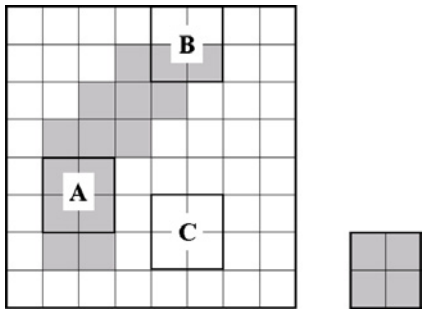


Figure 2.13: Exploring of the image on the left with a structuring element on the right (taken from [Eff00])

function [the]. In analogy with this, the term *morphological image processing* is used to describe a range of non-linear image processing techniques, used to extract image components that are useful to represent and describe region shapes, such as boundaries, skeletons and convex hull. In addition, there also exist pre- or post processing morphological techniques to reduce noise or other artifacts from binary images, such as morphological filtering, thinning, and pruning.

Morphological operations can be applied to binary or grayscale images and are based on set theory. Morphological techniques typically explore an image with a small template known as *structuring element*. The structuring element is moved over all possible positions (pixels) in the image and its structure is compared with the corresponding neighborhood of the pixels. The used comparison techniques are: (I) *fits*, where it is checked if the structuring element fits within the neighborhood; or (II) *hits*, where it is tested if it intersects with the neighborhood. These two comparison concepts are illustrated in Figure 2.13. It is clear that at A, the structuring element fits the image; at B, it hits the image; at C, it neither fits nor hits the image. *Dilation* and *erosion* are the fundamental operations to morphological processing. They are the basis for a broad class of more complex morphological algorithms. The *dilation* of an image f by a structuring element s is denoted as $f \oplus s$ and for each pixel (x, y) from image f is defined as:

$$g(x,y) = \begin{cases} 1 & \text{if } s \text{ hits } f, \\ 0 & \text{otherwise} \end{cases} \tag{2.21}$$

Erosion of an image f by a structuring element s is denoted as $f \ominus s$ and for each pixel (x, y) from image f is defined as:

$$g(x,y) = \begin{cases} 1 & \text{if } s \text{ fits } f, \\ 0 & \text{otherwise} \end{cases} \tag{2.22}$$

2.7 Feature Extraction

Feature or attribute extraction is a crucial step in most pattern analysis tasks and is the basis of content-based image retrieval. This process is often carried out intuitively and heuristically and it is based on the following main principles:

- *Discrimination:* Features of patterns in different classes should have significantly different values.
- *Reliability:* Features should have similar values for patterns of the same class.
- *Independence:* Features should not be strongly correlated to each other.
- *Optimality:* Redundant features should be deleted. In order to reduce the number of features, so called dimension reduction techniques such as principal component analysis, multi-dimensional scaling, etc. are used.

Features that can be extracted from objects are commonly called low-level features, because they are extracted directly from the digital representations of objects and have little or nothing to do with human perception. Features such as color, shape, texture, color layout, etc. are included.

Features that involve semantic information, such as present objects (human faces, cars, text, etc.), are called high-level (semantic) features.

Because of perception subjectivity, there is no single best presentation for a given feature. For any given feature, there exist multiple representations, which characterize the feature from different perspectives.

2.8 MPEG Video Compression

The MPEG (Moving Pictures Experts Group) standard specifies a technique for the compression of audio and video data and it is the leading compression technique for multimedia applications [MPE].

Video compression relies on two basic assumptions. First, the human sensitivity to noise depends on the frequency of the noise. Second, since a video is a sequence of images (i.e. frames) in time, there is a high degree of similarity between one frame and the next. The basic idea behind the MPEG video compression is to remove spatial redundancy within a video frame and eliminate temporal redundancy between video frames. In MPEG (similar to JPEG format), the Discrete Cosine Transform (DCT) is employed to analyze the two-dimensional spatial frequencies and reduce spatial redundancy. A block of pixels, typically 8×8, is converted into an array of coefficients

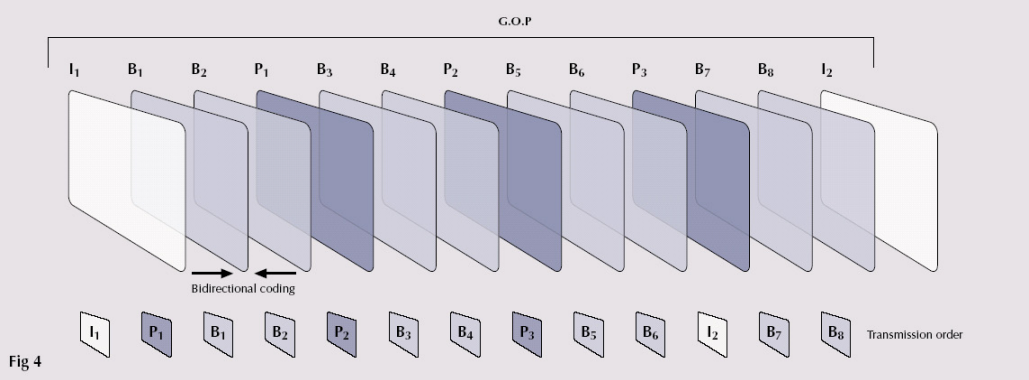


Figure 2.14: Illustration of the MPEG bi-directional compression (taken from [MPE])

using DCT. The magnitude of each coefficient represents the amount of a particular frequency which is present. The coefficient on the top left corner represents the so called DC component and shows the average brightness level of the respective (pixels') block. Moving to the right (down) the horizontal (vertical) spatial frequency coefficients are represented in increasing order. The coefficient in the bottom right corner represents the highest diagonal frequency. In reality, many of these coefficients have negligible or zero values, yielding sufficient compression possibilities. Motion compensation is used to exploit temporal redundancy. The successive frames in a video stream usually do not change much within small time intervals. Thus, the idea of motion-compensation is to encode a certain video frame based on other video frames which are close to it in time. At the coder, successive frames are compared and the motion of a macroblock from one frame to the next is evaluated and the so called motion vectors are obtained. The new position of an object in the next frame is predicted by shifting pixels from the previous frame using the estimated motion vectors. Possible prediction errors are eliminated by comparing the predicted frame picture with the actual picture. The coder encodes the motion vectors and the errors, which are used from the decoder to produce the next frame picture.

There are three types of frames in an MPEG stream: 1) Intra-coded frames (I-frame), 2) Predicted frames (P-frame) and 3) Bi-directional frames (B-frame). Figure 2.14 shows an example of a sequence of frames used in MPEG. The sequence begins with an I-frame as an anchor. All subsequent frames before the next I-frame, including the first I-frame are called a Group of Pictures (GOP). A GOP contains an I-frame and a number of forward predicted frames (P-frames) and B-frames. An I-frame is coded without any references to any other frames. The first P-frame is decoded based on the first I-frame, using motion estimation and adding difference data. The subsequent

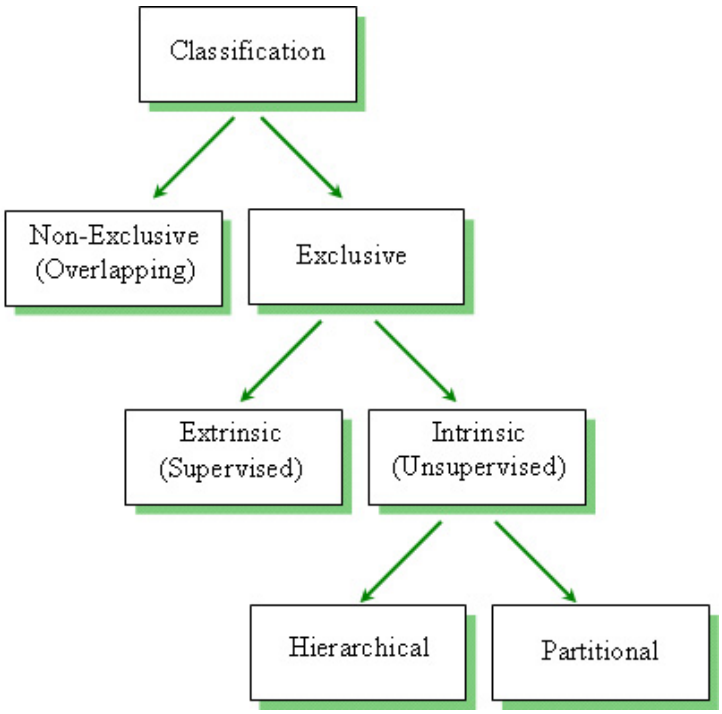


Figure 2.15: Tree of classification types

P-frames are decoded using the previous P-frame. B-frames may be decoded using motion vectors and difference data from I- or P-frames immediately before and afterwards. It is obvious that the backward encoded frames cannot be decoded if the decoder does not have the referenced frame. This problem is solved by decoding the frames out of display order. First, the I-frame and the next P-frame are decoded. Then, the B-frame in between can be decoded by moving motion compensated data forwards and backwards.

2.9 Classification Methods

In Figure 2.15, the tree of classification problems as suggested by Lance and Williams [LW67] is shown. Each leaf in the tree defines a different genre of classification problem ([JD88]).

Exclusive versus Non-Exclusive. An exclusive or hard classification consists of partitioning the objects into a number of subsets, where each object belongs to exactly one subset. Non-exclusive or fuzzy classification tends to classify an object in several groups. For example, a grouping of people by sex is exclusive, whereas a grouping by age (e.g. young or old) is non-exclusive, because the boundary between an old group and a young group is not clearly

defined. Fuzzy clustering, which is explained in detail in Section 2.9.2, is a type of non-exclusive classification in which an object is assigned to a subset to a certain degree, which is called membership degree.

Intrinsic versus Extrinsic. An intrinsic classification intends to partition the objects using only the similarity/dissimilarity matrix. Intrinsic classification is known in the literature as "unsupervised learning", due to the fact that no know-how about the partition of the object is known beforehand. Extrinsic or supervised classification, in contrast to intrinsic, uses category labels on the object as well as the similarity/dissimilarity matrix. The aim of an extrinsic classification is to define a discriminative surface that separates the objects according to the given categories. Some extrinsic classification methods (e.g. k-nearest neighbor, etc) will be discussed in Section 2.9.1.

Hierarchical versus Partitional. Classifications methods that are both exclusive and intrinsic too, are further divided into two main groups: a) Hierarchical and b) Partitional methods. A hierarchical method is based on a recursive partitioning of the objects, which may begin with a single cluster containing all objects, to a number of clusters equal to the total number of objects. Hierarchical methods make use of agglomerative (bottom-up) methods, which proceed with a series of fusions of the objects into groups, and divisive (top-down) methods, which separate the objects successively into finer groupings. A partitional classification consists of generating a single partition of the data, in an attempt to recover natural groups present in the data. The k-means algorithm is the most popular partitional clustering method and will be discussed in Section 2.9.2. Hierarchical clustering is very popular in biological, social and behavioral sciences, because of the need to construct taxonomies, and the fact that visual representation of the dendogram enables a data analyst to see how objects are being merged into clusters or split at successive levels of proximity. Partitional clustering is mostly used in engineering applications where single partitions are required and when the number of objects that have to be clustered is large.

2.9.1 Supervised Classification

The problem of supervised classification can formally be stated as follows. Let o be an object to be classified. Given a training set of objects $T = \{t_1, t_2, ..., t_M\}$ and the labels of classes where each of them belongs to, the goal of a supervised classifier is to find the label of the class this object belongs to, using the training data T.

Two different solutions of this problem are given in the following paragraphs.

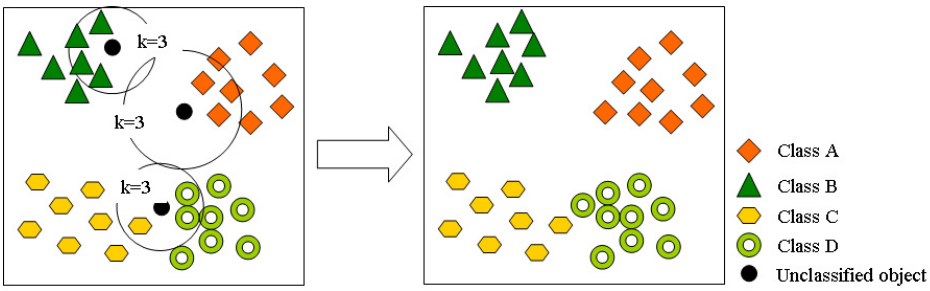


Figure 2.16: k-nearest neighbor classification

2.9.1.1 k-Nearest Neighbor Classifiers

The k-nearest-neighbor (k-NN) method dates back at least to 1951 [FH51]. These classifiers are memory-based, and do not require a model to be fit. Given a sample object o, this method finds the k observations in the training set T closest in distance to o, and then classifies using a majority vote among the k neighbors. For the object o, the decision function Y using the k-NN classifier is defined as follows:

$$Y(o) = \frac{1}{k} \sum_{t_j \in N_k(o)} y_j \tag{2.23}$$

where y_j is the label of the group where the object t_j belongs to and $N_k(o)$ is the neighborhood of o defined by the k closest points t_j in the training sample T. The closest points are defined employing an appropriate distance metric, which for the moment can be assumed to be the Euclidean distace (additional distances can be found in Chapter 7). After the k observations t_j closest to input object o have been found, their responses are then averaged to take the final decision for the label of the class where the input object o belongs to.

Despite its simplicity, k-NN has been successful in a large number of classification problems, including handwritten digits recognition, ECG analysis, etc. They have shown to be effective in the cases where each class has many possible prototypes, and the decision boundary is very irregular. However, one drawback of the k-NN method is its computational load, both in finding the neighbors and storing the entire training set.

2.9.1.2 Weighted Euclidean Distance Classifiers

The Weighted Euclidean Distance (WED) based classifier works as follows. Given the training set of data T, the mean μ^i and the standard deviation σ^i of the respective features are calculated for each class λ_i. Then, for each

test sample x, the distance between the sample data x and the class λ_i is evaluated using the formula:

$$distance(x, \lambda_i) = \sum_{k=1}^{d} |\frac{x_k - \mu_k^i}{\sigma_k^i}|, i = 1..M \tag{2.24}$$

where d shows the feature dimension and M is the number of the known classes. Finally, the test sample x gets assigned the label λ_i of the class with the minimum distance ($distance(x, \lambda_i)$ is the smallest evaluated distance).

2.9.2 Clustering Methods

Jain and Dubes [JD88] have defined cluster analysis as a process of classifying objects into subsets that have meaning in the context of a particular problem. Clustering can be regarded as a special form of classification in that it creates a labeling of objects with groups (clusters) labels. However, it derives these labels only from the data. Usually, the objects are characterized as points in a d-dimensional metric space and the relationships between them are represented in a so called similarity/dissimilarity (proximity) matrix, in which rows and columns correspond to objects. The proximity between two objects is equivalent to the distance between them. In this subsection, we explain two clustering algorithms:

- An exclusive unsupervised partitional clustering method, such as k-means clustering.
- A non-exclusive clustering method, such as fuzzy C-means clustering.

2.9.2.1 k-Means Algorithm

The problem of partitional clustering in general can be formally stated as follows. Let $X = \{x_1, x_2, .., x_N\}$ be a data set of N objects (patterns) in a d-dimensional Euclidean space R^d with a predefined norm $\| * \|$ and let K be a number larger than one.

The goal is to determine a partition of the N objects into K groups, or clusters, such that the objects in a cluster are more similar to each other than to objects in different clusters. The greater the similarity within a group and the greater the difference between groups, the better or more distinct the partitioning (clustering).

k-means [Mac67] represents a straightforward and the most popular iterative solution given to the problem of partitional clustering. It functions as follows. First, the initial centroids of the clusters are defined (c_k, $\forall k = 1..K$). The initialization of centroids is crucial because different initializations may

Algorithm 1: The pseudocode of the k-means algorithm

Input: N objects to cluster x_i i=1..n; number of clusters K

Output: A table of length N which shows the clusters membership for each object: $clusters[]$

```
Select K initial cluster centroids c_1, c_2, ..., c_K;
repeat
    for i=1 to N do
        for k=1 to K do
            d_ik = ||x_i − c_k||;
        end
        Assign object x_i to the cluster k with the minimum d_ik;
        clusters[i] = k
    end
    for k=i to K do
        Set S_k = {x | x ∈ cluster_k};
        c_k = (Σ_{∀x∈S_k} x) / |S_k|, where |S_k| is the cardinal of the set ;
    end
until convergence criteria is met;
```

lead to different results. So, the better choice is to place them as much as possible far away from each other. Secondly, each object x_i is associated with the cluster whose centroid has the minimal distance from the object. After this, the centroids c_k are recalculated as barycenters of the clusters $cluster_k$ resulting from the previous step. This process is repeated until the position of the centroids do not change any more or the number of iterations is completed.

The k-means algorithm aims at minimizing an objective function, in this case a squared error function, which is given in Equation 2.25.

$$J = \sum_{k=1}^{K} \sum_{i|x_i \in cluster_k} ||x_i - c_k||^2 \tag{2.25}$$

where $||x_i - c_k||^2$ implies the use of a metric to measure the distance between an object x_i and the center of the cluster c_k. Usually, the squared Euclidean distance is chosen as the dissimilarity measure.

The pseudocode of the k-means is given in Algorithm 1.

Algorithm 2: The pseudocode of the fuzzy C-Means algorithm

Input: N objects to cluster: x_i where $i = 1..N$; number of clusters C

Output: The fuzzy partitioning of the data $U_{N \times C}$

Initialize $U_{N \times C}$ randomly such as $\forall i = 1..N, \sum_{j=1}^{C} U(i,j) = 1$;
repeat
  Calculate the vectors of cluster centers $c_j = \frac{\sum_{i=1}^{N} u_{ij}^m \times x_i}{\sum_{i=1}^{N} u_{ij}^m}$ for j=1..C where m is the fuzzification factor, usually $m = 2$;
  Calculate $d_{ij} = \|x_i - c_j\|$;
  for $i = 1$ *to* N **do**
    for $j = 1$ *to* C **do**
      if $d_{ij} > 0$ **then**
        $u_{i,j} = \frac{1}{\sum_{k=1}^{C} (\frac{d_{i,j}}{d_{ik}})^{\frac{2}{m-1}}}$;
      else
        $u_{i,j} = 1$;
      end
    end
  end
until *(*$max\{\delta U\} < \epsilon$*) OR (*$Number_{Iterations}$ *reached)*;

2.9.2.2 Fuzzy C-Means Algorithm

In practice, quite often there are situations where an object does not belong fully to a cluster, but is more or less divided into several clusters, e.g. an object which is equidistant from two cluster centers, belongs thus to both clusters to some degree. Fuzzy set theory created by Zadeh [Zad65] permits an object to belong to a cluster with a grade of membership, which takes a value in the interval [0,1]. Membership grades are subjective in nature and are based on definitions rather than on measurements. Based on fuzzy set theory, a new clustering algorithm called fuzzy C-means (FCM) was developed [Dun74, Bez74, Bez76].

Fuzzy C-means is also a data clustering technique, but in contrast to the k-means algorithm where at the end an object is classified to one single cluster, here each data point belongs to a cluster to some degree (membership grade). The problem in this case is formulated slightly different than in the previous section. Let $X = \{x_1, x_2, .., x_N\}$ be a data set of N objects in a d-dimensional

Euclidean space R^d with a predefined norm $\| * \|$ and let C be a positive integer larger than one. C shows the number of clusters in which the data set X should be partitioned. The output of a FCM algorithm is the final membership matrix $U_{N\times C}$, whose elements show the membership grade of an object to a specific cluster.

FCM clustering is based on the minimization of the objective function J_m:

$$J_m = \sum_{i=1}^{N}\sum_{j=1}^{C} u_{ij}^m \|x_i - c_j\|^2 \tag{2.26}$$

where m is the fuzzification constant, which can have influence on the clustering performance of FCM ($m = 2$), u_{ij} is the degree of membership of x_i in j^{th} cluster ($cluster_j$), x_i is the i^{th} element in the data set X, c_j is the center of the $cluster_j$. Fuzzy partitioning is carried out through an iterative optimization of the objective function J_m, where the membership u_{ij} and the cluster centers c_j are updated using Equations 2.27 and 2.28:

$$u_{ij} = \frac{1}{\sum_{k=1}^{C} (\frac{\|x_i - c_j\|}{\|x_i - c_k\|})^{\frac{2}{m-1}}} \tag{2.27}$$

where

$$c_j = \frac{\sum_{i=1}^{N} u_{ij}^m * x_i}{\sum_{i=1}^{N} u_{ij}^m} \tag{2.28}$$

This iteration is stopped when $max_{ij}\{|u_{ij}^{k+1} - u_{ij}^k|\} < \epsilon$ or the number of iterations $Number_{Iterations}$ is reached. ϵ is a termination threshold between 0 and 1, whereas k shows the iteration steps.

Algorithm 2 shows the steps that define the FCM algorithm.

2.10 Summary

In this chapter, we have first discussed the foundations of digital image processing and then we have continued with the description of the different edge detection operations in the spatial domain as well as in the frequency domain. Both first-order edge detection operators such as Sobel kernels and second-order operators which included Laplacian of a Gaussian operation were introduced. Furthermore, the principles of multiresolution analysis and wavelet transformation were also treated. We have then described the idea of morphological images processing and its application. The MPEG standard for the compression of the audio and video data was also described. A short overview of different classification methods (supervised and unsupervised), which will be used in the next chapters of this thesis, have concluded this chapter.

Chapter 3

Text Detection and Localization in Images

There is no true interpretation of anything;
interpretation is a vehicle in the service of human comprehension...
- Andreas Buja -

3.1 Introduction

As mentioned in Chapter 1, text appearing in images can be classified into two groups: scene text and artificial text. Artificial text in contrast to scene text is often a good key for successful indexing and retrieval of images (or videos). At first, extraction of text may seem to be a trivial application for existing optical character recognition (OCR) tools. However, compared to OCR for document images, extracting text from real images faces numerous challenges due to lower resolution, unknown text color, size and position, or complex backgrounds. Text extraction and recognition, which include text detection, localization, segmentation and binarization, and recognition, is an useful process regarding text-based image indexing. Furthermore, it is a very important task towards enabling searching information in web sites or digital multimedia libraries (e.g. databases of images, videos or document images). The first three processing stages are important to achieve high-quality text recognition results when applying an OCR system.

This chapter is focused on the first two problems namely text detection and localization.

Three methods will be presented to detect and localize texts appearing in images/videos and make them ready for a subsequent segmentation process. Furthermore, the case of text having an arbitrary alignment is also addressed. Experimental results for different sets of images and video frames will be presented to demonstrate the high-quality detection/localization performance of the approaches. Part of the material presented in this chapter has been published in [GEF03a, GEF03b, GEF04b, GEF04a].

The remainder of this chapter is organized as follows. Section 3.2 gives an overview of related work in the field. Section 3.3 presents the proposed approaches to text detection/localization in detail and describes the experimental results obtained for several sets of images. Section 3.4 concludes the chapter.

3.2 Previous Work

In the literature, text detection and localization are regarded as an unique problem. Consequently, in this section the main methods proposed to deal with text detection/localization in images and videos will be reviewed. Jung et al. [JKJ04] have recently published a good overview of these methods.

Text detection/localization can be classified by whether they are originally designed to work on still images or video streams. In this chapter, the first category of methods which are also related to the approaches proposed in Section 3.3, is addressed. The other category of methods is reviewed in Chapter 4.

Jung et al. [JKJ04] have classified text detection/localization methods according to the employed features into two main groups: *region-based* and *texture-based*. The first class of methods [SDB98, JY98, AD99, CSB01] employs connected component analysis, which is based on the analysis of the geometrical arrangement of edges or homogeneous color and grayscale components that belong to characters. They are simple to implement, but they are not very robust for text localization in images with complex background and usually their performance depends on several threshold values. The second class of methods [HYZM03, LDK00, WMR99, KJPK01, LW02] considers texts as regions with distinct textural properties. They are more accurate when the text is embedded in a difficult background, whereas they suffer from detecting different sizes of text and are computationally expensive.

3.2.1 Region-Based Methods

Region-based methods can be further divided into two classes: *connected component (CC)-based* ([SDB98, JY98]) and *edge-based* ([SKHS98, SKH+99,

AD99, CSB01]). These methods are known also as bottom-up approaches, due to the way how they operate; by first identifying elementary (small) sub-structures such as CCs or edges, and then merging these sub-structures successively into larger structures, until all text areas are detected. Geometrical analysis based on different thresholds or several heuristics are finally applied in order to filter out possible false alarms. In CC-based methods, the basic elements are created using the similarity of neighbor pixels in grayscale or color levels, whereas the edge-based methods focus on the high contrast between the text and the background, identifying first the edges caused from the text contours and then grouping them, if possible.

Jain and Yu [JY98] first perform a color reduction by bit dropping and color clustering quantization, and afterwards a multi-value image decomposition algorithm is applied to decompose the input image into multiple foreground and background images. Then, connected component analysis is performed on each of them to localize text candidates. This method extracts only horizontal texts of large sizes.

Shim et al. [SDB98] use the homogeneity of intensity of text regions in images. Pixels with similar gray levels are merged into a group. After removing significantly large regions by regarding them as background, text regions are sharpened by performing a region boundary analysis based on the gray level contrast. The candidate regions are then subjected to verification using size, area, fill factor, and contrast. Neighboring text regions are examined to extract any text strings. The average processing time was 1.7 seconds per frame on a 133 MHz Pentium processor and the miss rate ranged from 0.29% to 2.68% depending on the video stream.

Smith and Kanade [SKHS98, SKH+99] apply a 3×3 horizontal differential filter to an input image and perform thresholding to find vertical edges. After a smoothing operation that is used to eliminate small edges, adjacent edges are connected and a bounding box is computed. Then, heuristics, including the aspect ratio, fill factor, and size of each bounding box are applied to filter out non-text regions. Finally, the intensity histogram of each cluster is examined to filter out clusters that have similar texture and shape characteristics.

Agnihotri and Dimitrova [AD99] have presented an algorithm which uses the red frame of the RGB color, with the aim to obtain high contrast edges for the frequent text colors. By means of the convolution process with different masks, they first enhance the image and then detect edges. This edge image is further processed by grouping neighboring edge pixels to single connected components structures. Finally, the candidates undergo a segmentation process in order to be ready for OCR.

Cai et al. [CSL02] have presented a text detection approach which uses character features like edge strength, edge density and horizontal alignment. First, they apply a color edge detection algorithm in YUV color space and filter out non-text edges using a low threshold. Then, a local thresholding

technique is employed in order to keep low-contrast text and further simplify the background. An image enhancement process using two different convolution kernels follows. Finally, projection profiles are analyzed to refine the localization text regions. Whereas Lyu et al. [LSC05] have proposed to detect the text on the grayscale image on the basis of a multi resolution schema. Furthermore, a segmentation method which consists of three steps: 1) adaptive thresholding; 2) a point labeling process; and 3) inward filling; have been proposed. The authors [LSC05] have reported a recall of 91.1% and a precision of 90.8% for a test of video sequences covering news, financial reports, sport and advertisements.

Chen et al. [CSB01] first detect edges in an image using the Canny operator. Second, in order to reduce the computational complexity, only one edge point for each small window is used during the estimation of scale and orientation. Then, some edge enhancement tasks takes place using the extracted scale information. Morphological dilation is performed to connect the edges into clusters. Some heuristic knowledge, such as the horizontal-vertical aspect ratio and height, is used at the end to filter out non-text clusters.

3.2.2 Texture-Based Methods

Texture-based methods are based on the observation that text present in images exhibits some distinct textural properties, which may be used to distinguish it from the background. Gabor filters, Wavelets, Fast Fourier transformation, etc. are usually used to extract the textural properties of a text region in an image.

Wu et al. [WMR99] have proposed an automatic text extraction system which at first uses distinctive characteristics of texts, such as the fact that text possesses certain frequency and orientation information or that characters of a text line show spatial cohesion to identify the possible text regions in an image. As a second step, bottom-up methods are applied to extract connected components. A simple histogram-based algorithm is proposed to automatically find the threshold value for each text region, to be used during the binarization step.

Li et al. [LDK00] scan the image using a small window of 16×16 pixels, and classify it as text or non-text using a three-layer neural network, based on the local features extracted from the high-frequency wavelet coefficients. For a successful detection of various text sizes they propose a three-level pyramid approach. A projection profile analysis is used to extract text elements from text blocks.

Kim et al. [KJPK01] use Support Vector Machines (SVMs) [Vap98] for classifying the textural properties of text in images. SVMs work well even in the high-dimensional space and can incorporate a feature extractor within

their architecture. After texture classification using SVM, a profile analysis is performed to extract text lines.

Chen et al. [CBT01] first detect possible text lines on the basis of edge analysis, baseline location and heuristic constraints, then employ a SVM to verify the real presence of text in the detected text lines. Texture features extracted from the edge-based distance map are used by the SVM.

Lienhart and Wernicke [LW02] have proposed an approach for detecting mainly horizontally aligned texts. Text lines are identified by using a multi-layer feed-forward network trained to detect text at a fixed scale and position. The gradient image of the input RGB image serves as their feature for text localization. A multi-scale schema is used to detect texts of different sizes. Localized text is scaled to a fixed height and segmented into a binary image.

Hao at al. [HYZM03] have presented an approach for text detection in video frames which works as follows. First, they use a color image edge detector to segment the image. Then, an artificial neural network is used for further classification of the text blocks and non-text blocks, using features obtained by Gabor filtering. To improve the precision of the system, the neural network is trained with every block which is falsely classified as a text block, until the desired results are obtained.

Ngo and Chan [NC05] have addressed the problem of text detection in video frames. The novelty of their approach is mainly based on its ability to adaptively detect the complexity of the current frame and consequently apply the appropriate operations for images of different complexities. The proposed method can distinguish between four types of complexities and accordingly it can apply four sets of different operations. At the end, an adaptive thresholding algorithm is proposed to segment the text image.

Ye et al. [YGWZ03, YHGZ05] have proposed an approach which consists of two parts. The first part deals with the initial localization of the text lines. First, the text pixels are detected based on their wavelet energy. Then, a density-based region growing process follows to connect the detected text pixels into possible text regions. Structural information are employed to divide the text areas into text lines. The second part consists of verifying the localized text lines by employing a SVM classifier.

3.3 Text Detection and Localization in Images/Videos

In Section 1.1.2, the characteristics of text in images/videos were summarized. From the point of view of multilingual text analysis, text characteristics can be divided into language dependent and language independent. Characteristics such as contrast, color, text polarity, alignment/orientation, motion,

compression and background characterize text of any language, whereas size of the text, inter-character distance, aspect ratio, stroke density and stroke statistics vary from Latin languages to Arabian or Chinese languages. Text usually possesses a large part of the characteristics described in Table 1.1. However, there are also cases, when for example the contrast of the text is not significantly high due to the complexity of the background where it is embedded.

Our intention is to build an automatic text detection and localization system which is able to accept different types of still images (or video streams) possibly with a complex background, analyze them and generate as output the coordinates of the found text, which in turn act as input for the text segmentation and binarization methods in Chapter 5. The design of the text detection/localization methods is based on the following assumptions:

1. The input to our system can be a grayscale or a color image as well as a video stream.

2. Only texts with a horizontal alignment are considered. This is mostly the case for the artificial text and often for the scene text.

3. Texts that are smaller than a certain (small) font size will not be detected. Text recognizable by humans is usually bigger than a certain size; therefore very small text will not be covered by the proposed methods.

In contrast to many other text detection approaches, our complete implementation has been written in the JAVA programming language, which allows the code to be easily distributed and run in parallel on heterogeneous platforms connected via the Internet. This allows to treat text localization as a scalable compute-intensive application of the *Grid computing* paradigm [FK03, EFGF04].

In the following, three solutions to the problem of text detection/localization are presented. First, in Section 3.3.2 two projection-based methods are introduced, then a texture based method is presented in Section 3.3.3.

3.3.1 Image Preprocessing

If the image data is not represented in the YUV color space, it is converted to this color space by means of an appropriate transformation. In contrast to the approaches presented in [AD99, LW02] where the RGB color space is used during further processing, the proposed system only uses the luminance data (Y channel of YUV). The Y component is evaluated from the RGB color space using Formula 3.1:

$$Y = 0.299R + 0.587G + 0.114B \tag{3.1}$$

After that, luminance value thresholding is applied to spread luminance values throughout the image and increase the contrast between the possibly interesting regions and the rest of the image.

3.3.2 Projection-Based Methods

Projection-based methods belong to the first category of text detection/localization approaches, namely region-based. They are top-down approaches, which means that they take as a first solution the whole image and then try to split it up in smaller regions iteratively using different properties. These methods are mainly based on the assumption that the text-background contrast is high and furthermore the density of edges in the areas of text contours is higher compared to the other parts of the image.

In this subsection, two novel projection-based methods are proposed. Both methods consist essentially of three steps: (I) Image preprocessing; (II) Edge detection; and (III) Text line localization analyzing the projection profiles. They differ from each other in the way how they proceed in Step 3, where the projection profile is analyzed. The first method employs a global threshold for the localization of text lines, whereas the second method employs an adaptive threshold depending on the complexity of the image.

In the subsequent paragraphs, the processing steps of the proposed methods are presented.

3.3.2.1 Edge Detection

This step focuses the attention to areas where text may occur. A simple algorithm similar to the Roberts operator [Eff00] to convert the grayscale image into an edge image is proposed. The algorithm (see Algorithm 3) is based on the fact that the character contours have high contrast to their local neighbors. As a result, all character pixels as well as some non-character pixels which also show high local intensity contrast are registered in the edge image. The algorithm functions as follows. The value of each pixel of the edge image ($edgeImg$) is evaluated as the largest difference between the grayscale values of the respective pixel in the original image and its neighbors (in horizontal, vertical and diagonal direction).

On the left of the Figure 3.1(b), the edge image of the original image in Figure 3.1(a) is shown.

Other edge detection algorithms such as Sobel, Prewitt or Canny edge detectors can also be applied to detect the edges (see Chapter 2).

Before proceeding with the next step, some noise edges are filtered out by means of a convolution with an appropriate mask (e.g. median filter).

Algorithm 3: The pseudocode of the edge detection algorithm

Input: A grayscale image: $grayImg_{M \times N}$

Output: The calculated edge image: $edgeImg_{M \times N}$

```
1  leftD = 0, upperD = 0, rightUpperD = 0;
2  for x = 0 to M − 1 do
3      for y = 0 to N − 1 do
4          if (0 < x < M − 1) then
5              leftD = |grayImg(x, y) − grayImg(x − 1, y)|;
6              upperD = |grayImg(x, y) − grayImg(x, y − 1)|;
7              rightUpperD = |grayImg(x, y) − grayImg(x + 1, y − 1)|;
8              edgeImg(x, y) = MAX(leftD, upperD, rightUpperD);
9          else
10             edgeImg(x, y) = 0;
11         end
12     end
13 end
14 edgeImg = sharpenEdges(edgeImg);
```

3.3.2.2 Horizontal Projection Profile

The horizontal projection profile HP (or the line edge histogram) of the edge image is computed as shown in Equation 3.2:

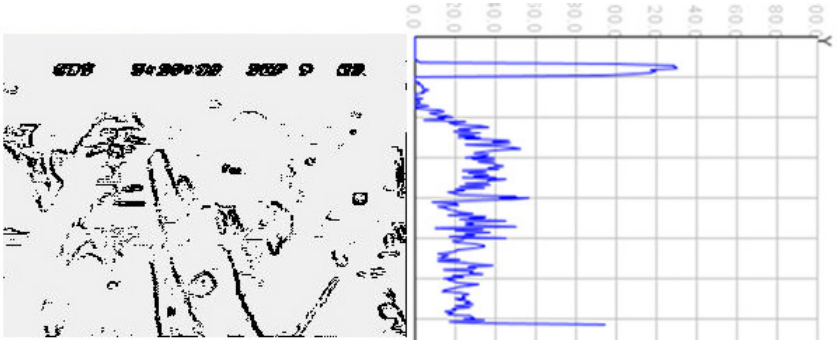
$$HP[y] = \sum_{\forall x | edgeImg(x,y) > edgeStrength} 1 \text{ for } y = 1..M \tag{3.2}$$

where the y^{th} element of HP shows the number of edge pixels in line y of the edge image exceeding a given value (e.g. $edgeStrength = 12$). In Figure 3.1(b) (right side), the evaluated horizontal projection profile HP for the edge image on the left side is presented.

3.3.2.3 TDL based on Global Thresholding

The TDL based on global thresholding (TDL-Global) analyses the horizontal projection profile of the evaluated edge image in order to locate potential text areas. Since text regions show high contrast values, it is expected that they produce high peaks in the horizontal projection (see Figure 3.1(b)).

(a) The grayscale image



(b) The respective edge image and its horizontal projection

Figure 3.1: Illustration of the TDL based on global thresholding

TDL via Global Thresholding. The text line candidates are found by the analysis of the line edge histogram HP employing two different thresholds, namely $MinEdges$ and $MinLineDiff$. A line y of the image is accepted as a text line candidate if either it contains a sufficient number of strong edges ($HP[y] >= MinEdges$) or the difference between the edge pixels in the current line to the previous line is bigger than a threshold ($|HP[y-1] - HP[y]| >= MinLineDiff$). Both thresholds are defined empirically and are fixed.

In this way, a text region is isolated which may contain several texts aligned horizontally (whereby their y-coordinates are already defined). In a later step, the x-coordinates of the leftmost and rightmost, top and bottom points of the text region are defined. The exact coordinates for each of the detected areas are used to create bounding boxes. The pseudocode of the TDL-Global method is given in Algorithms 4, 5 and 6, respectively.

Algorithm 4: The pseudocode of the text region localization algorithm

Input: An edge image: $edgeImg_{M \times N}$

Output: The coordinates of the detected text regions: TC

1 HP = calculateHorizontalProjectionProfile($edgeImg$);

2 TC = determineYCoordinate(HP) (using Algorithm 5);

3 TC = determineXCoordinate($edgeImg, TC$) (using Algorithm 6);

3.3.2.4 TDL based on Local Thresholding

The TDL based on local thresholding (TDL-Local) method is an extension of the TDL-Global method introduced in the previous section.

TDL-Global employs two global thresholds during the analysis of the histogram HP in order to identify text lines. However, it is clear that the employment of global thresholds limits the detection performance since the number of strong edges per line strongly depends on the background complexity and the contrast text-background. Consequently, the basic idea of the TDL-Local method is the application of a local threshold.

During a conducted empirical study, it was observed some similarity between the problem of detecting text in a down-sampled difference sequence of horizontal projection profile histograms (HP) and the problem of detecting cuts in videos using frame histogram differences. Thus, the application of an adapted version of a local thresholding technique instead of a global thresholding one is suggested. A similar approach has been successfully used in a well known video cut detection algorithm [YL95]. In the following, the novel TDL method based on a local thresholding technique is described.

TDL via Local Thresholding. In order to reduce noise effects, the histogram values that were produced in Section 3.3.2.2 undergo a down-sampling process. As the result of this process, it is expected that very high peaks on the down-sampled histogram will be caused by the beginning and the end of a text line. Additionally, down-sampling allows dealing with anti-aliased texts as well as with texts that are not aligned perfectly horizontally. The down-sampling process consists of the calculation of the two histogram difference sequences D and D' using the following equations:

$$D[\frac{y}{SF}] = HP[y + SF] - HP[y], \forall\ y \text{ such that } (y\ \%\ SF = 0) \tag{3.3a}$$

$$D'[\frac{y}{SF}] = \text{absolute}(D[\frac{y}{SF}]) \tag{3.3b}$$

where y is the line number and SF denotes the down-sampling factor. On the right side of Figure 3.2(b) an example of a sequence D' is presented.

Algorithm 5: The pseudocode of the algorithm, which determines the Y coordinate of a text region

```
/* Let textRegion be a data structure with four fields:
   x_0, y_0, x_1, y_1                                          */
Input: The line histogram: HP; MinLineDiff, MinEdges,
       MinLines
Output: The Y coordinates of the detected text regions:
        textRegion[] TC
1  textRegion textCandidate;
2  insideTextArea = false, y = 1, j = 0;
3  for i = 1 to M − 1 do
4      if (HP[i] > MinEdges) OR
       ((HP[i] − HP[i − 1]) > MinLineDiff) then
5          if not insideTextArea then
6              textCandidate(y_0) = i;
7              insideTextArea = true;
8          end
9      else if insideTextArea then
10         textCandidate(y_1) = i − 1;
11         if (textCandidate(y_1) − textCandidate(y_0) > MinLines)
           then
12             TC[j] = textCandidate;
13             j = j + 1;
14         end
15         insideTextArea = false;
16
17 end
```

The main idea of the TDL-Local method is to detect large single or double peaks in D' that represent either the beginning or the end of a text line. It is assumed that in case of a text line beginning (y) the sign of $D[\frac{y}{SF}]$ is positive since the number of edges increases when a text appears, and $D[\frac{y}{SF}]$ is negative in case of text line ending. A sliding window technique is used in

Algorithm 6: The pseudocode of the algorithm which determines the X coordinate of a text region

Input: Detected text regions and the edge image: textRegion[] $TC[]$, $edgeImg_{M \times N}$

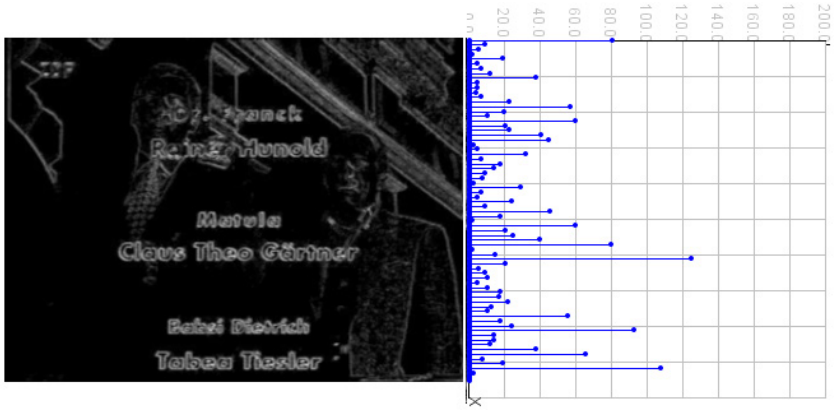
Output: The X coordinates of the detected text regions: textRegion[] TC

```
1  leftX = maxInt, rightX = −1;
2  for i = 0 to TC.length do
3      textCandidate_i = TC[i];
4      forall pixel_{x,y} ∈ textCandidate_i do
5          if edgeImg_{x,y} ≠ 0 then
6              if leftX > x then
7                  leftX = x;
8              end
9              if rightX < x then
10                 rightX = x;
11             end
12         end
13     end
14     textCandidate_i(x_0) = leftX;
15     textCandidate_i(x_1) = rightX;
16     TC[i] = textCandidate_i;
17 end
```

order to consider local image properties, such as the variation of background in case of text detection. For each text candidate at line j, a sliding window of width $2k + 1$ (with the following elements: $\{D'[j - k]$, $D'[j - k + 1]$,..., $D'[j + k]\}$) is considered, where k is a positive integer which defines the considered neighborhood. A line j is accepted as the beginning of a text line, if within a sliding window of size k, the value of $D'[j]$ satisfies the conditions 1 and 2 and either a or b.

1. $D'[j]$ is the *maximum* within the window AND EITHER
   (a) $(D'[j] > R * D'[s])$, where $D'[s]$ is the 2^{nd} *largest* value in the window and R is a positive number

(a) The grayscale image



(b) The respective edge image and its histogram of absolute differences D'

Figure 3.2: Example for the sequence of absolute histogram differences for an image

OR

(b) The 2^{nd} *largest* value $D'[s]$ is a *neighbor* of $D'[j]$, AND $(D'[s] > R * D'[t])$, where $D'[t]$ is the 3^{rd} *largest* value and $(\text{SIGN}(D[j]) = \text{SIGN}(D[s]))$ and R is a positive number;

2. $D[j] > 0$.

The line y of an image is considered as the end line of a text if the same conditions are fulfilled except for the last one, which is replaced by: $D[j] < 0$ (which means that the number of edges will be reduced when moving from line y to $y + 1$). The third condition (b) is included to deal with anti-aliased texts as well as with text not perfectly aligned horizontally. It should be remarked that the down-scaling factor SF and the neighborhood parameter k limit the detectable text height to a minimum of $2 * SF * k$ pixels.

Figure 3.3: The text detection/localization result when using the TDL-Global algorithm

(a) The results of TDL-Global method

(b) The results of TDL-Local method

Figure 3.4: Comparison of the text detection/localization results for both methods

3.3.2.5 Geometrical Analysis

Finally, geometric properties of the text characters like the possible height, width, width to height ratio are used to discard those text regions candidates whose geometric features do not fall into the predefined ranges of values.

3.3.2.6 Discussion

In this subsection, two novel projection-based text detection/localization methods were presented. The TDL-Global method operates on the basis of global thresholding techniques, whereas the TDL-Local method employs a local thresholding technique. In [GEF03a] and [GEF03b], additional details about these two methods and their performance in different set of images

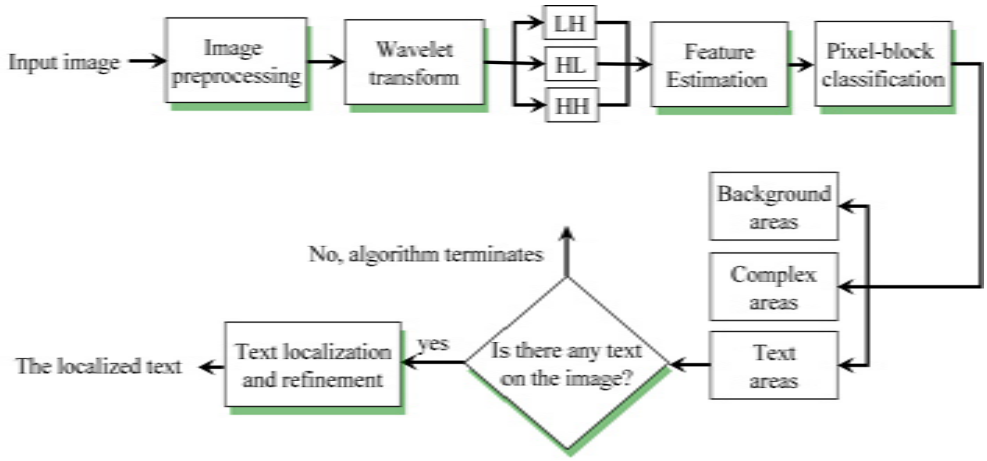


Figure 3.5: The flow chart of the unsupervised text detection and localization method

can be found. During the experiments, it was observed that the TDL-Global method performs very well in detecting text embedded in quite simple background although the contrast text/background may be quite low (see Figure 3.3), but its performance diminishes with the increasing of the background complexity. In contrast to this, the TDL-Local method detects and localizes the text more accurately, as illustrated in Figure 3.4. Yet, the TDL-Local method suffers in distinguishing a very complex background area (with the same geometrical characteristics as a text area) from a real text area.

Consequently, all these observations, drawbacks and advantages were analyzed and in the next subsection a more complex method will be proposed. The experiments in Section 3.3.3.10 will demonstrate its outstanding performance compared to the TDL-Global and TDL-Local methods and to other recent approaches proposed in literature.

3.3.3 Unsupervised Text Detection Based on High Frequency Wavelet Coefficients

In contrast to TDL-Global and TDL-Local, the new method that will be introduced in this subsection avoids taking the decision in its first stage based only on the analysis of the edge image's horizontal projection profile. Hence, the proposed method (UTDL-Texture) roughly consists of two parts: (I) Unsupervised texture-based TD; and (II) Projection profile-based refinement. During the first part, the algorithm operates on the assumption that text belongs to some generic class of textures, and it seeks to discriminate such a class from the many others present in an image by means of several texture

features combined with a clustering algorithm. The second part consists of refining the text detection results on the basis of projection profiles analysis.

Essentially, the proposed UTDL-Texture method works as follows: First, a wavelet transform is applied to the original image. Second, a sliding window is moved over the transformed image and the distribution of high-frequency wavelet coefficients is considered to characterize these areas. Using the k-means algorithm, the image is categorized into three predefined clusters: "text", "simple background" and "complex background", based on the extracted features. The choice of an unsupervised clustering method allows overcoming the dependency on training data and data selection of supervised methods. Third, the detected text areas undergo a projection analysis (see Section 3.3.2) in order to refine their localization. Finally, the textual properties of all remained text candidates are verified in order to discard possible false alarms. The case of images having no text at all is also handled in our approach, in order to eliminate the possible false alarms which arise during the application of the clustering algorithm with a fixed number of clusters ($k = 3$).

Unlike the other approaches which use supervised learning methods (e.g. [LDK00, HYZM03, KJPK01, LW02]) to classify text or non-text areas in an image, an unsupervised method in which only few parameters must be set is proposed.

The UTDL-Texture method is designed to localize horizontally aligned text of different languages with an arbitrary font, color and size (but exceeding a minimum height). However, the method can be modified to also detect text with an arbitrary alignment (see Section 3.3.3.7). The flow chart of the UTDL method is presented in Figure 3.5, whereas its individual steps are listed in Algorithm 7 and will be explained in the following paragraphs.

3.3.3.1 Presence of Text

To enable the subsequent k-means algorithm to work even if images do not contain any text, a copy of the image is appended to the bottom of the original image and one text line is artificially overlaid over the appended image on a predefined position. If this is the only text candidate detected in Section 3.3.3.4, it is concluded that there is no text in the image, and the algorithm terminates after this step.

3.3.3.2 Wavelet Transform of the Image

One important purpose of the wavelet transform is to decompose a signal into sub-bands at various scales and frequencies. In the case of images, the wavelet transform is useful to detect edges with different orientations. The wavelet transform can be implemented using filter banks consisting of high-

Algorithm 7: The pseudocode of the UTDL-Texture algorithm

Input: An image: $originalImage_{M \times N}$

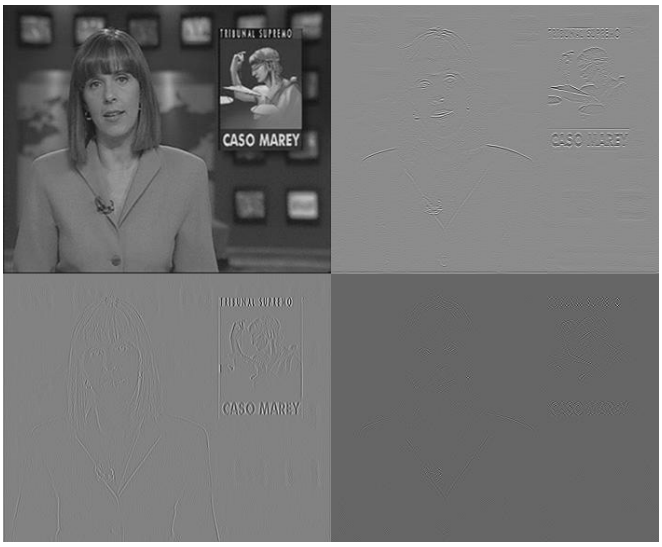
Output: The coordinates of the localized text: $textBox[]$

if *originalImage is a color image* **then**
  Convert $originalImage$ to a grayscale image: $grayscaleImg$;
end
$waveletImg$ = WaveletTransform($grayscaleImg$);
forall $block_{m \times n} \in waveletImg_{HL}$ **do**
  Create a feature vector: $features(x_1, ..., x_l)$;
end
Initialize k-means, where $k = 3$ ("text", "simple background" and "complex background" clusters);
Cluster the pixel blocks using the k-means algorithm;
Project the "text" cluster to an image with the same dimensions as the original image: $markedImage_{M \times N}$;
Estimate the connected components in the $markedImage_{M \times N}$: $cc[]$;
$textBox[]$ = BuildInitialBoundingBoxes($cc[]$);
$textBox[]$ = RefineInitialTextCoordinates($textBox[]$);
$textBox[]$ = VerifyTextCandidates($textBox[]$);

pass and low-pass filters. The application to an image consists of a filtering process in horizontal direction and a subsequent filtering process in vertical direction. For example, when applying a 2-channel filter bank (L: low pass filter, H: high-pass filter), four sub-bands are obtained after filtering: LL, HL, LH and HH. The three high-frequency sub-bands (HL, LH, HH) enhance at most edges in horizontal, respectively vertical or diagonal direction. Since text areas are commonly characterized by having high contrast edges, high valued coefficients can be found in the high-frequency sub-bands (see Section 2.4.2).

In our approach, a 5/3 filter bank evaluated by Villasenor et al. [VBL95] is chosen. Figure 3.6(b) presents the obtained four subbands from the application of the wavelet transform on the grayscale image shown in Figure 3.6(a).

(a) The grayscale image



(b) The four subbands of the 2D wavelet transform of the image

Figure 3.6: The illustration of the 2D wavelet transform of an image

3.3.3.3 Feature Vector Estimation

Several features are evaluated in order to find those who can distinguish better the textual texture from the other textures. They can be grouped into two main types: (I) intensity-based and (II) wavelet-based features.

The first group of features is directly extracted from the grayscale level (again only the luminance is considered), whereas the other type of features is evaluated on the wavelet domain.

A sliding window of size $m \times n$ (typical values are e.g. 32×8 or 16×8) pixels is moved over the image (grayscale image or its wavelet transformation) without overlapping, in order to calculate the different features. The value of

m is chosen usually bigger than n in order to describe the normal horizontal alignment of text. However, a slide step different from the width (i.e. height) of the window can be also selected. But the overlapping sliding windows will increase the number of objects (windows) to be clustered, which in turn will notably magnify the execution time of the k-means algorithm. Thus, the overlapping windows are avoided in this step and to compensate this, in Section 3.3.3.5, dilation and open operations are applied.

Intensity-Based Features. These features are computed on the grayscale image using only the Y channel similar to [WMR99]. Mean, standard deviation and the third-order central moment of the intensity values within a sliding window of dimension $m \times n$ are evaluated using Equations 3.4a, 3.4b and 3.4c.

$$\mu(window_{m\times n}) = \frac{1}{m*n} \sum_{\forall pixel_i \in window} Y[pixel_i] \tag{3.4a}$$

$$S_2(window_{m\times n}) = [\frac{1}{m*n-1} \sum_{\forall pixel_i \in window} (Y[pixel_i] - \mu)^2]^{-\frac{1}{2}} \tag{3.4b}$$

$$S_3(window_{m\times n}) = [\frac{1}{m*n-1} \sum_{\forall pixel_i \in window} (Y[pixel_i] - \mu)^3]^{-\frac{1}{3}} \tag{3.4c}$$

Wavelet-Based Features. For each window position and for each sub-band HL, LH, and HH, the mean and the standard deviation of the wavelet coefficients within the window are again evaluated. In addition, a new feature is also included, namely the standard deviation of the histogram HW of the wavelet coefficients within an window ($stDevH(window_{m\times n})$). This feature is computed as shown in Equation 3.5.

$$HW[i] = \sum_{\forall\ (x,y)\in window_{m\times n}\ |\ WaveletImg(x,y)=i} 1 \tag{3.5a}$$

$$f[i] = \begin{cases} i & \text{if } HW[i] = 0\ , \\ 0 & \text{otherwise} \end{cases} \tag{3.5b}$$

$$stDevH(window_{m\times n}) = [\frac{1}{K-1} \sum_{i=1}^{K} (f(i) - mean_f)^2]^{-\frac{1}{2}} \tag{3.5c}$$

where the value $HW[i]$ indicates the number of wavelet coefficients equal to i, K is the number of histogram bins, and $mean_f$ is the average of all $f[i]$. The choice of this feature was inspired from the observation reported by Lia and Gray [LG98]; "For pictures, the wavelet coefficients in the high frequency sub-bands, i.e., LH, HL and HH sub-bands, tend to follow a Laplacian distribution. According to the continuity of the distribution, it has been observed that the histogram of the coefficients of a text image suggests that the values

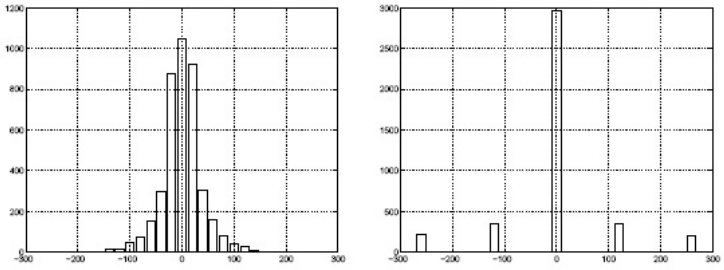


Figure 3.7: The histogram for the wavelet coefficients in the LH sub-band. On the left for a picture image; on the right for a text image (taken from [LG98])

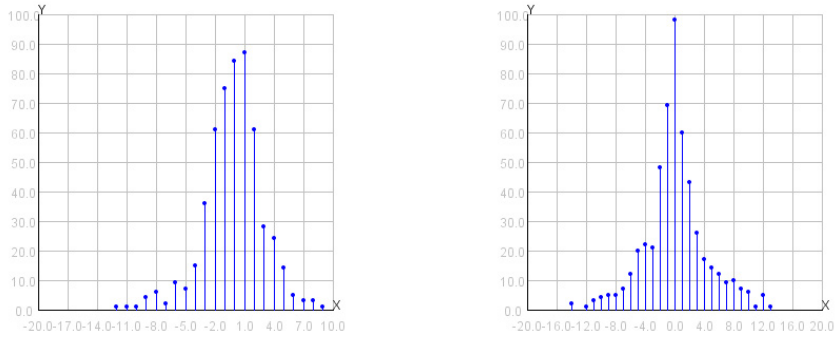


Figure 3.8: Illustration of the histogram for two sample non-text blocks

are concentrated on a few discrete values, whereas the histogram of a picture image shows a better continuity of distribution (see Figure 3.7)." However, this observation was made during the analysis of document images, which are less complex compared to the scene images and furthermore the text appears in black on a white background. Our observations have indeed shown that in the case of a complex image and a text embedded in a difficult background, the histograms are not as clear as those in Figure 3.7. Figures 3.8 and 3.9 illustrate the shape and the distribution of the histograms evaluated on "real" background and text areas. As both Figures 3.8 and 3.9 demonstrate, the histogram of a text area covers a broader range of coefficients. Moreover, it is also continuous due to the presence of many noise coefficients introduced by the background where the text appears. Consequently, it is expected that the text blocks will be characterized by higher values of the standard deviation of the histograms than other blocks.

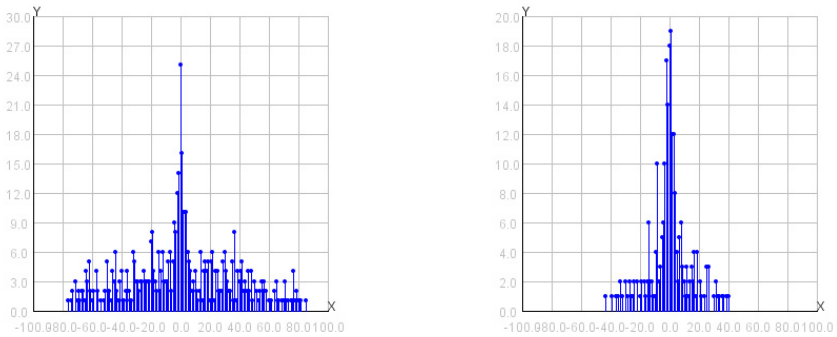


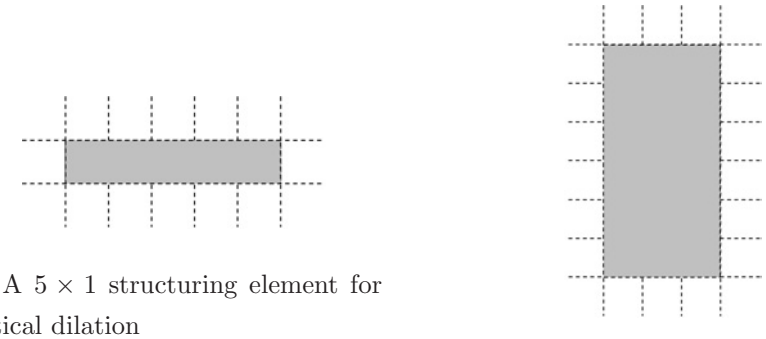
Figure 3.9: Illustration of the histogram for two sample text blocks

3.3.3.4 Unsupervised Pixel Block Classification

The k-means algorithm is employed to partition data objects into k clusters. It is assumed that the number of clusters k is known in advance. In our case there are two clearly distinguishable clusters of pixel blocks: "text" and "simple background". In order to deal even with complex background an intermediate category has been defined named "complex background". First, each component of the feature vector is normalized. For a given image, the maximum is computed for each component and used to normalize the components to an interval ranging from 0 to 1. Then, the "text" and "simple background" clusters are initialized with the feature vector whose corresponding elements have the minimum Euclidian distance to the ideal feature vector representing each of the clusters, named f_{text} and $f_{background}$. For the "text" and "simple background" cluster, this is a standard deviation of 1 and 0, respectively. The feature vector f_{mean} which is the mean of $f_{background}$ and f_{text} is calculated and is considered as the best representation for the cluster "complex background". The cluster named "complex background" is also initialized with the feature vector whose corresponding elements have the minimum Euclidian distance to f_{mean}. Finally, the classical k-means algorithm (see Section 2.9.2.1) is applied to obtain clusters whose members have the minimum Euclidian distance to the respective cluster mean feature vector. If there is only one text candidate in the "text" cluster and this is the one that has been inserted artificially in Section 3.3.3.1, the algorithm terminates and decides that there is not any text in the image. Otherwise, the classification step is repeated but only for the pixel blocks of the original image. Figure 3.10(a) illustrates the image blocks that are classified as "text".

(a) The classified text image (b) The filtered text image

Figure 3.10: The illustration of the unsupervised text detection process



(a) A 5×1 structuring element for vertical dilation

(b) A 3×6 structuring element for horizontal dilation

Figure 3.11: Vertical and horizontal "text" pixel dilation operators

3.3.3.5 Filtering and Initial Text Localization

First, a so called marked image is created based on the classified pixel blocks in the "text" cluster. Second, the possible gaps are filled applying a mathematical morphology operator, such as dilation. Two different structuring elements (e.g. 5×1 and 3×6 as illustrated in Figures 3.11(a) and 3.11(b)) can be used to dilate the marked image in the vertical and horizontal direction, respectively. The dilated marked image is then modeled as an undirected graph, where nodes represent text blocks and edges represent the fact that these text blocks are neighbors (eight-neighborhood). After that, the connected components are extracted applying a depth-first-search algorithm. Single, isolated components are discarded as "background" (see Figure 3.10(b)). Finally, the left-most, the right-most, and the top and bottom coordinates are taken to create an initial bounding rectangle. These coordinates act as the initial text

localization results, which will be further refined during the next step.

3.3.3.6 Refinement of Text Coordinates

As a result of the previous step, several text lines may have been localized together. In order to find the coordinates of single lines of text, in this step some refinement tasks take place. For this purpose, different algorithms can be applied. As a first solution, the projection-based algorithms introduced in Section 3.3.2 can be employed. Furthermore, in the following two new algorithms to split the localized multiple text lines will be presented.

Algorithm SDB: Standard deviation based. This solution is based on an empirical observation that the distances between sharp edges in the vertical direction will be more regular in case of a text line (the inter-character distance is usually uniform) than in the case of textured background. As a result, it is expected that the text lines will be characterized by a lower standard deviation of edge distances than the others. Given a text box $textBox$, the refinement algorithm works as follows.

1. For each line i, $line_i \in textBox$: Calculate the distances ($D_i[\,]$) between successive strong edge pixels in the LH sub-band image.
2. Evaluate the standard deviation $stDev_i$ of the distances $D_i[\,]$ for each line i individually.
3. Divide the original $textBox$ into two classes of smaller text boxes $textBox_k$, such that all $textBox_k$ that belong to one of the classes satisfy one of the following conditions:
   (a) $textBox_k \in class_1$, then $stDev_i < MaxStDev$, $\forall\ line_i \in textBox_k$
   (b) $textBox_k \in class_2$, then $stDev_i >= MaxStDev$, $\forall\ line_i \in textBox_k$
4. Disregard all text boxes, which belong to the second class, as false alarms; let the text boxes of the other class undergo subsequent verification tasks in Step 3.3.3.8.

In Figure 3.12(b), the results of this algorithm are illustrated. The experiments have shown (see also Figure 3.12(b)) that the algorithm performs very well in the cases where the text lines to be divided consist of the same length, but fails in splitting up two text lines of different lengths. Consequently, algorithm IAPT is proposed to deal with the separation of text lines of different lengths.

Algorithm IAPT: Iterative adaptive profile thresholding. This algorithm employs a coarse-to-fine methodology and it is based on an iterative adaptive analysis of the horizontal (i.e. vertical) projection profile of the text

(a) The TDL result from Step 3.3.3.5

(b) Aplication of Algorithm SDB

(c) Aplication of Algorithm SDB + IAPT

Figure 3.12: The result of the refinement tasks using the algorithm based on the analysis of the standard deviation alone and combined with the iterative adaptive profile thresholding methodology

box to divide it into its composed text lines. Given a text box $textBox$, the algorithm works as follows. First, the so called "union wavelet image" is created using Equation 3.6.

$$union_{LH \cup HL}(x,y) = \begin{cases} 1 & \text{if } HL(x,y) >= T_{horizontal} \text{ or } LH(x,y) >= T_{vertical}, \\ 0 & \text{otherwise} \end{cases} \tag{3.6}$$

where $T_{horizontal}$ and $T_{vertical}$ specify thresholds for the minimum horizontal and vertical edge strength, respectively. Both constraints are defined applying the well known Otsu algorithm [Ots79] on the absolute values of the HL (LH) coefficients. Second, the horizontal projection profile of $textBox$ is evaluated using the $union_{LH \cup HL}$. Then, the text box is divided into thinner

areas, such that the criteria "*the number of strong edges in each line within area is greater (or smaller) than* T_{width} " is fulfilled. The value of T_{width} is dependent on the width of the current $textBox$ (empirically set to $T_{width} = \frac{\text{width}(textBox_j)}{5}$). In the same way, the newly created text boxes ($textBox_k$) are refined analyzing the vertical projection profile (the threshold was again set empirically to $T_{height} = \frac{\text{height}(textBox_k)}{2}$). The last two steps are repeated until no more divisions are possible. All generated text boxes are then verified using the conditions proposed in Section 3.3.3.8.

Figure 3.12(c) illustrates the results of the refinement process when Algorithm IAPT is applied. In contrast to Figure 3.12(b), all text lines are localized separately in Figure 3.12(c) although they have different lengths.

3.3.3.7 Localization of Text with Arbitrary Alignment

The proposed unsupervised text detection process is independent of the text alignment, whereas the text localization methods introduced previously assume that text is horizontally aligned. In this paragraph, an algorithm which deals with the localization of text of any alignment is proposed. The main steps of the method are stated in Algorithm 8.

At the beginning, the connected components (cc) are evaluated. Then, for each cc the method proceeds as follows. The first and last intersection points with the current cc of the perpendicular to the x-axis at different positions are first determined. Second, in order to diminish the influence of the "noise" points, the coordinates of the points that lie in the middle of the extracted points at each position are evaluated. Then, the line ($y = a + bx$), which fits better with the set of points in the least squares sense is calculated by applying linear regression. Linear regression attempts to model the relationship between the variables x and y with a straight line fit to the data points and tries to determine the coefficients a and b by the condition that the sum of the square residuals is as small as possible. Finally, the initial coordinates of the polygon that bounds the present text (or cc) are defined as the intersection points between the two perpendiculars to the generated straight line at its extreme points with the two parallels to the line at a predefined distance (e.g. the largest distance of the contour's points of the cc from the line). This process is illustrated in Figure 3.13(c), whereas in Figure 3.13(d) samples of the evaluated polygons are shown.

The polygon (or cc) can be rotated with an angle of $(\arctan(b) * \frac{180}{\Pi})$ degrees using a shear transformation in order to have the text aligned horizontally. At this point, the methods explained in the previous paragraphs can be applied to refine the localization results. Furthermore, the subsequent verification tasks in Section 3.3.3.8 can be employed.

Algorithm 8: The pseudocode of the algorithm which localizes text of an arbitrary alignment

Input: Detected text blocks using k-means

Output: The coordinates of the polygon surrounding the present texts: $polygon[]$

```
1  shift = 8;
2  Find the connected components cc[];
3  for i = 0 to cc.length do
4      Find the x-coordinates of the leftmost (rightmost) lower points
       of the cc[i]: x_left-below, x_right-below;
5      for x = x_left-below to x_right-below do
6          Erect the perpendicular to the axis of abscissas at point x;
7          Determine the points where the perpendicular first and last
           intersects the cc[i]: first[] and last[];
8          x = x + shift;
9      end
10     mean[] = MEAN(first[], last[]);
11     Find the line describing the data points contained in mean[] in
       the least squares sense by applying linear regression;
12     Determine the coordinates of the polygon: polygon[];
13 end
```

3.3.3.8 Text Candidates Verification

The candidate text boxes that remained are then subject of a verification procedure. All text boxes are examined for typical text characteristics in order to remove present false alarms. First, all text candidates that do not fulfill the geometrical constraints stated in Equations 3.7a and 3.7b are regarded as false alarms and are removed from further verification procedures.

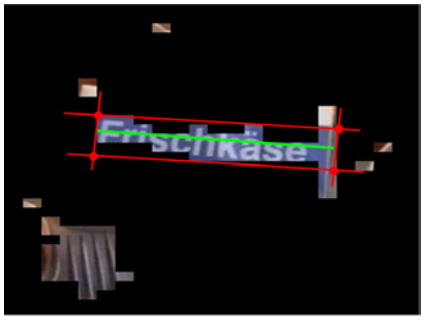
$$minTextHeight \leq \text{height}(textBox) \geq maxTextHeight \tag{3.7a}$$

$$\frac{\text{height}(textBox)}{\text{width}(textBox)} \geq minHVRatio \tag{3.7b}$$

where $minTextHeight = 8$, $maxTextHeight = 40$ and $minHVRatio = 0.5$. Furthermore, several features derived from the wavelet domain are evaluated, in order to further reduce the number of false alarms. To reduce the impact

(a) The original image

(b) The detected text blocks using k-means



(c) The generation of the polygon

(d) The result of localization

Figure 3.13: The localization of the text of an arbitrary alignment

of different character sizes on the evaluated features, the text candidates are first normalized scaling them to a preset height, while preserving the aspect ratio width/height. Then, the following properties are examined for each remaining text candidate $textBox$.

$$\frac{1}{h*w}\sum_{\forall(x,y)\,\in\,textBox} union_{LH\cup HL}(x,y) \geq minEdgeFillingFactor \tag{3.8a}$$

$$\frac{1}{h*w}\sum_{\forall(x,y)\,\in\,textBox} |HL(x,y)| \geq minHorizontalEdgeEnergy \tag{3.8b}$$

$$\frac{1}{h*w}\sum_{\forall(x,y)\,\in\,textBox} |LH(x,y)| \geq minVerticalEdgeEnergy \tag{3.8c}$$

where $minHorizontalEdgeEnergy = 17$, $minVerticalEdgeEnergy = 6$, $minEdgeFillingFactor = 0.31$ and $h * w$ denotes the area of the text box under consideration. The conditions (3.8b) and (3.8c) ensure the elimination

(a) Without verification (One false alarm)

(b) After verification

Figure 3.14: The impact of the verification step to the final text detection/localization results

of those candidates that do not contain enough vertical or horizontal edges with respect to the overall area of the candidate.

Finally, all candidates that do not satisfy at least one of the above introduced conditions are considered as false alarms. Figure 3.14 presents the localization results before and after the verification.

3.3.3.9 Multi Resolution Text Detection

The problem of detecting text of various sizes is solved by offering the user the possibility to apply a multi-level pyramid strategy. The original image will serve as the first level of the pyramid, whereas the images in the subsequent levels will be generated by halving the resolution of the respective image in the previous level. Thus, the image in the second level of the pyramid is obtained as a result of scaling down the resolution of the original image by two. The levels of the pyramid are given as parameter to the method. Then, the UTDL-Texture algorithm is separately applied at each level of the pyramid and at the end the results obtained from each run are fused together. If two detected text areas in different levels do not overlap with each other, then they are both accepted as independent text candidates; otherwise the text areas are merged in a bigger one, which contains all of them.

3.3.3.10 Performance Evaluation

The quantitative evaluation of text detection and localization methods is an open research issue due to two main reasons:

1. The lack of common image data bases.

2. The use of different measures by different researchers.

3. The non-existence of ground truth data. The ground truth data shows the exact position of a text in an image and has to be generated manually.

However, in this chapter the performance of the UTDL-Texture method is evaluated in terms of recall and precision on both the word and pixel level. Recall and precision are commonly used performance measures in the field of information indexing and retrieval. Recall is defined as:

$$\text{Recall} = \frac{\#\text{ Correctly localized words (text pixels)}}{\#\text{ Total number of words (text pixels)}} * 100 \tag{3.9}$$

whereas precision is defined as the number of correctly localized text words (pixels) divided by the number of all localized words (text pixels), including false alarms:

$$\text{Precision} = \frac{\#\text{ Correctly localized words (text pixels)}}{\#\text{ Correctly localized words (text pixels)} + \#\text{ False alarms}} * 100 \tag{3.10}$$

The word-based evaluation describes the word-based localization performance. In this case, correctness was determined manually by checking the localization results. A text word is considered as localized correctly, if the word is completely surrounded by a box (at least 90%) and its area is not significantly bigger than the ground-truth text area. A text box is considered as a false alarm, if there is no text in that box. The pixel-based measure evaluates the performance of the methods on the pixel level. First, the ground truth text boxes were drawn manually. Then, the evaluation is done automatically, comparing the ground-truth data with the localization results of the algorithm.

At the begining, evaluations have been conducted on two test sets containing various types of images (see Table 3.1). The first test set ($TestSet_{Mixed}$) consists of 51 images and covers a wide variety of background complexity and text types. These images (video frames) were captured from commercial televisions broadcasts, news videos, movie sequences and web pages. In total, there are 438 words in those 51 images. The second test set ($TestSet_{MPEG-7}$) consists of 45 video frames with a resolution of 384×288 pixels taken from the MPEG-7 video test set [XSLZ01]. There is a total of 241 words in this test set.

The UTDL-Texture algorithm was implemented as described in Section 3.3.3 and different parameter settings for W and H were tested. The best

Table 3.1: Detailed information about the test sets used during the experiments

Test Set	Resolution	# Images	# Words
$TestSet_{Mixed}$	384 × 288, 360 × 270, 285 × 198	51	438
$TestSet_{MPEG-7}$	384 × 288	45	241
Total		**95**	**679**
$TestSet_{Multilingual}$	352 × 240, 320 × 240	107	272 (Text boxes)

results were achieved when $W = 16$ and $H = 8$. A wavelet 5/3 filter bank evaluated in [VBL95] was used with low-pass filter coefficients {-0.176777, 0.353535, 1.06066, 0.353535, -0.176777} and the high-pass filter coefficients {0.353535, -0.707107, 0.353535}. Furthermore, the combination of Algorithm SDB and IAPT introduced in Section 3.3.3.6 led to the most accurate localization of the text. All the experiments were conducted using the original resolution of the images (i.e. one-level pyramid strategy).

In order to allow an objective comparison with the UTDL-Texture method, an alternative high-quality approach [CSL02] has been re-implemented. Several parameters have to be set in the approach of [CSL02] and, unfortunately, its localization performance depends noticeably on the parameter settings. Several experiments were conducted to estimate the parameters which gave the best results. Furthermore, this algorithm employs a coarse to fine projection analysis to localize the text candidates which is not described in detail in [CSL02]. The implementation of such a refinement process is done following a description which was kindly provided by the authors of [CSL02]. In this step, two thresholds are used: (I) the number of edges in a line ($numberEdgesLine$); and (II) the number of edges in a column ($numberEdgesColumn$). The following parameters are used during the experiments: $numberEdgeLine = 10$ and $numberEdgeColumn = 5$. In contrast to [CSL02], a threshold of 0.3 was used (instead of 0.6) to binarize the enhanced edge image as it led to better results. The remaining parameters were set as originally described in [CSL02].

The results obtained from the conducted experiments are shown in Table 3.2 (pixel-level) and in Table 3.3 (word level). The experimental results demonstrate the effectiveness of the proposed UTDL-Texture method, which has achieved an overall pixel-based recall of 84.43% and a precision of 72.31%,

Table 3.2: The detection and localization performance in terms of pixel-based recall and precision for the UTDL-Texture method and the re-implementation of [CSL02] for two test sets

Test Set	Method in [CSL02]		UTDL-Texture Method	
	Recall	**Precision**	**Recall**	**Precision**
$TestSet_{Mixed}$	74.43%	53.42%	82.56%	74.28%
$TestSet_{MPEG-7}$	76.60%	54.75%	88.88%	68.27%
Total	**74.36%**	**53.81%**	**84.43%**	**72.31%**

Table 3.3: The detection and localization performance in terms of word-based recall and precision for the UTDL-Texture method and the re-implementation of [CSL02] for two test sets

Test Set	Method in [CSL02]		UTDL-Texture Method	
	Recall	**Precision**	**Recall**	**Precision**
$TestSet_{Mixed}$	63.55%	87.74%	95.21%	96.30%
$TestSet_{MPEG-7}$	80.99%	91.16%	95.44%	88.80%
Total	**69.36%**	**89.06%**	**95.15%**	**93.47%**

whereas 646 out of 679 ground truth words were localized correctly leading to an overall word-based recall of 95.14% (see Table 3.3). Our re-implementation of the alternative approach [CSL02] was also tested for both test sets and the results are listed in Table 3.3 and 3.2. The re-implemented algorithm of [CSL02] has achieved an overall pixel-based recall of 74.36% and a precision of 53.81%, while on the word level a recall of 69.36% and a precision of 89.06% was obtained.

The experimental results show that the UTDL-Texture method clearly outperforms the re-implementation of [CSL02] in terms of both pixel level and word level evaluations. From Tables 3.2 and 3.3 it can be seen that the recall of the re-implemented approach [CSL02] is lower at the word level compared to the pixel level, whereas the UTDL-Texture approach achieves even a higher recall at the word level. This occurs because the conditions for

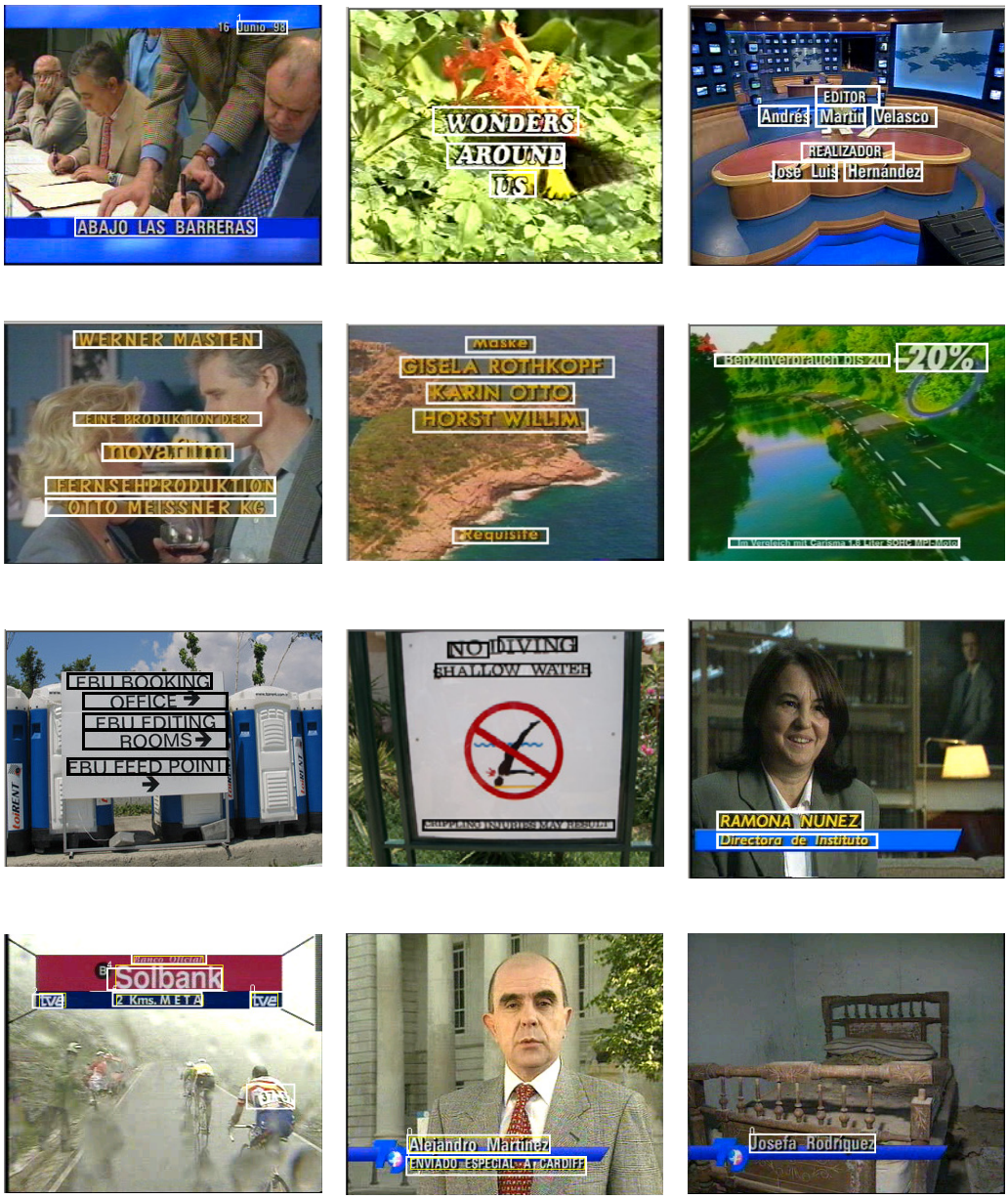


Figure 3.15: The text detection/localization results using the UTDL-Texture method

a word to be counted as localized correctly were satisfied more often while using the UTDL-Texture approach. In Figure 3.15, some exemplary results demonstrate visually the accuracy of the UTDL-Texture algorithm.

Table 3.4: The performance of the UTDL-Texture method and the re-implementation of [CSL02] during the analysis of the images with Chinese and Arabian text

Test Set	# Img.	GT TB	Method in [CSL02]		UTDL-Texture	
			Rec.	**Prec.**	**Rec.**	**Prec.**
$TestSet_{A+}$	47	181	64.1%	75.8%	79.0%	96.0%
$TestSet_{Ch+}$	60	91	62.7%	51.7%	96.7%	86.3%
$TestSet_{Mult}$	**107**	**272**	**63.6%**	**63.8%**	**78.3%**	**84.9%**

(a) The localization of Arabian text

(b) The localization of Chinese text

Figure 3.16: The text detection/localization results using the UTDL-Texture method

To evaluate the multilingual capabilities of the proposed UTDL-Texture approach, further evaluations are conducted using the test set named $TestSet_{Mult}$ ($TestSet_{A+}$+$TestSet_{Ch+}$). There are 270 text areas in those images, from which 164 consist of an Arabian (A) script, 82 of a Chinese (Ch) script and 26 of a Latin script (together with the other scripts as illustrated in Fig-

Table 3.5: Performance comparison on $TestSet_{MPEG-7}$ of the UTDL-Texture method with two other recent approaches

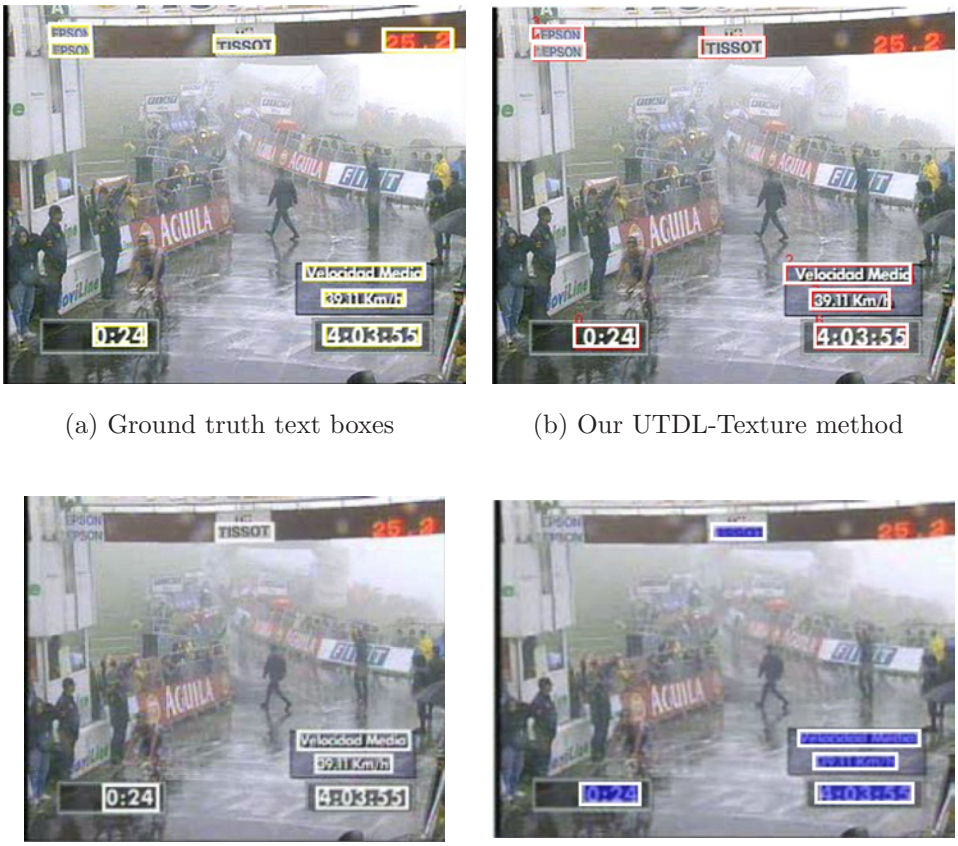
	# GT	**Recall**	**Precision**	**False alarm rate**
UTDL-Texture method	169	91.7%	92.3%	7.7%
Algorithm in [YHGZ05]	-	94.2%	97.6%	2.4%
Algorithm in [NC05]	169	91.5%	89.1%	10.9%

ure 3.16). The detailed results are given in Table 3.4 in terms of text box based recall and precision. 96.7% (i.e. 79.0%) of the Chinese (i.e. Arabian) text boxes were successfully localized, while a precision of 86.3% (i.e. 96.0%) was obtained using the UTDL-Texture method. The re-implementation of the [CSL02] has again shown a lower performance compared to the UTDL-Texture method with a recall and a precision of 62.7% and 51.7% in the case of Chinese text and a recall of 64.1% and a precision of 75.8% in the case of Arabian text. The low localization rate for the Arabian text is justified from the very low contrast and the small size of the text. In Figure 3.16(a), some illustrative examples are presented. No Chinese text was missed by our algorithm, except in the case where only a small part of the text appears separately as shown in the first image of Figure 3.16(b). These parts of text are lost during the refinement step.

In [YHGZ05] and [NC05], the same test set $TestSet_{MPEG-7}$ has also been used for performance evaluation and the results are reported on the text box level, which means that instead of the correctly detected words the correctly detected text boxes are counted to estimate the recall/precision. In Table 3.5, the text box based results of our method and those reported by the authors in their papers are shown. The second column named "# GT" presents the number of ground truth text boxes that are taken into consideration. The false alarm rate is defined using Formula 3.11.

$$\text{False alarm rate} = \frac{\#\text{ Falsely localized text}}{\#\text{ Total localized text}} * 100 \tag{3.11}$$

Our approach has achieved a text box based recall of 91.7% and a false alarm rate of 7.7%. Ye et al. [YHGZ05] have assessed the text detection algorithm in the same way as we have done and a recall of 94.2% associated from a false alarm rate of 2.4% are reported. Compared to our approach, the method in



(a) Ground truth text boxes

(b) Our UTDL-Texture method

(c) Taken from [YHGZ05]

(d) Taken from [NC05]

Figure 3.17: The illustrative comparison of three methods

[YHGZ05] has shown a slightly better recall rate and definitely a lower rate of false alarms. However, the authors do not give any details about the number of ground truth text boxes that were taken into account.

The best performance that Ngo and Chan have reported in their paper [NC05] is a recall of 91.5% and a false alarm rate of 10.9%. Compared to the approach of [NC05], our method has shown a better performance, attaining a slightly higher recall rate of 91.7% and a lower false alarm rate of 7.7% for the same number of ground truth text boxes. Nevertheless, no objective statement can be made about the quality of our approach compared to [NC05], since the authors of [NC05] have investigated other performance criteria, which include different weights for the detected text boxes in accordance with the detection difficulty and punishment of the detection and false alarm rate when a detected text box includes more than a ground truth text box. During our evaluation, all text boxes are equally weighted regardless of the background complexity and the difficulty to detect them.

In Figure 3.17, the visualized text detection results for our method and the methods of [YHGZ05, NC05] for a given image are presented. Figure 3.17(a) illustrates the ground truth text boxes, Figure 3.17(b) shows the localized text using our approach, whereas Figures 3.17(c) and 3.17(d) present the text boxes detected using the methods of [YHGZ05] and [NC05], respectively. From the figure it is clear that our approach has missed only one ground truth text box, whereas the other two methods have missed three of them.

To conclude this section, it should be stressed that although the experimental results are not entirely comparable with each other, the results of our proposed unsupervised solution for text detection/localization in images are competitive with the best results recently reported in literature.

3.4 Summary

In this chapter, the problem of text detection and localization in complex images was addressed. More attention was given to the detection/localization of artificial text, since its meaning is usually stronger related to the content of the image compared to the meaning of scene text. Consequently, three novel solutions were proposed.

The TDL-Global and TDL-Local methods try to localize the text in an image, by analyzing the horizontal projection of the corresponding edge image. The former method employs a global analysis, whereas the latter involves a locally adaptive thresholding technique inspired from the video cut detection field. Both methods perform quite well in cases of simple images, but their accuracy diminishes with the increase of the complexity of the background where the text appears. Thus, an unsupervised learning method was presented in Section 3.3.3 whose application significantly improves the text localization results. The standard deviations of high-frequency wavelet coefficients and of their histograms were used as the main features for the subsequent classification process. The classification was done by a slightly modified k-means algorithm. The text candidates undergo a projection profile analysis in order to refine the localization of the text. The very good experimental results show that the chosen features are robust with respect to the distinction between text regions and non-text regions in an image. Experimental results in Section 3.3.3.10 have shown the competitive performance of the UTDL-Texture method, compared also to other recent high-quality approaches proposed in the literature. Furthermore, in Section 3.3.3.7 an algorithm based on linear regression analysis was proposed to localize texts of arbitrary directions.

Chapter 4

Multiple-Frame Based Text Detection in Videos

The value of interpretation is in enabling others to fruitfully think about an idea.

- **Andreas Buja-**

4.1 Introduction

One of the main challenges in artificial text extraction is the widely varying complex background of the scene in general and of the part where the text is embedded in particular. This phenomenon leads to many false alarms, i.e. low precision and not very high recall rates.

Existing video text extraction methods can be grouped into two main categories. The first category of methods which were reviewed in detail in Chapter 3 deal mainly with text extraction in still images and treat a video stream as a sequence of still images. The second category employs the temporal information of the video. Many of them use low-level multi-frame integration techniques such as multi-frame averaging [LKD99] or time-based minimum/maximum pixel search [WJW04] during the text detection and segmentation tasks in order to enhance the contrast of the text and reduce the background complexity. Other video text extraction methods [HYZ02, LSC05] use inter-frame verification procedures to reduce the rate of false alarms and consequently to increase the accuracy of the system. The similarity between the text boxes in a set of frames is evaluated mainly in terms of their posi-

tions, but intensities and shape features are also used. These techniques have the drawback that usually many false alarms appear in several consecutive frames similar to text areas.

The method proposed in this chapter is based on the assumption that the same text always appears in several successive frames and has a relative stable position across the frames, whereas the background usually changes from one frame to the other. It differs from the existing methods [WJW04, HYZ02] in the way how the temporal information present in videos is used.

Our text detection method (*UTDL-Texture*) presented in Chapter 3 detects text blocks in individual frames independently by employing a k-means clustering algorithm. Considering the fact that in real-life videos static caption texts appear for at least 2 seconds and inspired from the principles and advantages of clustering ensembles techniques, the UTDL-Texture method is further extended and updated for video text extraction. First, instead of k-means, a fuzzy clustering algorithm is used, for adding more flexibility during the process of classifying the blocks of pixels. Second, a clustering ensemble is applied to make use of the temporal information in a video, by means of integrating the results obtained by applying the fuzzy C-means algorithm in different frames where the same static text appears. Third, a solution is proposed to determine the "correct" text cluster. Fourth, a simple procedure is presented to integrate text areas from different frames in order to reduce the complexity of the background and enhance the contrast. Finally, a coarse-to-fine projection based algorithm is employed to refine the localization of the text. Part of the material presented in this chapter has been accepted for publication in [GQF06a].

This chapter is organized as follows. Section 4.2 reviews current multi-frame based text detection and localization approaches. In Section 4.3, the fuzzy clustering ensemble algorithm originally proposed in [Qel06] is briefly explained. The adaptation of the fuzzy ensemble for text detection in videos is discussed in Section 4.4. Section 4.5 presents experimental results. Section 4.6 concludes the chapter.

4.2 Previous Work

In this section, an overview of the methods that make use of the temporal information during the detection/localization of the visual text in videos is given. The methods that use multi-frame integration techniques during the text segmentation are not subject of this section.

Tang et al. [TLG+02] first detect the appearance/disappearance of caption text in a video. Then, a grey-level vector that for each pixel captures the intensity values in each frame is created. Supposing that for a caption text pixel the vector will show stable intensities, whereas in the case of a back-

ground pixel the intensities will change significantly, the caption text pixels are identified.

Tang et al. [TGLZ02] employ a fuzzy-clustering neural network (FCNN) classifier to first detect the shots in a video and then localize the present caption (dis)appearance within a shot, based on a set of features extracted from frame differences. Then, the FCNN is again used to precisely localize the text lines directly on the critical frame difference (where each caption block appears or disappears).

Wang et al. [WJW04] apply a multiple frame integration process before the text detection task. After applying a time-based minimum (or maximum) pixel value search, a block-based text detection method is performed. A given block is classified as a text or non-text block based on features extracted using the Sobel operator. At the end, an iterative text line decomposition method is applied to refine the localization of the text.

Hua et al. [HYZ02] regard in their paper only the overlaid static text, too. They apply the text detection method on each frame and then use a multiple frame verification method to reduce false alarms. Then, a "man-made" frame is generated integrating text blocks from those frames where the text is most likely clear. Finally, a block-based adaptive thresholding procedure concludes the text extraction process.

Chang et al. [CCLL05] have proposed a system for Chinese text extraction in videos. First, the text is located at each frame by analyzing the edge image. If an area contains more than 10 lines where each of them includes at least M edge-pairs then it is identified as an text area. An edge-pair is a pair of two adjacent edges with opposite gradient signs. Second the caption changes are detected to find the set of successive frames that contain the same text. Then, the text is separated from the background employing an adaptive thresholding method. Finally, the segmented Chinese text is recognized using a SVM classifier.

There are a few publications which discuss the ensemble or fusion philosophy in the video object detection/localization context [Ant01, VJ01, JKB03, BH06, BVS05]. However, most of them are based on the application of the classification ensemble techniques. Thus, Antani [Ant01] considers a text localization method as a specific classifier. In order to improve the final text localization results, a given frame image is analyzed using several text localization algorithms, and the final result is generated combining the results obtained from the individual methods. Different combination rules such as: intersection rule, union rule, majority vote rule, generalized voting rule, weighted voting rule, etc., are discussed. Bargeron et al. [BVS05] have presented a framework which uses boosting to select and combine different input features into a robust text detector. Furthermore, transductive inference is employed to improve the results taken from the boosting. The used features are evaluated using a connected component analysis and describes their spa-

Algorithm 9: Template of the fuzzy clustering ensemble algorithm

Input: n objects to cluster; the number of clusters $c^{(Ensemble)}$

Output: The cluster ensemble partitioning

```
Let all elements of A_0 be initialized with 0;
for t=1 to N_max do
    Cluster the data using FCM algorithm in Chapter 2;
    for i=1 to n do
        for j=i to n do
            C[i,j]=SimilarityFunction(i,j);
        end
    end
    A_t = Merge(A_{t-1}, C);
end
Use similarity matrix A to create the final partitioning of the data;
Output the final partitioning;
```

tial relationships and their activity density. The resulting approach has been shown to detect the text fast, efficiently and accurately.

4.3 Fuzzy Cluster Ensemble

A classification ensemble consists of a set of classifiers, whose decisions are combined with the aim of improving the accuracy of the classification of a new object. Cluster ensembles are inspired by supervised learning approaches, i.e. classification ensembles. As mentioned in Chapter 2, the purpose of cluster analysis is to segment a group of objects into clusters where objects inside the same cluster are more closely related to each other than objects assigned to different clusters. In contrast to supervised classification, clustering is a data driven approach that does not require class labels. Since different clustering algorithms behave differently with different datasets and very often behave differently with the same dataset, cluster ensembles [SG02, Fre01, GMT05] have been introduced. The main purpose of cluster ensembles is to merge the results of several clusterings together in order to obtain a partition which ideally is robust and improves the quality of the individual clusters. In general, cluster ensemble algorithms consist of the following steps:

Algorithm 10: Template of the video text detection algorithm based on a fuzzy cluster ensemble

Input: N frames; the number of clusters $c^{(Ensemble)}$

Output: The coordinates of the detected text

1 $Partition_{Accumulated}$ = Output of Algorithm 11;
2 Find the text partition using $Partition_{Accumulated}$;
3 Use the text partition and the original N frames to create "the super text image";
4 Define the coordinates of the text on "the super text image";
5 Output the coordinates of the texts;

- **Step 1.** Obtain the different partitions of objects by applying different clustering algorithms or running several times the same clustering algorithm;
- **Step 2.** Build the similarity matrix for each obtained partition;
- **Step 3.** Merge the similarity matrices based on specific criteria;
- **Step 4.** Define the final partition of the objects using the resulting merged similarity matrix.

A way to improve the robustness of clustering is to combine several clustering algorithms, creating in this way a clustering ensemble. Qeli [Qel06] has proposed a fuzzy clustering ensemble, whose main steps are illustrated in Algorithm 9. The main idea is to combine the outputs of different fuzzy c-mean clusterings in order to improve the final partition of the data. The output of the algorithm depends on five criteria: I) The number of clusterings that take place: N_{max}; II) The fuzzy partition generated by each fuzzy C-means run in step 3; III) The similarity function used in step 6; IV) The way how the merging is done in step 9; and V) How the final partitioning is done in step 11. Qeli [Qel06] has discussed in his work all these criteria in detail.

In the following, the application of the fuzzy cluster ensemble for text detection in videos will be discussed.

4.4 Cluster Ensemble for TD in Videos

The fuzzy clustering ensemble (*FCE*) [Qel06] was extended and combined with the UTDL-Texture method for text detection in videos. In contrast

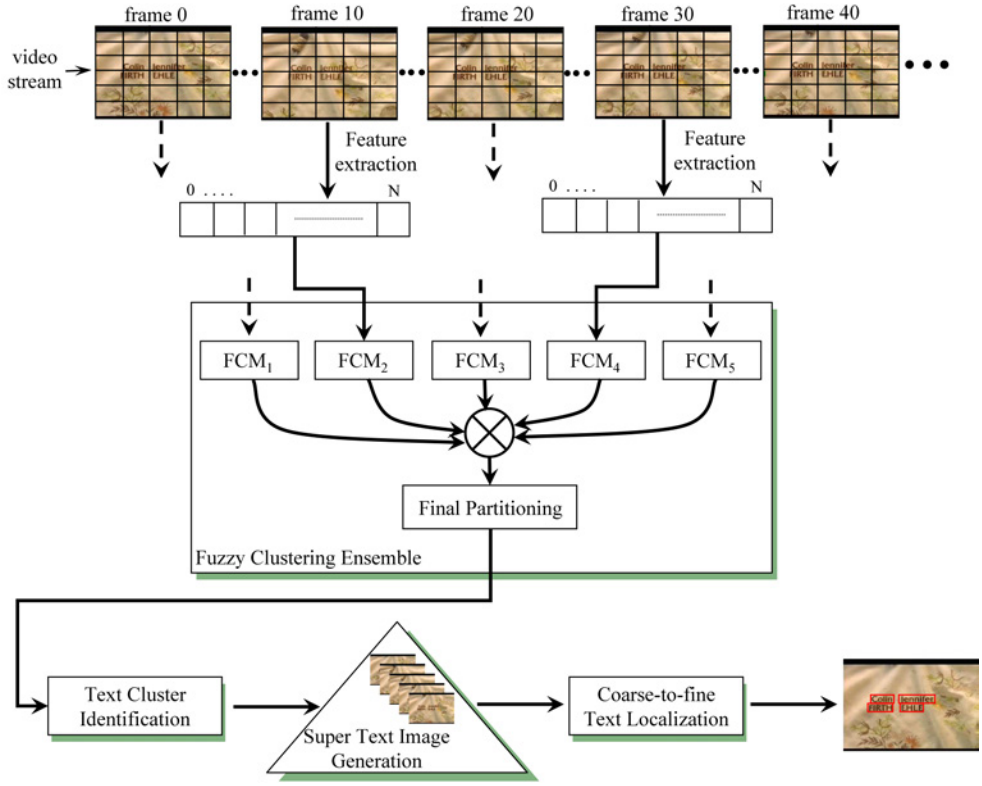


Figure 4.1: Fuzzy cluster ensemble for text detection in videos

to [Qel06] where the results of different runs of the FCM algorithm on the same set of objects (step 9 of Algorithm 9) are accumulated, we propose in Algorithm 11 to accumulate the results of runs of the FCM algorithm on "different" objects (i.e. spatially the same objects but their features are extracted from different frames). In this way, the temporal properties of the text in video which appears over several consecutive frames are used.

In Figure 4.1, the flow chart of the proposed fuzzy clustering ensemble for text detection in videos (*FCE-TDV*) is shown, whereas in the following subsections the details of the proposed method are discussed.

4.4.1 Application of a Fuzzy Clustering Ensemble

Let $S_c = \{f^1_{M\times N}, f^2_{M\times N}, ..., f^k_{M\times N}\}$ denote a complete sequence of video frames where the same static superimposed text appears. To reduce the computation time of our algorithm, only a subset (S_s) of S_c is selected for further investigation ($S_s \subset S_c$).

$$S_s = f^1_{M\times N}, f^{10}_{M\times N}, ..., f^i_{M\times N}, f^{i+10}_{M\times N}, ... \tag{4.1}$$

Algorithm 11: Template of the video based fuzzy cluster ensemble algorithm

Input: P frames, where the same text should be detected; the number of clusters $c^{(Ensemble)}$

Output: The video based cluster ensemble partitioning

```
Let all elements of A_0 be initialized with 0;
for p=1 to P do
    Divide Frame_p in T blocks of m × n dimension;
    Extract features features_t from each block_t;
    Cluster the blocks using FCM Algorithm stated in Chapter 2;
    for i=1 to T do
        for j=i to T do
            C[i,j]=SimilarityFunction(block_i,block_j);
        end
    end
    A_p = Merge(A_{p-1}, C);
end
Use the accumulated similarity matrix A_p to create the final
partitioning of the data;
Output the final partitioning;
```

Let us suppose that the proposed FCE consists of a fixed number of FCM instances denoted by P. Then, the FCE-TDV method which extends the UTDL-Texture method and in addition makes use of the clusterings ensemble techniques works as follows. First, consider the first P frames of set S_s. Second, divide each frame into blocks ($m \times n$) as described in Chapter 3. Third, extract the features from all created blocks in each frame separately. Fourth, relay the block features to the respective FCM instance as illustrated in Figure 4.1. The object of individual clusterings are blocks of pixels represented by a 2-dimensional set of features. Fifth, combine the individual clusterings by the FCE engine to yield the resulting three clusters, namely "text", "background" and "complex" clusters. Sixth, identify the "text" cluster among the created partitions and generate the so called "super text image" through the integration of the respective detected text blocks from different frames. Seventh, employ a coarse-to-fine projection method to extract the exact position of the text from the "super text image". Fi-

nally, verify all text candidates and discard possible false alarms (described in Section 3.3.3).

In the next subsection, the identification of the text cluster is explained in detail, followed by the description of the method used for the generation of the "super text image".

4.4.2 Text Cluster Identification

The output of the FCE are three clusters of data that are estimated from the accumulated partition matrix using the METIS method [KK98] (see Algorithm 11). The main problem raised at this point is the identification of the text cluster (i.e. the partition that contains the text blocks). For this purpose, a simple method is proposed. First, the features are accumulated by means of an averaging procedure that takes place along all frames analyzed by the FCE, as shown in Equation 4.2:

$$Features_{mean} = \frac{1}{P} \sum_{t=1}^{P} Features_t \tag{4.2}$$

where $Features_t$ denotes the set of features evaluated on the t^{th} frame. Then, the cluster c_k whose accumulated features have the smallest distance from the ideal text features ($FIdeal_{text}$ ={1, ..., 1}), is labelled as the text cluster. The Euclidean distance is used to estimate the distance of a cluster from the ideal text features vector, as presented in Equation 4.3:

$$\text{ODist}(c_k) = \frac{1}{|c_k|} \sum_{\forall\ data_i\ \in\ c_k} \text{Euclidean}(Feature_{mean}(data_i), FIdeal_{text}) \tag{4.3}$$

where $|c_k|$ shows the number of elements that belong to cluster c_k and $data_i$ represents a specific $m \times n$ block of pixels.

4.4.3 "Super Text Image" Generation

To integrate the text blocks from different frames, a simple technique similar to [MTM05] is proposed. Due to the fact that only static text is considered, no motion estimation is employed. The technique works as follows. First, the mean frame of the original frames is estimated using Equation 4.4.

$$F_{mean} = \frac{1}{\#\ \text{Frames}} \sum_{\forall i} frame_i \tag{4.4}$$

Then, two of the wavelet high-frequency sub-bands, namely LH and HL, are used to enhance the character edges in the horizontal and vertical direction

and reduce noise. Thus, two maximum high-frequency images are evaluated separately for each of the subbands (F_{max}^{LH} and F_{max}^{HL}) using Equation 4.5.

$$F_{max}^{LH\ (HL)} = \text{MAX}_{\forall i}(frame_i^{LH\ (HL)}) \tag{4.5}$$

where $frame_i^{LH\ (HL)}$ denotes the sub-band LH (HL) of the wavelet transformation of the grayscale frame $frame_i$. Then, the maximum high-frequency images are normalized as follows:

$$F_{max}^{LH\ (HL)}(i,j) = \frac{F_{max}^{LH\ (HL)}(i,j) - \text{MIN}(F_{max}^{LH\ (HL)})}{\text{MAX}(F_{max}^{LH\ (HL)}) - \text{MIN}(F_{max}^{LH(HL)})} \tag{4.6}$$

The final "super text image" is created by fusing together all the above evaluated images as given in Equation 4.7:

$$STI = \text{MEDIAN}(F_{mean} + F_{max}^{LH} + F_{max}^{HL}) \tag{4.7}$$

where MEDIAN denotes a median denoising filter which is applied after the fusion procedure.

4.4.4 Text Localization

After generating the "super text image", then the initial text boxes are evaluated using the technique introduced in Section 3.3.3.5. Then, one of the algorithms proposed in Section 3.3.3.6 is applied to refine the localization of the text. Furthermore, the verification procedure introduced in Section 3.3.3.8 is employed to reduce the number of false alarms.

4.5 Performance Evaluation

The proposed FCE-TDV method was tested with nine video sequences, whose details are described in Table 4.1. The first eight sequences were extracted from three different videos selected from the MPEG-7 content set [Gro98], namely "news1", "news2" and "animals". "news1.mpg" and "news2.mpg" are daily and weekly news broadcasts captured from a Spanish and Portuguese television program, respectively, whereas "animals.mpg" is an educational video about animals whose source is the Singapore Ministry of Education. "newsGer.mpg" is another news broadcast video captured from German television. The overall video sequences consist of a total of 10.92 minutes and 16363 frames of which 3905 frames contain text. There are 14657 ground truth text boxes in those 3905 frames. The ground truth text boxes usually contain more than one word and are counted manually. The

Table 4.1: Detailed information about the test sets used during the experiments

Sequence name	original video name	Length (in min)	# Frames	# Frames with text	# Frames without text
Seq_1	news1.mpg	01:24	2097	100	1997
Seq_2	news2.mpg	00:52	1280	377	903
Seq_3	news2.mpg	00:33	828	67	761
Seq_4	news2.mpg	00:54	1344	244	1100
Seq_5	news2.mpg	01:03	1584	230	1354
Seq_6	news2.mpg	00:40	1001	89	912
Seq_7	news1.mpg	02:01	3027	313	2714
Seq_8	animals.mpg	00:28	701	248	453
Seq_9	newsGer.mpg	03:00	4501	2237	2264
Total		**10:55**	**16363**	**3905**	**12458**

text that is present in those video sequences is static and the background of the scene is very challenging. Moreover 12458 frames without any text but rich in texture was also included to evaluate the precision of the proposed FCE-TDV method (see Figures 4.2 and 4.3). The performance of the FCE-TDV approach was evaluated using the following parameters. The FCE is composed of three different FCM instances ($P = 3$ in Algorithm 11). A sliding window of 32×16 dimension is used to estimate the features, whereas the standard deviation of the high-frequency wavelet coefficients (LH subband) and of their histogram were used as the main features. Every three successive I-frames are the subject of analysis of the FCE. For the refinement of the text coordinates, the algorithms SDB and IATP proposed in Section 3.3.3.6 are employed in combination with each other. Similar to other evaluations, throughout this work the wavelet 5/3 filter bank [VBL95] with the low-pass filter coefficients {-0.176777, 0.353535, 1.06066, 0.353535, -0.176777} and the high-pass filter coefficients {0.353535, -0.707107, 0.353535} were used to transform the image from the spatial domain into the frequency domain.

Furthermore, the original solution (UTDL-Texture method) introduced in Chapter 3 is employed on these video sequences in order to objectively assess the advantages of the FCE-TDV approach. Due to the fact that the UTDL-Texture method was originally proposed as a solution for text detec-

Table 4.2: The detection and localization performance in terms of TB-based recall and precision for the UTDL-Texture method and the FCE-TDV method

VSeq.	GT txtB	UTDL-Texture Method				**FCE-TDV Method**			
		CD txtB	FA	Rec. in %	Prec. in %	CD txtB	FA	Rec. in %	Prec. in %
Seq_1	200	194	475	97.0	39.0	161	46	80.5 ↓	77.8 ↑
Seq_2	7917	7465	103	94.3	98.6	7346	311	92.8 ↓	95.9 ↓
Seq_3	174	146	0	83.9	100	146	0	83.9 ·	100 ·
Seq_4	488	470	0	96.3	100	470	0	96.3 ·	100 ·
Seq_5	690	624	0	90.4	100	675	0	97.8 ↑	100
Seq_6	534	246	230	46.1	51.7	438	19	82.0 ↑	95.8 ↑
Seq_7	757	483	55	63.8	89.8	636	28	84.1 ↑	95.8 ↑
Seq_8	996	751	8	75.4	98.9	865	7	86.8 ↑	99.2 ↑
Seq_9	2901	2848	9	98.2	99.7	2759	48	95.1 ↓	98.3 ↓
Total	14657	13227	880	**90.24**	**93.76**	13490	459	**92.04**	**96.71**

tion/localization in still images, its application for TDL in videos proceeds as follows. First, the original UTDL-Texture method is applied to every I-frame in order to localize the present text. Then, an inter-frame verification procedure similar to other approaches [HYZ02] is employed to check the presence of the same text over successive frames and eliminate apart those boxes that represent possible false alarms. The position and dimension of two respective text boxes in different frames are used to decide the presence of the same text in those two text boxes, i.e. frames. Detected text boxes that do not appear over a sufficient (30 frames) number of consecutive frames are regarded as false alarms and are removed. During these evaluations, the parameters of the UTDL-Texture method are given the same values as in Section 3.3.3.10.

The results of our experiments are shown in Table 4.2. Column "VSeq" denotes the name of the video sequence, column "GT txtB" shows the number of ground truth text boxes in each of the video sequences, the columns "CD txtB" and "FA" give the number of correctly detected text boxes and the number of boxes which are falsely detected as text. The ground truth text boxes are manually counted and show the total number of text areas

Figure 4.2: Text detection and localization results obtained using the UTDL-Texture method

(i.e. lines) that appear in these video frames. The columns "Prec." and "Rec." show the obtained values of precision and recall in percentage values. The last row of the table presents the overall achieved performance for both approaches, whereas the previous rows present the performances of the approaches when analyzing the video sequences separately.

From Table 4.2 it can be seen that the slightly modified version of the UTDL-Texture method has correctly localized 13227 of 14657 ground truth text boxes, achieving an overall recall of 90.24%, whereas 880 areas are falsely detected as text, obtaining an overall precision of 93.76%. The proposed FCE-TDV approach has successfully localized 13490 of 14657 ground truth text boxes, while only 459 areas are falsely detected as text areas. Consequently, the FCE-TDV approach has attained a slightly better overall recall of 92.04% and definitely a higher precision of 96.71%. This considerable improvement in performance results from the application of the fuzzy clustering ensemble and from the integration of classified text areas from different frames as described in Section 4.4.3.

Except for the video sequence named Seq_6, for which the obtained results using the UTDL-Texture method are considerably lower than those obtained

Figure 4.3: Text detection and localization results obtained using the FCE-TDV method

through the application of the proposed FCE-TDV approach, both methods have obtained comparable recalls in the localization of the text in the other sequences. This difference is mainly due to the presence of a very complex background in this sequence and the failing of the modified UTDL-Texture method to correctly localized the appearing text. The first image in Figures 4.2 and 4.3 is extracted from Seq_6 and illustrates the text localization results using both methods.

Figures 4.2 and 4.3 show some additional results to visually compare the accuracy of the UTDL-Texture and FCE-TDV methods for text detection and localization in videos. The second image illustrates the presence of a false alarm when applying the FCE-TDV approach. The third image shows how the FCE-TDV approach achieves to localize most of the present text in the image despite its low contrast, while the UTDL-Texture method has localized only some of them. The last image illustrates an accurate localization of the text by both methods.

4.6 Summary

In this chapter, a novel method for text detection in videos which extends the UTDL-Texture method was proposed. The modification and the employment of the fuzzy clustering ensemble originally proposed by Qeli [Qel06] has enabled to use the temporal information of the texts present in videos. The FCE-TDV method was evaluated in a total of 10.92 minutes video materials or 16363 frames - 3905 of which contain text. The modified version of the UTDL-Texture method for text detection in videos has achieved an overall recall of 90.24% and an overall precision of 93.76%. On the other hand, the FCE-TDV approach has attained an overall recall of 92.04% and a precision of 96.71%, improving in this way the performance of the text detection/localization in videos.

Chapter 5

Text Tracking in MPEG Videos

Art is the imposing of a pattern on experience,
and our aesthetic enjoyment is recognition of the pattern.
- Alfred North Whitehead -

5.1 Introduction

The text appearing in videos can be static or moving (linearly or randomly) across several frames, providing a considerable amount of redundant information. Since the background (note that all non-text areas are considered as background) may also be static or moving, we can distinguish four possible combinations:

1. Static text and static background;
2. Static text and moving background;
3. Moving text and static background;
4. Moving text and moving background.

The consideration of the temporal occurrence of the text in video is necessary to avoid the processing of redundant information and to provide a means of reducing false alarms by integrating results over multiple frames. To exploit the temporal occurrence of text, a tracking algorithm is required. In the case of (1) or (2), a simple text box matching technique based on the intersection ratio between text and background (e.g. on the I frame level of

an MPEG video) would be adequate to determine the temporal occurrence, since the text position does not change over time. In the case of (3) or (4), the use of the intersection ratio would lead to discarding most of the moving texts, although they might have been accurately localized, since the text position may be displaced significantly. Obviously, if we find a solution for (3) and (4), this solution also works for the other cases. A straightforward solution is to analyze each frame separately, but this is very time-consuming.

In this chapter, a novel algorithm to track moving text within a group of pictures (GOP) is proposed. It operates directly on MPEG compressed data and employs MPEG motion vector information to predict the position of text in the next frame. Comparative experimental results for a set of videos are presented to demonstrate the performance of our approach. Part of the material presented in this chapter has been published in [GEF04c].

This chapter is organized as follows. Section 5.2 provides a brief overview of related work in the field. Section 5.3 presents the tracking algorithm. Section 5.4 describes experimental results obtained for a set of video sequences. Section 5.5 concludes this chapter.

5.2 Previous Work

Although several approaches [SDB98, LW02, LDK00, HYZM03, CAK03] (see also Chapter 3 and 4 for a complete review of these approaches) have been proposed to address the problem of text detection in videos, only some of them [LW02, LDK00, CAK03] assume that superimposed text in a video can move over time. In the following, only the proposed text tracking methods [LW02, LDK00, CAK03] are shortly reviewed.

Li et al. [LDK00] have addressed the problem of detecting and tracking moving text in videos. Tracking of a localized text consists of two steps. First, the speed of the movement of text is predicted as the difference of the same text event in two successive frames. Then, tracking is achieved using a pixel-based matching algorithm that minimizes the least squared error. To deal with growing or shrinking text, a post-processing step is added, which uses an edge detector to refine or correct the contours of the text boxes. The tracking algorithm assumes that text moves linearly and with constant speed. The tracking method fails when the text moves on a complex background, and it is time consuming since it is based on a pixel-based matching module.

Lienhart and Wernicke [LW02] have proposed an approach to track mainly horizontally aligned texts. After the text lines are identified by using a multi-layer feed-forward network (see Section 3.2), a text tracking algorithm is employed on the uncompressed domain to track the localized text box forward and backward. First, they calculate a characteristic signature, which allows the distinction of this text from the others. The signature is a derivative of

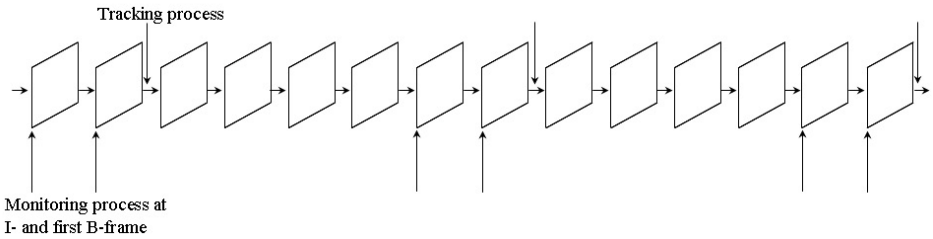


Figure 5.1: The text detection and tracking strategy in a video

the projection profile of the text. Then, they search in the next video frame for a region of the same dimension which matches the reference signature best. The search area is fixed and is defined according to the frame size and the video frame rate.

Crandall et al. [CAK03] have proposed two algorithms for the text tracking problem. The first one deals with rigid text whose font, size or color does not change with time, whereas the second one deals with text changing in font, size or color. The first algorithm makes use of MPEG motion vectors to track text. To refine the tracking results, an edge pixel based search algorithm is employed on a small neighborhood of the tracked text box. The second algorithm detects text in each frame and compares the detected text boxes with each other to determine if they belong to the same text or not. An edge pixel based matching algorithm is used.

The tracking algorithms proposed by Li et al. [LDK00] and by Lienhart and Wernicke [LW02] are both based on a matching technique. Li et al. [LDK00] have employed a pixel-based matching technique, while Lienhart and Wernicke [LW02] have used a signature-based technique to track text. The main drawbacks of the two methods are: (I) it is assumed that the same text moves with a constant velocity; (II) they both are very time consuming. Crandall et al. [CAK03] have proposed a more general tracking algorithm which uses MPEG motion vectors and an edge pixel-based matching method.

5.3 Text Tracking in Videos

The combination of text detection and tracking algorithms increases the system performance in terms of processing time. Text tracking in digital videos can be regarded as a multi-target tracking problem, since more than a text box can appear on a frame, and theoretically they can move in different directions. In order to obtain the new text entries in different frames at different time points, a text detection and localization module must be run in

Algorithm 12: The text detection/localization and tracking approach for video data

Input: The video which will be analyzed

Output: The set of the found text boxes

```
1  k = 0;
2  Text detection and localization in frame_k;
3  while (k < (numberOfFrames − 1)) do
4      if ((frame_k != P) and (frame_k != B)) then
5          tBox_k[]=Text detection in frame_k;
6          if (k > 0) then
7              Text box tracking verification;
8          end
9      else
10         Text tracking: for each tBox_k[i] ∈ frame_k predict the new
           position tBox_{k+1}[i] in frame_{k+1} using Algorithm 13;
11     end
12     k = k + 1;
13 end
```

conjunction with tracking.

In Figure 5.1 the text detection and tracking strategy is illustrated. Since text tracking is done within a GOP and it is based on the motion vectors extracted directly from the MPEG compressed domain, the text detection and localization process is employed in each I frame and its subsequent B frame.

Given a video, the tracking algorithm is combined with the text detection/localization method as shown in Algorithm 12.

5.3.1 Motion Vector Extraction

If we have a sequence of frames with types IBBP, the motion vectors of frame P point to reference blocks in frame I. The original forward motion vectors of frame P describe a movement that has occurred within three frames (IBB). Since we are interested in the movement between two successive frames, the forward motion vectors (see formula 5.1) are normalized to a frame distance of 1. Thus, the normalized forward motion vectors are an estimation of the

Figure 5.2: The visualization of the MPEG motion vectors

motion between the current and the previous frame. The normalization is done as follows:

$$frameDist = \|currentFrame - referenceFrame\| \tag{5.1a}$$

$$normalizedVectorX = \frac{originalVectorX}{frameDist} \tag{5.1b}$$

$$normalizedVectorY = \frac{originalVectorY}{frameDist} \tag{5.1c}$$

where $currentFrame$ is the number of the current frame, $referenceFrame$ is the number of the frame where the motion vectors point to, and $originalVectorX$ and $originalVectorY$ are the X and Y values of the motion vectors extracted from the current frame (in full pixel resolution). Applying motion vectors for text tracking is difficult, due to noise factors and the problem of identifying which motion vectors probably describe the text motion and which the background motion.

A visual representation of the motion vectors present in a specific frame is shown in Figure 5.2. In this video frame, the text scrolls vertically and the background rotates. The grid illustrates the macroblocks. Macroblocks with only one point have a motion vector with zero length, while empty macro blocks represent intra-coded blocks where encoding is independent of a reference frame and motion. For the macroblocks with a motion vector having a length different from zero, a vector is drawn from the center of the macroblock to represent the motion.

Experiments have shown that text blocks sometimes are encoded using only intra-blocks. Moreover, the motion vectors of a text block can also have different lengths and directions. These are some additional problems that a text tracking algorithm must deal with.

5.3.2 The Tracking Algorithm

The tracking algorithm deals with all four possible combinations of a moving or static text in a moving or static background. Motion vectors extracted directly from the MPEG-compressed video are used to predict the future text position with only little computation time.

5.3.2.1 Motion Vector Based Intra-GOP Tracking

Given the text box, $tBox_k[i]$, in the frame $frame_k$, the text tracking method in step 7 of algorithm 12 proceeds as follows. First, the equivalent coordinates of the text box $tBox_k[i]$ in the compressed domain are calculated. The best fitting rectangle with macroblock coordinates, $mbBox[(x'_0, y'_0), (x'_1, y'_1)]$, of a text box $tBox[(x_0, y_0), (x_1, y_1)]$ is calculated using Equation 5.2:

$$x'_i = Int(\frac{x_i + 8}{16}) \tag{5.2a}$$

$$y'_i = Int(\frac{y_i + 8}{16}) \tag{5.2b}$$

The normalized motion vectors (see Formula 5.1) are extracted for each macroblock whose intersection ratio with the text box is larger than a predefined threshold (e.g. 30%). This criterion is chosen to possibly filter out the motion vectors that probably describe the motion of the background instead of text motion. Then, the mode of the motion vector set is chosen to be the motion vector which predicts the text box position in the next frame. This is justified by the assumption that the mode describes most probably the possible motion of the entire text box. Choosing the mode (the object with the maximum frequency) instead of the mean, like in [CAK03], helps to further eliminate some noisy motion vectors without incorporating a filtering process. In case of a text box which is totally encoded with intra-blocks, the previous motion of the text box is used to predict the new position. To predict the new position of the text, the inverted mode motion vector is used. This is done because the motion vector describes the displacement in the preceding reference frame.

Algorithm 13 shows the pseudo code of the intra-GOP tracking method.

5.3.2.2 Intersection Based Inter-GOP Tracking and Verification

At every I-frame, a verification process takes place, in order to prevent possible errors introduced from the motion vector based tracking algorithm (see Figure 5.3). First, the UTDL-Texture algorithm is applied at every I-frame after decompressing it. Then, the best intersection is found between

Algorithm 13: The motion vector based intra-GOP tracking algorithm

Input: The text box, $tBox_k[i]$ in the $frame_k$, which will be tracked

Output: The coordinates of the text box: $tBox_k[i]$ through the successive frames

1 Find text macroblocks in $frame_k$ which intersect more than a predefined ratio with the given text box;

2 Extract forward motion vectors from these text macroblocks;

3 Find the mode MV (the element with the maximum frequency) in the motion vector set;

4 Calculate the new position of the given text box, $tBox_{k+1}$: $tBox_{k+1} = tBox_k - MV$, where MV is the mode motion vector;

the tracked text boxes ($tBox'_{k+i}[]$) and the text boxes determined using the UTDL-Texture algorithm ($tBox_{k+i}[]$). This process involves finding two text boxes whose intersection calculated using Equation 5.3 is the highest.

$$\text{inD}(tBox_{k+i}[], tBox'_{k+i}[]) = \frac{2 * \text{area}(tBox_{k+i}[] \bigcap tBox'_{k+i}[]}{\text{area}(tBox_{k+i}[]) + \text{area}(tBox'_{k+i}[])} \tag{5.3}$$

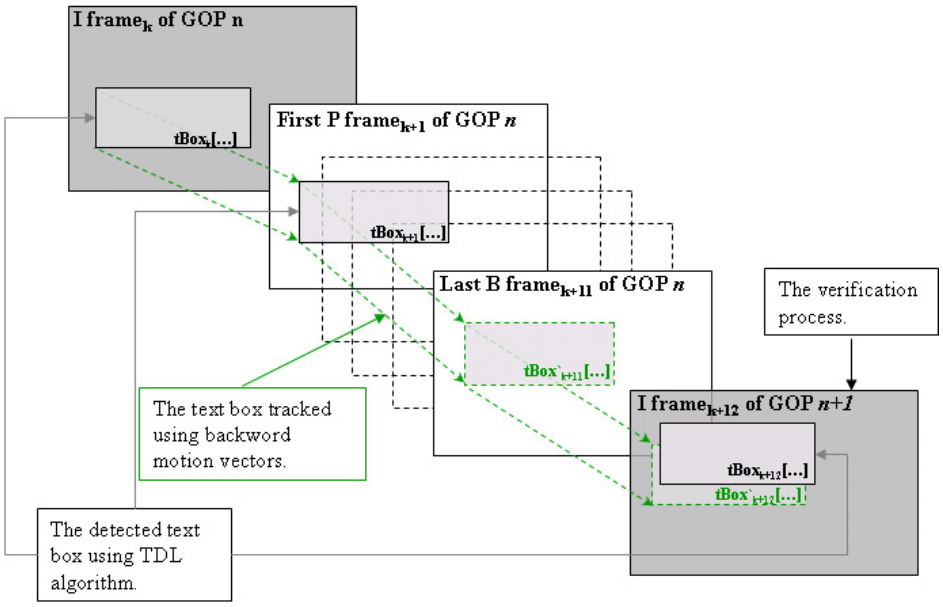


Figure 5.3: Intersection based tracking and verification on the I-frame level

Table 5.1: The details of videos belonging to $TestSet_{Moving}$ and $TestSet_{News}$

Test set $TestSet_{Moving}$			Test set $TestSet_{News}$		
Video name	# frames	# GT	Video name	# frames	# GT
seq-2	310	2248	news-1	380	307
seq-6	207	1191	news-2	459	1187
seq-4	144	803			
seq-5	145	649			
seq-StMb	168	772			
seq-MtMb	100	905			
Total	**1074**	**6568**	**Total**	**839**	**1494**

In Equation 5.3, the function area($tBox_{k+i}[]$) evaluates the area of the text box $tBox_{k+i}[]$. At the end, the intra-GOP tracked text boxes are rectified using the intersection results.

Furthermore, a "text comparison" procedure can be applied to verify if the content of the respective text boxes (e.g. $tBox_k[]$, $tBox_{k+1}[]$, $tBox'_{k+12}[]$ and $tBox_{k+12}[]$ in Figure 5.3) is the same. For this purpose, specific features (e.g. as suggested in [LW02, LDK00]) can be extracted from the respective text boxes and compared with each other. More complex text comparison methods are suggested in Chapter 8.

5.4 Performance Evaluation

The text tracking algorithm has been tested in combination with text detection and localization algorithms.

Two test sets of MPEG videos, namely $TestSet_{Moving}$ and $TestSet_{News}$, have been used. The first test set $TestSet_{Moving}$ includes videos that were kindly provided to us by Lienhart [LW02]. They consist mainly of movie and musical sequences with a lot of text lines scrolling downwards or from right to left. We have also included in our test set two news sequences with static as well as moving text. The second set of videos, $TestSet_{News}$ is included in order to cover a wider range of video types. Video sequences were selected to illustrate all cases of combinations between a moving (static) background and a moving (static) text (see Section 5.1). The used test sets consist of a total of 8 videos with 1913 frames and 8089 text boxes and all video sequences

Table 5.2: The results of the text detection and localization UTDL-Texture method and its combination with the proposed and alternative tracking algorithm

Algorithm	**Recall**	**Precision**
UTDL-Texture + proposed text tracking alg.	88.1%	92.6%
UTDL-Texture + Re-impl. of text tracking alg. [CAK03]	80.0%	84.1%

have a resolution of 320×240 pixels (see Table 5.1).

The performance of the system is evaluated on the text box level and on the frame level in terms of recall and precision. Recall is defined as the number of correctly detected and tracked text boxes divided by the number of all text boxes, whereas precision is defined as the number of correctly detected and tracked text boxes divided by the number of all detected and tracked text boxes, including false alarms. A text is considered (disregarded) when it completely enters (has left) a frame. The number of correctly detected and tracked text boxes as well as the number of falsely detected and tracked text boxes are counted manually. A text area is considered as detected (tracked) correctly, if the text is completely surrounded by a localization (tracking) box, while a detected (tracked) text box is considered as a false alarm, if no text appears in that box.

Furthermore, the motion vector based tracking algorithm presented by Crandall et al. [CAK03] was re-implemented and combined with the proposed text detection and localization algorithm (UTDL-Texture introduced in Chapter 3). The combination was tested on the same test set in order to obtain an objective evaluation of the proposed text tracking algorithm. The results of the experiments are presented in Table 5.2, where recall and precision are reported for the UTDL-Texture method in combination with the two text tracking algorithms. The combination of the UTDL-Texture with the proposed text tracking algorithm achieved a recall of 88.1% and a precision of 92.6%, while the combination of UTDL-Texture with the reimplementation of the text tracking algorithm [CAK03] achieved a recall of 80% and a precision of 84.1%. The proposed motion vector-based text tracking algorithm outperforms the re-implemented algorithm of [CAK03] due to a more robust tracking of static text over moving background.

Figures 5.4, 5.5, 5.6 and 5.7 show some examples of our text detection, localization and tracking approach for several frames extracted from four dif-

ferent video sequences. Images shown in Figures 5.4 and 5.6 are extracted from $TestSet_{Moving}$ and illustrate the result in case of a text scrolling vertically from bottom to top over a moving i.e. static and complex background. The images shown in Figures 5.6 and 5.7 are extracted from $TestSet_{News}$ and demonstrate the case of a video where both static and moving (scrolling from right to left) text appears.

5.5 Summary

The tracking of text within a group of pictures (GOP) using MPEG motion vector information extracted directly from the compressed video stream was the focus of this chapter. Normalized forward vectors of the text region are analyzed in the current frame to predict the text position in the next frame. Experimental results for a set of videos were presented to demonstrate the performance of our approach.

The proposed text tracking solution is significantly different from those proposed by Lienhart and Wernicke [LW02] and Li et al. [LDK00], since none of them makes use of MPEG motion vectors. It is also different from the one suggested by Crandall et al. [CAK03], since in [CAK03] the text position is estimated analyzing all macroblocks in the next frame to compute which forward motion vectors point to the text region in the previous reference frame. Furthermore, text tracking in B-frames is not considered in [CAK03]. In the proposed algorithm, the normalized forward vectors of the text region in the current frame are analyzed to predict the text position in the next frame. Additional advantages compared to the algorithm in [CAK03] are:

- Fewer macroblocks must be analyzed since only the macroblocks intersecting a text bounding box must be analyzed, while in [CAK03] every macroblock in a frame is analyzed;
- The choice of the mode MV avoids the compute-intensive clustering process to find the reliable motion vectors;
- The motion vectors considered belong to text regions and most probably describe motion of text and not of background.

(a) frame 9 (b) frame 13 (c) frame 37

Figure 5.4: The results after applying the text tracking algorithm to each of the localized texts for the video named "abs2"

(a) frame 38 (b) frame 38

(c) frame 44 (d) frame 50

Figure 5.5: The results after applying the text tracking algorithm to each of the localized texts for the video named "abs4"

(a) frame 1021 (b) frame 1028 (c) frame 1091

Figure 5.6: The results after applying the text tracking algorithm to each of the localized texts for the video named "news2"

(a) frame 5047 (b) frame 5058

(c) 5086 (d) frame 5125

Figure 5.7: The results after applying the text tracking algorithm to each of the localized texts for the video named "news2"

Chapter 6

Text Segmentation and Binarization

The whole is more than the sum of its parts.

- Euclid -

6.1 Introduction

When a human observer views a scene, the visual system processing that takes place essentially segments the scene in its composing parts. This process is so efficient, that a person in reality does not see a complex scene, but rather the collection of objects that the scene contains.

With digital systems, holistic image analysis requires information to be extracted once the appropriate pre-processing steps of various filters are completed. In order to accomplish this, a digital image must be *segmented*, or partitioned into its constituent parts of objects. The segmentation process can also be thought of as the process of separating the image into disjoint, or non-overlapping regions, where a traceable path between any two points within the same region must be possible, in order to connect them without leaving the region [Eff00].

In this context, text appearing in images/videos can be considered as an object whose analysis is of utmost importance to understand the content of the media (images/videos). Often, text is superimposed over a complex background and its successful recognition by a commercial optical character recognition engine is very difficult, although the text may be correctly localized. Thus, text segmentation methods which include the separation of the text from the background and the binarization of the text image are crucial and usually they represent the last steps before proceeding further with an

OCR.

In this chapter, the problem of text segmentation is addressed and two novel unsupervised learning methods [GESF04, GF05a] for text segmentation in images with complex backgrounds are presented. We have used a commercial standard OCR software system to investigate the impact of our segmentation approach to recognition performance. The very good performance of our approaches is demonstrated by presenting experimental results for a set of images containing 441 words or 2684 Latin characters. Part of the material presented in this chapter has been published in [GESF04, GEF04a, GF05a].

The chapter is organized as follows. Section 6.2 gives an overview of related work in the field. Section 6.3 reviews two different color models, namely RGB and CIE Lab color model, and possible metrics to appropriately measure the distance between two given colors. The sections that follow introduce the individual steps of the proposed text segmentation methods in detail. Section 6.5 describes experimental results obtained for two sets of images and compares them to other text segmentation methods. Section 6.6 summarizes the chapter.

6.2 Previous Work

Based on how the problem of separating text pixels from background pixels is tackled, we may divide text segmentation approaches into two main classes:

1. Threshold-based approaches [AD99, LW02, WMR99, LKD99, WJC02]

2. Machine learning approaches [GESF04, GF05a, YGH04, OC02, CO03]

The first group of methods [AD99, LW02, WMR99, LKD99, WJC02] tries to find the optimal threshold to binarize the text image. They are simple to implement, but they are not very robust when there are pixels in the background whose color is similar to the text pixels. The second class of methods applies statistical techniques to separate the text from the background pixels. They use unsupervised [GF05a, GESF04, YGH04] or supervised [OC02, CO03] learning algorithms to separate the text pixels from the background pixels.

In the following, an overview of text segmentation approaches proposed in the last years is given.

Agnihotri and Dimitrova [AD99] use the average of pixel values of the text image as the optimal global threshold for the binarization step. The authors assume that the average of pixel contours of the text box is closer to the average of the pixels marked as background on the text image. The problem of text embedded in complex background is not addressed, and performance results for segmentation or recognition are not reported.

Antani et al. [ACK00] have presented a simple text segmentation method. To agree with the different polarity of text in an image, two segmented text regions are generated. A connected-component method is applied to the segmented result to remove the components that do not fulfill the specified aspect ratios. Finally, a score is assigned to each of the segmented images based on their text-like characteristics. The image with the highest score is selected as the input for an OCR system. No performance results for segmentation or recognition are presented.

Ngo and Chan [NC05] first compute a 16-bin normalized histogram of the gray level values. Then, by scanning the bins backward, the k^{th} bin that corresponds to the first valley in front of the first peak is located. Finally, a threshold with a value of $16 * (k - 1)$ is used to binarize the text image.

Wu et al. [WMR99] use a low pass Gaussian filter to first smooth the image and then compute an intensity histogram. Then, the histogram is also smoothed and the first peak from the left on the smoothed histogram is used as a threshold for the binarization process. The algorithm was tested on a set of photographs, newspapers, advertisements, and personal checks (with 300 dpi). A character recognition rate of 84% is reported.

After enhancing the image using Shannon up-sampling, Li et al. [LKD99] apply a local thresholding method to binarize the enhanced image. A block is marked as background only if its standard deviation is smaller than a fixed threshold. The recognition rate achieved for the used test set (images with low resolution) is 67.8%.

Wolf et al. [WJC02] propose an adaptive local based thresholding method for multimedia documents. The detected text boxes in multiple consecutive frames are used to create a high resolution text box using bilinear interpolation. A new local threshold is calculated for each block separately, combining the thresholds evaluated using the Niblack [Nib86] and Sauvola [SSP97] algorithms. The method is tested on 60000 frames of different MPEG videos. The authors report to have achieved a character recognition rate of 85%.

In an approach proposed by Lienhart and Wernicke [LW02], the possible text and background color is estimated first. A 4(8)-neighborhood seed filling algorithm is applied to each text (background) pixel separately. Components that do not fulfill certain geometrical restrictions are removed. A binarization process follows where the global threshold is calculated as the mean of the text and background color. The threshold procedure is applied differently for inverted and normal text. A recognition performance of 69.9% is given. The video test set used contains credits, commercial and news sequences.

Loprestie and Zhou [LZ00] have integrated the process of text segmentation into the recognition process. Two different methods are proposed for this purpose. The first one uses a polynomial surface fitting algorithm to recognize the characters. The second method is based on a fuzzy n-tuple classifier. Their methods are tested on a set of web pages. For the first method, a

recognition rate of 69.7%, and for the second method, a recognition rate of 89.3 % is reported. The two methods are trained with half of the test set.

Sato et al. [SKH+99] employ a sub-pixel interpolation method and a multi-frame integration schema to enhance the text image. Furthermore, the strokes of the characters are again enhanced by applying filters in four different directions (0, 45, 90 and 135) and combining their results. Finally, the image is binarized using a global threshold. The recognition rate using their own OCR is 83.5% for a CNN headline news test set.

Hua et al. [HYZ02] have proposed a multiple frame text extraction schema. The frames where the same text appears in a clearly recognizable manner are averaged with each other to get a so called "man-made" frame. A block-based adaptive thresholding procedure applied to this frame concludes the segmentation process. They have reported a character recognition rate of 78% for a test set of MTV sequences.

Odobez and Chen [OC02] have presented a multi-hypotheses approach based on a Markov random field (MRF) and on grayscale consistency constraints for text segmentation. The gray level distribution in text images is modeled as a mixture of Gaussian distributions. The assignment of each pixel to one of the Gaussian layers is based on prior contextual information, which is modeled by a MRF. Each layer is considered as a binary text image and is fed into the OCR system as one segmentation hypothesis. The text image which gives the best recognition performance is considered as the output of the system. The authors have reported a recognition rate of 93%. In their experiments, frames extracted from sports videos were used.

Ye et al. [YGH04] propose an unsupervised learning method. Samples of text pixels are extracted based on the heuristic that they lie between an edge couple. Then, a Gaussian Mixture Model (GMM) is used to model the intensity of text pixels and is trained with the extracted samples. Finally, text pixels are extracted from the background using the trained GMM and spatial connectivity properties of text.

Perroud et al. [PSBH01] present two histogram based clustering algorithms for text extraction from color documents. The first is based on the RGB space while the second also uses the spatial information.

After estimating the representative colors, Hase et al. [HYT+04] evaluate the color uniformity of character area. Then, an adaptive segmentation method based on the least color difference in the La^*b^* color space is used to extract text characters.

6.3 Color Models and Distance Measurements

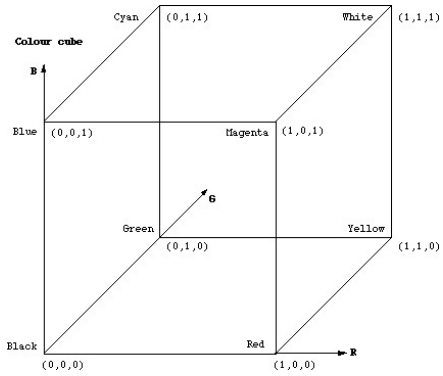
The use of a certain color model during the text segmentation process is motivated by two main factors [GW01]:

- First, color is a powerful descriptor that often simplifies object (i.e. text) extraction from a scene.
- Second, humans can discern thousands of color shades and intensities, compared to about only two dozen shades of gray.

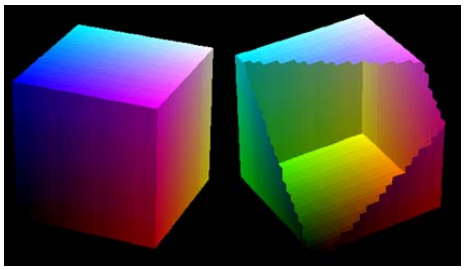
In the following subsections, two color models, namely RGB and CIE color models, will be presented.

6.3.1 RGB Color Model

In the RGB color model, each color appears in its primary spectral components of red, green, and blue and is based on a Cartesian coordinate system. This color model is often visualized by a unit cube where each color (red-R, green-G, blue-B) is assigned to one of the three orthogonal coordinate axes in the 3D space as illustrated in Figure 6.1(a) and in Figure 6.1(b). The values of each of the R, G, and B components move from 0 to 255.



(a) The RGB color cube



(b) The colored presentation of the RGB model

Figure 6.1: Two different visualizations of the RGB color model

6.3.2 CIE Color Model

The main aim of the CIE color model is to provide an uniform color space that facilitates direct measurement of the distance between two different colors. CIE $L^*a^*b^*$ is one of such a color model. The CIE $L^*a^*b^*$ color model was adopted as an international standard in the 1970's. The mapping of an input image into a perceptually uniform color space is normally based on a nonlinear transformation. The components L^*, a^* and b^* are defined from the R, G, B

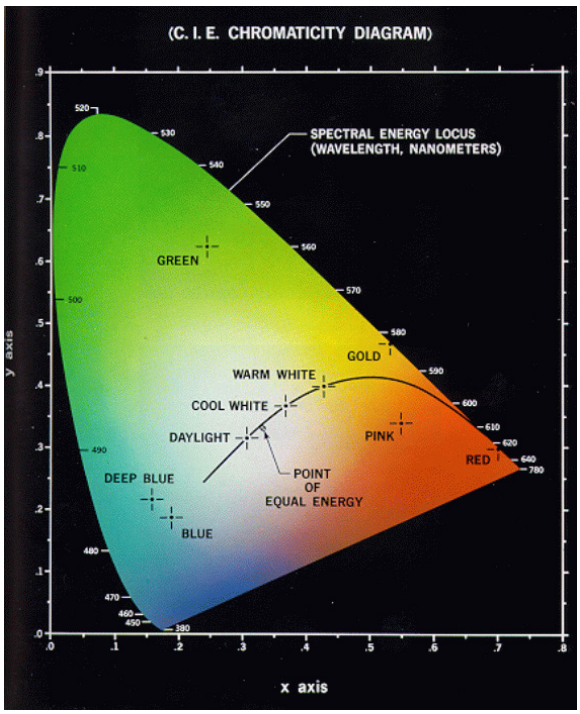


Figure 6.2: The visualization of the CIE color model

color components using the following equations:

$$L^* = \begin{cases} 116(Y/Y_0)^{1/3} - 16 & \text{if } (Y/Y_0) > 0.008856, \\ 903.29(Y/Y_0) & \text{otherwise} \end{cases} \tag{6.1a}$$

$$a^* = 500[(X/X_0)^{1/3} - (Y/Y_0)^{1/3}] \tag{6.1b}$$

$$b^* = 200[(Y/Y_0)^{1/3} - (Z/Z_0)^{1/3}] \tag{6.1c}$$

where the constants X_0, Y_0 and Z_0 are the tristimulus values of the standard white, and the values X, Y and Z are derived from R, G and B values using the equations:

$$X = 0.490R + 0.310G + 0.200B \tag{6.2a}$$

$$Y = 0.177R + 0.812G + 0.011B \tag{6.2b}$$

$$Z = 0.000R + 0.010G + 0.990B \tag{6.2c}$$

In the CIE $L^*a^*b^*$ color model, the component L^* represents the intensity. A combination of the two other components a^* and b^* represents the color. The values for the components L^*, a^* and b^* range from 0 to 100, from -86 to 86 and from -111 to 90, respectively. A visual illustration of the CIE color model is given in Figure 6.2.

6.3.3 Distance Measurements

The selection of an appropriate metric for efficiently and accurately measuring the distance between two colors is one of the problem encountered during the color image segmentation in general and the color text segmentation in particular. During color image segmentation, a certain color distance is usually used to measure the similarity between two color points, and the points that satisfy a certain degree of color similarity are grouped to form homogeneous regions.

As mentioned in the previous subsection, the CIE L*a*b* color space is a perceptually uniform color space, which means the Euclidean distance between two color points in the CIE L*a*b* color space corresponds to the perceptual difference between the two colors by the human vision system [WS00].

The Euclidean distance between two color points in the CIE L*a*b* color space is defined in the following.

Definition 1 (*Euclidean distance between two color points.*)
Given two colors x and y from the CIE L*a*b* color space, which are denoted by $x = (x_L, x_a, x_b)$ and $y = (y_L, y_a, y_b)$, their Euclidean distance $\delta_{Euclidean}(x, y)$ is evaluated using the Equation 6.3.

$$\delta_{Euclidean}(x, y) = \sqrt{(x_L - y_L)^2 + (x_a - y_a)^2 + (x_b - y_b)^2} \tag{6.3}$$

Even though the Euclidean distance between two points in the CIE color model allows to determine the perceptual difference of the colors points, the RGB color model remains an efficient color space because no conversion is required. Although it suffers from the non-uniformity problem where the same distance between two color points within the color space may be perceptually quite different in different parts of the space, the color consistency may be defined using certain color thresholds.

To define an appropriate similarity metric in the RGB color space, an experiment was conducted by Loo and Tan [LT04]. The experiment starts with a pivot color. Then, it randomly generates 620 non-repeating variations of color points with the same distance from the pivot color computed by using the same distance function (i.e. Euclidean or Manhattan). The generated color point is visualized as a 20×20 pixels square (the size of a 12 point character). All colors are then visually inspected by ten human subjects to determine their similarities. Each observer will vote for one of the seven categories shown in Table 6.1. This process is repeated for color distances in the range of 10 to 500 by a step of 10, computed by different distance measurements. The final result is obtained by taking the majority vote.

The authors [LT04] conclude that the Manhattan distance is a better distance measurement when the generated color points exhibit a more stable visual color similarity. In contrast to the Manhattan distance, the Euclidean distance measurement will produce a wider variation of color perception with the same color distance. This observation shows that the distance between two color points in the RGB color space can be better derived by the addition of the variations of the red, green and blue intensities as by the physical Euclidean distance between the two color points. The definition of the Manhattan color distance is given below.

Definition 2 (*Manhattan distance between two color points.*)
Given two colors points x and y from the RGB color space, which are denoted by $x = (x_R, x_G, x_B)$ and $y = (y_R, y_G, y_B)$, their Manhattan color distance $\delta_{Manhattan}(x, y)$ is evaluated using the Equation 6.4:

$$\delta_{Manhattan}(x, y) = (|x_R - y_R| + |x_G - y_G| + |x_B - y_B|) \tag{6.4}$$

where the sign $|.|$ is used to denote the absolute differences.

The various categories of thresholds limits obtained through the experiment in [LT04] are shown in Table 6.1. It can be seen that the similarities between two color points in the RGB color space can be categorized into four main groups. The difference between two same colors with a very low intensity variance usually does not exceed a value of 71. In the second group, the differences range from 71 to 120 and the colors which appear to be from the same color series (e.g. dark/light brown) with a varying degree of intensity are included. Color differences between colors with varying color ranges vary from 121 to 190 values. Differences above a value of 190 were quite random and thus were considered as undefined and cannot be interpreted.

Based on the above mentioned experimental result, the Manhattan distance is slightly modified and a new color similarity metric is derived in [LT04] as defined below:

Definition 3 (*Modified Manhattan distance between two color points.*)
Given two colors points x and y from the RGB color space, which are denoted by $x = (x_R, x_G, x_B)$ and $y = (y_R, y_G, y_B)$, their modified Manhattan distance (from now on it is refered as MManhattan distance) is:

$$r' = |x_R - y_R| \tag{6.5a}$$
$$g' = |x_G - y_G| \tag{6.5b}$$
$$b' = |x_B - y_B| \tag{6.5c}$$
$$\sigma = (|r' - g'| + |r' - b'| + |g' - b'|)/3 \tag{6.5d}$$
$$\delta_{MManhattan}(x, y) = r' + g' + b' + \sigma \tag{6.5e}$$

Table 6.1: Categories of color threshold limits (taken from [LT04])

Threshold	**Visual inspection result**
10 to 30	Same color
31 to 70	Same color, low intensity variance
71 to 90	Same color series
91 to 120	Same color series, low intensity variance
121 to 150	Different color, small color range
151 to 190	Different color, wider color range
above 191	Very randomly occurring color

The modified Manhattan metric computes first the total absolute variation of the respective RGB channels between two color points. In addition, a factor σ is calculated to discriminate between a well distributed color variance among all RGB values and those with un-even variance distribution. The former reflects a color consistency better than the latter.

6.4 Text Segmentation in Complex Images

Text segmentation usually consists of two main steps. First, the image quality is enhanced by applying different filters in the still image domain or through multi-frame integration in the video domain. Second, the text pixels in the enhanced image are separated from the background pixels by means of statistical methods or thresholding algorithms.

The proposed methods are designed to segment text strings of arbitrary font, size and color. No assumption is made about the text color polarity in contrast to most of the existing text segmentation methods, which assume that the text has a specific color polarity (text color is always dark or light), restricting thus their application in segmenting real video texts that have various appearances and complex backgrounds. In this thesis, it is assumed that the text string consists of a homogeneous color and aligns horizontally, which normally is the case for artificial text.

The input of the methods is the original image and the coordinates of the text bounding boxes, which can be generated using one of the algorithms introduced in Chapter 3. The excerpt shown in Figure 6.3 consisting of five text images extracted from the test sets used to evaluate the proposed

Figure 6.3: Example of the input images of the proposed text segmentation algorithm

methods illustrates various complex backgrounds where text is embedded.

Both the proposed methods overcome the difficulties for finding the optimal global or local threshold [LW02, WMR99, LKD99, WJC02] or the request for different training samples in [OC02, CO03, LZ00] by applying unsupervised clustering.

In the following, the enhancement step in the still image domain is first explained, before proceeding further by explaining the feature extraction process and the individual steps of each of the proposed text segmentation techniques.

6.4.1 Resolution Enhancement

Videos typically are digitized using a low resolution of 72 dpi (related to A4 paper format i.e. 99×210 mm), which is not very appropriate for text segmentation and text recognition. This causes that often in MPEG-1 videos there are characters that have a height of less than 11 pixels. Although such text occurrences are still recognizable for humans, they still remain a challenge for standard OCR systems due to anti-aliasing, spatial sampling and compression artifacts. The standard OCR systems have been designed to recognize text in document images, which were scanned at a high resolution of 300 dpi, resulting in a minimal text height of at least 40 pixels.

Furthermore, as will be shown in more details later in Section 6.5, an experiment has been conducted with a set of images. The segmentation

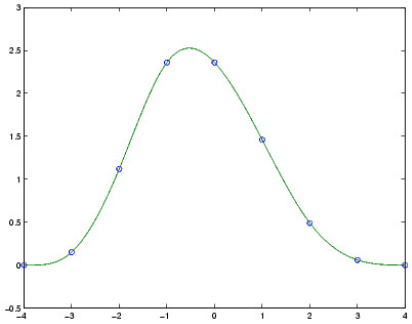


Figure 6.4: One dimensional cubic interpolation function

methods and the OCR steps have been tested with the same test of images, first using the original resolution of 72 dpi and second by enhancing the image resolution to 300 dpi. The results have demonstrated that the segmentation and the subsequent OCR steps perform much better on a higher resolution than on the original video frame resolution of 72 dpi. In addition, both segmentation algorithms perform better, if the text is not too small.

Thus, to obtain good segmentation results and accurate recognized characters/words when using standard OCR systems, it is necessary to first enhance the resolution of the text images.

The enhancement of the text image resolution is a common pre-processing task. The resolution of the text image is enhanced up to 300 dpi using a cubic interpolation, while the aspect ratio is still preserved. The cubic interpolation uses a 16×16 neighborhood of pixels surrounding the calculated point to compute its value. The calculation can be thought of as a convolution of a 16×16 neighborhood with a cubic interpolation function. In Figure 6.4, a one-dimensional function representing the cubic interpolation is plotted. The interpolation function acts as a low pass filter. The cubic function has a continuous derivative in contrast to a linear function which shows sudden changes in slope. A cubic function has also negative lobes. Thus, a value interpolated using a cubic function will be a weighted sum of nearby pixel values, minus some contribution from pixels slightly farther away. This property of the cubic function reduces the low pass filtering effect and produces sharper results compared to those using linear functions.

6.4.2 Feature Extraction and Normalization

Several color and texture features are considered to find the best ones to classify pixels as text or background.

The basic color features consist of the pixel color components. Generally, raw color data are expressed in the RGB color space (Section 6.3.1). However,

the RGB color space and its linear derivatives do not constitute uniform color spaces. In contrast, the CIE L*a*b* color space (Section 6.3.2) is perceptually an uniform color space, which means that the Euclidean distance between two color points in the CIE L*a*b* color space corresponds to the perceptual difference between the two colors by the human vision system (see Section 6.3). Therefore, color features are considered in both the RGB and CIE L*a*b* color space.

Furthermore, a small sliding window of dimension $m \times n$ (e.g. 3×3 pixels) is moved over the text box to consider local image properties. This technique is motivated by two observations:

- Characters usually have a unique texture;
- The border of superimposed characters results in high-contrast edges.

Consequently, the wavelet transform is applied to the image to consider these properties and pass them to the subsequent clustering algorithm. The standard deviation of wavelet coefficients in the sliding window is expected to be low within a character's texture, but high at its boundary. Thus, the character boundaries are enhanced in the segmented image by using this feature. Since the text-to-background contrast is expected to be high in the gray-scale transformed image but not in each color channel, the wavelet transform is again applied to the gray-scaled version of the image. In this way, to consider the texture of the pixels, the wavelet coefficients in the high frequency sub-bands (e.g. LH and HL) and their standard deviations are also included in the set of possible features. The standard deviation of wavelet coefficients in a sliding window of $m \times n$ dimension, centered at position (x, y) is defined as follows:

$$\mathrm{stDev}_{\mathrm{window}}(x,y) = \sqrt{\frac{1}{mn}\sum_{i=-\frac{m}{2}}^{\frac{m}{2}}\sum_{j=-\frac{n}{2}}^{\frac{n}{2}}(W(x+i,y+j)-\mathrm{mean}_{\mathrm{window}}(x,y))^2} \tag{6.6}$$

where $W(x+i, y+j)$ is the value of the wavelet coefficient at position $(x+i, y+j)$ and $\mathrm{mean}_{window}(x, y)$ is evaluated using Equation 6.7.

$$\mathrm{mean}_{window}(x,y) = \frac{1}{mn}\sum_{i=-\frac{m}{2}}^{\frac{m}{2}}\sum_{j=-\frac{n}{2}}^{\frac{n}{2}} W(x+i,y+j) \tag{6.7}$$

Finally, all extracted feature components are normalized to the range [0, 1].

6.4.3 Unsupervised Text Segmentation Method

The flow chart of the proposed unsupervised text segmentation (UTS) method is shown in Figure 6.5. The UTS method can be divided into four main steps:

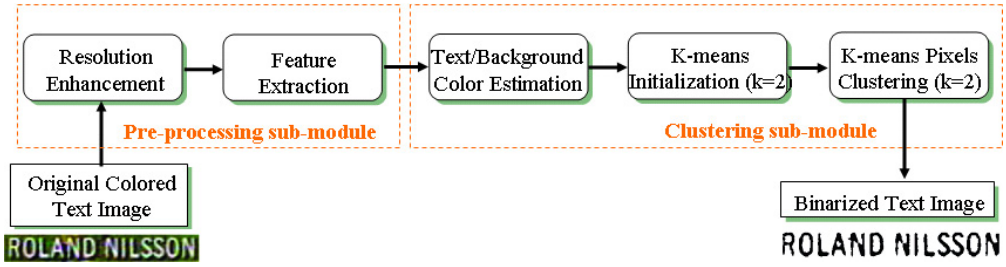


Figure 6.5: The flow chart of the unsupervised text segmentation method

1. Resolution enhancement;
2. Feature extraction and normalization;
3. Text/background color estimation;
4. Unsupervised pixel classification.

The steps 1 and 2 were already introduced in Sections 6.4.1 and 6.4.2, respectively, whereas the steps 3 and 4 will be explained in detail in the following paragraphs.

6.4.3.1 Text/Background Color Estimation

In this step, the dominating text and background colors are estimated for each text box, following an approach suggested by Lienhart and Wernicke [LW02]. This algorithm consists of two substeps. First, the colors of the image are reduced using a color quantization process. Second, the possible text and background colors are estimated.

Substep 1: *Image Color Quantization.* Color quantization can be abstracted as a 3D clustering process. A color image in the RGB color space corresponds to a set of points ($c = (c_R, c_G, c_B)$) in a three-dimensional discrete space. The intensity of each color often is discrete and encoded by m bits; hence the image can be further abstracted as a set S of $2^m \times 2^m \times 2^m$ points. The quantization of the color image into K colors consists of partitioning the set S into K subsets $S_k, 1 \leq k \leq K$, $\sum_{c \in S_k} P(c) \neq 0$, where $P(c)$ is the frequency of the color c, $\cup_{1 \leq k \leq K} S_k = S$, $S_j \cap_{j \neq k} S_k = 0$, and all colors $c \in S_k$ are represented by an unique color $q_k = (q_{kR}, q_{kG}, q_{kB})$. Based on this formulation of the quantization problem, the expected quantization error is defined as

$$E(S_1, S_2, ..., S_K) = \sum_{1 \leq k \leq K} \sum_{c \in S_k} P(c)\psi(c - q_k) \tag{6.8}$$

(a) The original image (b) The image quantized to 8 colors

Figure 6.6: Color image quantization using the method proposed in [Wu96]

(a) The original text image (b) The quantized text image $nr_{colors} = 8$

Figure 6.7: Color text image quantization using the method proposed in [Wu96]

where ψ is a dissimilarity measure between two colors c and q_k. The Euclidean color distance similar to Equation 6.3 is used as dissimilarity measure. The aim is now the minimization of the quantization error E in Equation 6.8, for given $P(c), \psi$, and K. The fact that there are approximately $K^{|S|}$ different partitions, makes optimal color quantization a challenging problem.

The color quantization algorithm proposed by Wu [Wu96] is optimized through a linear search and it basically functions as follows. A plane is moved across the RGB cube perpendicular to each of the R, G, B axes separately, and the plane which minimizes the sum of variances at both sides is chosen to divide the cube into two sub-cubes. Then, the sub-cube with the highest variance is divided again using the same procedure. This process continues until the K sub-planes are created. Finally, the mean of the colors belonging to a sub-cube is selected as the representative color for this sub-cube.

In Figure 6.6, the color quantization process for a given color image using the described method is illustrated. In this example, the number of colors in the image is reduced to eight colors.

In our approach, the number of colors of the localized text is reduced to a fixed number nr_{colors} using the color quantization method proposed by Wu [Wu96] and described above. In Figure 6.7, the result of the quantization process for a text image is presented.

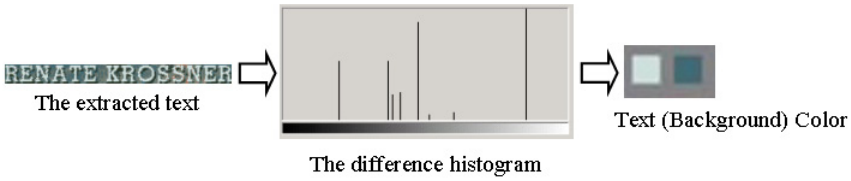


Figure 6.8: The difference histogram and the estimation of the text and background color as the maximum and the minimum of the histogram respectively

Substep 2: *Text and Background Color Estimation.* Using the quantized text image, two color histograms are calculated: the histogram of a given number ($nr_{textRows}$) of center rows in the text box and the histogram of a given number ($nr_{bckgRows}$) of rows directly above and below of the text box.

Finally, the difference histogram of those two histograms is calculated. The text and the background color are defined as the maximum and the minimum of this difference histogram, as shown in Figure 6.8.

6.4.3.2 Unsupervised Pixel Classification

The k-means clustering algorithm (see Section 2.9.2.1) is used to classify the image pixels into k clusters. We have two clearly distinguishable classes of pixels: "text" and "background" ($k = 2$).

Usually, the initial solution of the k-means algorithm is chosen randomly, but in our case, a new initialization method is introduced. The "text" cluster is initialized with the feature vector f_{text} that has the minimum distance (e.g. applying Euclidean metric) to the ideal feature vector

$$f_{tIdeal} = (R_{textColor}, G_{textColor}, B_{textColor}, 1, ...) \tag{6.9}$$

representing the "text" cluster, whereas the "background" cluster is initialized in a similar way but with the feature vector $f_{background}$ that has the maximum distance to

$$f_{bIdeal} = (R_{bckgColor}, G_{bckgColor}, B_{bckgColor}, 0, ...) \tag{6.10}$$

For the "text" ("background") cluster, the ideal feature vector includes the predefined "text" ("background") color and the normalized wavelet coefficients (respectively their standard variation in the sliding window), which are 1 (0) in the ideal (non-ideal) case. Then, the k-means algorithm is applied to obtain clusters whose members have the minimum Euclidean distance to the respective cluster mean feature vector.

Finally, a binarized text image is generated where the pixels of the cluster "text" are painted in black, while the other pixels are painted in white.

RENATE KRÖSSNER

Figure 6.9: The unsupervised text segmentation result

Figure 6.9 illustrates the text segmentation result for the colored text image given in Figure 6.7.

6.4.4 Adaptive Fuzzy Text Segmentation Method

The UTS method faces difficulties when the distribution of the colors in a text image is not bimodal, which means that the partition of pixels into two clusters does not bring satisfactory results ($k = 2$ is not effective enough). To deal with this problem, a novel method is proposed in this section, where the number of clusters depends on the modality of the color distribution. In addition, due to the fact that color groups do not have hard boundaries and similarity measures between color data are relative, a fuzzy similarity measure is thought to be more appropriate than a hard one. Thus, fuzzy C-means is chosen to cluster the pixels of the text image. However, there are two major difficulties in applying the fuzzy clustering algorithm introduced in Section 2.9.2.2:

- The determination of the number of clusters C;
- The initial fuzzy partition of objects into clusters (cluster centroids).

Solutions are proposed for both of these problems in the following paragraphs.

The flow chart of the proposed adaptive fuzzy text segmentation (AFTS) method is shown in Figure 6.10. The AFTS method can be divided into seven main steps:

1. Resolution enhancement;
2. Feature extraction and normalization;
3. Determination of the initial number of clusters;
4. Determination of the initial clusters centroids;
5. Adaptive fuzzy clustering (AFC);
6. Binarization, and text identification;
7. Text image enhancement.

The first and the second steps have already been introduced in Sections 6.4.1 and 6.4.2, respectively. The other steps will be explained in details in the following paragraphs.

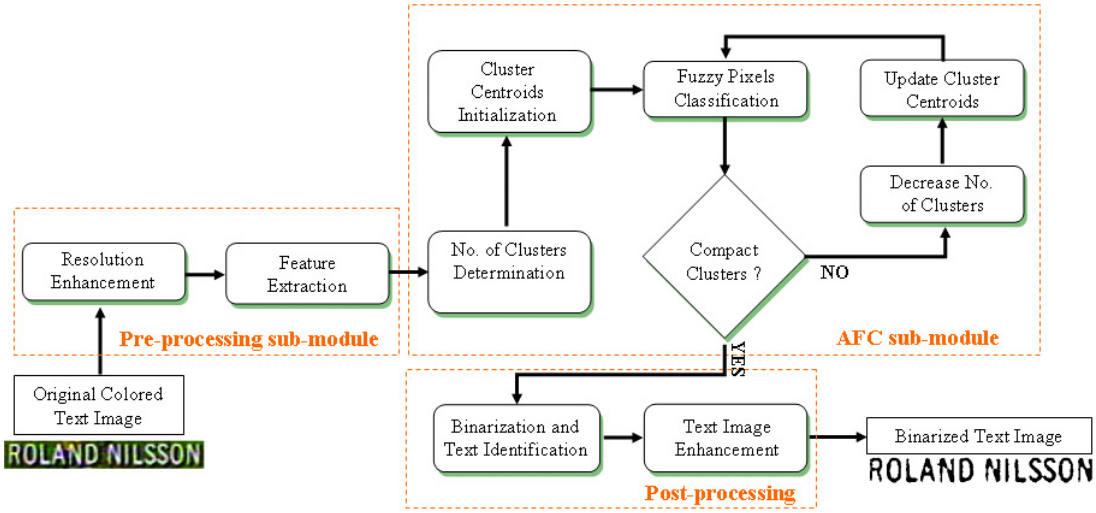


Figure 6.10: The flow chart of the adaptive fuzzy text segmentation method

6.4.4.1 Determination of the Initial Number of Clusters

In contrast to the UTS method (Section 6.4.3) where the number of clusters is fixed to two ("text" and "background" clusters), in this approach the number of clusters is defined at run time and is adaptive depending on the image that will be segmented. This solution is motivated by the fact that the background under the text is often multicolored and can show different texture properties from image to image. The number of clusters is defined using a heuristic approach which consists of the following steps:

- **Step 1.** A certain number of significant pixels from the input image is selected (e.g. the pixels that lie on the middle row).
- **Step 2.** The selected pixels are divided into sets of pixels with similar colors. An appropriate color distance (in the selected color space) is used to measure the similarity between two different colors. If the difference between the colors of two pixels is below a threshold ($th_{similar}$), then the pixels are considered to be similar. The output of this stage are groups of pixels which have similar colors.

It is assumed that the number of generated groups of colors is an indirect indicator of the possible different colors in the image. Thus, this number is used as the initial number of clusters C for the fuzzy C-means algorithm.

6.4.4.2 Initialization of the Cluster Centroids

There is no generally accepted approach for the initialization of the fuzzy clustering algorithm. The initialization step is important because different

selections of the initial cluster centroids can potentially lead to different partitions, which in our case will be translated as different segmentation results. The proposed initialization method uses the groups of colors generated in the previous step. It works as follows:

- **Step 1.** A representative color is calculated for each group as the mean color of all colors that belong to the group. It is assumed that the estimated colors c_j for $j = 1..C$ compose the representative colors of the input image and are used as centers for the initial fuzzy partitioning during the next step.

- **Step 2.** The membership function of color u_{ij} between a color x_i and a center color c_j is defined as originally proposed by Kim et al. [KLL04] using Formula 6.11:

$$u_{ij} = \begin{cases} 1.0 & \text{if } \delta(x_i, c_j) = 0 \ , \\ 0.0 & \exists\ k \neq j \text{ fulfilling } \delta(x_i, c_k) = 0, \\ (\sum_{k=1}^{C} (\frac{\delta(x_i,c_j)}{\delta(x_i,c_k)})^{\lambda})^{-1} & \text{otherwise} \end{cases} \quad (6.11)$$

  where λ is a weighting parameter for the membership of x_i to c_j usually with the same value as the fuzziness parameter of the FCM algorithm m ($m = \lambda = 2$) and $\delta(x_i, c_j)$ is the Euclidean distance between the colors x_i and c_j.

6.4.4.3 Adaptive Fuzzy Clustering (AFC)

The FCM algorithm is applied to cluster the pixels of the text image into C clusters, where C and the initial membership matrix u are determined as in the previous paragraphs. The Euclidean distance is used to measure the similarity between the features that represent each pixel. After the clustering process has converged, the algorithm continues as follows:

- **Step 1.** The fuzzy membership is converted into a hard membership based on the maximum criterion, i.e. each pixel is classified to belong to the cluster with a maximum membership value.

- **Step 2.** The mean ($mean_j$) of the degrees of membership u_{ij} of the objects x_i in each $cluster_j$ is calculated using Equation 6.12:

$$mean_j = \frac{\sum_i u_{ij}}{|cluster_j|}, \forall i \mid x_i \in cluster_j \quad (6.12)$$

  As the membership degree u_{ij} provides a measure of the similarity between the point x_i and the center of the $cluster_j$, the value of $mean_j$ will offer an approximate information how compact the $cluster_j$ is.

(a) The input image

(b) The segmented image without the AFC; six clusters remain at the end, $C = 6$

(c) The segmented image with AFC; four clusters remain at the end, $C = 4$

Figure 6.11: The impact of AFC on the final result of the text segmentation

- **Step 3.** If the condition shown in Equation 6.13:

$$mean_j >= th_{compact}, \forall cluster_j \tag{6.13}$$

  is not fulfilled, then the number of clusters is decremented: $C = C - 1$. The two clusters with the lowest value of $mean_j$ are melted into one and a new center is calculated as the mean of the old centers. However, other criteria could be also considered, e.g. merging the cluster with the lowest value of $mean_j$ with its closest cluster in the color space. After the initialization with the new centers is done, the pixels are clustered again. The clustering process is repeated until the aforementioned condition is fulfilled or $C = 2$. In this way, possible oversegmentation of the text characters is avoided.

In Figure 6.11, an example is shown how the segmentation result is improved when applying the presented AFC algorithm instead of a simple FCM.

6.4.4.4 Binarization and Text Identification

After applying the steps explained above, C clusters of pixels are created. A binary image $bImage_j$, with the same dimension as the original text image is generated for each $cluster_j$ marking all pixels which belong to the $cluster_j$ with black and the rest with white.

To proceed further, the correct cluster, i.e. the cluster containing text pixels should be found. For this purpose, a rating algorithm is proposed (see Algorithm 14). It makes use of the fact that connected components and their bounding rectangles (see Figure 6.12 for sample illustrations) in the binary

(a) The pixels classified in the first cluster

(b) The pixels classified in the second cluster

(c) The pixels classified in the third cluster

Figure 6.12: The visualized bounding rectangles of the extracted connected components for each of the candidate text images ($C = 3$) for the input image shown in Figure 6.3(b)

text image will show similar spatial and geometric properties compared to those of the other candidate images. The rationale behind this is that the characters (i.e. components) in a text often are aligned horizontally and lie on a straight line. Thus, it is expected that the y-coordinates of the lower right corners of the bounding rectangles will have very similar values when estimated in the correct binary text image. Consequently, their standard deviation ($stDevYCC_j$) will be close to zero. Moreover, it was observed that the bounding rectangles of the components in the text image show similar heights and width/height ratios, too.

After extracting all connected components and evaluating the respective bounding rectangles in each $bImage_j$, different features are estimated based on the above-mentioned observations. The density of blanks in the vertical projection ($densityBlank_j$) and the mean of the degrees of membership ($mean_j$) for each $bImage_j$ is also taken into consideration. Then, the rating algorithm functions as follows. First, all candidate images that do not fulfill the minimal criteria e.g. $((densityBlank_j > 0)$ and $(densityBlank_j < 0.7))$ are excluded from further evaluations. Afterwards, an evaluation process consisting of several steps takes place, where each of the remaining candidates is rated based on the values of each of the estimated features. The maximum value for a feature rating is $max = C - 1$, where C denotes the number of candidate images. Within an evaluation step, the candidate with the best feature value will be given the maximum feature rating, whereas the next best candidate will be given a feature rating of $max - 1$, and so on.

Algorithm 14: The pseudocode of the text cluster identification algorithm

Input: The clusters created by the AFC

Output: The binarized text image: $textImg_{M \times N}$

For each $cluster_j$ create a binary image: $bImage_j$;
foreach $bImage_j$ **do**
    Extract the connected components: $cc_j[]$;
    Estimate the corresponding bounding rectangles: $bRect_j[]$;
end
for $j = 1$ *to* C **do**
    $stDevHCC_j$ = StandardDeviation(Height($bRect_j[]$));
    $stDevWCC_j$ = StandardDeviation(Width($bRect_j[]$));
    $stDevSizeCC_j$ = StandardDeviation(Size($bRect_j[]$);
    $stDevYCC_j$ = StDevY-coordLowerCorner($bRect_j[]$));
    $mean_j$ = MeanClusterMembership($cluster_j$);
    $densityBlank_j$ = Density(VerticalProjection($bImage_j$));
    ...;
end
for $j = 1$ *to* C **do**
    if *(*$densityBlank_j > 0$*) AND (*$densityBlank_j < 0.7$*)* **then**
        Rate $bImage_j$ based on the evaluated features;
    end
end
$textImg$ = HighestRate($bImage_j$);

For example, for the feature $stDevYCC_j$, the respective step of the rating algorithm will proceed as follows: the image which has the lowest standard deviation of the y will be given the maximum feature rating and the image which has the highest standard deviation of the y will be given the minimum feature rating. After each $bImage_j$ is rated for each of its features, the image with the highest overall rating is identified as the correct binary text image.

(a) The input image

(b) The segmented image before the enhancement process

(c) The segmented image after applying the enhancement process

Figure 6.13: The impact of the enhancement step in the final result of the text segmentation process

6.4.4.5 Text Image Enhancement

Finally, an enhancement process takes place on the binarized text image. All connected components that either: 1) are too small or 2) lie on the boundaries of the text image and their bounding rectangles have an intersection with other bounding rectangles are removed. The components that lie on the boundaries of the text image but their bounding rectangles do not have an intersection with other bounding rectangles are only partially removed.

The application of a morphological "open" operation for breaking the possible bridges across the characters concludes the operation.

An illustration of the enhancement process is given in Figure 6.13.

6.5 Performance Evaluation

To evaluate the performance of the proposed text segmentation methods (UTS method and AFTS method), character/word recognition experiments have been conducted. The recognition rate is used to objectively measure the performance of the text segmentation methods. In addition, for comparison purposes, the Otsu thresholding method [Ots79] which is briefly described in Section 6.5.2 was selected.

Two different test set of images are used to evaluate the UTS and AFTS methods. The first test set ($TestSet_{MPEG-7}$) is the public MPEG-7 test set [XSLZ01] which consists of 45 images. There are 265 words (including prepositions and acronyms of names, which are also counted as words) or

1481 characters in this test set. The second test set ($TestSet_{Segmentation}$) consists of 18 video frames with about 176 words or 1203 characters. Both test sets are selected to cover a wide variety of background complexity and different text color, font, size and polarity. In total, there are 441 words or 2684 Latin characters.

6.5.1 OCR Software

We have used a demo version of the commercial OCR software ABBYY FineReader 7.0 Professional [Abb] for recognition purposes. After segmentation, the segmented binary text image was fed manually into the OCR software and the correctly recognized characters and words were manually counted. The character and word recognition rate are defined using the formula:

$$\text{CRR (WRR)} = \frac{\text{Correctly Recognized Characters (Words)}}{\text{Total Number of Characters (Words)}} * 100 \quad (6.14)$$

6.5.2 Otsu Algorithm

The Otsu method [Ots79] is a well known used thresholding techniques in grayscale image analysis, which is widely used in image segmentation. Suppose that $\{p_i\}_{i=0}^{L}$ is the gray-level image histogram of an image, and t is the selected threshold gray level value. We can then calculate the probabilities of the background and of the foreground when the threshold t is used, using Equation 6.15.

$$P_B^t = \sum_{i=0}^{t} \frac{1}{p_i} \text{ and } P_F^t = 1 - P_B^t = \sum_{i=t+1}^{L-1} \frac{1}{p_i} \quad (6.15)$$

Based on Equation 6.15, we can further define the mean and the variance of the pixels classified as background and as foreground as shown in Equations 6.16a and 6.16b, respectively.

$$\mu_B^t = (\frac{1}{P_B^t}) \sum_{i=0}^{t} ip_i \text{ and } {\sigma_B}^t = (\frac{1}{P_B^t}) \sum_{i=0}^{t} (i - \mu_B^t)^2 p_i \quad (6.16a)$$

$$\mu_F^t = (\frac{1}{P_F^t}) \sum_{i=t+1}^{L-1} ip_i \text{ and } {\sigma_F}^t = (\frac{1}{P_F^t}) \sum_{i=t+1}^{L-1} (i - \mu_F^t)^2 p_i \quad (6.16b)$$

The between class variance is denoted as:

$$\sigma_{between-class}^t = P_B^t(\mu_B^t - \mu)^2 + P_F^t(\mu_F^t - \mu)^2 = P_B^t P_F^t(\mu_B^t - \mu_F^t)^2 \quad (6.17)$$

Table 6.2: The recognition performance when the OCR is applied directly on the original text image (without taking place previously text segmentation tasks)

Image resolution	**CRR**	**WRR**
72 dpi	49.1%	24.9%
300 dpi	65.1% **(+ 16.0%)**	34.1% **(+ 9.2%)**

where μ is the overall mean of the grayscale values of the image and it is calculated as $\mu = \sum_{i=0}^{L-1} ip_i$. The within class variance is defined as shown in Equation 6.18.

$$\sigma^t_{within-class} = P^t_B var^t_B + P^t_F var^t_F \tag{6.18}$$

Applying the Otsu algorithm, the optimal threshold t_{Otsu} is the threshold t that maximizes the parameter $\sigma^t_{between-class}$ which in turn minimizes $\sigma^t_{within-class}$ as denoted in the following equation.

$$t_{Otsu} = \arg\{\max_{1\leq t\leq L}(\sigma^t_{between-class})\} = \arg\{\min_{1\leq t\leq L}(\sigma^t_{within-class})\} \tag{6.19}$$

6.5.3 Resolution Enhancement

Twenty images were selected from $TestSet_{MPEG-7}$ and $TestSet_{Segmentation}$ test sets to conduct some preliminary evaluation experiments. There are 205 words and 1404 characters in those 20 images.

To test the impact of resolution enhancement to subsequent text segmentation and character recognition tasks, two experiments were conducted (see [GESF04]). First, the accuracy of the OCR is measured when it is applied directly on the original image (resolution 72 dpi) and the enhanced image (resolution 300 dpi). Second, the original (and enhanced) text image is first segmented (e.g. using the proposed UTS method) and then the accuracy of the OCR is again evaluated.

The obtained OCR results for both experiments are shown in Tables 6.2 and 6.3, respectively. A character (word) recognition rate of 65.1% (34.1%) is obtained when the OCR is employed directly on the enhanced text image (see Table 6.2). From Table 6.3 it can be observed that the enhancement of the image resolution followed by the application of a text segmentation method improves further the performance, leading to the highest OCR accuracy with a CRR and a WRR of 85.9% and 65.4% respectively (CRR is increased by 20.7% and WRR by 31.3%).

Table 6.3: The recognition performance after applying the UTS method on different image resolutions. Note that the same feature vector (color + StdDev. of Wavelet Coefficients) is used in both experiments

Image resolution	**CRR**	**WRR**
72 dpi	70.7%	46.8%
300 dpi	85.9% **(+ 15.2%)**	65.4% **(+ 18.6%)**

Table 6.4: Word and character recognition performance when applying the UTS method on the ground truth data

	$TestSet_{MPEG-7}$	$TestSet_{Segmentation}$	**Overall**
WRR	65.7%	54.1%	**59.9%**
CRR	78.1%	70.2%	**74.1%**

To conclude, the enhancement of the image resolution and the employment of a text segmentation method before subsequent optical character recognition tasks is very important in order to increase the accuracy of the OCR results.

6.5.4 Text Segmentation Performance Evaluation

All the parameters used in the proposed methods are evaluated experimentally, and the values that gave the best results on the average are used throughout these experiments. The resolution of images is first increased to 300 dpi when it is smaller. During the feature extraction process, the wavelet 5/3 filter bank [VBL95] was used again, with the low-pass filter coefficients {-0.176777, 0.353535, 1.06066, 0.353535, -0.176777} and the high-pass filter coefficients {0.353535, -0.707107, 0.353535}. The sliding window size was set to 3×3 pixels during the calculation of the standard deviation. In order to find the combination of features which give the best results, eight different combinations were investigated:

- RGB or CIE La*b* color Components;

Table 6.5: Word and character recognition performance when applying the AFTS method on the ground truth data

	$TestSet_{MPEG-7}$	$TestSet_{Segmentation}$	**Overall**
WRR	78.1%	78.0%	**78.0%**
CRR	90.7%	90.0%	**90.4%**

Table 6.6: Word and character recognition performance when applying Otsu algorithm on the ground truth data

	$TestSet_{MPEG-7}$	$TestSet_{Segmentation}$	**Overall**
WRR	68.0%	52.8%	**60.0%**
CRR	79.7%	70.6%	**74.9%**

- RGB or CIE La*b* color Components + Wavelet Coefficients;
- RGB or CIE La*b* color Components + Standard Deviation of Wavelet Coefficients;
- RGB or CIE La*b* color Components + Wavelet Coefficients + Standard Deviation of Wavelet Coefficients

The UTS method has achieved the best results in terms of both CRR and WRR using as features the components of the RGB color space and the standard deviation of the wavelet coefficients. The parameters for the estimation of text and background color were set to: $nr_{color} = 6$, $nr_{\text{text rows}} = 4$ and $nr_{\text{backgr rows}} = 2 + 2$.

The AFTS method has shown best results using the components of the RGB color space and the wavelet coefficients as features and setting the rest of parameters to the following values: in the RGB color space $th_{similar} = 60$, $th_{compact} = 0.7$ and the Euclidean distance is employed to evaluate the distance between two certain colors.

Furthermore, the Otsu method was selected for comparison purposes.

The proposed methods are evaluated in two groups of experiments. In the first group, the ground truth data are used as the input of the segmentation methods in order to avoid the impact of the localization algorithms on their

accuracy. In the second group, the text boxes generated by the text localization method, namely UTDL-Texture approach presented in Section 3.3.3, are used instead of the ground truths.

6.5.4.1 UTS and AFTS Segmentation Accuracy

The recognition results obtained during the first group of experiments for the UTS, AFTS and Otsu methods are listed in Tables 6.4, 6.5 and 6.6, respectively. The second row of the tables shows the word recognition rate (WRR) while the third row shows the character recognition rate (CRR). The second and third column give the results for the test set $TestSet_{MPEG-7}$ and $TestSet_{Segmentation}$ separately, while the last column shows the overall WRR and CRR values.

An overall CRR of 74.1% and a WRR of 59.9% is obtained when the UTS method is employed for text segmentation purposes. The Otsu algorithm has performed similarly to the UTS method, attaining a CRR of 74.9% and a WRR of 60.0%. The AFTS method has achieved the best performance with an overall CRR of 90.4% and an overall WRR of 78% outperforming both the other two methods.

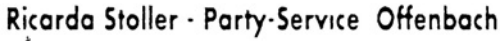
In Figures 6.14, 6.15 and 6.16, the segmentation results using the AFTS, UTS and Otsu method and the corresponding OCR results for the images in Figure 6.3 are shown.

From the experiments it was observed that the UTS and Otsu methods perform well until the background where the text lies is simple and they fail in segmenting accurately the text when the background gets more complex. In those cases, the UTS method could not determine the color of the text and that of the background correctly and moreover, more than two clusters of colored pixels are present. The AFTS method outperforms Otsu and UTS, due to its flexibility in the determination of the number of clusters and the employment of a fuzzy clustering algorithm instead of a hard one. However it still fails during the segmentation of a text that appears without a dividing border in a very similarly colored background. Figure 6.17 illustrates one of these scenarios.

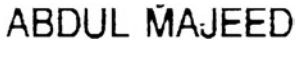
In [OC02], [WJC02] and [YGH04], the binarization method of Otsu [Ots79] was implemented and used for comparison purposes. In [WJC02], the authors' implementation of the Otsu method led to a low CRR of 47.3%, and their own approach achieved a CRR of 85%. In [OC02], the authors reported a WRR and CRR of 91.7% and 92.5% for their own approach, which to the best of our knowledge belongs to the highest performance in terms of WRR/CRR reported in literature for a text segmentation method. Their implementation of the Otsu algorithm [Ots79] in their test set has achieved a WRR and CRR of 89.3% and 88.4%. Otsu's method [Ots79] tested in our test set has achieved a WRR and CRR of 60.0% and 74.9%, whereas our

(a) The OCR result is: "es wirkten mit"

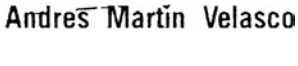
(b) The OCR result is: "PRODUKTIONSLEITUNG"



(c) The OCR result is: "Ricarda Stoller-Party-Service Offenbach"



(d) The OCR result is: "ABDUL MAJEED"



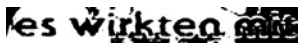
(e) The OCR result is: "Arrdres "Martin Velasco"

Figure 6.14: The segmentation using the AFTS method and the respective OCR results for the images in Figure 6.3

AFTS method has obtained an overall WRR and CRR of 78.0% and 90.4%. It is obvious that the proposed AFTS method as well as our implementation of Otsu method have shown a lower value of WRR when applied in our test sets. Furthermore, the authors of [YGH04] have also reported for their unsupervised text segmentation method a WRR of 85.5%, which is slightly lower than ours. Consequently, a clear conclusion about the quality of our method compared to [WJC02, OC02, YGH04] cannot be drawn due to the fact that the results are reported for different test sets and OCR engines. However, we believe that the reported performance results for our AFTS method are at least competitive to the best results reported by other researchers.

6.5.4.2 UTS and AFTS Combined with TDL

In the second group of experiments, evaluations were conducted only on $TestSet_{MPEG-7}$, and the text boxes generated by the text localization method presented in Section 3.3.3, namely UTDL-Texture, are used as the input of the three different segmentation algorithms. The UTDL-Texture method has



(a) The OCR result is: "fes wlrtoea"

(b) The OCR result is: "P tföD U K T10 N S L EliTiÜ N G"

(c) The OCR result is: "IJScorooiStfflieERorraSeSyice, Offenbach"

(d) The OCR result is: "AU3HWE1D"

(e) The OCR result is: "AndreiTVIaffih Veiascb"

Figure 6.15: The segmentation using the UTS method and the respective OCR results for the images in Figure 6.3

localized 88.88% of the text pixels present in $TestSet_{MPEG-7}$ (see [GEF04a]).

The combination of the UTDL-Texture method with the AFTS method using the ABBYY FineReader OCR engine [Abb] has achieved an overall CRR of 77%. When the AFTS method was substituted by the Otsu method [Ots79] (or the UTS method presented in Section 6.4.3), an overall CRR of 50% (or 41.4%) was obtained. Even in this case, the proposed AFTS method has shown a better performance than the methods introduced in [Ots79] and the UTS method. Figure 6.18 shows two examples of how the different text segmentation methods perform.

A similar experiment was also conducted in [NC05] using the same data set as $TestSet_{MPEG-7}$. The authors have applied their text detection and segmentation method, whereas the recognition of characters is performed using the OmniPage OCR engine [omn]. The authors have indicated a CRR of 72% (1006 out of 1481 characters were correctly recognized) and a precision

(a) The OCR result is: "fes wirkten"

(b) The OCR result is: "PRgDUKTlQNSl&JMNS"

(c) The OCR result is: "lRKOwKtolieERon¿5wice, Offenbach"

(d) The OCR result is: "BflMCTL"

(e) The OCR result is: "AntirenWafBh Veiasco"

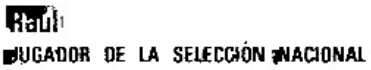
Figure 6.16: The segmentation using the Otsu method and the respective OCR results for the images in Figure 6.3

of 73%, where the precision is defined as denoted in Equation 6.20.

$$\text{Precision} = \frac{\text{Correctly Recognized Characters}}{\text{Total Characters Output from OCR}} \tag{6.20}$$

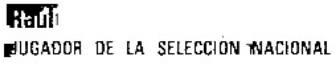
The combination of our UTDL-Texture algorithm in Section 3.3.3 with our AFTS method has shown a better performance (CRR = (72 + **7**)% and Precision = (73 + **19**)%) compared to the performance recently presented in [NC05]. However, it would be emphasized the fact that different OCR engines were used. Figure 6.19 gives some additional examples to illustrate the different steps of the text extraction process for different input images. In the last two examples, although the UTDL-Texture method has correctly localized all the text present in those images, the AFTS method has failed to segment all of them accurately. The reason is that during the segmentation of the words "WILLIE" and "US", the text cluster is not identified correctly. In the first case it is caused from the noise present in the "right" text cluster, whereas in the second case it is caused due to the fact that the word "US" is very short.

(a) The localized text using the UTDL-Texture method



(b) Otsu segmentation

(c) UTS segmentation



(d) AFTS segmentation

Figure 6.17: The illustration of a non-accurately segmented text images

Finally, it must be pointed out that a performance measure such as CRR i.e. WRR not only depends on the accuracy of the used segmentation methods but also on the quality of the OCR engine. During the experiments it was observed that in some cases, despite visually good segmentation results, the OCR fails to recognize the text image correctly. Figure 6.20 describes some examples where the OCR engine did not recognize any characters although the segmentation results are very good. This is probably due to the present font. The combination of the results obtained from several OCR engines, through a voting strategy, can help in improving the accuracy of the OCR. In addition, applying a final linguistic analysis can further correct some errors.

6.5.5 Time Complexities of the Algorithms

The UTS method proposed in Section 6.4.3 has a complexity of $O(NCk + N^{1/3})$, where N is the total number of pixels in the text image, C is the maximum number of iterations of the k-means algorithm, k is the number of clusters and is equal to 2, $O(N^{1/3})$ is the complexity of the quantization method, and $O(NCk)$ is the complexity of the k-means algorithm.

The AFTS algorithm introduced in Section 6.4.4 has a complexity of $O(NC^2k)$, since the maximum number of iterations for the AFTS is bounded

(a) The localized text using the UTDL-Texture method

(b) The corresponding Otsu segmentation

(c) The corresponding UTS segmentation

(d) The corresponding AFTS segmentation

Figure 6.18: Illustration of the text extraction results

by C, and the complexity of the fuzzy clustering algorithm is $O(NCk)$.

The Otsu thresholding method [Ots79] has a time complexity of $O(N)$.

Considering that the initial number of clusters in the UTS method (in Section 6.4.3) has a value of two and for the AFTS method (in Section 6.4.4) it depends on the image colors, the AFTS algorithm is clearly computationally more expensive. However, the increase in complexity is justified by two reasons: (I) the AFTS method has shown a much better performance than the other two methods in terms of CRR and WRR; and (II) the segmentation process usually takes place offline, since the final purpose is the text-based indexing of the images.

6.6 Summary

In this chapter, two novel algorithms for text segmentation in images with complex backgrounds were presented.

The first method (UTS method introduced in Section 6.4.3) proceeds as

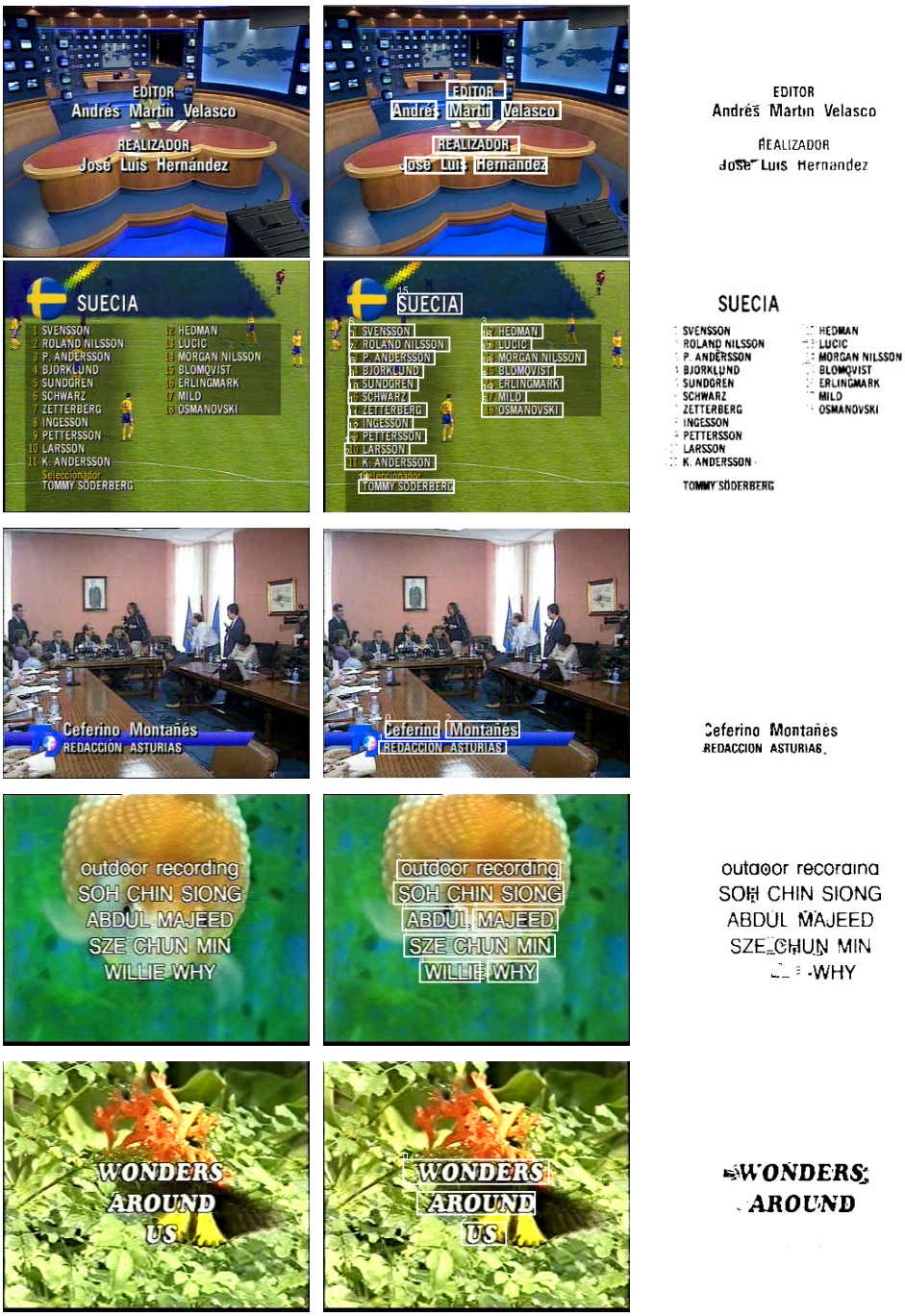


Figure 6.19: Illustration of the text extraction process; Input image; The localization of the text; The segmentation results using the AFTS method

follows. First, the text color is determined using a color-quantizer and line histograms. Then, the R, G, and B color components and the three high-

Figure 6.20: Illustration of two cases where OCR fails in recognizing the characters; The original text image and the segmentation results using the UTS (on the left) and the AFTS (on the right) method

frequency wavelet coefficients are used as the best features for the subsequent classification into two clusters: "text" and "background" pixels. The classification is done by a slightly modified k-means algorithm.

In Section 6.4.4, the idea of segmenting the localized text from complex background employing unsupervised learning methods is further explored and extended by proposing a novel method (called AFTS) where the following new aspects are introduced: I) a flexible number of clusters, which depends on the image colors; II) unsupervised fuzzy classification of image pixels using the number of clusters defined in the first step; III) evaluation of the compactness of the clusters and their iterative improvement; IV) determination of the cluster where the text pixels are classified and its enhancement.

The character/word recognition rates were used to evaluate our methods. Both methods were evaluated on two test set of images with a total number of 441 words and 2684 Latin characters. The best results were achieved when the resolution of the original image was increased from 72 dpi up to 300 dpi. The UTS method has achieved an overall WRR and CRR of 59.9% and 74.1%, respectively. AFTS has shown a better accuracy with an overall WRR and CRR of 78.0% and 90.4%, respectively.

Chapter 7

Script Recognition

A pattern has an integrity independent of the medium by virtue of which you have received the information that it exists...

- R. Buckminster Fuller -

7.1 Introduction

Several methods have been proposed for text extraction in images and videos during the last years. However, most of them do not address the problem of multilingual scripts. The script differences, such as font size, stroke density and stroke directions, which are not critical during text detection and localization, play a decisive role during text segmentation and binarization, since they may affect the binary results that will be further processed by an OCR engine [LSC05]. The multilingual capabilities of image and video text extraction systems are very important to digital video libraries. We attempt to solve the problem of script recognition, where either all localized text in an image or a part of it is classified into one of the two scripts (Latin or Ideographic), with the aim of facilitating the text segmentation process or indexing and retrieval of images, and improving the accuracy of text recognition by employing the appropriate OCR algorithm.

The problem of recognizing the script in printed materials [Spi97, HKTK97, Tan98, TLH99, SBN+98, WBSK98, MD03, MD04, JMD05] has gained particular attention in document image analysis and retrieval. Different processing tasks, such as indexing or translation, need information about the language used in a document. Thus, the recognition of the script is equivalent to the

identification of the language for scripts used by only one language, and is necessarily the first step towards language identification for scripts shared by many languages. For example, after recognizing the used script in a document as Latin, character and word shapes can be analyzed in order to identify the language e.g. English, German, Italian, etc. However, the problem of identifying the actual language is beyond the scope of this chapter.

In analogy to the problem of indexing document images, the possible presence of Latin and Ideographic text in images/videos coming from international sources is an obstacle towards the automation of the text extraction process in general and the text segmentation and the OCR procedure in particular.

In this chapter, an approach for discriminating between Latin and Ideographic script is presented. The proposed approach proceeds as follows. After, a set of low-level features has been extracted from the localized text image, the decision about the type of the script is made using a k-Nearest Neighbour classifier. Experimental results for a set a set of 230 text images containing different scripts demonstrate the good performance of the system with an accuracy of 85.3% and of 89.0% for the Latin and Ideographic script, respectively. Part of the material presented in this chapter has been published in [GF05b].

The chapter is organized as follows. Section 7.2 gives a brief overview of the work done in the area of script recognition in document images. In Section 7.3, the problem of script recognition in complex images is discussed. Section 7.4 introduces the individual steps of our script recognition approach in detail. Section 7.5 presents the experimental results obtained for a set of images. Section 7.6 concludes the chapter.

7.2 Previous Work

Previous work dealing with the problem of recognizing the script in text documents can be classified into two main groups:

- *block-based approaches* [Spi97, HKTK97, Tan98, TLH99, SBN+98] and [WBSK98] base the decision on the whole document image, assuming that it is written with the same script;

- *word-based approaches* [MD03, MD04, JMD05] base the decision only on a word, allowing in this way the identification of different scripts in the same document image.

One of the first attempts to enable automatic script identification was done by Spitz [Spi97]. The method first classifies the script into two main classes:

Roman and Han. This classification is done using the spatial relationship of features related to the upward concativities in the character structures.

Hochberg et al. [HKTK97] have described an automated script identification system for document images. First, the most frequently occurring templates of each script are created by clustering the extracted textual symbols (connected components that meet certain size requirements) from a training set. Then, symbols from a new image are compared to the templates to find the best script. The proposed approach is based on the observation that different scripts have unique shape characteristics and can distinguish between thirteen different scripts.

Tan [Tan98] has presented a set of rotation invariant texture features based on the multi-channel Gabor filtering technique, to identify six types of script on machine printed documents. Similar to Hochberg et al. [HKTK97], representative features are extracted for each script by computing the average values of the texture features for all training document images. For an unknown document, the same features are extracted and compared to the representative features in order to identify the script of the new document.

Tan et al. [TLH99] have also proposed a method to identify three different scripts: Roman, Chinese and Tamil. They use basic document features like elongation of bounding boxes of character cells and the position of upward concavities, which were used in [Spi97], too.

Suen et al. [SBN+98] have presented a solution for script classification which is based on analyzing the connected components and the horizontal projections of the text image. Four script classes were considered in their work: Roman, Arabic, Ideographic and Cyrillic.

Walked et al. [WBSK98] classify scripts by analyzing the distribution of the heights of the connected components and the horizontal projection, similar to [SBN+98].

Ma and Doermann [MD03] have proposed a supervised multi-class classifier to identify three types of scripts. After the input images has been scaled and replicated (if necessary), a set of features is extracted using Gabor filters. Classification is performed at the word level based on given training samples for each class.

Ma and Doermann [MD04], have further extended the system presented in [MD03], and results using three different classifiers, namely nearest neighbors, Gaussian mixture models and support vector machines, were reported. Jaeger et al. [JMD05] have additionally shown that combining informational confidence values can improve the results of individual classifiers.

Ablavsky and Stevens [AS03] have proposed a system that identifies two target scripts based on a set of extracted features (like Cartesian moments and compactness) from the binary image. The k-Nearest Neighbors algorithm is used to recognize the script.

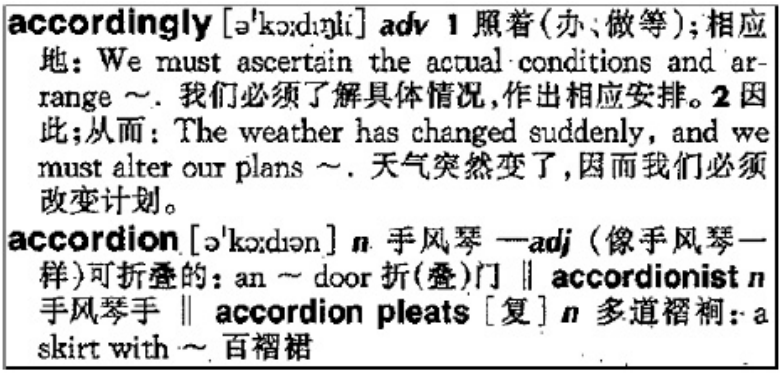


Figure 7.1: A sample image from the set of document images used in [MD04]

7.3 Script Recognition in Complex Images

The approaches discussed in Section 7.2 address the problem of script recognition in machine printed document images (see Figure 7.1), which means that the text in black appears on a simple background (mostly white). Most of them [Spi97, HKTK97, Tan98, TLH99, SBN+98, WBSK98] operate on the block or page level, and only a few of them [MD03, MD04, JMD05] try to recognize the script at the word level.

In this chapter, it is attempted to extend the script recognition task to cases significantly more challenging than previously reported, which consist of recognizing script under complicated conditions including: complex background, colored text, low resolution, noise caused by the image acquisition and compression process, and recognition at the word/text line level.

The motivation to do script recognition in images/videos with complex backgrounds originates from the effort to extract the text in images or videos from different multilingual sources. Language dependent characteristics such as: (I) stroke density; (II) font size; (III) aspect ratio; and (IV) stroke statistics affect the results of text segmentation and binarization (TSB) methods [LSC05]. Many TSB methods are based on the assumption that text pixels have a different color than background pixels, thus thresholding or clustering methods [GF05a] are used to separate the text from the background. In both cases, text-like characteristics (geometric and statistical properties of text characters) are used to find out which of the segmentation results represent text. Other TSB methods employ different filters, such as the asymmetric Gabor filter [CSB01] or the four direction character extraction filter [SKHS98], to enhance the text strokes. However, these filters are more adequate to segment Latin characters than Ideographic characters [LSC05]. In this context, it is necessary to know the script in order to adapt the text-like characteristics (in the first group of TSB methods) and the filters (in the second group of

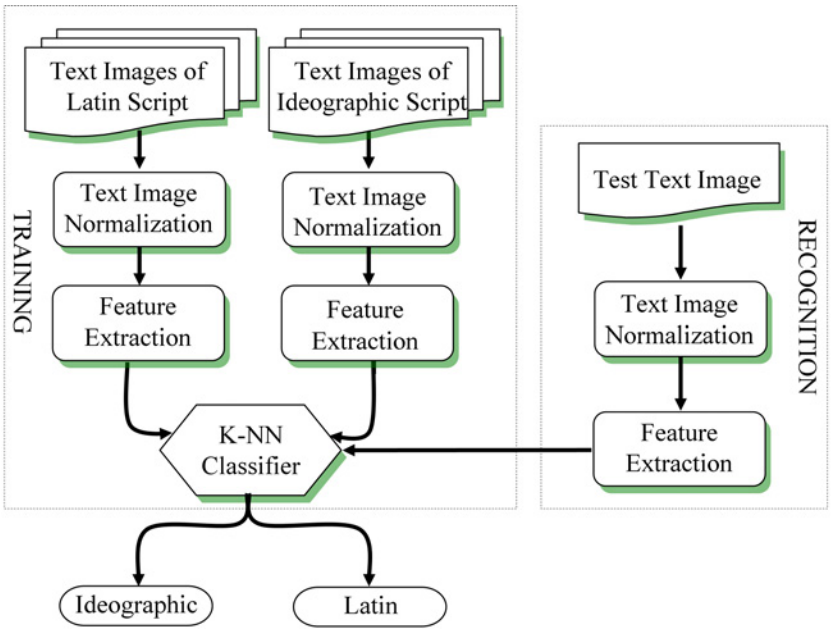


Figure 7.2: The flow chart of the script recognition module

TSB methods), before using the TSB method. Moreover, the recognition of the script can help to increase the accuracy of the OCR engines by employing the right OCR for each different script.

7.4 Script Recognition Module

The proposed approach is based on the following strategy. First, given the coordinates of a text in an image (that are found using the UTDL-Texture algorithm presented in Section 3.3.3, which does not require prior knowledge about the language of the text), the corresponding text image is extracted. Then, after some normalization procedures, different features are estimated from the localized text. Finally, the decision about the type of the script is made based on the estimated features using a previously trained classifier.

Figure 7.2 presents the flow chart of the system consisting of three main steps: I) text normalization; II) feature extraction; and III) script classification. The input of the system is the original text image. Examples from the test set used in this paper are shown in Figure 7.3. In the following subsections, the different steps will be discussed in detail.

(a) Six Latin scripts

(b) One Ideographic script

(c) One Ideographic script

Figure 7.3: The results of text localization

7.4.1 Text Normalization

In different images, text may occur with various widths and heights. To have consistent features through all the text images and to reduce the variance of the size of the characters, the height of a text image is normalized. The height of the text image is scaled to a predefined size H_f (e.g. $H_f = 30$ pixels) using a cubic interpolation, whereas the aspect ratio width/height is still preserved in order to avoid distortion of individual characters of the text image.

7.4.2 Feature Estimation

Given the normalized text image (I_{NT}), a set of features which characterize the present text are estimated. We evaluated five different kinds of low-level features in order to find those which allow to distinguish between a Latin text and an Ideographic text under a complex and colored background. The feature extraction process takes place on the wavelet domain (see Chapters 2 and 3, [VBL95]). The wavelet coefficients of the high frequency sub-bands are again used here to evaluate the features.

Furthermore, the text image, I_{NT}, is divided into m different areas, and the features are extracted from each of the areas separately in order to capture local properties of the text image. In Figure 7.4, different division possibilities are illustrated.

In the following subsections, the selected features are discussed in detail.

7.4.2.1 Mean and Standard Deviation of the Edge Pixels

The first features that are evaluated are the mean M_{A_i} and standard deviation $stDev_{A_i}$ of wavelet coefficients in a certain area A_i. The mean is calculated

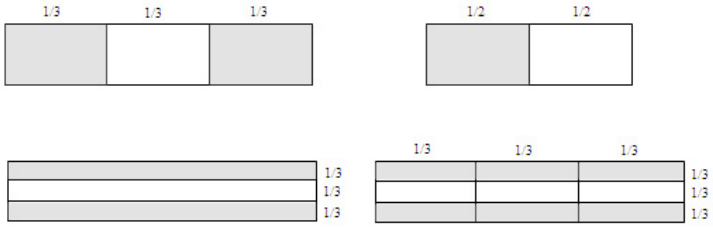


Figure 7.4: Different possible patterns to divide the text image into several areas

as follows:

$$M_{A_i} = \frac{1}{|A_i|} \sum_{(x,y)\in A_i} w(x,y), \text{ for } i = 1..m \tag{7.1}$$

where $|A_i|$ is the number of pixels in area A_i, $w(x,y)$ is the value of the wavelet coefficient in one of the sub-bands at position (x,y). The standard deviation of the edge pixels is defined as:

$$stDev_{A_i} = \sqrt{\frac{1}{|A_i|} \sum_{(x,y)\in A_i} (w(x,y) - M_{A_i})^2}, \text{ for } i = 1..m \tag{7.2}$$

7.4.2.2 Density of Edge Pixels

The density of edge pixels in each area A_i is calculated from each of the wavelet sub-bands (LH and HL) separately. It indicates the percentage of edge pixels in a certain area A_i.

$$D_{A_i} = \frac{N_{EP}}{|A_i|}, \text{ for } i = 1..m \tag{7.3}$$

where N_{EP} is the number of strong edge pixels and $|A_i|$ shows the total number of pixels in the area A_i. An edge pixel is classified as a strong edge pixel if its absolute value is higher than a fixed threshold Th. This feature is based on the fact that the edge pixels are spread evenly over all three zones of an Ideographic script, whereas in the case of a Latin script they are concentrated in the middle zones (lowercase letters) or in the border zones (uppercase letters).

7.4.2.3 Energy of Edge Pixels

Considering the contrast between text and background pixels, the wavelet coefficients around the text pixels will normally show large values. The energy of the edge pixels in an Ideographic text is higher than in a Latin text, since

an Ideographic text typically contains more strokes than a Latin text. The energy of edge pixels (E) can be calculated from the wavelet sub-bands LH or HL for each area A_i of the text image using the Equation 7.4:

$$E_{A_i} = \frac{1}{|A_i|}(\sum_{(x,y)\in A_i} |w(x,y)|) \text{ for } i = 1..m \tag{7.4}$$

where $|w(x,y)|$ is the absolute value of the wavelet coefficient at position (x,y).

The short term energy of edge pixels ($ShTE$) can be evaluated from the formula:

$$ShTE_{A_i} = \frac{1}{|A_i|}\sqrt{\sum_{(x,y)\in A_i} w^2(x,y)} \text{ for } i = 1..m \tag{7.5}$$

Due to the reason that an Ideographic script consists of more strokes than a Latin script, it is assumed that the former will show a higher overall energy compared to the latter. The overall energy OE of the text image (I_{NT}) is calculated by integrating the wavelet coefficients in the LH and HL sub-bands using the formula:

$$OE_{I_{NT}} = \frac{1}{|I_{NT}|}\sum_{(x,y)\in I_{NT}} (|w_{HL}(x,y)| + |w_{LH}(x,y)|) \tag{7.6}$$

where $|I_{NT}|$ shows the total number of pixels in the text image I_{NT}.

The overall short term energy $OShTE$ of the text image I_{NT} is calculated using the following formula:

$$OShTE_{I_{NT}} = \frac{1}{|I_{NT}|}\sqrt{\sum_{(x,y)\in I_{NT}} w^2_{HL}(x,y) + w^2_{LH}(x,y)} \tag{7.7}$$

7.4.2.4 Horizontal Projection

The horizontal projection HP of the normalized text image with $W \times H_f$ dimension is evaluated where each y^{th} element of the HP shows the number of strong edge pixels in the y^{th} row of the text image. The standard deviation of the evaluated horizontal projection is used as a feature to capture the distribution of the edge pixels within the rows of the text image.

$$HP_y = \sum_{\forall x | w_{LH}(x,y) > Th} 1 \text{ for } y = 1..H_f \tag{7.8}$$

7.4.2.5 Cartesian Moments of the Edge Pixels

The Cartesian moments as 2D shape descriptors are calculated using the following formulas:

$$m_{pq} = \int\int x^p y^q w(x,y) dx dy \tag{7.9}$$

where $w(x, y)$ is the wavelet coefficient of the text image at position (x, y), and usually $p, q = 0, 1, 2$. The normalized moments are computed (Equation 7.10) in order to remove the sensitivity to scale and character position.

$$\eta = \frac{\mu_{pq}}{\mu_{pq}^{\gamma}} \text{ where } \gamma = \frac{p+q}{2} + 1 \tag{7.10}$$

where μ_{pq} are the centralized moments and are calculated as:

$$\mu_{pq} = \int \int (x - \bar{x})^p (y - \bar{y})^q w(x, y) dx dy \tag{7.11}$$

7.4.3 Script Classification

Supervised classification methods can be used to identify a sample pattern. In this section, the k-nearest neighbor classifier is chosen to classify a text image into one of two groups: Latin or Ideographic script.

The k-nearest neighbor classification method uses the k observations in the training set T closest to I_{NT} in the feature space to form the decision function $Y()$. Specifically, the k-nearest neighbor for the input text image I_{NT} is defined as follows:

$$Y(I_{NT}) = \frac{1}{k} \sum_{x_j \in N_k(I_{NT})} y_j \tag{7.12}$$

where $N_k(I_{NT})$ is the neighborhood of I_{NT} defined by the k closest points x_j in the training sample and y_j is the label (e.g. 0 for Latin and 1 for Ideographic) of the group where the point x_j belongs to. After the k observations x_j closest to I_{NT} have been found, their responses are then averaged to take the final decision for the script of the text image I_{NT}. If $Y(I_{NT})$ is larger than 0.5, then the script is identified as Ideographic, otherwise as Latin.

The term closeness implies a metric. For this purpose, three different metrics [MKB79] are investigated:

1. Euclidean distance, as shown in Equation 7.13a;

2. Manhattan distance, as shown in Equation 7.13b;

3. Bhattacharyya distance, as shown in Equation 7.13c.

In Equation (7.13), d shows the dimensions of the feature space and x and z

Table 7.1: The accuracy of the script recognition approach based on a k-NN classifier for different values of k and using three different distances to define the closeness between two text images

Distance	Script	Accuracy of the k-NN classifier with			
		$k=7$	$k=5$	$k=3$	$k=1$
Euclidean	Latin	85.3%	**83.7%**	80.6%	77.5%
	Ideographic	80.8%	**89.0%**	93.2%	91.9%
Manhattan	Latin	87.6%	83.7%	**85.3%**	79.8%
	Ideographic	78.1%	79.5%	**89.0%**	87.7%
Bhattacharyya	Latin	86.1%	**84.5%**	81.4%	79.9%
	Ideographic	79.5%	**89.1%**	90.4%	93.2%

represent the features of two different objects (text images).

$$Euclidean(x,z) = \left(\sum_{l=1}^{d} (x_l - z_l)^2\right)^{1/2} \tag{7.13a}$$

$$Manhattan(x,z) = \sum_{l=1}^{d} \frac{1}{d}|x_l - z_l| \tag{7.13b}$$

$$Bhattacharyya(x,z) = \left(\sum_{l=1}^{d} (\sqrt{x_l} - \sqrt{z_l})^2\right)^{1/2} \tag{7.13c}$$

7.5 Performance Evaluation

We have tested our script recognition algorithm using two test sets: $TestSet_{MPEG-7}$ and $TestSet_{Ideographic}$. There are 107 images in both test sets. These test sets were selected to cover a wide variety of background complexity and different text color, font, size and script types. The UTDL-Texture algorithm has localized 230 text areas in those images, where 149 consist of a Latin script and 81 consist of an Ideographic script. The text areas are typically composed of 1-7 words in the case of a Latin script. The evaluation of the proposed approach is done in terms of an accuracy measure for each of the scripts separately. The number of false alarms during these

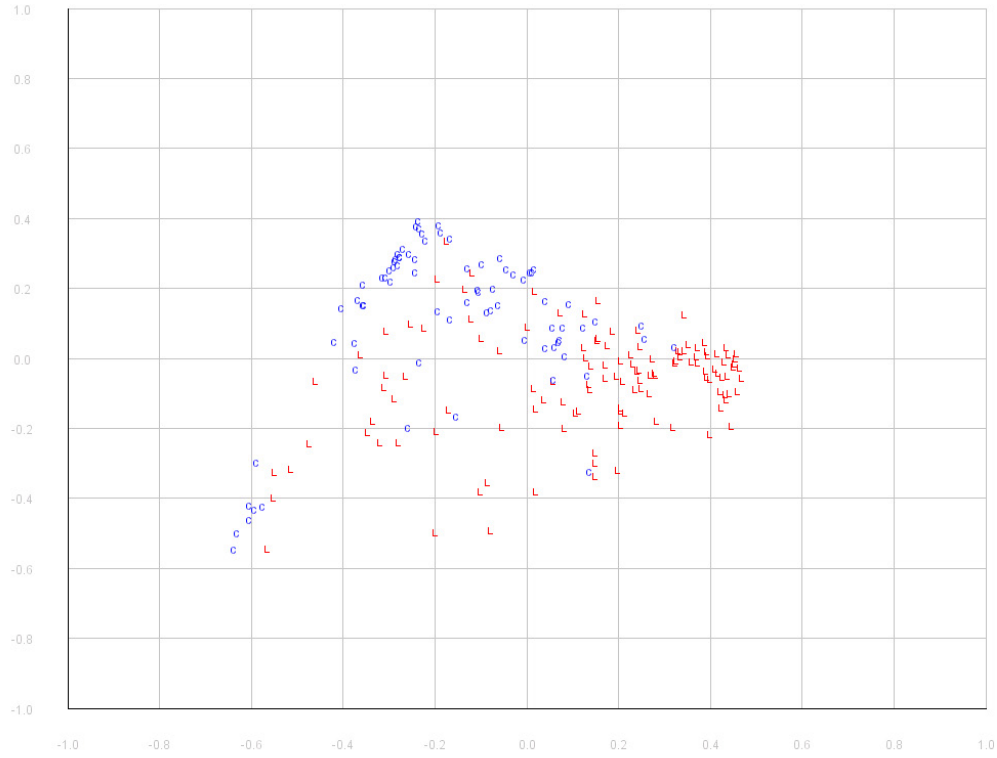


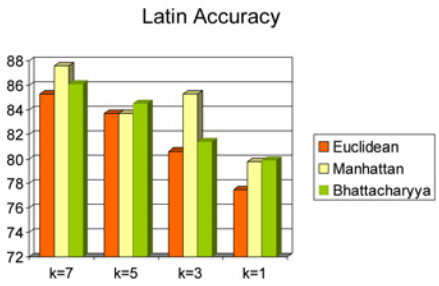
Figure 7.5: The 2D MDS plot of the text images used to evaluate the script recognition method. The Chinese text images are plotted in blue, whereas the Latin text images are plotted in red

experiments is bounded by the total number of images and for this reason the accuracy is chosen instead of the recall and precision. However, the definition of accuracy using the following formula is similar to that of recall in the previous chapters.

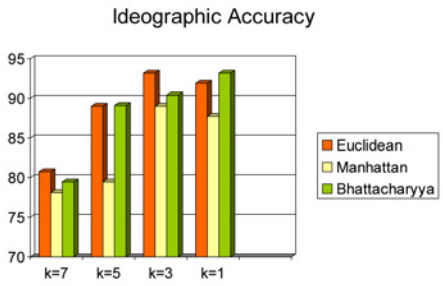
$$ACCURACY = \frac{\#\text{Correctly Recognized Text Images}}{\text{Total Number of Text Images}} \tag{7.14}$$

20 Latin and 8 Ideographic text areas were extracted from the test set to train the k-NN classifier. These text areas were excluded from the set of text areas which was used to test the accuracy of the approach. After that, the trained classifier is applied to all test text areas (129 Latin + 73 Ideographic). To evaluate the accuracy of the k-NN classifier, different values for k, namely $k = 1$, $k = 3$, $k = 5$ and $k = 7$ are tested. In addition, the Euclidean distance, the Manhattan distance and the Bhattacharyya distance were tested.

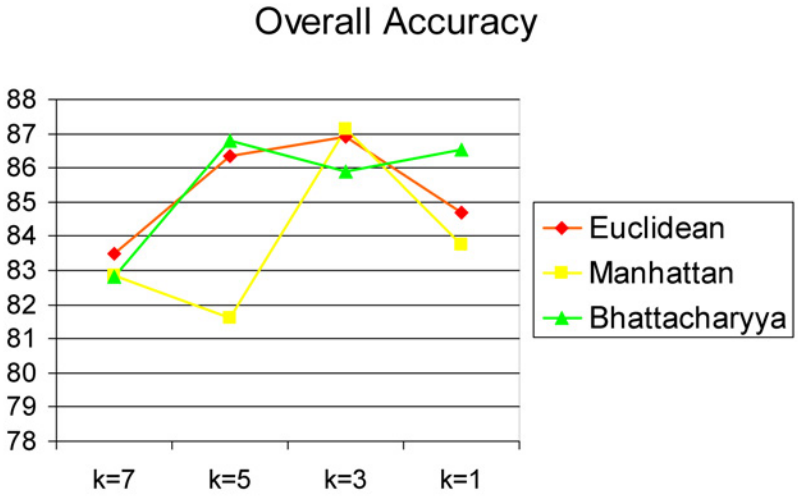
The wavelet 5/3 filter bank [VBL95] with the low-pass filter coefficients {-0.176777, 0.353535, 1.06066, 0.353535, -0.176777} and the high-pass filter



(a) Latin recognition accuracy



(b) Ideographic recognition accuracy



(c) Latin and Ideographic recognition accuracy

Figure 7.6: The visualization of the script recognition accuracy for different values of k and metrics

coefficients {0.353535, -0.707107, 0.353535} are used during the experiments. To find the set of features which give the best results, different combinations were investigated. The best results in terms of accuracy were achieved using the following parameters and set of features: the text image is divided into nine areas ($m = 9$ in Section 7.4.2), the density of edge pixels (D_{A_i}), the energy of edge pixels (E_{A_i}), the overall energy of edge pixels ($OE_{I_{NT}}$) and the standard deviation of the horizontal projection (HP).

Figure 7.5 presents the MDS (Multi-Dimensional Scaling) [MKB79] plot of the distance matrix built using the above selected features for all text images

used to evaluate the script recognition method. The MDS plot presents a configuration of all text images in the 2D Euclidean space using the evaluated distances between them. A character "C" i.e. "L" is drawn to discriminate between a Chinese i.e. Latin script. The figure shows that the groups of text images consisting of different scripts are not clearly distinguishable from each other.

In Table 7.1, the performance of the script recognition system for different parameter combinations is shown, whereas in Figure 7.6 the individual script (Latin or Ideographic) accuracy and their mean are visualized. The experiments indicate that the proposed approach has achieved the highest overall accuracy of 85.3% for Latin and of 89.0% for Ideographic scripts when k was set to 3 and the Manhattan distance was used. From Figure 7.6(c) it can be seen that the approach in the worst case yields an overall accuracy of 83.7% and 79.5% (using the Manhattan distance and $k = 5$) for the Latin and Ideographic script, respectively.

7.6 Summary

To extract text from images or videos coming from unknown international sources, it is necessary to know the script beforehand in order to employ suitable text segmentation and optical character recognition (OCR) methods. Thus, in this chapter, the problem of recognizing the script of a localized text in images with complex backgrounds has been addressed. A supervised method which bases its decision on a set of features extracted directly from the original image has been presented. Experimental results have demonstrated the good performance of the proposed method by distinguishing between a Latin and an Ideographic script with an accuracy of 85.3% and 89.0%, respectively.

Chapter 8

Comparison of Text Images

Each of the chemical elements is a pattern integrity. Each individual is a pattern integrity. The pattern integrity of the human individual is evolutionary and not static.

- R. Buckminster Fuller -

8.1 Introduction

Machine-printed character recognition research is typically concentrated on developing *analytical character recognition* methods. In this paradigm, words are segmented into elementary units (in the best case the elementary units are the characters), which are then recognized individually. Finally, the recognized characters are combined to identify the original word. Consequently, the correctness of these methods strongly depends on several factors: (a) the accuracy of the segmentation of the word into characters; (b) the variation in used fonts; and (c) the touching effect created when two characters intersect with each other. These problems have led researchers to consider other methods for text recognition such as *holistic recognition approaches*.

Cognitive psychology has indicated that *humans commonly recognize words holistically when reading a text* [Raw76]. *"Holistic recognition"* means that the word is treated as an inseparable unit, i.e. the whole word is recognized and not based on the characters that compose it. Holistic approaches have gained a lot of attention during the last years mainly due to their parallels to the strategy used by human readers. They have also found a broad application for handwritten word recognition. This is due to the difficulties

encountered during the segmentation of text into characters and to the fact that the shape of handwritten characters changes from one person to another, which makes the modeling of a specific character very difficult.

Text that appears in videos in general and scene text in particular are also very difficult to be recognized by conventional OCR systems due to several reasons: (I) the difficulties to segment and binarize the text accurately due to the complex background where it is embedded; (II) the use of diverse fonts; (III) the low quality of the image; and (IV) artistic effects. Consequently, the employment of holistic techniques to compare such text images would be very useful to recognize their meaning, or in the context of an automatic content-based image retrieval application, to retrieve images where a similar text appears.

In this chapter, a novel holistic technique to compare text images is proposed. The technique consists of three steps: (I) shape definition; (II) alignment of shapes; and (III) dissimilarity estimation. A slight modification of a well known corner detection algorithm [HS88] is suggested to extract the salient points in the text images. The Scott and Longuet-Higgins alignment method [SLH91] is employed to associate the detected salient points with each other. Furthermore, a new sequential correspondence finding algorithm is proposed as an alternative solution to the shape alignment problem. Finally, a measure is presented to evaluate the (dis)similarity level between text images on the basis of the established alignment. Experimental results for a set of noisy binary text images extracted from the commonly available MPEG-7 test set [XSLZ01] will be presented to demonstrate the performance of the proposed holistic approach. Part of the material presented in this chapter has been accepted for publication in [GQF06b].

This chapter is organized as follows. In Section 8.2, previous work related to the subject of this chapter, is shortly reviewed. The proposed holistic text comparison method is described in Section 8.3. Section 8.4 presents evaluation results related to the performance of the proposed methodology. Section 8.5 summarizes this chapter.

8.2 Previous Work

The proposed methods for text recognition are grouped into two main classes: (I) character based approaches; and (II) holistic approaches/word spotting. An overview of the first group of methods is written by Uchida and Sakoe [US05]. In this section, we will concentrate mainly on the methods that are related to the holistic approach.

Alon et al. [AAS05] present a method to calculate the similarity between two characters based on their exact alignment to a small number of prototypes.

Marinai et al. [MMS03] propose a system for indexing and retrieval of words in old documents. They employ an unsupervised prototype clustering of word images, followed by a string encoding for efficient string matching.

Lavrenko et al. [LRM04] suggest an approach for holistic recognition of words in handwritten historical documents. A Hidden Markov Model (HMM) is used to describe a document, where the words to be recognized represent the states of the HMM. The state transition probabilities are estimated from the frequencies of the word bigrams, whereas the observation probabilities are estimated using the different word features, such as word profile, length, height, etc.

Rothfeder et al. [RFR04] use the correspondence between the corners to rank handwritten word images in a document by their similarity. The similarity is measured by the accumulated displacement of corresponding image locations. In [RM04], the dynamic time warping technique is used to compare the word images. Features such as projection profile, word profile and background/ink transitions are used to characterize the words.

Mori and Malik [MM03] apply the well known shape context descriptor [MBM05] to break a visual CAPTCHA (which is an acronym for Completely Automated Public Turing test to tell Computers and Humans Apart. A test drive is EZ-Gimpy [cap]). The matching between two cluttered word images is done in two phases: fast pruning and detailed matching [BMP01].

Moy et al. [MJHP04] also address the problem of solving visual CAPTCHAs (EZ-Gimpy and Gimpy-r) and describe two distortion estimation techniques for object recognition to solve them.

A segmentation-free approach based on multi-angled parallel matching is proposed by Nakamura and Yamamoto [NY03] for text recognition in video frames. The character size variation is addressed and the application of Gaussian filtering and multi-sized reference patterns is discussed.

Mariano [Mar03] has also proposed a method for text matching in videos. After detecting a text in a frame, the approach looks for similar text in a database of video frames. La*b* color features are used to compare pairs of text boxes.

Recently, Silapachote et al. [SWH+05, SHW05] have proposed an approach for automatic sign detection and recognition. After detecting a sign region, it is then matched against a known database of signs in order to recognize it. A color support vector machine classifier is first employed to reduce the number of signs in the subsequent detailed matching, which consists of finding the correspondences of corners and their associated shape contexts.

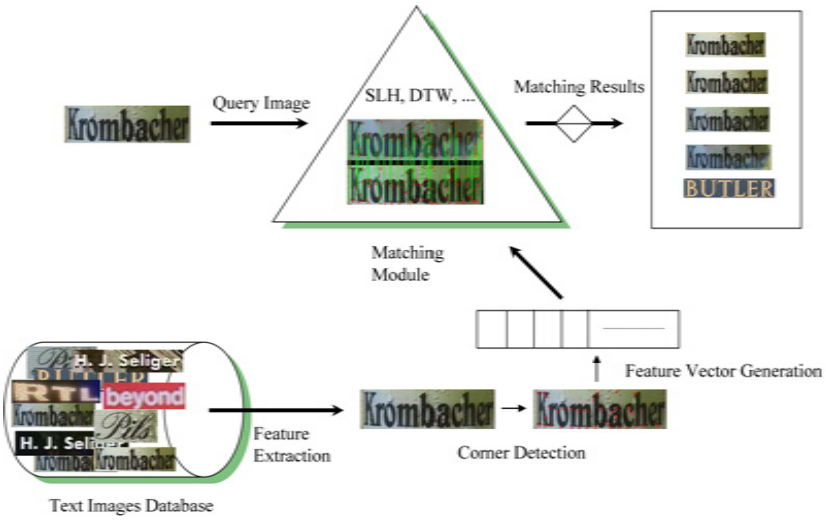


Figure 8.1: The flow chart of the proposed approach for holistic comparison of text images

8.3 Holistic Comparison of Text Images

The problem that is considered in this chapter can be formulated as follows. Let I and J be two different text images. The aim of the algorithm is to numerically define the apparent resemblance or dissimilarity between two given text images I and J. For this purpose, the holistic comparison of two text images is considered. This process (see Algorithm 15) works as follows. First, the shape of each of the text images is defined, by extracting the salient points. Second, the correspondence between the shapes is established by employing an alignment algorithm. Finally, the dissimilarity level is estimated. The flowchart of the proposed system is shown in Figure 8.1.

In the following subsections, the individual steps of the text comparison algorithm will be discussed in detail.

8.3.1 Shape Definition

The shape of a text image is represented by a set of n corner points in two dimensions. Corners are local image features identified by points where variations of intensity in both horizontal (X) and vertical (Y) directions are high, which means that both partial derivatives I_x and I_y of the intensity function $I(x, y)$ are large. In Figure 8.2, the differences between a flat region, an edge and a corner are illustrated.

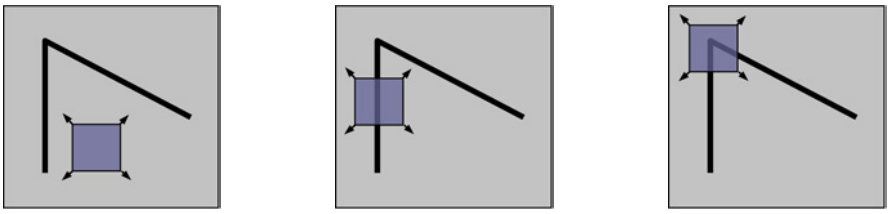
Image corner detection is an important task in various computer vision and image understanding systems. Its application areas include motion tracking,

Algorithm 15: Template of the holistic text comparison algorithm

Input: Two text images: I and J

Output: The (dis)similarity level (Dis)Sim(I, J) between I and J

1 Extract salient points from text images I and J;
2 Compute the feature vectors $FV_i^{(I)}$ for each salient point $i \in I$;
3 Compute the feature vectors $FV_j^{(J)}$ for each salient point $j \in J$;
4 Reconstruct the correspondence between the two set of salient points using the computed features $FV^{(I)}$ and $FV^{(J)}$;
5 Calculate the (dis)similarity level based on the reconstructed correspondence: (Dis)Sim(I, J);



(a) Flat region: No changes in all directions

(b) Edge: No changes in the edge direction (I_x is large, I_y is small)

(c) Corner: Significant changes in all directions

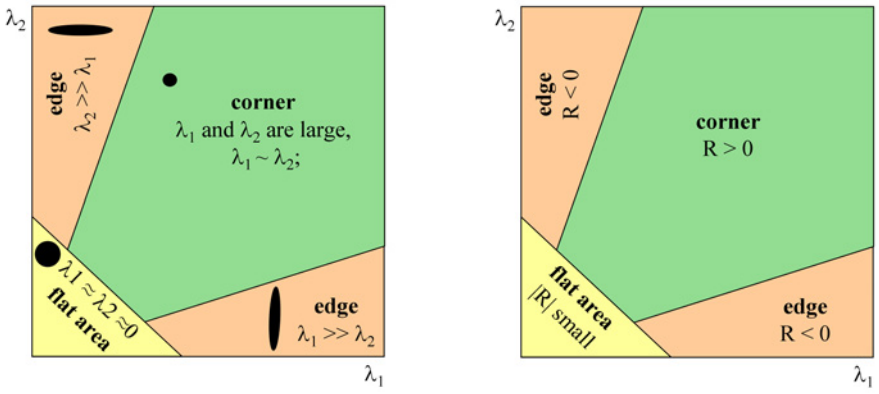
Figure 8.2: Visual differences between a flat region, an edge and a corner

object recognition, stereo matching and image database retrieval.

In Section 8.3.1.1, an algorithm for corner detection initially proposed by Harris and Stephens [HS88] is explained. Then, a modification of the original algorithm is proposed in Section 8.3.1.2, in order to improve the corner detection accuracy.

8.3.1.1 Harris/Stephens Corner Detector

The Harris/Stephens algorithm [HS88] is a well known corner detector, which makes use of the local structure matrix to find the corner positions. The algorithm works as follows. First, the locally averaged moment matrix is evaluated using the image gradients. Then, the eigenvalues of the moment matrix are combined to calculate a parameter called corner strength. As a result, every corner position is associated with a maximum value of corner



(a) Depending on λ_1, λ_2 (b) Depending on the value of R

Figure 8.3: Division of eigenvalue space into distinct feature regions

strength. The local structure matrix C_{str} is defined as follows:

$$C_{str} = w_G(r;\sigma) * M \tag{8.1}$$

where

$$M = \begin{pmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{pmatrix}$$

As Formula 8.1 shows, first the derivatives of the intensity function $I(x, y)$ are calculated for each point of the image. Then, the elements $I_x, I_y, I_x I_y$ of the matrix M are evaluated. Finally, each of the elements is smoothed by applying a Gaussian filter $w_G(r;\sigma)$ with size σ in a $r \times r$ neighborhood. After the smoothing process, the singular value decomposition of the matrix C_{str} is computed. If the evaluated eigenvalues are denoted with λ_1 and λ_2, and the eigenvectors with $v_i = (v_{1i}, v_{2i})'$ for $i = 1, 2$, respectively, then C_{str} can be written as:

$$C_{str} = \begin{pmatrix} v_{11} & v_{12} \\ v_{21} & v_{22} \end{pmatrix} \begin{pmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{pmatrix} \begin{pmatrix} v_{11} & v_{21} \\ v_{12} & v_{22} \end{pmatrix}$$

The values of the eigenvalues λ_1 and λ_2 are geometrically interpreted as follows:

- $C_{str} = 0$, $\lambda_1 = 0$ and $\lambda_2 = 0$: indicates a perfectly uniform area.
- $\lambda_1 > 0$ and $\lambda_2 = 0$: indicates the presence of an edge.
- $\lambda_1 \geq \lambda_2 > 0$: indicates the presence of a corner.

(a) Binarized text image

(b) Colored text image

(c) Grayscale scene image

Figure 8.4: The visualization of the detected corners by small squares

The corner strength R is calculated using the eigenvalues λ_1 and λ_2 as defined in Formula 8.2:

$$R = \det M - k(\text{trace}M)^2 \tag{8.2}$$

where

$$\det M = \lambda_1 \lambda_2 \tag{8.3a}$$
$$\text{trace}M = \lambda_1 + \lambda_2 \tag{8.3b}$$
$$k \in [0.04; 0.06] \tag{8.3c}$$

In Figure 8.3, the eigenvalues space is visualized dependending on the eigenvalues λ_1 and λ_2, and on the value of R. Figure 8.3(b) shows that the corner strength parameter R has: (I) a large value when a corner is located; (II) a negative value with a large magnitude in the case of an edge; (III) a small absolute value for a flat area.

The Harris/Stephens detector identifies as corners all the points which have a corner strength R higher than a preset threshold Th ($R > Th$).

Figure 8.4 illustrates the detected corners using the described Harris/Stephens algorithm for a binary text image, colored text image and a grayscale scene image. Algorithm 16 describes in details the pseudocode of the Harris/Stephens corner detector method.

8.3.1.2 Adaptive Corner Detection

The Harris/Stephens corner detector is widely used because of its high detection and repeatability rate over other corner detection algorithms. However, the use of gradient or high-order derivatives makes the Harris/Stephens algorithm sensitive to noise (e.g. any pixel affected by the salt noise will have a

Algorithm 16: Template of the Harris/Stephens corner detection algorithm

Input: A grayscale image; Gaussian variance (windows typically have a radius of 3 times the standard deviation); the values of constant k and threshold Th

Output: A map where the position of each detected corner is indicated

1 For each pixel (x, y) from the image calculate the autocorrelation matrix M:

$$M = \begin{pmatrix} A & C \\ C & B \end{pmatrix}$$

where: $A = (\frac{\partial I}{\partial x})^2 \otimes w$, $B = (\frac{\partial I}{\partial y})^2 \otimes w$, $C = (\frac{\partial I}{\partial x}\frac{\partial I}{\partial y}) \otimes w$
$\otimes$ is the convolution operator
w is the Gaussian kernel ;

2 Construct the cornerness map by calculating the cornerness measure $C(x, y)$ for each pixel (x, y):
$C(x, y) = \det(M) - k(\text{trace}(M))^2$
$\det(M) = \lambda_1\lambda_2 = AB - C^2$
$\text{trace}(M) = \lambda_1 + \lambda_2 = A + B$;

3 Threshold the interest map by setting all $C(x, y)$ below a threshold Th to zero ;

4 Perform non-maximal suppression to find local maxima ;

5 All non-zero remaining points in the $C(x, y)$ are corners.

large gradient in all directions). Usually, increasing the size of the Gaussian window r ($w_G(r; \sigma)$) will cause a reduction of the relative weight of any pixel affected by noise, which in turn will improve the performance of the corner detector in presence of the noise. However, this solution will further increase the computational demand of the Harris/Stephens algorithm, and additionally it is also not adaptive to the image complexity.

In this context, it is proposed to first threshold the edge (gradient) image before proceeding further with the next steps of the corner detector algorithm. The use of an appropriate threshold will cause the reduction of noise pixels and consequently will meliorate the results of the corner detector. The Otsu algorithm [Ots79] is applied on the edge image to estimate the thresh-

old. Then, all pixels in the edge image, whose values do not exceed the found threshold, are removed from the edge image, before proceeding with the subsequent steps of the corner detector algorithm. The proposed thresholding process will help to eliminate possible noise edge pixels caused by the complex background where the text appears. Moreover, the Otsu threshold is adaptive to the image complexity.

8.3.2 Alignment of Shapes

The next problem that needs to be solved is the matching of the shapes with each other using different alignment algorithms. This problem can be formulated as follows. Let $\{i^{(I)}\}$ ($|i^{(I)}| = m$) and $\{j^{(J)}\}$ ($|j^{(J)}| = n$) be two sets of points that represent the shapes of images I and J respectively. For each point $i \in \{i^{(I)}\}$ ($j \in \{j^{(J)}\}$) let $FV_i^{(I)}$ ($FV_j^{(J)}$) be the set of features associated with this point, which in our case are the 2D cartesian coordinates. The aim of the alignment algorithm is to find the best one-to-one correspondence between $i^{(I)}$ and $j^{(J)}$ on the basis of these features.

According to the type of elements to be associated, there are two classes of correspondence algorithms:

1. *Correlation-based* algorithms; elements to be matched are image patches of a fixed size.

2. *Feature-based* algorithms; elements are represented by feature vectors.

The proposed approach in this chapter belongs to the second category, and Section 8.3.2.1 explains the used elements and their representing features. Furthermore, Sections 8.3.2.2 and 8.3.2.3 describe two different methods (SLH and SCF) to find the best alignment between the elements of two different images. The SLH algorithm employs the properties of the singular value decomposition (SVD) [DUP99] and tries to model the distortion between two patterns as an affine transform. The SCF method employs various geometrical spatial constraints and the rate of similarity between two elements to ensure the most accurate association between two pattern images.

8.3.2.1 Corner Based Feature Estimation

For each salient point, the feature vector $FV()$ is mainly composed of the coordinates x and y of the position of the corner. Furthermore, the including of pixels values (in the case of grayscale (color) image analysis the Y values (and the RGB values)) in a fixed size neighborhood of the current corner as complementary features is also evaluated.

8.3.2.2 The Scott and Longuet-Higgins Algorithm

Scott and Longuet-Higgins [SLH91] have proposed a solution to the problem of associating features of two different patterns. They model the distortion between two patterns as an affine transformation using the properties of the SVD. An affine transformation is a linear transformation between two set of points in the same coordinative space, which in 2-D space is described by the equation:

$$r' = Ar + t \tag{8.4}$$

where t is a 2-D vector that describes the translation, A is an 2×2 matrix which captures the deformation, r and r' are the coordinates of two set of points between which the affine transformation must be recovered. An affine transformation can capture deformation caused by scaling or shear in both directions and rotation.

The SLH algorithm consists of three main steps: (I) evaluation of the proximity matrix; (II) singular value decomposition; and (III) correspondence establishment. In the following, the steps are explained in detail.

1. ***Evaluation of the proximity matrix:*** $G_{m \times n}$
   Each element $g_{i,j} \in G_{m \times n}$ shows the Gaussian weighted distance between $FV_i^{(I)}$ and $FV_j^{(J)}$, which is defined as denoted in Equation 8.5:

   $$g_{i,j} = e^{-d_{i,j}^2/2\sigma^2}, i = 1..m, j = 1..n \tag{8.5}$$

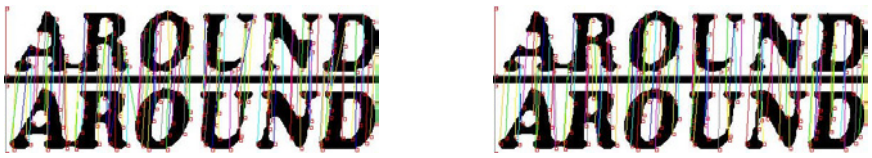
   where $d_{i,j}$ is the distance (e.g. Euclidean distance), between $FV_i^{(I)}$ and $FV_j^{(J)}$. $G_{m \times n}$ is positive definite and its elements $g_{i,j}$ decrease from 1 to 0 with the distance. The value of σ controls the degree of interaction between the two feature vectors: a small value of σ enforces local interactions, while a larger value allows more global interactions.

2. ***Singular value decomposition***
   The matrix $G_{m \times n}$, with $m > n$ can be written using the so-called singular value decomposition of the form:

   $$G = TDU^T \tag{8.6}$$

   where T is an $m \times n$ matrix and U is an $n \times n$ matrix, both of which are matrices with orthogonal columns ($T^T = U^T U = I$). D is an $n \times n$ diagonal matrix whose elements along the diagonal are the positive singular values in descending numerical order. In the case when $m < n$ only the first m columns of U are important.

3. ***Correspondence establishment***
   First, the matrix of singular values D is transformed into a new matrix



(a) SLH-based correspondence (b) SCF-based correspondence

Figure 8.5: The visualization of the established correspondece using different algorithms

E, where all the elements of the diagonal ($d_{i,i} \in D$) are replaced with 1. Then, the following product is computed:

$$P = TEU^T \tag{8.7}$$

The matrix P has the same dimension $m \times n$ as the proximity matrix G and moreover has the crucial property of "amplifying" good pairing while "attenuating" bad ones. Based on this property, it is decided that there exists a correspondence between two elements $FV_i^{(I)}$ and $FV_j^{(J)}$, if and only if the corresponding element $p_{i,j} \in P$ is both the greatest element in the i^{th} row and the greatest element in the j^{th} column.

The SLH algorithm embeds simultaneously the proximity and the exclusion principles. The former is inherited from the nature of the proximity matrix (Step 1), whereas the latter arises from the orthogonality of the matrix P.

According to [SLH91], "the fact that the squares of the elements in each row of P must add up to 1 implies that a given element $FV_i^{(I)}$ cannot be strongly associated with more than one element $FV_j^{(J)}$. The mutual orthogonality of the rows tends to keep different elements from the first vector from becoming closely associated with the same element of the second vector".

The authors have also shown that the SLH algorithm performs well with deformations of different types such as translation, shearing and scaling, and with moderate rotation [SLH91]. They have also suggested different criteria for the choice of the parameter σ.

Figure 8.5(a) illustrates the established alignment between two different text images using the SLH algorithm.

8.3.2.3 Sequential Correspondence Finding

The rationale behind the sequential correspondence finding (SCF) method is that a word is composed of a sequence of consecutive characters (i.e. corner points). In contrast to the SLH algorithm, the SCF tries to align the corners

with each other sequentially. In this way, possible intersections are avoided (i.e. if the i^{th} corner of image I is associated with the j^{th} corner of the image J, the SCF method does not allow to associate any successive corner of the i^{th} corner from the image I with any previous corner of the j^{th} corner from the image J).

The SCF algorithm consists of two main steps. First, spatial geometrical constraints to reduce the size of the search space are employed. For the corner $i \in \{i^{(I)}\}$ at position (x_i, y_i) and corner $j \in \{j^{(J)}\}$ at position (x_j, y_j) it is checked if they fulfill the condition shown in Equation 8.8.

$$\sqrt{(\frac{x_j}{\mathrm{W}(J)} - \frac{x_i}{\mathrm{W}(I)})^2 + (\frac{y_j}{\mathrm{H}(J)} - \frac{y_i}{\mathrm{H}(I)})^2} <= r \tag{8.8}$$

where the functions H() and W() return the height and width of the respective text image, and r is the radius of the area where two corresponding elements can move. Second, if the pair of corners satisfies the spatial condition, the pixel-based dissimilarity between them is measured. The Euclidean metric is employed to evaluate the distance between the grayscale values of pixels in the respective neighborhoods of the two corners.

Two corners $i^{(I)}$ and $j^{(J)}$, represented by the feature vectors ($FV_i^{(I)}$ and $FV_j^{(J)}$) respectively, are associated with each other if and only if they show the lowest dissimilarity level and the dissimilarity level is below a threshold (e.g. $max_{DsL} = 10$).

The obtained alignment using the SCF method for the same text images as in Figure 8.5(a) is illustrated in Figure 8.5(b).

8.3.3 Overall Dissimilarity Derivation

The overall dissimilarity level between two text images I and J (Diss(I,J)) is defined relying on the established correspondence and is estimated as denoted in Equation 8.9.

$$\mathrm{Diss}(I, J) = \frac{\sum_{\forall(i,j)\in C} SSD(FV_i^{(I)}, FV_j^{(J)})}{|C|} * \frac{\mathrm{corners}(I)}{|C|} \tag{8.9}$$

where $C = \{(i, j)\}$ where $i \in \{i^{(I)}\}, j \in \{j^{(J)}\}$, is the set of the established pair correspondences, $|C|$ is the cardinality of the set C, corners(I) returns the total number of corners in image I and SSD denotes the sum-of-squared distances.

8.4 Performance Evaluation

The method was tested on a database of 80 binary images of words (named $TestSet_{Binary}$), which are segmented and binarized using each of the methods

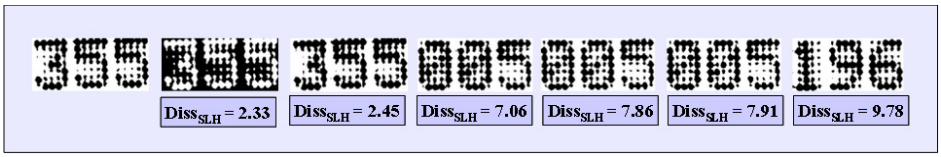


Figure 8.6: The query image and the first six most similar images and the respective dissimilarity levels

presented in [GF05a, GESF04, Ots79]. The database contains 19 classes of text images, while the number of instances in each class varies from three to six. We have selected these text images for evaluations due to the fact that the used OCR engine [Abb] fails partly to successfully recognize the text in them. This is caused by the noise effects introduced by the segmentation and binarization process (resulted from the complex background) or the used fonts.

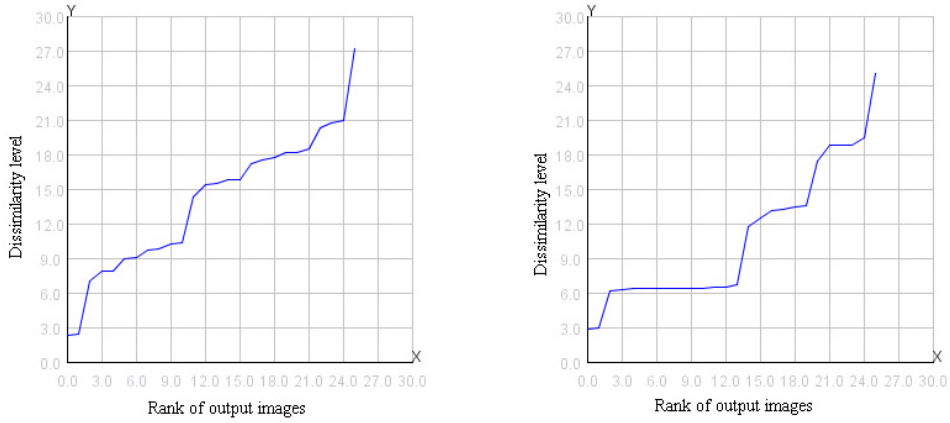
Given a query text image, the proposed comparison method attempts to find the most similar text images in the database. To reduce the number of detailed comparisons as described in Section 8.3, each pair of images undergoes first a simple geometric examination. Similar to the pruning technique described by Manmatha et al. [MHR95], it is checked whether the area and the aspect ratio of the candidate and the query image fulfill some simple geometric conditions. Given the query image I and the database image J, the conditions shown in Equation 8.10 are verified.

$$\frac{1}{\alpha} \leq \frac{\text{area}(J)}{\text{area}(I)} \leq \alpha, \text{ where } \alpha = 1.2 \tag{8.10a}$$

$$\frac{1}{\beta} \leq \frac{\text{aspectRatio}(J)}{\text{aspectRatio}(I)} \leq \beta, \text{ where } \beta = 1.4 \tag{8.10b}$$

Then, the dissimilarity of each pair of images that satisfies the conditions is further evaluated using the proposed holistic comparison approach. For an accurate comparison, it is expected that the most similar text images are listed at the beginning, which means that the algorithm has granted these text images a lower dissimilarity rate compared to the others. During the experiments, both the SLH and SCF correspondence algorithms were evaluated and their performances are reported.

The system was first tested with 80 different queries. Each of the text images is chosen as a query image, and it is compared against all the images in the database except itself. If only the image ranked first (i.e. with the lowest dissimilarity) is considered, then the proposed holistic comparison method achieved an overall accuracy of 92.5% when the SCF algorithm was used to associate the corners with each other. The application of the SLH method has attained a better accuracy of 96.3%.

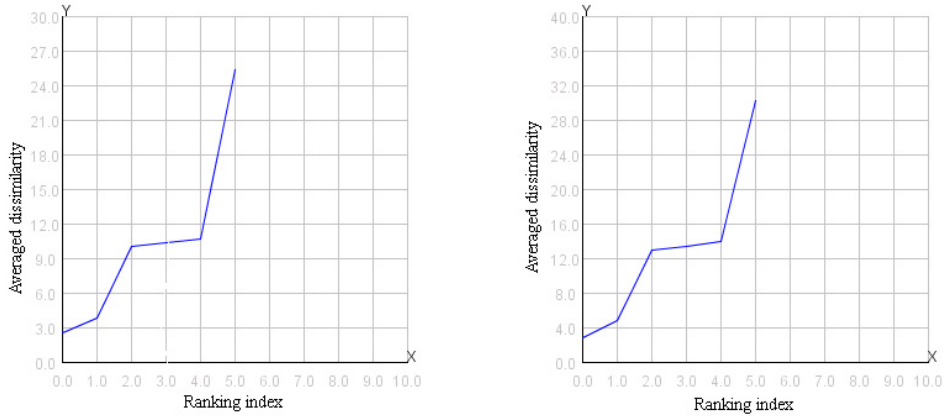


(a) Using SLH (b) Using SCF

Figure 8.7: The dissimilarities trend for the output images

Figure 8.6 illustrates a sample of the output of our approach. The first image presents the query image that contains the number "355", whereas the others represent the first six images sorted in the increasing order based on the respective dissimilarity levels. Figure 8.7 describes the distribution of the dissimilarity levels for all text images which remained after the pruning process. It must be pointed out that only two other text images in the database contain the same textual information as the query image. From the Figure 8.6 it is clear that both of these images have a significantly lower dissimilarity level compared to the other output images. Furthermore, Figure 8.7(a) shows that beginning from the second rank (x=1.0) the level of SLH-dissimilarities increases significantly. Figure 8.7(b) demonstrates that even when the SCF method is applied, a sharp change is observed in the second rank (x=1.0). However, the trends of dissimilarities lead to the conclusion that the classes of the same text images are better separated when the SLH algorithm is employed.

Furthermore, the dissimilarity levels were averaged along the first six ranks (x=0.0-5.0) over all queries, whose answer on the first rank is correct. Figure 8.8(a) shows the obtained curve with the averaged dissimilarities in the case when the SLH algorithm is employed, whereas in Figure 8.8(b) the trend in the case of the SCF algorithm is illustrated. Both figures show that from the second (x=1.0) to the fifth (x=4.0) rank a slightly increase in dissimilarity level is present, whereas from the fifth (x=4.0) to the sixth (x=5.0) rank the curve shows a strong change. This phenomenon is explained by the fact that only three different text images for the same text are present in most of the



(a) Using SLH (b) Using SCF

Figure 8.8: The averaged dissimilarities along five first ranks for the positive answered queries

cases of, while only three classes contain six prototypes of the same text. This is an encouraging indication that the binary images of the same text are well separated from the rest of the images.

Thus, the experiments have shown that given a binary text image (i.e. image of a word) as a query, it is possible to successfully find the visually similar images in the database (in the best case they contain the same ASCII text), although in some of them the OCR engine has failed to recognize them correctly.

8.5 Summary

In this chapter, a novel method to compare two given text images was proposed. First the salient points in each of the images were estimated using a slightly modified version of the Harris/Stephens corner detector. Second, the best mapping is determined applying one of the two proposed algorithms, namely SLH and SCF. The SLH algorithm models the correspondence between the set of salient points as an affine transformation and uses SVD properties to find the corresponding pairs. The SCF is based on spatial geometrical constraints and different distance measures to define the right mapping. Then, the similarity between two given images is defined based on the established correspondence. Experiments on a set of binary text images

have shown promising results using the proposed methodology.

Chapter 9

Semantic Browsing for Content Based Image Retrieval

One picture is worth a thousand words.

- Fred R. Barnard -

9.1 Introduction

Content-based image retrieval (CBIR) has been a very active research area since the 1990s, although first attempts which are based on annotated text date back to the 1970s. In general, content-based image retrieval techniques use the visual content of an image described by the spatial layout and different low level features such as color, shape and texture to index an image. In Figure 9.1, the diagram of a typical content-based image retrieval system is shown. Commonly, the visual content of the images is extracted off-line and is usually represented by multi-dimensional feature vectors. The set of extracted feature vectors create the so called feature database. To retrieve the required images, the user interacts with the system by providing query examples or sketched figures. Then, the system transforms the input image into its internal representation as a feature vector. The similarity/distances between the feature vector of the input example and the feature vectors of the images in the database are calculated and retrieval is performed through an indexing scheme, which provides an efficient way to search in the multi-dimensional feature database.

Recently, relevance feedback has been introduced into content-based mul-

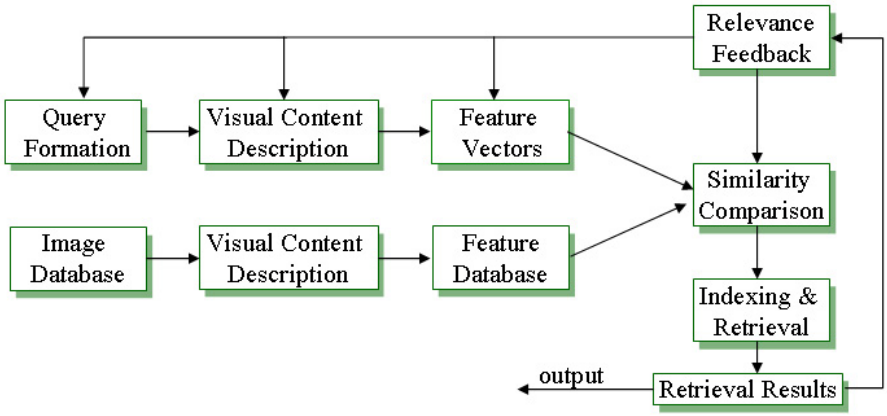


Figure 9.1: The diagram of a content-based image retrieval system

timedia retrieval and especially CBIR as a technique for overcoming many of the problems faced in fully automatic CBIR systems. The idea behind relevance feedback techniques is to involve the user in the retrieval process asking her/him to tune different system parameters. Relevance feedback algorithms improve the performance of retrieval systems and deal to a certain degree with the problem of image polisemy [HR04a], but their successful application is sometimes difficult for a common user [ZH03] because they imply that the user possesses the ability to steer the retrieval process. In addition, their tendency to converge to a local optimum limits their usage. Local optimum in this context means selecting a subset of images, which fulfills partially the user's requirements. Moreover, positive feedback fails, if the initially obtained results are irrelevant to the user. Thus, the employment of negative examples has become the focus of research. According to Mueller et al. [MMS+00], negative feedback is useful due to the options it offers the user to navigate through a database of images, to move in feature space and find the target images. Lew et al. [LSDJ06] conclude that researchers should address as much as possible the requirements of the user, who may be more interested in exploring instead of searching the media. The same authors have noted that decision makers need to explore an area to acquire valuable insights before taking a decision. Thus, systems which stress the exploration aspect are strongly encouraged.

It should be emphasized that the need for browsing as well as retrieval methods increases considerably when large image data sets are analyzed. Inspired by the comments regarding the limitations of the traditional query-by-example systems and relevance feedback principles, the interest of the user to explore instead of to search and the success of information visualization techniques, a novel visualization technique for exploring the relationships be-

tween images in a data set, namely the Image Proximity Matrix (IPM) is proposed, in this chapter. Furthermore, two dimension reduction visualization techniques are also introduced as complementary views to the IPM.

This chapter is structured as follows. Section 9.2 will give an overview of the related image retrieval and browsing work. Section 9.3 will describe the proposed browsing environment. In Section 9.4, experimental results will be presented. Section 9.5 will conclude this chapter.

9.2 Previous Work

The area of the visualizing and browsing the query results has not been researched extensively, despite the increasing number of multimedia data and the successful employment of information visualization techniques in other fields. In the following, several CBIR approaches and browsing methods are reviewed. However, for an extended overview of the state of the art of CBIR, the reader is referred to several publications [VT00, SWS+00, LSDJ06].

The QBIC system [FSN+95] uses color, texture and shape features to search for the images which satisfy the user query. The images are visualized sequentially and the best matches are presented in decreasing similarity order.

In Smith and Chang's approach [SC97], each image is automatically decomposed into regions of equally dominant colors. For each region, feature properties and spatial properties are estimated for a given query, which consists of finding the images that contain the most similar arrangements of similar regions.

The idea to navigate an image database was first introduced by Rubner et al. [RGT97], where multi-dimensional scaling [MKB79] was combined with the Earth Mover's Distance [RTG98] for color based image retrieval.

Rodden et al. [RBSW01] have found that the arrangement of the images according to their mutual similarity is more useful to the users compared to their presentation in a default order on the screen, "especially when they wish to narrow down their requirement to a particular subset".

Torres et al. [TSMR03] have presented two visualization techniques based on Spiral and Concentric Rings to explore query results. They tend to focus the user on both the query image and the retrieved results.

Carey et al. [CHR03] have included several visualization front-ends (e.g. Sammon view, Dendro Map and Radial Interactive Visualization) in their text document search engine to enable navigation through the documents.

Santini et al. [SGJ01, SWS+00] have combined query-based search with browsing. The images are displayed on the screen, while trying to preserve their mutual distances. Feedback techniques are also included to increase the performance of the system.

Heesch and Rueger [HR04a] describe a technique to support content-based

image search. The main idea behind the NN^k Network structure is to represent each image as a vertex in a directed graph. An edge is established between two images if there exists at least one feature combination for which the respective images are the nearest neighbors. In [HR04b], the NN^k Network system is further extended and the weight of the similarity function is updated by employing relevance feedback techniques. The initial feature weighting is avoided by means of repeating the retrieval process for different feature combinations and deriving initial weights from the top ranked results. In [HR05], a theoretical explanation of the distribution of the images belonging to the same semantic clusters across the NN^k Network structure is given.

Pathfinder networks are used by Chen et al. [CGR00] for content-based image retrieval. The similarity between images is used as the edge weight. Color, layout and texture are estimated to describe the images.

Nguyen and Worring [NW04] use the IsoMap algorithm of [TdSL00] to preserve the relationships between images in the high dimensional space and to reduce the overlapping of the images in the final 2-dimensional visualization. Snoek et al. [SWvG+05] have further extended the method to facilitate semantic browsing of news videos.

Cai et al. [CHL+04] use spectral clustering techniques to cluster image search results in the WWW based on visual, textual and link information. The representative images of each cluster are shown in different rows. In [LXT+04], the search results are fitted into a grid view based on their similarity, while trying to avoid image overlapping. Furthermore, the Fisheye view is applied to help the user explore both local and global information at the same time.

A 3D visualization method similar to treemaps has been proposed by Chiu et al. [CGL+05]. Based on a city metaphor, the directories are mapped to city layouts and each multimedia document is represented by a building. The sides of the building are used to represent the visual summaries of the respective media. A swooping technique enables the navigation between high overviews and detail views.

9.3 Semantic Browsing Environment

In contrast to the work reviewed in the previous section, our approach offers a novel solution to visualize the entire structure of a collection of images based on a given set of image descriptors and a proximity measure. The proposed reordable Image Proximity Matrix visualization lends itself particularly well for exploring relationships in image collections by overcoming the problem of overlapping images encountered in most of the previous approaches. Furthermore, the user can interactively explore the present clusters in contrast,

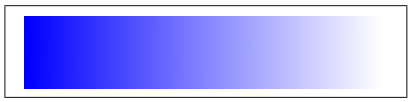


Figure 9.2: The blue-white spectrum

to for example, the work of [CHL+04], where the number of clusters must be known beforehand. The interaction of the reordable matrix view with the multi-dimensional scaling [MKB79] or Sammon Mapping [MKB79] aid the user to explore the present image clusters and their relationships.

In the following, first the Image Proximity Matrix view is described. Then, two other visualization methods, namely multi-dimensional scaling and Sammon mapping are presented. Furthermore, the interactive aspects among the different methods and their benefits are also briefly discussed.

9.3.1 Image Proximity Matrix View

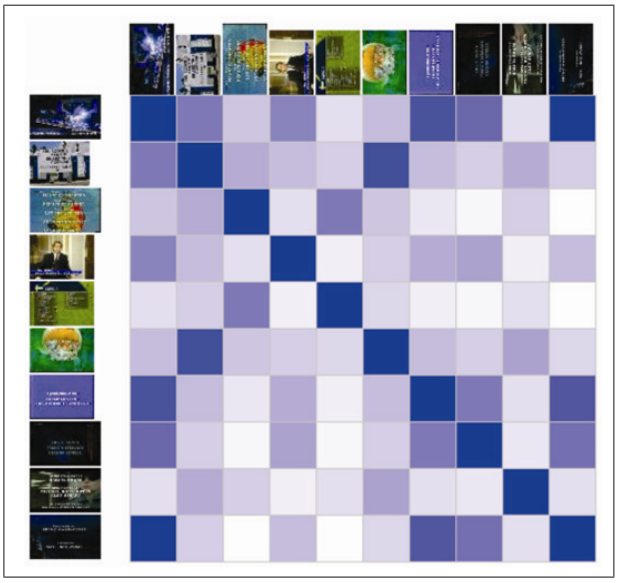
Matrix-based visualizations are useful for exploring relationships between related records in a data set. They have been used for a long time in many fields [QWF04, QWF05] for the visualization of graphs or similarity matrices. In this section, their use for interactive browsing and exploratory analysis of image data sets will be explored.

The proposed Image Proximity Matrix (IPM) method consists of the following two main steps. First, the (dis-)similarity levels between each pair of images are estimated. Second, the relationships between images in a collection are represented as colored points on a plot. Thus, the proximity matrix is transformed into a matrix of colors. For this purpose, a color spectrum is needed to transform numbers into colors. Throughout this paragraph, the blue-white spectrum is used corresponding to the segment $[0, 1]$ of numerical values. The blue-white color spectrum is defined by the two border colors blue and white, respectively and the transition colors as shown in Figure 9.2. The $[0, 1]$ segment is divided in as many small segments as nuances of colors are used (256 colors are used) in our case, and the transformation of a *value* into the corresponding color is done according to Equation 9.1.

$$\text{Color} = \text{ColorMap}(value * 255) \tag{9.1}$$

Figure 9.3(a) represents the Image Proximity Matrix view corresponding to a small set of images. The proximity matrix is estimated by means of analyzing the position and the quantity of the text that appear in those images. The darker the intensity of the blue color the more similar the respective images are according to the given proximity matrix.

Ordering of the Similarity Matrix. The columns and rows of the similarity matrix must be reorganized via manual permutations or by al-



(a) The unordered IPM view

(b) The IPM view ordered using the TSP-based method

Figure 9.3: The visualization of a small collection of images using the Image Proximity Matrix view. The present clusters are highlighted in red

gorithms for the automatic generation of the most suitable permutations, since the color visualization alone does not allow the easy detection of cluster structures in the data. The manual interaction of the user with the matrix is

Algorithm 17: Template of the VAT ordering algorithm [BH02]

Input: The dissimilarity matrix $R_{n\times n}$ corresponding to the set of images $O = \{o_1, ..., o_n\}$

Output: The ordered dissimilarity matrix $\widetilde{R}$

1 Set $K = \{1, 2, ..., n\}$; $I = J = \phi$; $P[0] = (0, ..., 0)$;
2 Select $(i, j) \in \arg\max_{p\in K, q\in K} R_{pq}$;
3 Set $P(1) = i; I = \{i\};$ and $J = K - \{i\}$;
4 **for** $r = 2$ *to* n **do**
5 Select $(i, j) \in \arg\min_{p\in I, q\in J} R_{pq}$;
6 Set $P(r) = j$; Replace $I = I \bigcup \{j\};$ and $J = J - \{j\}$;
7 $r = r + 1$;
8 **end**
9 Obtain the ordered dissimilarity matrix $\widetilde{R}$ using the ordering array P as: $\widetilde{R}_{ij} = R_{P(i)P(j)}$, for $1 \leq i, j \leq n$.

described in Section 9.3.4, whereas this section will concentrate on automatic ordering methods. The problem of automatically computing the "optimal" permutations for the image (dis-)similarity matrix is a complex problem addressed by many researchers.

The first solution that was used to solve this problem is the Visual Assessment Tendency (VAT) algorithm initially proposed by Bezdek and Hathaway [BH02]. The VAT algorithm uses a variant of Prim's algorithm for finding a minimal spanning tree (MST) of a weighted graph. However, the main difference between VAT and MST is that the VAT algorithm does not represent the MST, but it only finds the order in which the vertices are added as it is grown. Furthermore, in contrast to MST, VAT defines the initial vertex depending on the maximum edge weight in the graph. The pseudocode of the VAT algorithm is presented in Algorithm 17. It takes as input the dissimilarity matrix R and gives as output the matrix $\widetilde{R}$ whose rows and columns are ordered based on their similarities. Note that the array P is used to store the permuted indices of the n images, whereas the dissimilarity matrix can be obtained from the similarity matrix S applying the equation: $R_{ij} = S_{max} - S_{ij}$, where S_{max} denotes the largest similarity value.

The second solution that is considered to order the proximity matrix is based on a heuristic that makes use of the well known travelling salesman problem (TSP) [JM97]. This method has been proposed by Qeli et al. [QWF05] and considers the ordering problem as an optimization problem. The similarity that one dimensional ordering has with the TSP is used to

solve the problem.

The TSP is defined as follows: A salesman has to visit N cities. Each city is visited only once and the final city is the same as the starting city. In which order should the salesman visit the cities such that the distance traveled is minimized? In contrast to the TSP, the path traversed here is not a cycle. Let $S_{n\times n}$ represent the similarity matrix with dimensions $n \times n$ and let $D_{n\times n}$ be the Euclidean distance matrix obtained when considering the columns of S as vectors in the space R^n. The goal is to find the permutation $\pi = \{\pi_1, ..., \pi_n\}$, which minimizes the value $\sum_{i=1}^{n} d(\pi_i, \pi_{i+1})$. To optimize this sum, a range of techniques can be used from local search to more sophisticated techniques such as genetic programming and simulated annealing.

From our experience, the VAT algorithm does not perform very well for large data sets. The TSP based heuristics perform well and are thus employed to reorder the IPM interactively. Considering that the reordering of the IPM is used in an interactive context, it can be traded off some of the quality of the reordering for obtaining results in a timely manner. Figure 9.3(b) illustrates the corresponding ordered IPM view of the matrix in Figure 9.3(a) using the *next best* criteria for the optimization. In contrast to the IPM view in Figure 9.3(a) where the detection of clusters is not so easy, in Figure 9.3(b) the user can easily distinguish four possible clusters of "similar" images.

9.3.2 Classical Multidimensional Scaling View

Multi-dimensional scaling (MDS) [MKB79] is concerned with the construction of a configuration of n points in Euclidean space using information about the distances between these points. MDS is often used to project data non-linearly from a high dimensional space to a low dimensional one, usually a 2D or 3D space. The purpose of MDS is that the distances between points in the lower-dimensional space approximate the distances between points in the higher-dimensional space best. MDS allows the use of similarity/dissimilarity measures instead of strict distances and thus enables to flexibly view the relationships between data items.

There are two large groups of methods for projection of high dimensional data based on interpoint distances: *Metric MDS* and *Non-metric MDS* approaches. The difference between the two is that metric MDS assumes that there is a true configuration of points in the low dimensional space with the specified interpoint distances in high dimensional space, whereas non-metric MDS assumes a less rigid relationship between the interpoint distances in low and high dimensional space. Thus, non-metric MDS considers only the rank order between object distances.

The classical algorithm for metric multi-dimensional scaling [MKB79] has been modified in order to ensure good response times with a large number of

Figure 9.4: The visualization of a small collection of images using Classical Multi-Dimensional Scaling

images. Let $\{I_1, ..., I_n\}$ represent a set of images and $\{FV_1, ..., FV_n\}$ the representative selected features. Then, the implemented MDS algorithm consists of the steps listed in Algorithm 18.

The MDS method is also used by Carey et al. [CHR03] for visualization and browsing purposes. Although it preserves the real distances along pairs of images, it has the drawback that during the projection different parts of images may overlap.

In Figure 9.4, a screen shot of the MDS view for the same set of images and the same proximity matrix as in Figure 9.3 is shown. Even from Figure 9.4 the user can easily distinguish four different clusters; however, the further analysis of the individual clusters is difficult due to the overlapping images.

9.3.3 Sammon View

The Sammon Mapping [MKB79] is a non-metric MDS technique, which in contrast to classical MDS preserves better small distances rather than large ones.

Figure 9.5 illustrates the 3D plot of the same images and the same proximity matrix as in Figure 9.3, but using the Sammon mapping technique. Five clusters of images can be distinguished in this plot, but the Sammon Mapping does also not avoid the overlapping of the images during their projection in the 2D (or 3D) space.

9.3.4 Interaction and Visual Pattern Exploration

To help the user during the exploration process, all visualization methods, MDS, Sammon Mapping and the IPM View, have been made interactive.

Algorithm 18: The Classical MDS-based algorithm for exploring images

Input: The images $\{I_1, ..., I_n\}$ and the features $\{FV_1, ..., FV_n\}$

Output: The thumbnail images projected in 2D or 3D space

1 Compute the matrix $D_{n\times n}$, where each element d_{ij} of this matrix shows the dissimilarity between I_i and I_j ;

2 Construct the matrix $A_{n\times n}$ where: $a_{ij} = -\frac{1}{2}d^2_{i,j}$;

3 Construct the matrix $B_{n\times n}$ where $b_{ij} = a_{ij} - \overline{a_{i.}} - \overline{a_{.j}} + \overline{a_{..}}$ where $\overline{a_{i.}}$, $\overline{a_{.j}}$ and $\overline{a_{..}}$ are the mean of row i, the mean of column j and the overall mean, respectively;

4 Compute the k largest eigenvalues λ_1, λ_2,..., λ_k where k is the space dimension ($k = 2$ or $k = 3$);

5 Get the corresponding k eigenvectors v_1,... , v_k and normalize them by $v'_i \cdot v_i = \lambda_i$;

6 Output the normalized eigenvectors as a solution of MDS;

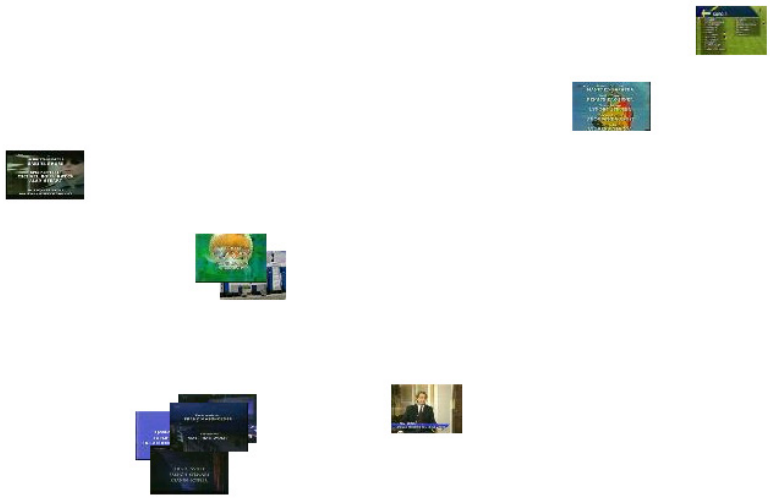


Figure 9.5: The visualization of a small collection of images using the Sammon mapping

All visualizations allow zooming in and out, translation and selection. Our implementation of the IPM method allows the user to swap columns or rows of matrices to find the more satisfactory clusters of images as well as to

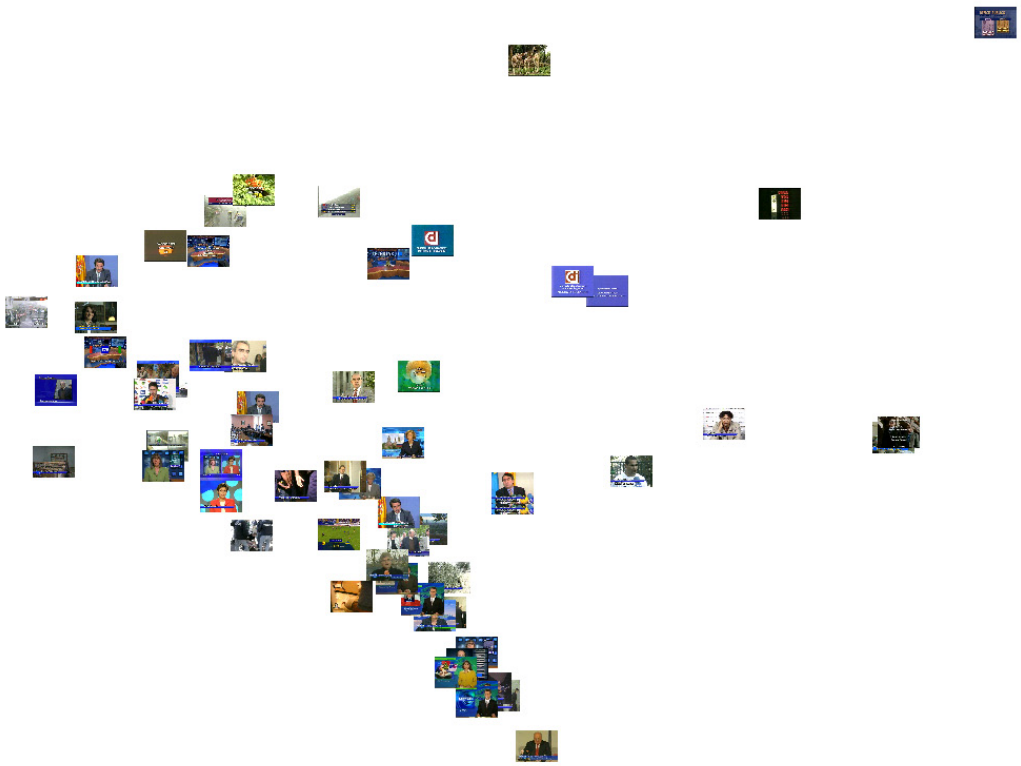


Figure 9.6: The Spatial Correspondence Visualization of the collection of images $TestSet_{CBIR}$ using the Classical MDS mapping method

apply the VAT algorithm or the TSP heuristic based method for automatic ordering. The IPM view is the master view, and the interaction with MDS or Sammon Mapping is the subordinate view.

This kind of interaction can help the user to discover in more detail the clusters of images that are more interesting to him or her.

9.4 Visualization Examples

During the experiments, two data sets of images were used: $TestSet_{CBIR}$ which contains 71 images with artificial texts appearing in different positions; and $TestSet_{Binary}$ which consists of 80 binary text images. The last test set is the same used in Chapter 8 to evaluate the performance of the holistic comparison methods.

The proximity matrix for the visualization of the images from $TestSet_{CBIR}$ is generated using semantic features estimated from the respective images. First, the present texts in images of the $TestSet_{CBIR}$ are localized using the

Figure 9.7: The Spatial Correspondence Visualization of the collection of images $TestSet_{CBIR}$ using the Sammon mapping method

UTDL-Texture algorithm which was introduced in Section 3.3.3. The position of the localized text in respective images is used to evaluate the spatial correspondence, which is then used to generate the proximity matrix. The resulting proximity matrix is given as input to the three different visualization methods. In the "Spatial Correspondence Visualization" it is expected that the images where texts appear in similar positions will be close to each other. Figure 9.6 shows the MDS plot of the $TestSet_{CBIR}$ in the 3-dimensional space; Figure 9.7 illustrates the Sammon mapping view, whereas the IPM view is shown in Figure 9.8. Higher degrees of similarities are represented by darker blue colors in the IPM view. From the three visualizations it is clear that the clusters can be distinguished easier using the IPM view, however the MDS plot and the Sammon mapping are also very useful and complement the IPM view. From Figure 9.7 it can be seen that the text in the highlighted images appear in a similar position, while the respective three images are plotted close to each other.

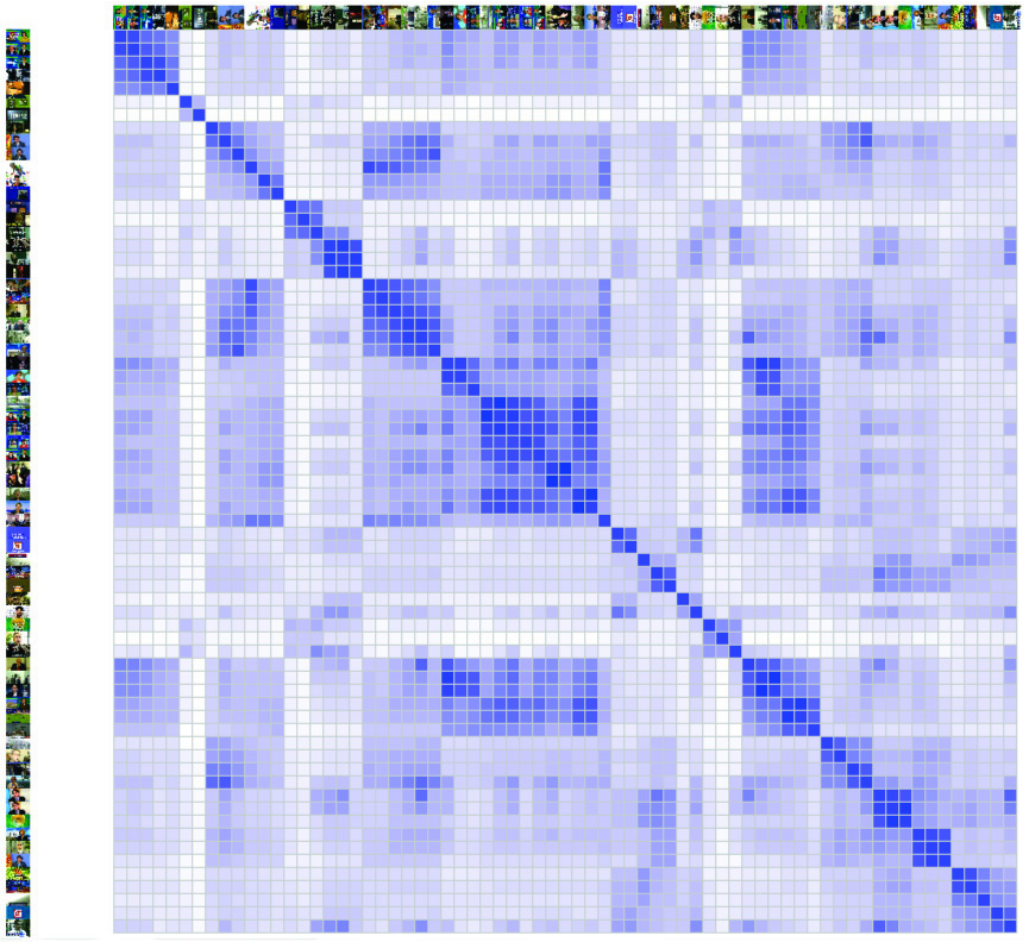


Figure 9.8: The Spatial Correspondence Visualization of the collection of images $TestSet_{CBIR}$ using the Image Proximity Matrix mapping method. The ordering of the matrix is done using the TSP-Heuristic method

The images of $TestSet_{Binary}$ are visualized based on their visual similarity and the proximity matrix contains information about mutual holistic comparisons between text images. After evaluating the salient points in each of the word images the SLH algorithm is employed to align the shapes. The SLH algorithm models the alignment as an affine transformation and makes use of the SVD properties to find the most appropriate correspondence. Finally, the dissimilarity level is evaluated based on the estimated correspondence. More details can be found in Chapter 8. This kind of visualization is named as "Visual Similarity Visualization". It is again expected that the word images that are similar to each other will appear also close to each other. The MDS plot and Sammon Mapping are presented in Figure 9.9 and 9.10, respectively. In Figure 9.11 the IPM view after employing the TSP heuristic to order the

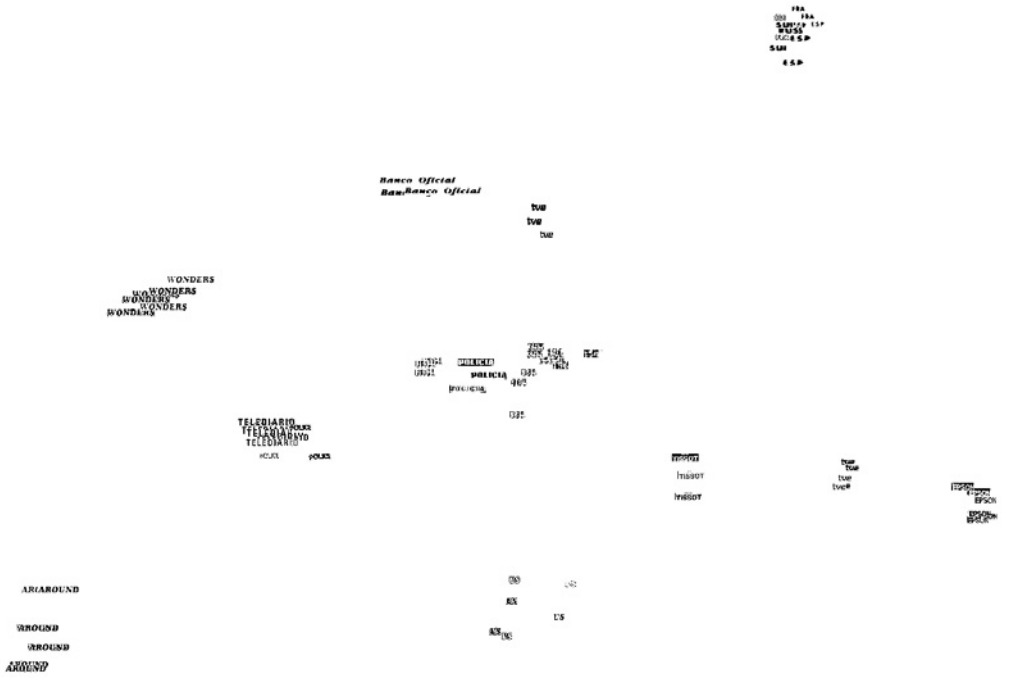


Figure 9.9: The Visual Similarity Visualization of the collection of binary images using the Classical MDS method as described in Algorithm 18

columns i.e. rows on the raw distances, is shown. From all three figures, the MDS plot gives the more clear understanding about the structure of the $TestSet_{Binary}$. The IPM view is very confusing and it is very difficult to identify different possible patterns present in this collection. This is caused by the fact that the proximity matrix is very noisy. For this purpose, in analogy with the noise reduction via principal component analysis (PCA), the proximity matrix is transformed in a similar way to PCA in order to reduce the noise before proceeding further with its visualization. Instead of visualizing the noisy proximity matrix directly, the approximated distance matrix from the MDS representation is first estimated. Algorithm 18 considers only the two or three largest eigenvalues during the estimation of lower dimensional projection. By ignoring the smaller eigenvalues and the corresponding eigenvectors, the noise effect in the original proximity matrix is also reduced, achieving the same effect of PCA in signal analysis. Whereas PCA is applied to the original data matrix, our method works with the proximity matrix in a similar fashion. In Figure 9.12 the IPM view is shown after the transformation and the ordering of columns i.e. rows. It is obvious that the new IPM view is much clearer than the first one and that the clusters of text images present in $TestSet_{Binary}$, which are highlighted in red, are easily distinguishable. In addition, from the IPM view it is also possible to observe

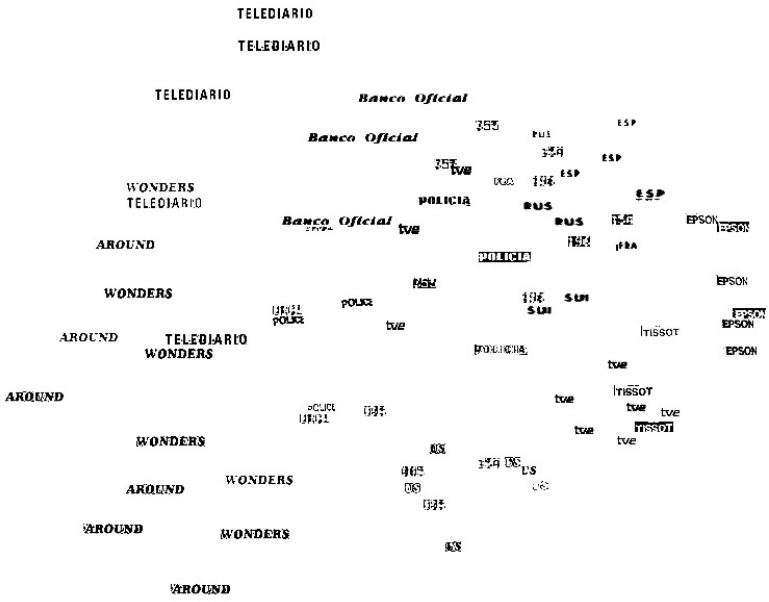


Figure 9.10: The Visual Similarity Visualization of the collection of binary images using the Sammon mapping method. Sammon mapping in contrast to classical MDS preserves smaller distances better

not only the within-clusters relationships, but also the between-clusters relationships. In this context, the set of images, highlighted in green, are similar to the cluster of images bordered by a black line.

During the previous illustrations, the proximity matrices were evaluated based on the features related to the present textual information. However, global image descriptors, namely color histograms of the whole image or different texture feature [DKN04], can also be included in the system.

The used algorithms to estimate the features were separately evaluated in the previous chapters. The evaluation of the IPM visualization technique and its combination with MDS and Sammon Mapping implies the feedback of the users in a large field study, which could not be performed as part of this thesis.

Since the space available for the visualization of the images is limited and fixed, the number of images that can be displayed is also limited. However, in

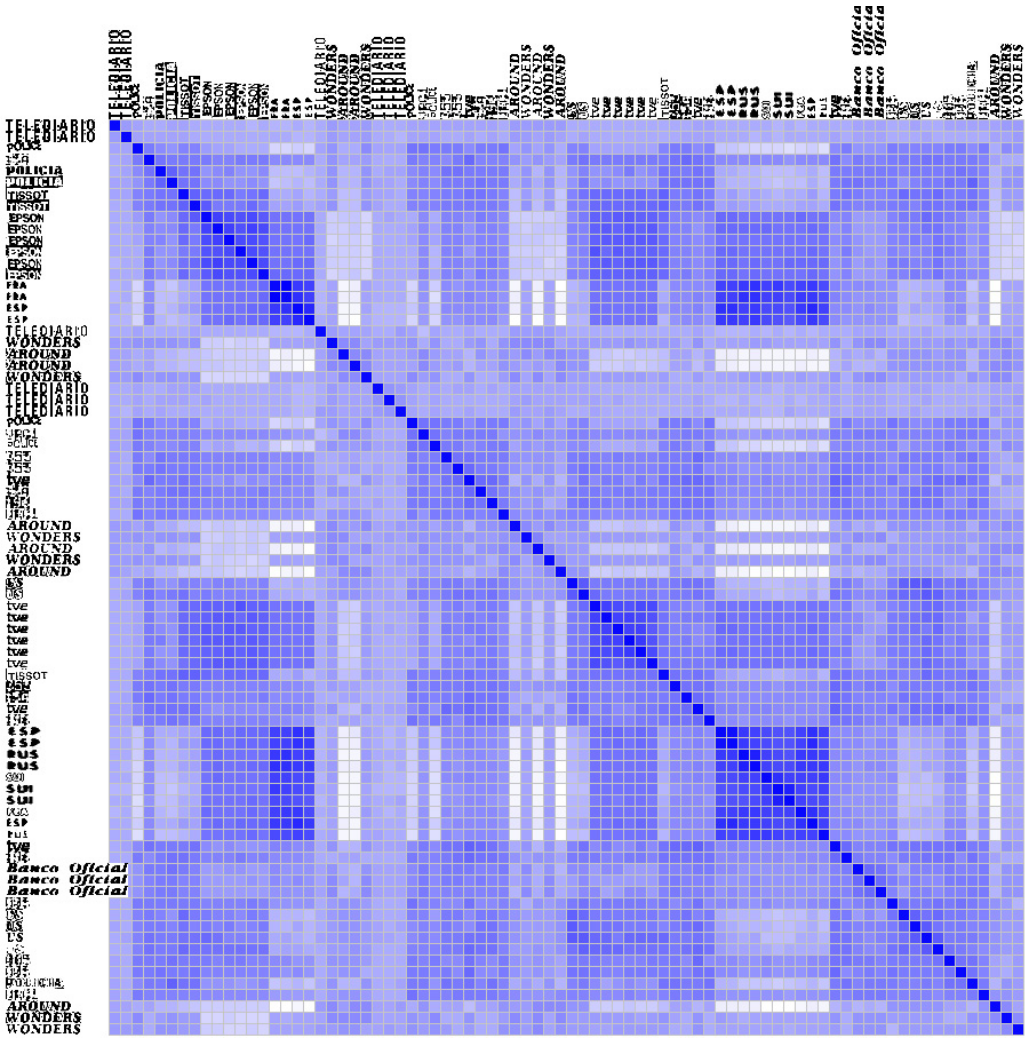


Figure 9.11: The ordered Visual Similarity Visualization of the raw proximity matrix of a collection of binary images. The ordering of the matrix is done using the TSP-based method

the case of the visualization of large data sets of images, *distortion techniques* can be used. This issue remains subject of further work.

9.5 Summary

In this chapter, several visualization techniques were proposed to facilitate the interactive exploratory analysis of data sets of images and assist the user during semantic search. The main idea of the proposed Image Proximity Matrix (IPM) visualization technique is to display the matrix of proximities

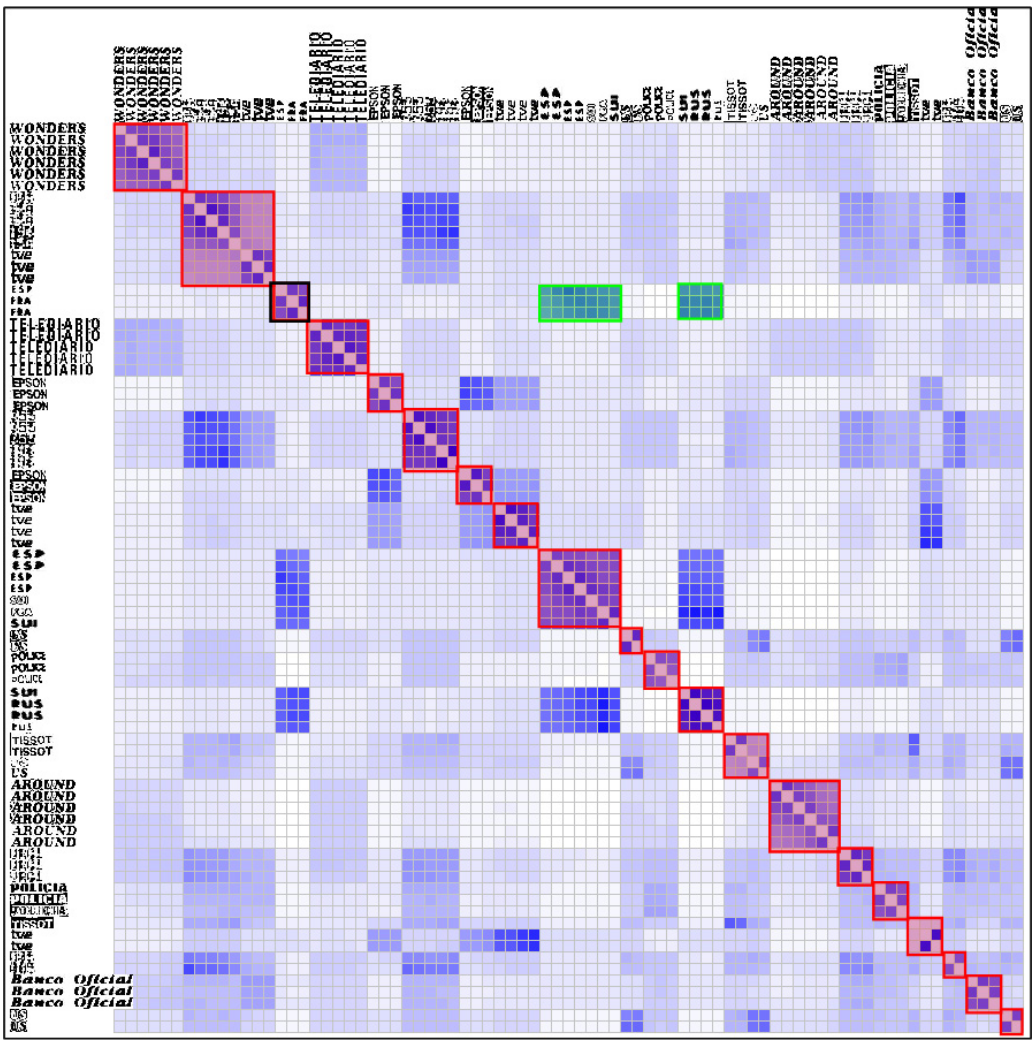


Figure 9.12: The ordered Visual Similarity Visualization of the proximity matrix of a collection of binary images obtained after MDS projection. The preprocessing of the proximity matrix reduces the noise effect. The ordering of the matrix is done using the TSP-based method

between the images in a tabular form. It provides a means to explore the visual patterns in a set of images. The automatic ordering of the Image Proximity Matrix makes an easy exploration of the image clusters possible. Furthermore, the IPM method is more suitable for exploring the relationships in sets of images as compared to MDS or Sammon Mapping. The visualization methods are synchronized with each other to allow the combined interaction with the user and offers the possibility to project certain clusters of images in 2D or 3D space using the proposed dimension reduction techniques.

Chapter 10

Conclusions and Future Work

Have no fear of perfection - you'll never reach it.

- Salvador Dalí -

10.1 Conclusions

Text is a very useful semantic media clue whose successful extraction from images and videos is an important part of a comprehensive content-based indexing and retrieval system. Text extraction in images/videos includes several tasks such as (I) Text Detection; (II) Text Localization; (III) Text Tracking; (IV) Text Segmentation and Binarization; and (V) Character Recognition. Therefore, several methods for the extraction and analysis of textual information in images and videos were proposed and presented in this thesis. The major problems and their solutions discussed in this work are summarized below.

Text detection and localization were the focus of the Chapters 3 and 4. Whereas in Chapter 3 the problem of detecting/localizing the text in still images was investigated, in Chapter 4 the same problem in the context of videos was covered. Specifically, three methods for text detection and localization in images were proposed in Chapter 3. For the proposed UTDL-Texture method, an overall word-based recall rate of 95.15% and precision rate of 93.47% were obtained for Latin text. For Arabian text a text box-based recall of 79.0% and precision of 96.0% were reported, and for the Chinese script an overall recall and precision of 96.7% and 86.3%, respectively were achieved. The detection/localization of text in videos was carried out based

on the application of a fuzzy clustering ensemble, which employs the temporal information present in a video by means of fusing the information obtained from different frames where the same text appear. An overall recall rate of 92.04% and a precision of 96.71% were achieved on real TV material of 10.92 minutes and 16363 frames of which 3905 contain text.

In Chapter 5, an algorithm to track the moving text in videos was proposed. It operates directly in the compressed domain and employs MPEG motion vector information to predict the position of text in the next frame. The combination of the UTDL-Texture method with the tracking algorithm has achieved an overall recall of 88.1% and a precision of 92.6% when evaluated for 8 video sequences with scrolling text.

Two unsupervised text segmentation techniques were presented in Chapter 6 to enable the separation of text pixels from the background. The first employs the k-means algorithm to classify pixels into two clusters on the basis of different features. The second method is based on a fuzzy C-means algorithm, and in contrast to the first method, the number of clusters is adaptive depending on the input image. Furthermore, a strategy to choose the correct text cluster by analyzing the text-like characteristics of the present components was also proposed. Both methods were evaluated on two test sets of images with a total number of 441 words and 2684 Latin characters. The character/word recognition rates were used to evaluate our methods. The best performance was achieved by the second method with an overall word recognition rate and character recognition rate of 78.0% and 90.4%, respectively.

Additional problems encountered during this research such as the recognition of the script and the comparison of text images were the subject of Chapters 7 and 8, respectively. The identification of the script before proceeding with the segmentation (i.e. recognition) of the text is necessary to employ the adequate segmentation approach (i.e. OCR engine). Consequently, a supervised learning approach was proposed to distinguish between a Latin and Chinese script based on a set of low-level features. The Latin script was identified with an accuracy of 85.3%, while the Chinese script was identified with an higher accuracy of 89.0%. Furthermore, the holistic comparison of text images was treated in Chapter 8 and two methods were proposed. The first models the correspondence between the set of salient points as an affine transformation and uses Singular Value Decomposition properties to find the corresponding pairs, whereas the second method is based on spatial geometrical constraints and different distance measures to define the right mapping. The initial evaluations of the proposed comparison methods have shown encouraging results.

Based on the extracted text-related properties in the previous chapters, Chapter 9 presented a semantic browsing environment. In particular, to help the user in exploring collections of images, a novel visualization method named Image Proximity Matrix (IPM) was proposed. Moreover, two dimension reduction methods were presented, namely Multidimensional Scaling and Sammon Mapping, to complement the IPM view. The evaluation of the performance of the system was beyond the scope of this thesis because it implies the feedback of the users.

10.2 Open Issues and Future Research

There are some open issues in the different areas covered in this thesis.

- **Text detection/localization in complex images**

  The current text localization techniques are mainly based on the assumption that the text strings are aligned horizontally. Although in Section 3.3.3.7 the localization of the text of an arbitrary alignment is addressed, the experiments have shown that the presented solution to this problem does not perform very accurately in the cases where parts of the background in which the text is embedded are falsely detected as "text" due to their texture similarities with the text. Consequently, there is a need for further research to develop robust methods to localize text of any alignment. Furthermore, the incorporation of other classification methods such as supervised learning classifiers into the proposed UTDL-Texture approach can further improve the detection results. However, in this case the performance of the approach will depend on the training data. The experiments have shown that the UTDL-Texture method can also detect scene text when its contrast with the background is high and only little distortion is present. Thus, the extension of the methods to further improve the detection/localization of scene text would be interesting according to the challenges it offers.

- **Fuzzy clustering ensemble for text detection in videos**

  The application of the clustering ensemble with different features could possibly lead to further improvements. Furthermore, the combination of different clustering algorithms instead of different instances of FCM could be of interest. The generality of the proposed methodology makes it interesting to be applied also for the detection of any static object present in a video.

- **Text tracking in videos**

  The text tracking method proposed in Chapter 5 is based on the assumption that the text is static or moves horizontally or vertically in a linear form. The extension of the tracking algorithm to enable tracking of a non-rigid moving text (e.g. the movement of scene text) remains future work. Second, the tracking results should be improved for moving thin text whose intersection area with the background is bigger. Finally, the use of backward motion vectors in B-frames might further improve the tracking accuracy.

- **Text segmentation and binarization**

  The text segmentation methods in Chapter 6 assume that the characters within a localized text string have a homogeneous color, which is the case for most of artificial and scene text. However, theoretically text can be also multicolored. The text segmentation method in Section 6.4.4 can be modified to segment multicolored text strings too, by the development of an algorithm which is capable to merge different text components with each other. During our experiments in Chapter 6, a commercial optical character recognition tool [Abb] was used. It is clear that the use of a domain dependent dictionary can improve soundly the performance of the recognition process in general and the word recognition rate in particular. This will subsequently lead to a better indexing and annotation of the respective media data. Furthermore, the development of a new OCR system or the integration of an existing open source OCR into the current system in order to complete the full chain of a text extraction system (from the input image to the ASCII i.e. Unicode text) is still necessary. Finally, including algorithms that enable the integration of successive frames to support the video text segmentation process for example by the employment of the clustering ensemble in a similar way as explained in Chapter 4 or by means of increasing the resolution and reducing the background complexity [DM05, MTM05] is an interesting area for future work.

- **Script recognition in complex images**

  The use of other features like Gabor filters [Big94] during the recognition of the script is an interesting area of further research. Furthermore, the extension of the algorithm to distinguish between more than two scripts can be also of interest for the future.

- **Holistic comparison of text images**

  The application of other corner detector algorithms, possibly rotation invariant [STL+02], could further improve the performance of the

proposed approach. Furthermore, the evaluation of other alignment methods and other features such as shape context features similar to [MBM05] would be of a special interest.

- **Semantic browsing of images**

  Visualization techniques have gained a lot of attention recently in order to make it more easy for the user to understand the nature of the results. A first step in this direction has been performed through the introduction of several visualization/browsing methods in Chapter 9. However, their further improvement by, for example, including active learning aspects and user-based evaluation still remain a challenge for future research. Furthermore, the proposed matrix-based visualization (IPM) method is of a general nature and can be applied in a wider scope. Its usefulness extends to any multimedia objects for which the definition of an proximity matrix is possible, including list of document images, text documents or pieces of music or speeches.

- **Other directions**

  Two other aspects need to be mentioned for developing efficient images/videos text based annotation and retrieval systems. The first is related to the optimization of the proposed algorithms in order to reduce the memory resources and the computation time they require. The application of the Grid Computing and the Grid Service paradigms for a distributed text detection/localization and segmenting system in analogy to [EFGF04] could be a possible solution. The second aspect concerns the study and development of effective multimedia indexing structures for fast and efficient retrieval of multimedia data.

Bibliography

[AAS05] J. Alon, V. Athitsos, and S. Sclaroff. Online and Offline Character Recognition Using Alignment to Prototypes. Proceedings of the International Conference on Document Analysis and Recognition, pages 839–845. IEEE Press, 2005.

[Abb] ABBYY FineReader 7.0 Professional. *www.abbyy.com.*

[ACK00] S. Antani, D. Crandall, and R. Kasturi. Robust Extraction of Text in Video. Proceedings of the International Conference on Pattern Recognition (Vol. 1), pages 1445–1449. IEEE Computer Society, 2000.

[AD99] L. Agnihotri and N. Dimitrova. Text Detection for Video Analysis. Proceedings of the International Conference on Multimedia Computing and Systems, pages 109–113. IEEE Press, 1999.

[Ant01] S. K. Antani. *Reliable Extraction of Text from Video.* PhD thesis, The Pennsylvania State University. The Graduate School. Department of Computer Science and Engineering. 2001.

[AS03] V. Ablavsky and M.R. Stevens. Automatic Features Selection with Applications to Script Identification of Degraded Documents. Proceedings of the International Conference on Document Analysis and Recognition, pages 750–754. IEEE Computer Society, 2003.

[BA83] P.J. Burt and E.H. Adelson. The Laplacian Pyramid as a Compact Image Code. *Transactions on Communications*, COM-31(4):532–540, 1983.

[Bez74] J.C. Bezdek. Cluster Validity with Cluster Sets. *Journal of Cybenetics*, 3:58–73, 1974.

[Bez76] J.C. Bezdek. A Physical Interpretation of Fuzzy ISODATA. *Transactions on Systems, Man and Cybernetics*, 6:387–389, 1976.

[BH02] J.C. Bezdek and R.J. Hathaway. VAT: A Tool for Visual Assessment of (Cluster) Tendency. Proceedings of the International Joint Conference on Neural Networks, (Vol. 3), pages 2225–2230. IEEE Press, 2002.

[BH06] R. Benmokhtar and B. Huet. Classifier Fusion: Combination Methods for Semantic Indexing in Video Content. Proceedings of International Conference on Artificial Neural Networks, (Part II), pages 65–74. Springer Verlag, 2006.

[Big94] J. Bigun. Speed, Frequency, and Orientation Tuned 3-d Gabor Filter Banks and their Design. Proceedings of International Conference on Pattern Recognition, (Vol. 2), pages 184–187. EEE Computer Society, 1994.

[BMP01] S. Belongie, J. Malik, and J. Puzicha. Matching Shapes. Proceedings of the International Conference on Computer Vision, pages 454–461. IEEE Press, 2001.

[BVS05] D. Bargeron, P. Viola, and P. Simard. Boosting-Based Transductive Learning for Text Detection. Proceedings of the International Conference on Document Analysis and Recognition, pages 1166–1171. IEEE Computer Society, 2005.

[CAK03] D. Crandall, S. Antani, and R. Kasturi. Extraction of Special Effects Caption Text Events from Digital Video. *International Journal on Document Analysis and Recognition*, 5(2–3):138–157, 2003.

[cap] The CAPTCHA Project. *http://www.captcha.net/*.

[CBT01] D. Chen, H. Bourlard, and J.-P. Thiran. Text Identification in Complex Background using SVM. Proceedings of the International Conference on Computer Vision and Pattern Recognition (Vol. 2), pages 621–626. IEEE Computer Society, 2001.

[CCLL05] F. Chang, G.-C. Chen, C.-C. Lin, and W.-H. Lin. Caption Analysis and Recognition for Building Video Indexing Systems. *Multimedia Systems*, 10(4):344–355, 2005.

[CGL+05] P. Chiu, A. Girgensohn, S. Lertsithichai, W. Polak, and F. Shipman. MediaMetro: Browsing Multimedia Document Collections

with a 3D City Metaphor. International Multimedia Conference, pages 213–214. ACM Press, 2005.

[CGR00] C. Chen, G. Gagaudakis, and P. Rosin. Similarity-Based Image Browsing. Proceedings of the International Conference on Intelligent Information, pages 206–213. Publishing House of Electronics Industry, 2000.

[CH97] Y. Cui and Q. Huang. Character Extraction of License Plates from Video. Proceedings of the International Conference on Computer Vision and Pattern Recognition, pages 502–507. IEEE Press, 1997.

[CHL+04] D. Cai, X. He, Z. Li, W-Y. Ma, and J-R. Wen. Hierarchical Clustering of WWW Image Search Results Using Visual, Textual and Link Information. Proceedings of the ACM Multimedia Conference, pages 952–959. ACM Press, 2004.

[CHR03] M. Carey, D.C. Heesch, and S.M. Rüger. Info Navigator: A Visualization Tool for Document Searching and Browsing. International Conference on Distributed Multimedia Systems, pages 23–28, 2003.

[CO03] D. Chen and J.M. Odobez. Sequential Monte Carlo Video Text Segmentation. International Conference on Image Processing, pages 21–24. IEEE Press, 2003.

[CSB01] D. Chen, K. Shearer, and H. Bourlard. Text Enhancement with Asymmetric Filter for Video OCR. Proceedings of the International Conference on Image Analysis and Processing, pages 192–197. IEEE Computer Society, 2001.

[CSL02] M. Cai, J. Song, and M. R. Lyu. A New Approach for Video Text Detection. Proceedings of the International Conference on Image Processing, pages 117–120. IEEE Computer Society, 2002.

[DKN04] T. Deselaers, D. Keysers, and H. Ney. Features for Image Retrieval: A Quantitative Comparison. Proceedings of Annual Meeting of the German Association for Pattern Recognition, pages 228–236. Springer Verlag, 2004.

[DM05] K. Donaldson and G.K. Myers. Bayesian Super Resolution of Text in Video with Text-Specific Bimodal Prior. *International Journal on Document Analysis and Recognition*, 7(2–3):159–167, 2005.

[Dun74] J.C. Dunn. Well-Separated Clusters and Optimal Fuzzy Partitions. *Journal of Cybernetics*, 3:95–104, 1974.

[DUP99] B. Davis, J. Uhl, and H. Porta. *Vector Calculus and Mathematica (CD ROM)*. Wolfram Research, 1999.

[EBK+05] R. Ewerth, C. Beringer, T. Kopp, M. Niebergall, T. Stadelmann, and B. Freisleben. University of Marburg at TRECVID 2005: Shot Boundary Detection and Camera Motion Estimation Results. Online Proceedings of TREC Video Retrieval Evaluation. IEEE Press, 2005.

[EF03] R. Ewerth and B. Freisleben. Frame Difference Normalization: An Approach to Reduce Error Rates of Cut Detection Algorithms for MPEG Videos. Proceedings of the International Conference on Image Processing, pages 1009–1012. IEEE Press, 2003.

[EF04a] R. Ewerth and B. Freisleben. Improving Cut Detection in MPEG Videos by GOP-Oriented Frame Difference Normalization. Proceedings of International Conference on Pattern Recognition, (Vol. 2), pages 807–810. EEE Computer Society, 2004.

[EF04b] R. Ewerth and B. Freisleben. Video Cut Detection Without Thresholds. Proceedings of Workshop on Signals, Systems and Image Processing, pages 227–230. PTETiS Press, 2004.

[EF06a] R. Ewerth and B. Freisleben. Computerunterstützte Filmanalyse mit Videana. *Augenblick: Hefte zur Medienwissenschaft, Schüren-Verlag*, 2006.

[EF06b] R. Ewerth and B. Freisleben. Gesichtsdetektion und -erkennung in bildern und videos für die medienwissenschaftliche analyse. *Universi Verlag, W. Beilenhoff, W. Hülk, K. Kreimeier, und M. Erstic (eds.)*, 2006.

[EF06c] R. Ewerth and B. Freisleben. Self-Supervised Learning for Robust Video Indexing. Proceedings of International Conference on Multimedia and Expo. IEEE Press, 2006.

[Eff00] N. Efford. *Digital Image Processing: a Practical Introduction Using Java*. Addison Wesley, 2000.

[EFGF04] R. Ewerth, T. Friese, M. Grube, and B. Freisleben. Grid Services for Distributed Video Cut Detection. Proceedings of the International Symposium on Multimedia Software Engineering, pages 164–168. IEEE Press, 2004.

[EGG+03] R. Ewerth, J. Gllavata, M. Gollnick, F. Mansouri, E. Papalilo, R. Sennert, J. Wagner, B. Freisleben, and M. Grauer. Methoden und werkzeuge zur rechnergestützten medienwissenschaftlichen analyse. pages 306–320, 2003.

[EMF06] R. Ewerth, Markus Muehling, and B. Freisleben. Self-Supervised Learning of Face Appearances in TV Casts and Movies. Proceedings of the IEEE International Symposium on Multimedia, page (to appear). IEEE Press, 2006.

[ESTF04] R. Ewerth, M. Schwalb, P. Tessmann, and B. Freisleben. Estimation of Arbitrary Camera Motion in MPEG Videos. Proceedings of the International Conference on Pattern Recognition (Vol. 1), pages 512–515, 2004.

[FH51] E. Fix and J. Hodges. Discriminatory Analysis- Nonparametric Discrimination: Consistency Properties. Technical Report ,US Air Force, School of Aviation Medicine, 1951.

[FK03] I. Foster and C. Kesselman. *The Grid 2: Blueprint for a New Computing Infrastructure*. Elsevier Ltd., 2003.

[Fre01] A.L.N. Fred. Finding Consistent Clusters in Data Partitions. *Multiple Classifier Systems*, 2096:309–318, 2001.

[FSN+95] M. Flickner, H. Sawhney, W. Niblack, J. Ashley, Q. Huang, B. Dom, M. Gorkani, J. Hafner, D. Lee, D. Petkovic, D. Steele, and P. Yanker. Query by Image and Video Content: The QBIC System . *The Flagship Magazine of the IEEE Computer Society*, 28(9):23–32, 1995.

[GEF03a] J. Gllavata, R. Ewerth, and B. Freisleben. A Robust Algorithm for Text Detection in Images. Proceedings of the International Symposium on Image and Signal Processing and Analysis, pages 611–616. IEEE Press, 2003.

[GEF03b] J. Gllavata, R. Ewerth, and B. Freisleben. Finding Text in Images via Local Thresholding. Proceedings of the International Symposium on Signal Processing and Information Technology, pages 539–542. IEEE Press, 2003.

[GEF04a] J. Gllavata, R. Ewerth, and B. Freisleben. A Text Detection, Localization and Segmentation System for OCR in Images. Proceedings of the International Symposium on Multimedia Software Engineering, pages 310–317. IEEE Press, 2004.

[GEF04b] J. Gllavata, R. Ewerth, and B. Freisleben. Text Detection in Images Based on Unsupervised Classification of High-Frequency Wavelet Coefficients. Proceedings of the International Conference on Pattern Recognition (Vol. 1), pages 425–428. IEEE Computer Society, 2004.

[GEF04c] J. Gllavata, R. Ewerth, and B. Freisleben. Tracking Text in MPEG Videos. Proceedings of the Annual ACM International Conference on Multimedia, pages 240–243. ACM Press, 2004.

[GESF04] J. Gllavata, R. Ewerth, T. Stefi, and B. Freisleben. Unsupervised Text Segmentation Using Color and Wavelet Features. Proceedings of the International Conference on Image and Video Retrieval, pages 216–224. Springer Verlag, 2004.

[GF05a] J. Gllavata and B. Freisleben. Adaptive Fuzzy Text Segmentation in Images with Complex Backgrounds Using Color and Texture. Proceedings of the International Conference on Computer Analysis of Images and Patterns, pages 756–765. Springer Verlag, 2005.

[GF05b] J. Gllavata and B. Freisleben. Script Recognition in Images with Complex Backgrounds. Proceedings of the International Symposium on Signal Processing and Information Technology, pages 589–594. IEEE Press, 2005.

[GF06] J. Gllavata and B. Freisleben. Combination of Crisp and Fuzzy Clustering Methods for Text Extraction in Complex Images. Proceedings of the International Conference on Application of Fuzzy Systems and Soft Computing, pages 148–157. b-Quadrat Verlag, 2006.

[GMT05] A. Gionis, H. Mannila, and P. Tsaparas. Clustering Aggregation. Proceedings of the International Conference on Data Engineering, pages 341–352. IEEE Press, 2005.

[GQF06a] J. Gllavata, E. Qeli, and B. Freisleben. Detecting Text in Videos Using Fuzzy Clustering Ensembles. Proceedings of the IEEE International Symposium on Multimedia, page (to appear). IEEE Press, 2006.

[GQF06b] J. Gllavata, E. Qeli, and B. Freisleben. Holistic Comparison of Text Images for Content-Based Retrieval. Proceedings of the IEEE International Symposium on Multimedia, page (to appear). IEEE Press, 2006.

[Gro98] MPEG-7 Requirement Group. Description of MPEG-7 Content Set. MPEG Atlantic City Meeting, Doc. ISO/IEC JTC1/SC29/WGll/N2467, 1998.

[GW01] R.C. Gonzales and R.E. Woods. *Digital Image Processing*. Prentice Hall, 2001.

[HKTK97] J. Hochberg, P. Kelly, T. Thomas, and L. Kerns. Automatic Script Identification from Document Images using Cluster-based Templates. *Transactions on Pattern Analysis and Machine Intelligence*, 19(2):176–181, 1997.

[HR04a] D. Heesch and S. Rueger. NN^k Networks for Content-Based Image Retrieval. Proceedings of the European Conference on Image Retrieval (Vol. 2997), pages 253–266. Springer Verlag, 2004.

[HR04b] D. Heesch and S. Rueger. Three Interfaces for Content-Based Access to Image Collections. Proceedings of the International Conference on Image and Video Retrieval (Vol. 3115), pages 491–499. Springer Verlag, 2004.

[HR05] D. Heesch and S. Rueger. Image Browsing: Semantic Analysis of NN^k Networks. Proceedings of the International Conference on Image and Video Retrieval, pages 609–618. Springer LNCS, 2005.

[HS88] C. Harris and M. Stephens. A Combined Corner and Edge Detector. Proceedings of the 4th Alvey Vision Conference, pages 147–151, 1988.

[HYT+04] H. Hase, M. Yoneda, Sh. Tokai, J. Kato, and Y.S. Ching. Color Segmentation for Text Extraction. *International Journal on Document Analysis and Recognition*, pages 271–284, 2004.

[HYZ02] X-S. Hua, P. Yin, and H.-J. Zhang. Efficient Video Text Recognition Using Multiple Frame Integration. Proceedings of the International Conference on Image Processing (Vol. 2), pages 397–400. IEEE Press, 2002.

[HYZM03] Y. Hao, Z. Yi, H. Zengguang, and T. Min. Automatic Text Detection in Video Frames based on Bootstrap Artificial Neural Network and CED. *International Journal of WSCG, 11th International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision*, 11(1), 2003.

[JD88] A.K. Jain and R.C. Dubes. *Algorithms for Clustering Data*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1988.

[JKB03] H.-M. Je, D. Kim, and S. Y. Bang. Human face detection in digital video using svmensemble. *Neural Process Letters*, 17(3):239–252, 2003.

[JKJ04] K. Jung, K.I. Kim, and A.K. Jain. Text Information Extraction in Images and Videos: A Survey. *Pattern Recognition Letters*, 37:977–997, 2004.

[JM97] D. Johnson and L. McGeoch. *Local Search in Combinatorial Optimization, ch. The Traveling Salesman Problem: A Case Study*. Wiley and Sons, 1997.

[JMD05] S. Jaeger, H. Ma, and D. Doermann. Identifying Script on Word-level with Informational Confidence. Proceedings of the International Conference on Document Analysis and Recognition, page to appear. IEEE Computer Society, 2005.

[JY98] A. K. Jain and B. Yu. Automatic Text Location in Images and Video Frames. Proceedings of the International Conference on Pattern Recognition, pages 1497–1499, 1998.

[JZ96] A.K. Jain and Y. Zhong. Page Segmentation Using Texture Analysis. *Pattern Recognition Letters*, 29(5):743–770, 1996.

[KC01] D. S. Kim and S. I. Chien. Automatic Car License Plate Extraction using Modified Generalized Symmetry Transform and Image Warping. Proceedings of the International Symposium on Industrial Electronics (Vol. 3), pages 2022–2027, 2001.

[KJPK01] K.I. Kim, K. Jung, S.H. Park, and H.J. Kim. Support Vector Machine-based Text Detection in Digital Video. *Pattern Recognition Letters*, 34(2):527–529, 2001.

[KK98] G. Karypis and V. Kumar. A Fast and High Quality Multilevel Scheme for Partitioning Irregular Graphs. *SIAM Journal of Scientific Computing*, 20(1):359–392, 1998.

[KLL04] D.W. Kim, K.H. Lee, and D. Lee. A Novel Initialization Scheme for the Fuzzy C-means Algorithm for Color Clustering. *Pattern Recognition Letters*, 25:227–237, 2004.

[LDK00] H. Li, D. Doermann, and O. Kia. Automatic Text Detection and Tracking in Digital Videos. *Transactions on Image Processing*, 9(1):147–156, 2000.

[LFG+02] H. Luttermann, B. Freisleben, M. Grauer, U. Kelter, T. Kamphusmann, U. Merten, G. Rößling, T. Unger, and J.Waldhans. Mediana: A Workbench for the Computer-Based Analysis of Media Data. Proceedings of the Wirtschaftsinformatik 44 1, pages 41–51, 2002.

[LG98] J. Lia and R.M. Gray. Text and Pictures Segmentation by the Distribution Analysis of Wavelet Coefficients. International Conference on Image Processing, pages 790–794. IEEE Press, 1998.

[LKD99] H. Li, O. Kia, and D. Doermann. Text Enhancement in Digital Videos. Document Recognition and Retrieval VI (Vol. 3651), pages 2–9. SPIE, 1999.

[LRM04] V. Lavrenko, T.M. Rath, and R. Manmatha. Holistic Word Recognition for Handwritten Historical Documents. Proceedings of the International Workshop on Document and Image Analysis for Digital Libraries, pages 278–286. IEEE Computer Society, 2004.

[LSC05] M.R. Lyu, J. Song, and M. Cai. A Comprehensive Method for Multilingual Video Text Detection, Localization, and Extraction. *Transactions on Circuits and Systems for Video Technology*, 15(2):243–255, 2005.

[LSDJ06] M.S. Lew, N. Sebe, C. Djerba, and R. Jain. Content-Based Multimedia Information Retrieval: State of the Art and Challenges. *Transactions on Multimedia Computing, Communications, and Applications*, 2(1):1–19, 2006.

[LT04] P.K. Loo and C.L. Tan. Adaptive Region Growing Color Segmentation for Text Using Irregular Pyramid. International Workshop on Document Analysis Systems, pages 264–275. Springer Verlag, 2004.

[LW67] G.N. Lance and W.T. Williams. A General Theory of Classificatory Sorting Strategies: II. Clustering Systems. *Computer Journal*, 10:271–277, 1967.

[LW02] R. Lienhart and A. Wernicke. Localizing and Segmenting Text in Images and Videos. *Transactions on Circuits and Systems for Video Technology*, 12(4):256–268, 2002.

[LXT+04] H. Liu, X. Xie, X. Tang, Z-W. Li, and W-Y. Ma. Effective Browsing of Web Image Search Results. Proceedings of the ACM

Multimedia Information Retrieval Workshop, pages 84–90. ACM Press, 2004.

[LZ00] D. Loprestie and J.Y. Zhou. Locating and Recognizing Text in WWW Images. *Journal of Information Retrieval*, pages 177–206, 2000.

[Mac67] J. B. MacQueen. Some Methods for Classification and Analysis of Multivariate Observations. Proceedings of 5th Berkeley Symposium on Mathematical Statistics and Probability, pages 281–297. University of California Press, 1967.

[Mar03] V.Y. Mariano. *Video Object Detection and Matching*. PhD thesis, The Pennsylvania State University. The Graduate School. Department of Computer Science and Engineering. 2003.

[MBM05] G. Mori, S. Belongie, and J. Malik. Efficient Shape Matching Using Shape Contexts. *Transactions on Pattern Analysis and Machine Intelligence*, 27(11):1832–1837, 2005.

[MD03] H. Ma and D. Doermann. Gabor Filter Based Multi-Class Classifier for Scanned Document Images. Proceedings of the International Conference on Document Analysis and Recognition, pages 968–972. IEEE Computer Society, 2003.

[MD04] H. Ma and D. Doermann. Word Level Script Identification for Scanned Document Images. Proceedings of the International Conference on Document Recognition and Retrieval, pages 178–191. SPIE Press, 2004.

[MHR95] R. Manmatha, C. Han, and E. M Riseman. Word Spotting: A New Approach to Indexing Handwriting. Technical Report UM-CS-1995-105, Computer Science Dept, University of Massachusetts at Amherst, 1995.

[MJHP04] G. Moy, N. Jones, C. Harkless, and R. Potter. Distortion Estimation Techniques in Solving Visual CAPTCHAs. Proceedings of the International Conference on Computer Vision and Pattern Recognition (Vol. 2), pages 23–28. IEEE Computer Society, 2004.

[MKB79] K.V. Mardia, J.T. Kent, and J.M. Bibby. *Multivariate Analysis*. Academic Press, 1979.

[MM03] G. Mori and J. Malik. Recognizing Objects in Adversarial Clutter: Breaking a Visual CAPTCHA. Proceedings of Computer

Vision and Pattern Recognition (Vol. 1), pages 134–142. IEEE Press, 2003.

[MMS+00] H. Müller, W. Müller, D.McG. Squire, S. Marchand-Maillet, and T. Pun. Strategies for Positive and Negative Relevance Feedback in Image Retrieval. Proceedings of the International Conference on Pattern Recognition (Vol. 1), pages 5043–5047. IEEE Computer Society, 2000.

[MMS03] S. Marinai, E. Marino, and G. Soda. Indexing and Retrieval of Words in Old Documents. Proceedings of the International Conference on Document Analysis and Recognition (Vol. 1), pages 223–228. IEEE Computer Society, 2003.

[MPE] MPEG Encoding Basics *www.media-matters.net/docs/resources/.*

[MTM05] C. Mancas-Thillou and M. Mirmehdi. Super-Resolution Text Using the Teager Filter. Proceedings of the First International Workshop on Camera-Based Document Analysis and Recognition, pages 10–16, 2005.

[NC05] Ch.-W. Ngo and Ch.-K. Chan. Video Text Detection and Segmentation for Optical Character Recognition. *Multimedia Systems*, 10:261–272, 2005.

[Nib86] W. Niblack. *An Introduction to Digital Processing.* Prentice Hall, 1986.

[NW04] G.P. Nguyen and M. Worring. Optimizing Similarity Based Visualization in Content Based Image Retrieval. International Conference on Multimedia and Expo, pages 759–762. IEEE Press, 2004.

[NY03] A. Nakamura and K. Yamamoto. Caption Text Recognition in Video Frames by MAP Matching. Proceedings of the International Conference on Document Analysis and Recognition (Vol. 2), pages 650–655. IEEE Computer Society, 2003.

[OC02] J.M. Odobez and D. Chen. Robust Video Text Segmentation and Recognition with Multiple Hypotheses. International Conference on Image Processing (Vol. 2), pages 433–436. IEEE Press, 2002.

[omn] OmniPage Pro 12. *www.scansoft.com/omnipage/.*

[Ots79] N. Otsu. A Threshold Selection Method from Gray-Level Histograms. *Transaction on Systems, Man and Cybernetics*, 9:62–66, 1979.

[PSBH01] T. Perroud, K. Sobottka, H. Bunke, and L. Hall. Text Extraction from Color Documents-Clustering Approaches in Three and Four Dimensions. Proceedings of the International Conference on Document Analysis and Recognition, pages 937–941. IEEE Press, 2001.

[Qel06] E. Qeli. *Information Visualization Techniques for Metabolic Engeniering*. PhD thesis, University of Marburg, Department of Mathematics and Computer Science, (in Preparation). 2006.

[QWF04] E. Qeli, W. Wiechert, and B. Freisleben. Visualizing Time-Varying Matrices Using Multidimensional Scaling and Reorderable Matrices. Proceedings of the International Conference on Information Visualization, pages 561–567. IEEE Press, 2004.

[QWF05] E. Qeli, W. Wiechert, and B. Freisleben. The Time-Dependent Reorderable Matrix Method for Visualizing Evolving Tabular Data. Proceedings of the IST/SPIE Conference on Visualization and Data Analysis, pages 199–207. SPIE Press, 2005.

[Raw76] G. Rawlinson. *The Significance of Letter Position in Word Recognition*. PhD thesis, Psychology Department, University of Nottingham, 1976.

[RBSW01] K. Rodden, W. Basalaj, D. Sinclair, and K.R. Wood. Does Organisation by Similarity Assist Image Browsing? In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 190–197. ACM Press, 2001.

[RFR04] J.L. Rothfeder, S. Feng, and T.M. Rath. Using Corner Feature Correspondences to Rank Word Images by Similarity. Proceedings of the International Workshop on Document Image Analysis and Retrieval. IEEE Computer Society, 2004.

[RGT97] Y. Rubner, L.J. Guibas, and C. Tomasi. The Earth Movers Distance, Multi-dimensional Scaling, and Color-based Image Retrieval. DARPA Image Understanding Workshop, 1997.

[RM04] T.M. Rath and R. Manmatha. Word Image Matching Using Dynnamic Time Warping. Proceedings of the Conference on Computer Vision and Pattern Recognition (Vol. 2), pages 521–527. IEEE Computer Society, 2004.

[RTG98] Y. Rubner, C. Tomasi, and L.J. Guibas. A metric for distributions with applications to image databases. Proceedings of the International Conference on Computer Vision, pages 59–66. IEEE Press, 1998.

[SBN+98] C.Y. Suen, S. Bergler, N. Nobile, B. Waked, C.P. Nadal, and A. Bloch. Categorizing Document Images into Script and Language Classes. Proceedings of the International Conference on Advances in Pattern Recognition, pages 297–306. Springer Verlag, 1998.

[SC97] J.R. Smith and S.-F. Chang. Querying by Color Regions Using the VisualSEEk Content-Based Visual Query System. Proceedings of the Intelligent Multimedia Information Retrieval, pages 23–41. MIT Press, 1997.

[SDB98] J.C. Shim, C. Dorai, and R. Bolle. Automatic Text Extraction from Video for Content-Based Annotation and Retrieval. Proceedings of the International Conference on Pattern Recognition, pages 618–620. IEEE Press, 1998.

[SF06] T. Stadelmann and B. Freisleben. Fast and Robust Speaker Clustering Using the Earth Movers Distance and MIXMAX Models. Proceedings of International Conference on Acoustics, Speech, and Signal Processing. IEEE Press, 2006.

[SG02] A. Strehl and J. Ghosh. Cluster Ensembles - A Knowledge Reuse Framework for Combining Multiple Partitions. *Journal of Machine Learning Research*, 3:583–617, 2002.

[SGJ01] S. Santini, A. Gupta, and R. Jain. Emergent Semantics Through Interaction in Image Databases. *Transactions on Knowledge and Data Engineering*, 13(3):337–351, 2001.

[SHW05] P. Silapachote, A. Hanson, and R. Weiss. A Hierarchical Approach to Sign Recognition. Proceedings of the International Workshop on Applications of Computer Vision (Vol. 1), pages 22–28. IEEE Computer Society, 2005.

[SKH+99] T. Sato, T. Kanade, E.K. Huges, M.A. Smith, and S. Satoh. Video OCR: Indexing Digital News Libraries by Recognition of Superimposed Caption. Multimedia Systems (Vol. 7), pages 385–395. ACM Press, 1999.

[SKHS98] T. Sato, T. Kanade, E.K. Hughes, and M.A. Smith. Video OCR for Digital New Archives. Proceedings of the International Workshop on Content-Based Access of Image and Video Databases, pages 52–60. IEEE Computer Society, 1998.

[SLH91] G.L. Scott and H.C. Longuett-Higgins. An Algorithm for Associating the Features of Two Patterns. Proceedings of the Royal Society of London (Vol. B244), pages 21–26, 1991.

[Spi97] A.L. Spitz. Determination of the Script and Language Content of Document Images. *Transactions on Pattern Analysis and Machine Intelligence*, 19(3):235–245, 1997.

[SSP97] J. Sauvola, T. Seppänen, and S. Haapakoski M. Pietikäinen. Adaptive Document Binarization. International Conference on Document Binarization (Vol. 21), pages 147–152, 1997.

[STL+02] N. Sebe, Q. Tian, E. Loupias, M. Lew, and T. Huang. Evaluation of Salient Point Techniques. Proceedings of Conference on Image and Video Retrieval, pages 367–377. IEEE Press, 2002.

[SWH+05] P. Silapachote, J. Weinman, A. Hanson, R. Weiss, and M.A. Mattar. Automatic Sign Detection and Recognition in Natural Scenes. Proceedings of the International Conference on Computer Vision and Pattern Recognition (Vol. 3), pages 22–27. IEEE Computer Society, 2005.

[SWS+00] A.W.M. Smeulders, M. Worring, S. Santini, A. Gupta, and R. Jain. Content-Based Image Retrieval at the End of the Early Years. *Transactions on Pattern Analysis and Machine Intelligence*, 22(12):1349–1380, 2000.

[SWvG+05] C.G.M. Snoek, M. Worring, J. van Gemert, J.-M. Geusebroek, D. Koelma, G.P. Nguyen, O. de Rooij, and F. Seinstra. MediaMill: Exploring News Video Archives Based on Learned Semantics. International Multimedia Conference, pages 225–226. ACM Press, 2005.

[Tan98] T.N. Tan. Rotation Invariant Texture Features and their Use in Automatic Script Identification. *Transactions on Pattern Analysis and Machine Intelligence*, 20(7):751–756, 1998.

[TdSL00] J.B. Tenenbaum, V. de Silva, and J.C. Langford. A Global Geometric Framework for Nonlinear Dimensionality Reduction. *Science*, 290:2319–2323, 2000.

[TGLZ02] X. Tang, X. Gao, J. Liu, and H. Zhang. A Spatial Temporal Approach for Video Caption Detection and Recognition. *Transactions on Neural Networks*, 13(4):961–971, 2002.

[the] The free dictionary. `http://www.thefreedictionary.com`.

[TL04] C.L. Tan and Q.H. Liu. Extraction of Newspaper Headlines from Microfilm for Automatic Indexing. *International Journal on Document Analysis and Recognition*, 6:201–210, 2004.

[TLG+02] X. Tang, B. Luo, X. Gao, E. Pissaloux, and H. Zhang. Video Text Extraction Using Temporal Feature Vectors. Proceedings of the International Conference on Multimedia and Expo, pages 85–88. IEEE Press, 2002.

[TLH99] C.L. Tan, P.Y. Leong, and S. He. Language Identification in Multilingual Documents. Proceedings of the International Symposium on Intelligent Multimedia and Distance Education, pages 59–64, 1999.

[TLS96] Y.Y. Tang, S.W. Lee, and C.Y. Suen. Automatic Document Processing: A Survey. *Pattern Recognition Letters*, 29(12):1931–1952, 1996.

[TSMR03] R.S. Torres, C.G. Silva, C.B. Medeiros, and H.V Rocha. Visual Structures for Image Browsing. Proceedings of the Conference on Information and Knowledge Management, pages 49–55. ACM Press, 2003.

[US05] S. Uchida and H. Sakoe. A Survey of Elastic Matching Techniques for Handwritten Character Recognition. *IEICE Transactions on Information and Systems*, E88-D(8):1781–1790, 2005.

[Vap98] V.N. Vapnik. *The Nature of Statistical Learning Theory.* Springer Verlag, 1998.

[VBL95] J. Villasenor, B. Belzer, and J. Liao. Wavelet Filter Evaluation for Image Compression. *Transactions on Image Processing*, 4(8):1053–1060, 1995.

[VJ01] P. Viola and M. Jones. Rapid Object Detection Using a Boosted Cascade of Simple Features. Proceedings of the International Conference on Computer Vision and Pattern Recognition (Vol. 1), pages 511–518. IEEE Computer Society, 2001.

[VT00] R.C. Veltkamp and M. Tanase. Content-Based Image Retrieval Systems: A Survey. Technical Report UU-CS-2000-34, Institute of Information and Computing Science, Utrecht University, 2000.

[WBSK98] B. Waked, S. Bergler, C.Y. Suen, and S. Khoury. Skew Detection, Page Segmentation and Script Classification of Printed Document Images. Proceedings of the International Conference on Systems, Man, and Cybernetics, pages 4470–4475. IEEE Computer Society, 1998.

[WJC02] C. Wolf, J.M. Jolion, and F. Chassaing. Text Localization, Enhancement and Binarization in Multimedia Documents. International Conference on Pattern Recognition (Vol. 4), pages 1037–1040, 2002.

[WJW04] R. Wang, W. Jin, and L. Wu. A Novel Video Caption Detection Approach Using Multi-Frame Integration. Proceedings of the International Conference on Pattern Recognition (Vol. 1), pages 449–452. IEEE Press, 2004.

[WMR99] V. Wu, R. Manmatha, and E.M. Riseman. Textfinder: An Automatic System to Detect and Recognize Text in Images. *Transaction on Pattern Analysis and Machine Intelligence*, 21:1224–1229, 1999.

[WS00] G. Wyszecki and W.S. Stiles. *Color Science: Concepts and Methods, Quantitative Data and Formulae*. Wiley-Interscience Publication, 2000.

[Wu96] X. Wu. YIQ Vector Quantization in a New Color Palette Architecture. *Transactions on Image Processing*, 5(2):321–329, 1996.

[XSLZ01] X.-S.Hua, W. Liu, and H.J. Zhang. Automatic Performance Evaluation for Video Text Detection. Proceedings of the International Conference on Document Analysis and Recognition, pages 545–550. IEEE Press, 2001.

[YGH04] Q. Ye, W. Gao, and Q. Huang. Automatic Text Segmentation from Complex Background. International Conference on Image Processing (Vol. 5), pages 2905–2908. IEEE Press, 2004.

[YGWZ03] Q. Ye, W. Gao, W. Wag, and W. Zeng. A Robust Text Detection Algorithm in Images and Video Frames. Proceedings of the Joint Conference of the Fourth International Conference on Information, Communications and Signal Processing, and the

Fourth Pacific Rim Conference on Multimedia (Vol. 2), pages 802–806. IEEE Press, 2003.

[YHGZ05] Q. Ye, Q. Huang, W. Gao, and D. Zhao. Fast and Robust Text Detection in Images and Video Frames. *Image and Vision Computing*, 23:565–567, 2005.

[YJM97] B. Yu, A. K. Jain, and M. Mohiuddin. Address Block Location on Complex Mail Pieces. Proceedings of the International Conference on Document Analysis and Recognition, pages 897–901, 1997.

[YL95] B. Yeo and B. Liu. Rapid Scene Analysis on Compressed Video. *Transactions on Circuits and Systems for Video Technology*, 5(6):533–544, 1995.

[Zad65] L.A. Zadeh. Fuzzy Sets. *Information and Control*, 8:338–353, 1965.

[ZGST94] H. J. Zhang, Y. Gong, S. W. Smoliar, and S. Y. Tan. Automatic Parsing of News Video. Proceedings of the International Conference on Multimedia Computing and Systems, pages 45–54. IEEE Press, 1994.

[ZH03] X.S. Zhou and T.S. Huang. Relevance Feedback in Image Retrieval: A Comprehensive Review. *Multimedia Systems*, 8(6):536–544, 2003.