Text Segmentation in Web Images Using Colour Perception and Topological Features

Dimosthenis A. Karatzas

Abstract

The research presented in this thesis addresses the problem of Text Segmentation in Web images. Text is routinely created in image form (headers, banners etc.) on Web pages, as an attempt to overcome the stylistic limitations of HTML. This text however, has a potentially high semantic value in terms of indexing and searching for the corresponding Web pages. As current search engine technology does not allow for text extraction and recognition in images, the text in image form is ignored. Moreover, it is desirable to obtain a uniform representation of all visible text of a Web page (for applications such as voice browsing or automated content analysis). This thesis presents two methods for text segmentation in Web images using colour perception and topological features.

The nature of Web images and the implicit problems to text segmentation are described, and a study is performed to assess the magnitude of the problem and establish the need for automated text segmentation methods.

Two segmentation methods are subsequently presented: the Split-and-Merge segmentation method and the Fuzzy segmentation method. Although approached in a distinctly different way in each method, the safe assumption that a human being should be able to read the text in any given Web Image is the foundation of both methods' reasoning. This anthropocentric character of the methods along with the use of topological features of connected components, comprise the underlying working principles of the methods.

An approach for classifying the connected components resulting from the segmentation methods as either characters or parts of the background is also presented.

To my parents, Aris and Kitty
And my grandmother, Mari.

Aknowledgements

It would be considerably more difficult to walk along the twisting and occasionally disheartening road of research, without the help of numerous people who with their myriad contributions, suggestions, friendly ears and timely distractions, supported me during the last few years.

Firstly, thanks must go to my supervisor, Apostolos Antonacopoulos, for his support and feedback throughout my period of research. Without his guidance and his valuable contributions, this thesis would not be possible. Also in the department, I would like to thank my fellow doctoral students for their cherished comradeship. In particular, I wish to thank Dave Kennedy, who shared an office with me for the last three years, for maintaining a cheerful atmosphere by frequently distracting me with pure English jokes (which admittedly I didn't always understand).

Special thanks must also go to my ex-flatmate Dimitris Sapountzakis, with whom I spent one of my best years in Liverpool. Vicky Triantafyllou, for her love and support. Alexandros Soulahakis, for his delicious "loukoumades" and for keeping me up to date with the latest music. Maria and Vasilis Mavrogenis for the coffee breaks and the occasional barbeques. Paula and Dimitris for dragging me out for a drink once in a while. Chrysa, and Elena for keeping me home the rest of the times. Nikos Bechlos for keeping me up to date with mobile technologies, and numerous others who made my stay in Liverpool an enjoyable time.

Finally, I owe much to my family who with their tireless interest and constant support helped me immensely to accomplish this task. Thank you all for being there.

Declaration

I declare that this doctoral thesis was composed by myself and that the work contained therein is my own, except were explicitly stated otherwise in the text. The following articles were published during my period of research.

1. A. Antonacopoulos and **D. Karatzas**, "*Fuzzy Text Extraction from Web Images Based on Human Colour Perception*", to appear in the book: "*Web Document Analysis – Challenges and Opportunities*", A. Antonacopoulos and J. Hu (eds.), World Scientific Publishing Co., 2003
2. **D. Karatzas** and A. Antonacopoulos, "*Visual Representation of Text in Web Documents and Its Interpretation*", proceedings of 2nd Workshop on Visual Representations and Interpretations (VRI'2002), Liverpool, September 2002
3. A. Antonacopoulos and **D. Karatzas**, *"Fuzzy Segmentation of Characters in Web Images Based on Human Colour Perception"*, proceedings of the 5th IARP Workshop on Document Analysis Systems (DAS'2002), Princeton, NJ, August 2002
4. A. Antonacopoulos and **D. Karatzas**, *"Text Extraction from Web Images Based on Human Perception and Fuzzy Inference"*, proceedings of the 1st International Workshop on Web Document Analysis (WDA'2001), Seattle, Washington, September 2001
5. A. Antonacopoulos, **D. Karatzas** and J. Ortiz Lopez, *"Accessing Textual Information Embedded in Internet Images"*, proceedings of Electronic Imaging 2001: Internet Imaging II, San Jose, California, USA, January 2001
6. A. Antonacopoulos and **D. Karatzas**, *"An Anthropocentric Approach to Text Extraction from WWW Images"*, proceedings of the 4th IARP Workshop on Document Analysis Systems (DAS'2000), Rio de Janeiro, Brazil, December 2000

Table of Contents

Chapter 1

Introduction

Scientific and cultural progress would not be possible without ways to preserve and communicate information. Nowadays, automatic methods for retrieving, indexing and analysing information are increasingly being used in every day life. Accessibility of information is therefore a significant issue. World Wide Web is possibly the upshot of information exchange and an area where problems in information retrieval are easily identifiable.

Images constitute a defining part of almost every kind of document, either serving as a carrier of information related to the content of the document (e.g. diagrams, charts, etc), or used for aesthetic purposes (e.g. background, photographs, etc). At first, mostly due to limited bandwidth, the use of images on Web pages was rather restricted and their role was mainly to beautify the overall appearance of a page without carrying important (semantic) information. Nevertheless, as the overall speed of Internet connections is rising an increasing trend has been noticed to embed semantically important entities of Web pages into images. More specifically, designing eye-catching headers and menus and enhancing the appearance of a Web page using images for anything that the visitor should pay attention to (e.g. advertisements), is a strong advantage in the continuous fight to attract more visitors. Furthermore, certain parts of a document, such as equations, are bound to be in image form, as there is no alternative way to code them in Web pages.

Regardless of the role images play in Web pages, text remains the primary (if not the only) medium for indexing and searching Web pages. Search engines cannot access any text inside the images [4], while analysing the alternative text (ALT tags) provided, proves to be rather a disadvantage, since in almost half of the cases it is totally misleading as will be seen in Section 1.2. Therefore, important information on Web pages is not accessible, introducing a number of difficulties regarding automatic processes such as indexing and searching.

The research described in this Thesis addresses the need to extract textual information from Web images. The aim of the project is to examine possible ways to segment and identify characters inside colour images such as the ones found on Web pages. Towards the text segmentation in Web images, two different methods were implemented and tested. The first method works in a split-and-merge fashion, based on histogram analysis of the image in the *HLS* colour system, and connected component analysis. The second method is based on the definition of a propinquity measure between connected components, defined in the context of a fuzzy inference system. Colour distance and topological properties of components are incorporated into the propinquity measure providing a comprehensive way to analyse relationships between components. The innovation of both approaches, lies in the fact that they are both based on available (existing and experimentally derived) knowledge on the way humans perceive colour differences. This anthropocentric character of the two approaches is evident primarily through the way colour is manipulated, making use of human perception data and employing colour systems that are efficient approximations of the psychophysical manner humans understand colour.

The concept of the *“Web document”* will be introduced in the next section, and text extraction from Web images will be discussed in the context of document analysis. The impact and consequences of using images in Web pages is assessed in Section 1.2. The significant need for a method that extracts text from Web images is discussed in Section 1.3, along with possible applications for such a method. Section 1.4 examines the distinguishing characteristics of Web images and summarizes interesting observations made. Characteristic paradigms of Web images are also given in Section 1.4. The aims and objectives of this project are detailed in Section 1.5. Finally an outline of the thesis is given in Section 1.6.

1.1. Images in Web Document Analysis

Following the exploding expansion of World Wide Web, the classical definition of what a document is had to change in order to accommodate the new form of electronic document that appeared: the *"Web Document"*. Two distinct descriptions exist for every Web page, one is the code used to produce the output, and the other is the actual output itself. Although either description should ideally be adequate to describe a given Web page, not the same information is stored in each representation. For instance, information about the filenames of images contained in the document is only available in the Web page's code, whereas the images themselves, thus the information they add to the document, are only part of the two-dimensional viewable output. The definition of Web Document is therefore not trivial, and should incorporate both representations.

The key-role images play in Web Pages makes them an important part of every Web Document Analysis method. This stands true for all types of images, as images are used in Web pages for a variety of purposes, such as to define a background pattern for the document, as bullets in a list, deliminators between different sections, photographs, charts etc. The most significant kind of images, in respect to the amount of information they carry, are the ones containing text, such as headers, menus, logos, equations, etc. Images containing text carry important information about both the layout and the content of the document, and special attention should be paid in incorporating this information in every Web Document Analysis method.

1.2. The Use of Images in the Web – A Survey

In order to assess the impact of the presence of text in Web images, the Pattern Recognition and Image Analysis (PRImA) group of the University of Liverpool delivered a survey over the contents of images of around *200* Web pages [9]. The Web pages included in the survey originated from a variety of sites that the average user would be interested in browsing. Sites of newspapers and TV stations, on-line shopping, commercial, academic and other organizations' sites as well as sites dealing with leisure activities, work-related activities and other routine activities were all included, so that the sample set reflects the average usage of World Wide Web today. The Web pages analysed were all in English language, and therefore the majority of sites were from the United States and the United Kingdom. For the purpose of this survey, there should not be loss of generality by this choice.

In order to assess the impact of the presence of text in Web images, certain properties were measured for each image, related both to the textual contents of the images as well as to the contents of the ALT tags associated with each image. The investigation of the alternative text provided is essential, in order to assess the necessity of a method to extract text from images, as this would depend on the availability of an alternative representation of the textual content of the images. In terms of the textual content of the images the following were measured for each Web page:

- The total number of words visible on the page. This is the number of words that can be viewed on the Web page, regardless of whether they are part of the text or embedded in an image.
- The number of words in image form. This is the number of the words that are embedded in images only.
- The number of words in image form, that do not appear anywhere else on the page. This is an important measure, since it indicates whether a word can be indexed and consequently searched upon or not.

Regarding the presence of an alternative description in the form of ALT tags associated with the text in images, the following were measured:

- The number of correct descriptions. A description is considered correct if all the text in the image is contained in the ALT tag.
- The number of false descriptions. These are descriptions were the ALT tag text disagrees with the text in the image.
- The number of incomplete descriptions, where the ALT tag text contains some, but not all of the text in the image.
- The number of non-existent descriptions, where no ALT tag text is provided.

Overall, an average of *17%* of the words in a Web page are embedded in images (see Figure 1-1). This is a significant percentage, considering that text in Web images is inaccessible using current technology, as will be discussed in Section 1.3.

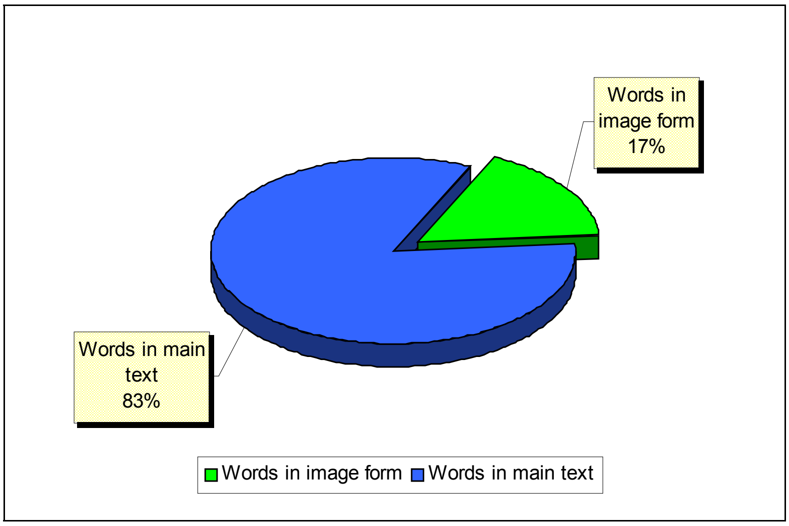


Figure 1-1 – Percentage of words in Web pages found in image form and in the main text.

Even more alarming are the results about the usage of ALT tags. As can be seen in Figure 1-2 that follows, only *44%* of the ALT tag's text is correct. The remaining *56%* is either incomplete (*8%*), false (*3%*), or non-existent (*45%*). That makes more than half of the text in images totally inaccessible without a method to extract it directly from the image.

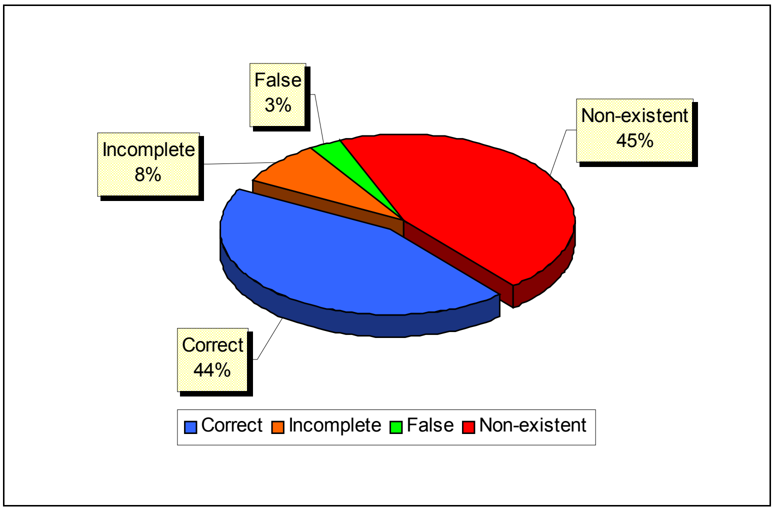


Figure 1-2 – Percentage of correct and incorrect ALT tags in Web pages.

Not encouraging either is the fact that of the total number of words embedded in images, *76%* do not appear anywhere else in the main text (see Figure 1-3). This corresponds to *13%* of the total visible words on a page not accessible for indexing and/or searching purposes.

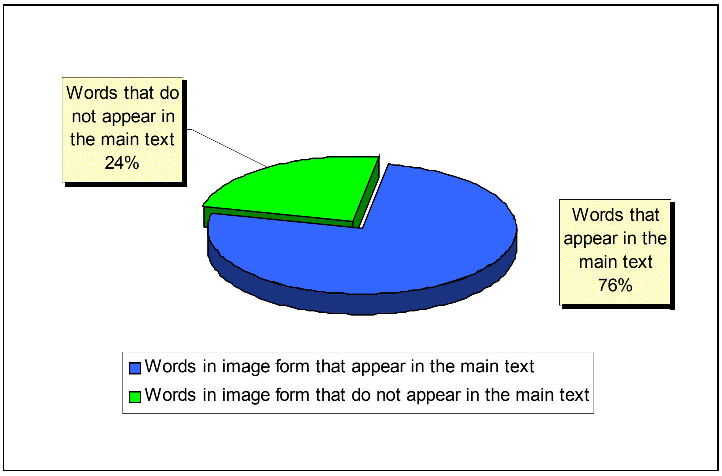


Figure 1-3 – Percentage of words embedded in images not appearing in the main text.

The results reported are in broad agreement with an earlier survey of *100* Web pages by Lopresti and Zhou [218]. This fact demonstrates that the situation is not improving. Furthermore, as the volume of information on the World Wide Web grows the total number of inaccessible material increases considerably. More recently Kanungo, Lee and Bradford [90] reported similar results over a small number of images selected from web pages returned by the *Google* Internet search engine for the keyword “newspapers”. Although a very restricted sample was evaluated, the results are equally alarming, with a *59%* of the words in images not appearing anywhere else in the Web page. Further verification of the situation can be found in Munson and Tsymbanenko [125] who attribute the poor recall results they obtain when searching for images by means of their associated text information, to exactly the fact that the ALT tag attribute value is rarely both present and relevant to the image’s content.

An aspect not covered in any of these surveys, is the incorporation of a measure of importance for each image. Although it is a fact that important entities on Web pages are present in the form of images, a measure of importance for each image in terms of the information it carries would give a clearer view of the problem and should be included in any future work. Various ways exist to measure the importance of each image, most of them based on structural analysis of the Web page. The location of each image is vital, and is a feature incorporated in many new applications. For example, when performing Web page analysis for voice browsing, the location of the image is announced along with any associated caption [21]. Another way to assess the importance of an image could be employing techniques used nowadays for evaluating the quality of Web pages, usually based on human visual perception and eyes movement [57].

1.3. The Need – Applications of the Method

There is a plethora of possible applications for a method that extracts text from colour images. An immediate use of the method would be at the domain of indexing and searching. Search engines are unable to access any textual information found in image form directly. Most search engines index instead any alternative information that is readily available for the image, such as the filename or the ALT tag associated with the image. This, results to important terms of the Web page to be ignored, thus the information finally indexed is incomplete or even misleading if one considers the existence of incorrect alternative descriptions. A method to extract text from the Web images would be useful in many different stages. First, any search engine would be able to access and consequently index the textual information embedded inside Web images. This would result into improved indexing and more efficient and precise searching. Text extraction could also take place at a different stage, at the time of the creation of the document. A text extraction method could be used to automatically provide the correct alternative description for the images that the user has not already supplied one, or to double check the correctness of user inserted alternative descriptions and possibly propose the appropriate changes.

The Web browser could also benefit in many ways from a method to extract text from images. Knowing the textual content of an image, it could provide a number of new functions, such as filtering capabilities or certain accessibility utilities for users with special needs. Filtering would be straightforward if the textual contents of images were known. Images containing certain offensive terms, or terms indicating that they are advertisements could be easily removed from the Web page. If filtering was enabled one stage earlier, for example at the cache server asked for the Web page, filtering of advertisements or other types of images would be possible at the company/provider level, and would also result in lesser downloading times, since only the useful information would be passed on to the final user. As far as it concerns accessibility functions, there is a number of ways in which a text extraction method could prove of great value. Knowing the exact textual content of a Web image, would allow the browser to present the text in a different way, for users with special needs.

For example, the text could be presented enlarged, in an easier to read font, using high contrasting colours.

The most important application of a Web image text extraction method would probably be converting the Web page to voice. Conversion of the Web page to voice is not necessarily a function used only by users with special needs. There are many cases where having the computer read a Web page would save time and allow the user to continue with other activities simultaneously. For example, reading traffic information or a newspaper page while driving, or enabling bi-directional voice browsing. In current applications images are simply ignored, or in very few cases, positioning information or any alternative text is read instead [21].

Converting a Web page to text allows for a series of other applications apart from voice browsing. For example, it would enable applications to summarize Web pages and produce smaller versions, suitable to be presented on the small display of a PDA or a mobile phone. Considering the rapid progress and wide acceptance of mobile technologies, this will probably be an important issue in the near future.

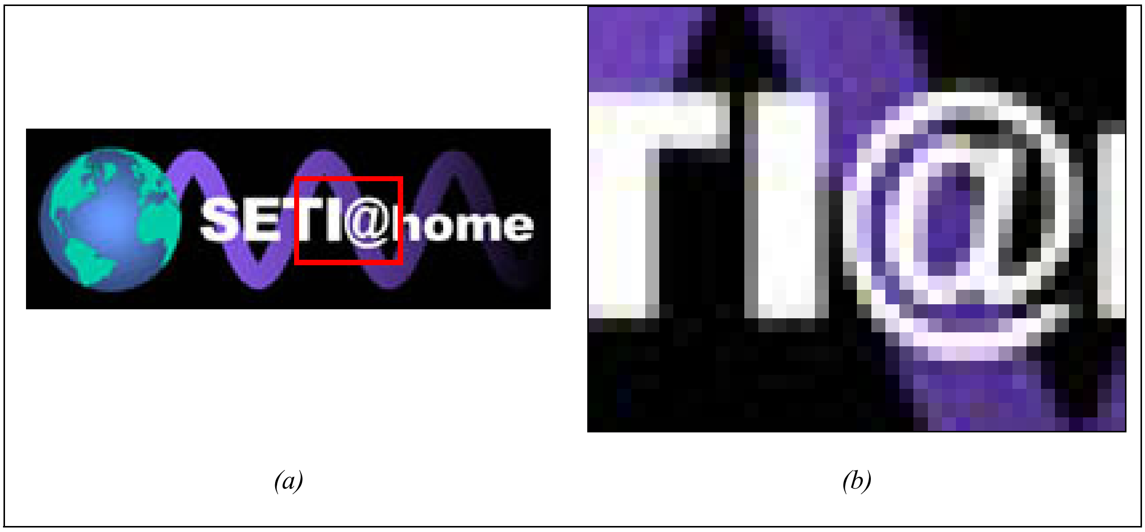
Finally, the applications of a method to extract text from colour images are not necessarily limited to the World Wide Web. There are many other fields that such a method would be of great value, for example video indexing, or real life scenes analysis. The method could be used to extract text from video frames, such as names and titles from newscasts or scores from athletic events. Furthermore, it could be used for applications involving real life scenes, such as extracting the registration numbers of car plates, or the textual content of street and shop signs.

1.4. Working with Web Images – Observations and Characteristics

In this section, certain characteristics of Web images will be discussed derived from observations made over a large number of images that comprise the dataset used for the development and evaluation of the method (described analytically in Chapter 7). The observations made and the characteristics identified, shaped the way this research was conducted.

The basic observation one can make about Web images is that they are computer oriented. Contrary to the types of images that typical OCR applications require, which in most of the cases are scanned documents, Web images are created with the use of computers to be viewed on a computer monitor.

The fact that computers are employed for the creation of the images entails that a plethora of tools are readily available at that stage. These tools include a wide range of filters, antialiasing capabilities, a variety of different effects to render characters and many more. Therefore, although Web images do not suffer from artefacts introduced during the digitisation process (skew, noise etc), artefacts produced by the software used for the creation of the image are evident in most of the cases. Antialiasing is probably the most common kind of artefact that strongly affects our ability to differentiate characters from the background. Antialiasing is the process of blending a foreground object to the background, by creating a smooth transition from the colours of one to the colours of the other. This produces characters with poorly defined edges, in contrast to the characters in typical document images (Figure 1-4). A second, but equally important problem, is the use of special effects when rendering characters in an image. These effects range from simple outlines and shadows, to 3D-effects and the placement of words in arbitrary directions, and pose many difficulties to any text extraction method.



(a) *(b)*

Figure 1-4 – (a) An image with antialiased characters. The area marked with the red rectangle appears in magnification on the left. (b) Magnified part of the image, where the antialiased edges of characters are visible.

As mentioned before, Web Images are designed to be viewed on computer monitors. This entails certain things about the size and the resolution of the images.

Contrary to typical document images, which have a minimum resolution of *300dpi* and a typical size of an *A4* page (thus their dimensions are in the range of thousands of pixels), Web Images have an average resolution of *75dpi*. This is adequate for viewing on a monitor, which is the primary use for those images. Consequently, the size of the images is in most of the cases small. Since the images are to be viewed on a monitor with an average resolution of *800x600* pixels, the dimensions of Web Images are never larger than some hundred pixels. Another observation that would lead us to similar assumptions about the average size of Web Images is the function of the images in the Web pages. Headers, menu items and section titles occupy usually a small area of the page, which is reflected in the size of the images used for these entities. Furthermore, the size of the characters used is also much smaller than the size of characters on scanned documents (Figure 1-5). An expected character size in scanned documents is *12pt* or larger, whereas in Web pages characters can be as small as to an equivalent of *5* to *7pt*.

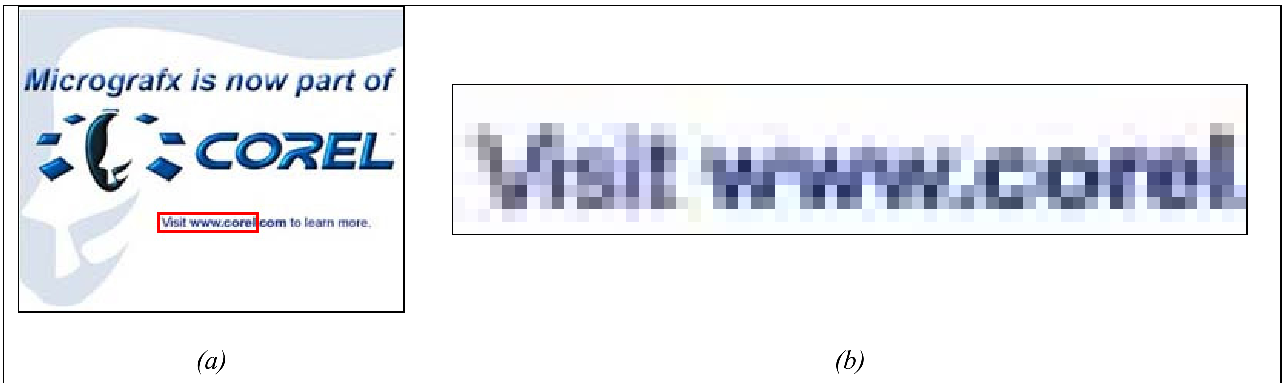
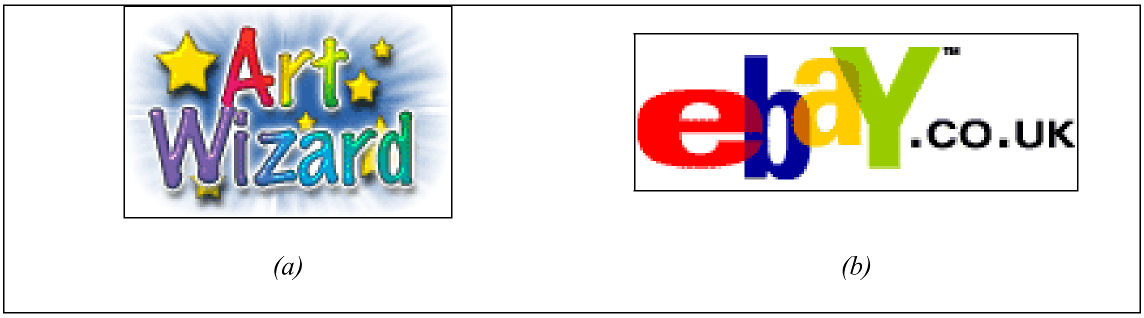


Figure 1-5 – (a) An image containing small characters. (b) Magnified part of the image.

Another important observation is that Web images are used in order to create impact. The ultimate function of images in Web pages is not only to beautify the overall look of the pages, but also to attract viewers. This should be combined with the fact that the creation of Web pages is not limited to professional designers (that could possibly comply to certain designing rules), but essentially open to everyone that owns a computer. Creativity is therefore the only limit when designing a Web page. Consequently, complex colour schemes are used most of the times, resulting to images having multi-coloured text over multi-coloured background (Figure 1-6). One would anticipate that just because creating impact is an important issue, high contrast schemes between characters and background would be used. Although it is expected that the colours of pixels belonging to characters will be more similar between them than to colours of the background, the contrast between the two classes (text and background) is not always adequate to distinguish them.

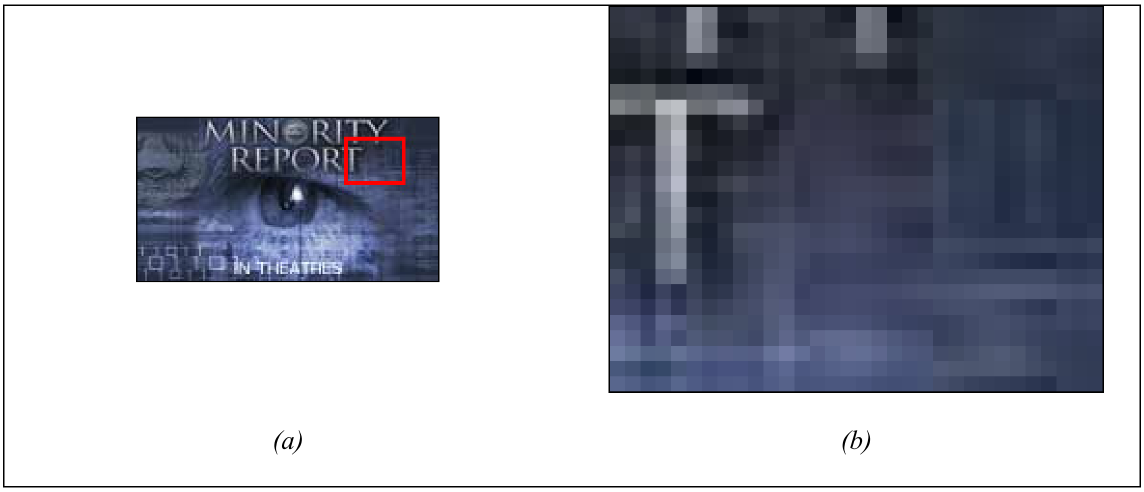


(a)

(b)

Figure 1-6 – (a) An image containing both multi-coloured characters and multi-coloured background. (b) An image with multi-coloured overlapping characters.

A final observation about Web images is that they are created with file-size in mind, since they have to be easily communicated through the Web. This directly suggests that some kind of compression should be used, which is actually the case with Web images most of the times. The vast majority of images in the Web are saved as *JPEG* files, using in some extent *JPEG*'s compression capabilities. This type of compression introduces certain artefacts in the images that have no particular effect in areas of almost constant colour, but can produce significant distortions to characters. This kind of lossy compression is even more noticeable when colour analysis of the image takes place (as lightness information is mostly preserved, but colour information is to a great extent discarded in *JPEG* compression). The next most popular file type used for Web images, is *GIF*. Although *GIF* format preserves much more information than *JPEG*, it is limited to 8-bit colour palettes. This vastly reduces the number of colours in images, and can introduce significant quantization artefacts in the images, as well as dithered colours to appear.



(a) *(b)*

Figure 1-7 – (a) A JPEG compressed image. (b) Magnified part of the image. The block structure produced by JPEG compression is visible in large areas, while more compression artefacts are visible near the characters.

A summary of the main characteristics associated with Web images, and a comparison with typical document images required by *OCR* applications is given in Table 1-1.

Characteristics	***Web Images***	***Typical Scanned Document Images***
Spatial Resolution	~75dpi	>300dpi
Images' Size	100s of pixels	1000s of pixels
Characters' Size	Can be as small as 5-7pt	>12pt
Colour Schemes	Multi-colour text over multi-colour background	Black text over white background
Artefacts	Antialiasing, compression, dithering	Skew, digitisation artefacts
Character Effects	Characters not always on a straight line, 3D-effects, shadows, outlines etc.	Characters usually on a straight line, of the same font

Table 1-1 – Summary of Web Images' main characteristics and comparison to typical scanned document images.

1.5. Aims and Objectives of this Project

This research aims in identifying novel ways to extract text from images used in the World Wide Web. Text extraction comprises of two main steps: Segmentation of the image into regions, and Classification of the produced regions as text or non-text.

The objective of this project is the implementation and testing of new ideas for performing text extraction from web images, as dictated from the specific characteristics of the problem. A successful solution should be able to cope with the majority of web images, producing precise segmentation results and high classification rates. Given the volume of fundamentally different images existent in the World Wide Web, this is expected to be a complicated task.

The final solution should be able to correctly segment images containing text and background of varying colour, mainly gradient and antialiased text. It should be also able to cope with various text layouts (e.g. non-horizontal text orientation)

The methods developed should not set any special requirements for the input images, while the assumptions about the contents of the image should be kept to an absolute minimum.

Given the volume of web images available and the special types of applications where a web image text extraction method is expected to be used, it is important that the execution time is kept short. Nevertheless, since the focus of this research is on the implementation and evaluation of novel methods, the execution time for the prototypes should be allowed to be reasonably long, as long as optimisation and improvement is possible.

The ability of the text extraction method to decide whether an image contains text or not, is a desired property, but considered to be out of the scope of this research. For the evaluation of this research, only images containing text will be employed.

1.6. Outline of the Thesis

Following this introductory chapter, Chapters 2 and 3 provide a theoretical background for this research. Chapter 2 gives a detailed overview of segmentation methods, starting with methods used for greyscale image analysis, followed by a brief introduction to colour and a critical review of colour segmentation methods. Chapter 3 provides an overview of text image segmentation and classification methods. Several aspects of text image segmentation are discussed and a number of classification methods for the segmented regions are presented and appraised. Previous work on the topic of text extraction from colour images is also part of Chapter 3.

Chapter 4 describes in detail the first method used for segmenting text in Web images. This method works in a split-and-merge fashion, making use of the *HLS* colour space. The method's basic concepts and implementation aspects are discussed, and results obtained are presented.

Chapter 5 describes the second segmentation method developed, which is based upon the definition of a propinquity measure that combines colour distance and topological characteristics of connected components with the help of a fuzzy inference system. An analytical description of the fuzzy defined propinquity metric takes place, followed by an explanation of the method and the presentation of results obtained.

The connected component classification method developed is presented in Chapter 6. The classification of the connected components produced by the segmentation methods is a necessary post-processing step aiming in identifying which of them correspond to characters and which to the background of the image.

More results of both segmentation methods and the classification method are presented and critically appraised in Chapter 7. A comparison between the two segmentation methods and the possibility of applying the methods developed to other fields is also included in Chapter 7. The chapter closes with a discussion summarizing on optimisation possibilities and key issues identified. Finally, Chapter 8 concludes this thesis reassessing the aims and objectives set and commenting on future work and open problems.

Appendices on topics not covered in the theoretical chapters of this thesis are also provided. Details on human vision and colorimetry are given in Appendix A. Appendix B is a brief introduction in fuzzy logic and fuzzy inference systems. Finally, Appendix C presents the dataset used, along with the corresponding statistical information.

Chapter 2

Image Segmentation Techniques

"Segmentation is the process of partitioning an image into a number of disjoined regions such that each region is homogeneous with respect to one or more characteristics and the union of no two adjacent regions is homogeneous" [70, 140, 143].

A more mathematical definition of segmentation can be found in [56, 197, 221]. Let *X* denote the grid of sample points of a picture, and let *P* be a logical predicate defined on a set of contiguous picture points. Then segmentation can be defined as a partitioning of *X* into disjoint non-empty subsets $X_1, X_2, \ldots X_n$ such as:

1. $\bigcup_{i=1}^{N} X_i = X$
2. X_i for *i*=1,…*N* is connected
3. $P(X_i) = TRUE$ for *i*=1,…*N*
4. $P(X_i \cup X_j) = FALSE$, for *i≠j* where X_i and X_j are adjacent.

Eq. 2-1

The first condition states that after the completion of the segmentation process, every single pixel of the image must belong to one of the subsets X_i. It should be noted, that the condition:

$$\bigcap_{i,j=1}^{N} X_i, X_j = 0,\ i \neq j \qquad \textit{Eq. 2-2}$$

even though not mentioned, should also be in effect for the above definition to be complete. That last condition states that no pixel of the image can belong to more than one subset.

The second condition implies that regions must be connected, i.e. composed of contiguous lattice points. This is a very important criterion since it affects the central structure of the segmentation algorithms [119, 197] . This second condition is not met in many approaches [31, 81, 141, 173].

A uniform predicate *P* as the one associated with the third condition is according to Fu and Mui [56] one which assigns the value *TRUE* or *FALSE* to a non-empty subset *Y* of the grid of sample points *X* of a picture, depending only on properties related to the brightness matrix for the points of *Y* . Furthermore, *P* has the property that if *Z* is a non-empty subset of *Y*, then *P(Y)=TRUE*, implies always that *P(Z)=TRUE*. The above definition is rather restricted in the sense that the only feature on which *P* depends on is the brightness of the points of the subset *Y*. In Tremeau and Borel [197] this is generalized in order to address other characteristic features of the subset pixels as well. According to them, the predicate *P* determines what kind of properties the segmentation regions should have, for example a homogeneous colour distribution.

Finally, the fourth condition entails that no two adjacent regions can have the same characteristics.

Segmentation is one of the most important and at the same time difficult steps of image analysis. The immediate gain of segmenting an image is the substantial reduction in data volume. For that reason segmentation is usually the first step before a higher level process which further analyses the results obtained.

Segmentation methods are basically ad hoc and their differences emanate from each problem's trait. As a consequence, a variety of segmentation methods can be found in literature, the vast majority of which addresses greyscale images.

In the present chapter an overview of some of the existing methods for segmentation will be attempted, paying particular attention to segmentation methods created for colour images. The next section gives a brief anaphora on greyscale segmentation methods, followed by the main section of this chapter, which focuses on colour image segmentation.

2.1. Greyscale Segmentation Techniques

Fu and Mui [56] in their survey on image segmentation classified image segmentation techniques as characteristic feature thresholding or clustering, edge detection and region extraction. This survey was done from a biomedical viewpoint, and the evaluation of techniques is based on cytology images. Authors' comments are objective, but the main interest is clearly cytology imaging. A review of several methods for thresholding and clustering, edge detection and region extraction was performed. Most of the methods reviewed are towards greyscale segmentation, with the exception of Ohlander's work [134] on colour image thresholding and a clustering method proposed by Mui, Bacus and Fu [124], which only uses colour-density histograms at a later stage, after initial segmentation has already been obtained.

Haralick and Shapiro [69] categorized segmentation techniques into six classes: measurement space guided spatial clustering, single linkage region growing schemes, hybrid linkage region growing schemes, centroid linkage region growing schemes, spatial clustering schemes and spit and merge schemes. The survey mainly focuses on the first two classes of segmentation techniques; measurement space guided spatial clustering and region growing techniques for which a good summary of different types of region growing methods has been presented. The recursive clustering method proposed by Ohlander [134] and the work of Ohta, Kanade and Sakai on colour variables [136] are detailed among others in the section concerning clustering. There is also a small section about multidimensional measurement space clustering, where Haralick and Shapiro propose to work in multiple lower order projection spaces and then reflect these clusters back to the full measurement space.

Pal and Pal [140] have made a somewhat more complete review of image segmentation techniques. They cover areas not addressed in previous surveys such as fuzzy set and neural networks based segmentation techniques as well as the problem of objective evaluation of segmentation results. Furthermore, they consider the segmentation of colour images and range images (basically magnetic resonance images). They identify two approaches for segmentation: classical approach and fuzzy mathematical approach, each including techniques based on thresholding, edge detection and relaxation. The paper attempts a critical appreciation of several methods in the categories of grey level thresholding, iterative pixel classification, surface based segmentation, colour image segmentation, edge detection and fuzzy set based segmentation.

Additional image segmentation surveys can be found in Zucker [221], Riseman and Arbib [158] and Kanade [89]. Also surveys on threshold selection techniques can be found in Weszka [204] and Sahoo *et al.* [172].

Following the suggestions of previous reviews, segmentation methods will be categorized as thresholding or clustering methods, edge detection methods and region extraction methods. In addition to these, there are certain methods that do not fall clearly in any of the above classes, rather combine two or more of the aforementioned techniques to achieve segmentation. The remainder of this section will address these categories of segmentation techniques, attempting a critical review of selected methods found in literature.

2.1.1. Thresholding and Clustering Methods

According to Haralick and Shapiro [70] "Thresholding is an image point operation which produces a binary image from a grey-scale image. A binary one is produced on the output image whenever a pixel value on the input image is above a specified minimum threshold level. A binary zero is produced otherwise". There are cases where more than one threshold is being used for segmentation. In these cases, the process is generally called multi-level thresholding, or simply multi-thresholding. Multi-thresholding thus is defined as "a point operator employing two or more thresholds. Pixel values which are in the interval between two successive threshold values are assigned an index associated with the interval" [70]. Multi-thresholding is described mathematically as:

$$S(x,y) = k \text{ , } \mathit{if} \; T_{k-1} \le f(x,y) < T_k \text{ , } k=0,\ 1,\ 2,\ \ldots,\ m \qquad \textit{Eq. 2-3}$$

where *(x, y)* is the *x* and *y* co-ordinate of a pixel, *S(x, y)*, *f(x, y)* are the segmented and the characteristic feature functions of *(x, y)* respectively, T_0, …, T_m are threshold values with T_0 equal to the minimum and T_m the maximum and *m* is the number of distinct labels assigned to the segmented image [56]. Bi-level thresholding can be seen as a special case of multi-thresholding for $m=2$, thus two intervals are defined one between T_0 and T_1 and the second between T_1 and T_2.

Level slicing or density slicing is a special case of bi-level thresholding by which a binary image is produced with the use of two threshold levels. "A binary one is produced on the output image whenever a pixel value on the input image lies between the specified minimum and maximum threshold levels" [70].

Based on the aforementioned definition a threshold operator T can be defined as a test involving a function of the form [60]:

$$T(x, y, N(x, y), f(x, y)) \qquad \textit{Eq. 2-4}$$

where $N(x, y)$ denotes some local property of the point (x, y) for example the average grey level over some neighbourhood. Depending on the functional dependencies of the threshold operator T, Weszka [204] and Gonzalez and Woods [60] divided thresholding into three types. The threshold is called global if T depends only on $f(x, y)$. It is called a local threshold when T depends on both $f(x, y)$ and $N(x, y)$. Finally if T depends on the coordinate values x, y as well, then it is called a dynamic threshold.

A global threshold can be selected in numerous ways. There are threshold selection schemes employing global information of the image (e.g. the histogram of a characteristic feature of the image, such as the grey scale value) and others using local information of the image (e.g. co-occurrence matrix, or the gradient of the image).

Taxt *et al* [192] refer to these threshold selection schemes using global and local information as contextual and non-contextual thresholding respectively. According to Taxt *et al*, under these schemes, if only one threshold is used for the entire image then it is called global thresholding, whereas if the image is partitioned into sub-regions and a threshold is defined for each of them, then it is called local thresholding [192], or adaptive thresholding [214]. For the rest of this section, the definitions given by Weszka [204] and Gonzalez and Woods [60] will be used.

Global threshold selection methods

If there are well defined areas in the image having a certain grey-level value, then that would be reflected in the histogram to well separated peaks. In this simple case, threshold selection becomes a problem of detecting the valleys of the histogram. Unfortunately, more often than not, this is not the case. Most of the times the different modes in the histogram are not nicely separated by a valley or the valley may be broad and flat, impeding the selection of a threshold.

A number of methods are proposed that take into account some extra information about the image contents in order to make a threshold selection. For bi-modal pictures, Doyle [44] suggested the *p-tile* method which chooses as a threshold the grey-level which most closely corresponds to mapping at least *(1-p)* per cent of the grey level pixels into the object. If for example we have a prior knowledge that the objects occupy *30%* of the image, then the image should be thresholded at the largest grey-level allowing at least *30%* of the pixels to be mapped into the object. Prewitt and Mendelsohn [155] proposed a technique called the mode method, which involves smoothing of the histogram into a predetermined number of peaks. The main disadvantage of these methods is that knowledge about the class of images we are dealing with is required.

A number of other methods have also been proposed to separate the different modes in the histogram. Nakagawa and Rosenfeld [130] assumed that the object and background populations are distributed normally with distinct means and standard deviations. They then selected a threshold by minimizing the total misclassification error. Pal and Bhandari [139] assumed Poisson distributions to model the grey-level histogram and proposed a method which optimises a criterion function related to average pixel classification error rate, finding out an approximate minimum error threshold.

Local threshold selection methods

A common drawback of all these methods is that they totally ignore any spatial information of the image. A number of methods were proposed that combine histogram information and spatial information to facilitate threshold selection. Weszka [205] proposed a way to sharpen the valley between the two modes for bi-modal images, by histogramming only those pixels having high Laplacian magnitude. Such a histogram will not contain those pixels in between regions, which help to make the histogram valley shallow.

Weszka and Rosenfeld [206] introduced the busyness measure for threshold selection. For any threshold, busyness is the percentage of pixels having a neighbour whose thresholded value is different from their own thresholded value. Busyness is dependent on the co-occurrence of adjacent pixels in the image. A good threshold is the point near the histogram valley, which minimizes the busyness.

Watanabe *et al* [201] used the gradient property of the pixels to determine an appropriate threshold. For each grey-level they compute the sum of the gradients taken over all pixels having that specific grey-level value. The threshold is then chosen at the grey-level for which this sum of gradients is the highest. They reason that since this grey-level has a high proportion of large difference points, it should occur at the borders between objects and background. Kohler [95] suggests a modification of the Watanabe's idea. He chooses that threshold which detects more high-contrast edges and fewer low-contrast edges than any other threshold. Kohler defines the set of edges detected by a threshold T to be the set of all pairs of neighbouring pixels, for which one pixel has a grey-level value less than or equal to T and the other has a grey value greater than T. The contrast of an edge comprising of pixels a and b is given as $min\{|I(a) - T|, |I(b) - T|\}$, where $I(a)$ and $I(b)$ is the grey-level value (intensity) of the pixels a and b respectively. The best threshold is determined by that value that maximizes the average contrast, which is given as the sum of the contrasts of the edges detected by that threshold, divided by the number of the detected edges.

Adaptive thresholding

In cases where the image in question is noisy or the background is uneven and illumination variable, objects will still be darker or lighter than the background but any fixed threshold level for the entire image will usually fail to separate the objects from the background. In that cases adaptive thresholding algorithms must be used. As mentioned before in adaptive thresholding the image is partitioned into several blocks and a threshold is computed for each of these blocks independently. As an example of adaptive thresholding, Chow and Kaneko [34] determined local threshold values for each block using the sub-histogram of the block. Then they spatially interpolated the threshold values to obtain a spatially adaptive threshold for each pixel.

Clustering

Clustering, in the context of image segmentation, is the multidimensional extension of the concept of thresholding. Typically, two or more characteristic features are used and each class of regions is assumed to form a distinct cluster in the space of these characteristic features.

A cluster according to Haralick and Shapiro [70] is a homogeneous group of units, which are very like one another. Likeness between units is determined by the association, similarity, or distance between the measurement patterns associated with the units. Clustering is then defined as the problem concerned with the assignment of each observed unit to a cluster.

Haralick and Shapiro [69] state that the difference between clustering and segmentation is that in clustering, the grouping is done in measurement space, while in segmentation the grouping is done in the spatial domain of the image. This is partially misleading since segmentation aims to do groupings in the spatial domain, but this can be achieved through groupings in the measurement space. The resulting clusters in the measurement space are then mapped back to the original spatial domain to produce a segmentation of the image.

Any feature that one thinks is helpful to a particular segmentation problem can be used in clustering. Characteristic features commonly used for clustering include grey values taken with the use of different filters [59], or even features defined for the specific problem. As an example, Carlton and Mitchell [27] used a texture measure that counted the number of local extrema in a window centred at each pixel. They created three grey-level images using different thresholds. The grey-level value of the units in each of these resulting images is the number of local extrema produced by the used threshold. These three resulting images along with the original intensity image were used to define a four-dimensional space in which the clustering was performed.

Clustering is mainly used for multi-spectral images. Consequently, clustering techniques are broadly used in colour image segmentation. As such, a detailed discussion of clustering methods will follow in chapter 2.2.3.

2.1.2. Edge Detection based Methods

Edge detection based segmentation techniques are based on the detection of discontinuities in the image. Discontinuities in a grey-level image are defined as abrupt changes in grey-level intensity values. Edges are therefore defined as the points where significant discontinuities occur.

The importance of edge detection lies to the fact that most of the information of an image lies on the boundaries between different regions, and that biological visual systems seem to make use of edge detection rather than thresholding [165].

Ideally, edge detection should yield pixels lying only on the boundary between the regions one tries to segment. In practice, the identified pixels alone rarely define a correct boundary sufficiently because of noise, breaks in the boundary and other effects. More often than not, the edge detection stage is followed by an edge linking and boundary detection stage, intending to combine pixels into meaningful boundaries.

According to Davis [39], edge detection methods can be classified as parallel and sequential. In parallel techniques, the decision whether a pixel is an edge pixel or not is not dependant on the result of the operator on any previously examined pixels, whereas in sequential techniques it is. The edge detection operator in parallel techniques can therefore be applied simultaneously everywhere in the image.

As far as it concerns sequential techniques, their outcome is strongly dependent on the choice of an appropriate starting point and on the way previous results affect both the selection and the result at the next pixel. Kelly [92] and Chien and Fu [33] used guided search techniques for this. The rest of this section will focus on parallel techniques.

Since discontinuities in image are associated with high frequencies, an obvious way to enhance the edges would be high frequency filtering [60]. If for example we take the Fourier transform of the image and multiply it by a spatial filter, then the invert transform will yield enhanced edges. The design of the filter used is the crucial part for this process.

Local operators

Another common idea underlying many edge detection techniques is the computation of a local derivative operator. The first derivative of an image would feature local extrema at the points of transitions between regions of constant grey-level and zero plateaus at points in regions of constant grey-level. The second derivative is zero in areas of constant grey-level and presents zero crossings at the midpoints of transitions (Figure 2-1).

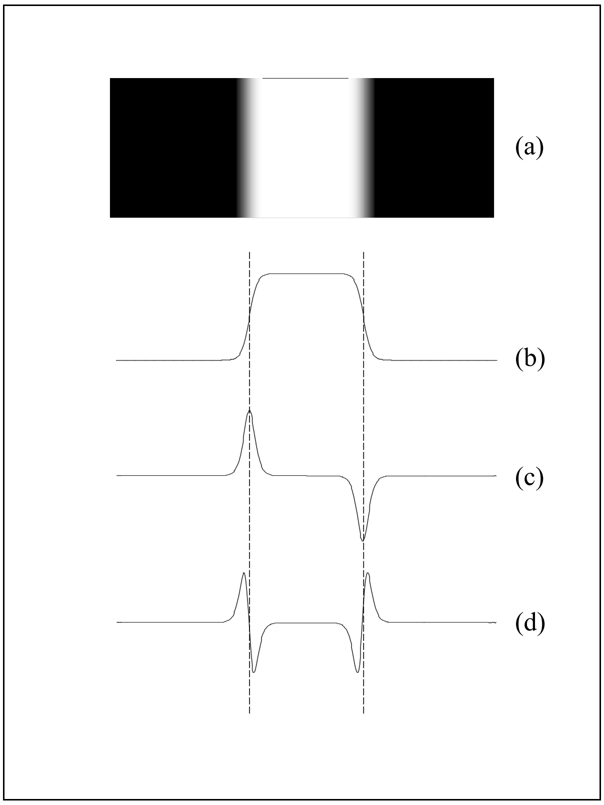


Figure 2-1 – (a) An image of a vertical white stripe over black background. (b) Profile of an horizontal line of the image, note the gradient edges of the stripe, which is usually the case with edges in images. (c) First derivative of the horizontal profile, local extrema appear at the points of transitions. (d) Second derivative of the horizontal profile, zero crossings appear at the midpoints of transitions.

The first derivative at any point in an image is obtained by using the magnitude of the gradient vector at that point. The gradient vector of an image at location *(x, y)* is defined as:

$$\overrightarrow{\nabla f} = \begin{bmatrix} G_x \\ G_y \end{bmatrix} = \begin{bmatrix} {}^{\partial f}\!/_{\partial x} \\ {}^{\partial f}\!/_{\partial y} \end{bmatrix} \qquad \textit{Eq. 2-5}$$

The magnitude of the gradient vector is generally referred to simply as the *gradient*.

There is a number of operators approximating the first derivative such as Prewitt [154, 186], Sobel [45, 60, 186], Robinson [186], and Kirsch [94, 186] operators (Figure 2-2), which are based on a *3x3* neighbourhood. The Sobel operator has the advantage of providing both a differencing and a smoothing effect, which is a particularly attractive feature, since derivatives enhance noise. Prewitt's operator is also adequately noise-immune, but it has a weaker response to the diagonal edge.

Direction	↑	↗	→	↘
Prewitt	$\begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix}$	$\begin{bmatrix} 0 & 1 & 1 \\ -1 & 0 & 1 \\ -1 & -1 & 0 \end{bmatrix}$	$\begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix}$	$\begin{bmatrix} -1 & -1 & 0 \\ -1 & 0 & 1 \\ 0 & 1 & 1 \end{bmatrix}$
Sobel	$\begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$	$\begin{bmatrix} 0 & 1 & 2 \\ -1 & 0 & 1 \\ -2 & -1 & 0 \end{bmatrix}$	$\begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$	$\begin{bmatrix} -2 & -1 & 0 \\ -1 & 0 & 1 \\ 0 & 1 & -2 \end{bmatrix}$
Robinson	$\begin{bmatrix} 1 & 1 & 1 \\ 1 & -2 & 1 \\ -1 & -1 & -1 \end{bmatrix}$	$\begin{bmatrix} 1 & 1 & 1 \\ -1 & -2 & 1 \\ -1 & -1 & 1 \end{bmatrix}$	$\begin{bmatrix} -1 & 1 & 1 \\ -1 & -2 & 1 \\ -1 & 1 & 1 \end{bmatrix}$	$\begin{bmatrix} -1 & -1 & 1 \\ -1 & -2 & 1 \\ 1 & 1 & 1 \end{bmatrix}$
Kirsch	$\begin{bmatrix} 3 & 3 & 3 \\ 3 & 0 & 3 \\ -5 & -5 & -5 \end{bmatrix}$	$\begin{bmatrix} 3 & 3 & 3 \\ -5 & 0 & 3 \\ -5 & -5 & 3 \end{bmatrix}$	$\begin{bmatrix} -5 & 3 & 3 \\ -5 & 0 & 3 \\ -5 & 3 & 3 \end{bmatrix}$	$\begin{bmatrix} -5 & -5 & 3 \\ -5 & 0 & 3 \\ 3 & 3 & 3 \end{bmatrix}$

Figure 2-2 – 3x3 Convolution Masks of some operators approximating the first derivative. Four out of the eight directions are depicted.

Gradient operators are widely used, since they comprise a straightforward, computationally cheap parallel process. Local operators based on the second derivative have also been proposed. The Laplacian of a function is a second order derivative defined as:

$$\nabla^2 f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2} \qquad \textit{Eq. 2-6}$$

The Laplacian operator may be implemented in various ways. It presents a number of disadvantages. Being a second order derivative operator, the Laplacian operator is very sensitive to noise. Furthermore, it produces double edges.

A good edge detector should be a filter able to act at any desired scale, so that large filters can be used to detect blurry shadow edges and small ones to detect sharply focused fine details. Marr and Hildreth [113] proposed the Laplacian of

Gaussian (*LG*) operator as one that satisfies the above statement. The Gaussian part of the *LG* operator smoothes the image at scales according to the scale of the Gaussian. Then the zero crossings in the output image produced by the Laplacian part of the operator indicate the positions of edges. The space described by the scale parameter of the Gaussian and the zero crossing curves of the output image is known as the scale-space. Techniques based on scale-space analysis [26, 207] are also used to identify interesting peaks in histograms and will be discussed again in section 2.2.2.

The Laplacian of a two dimensional Gaussian function of the form of Eq. 2-7 is given in Eq. 2-8. In Figure 2-3 the plot of the Laplacian of the Gaussian is shown, as well as a cross-section. The zero crossings of the function are at $r=\pm\sigma$, where σ is the standard deviation of the Gaussian.

$$h(x,y)=\exp\left(-\frac{r^2}{2\sigma^2}\right),\quad r^2=x^2+y^2 \qquad \textit{Eq. 2-7}$$

$$\nabla^2 h=\frac{1}{\sigma^2}\left(\frac{r^2}{\sigma^2}-1\right)\exp\left(-\frac{r^2}{2\sigma^2}\right) \qquad \textit{Eq. 2-8}$$

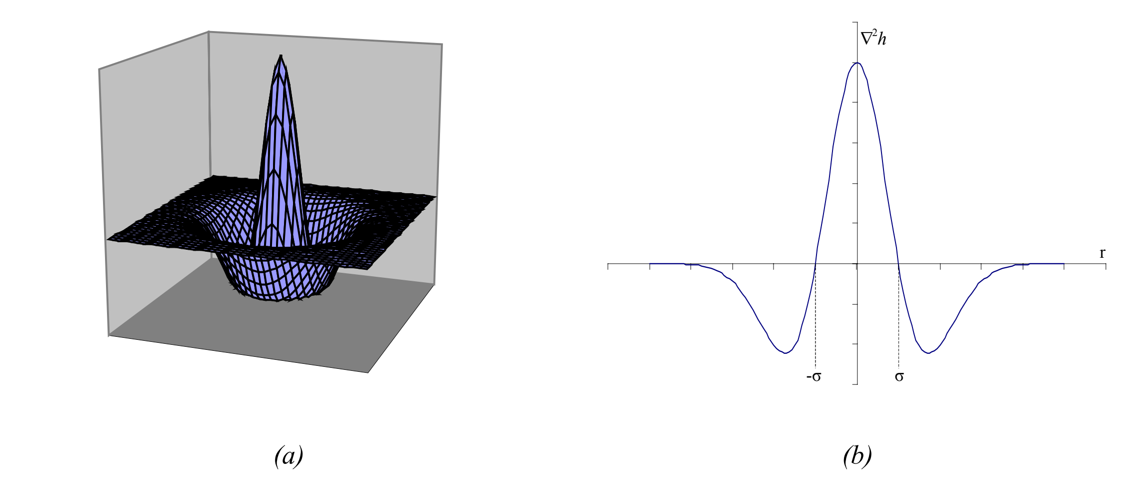


Figure 2-3 – (a) A plot of the Laplacian of the Gaussian ($\nabla^2 h$); (b) a cross-section of $\nabla^2 h$.

The importance of Marr and Hildreth’s method lies partially to the introduction of the concept of scale to edge detection, a concept that has been widely used since [51, 54]. Lu and Jain [108] studied the behaviour of edges in the scale-space using the *LG* operator, aiming to derive useful rules for scale-space reasoning and for high-level computer vision works. They gave a number of interesting theorems, facts and assertions, and analytically described the behaviour of edges in scale-space.

Functional approximation techniques

Apart from local operator based methods, there are a number of methods proposed which consider edge detection as an approximation problem. Hueckel [78] modelled the ideal edge by a step function and considered the actual edges in the image as noisy forms of this ideal model. Hueckel's ideal edge was given by:

$$F(x, y, c, s, p, b, d) = \begin{cases} b & if \quad cx + sy \leq p \\ b + d & if \quad cx + sy > p \end{cases} \qquad \textit{Eq. 2-9}$$

The best fit would be an edge of the above form that minimized a measure of closeness *E* between itself and the actual edge in the image. The measure chosen was the square of the Hilbert distances between *F* and *f* - where *f(x, y)* is the grey value of the pixel *(x, y)* - over a circular neighbourhood *D* around the point in question:

$$E = \int_D \left[f(x, y) - F(x, y, c, s, p, b, d) \right]^2 \, dx\, dy \qquad \textit{Eq. 2-10}$$

The results of the minimization procedure were the best fit and a measure of goodness for the fit.

Elder and Zucker [49] state that the model of edges as large step changes in intensity would fail to detect and localize edges in natural scene images. They reason that a number of natural phenomena such as focal blurring, penumbral blurring and shading would produce a blurred edge rather than an ideal step response. They state that all these situations predict a sigmoidal luminance transition of the same form:

$$I(x) = f(x / r) \text{ , where } f(u) = \frac{1}{\pi}\left(\arccos(u) - u\sqrt{1 - u^2}\right) \qquad \textit{Eq. 2-11}$$

where *r* determines the degree of blur in the edge. Elder and Zucker constructed a model for the edge that encompasses a base step function for the edge, a Gaussian blurring kernel to model the blur and a sensor noise function to model the zero-mean white noise introduced by the sensor.

They state that because of a broad range of conditions, a number of edges having variable characteristics will be produced. Regardless of the physical structure from which they project, all edges should generally project to the image as sigmoidal luminance transitions according to the model proposed, over a broad range of blur scales. Elder and Zucker agree thus with other researchers [14, 104, 108, 113, 207] that operators of multiple scales must be used, since the appropriate spatial scale for local estimation depends on the local structure of the edge, and varies unpredictably over the image. They propose a way to define a locally computable minimum reliable scale (where reliable is defined as the minimum scale for which the likelihood of error due to sensor noise is below a standard tolerance) for local estimation at each point in the image, based on prior knowledge of sensor properties.

Haralick [67, 68] assumed that the intensity function of the image could be fitted with a number of sloped planes according to the gradient around each pixel, then edges were identified as the points having significantly different planes on either side of them. He attempted to least squares fit each pixel neighbourhood with a cubic polynomial in two variables. Having done that, the first and second derivatives could be easily calculated from the polynomial. The first partial derivatives determine the gradient direction. Given the gradient direction, the second directional derivative is used to determine whether a pixel is an edge location or not.

Canny edge detector

Canny [25] tried to mathematically derive an edge detector based on a set of certain rules governing the computation of edge points. According to Canny, a good edge detector should comply with three basic criteria. First, there should be a low probability of failing to mark real edge points, and low probability of falsely marking non-edge points. Thus, the edge detector should present good detection. The second criterion for the edge detector is that the points marked as edges should be as close as possible to the centre of the true edge. This is called good localization. Finally, the edge detector should give only one response to a single edge. Canny used the signal to noise ratio to mathematically model the first criterion about good detection. By maximizing the signal to noise ratio, good detection is ensured. The measure used for the localization criterion was the reciprocal of the root-mean-squared distance of the marked edge from the centre of the true one. This measure increases as localization improves. To maximize simultaneously both good detection and good localization criteria, Canny used the product of signal to noise ratio and the reciprocal of standard deviation of the displacement of edge points. The third criterion, the single-response one, although implicitly captured in the first criterion, it was not captured in the mathematical form of it, and that was the reason why it had to be included as a third distinct criterion. Canny introduced a functional constraint, which eliminated multiple responses. The maximization of the product is done subject to this constraint.

To generalize in two dimensions, Canny defined edges at directional maxima of the first derivative of the intensity function of the image and proposed a complex system of rules to combine edges detected at multiple scales. Canny's operator was essentially one-dimensional, and was shown less accurate for orientation estimation of two-dimensional edges [109].

Edge linking with local processing techniques

As mentioned before, edge linking usually follows the edge detection process. This edge linking (or boundary detection) process, aims to combine individual identified edge pixels into meaningful edges. The most straightforward way to do this, is to analyse the characteristics of pixels in a small neighbourhood around every identified edge pixel in the image [60], and link similar edge pixels. Similarity in that case, is generally based on two properties: the strength of the response of the gradient operator used to produce the edge pixel (or any other metric indicating the strength of the edge pixel), and the direction of the gradient. In the case where a gradient operator was used to identify edge pixels, the strength of the response and the direction of the gradient are derived from the definition of the gradient vector [Eq. 2-5] and are given by equations:

$$\nabla f = mag(\overrightarrow{\nabla f}) = \left[G_x^2 + G_y^2\right]^{1/2} \text{ or } \nabla f \approx |G_x| + |G_y| \qquad \textit{Eq. 2-12}$$

$$\alpha(x,y) = \tan^{-1}\left(\frac{G_x}{G_y}\right) \qquad \textit{Eq. 2-13}$$

Further processing usually consists of linking edge elements separated by small gaps and removing short segments that are unlikely to be part of an important boundary. Local processing methods for edge linking are usually generic and they do not give good results in many situations. More often than not methods aiming to identify certain shapes in the set of produced edge pixels are used.

Line and curve fitting techniques for edge linking

Lowe [107] describes a method to recognize three-dimensional objects from single two-dimensional images. His method is based on matching certain structures of identified lines with the three-dimensional description of the model in question. In order to group the edge points resulting from an edge detection process into lines, Lowe proposed a method based on a significance metric for each line fit. The significance of a straight line fit to a list of points was estimated by calculating the ratio of the length of the line segment divided by the maximum deviation of any point from the line. This measure remains constant under different scales of the image. A line segment is then recursively subdivided at the point of maximum deviation, giving two smaller line segments, and the process is repeated until no line segment is larger than *4* pixels. A binary tree of possible subdivision for each line segment is thus created, and the significance of each sub-segment is calculated. Then following a bottom-up procedure a decision is made at each junction as to whether to keep the two sub-segments or replace them with the higher-order one. If the significance of any of the sub-segments is greater than the significance of the complete segment, then the sub-segments are preferred. This is a very compact approach, which manages to approximate any set of linked pixels with a number of line segments, independently from the scale of the image in hand.

An extension of the above algorithm was proposed by Rosin and West [166]. They suggested that circular arcs could be detected in the straight-line description using a similar algorithm to find the best fit of arcs and straight lines to the data. At each level of recursion, a decision is taken as to whether an arc is a better fit for the given set of points than a lower level description consisting of straight lines and arcs. Rosin and West used the reciprocal of Lowe’s significance metric, so that division by zero could be avoided when all the pixels lay directly on the straight line, thus the maximum deviation is zero. This proved to happen often with short lines. For this metric, a lower significance value would indicate a more significant line. The significance metric for an arc is calculated similarly to the significance of line, as the maximum deviation of any point from the circle divided by the length of the arc segment. This is equivalent to the significance measure for straight lines, allowing direct comparison between the two.

Hough transform based techniques for edge linking

When the image consists of objects of known shape and size, locating those objects is generally a trivial task. In the vast majority of cases though, the actual objects in the image differ substantially from the description given, due to rotation, zoom, and certain shape distortions. In these cases even if an analytical expression of the shape is known beforehand, matching it with the edge pixels identified in the image is a complicated task.

Suppose for example, the simple case where a number of edge pixels have been identified and the goal is to find lines that best fit on these pixels. The straightforward way to achieve it, would be to compute all the lines determined by every pair of pixels, and check the proximity of the rest of the pixels to each line. This is a computational prohibitive approach. One very effective method to solve the problem was proposed by Hough [77].

There are an infinite number of lines passing through a specific point on a plane. Each line can be described by it's slope and intercept. The slope and intercept of every line passing through a specific pixel in the image plane would produce a straight line in the slope-intercept space. The Hough transform, involves transforming each identified pixel in the image into its equivalent straight line in the slope-intercept space. Rosenfeld [162] described a method due to Hough for replacing the problem of finding collinear pixels in the image, by the equivalent one of finding concurrent lines in the slope-intercept space. In general, the n lines in the slope-intercept space would intersect in $n(n-1)/2$ points, corresponding to the lines between each pair of points in the image plane (Figure 2-4). Finding exactly collinear sets of image points is possible by finding coincident points of intersection in the slope-intercept space, but this approach would be computationally exhaustive. Instead, Hough proposed to specify an acceptable error in the line parameter values, thus quantize the slope-intercept space in a number of cells. This array of cells is called the accumulator array. The method described reduces significantly the computational weight of working directly into the image space, while it is insensitive to missing parts of lines, noise and other non-line structures present in the image.

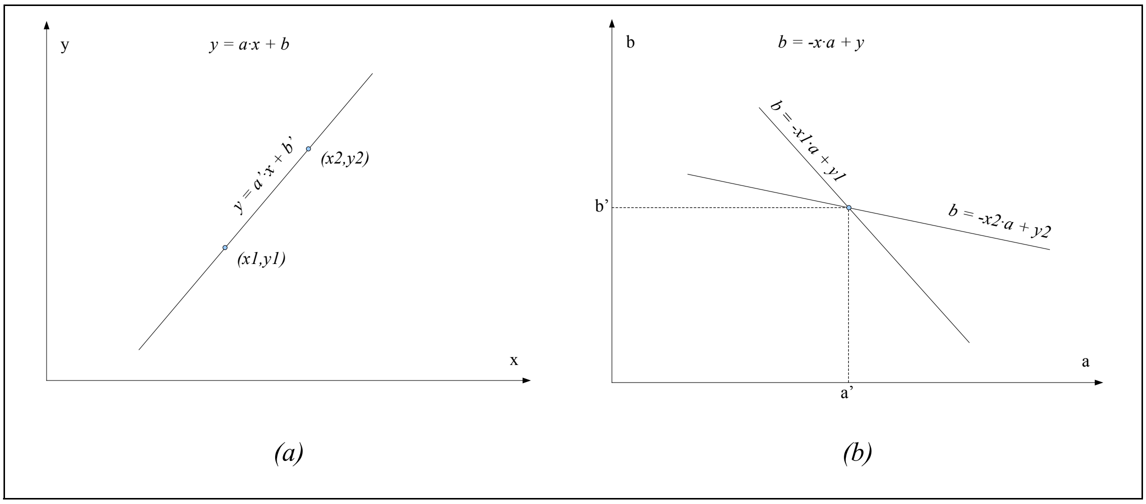


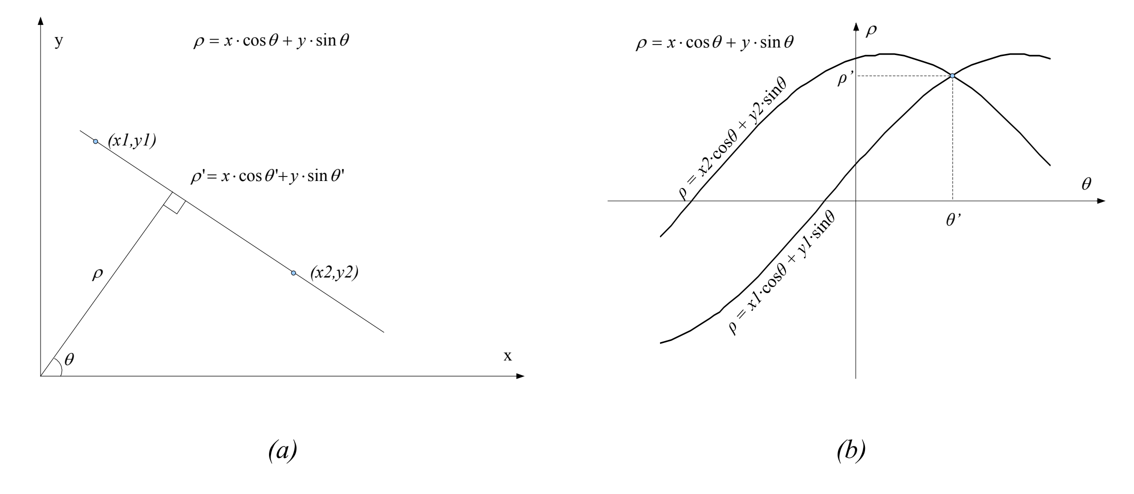
Figure 2-4 – (a) Image space for Hough transform; (b) Slope-intercept space for Hough transform.

One basic problem with that approach, is that both slope and intercept, which were the parameters used by Hough are unbounded (slope is infinite for vertical lines), complicating the application of the technique. Duda and Hart [46], proposed the use of a different parameter space. They proposed the use of the so-called normal parameterisation to describe a line. According to that, each line can be described by the angle of its normal θ and the algebraic distance from the origin ρ. Using these parameters, the equation of a line would be given by:

$$\rho = x \cdot \cos\theta + y \cdot \sin\theta \qquad \textit{Eq. 2-14}$$

$$0 \le \theta < \pi\,,\ -\sqrt{2}D \le \rho < \sqrt{2}D \qquad \textit{Eq. 2-15}$$

If θ is restricted to the interval [*0, π*] then the normal parameters for a line are unique. Each line in the image space would correspond to a sinusoidal curve, given by Eq. 2.1-10. Duda and Hart, transformed each identified pixel in the image to the corresponding sinusoidal curve in the parameter space. Collinear pixels in the image space, would correspond to two intersecting sinusoidal curves in the parameter space, thus the problem again was converted in finding concurrent curves in the parameter space (Figure 2-5). The advantage of using the normal parameterisation to describe lines, is that both θ and ρ can be confined as shown in Eq. 2-15 where D is the distance between corners in the image, since points outside this rectangle correspond to lines outside the image in the image plane.


y
$\rho = x \cdot \cos\theta + y \cdot \sin\theta$
(x1,y1)
$\rho' = x \cdot \cos\theta' + y \cdot \sin\theta'$
ρ
(x2,y2)
θ
x
$\rho = x \cdot \cos\theta + y \cdot \sin\theta$
ρ
ρ'
$\rho = x2\cdot\cos\theta + y2\cdot\sin\theta$
θ
θ'
$\rho = x1\cdot\cos\theta + y1\cdot\sin\theta$

(a) *(b)*

Figure 2-5 - (a) Image space for Hough transform; (b) θ-ρ space for Hough transform.

Generalization to more complex curves that can be described by an analytical equation is straightforward. In each case, the appropriate parameter space is constructed and quantized within the limits of the parameters. Then each identified edge pixel in the image would map to a number of accumulator cells for different sets of parameters. The accumulator cells with the higher number of counts would give the most probable parameters for the shape in question. If the desired region borders cannot be described using an analytical situation, a generalized Hough transform [12, 40, 83] can be used. In this case, a parametric curve description is constructed based on sample situations detected in the learning stage. Finally, even if the exact shape of objects is unknown, as long as there is enough a priori knowledge to form an approximate model, a fuzzy Hough transform can be used [148].

Graph techniques for edge detection and linking

Some methods have also been proposed to link edge elements based on representing them in the form of a graph and searching for an optimum path in it that corresponds to a meaningful boundary. A *graph* [59, 186] is a general structure consisting of a set of *nodes* n_i and *arcs* between the nodes [n_i, n_j]. A graph in which the arcs are directed is called a *directed graph*. If an arc is directed from node n_i to node n_j, then n_j is called a *successor* of its *parent* node n_i. The one node at level zero of the graph is called the *start* node, and the nodes in the last level are called the *goal* nodes. A numerical weight called *cost* can be associated with each arc. A sequence of nodes $n_1, n_2, ..., n_k$ with each node n_i being a successor of node n_{i-1} is called a *path* from n_1 to n_k and the cost associated with the path is the sum of the costs associated with its arcs.

Martelli [114, 115] expressed the problem of detecting edges as a heuristic search for the optimal path in a graph. The nodes of the graph were the edge elements defined by two neighbouring pixels. Martelli introduced some constraints for the sake of simplicity. He assumed that the edge starts at the first row, ends at the last row, contains no loops and has no element whose direction is "up". He constructed an evaluation function taking into account properties of edges, and the problem was reduced to finding a path in the graph corresponding to an edge that minimized the evaluation function.

Montanari [121] proposed a method for detecting curves in noisy pictures, where the properties of a curve are embedded in a figure of merit. This figure of merit was used to determine the relative value of different paths, but was not used to direct the search as in Martelli's method. Instead, all the paths were enumerated, and then the best one was chosen based on the figure of merit. The figure of merit was defined as a function of the grey-level value of the pixels corresponding to the path and the slopes between adjacent pixels of the path. Dynamic programming was used to arrive at an optimum solution.

The main advantage of using dynamic programming over boundary tracing as described in Martelli's method is that a priori knowledge of start and end points is not necessary. Furthermore, Martelli's method being sequential in nature does not provide for backtracking, so that once a mistake is made at some point, the resulting edge could prove far off from the actual one. On the other hand, Montanari's approach based on dynamic programming requires higher execution time and more memory.

Relaxation Techniques for edge detection and linking

Considering identified edge pixels in the context of their neighbours can increase or decrease the quality of segmentation. For example a weak edge pixel between two strong edge pixels, could give an indication that the weak edge pixel should be part of the final edge segment. Contrary, a strong edge pixel positioned by itself with no supporting context, is probably not a part of any border. Edge relaxation is the process of the iterative evaluation of edge properties of each pixel towards achieving a better confidence of each edge. Relaxation methods have been proposed by Rosenfeld [163, 164], Zucker [222], Riseman and Arbib [157] and more recently Hancock and Kittler [65], which increase or decrease the confidence of each edge based on the strength of edges in a specified local neighbourhood. Hancock and Kittler make use of a dictionary to represent labelling possibilities for the neighbourhood of each pixel. A method based on crack edges (edges located between pixels) has also been proposed [66, 153]. Relaxation methods are parallel processes, and utilize spatial information, although the convergence rate of the process can often prove slow.

2.1.3. Region Extraction Methods

The basic idea of region extraction methods is to divide an image into regions of maximum homogeneity. Homogeneity therefore is used as the main segmentation criterion here and can be defined, depending on the particular implementation, over a number of properties such as intensity, texture, colour or even semantic information possibly available. The selection of the homogeneity criterion is a very important side of region extraction techniques. Another equally important part is the selection of the threshold -based on the same property as homogeneity- under which two regions can be characterised as being similar.

Many region extraction methods can be found in the literature. They can roughly be categorized into region growing and merging techniques, region splitting techniques, region splitting and merging techniques and semantic region extraction techniques.

Region growing and merging techniques

The simplest form of region growing is pixel aggregation [60], which starts with a set of "seed" pixels in the image and from these grows regions by appending to each seed those neighbouring pixels that are similar to either the seed pixel or the pixels already in the region. Similarity is defined by the selection of the proper criterion for homogeneity and the proper threshold, as mentioned before. The results of the method are very sensitive to these selections as well as to the selection of the seed pixels.

One of the first approaches to region growing is the one of Muerle and Allen [122]. A region was defined to be any portion of the image in which the distribution of intensity (grey-values) is reasonably uniform, thus they defined homogeneity in terms of intensity. The first step in their approach was to segment the image into a number of cells of small size (*2*x*2*, *4*x*4*, or *8*x*8*) and calculate a statistical measure of intensity over the cells. Then, beginning with the first cell in the upper-left corner of the image, they compared the statistics with those of each neighbouring cell to determine if they are similar, in which case they joined the two cells into a bigger fragment. The process was continued; growing the fragment by comparing it to all of its neighbours, until no neighbours remained that could be joined to the fragment. The next uncompleted cell was then used as the starting one and the process was repeated until all cells were assigned to some region. The only property employed by this method was intensity information. The results of the method are dependant to the order in which the cells are being assessed, as is every region growing and merging method.

Brice and Fennema [20] used heuristics that evaluated parameters depending on more than one region. They started with individual pixels and following a process similar to that of Muerle and Allen [122] created regions of pixels having equal intensity. This first stage produces a large number of atomic regions in most of the cases. The concept of boundary was introduced. The boundary of each region is composed of a set of simply connected boundary segments, which are assigned strength according to the difference of intensities at each side of them. For each region, we can count the number of boundary segments having strength below a specific tolerance, as well as the total number of boundary segments, thus the perimeter of the region. The heuristics introduced by Brice and Fennema were based on these boundary segments. The first heuristic, called the "phagocyte" heuristic, merges two adjacent regions if the boundary between them is weak (strength below a tolerance) and the resulting region has a shorter boundary than the previous two. The second heuristic, called the "weakness" heuristic merges two regions if the weak portion of their common boundary is some predetermined percentage of their total shared boundary. The first heuristic is more general, and the second is used to refine the results of the "phagocyte" heuristic, but cannot be used on its own since it does not consider the influence of different region sizes.

Pavlidis [145] proposed a different approach to the problem of region growing using functional approximations. His technique was based on dividing the image into regions that could be sufficiently approximated by a single approximating function. He proved that if an image has been approximated by a large number of regions, then the number of approximating regions can be decreased by merging regions that have similar coefficients. Pavlidis initially sliced the image into one pixel wide stripes, and divided the stripes into segments such as the intensity of each of those segments could be approximated by a simple one-dimension linear function. These segments were the starting regions for the method, which considered joining regions with similar coefficients with the help of a graph representation of the starting segments. The only coefficient of the approximating function used in this implementation was its slope.

Region splitting techniques

Region splitting can be considered the opposite of region merging. While region merging is a bottom-up process combining smaller regions into larger ones, region splitting is a top-down process, starting with the whole image as one region, and dividing it in sub-regions so that the resulting regions conform to a homogeneity criterion. Although region merging and region splitting methods employ the same criteria about homogeneity and region similarity, the two methods are not dual, in the sense that even if identical homogeneity criteria were used, the two methods would not result in the same segmentation as can be seen in Figure 2-6.

One of the first techniques using region splitting was proposed by Robertson *et al.* [160]. They defined a criterion for region uniformity, called *G-regularity*. Although the original algorithm was developed for multi-spectral images, the grey-scale equivalent of *G-regularity* would be the mean grey level (intensity) of any sub-region to be the same as the mean grey level of the region. The algorithm subdivides regions imposing either a vertical or a horizontal partition, and continues doing so as long as a sub-region can be found whose mean grey level is sufficiently different from that of its parent region. Robertson *et al.* quantitatively defined the partition quality error of a region as the weighted sum of the grey level variance over every sub-region. The weights were given by the relative sizes of the sub-regions to the parent region.

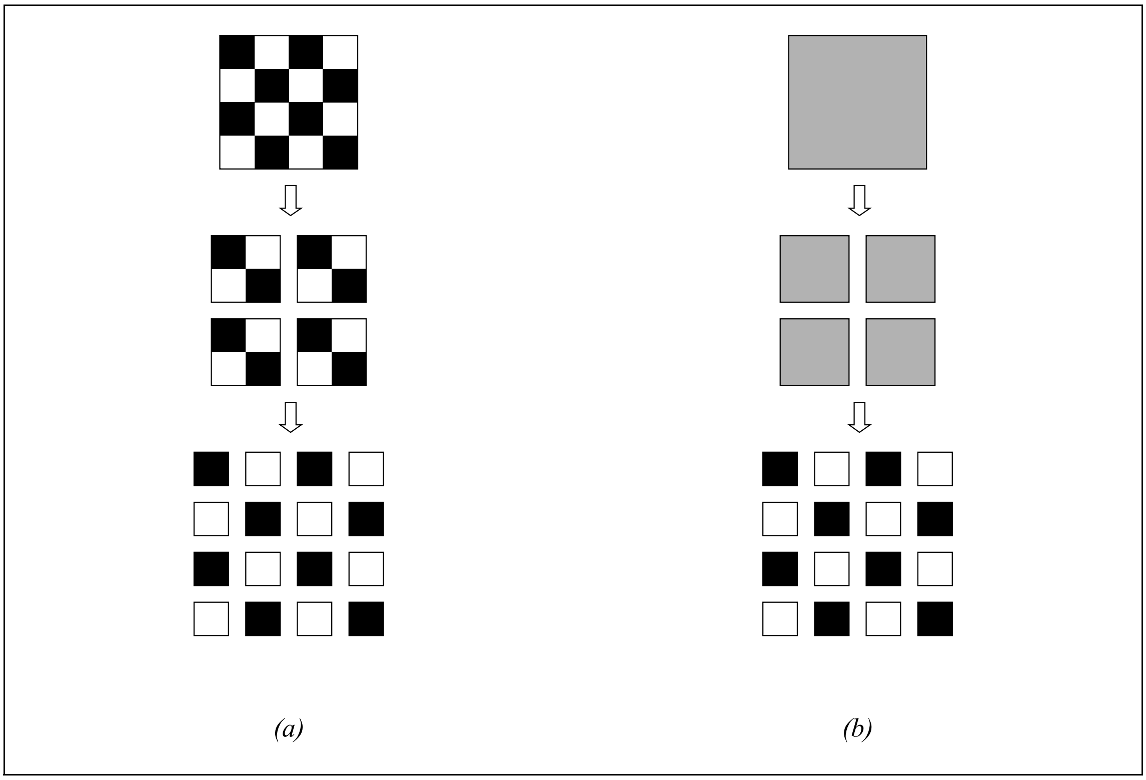


Figure 2-6 – (a) Different pyramid levels for a chessboard image; (b) The average grey-level for each pyramid level. Splitting the upper pyramid level does not result to regions of different average grey-levels, so no splitting occurs. The lowest level's regions have different average grey-levels so no merging can take place. Splitting and merging in this case would produce different segmentations, even if they use the same criteria.

Splitting and merging techniques

The next expected step in region extraction, was the joining of splitting and merging techniques. Horowitz and Pavlidis [76, 146] proposed a functional approximation split and merge technique, in which regions are described again in terms of an approximating function. The approximations are two dimensional here, in contrast with the previous work of Pavlidis [145]. A pyramidal structure was introduced (Figure 2-7), which is a stack of pictures beginning with the original picture at the bottom and pictures of decreased resolutions at higher levels. The picture at one level is produced from the picture at the level below by averaging the intensities of pixels in non-overlapping *2*x*2* squares. Thus, the picture at any level would be half the width and height of the picture at the level below. If any region in any pyramid level is not homogeneous, it is split into four regions, which are the corresponding regions at the level below. Similarly, if four regions exist at a pyramid having similar homogeneity values, they are merged into a single region in the above pyramid level. Splitting and merging can therefore be described as moving up and down in this pyramidal structure. This pyramidal structure can be expressed as a quadtree where the root is the top level of the pyramid structure and each node is an element of some pyramidal level. A constraint imposed by the pyramidal structure used, is the assumption of square regions. A grouping operation that follows the split and merge operation is used in Horowitz and Pavlidis method that addresses this issue by merging adjacent regions regardless of the pyramidal structure.

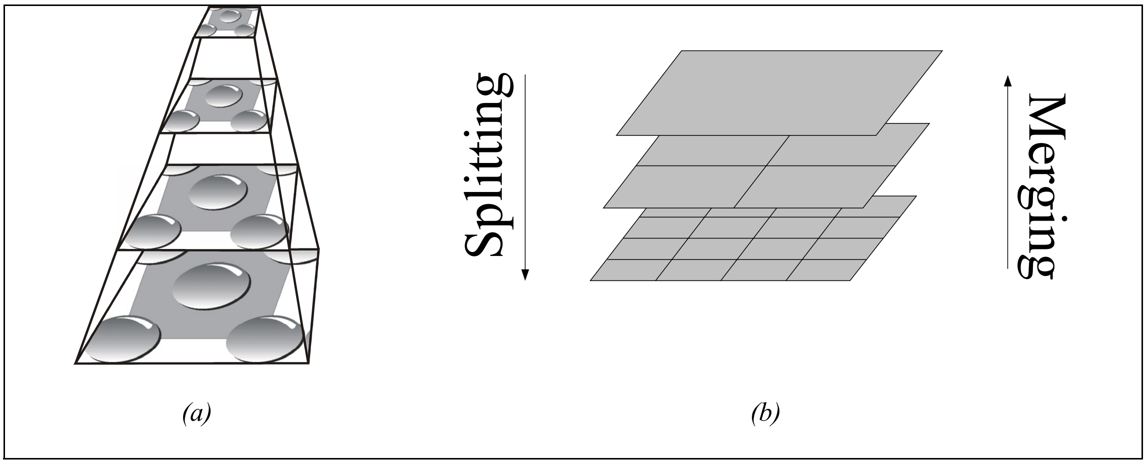


Figure 2-7 – (a) Pyramid structure of an image. Each level is a decreased resolution copy of the image below; (b) Splitting and merging expressed as moving between levels of the pyramid.

Based on the concept of splitting and merging, many approaches have been proposed. Chen *et al.* [32] proposed an adaptive split and merge algorithm. A modification of the pyramidal structure, introducing overlapping regions, was proposed by Pietikainen and Rosenfeld [149-151], where each region has four potential parent regions and each parent region has sixteen possible child regions. A single-pass split and merge method was proposed by Suk and Chunk [191], using a dictionary of the twelve possible splitting patterns for a *2*x*2* block of pixels. A single-pass method is advantageous it terms of memory usage which can prove high for split and merge methods [22].

Semantic region extracting techniques

In all the methods discussed up to this point, only heuristics based on local properties of regions were used, like intensity. Here, a number of techniques suggested that employ semantic information to facilitate merging will be described. Semantic segmentation methods interpret the regions as they are formed, using a-priori information about the contents of the image, and let these interpretations influence the merging process. Semantic information was first incorporated in the region extraction process by Gupta and Wintz [63, 64] and Feldman and Yakimovsky [50, 212].

Gupta and Wintz [63, 64] developed a system to segment satellite images. The classes of objects expected to be found in such images are related to earth structures such as water, sand, forest etc. This set of predetermined classes was the semantic information incorporated in the technique. The first step of the process is similar to that of Muerle and Allen [122]. The image is partitioned in cells and neighbouring cells having similar intensity distributions are merged. Following that step, each resulting region is interpreted as belonging to one of the predetermined classes. As a final step, neighbouring regions are checked again and merging of regions having been assigned to the same class occurs.

Feldman and Yakimovsky [50, 212] took the idea one step further. They did incorporate semantic information about a number of possible classes in which each identified region should fall, but they also checked relations between regions, based on this same semantic information. The goal of the approach was to maximize the probability that the image was correctly segmented, thus maximize a predefined objective function. The objective function used is conditioned upon both the probability that the regions were correctly interpreted as one of the predetermined classes, as well as the probability that the boundaries were correctly placed between regions. For example, if two predetermined classes were defined, one for the "sky" and the other for "ground", then we expect "sky" to be above "ground", and the interpretation of the borders should reflect this knowledge. Feldman and Yakimovsky's approach starts with a region merging process, controlled by general heuristics like the ones mentioned earlier. When this initial merging terminates, semantic information is incorporated and for every two adjacent regions, the probability that their border separates them into two regions of the same interpretation is computed. If this probability is exceeding a specified tolerance, the two regions are merged. After this step terminates, the probability that a region belongs to each of the predetermined classes is computed for each region. The region with the highest confidence is assumed correct and marked as final. The interpretation probabilities of all its neighbouring regions are updated, in order to maximize the objective function mentioned before. The next higher confidence region is marked as final, and the process continues until all the regions have been assessed.

Semantic information in the aforementioned techniques was incorporated at a later stage, after an initial segmentation had already occurred. Although this initial segmentation reduces the computational overhead of the method, it is difficult to stop the process at a point when there are neither to many or too few regions in the image. Thus the conventional split and merge techniques, result in under-segmented or over-segmented images. Furthermore, the results are sensitive to the split / merge order, so even if a merger produces a homogeneous region, there is no way to say that a different merger would not produce an even better one. Starting from partially processed data, semantic segmentation tries to maximize an objective function. Both the fact that some information could already have been lost, and the fact that the semantic segmentation part is also sensitive to the merge order, could result in finding only a local maximum of the objective function, thus the process ends in a local optimum of region labelling.

Sonka *et al.* [186] describe a semantic genetic segmentation method that addresses these problems. The genetic segmentation method uses an objective evaluation function, similar to the ones used in previously mentioned methods, which gives a measure of the goodness of the segmentation. An over-segmented image called a primary segmentation is used as the starting point. The genetic algorithm is then responsible to generate a population of segmentation and interpretation hypotheses and test them according to the evaluation function. The testing of the whole population occurs in a single step, in which a decision is made about which hypotheses should survive and which should die, and a new population is generated from the survived members. In that manner, good mergers (as assessed by the objective function) are allowed to survive and bad ones are discarded. In a method like this, no merge or split is final. A better segmentation is looked for, even if the current ones are already good. This means that ultimately the global maximum of the objective evaluation function will be reached, and not a local one.

Incorporating semantic information into the segmentation process is advantageous, but it is also difficult and usually results in complex techniques expensive in memory usage and time.

2.2. Colour Segmentation techniques

Grey-scale image segmentation is not too complicated in the sense that the feature space is one-dimensional. The difficulty increases with the introduction of colour in images. The feature space of colour images is inherently multi-dimensional, since three (in some cases more) components are needed in order to describe each colour. Nevertheless, generally colour segmentation approaches do not treat a pixel's colour as a point in a colour space; instead, they decompose it into three separate values, which they recombine later on. This is the natural result of trying to extend to colour images, methods originally proposed for grey-scale ones. This introduces a number of problems, which will be explained later on.

In the first section of this chapter, a definition of colour is given and a brief description of colour systems and their importance for colour image segmentation is examined. The remainder of this chapter comprises a review of techniques used for colour image segmentation along with a critical appreciation of methods proposed. Finally, concluding remarks are given in the last section.

2.2.1. Colour

Colour is the perceptual result of light in the visible region of the spectrum (having wavelengths between approximately *400nm* to *700nm*). More formally, the word colour can be used to define two similar but distinct effects. Colour is used to define the aspect of human perception concerned with the ability to make a distinction between different spectral power distributions, in which case it is called *"perceived colour"*. The word colour is also used to define the characteristic of a visible radiant energy itself, which causes such a sensation, in which case it is called *"psychophysical colour"*.

Wyszecki and Stiles [210] define psychophysical colour as following: *"Colour is that characteristic of visible radiant energy by which an observer may distinguish differences between two structure-free fields of view of the same size and shape, such as may be caused by differences in the spectral composition of the radiant energy concerned in the observation"*. For simplicity reasons, the word colour is used throughout this thesis referring to psychophysical colour.

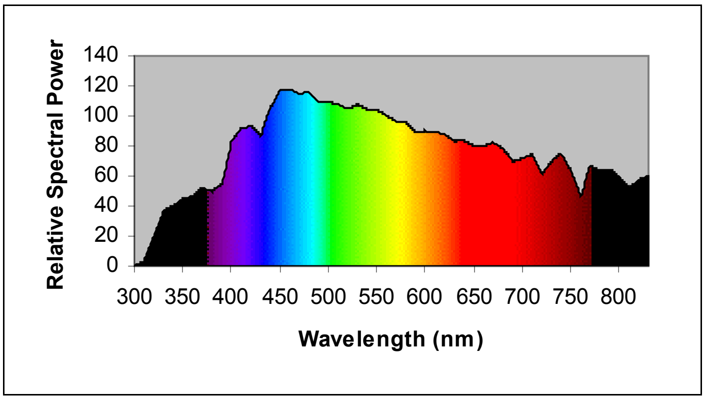


Figure 2-8 – CIE standard daylight illuminant D_{65} relative spectral power distribution. The power of each wavelength is normalized so that the power at λ=560nm equals 100.

A colour stimulus is radiant energy of given intensity and spectral composition, entering the eye and producing a sensation of colour. This radiant energy can be completely described by it's spectral power distribution. This is often expressed in *31* components, each representing power in a *10nm* band from *400nm* to *700nm*. For example, Figure 2-8 shows the spectral power distribution of *CIE* (Commission Internationale de l'Eclairage or International Commission on Illumination) standard daylight illuminant D_{65}. Using *31* components is a rather impractical and inefficient way to describe a colour, especially when a number of colours must be described and communicated, which is the case with computer graphics. A more efficient way would be to determine a number of appropriate spectral weighting functions to describe a colour, and it proves that just three components are adequate for that, based on the trichromatic nature of vision. CIE standardized in 1931 a set of spectral weighting functions, called Colour Matching Functions, which model the perception of colour. These curves are referred to as $\bar{x}$, $\bar{y}$, and $\bar{z}$, and are illustrated in Figure 2-9. A more detailed discussion on human vision and colour systems is available in Appendix A, while colour systems in the context of colour image segmentation are briefly discussed later in this section.

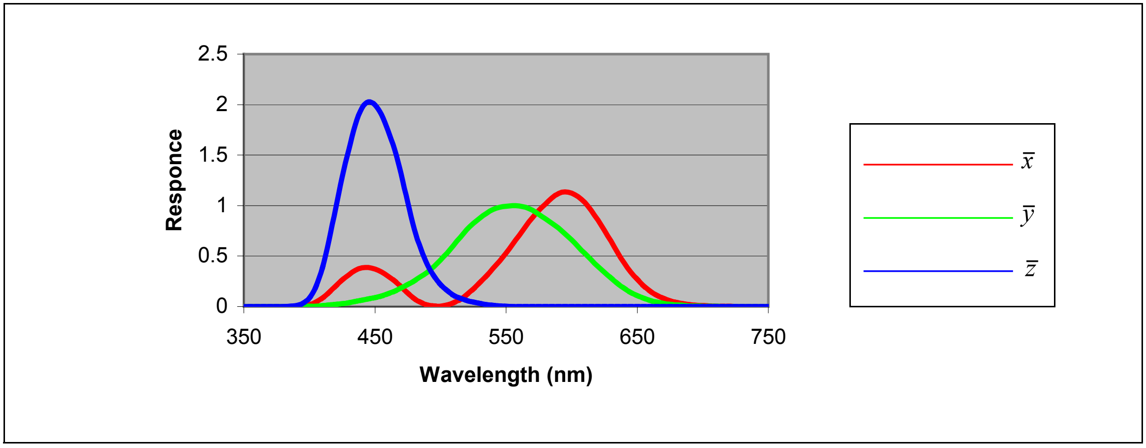


Figure 2-9 – CIE Colour Matching Functions.

Colour in Computers

Colour is reproduced in Cathode Ray Tube (CRT) displays in an additive manner by mixing three lights of different colours (red, green and blue) produced by the phosphors of the screen. Thus three components are being used, namely *R*, *G* and *B* which express the participating power of each mixing colour. Each component is quantised in $2^8=256$ levels, thus a CRT display can produce 256^3 colours, by mixing different amounts of light of each component. Depending on the technical and physical characteristics of the CRT display, only a certain gamut of colours can be produced. The largest range of colours will be produced with primaries that appear red, green and blue, and that is the reason why phosphors producing colour stimulus with exactly those primaries are employed. Nevertheless, since there are no standard primaries and no standard white point (see Appendix A) *RGB* information alone is not adequate to determine the actual colours of an image. A set of primaries that closely represent the primaries used in CRT monitors are the ones specified for the HDTV protocol by the standard *ITU-R recommendation BT.709* [84]. The majority of monitors conform to *Rec.709* within some tolerance, so it is a safe assumption that the same *RGB* code will produce the same colour on different CRT monitors.

Size of Colour Images

Since three components – namely *R*, *G* and *B* – are required to code each colour, in contrast to only one grey-scale component needed to code intensity, a colour image would generally contain three times the amount of information contained in a greyscale one. Each component is usually quantised to 8 bits in order to eliminate distinguishable quantisation steps. Older hardware used to have problems handling full-colour images, due to the large memory requirements and the size of data that had to be manipulated. The large size of colour images does not pose that many problems to current hardware, nevertheless when size and transmission speed is an issue (e.g. World Wide Web), methods to reduce the overall size of colour information are welcome.

Most of the existing techniques to reduce the size of colour images achieve so by discarding some amount of "excessive" colour information. Some common techniques used are colour quantisation and bit dropping. Many image display devices and graphics file formats allow only a limited number of colours to be simultaneously displayed. This set of available colours is called *palette*. Colour quantisation is the process of selecting the optimal colour palette and the optimal mapping of each pixel of the image to a colour from the palette. A number of methods have been proposed for colour quantisation [19, 73, 138, 142]. When images are quantised to very few levels, error diffusion techniques are usually employed to achieve the optimal viewing result [23, 138]. Bit-dropping is a special case of colour quantisation, where the palette of available colour is constructed by removing the lower order bits of each component. Usually three bits are used for the red and green component and two bits for the blue one, resulting to an 8-bit description of each colour. All types of colour reduction techniques discard useful colour information and introduce certain artefacts to the images that complicate image processing and image analysis techniques.

Colour Systems

Many colour systems have been proposed throughout history for the systematic ordering of colours. The choice of the colour system to use with a colour image segmentation method has a significant impact on the results. An understanding of the basic colour systems that are in wide use nowadays is vital for understanding colour image analysis. In this section, some basic information about colour systems will be summarized in the context of colour image segmentation. A more analytical discussion about colour systems and human perception along with mathematical transformations between key colour systems is given in Appendix A.

The most widely used colour system in computer applications is *RGB*. As mentioned before, the *RGB* colour system directly describes the way colours are mixed on computer monitors. Although *RGB* is hardware dependant, in the sense that the same *RGB* colour may be slightly different between different monitors, it is the default choice for most applications because of its simplicity and low computational cost.

Various kinds of colour attributes can be calculated from the *RGB* components. An interesting set of attributes, in the sense that they are representative of human perception, is Hue, Lightness and Saturation. These are psychological attributes related to human impressions of colour. The use of such perceptually based quantities can prove more suitable for the analysis of images created to be viewed by humans [98, 195]. *HLS*, *HVC* and *HSI* are colour systems based on these attributes.

A significant problem with most colour systems is that the distance of two colours in the colour system does not correlate in any way with the perceived distance of the colours (how similar or dissimilar they are). For this reason, CIE proposed certain transformations of the *XYZ* colour system, resulting in systems that exhibit greater perceptual uniformity than *XYZ*. CIE $L^*a^*b^*$ [116, 159] and CIE $L^*u^*v^*$ [30] are such colour systems, and are used when a colour distance measure that correlates well to the perceptual colour distance is needed. The only disadvantage of using either of these systems is that the systems being hardware independent it is difficult to convert to and from them, and extra information about the hardware used is needed.

Colour Systems for Colour Image Segmentation

Many researchers tried to identify which colour system, or which colour features are the best to use for computer imaging. Ohlander, Price and Reddy [135] employed nine redundant colour features (*Red*, *Green*, *Blue*, *Y*, *I*, *Q*, *Hue*, *Saturation* and *Intensity*) for colour image segmentation and reported that *Hue* was most useful, whereas the set of *YIQ* was rarely used. Nevatia [131] also concludes by extending the Hueckel operator for colour images that the intensity and chromaticity coordinates give better results than the R, G, B values in detecting colour edges.

Ohta, Kanade and Sakai [136] conducted a systematic experiment of segmentation, and derived three effective colour features in terms of a linear combination of *R*, *G* and *B* components. They performed a recursive thresholding segmentation algorithm similar to Ohlander's and calculated new colour features at each step by the Karhunen - Loeve transformation of *R*, *G* and *B*. They found that $I1=(R + G + B)/3$, $I2=R - B$ and $I3=(2G - R - B)/2$ are effective, and that in many cases the first two features only were adequate for a good segmentation. These three features strongly correlate to the three components of the CIE $L^*a^*b^*$ colour system.

Schacter, Davis and Rosenfeld [176] also suggested that the use of a uniform colour system such as the CIE $L^*a^*b^*$ could improve the performance of colour clustering. Furthermore, Zhang and Wandell [217] proposed a spatial extension of the CIE $L^*a^*b^*$ colour system in order to simulate the spatial blurring that naturally happens by the human visual system. The image is transformed into an opponent colours space and each opponent-colours image is convolved with a kernel whose shape is determined by the visual sensitivity to that colour area.

Choosing the appropriate colour system to use is not a trivial task. Each application has different objectives and requirements, and one colour system cannot always be the right choice. The latest developments are towards systems that are hardware independent and computationally inexpensive. In order to preserve colour appearance between different monitors, special hardware profiles are increasingly used, and new colour systems are developed. *sRGB* [190] is such a colour system, developed for use in the World Wide Web.

2.2.2. Histogram Analysis Techniques

Although colour information is fully described by three or more components as discussed before, the complexity of working with all colour components simultaneously, lead to simpler approaches that work with one or two colour components at a time. Some of the earliest approaches for segmenting colour images were based on 1D histogram analysis of colour components of the image. In most of these methods, histograms for all colour components available are computed, but segmentation happens by working on one of the histograms, and recursively splitting the feature space into clusters, one colour component at a time. A number of histogram analysis methods will be discussed in this section, working our way to more complex clustering schemes in section 2.2.3.

Histogram Analysis Techniques

One of the first colour segmentation methods proposed is by Ohlander, Price and Reddy [135]. The method is based on selecting the best peak from a number of histograms of different colour features, and recursively splitting the image in two, based on the peak selected. The nine colour features used were collected from three colour systems: *RGB*, *HSI* and *YIQ*. The method starts by considering the whole image as one region, and computing all nine histograms for it. The histograms are smoothed in order to eliminate small peaks, and the best peak is then selected among the remaining ones. The goodness of a peak is determined by a set of seven conditions based on previous work by Ohlander [134]. The image is then thresholded according to the peak selected. Points within the thresholds defined by the peak (the left and the right minimum) are set to "1" and the rest are set to "0", thus creating a binary image. This image is subsequently smoothed in order to eliminate small holes in regions, small regions or thin connections between regions. Connected regions are then identified in the thresholded and smoothed binary image, and a similar region segmentation is performed for the pixels of each region, and for the remainder pixels of the image not assigned to any of the regions extracted. The process stops when too few points are left, or no good peaks can be identified.

Ohlander's method has been the base of a number of approaches, which follow the same scheme of recursive thresholding of 1D histograms. Methods using different colour features, and a variety of conditions to select the best peak of the histograms have been proposed. For the experiment of Ohta, Kanade and Sakai [136] mentioned in the previous section segmentation is performed in a similar manner, and many of the methods that follow are also influenced by Ohlander's approach. In addition, Ohlander, Price and Reddy, suggested several improvements for the basic scheme, such as removing small regions, work on a reduced image version, using of textural operators to produce more features (especially for monochromatic images) etc.

Tominaga [195] proposed a recursive method based on histogram analysis of the three perceptual attributes of the Munsell colour system. The three attributes of the system are *Hue*, *Value* (which corresponds to *Lightness*) and *Chroma* (which corresponds to *Saturation* or *Colour Purity*). A mapping between four components (*B/W*, *R*, *G*, *B*) measured with a drum scanner and the three components of the *HVC* colour system, is determined by measurements of a number of standard-colour chips. The histogram of each attribute is then calculated for the whole image, and the most significant peak is selected from the set of the three histograms. Peak selection is performed based on certain peak characteristics and a criterion defined as:

$$f = \frac{S_p}{T_a} \cdot \frac{100}{fwhm} \qquad \textit{Eq. 2-16}$$

where S_p is the peak area, T_a is the area of the whole histogram, and *fwhm* is the full width at half-maximum of the peak. The pixels that fall in the range demarcated by the peak define a sub-region. The sub-region is extracted and the thresholding process is repeated for the pixels of the sub-region, leading to the detection of the most significant cluster. The process finishes when the histograms become mono-modal. Following a labelling step based on the above segmentation, the same procedure of recursive thresholding is applied to the remaining of the pixels, identifying in this way the rest of the important clusters.

Tominaga also proposed a modification of the aforementioned algorithm [196]. The first step here is essentially the same as described above, modified to overcome the problem of handling overlapping clusters. The colour space used this time is the CIE $L^*a^*b^*$. A second step is supplemented to the algorithm, which classifies again the pixels, based on a colour distance metric in CIE $L^*a^*b^*$. A number of colours are initially identified according to the regions the image was initially partitioned into. So if K is the number of regions resulted by the first step, K colours (the colours of the regions) are identified as representative of the image. Let $k_1, k_2, k_3 \ldots, k_n$ be the set of representative colours. The first colour is used as the centre for a first cluster m_1, so that $m_1=k_1$. The second colour is then compared to m_1 in terms of its distance in CIE $L^*a^*b^*$. If the distance is more than a threshold T, then a new cluster is created with centre $m_2=k_2$; otherwise, k_2 is assigned to the domain of cluster m_1. In a similar fashion, the colour difference of each representative colour to each established cluster centre is computed and thresholded. A new cluster is created if all of these distances exceed the threshold T, otherwise the colour is assigned to the class to which it is closest.

An approach that operates in the CIE $L^*a^*b^*$ colour space has been proposed by Celenk [31]. Celenk defines a set of cylindrical coordinates in CIE $L^*a^*b^*$ which resemble the Munsel colour system, and concur well with the accepted physiological model of colour vision. The coordinates used are named L^*, $H°$ and C^* and stand for *Lightness*, *Hue* and *Chroma*. The method proposed is similar to Ohlander [135] and Tominaga [195] in the sense that clusters are identified by recursive thresholding of the 1D histograms of the colour features. The colour space is projected onto the selected coordinate axes repeatedly until the image clusters are enclosed in some specified volume elements. The innovative point in Celenk's approach is that a cluster isolation step is also employed to separate neighbouring clusters, based on Fisher linear discriminant function.

When projecting the feature space onto a lower-dimensional space, it is expected that some clusters that were separable in the original feature space, are no longer separable in the lower-dimensional space. For example, a peak in a 1D histogram for a colour image may reflect a number of clusters (Figure 2-10a). Furthermore, any clustering that emanates from thresholding 1D histograms can only produce rectangular clusters in the original (Cartesian) feature space, limiting vastly the flexibility of the method (Figure 2-10b). Nevertheless, the lower complexity and computational cost of these methods makes them a preferable choice in many cases.

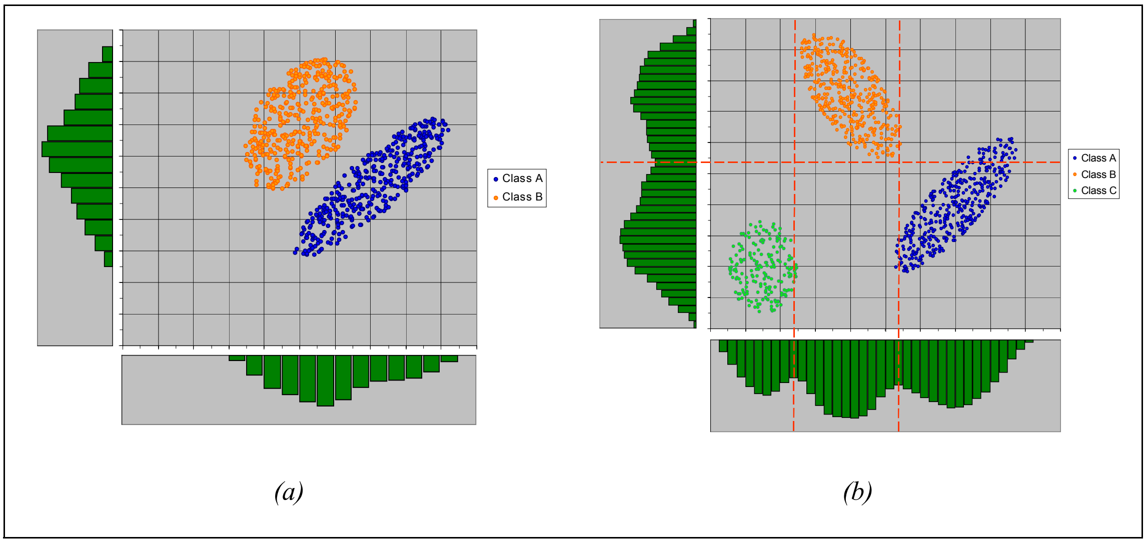


Figure 2-10 – (a) Peaks in the 1D Histograms, reflect more than one cluster. (b) Even if a good separation of the feature space can be achieved by analysing 1D Histograms, only rectangular clusters can be obtained.

Methods based on lower-dimensional space analysis, especially recursive thresholding techniques [135], are commonly used as a first step in more complicated methods [177] as will be seen in later sections.

2.2.3. Clustering Techniques

The multidimensional extension of the concept of thresholding is called clustering. Clustering is the process of assigning units that share common characteristics in a number of homogeneous partitions in the feature space called clusters. Clustering algorithms in the literature can be broadly divided in hierarchical clustering and partitional (or non-hierarchical) clustering algorithms.

Hierarchical clustering algorithms produce a tree structure of a sequence of clusterings. According to their tree-structure, hierarchical clustering algorithms can be categorized into nested and non-nested. In nested hierarchical clusterings, each cluster fits itself in whole inside a larger cluster of a higher scale, whereas in non-nested hierarchical algorithms a cluster obtained at a smaller scale can divide itself into several parts and fit those parts in different clusters at a higher scale. Nested tree-structures are usually easier to use, nevertheless once a cluster is formed, its members cannot be separated subsequently, making nested hierarchical clustering algorithms less flexible than non-nested ones.

Partitional clustering algorithms produce a single partitioning of the data set in contrast to hierarchical ones. Most partitional clustering algorithms achieve the clustering through the minimization of appropriate measures such as cost functions. The high complexity and computational cost of those algorithms necessitates the use of techniques such as simulated or deterministic annealing to lower the computational overhead and ensure to a certain degree that the global minimum of the criterion used has been reached. K-means clustering, ISODATA and c-means clustering are just a few examples of partitional clustering algorithms.

Clustering is a generic process used across a variety of fields. In the context of colour image segmentation, the feature space commonly used for clustering is the colour space employed for the description of the image. Since colour spaces are inherently three-dimensional, clustering is usually a computationally expensive process, therefore a number of methods have been suggested that work in lower-dimensional spaces. A brief description of two two-dimensional clustering approaches will be given first, followed by a critical review of several multi-dimensional hierarchical and partitional clustering techniques.

Two-dimensional Clustering Techniques for Colour Image Segmentation

Concerning multidimensional measurement space clustering, Haralick and Shapiro [69] propose to work in multiple lower order projection spaces and then reflect the clusters identified back to the full measurement space. "Suppose, for example, that the clustering is done on a four band image. If the clustering done in bands 1 and 2 yields clusters c_1, c_2, c_3 and the clustering done in bands 3 and 4 yields clusters c_4 and c_5, then each possible 4-tuple from a pixel can be given a cluster label from the set $\{(c_1, c_4), (c_1, c_5), (c_2, c_4), (c_2, c_5), (c_3, c_4), (c_3, c_5)\}$. A 4-tuple (x_1, x_2, x_3, x_4) gets the cluster label (c_2, c_4) if (x_1, x_2) is in cluster c_2 and (x_3, x_4) is in cluster c_4." However, as Pal and Pal [140] comment on this suggestion, this virtually assigns a point (a 4-tuple) in two different classes, without it being any kind of probabilistic or fuzzy assignment.

In an early work Ali, Martin and Aggarwal [3] employed 2D scatter plots of the three colour components in order to help define the clusters in the 3D space. The colour components used were X, Y, and I, where I approximates the intensity and X and Y are two chromaticity coefficients. The segmentation operation proposed was an interactive procedure which allowed the user to define the 3D clusters by selecting rectangular areas on the projections of the XYI normalized colour space onto the X-Y, X-I and Y-I planes. Areas of more complicated shapes were included in later refinements of the application [173], these included ellipsoid areas and areas bounded by two second-order curves in X or Y and a range in I.

Hierarchical Clustering

Leung, Zhang and Xu [99] give a comprehensive study of hierarchical clustering algorithms. Their rationale is that since the human visual system has become optimal in clustering images through years of evolution, a clustering approach should be sought which is influenced by the way humans cluster data. Therefore, they use scale-space theory to model the blurring effect of lateral retinal interconnection (by which we perceive an abstract image first, and then focus on the details). A data set, according to the authors, can be considered as an image, were each datum is essentially a light point. By blurring this image, each datum becomes a light blob and smaller blobs merge into larger ones until the image contains only one big light blob. If each blob is considered to be a cluster, then the whole process can be described by a hierarchical clustering with resolution as the height of the tree. The authors describe both nested and non-nested hierarchical clustering algorithms based on scale-space theory, but the centre of attention is at ways to assess the goodness of a cluster and to select the best clustering. Towards this end, they propose a set of validity criterions for clusters. Exploiting the hierarchical clustering structure, they reason that a validity criterion can be the range of scales in which a cluster is visible, in other words, the lifetime of a cluster. They also define a mean lifetime for a particular clustering, based on the lifetimes of its clusters, and propose the use of this metric to facilitate the selection of the best clustering. One of the main advantages of this approach is that the hierarchical algorithms suggested have no need for initialisation, while scale-space theory provides a base for the construction of new rules for the validity of clusters and the selection of the best clustering.

K-means Clustering

One of the most popular algorithms for partitional cluster analysis is the K-means algorithm, first proposed by MacQueen [111]. The number of clusters for the K-means algorithm must be known beforehand and is assumed to be *K*. In reality, when the number of classes in the image is not known, K-means clustering can be repeated for different values of *K* and the best clustering finally decided upon.

Let m_i be the mean of the vectors of cluster *i*, where $1<i<K$. Let also $x_1, x_2, x_3 \dots x_n$ be the feature vectors (points) of the data set. The algorithm starts by making initial guesses for the mean vectors of the *K* clusters. There are many methods proposed to select the initial set of cluster means [2, 18], although in several cases a random choice is made. Having defined the means of the clusters, the next step is to assign every feature vector of the data set to one of the clusters according to their distance (Euclidean) from the mean of the cluster. The mean of each cluster is then recalculated as the mean of all the feature vectors allocated to the cluster. The final step of the algorithm is to reassign all the feature vectors of the data set to the new clusters.

There are many variations to this algorithm, the most common one being to repeat the second part of recalculating the means of the clusters and reassigning the feature vectors of the data set to the new clusters until convergence, that is until no changes in the mean of any cluster occur. Another way to modify the K-means clustering procedure is to update the means one feature vector at a time, rather than all at once. This is often referred to as sequential K-means clustering and is particularly attractive in some applications where data are acquired over a period of time.

Essentially, K-means algorithm aims to minimize an appropriate criterion. The sum of squares criterion, given in Eq. 2-17, is most commonly used. More often than not, a local minimum is reached instead of the global one, necessitating the use of techniques such as simulated or deterministic annealing in order to achieve a better solution.

$$D = \sum_{i=1}^{K} \sum_{n \in S_i} \left\| x_n - m_i \right\|^2 \qquad \textit{Eq. 2-17}$$

K-means clustering has been used a lot for colour image segmentation. An example is the method proposed by Weeks and Hague [203], who perform K-means clustering in the *HSI* colour space. They also propose breaking the clustering process in two, one in the Intensity – Saturation two-dimensional space and one in the Hue one-dimensional space, biasing that way the segmentation process towards a colour's Hue value, which they the consider more important for human perception.

The K-means clustering technique has a number of weaknesses. The most prominent disadvantage is the fact that the results of the clustering depend on the number of clusters, and the way the initial means for them are initialised. It frequently happens, that non-optimal partitions are found. The standard solution to this problem, is to try a number of different starting points, or the use of techniques such as simulated or deterministic annealing as mentioned before. Depending on the initial selection of cluster means, it is possible that the set of feature vectors closest to one of the means is empty therefore that specific mean cannot be updated. Special cases like these have to be properly handled by each implementation. K-means clustering is often used as a first step in more complicated approaches. An example of such an approach will be given next.

K-means Clustering and Probabilistic Assignment

Mirmehdi and Petrou [118] suggested a method to segment colour image textures based on human perceptual characteristics. When an observer deals with multi-coloured objects, their colour matching behaviour is affected by the spatial properties of the stimuli. For example, when a number of pixels of different colours are contained in a small area, humans cannot discriminate between the colours; instead they will perceive an average of the colours present. In order to simulate this characteristic of human perception, a multi-scale representation is constructed for the image. Images of different scales are produced by smoothing the original image in three bands according to the suggestions of Zhang and Wandell [217], as discussed in section 2.2.1. A K-means clustering algorithm with a large K is then used to initialise the segmentation at the coarser level. After this first segmentation, clusters that share a common border are checked in terms of the average colour of their border pixels in the CIE $L^*u^*v^*$ colour system, and if their colour distance (Euclidean distance in CIE $L^*u^*v^*$) is smaller than a tolerance, they are merged into a bigger cluster. Core clusters are then identified, by means of checking a confidence measure for each pixel with which it may be associated with each cluster. Pixels that present a confidence higher than *50%* are assigned to a cluster. The image pixels of the core clusters are used to construct 3D colour histograms, which are then used for probabilistic assignment of all other pixels to the core clusters. The 3D histograms are recalculated for every resolution following a coarse to fine scheme, and the probabilistic assignments are updated in each step.

Fuzzy c-means Clustering

The fuzzy c-means algorithm [15] uses iterative optimisation of an objective function based on a weighted similarity measure between the pixels of the image and the means of each of the clusters. The number of clusters is once again predefined, although methods have been proposed that take advantage of the fuzzy character of the algorithm to determine the appropriate number of clusters, e.g. one can dynamically select the appropriate number of clusters depending on the strength of memberships across clusters [24]. The objective function used is defined as

$$W_m(U,V) = \sum_{k=1}^{n} \sum_{i=1}^{c} (\mu_{ik})^m (d_{ik})^2 \qquad \textit{Eq. 2-18}$$

where μ_{ik} is the fuzzy membership value of the pixel k in the i cluster, d_{ik} is any inner product induced norm metric, m controls the nature of clustering (with hard clustering for $m=1$ and increasingly fuzzier clustering for higher values of m), and U is the fuzzy c-partition of the image over the set V of the means of the c clusters. Local extrema of this objective function are indicative of an optimal clustering of the input data.

The algorithm starts by fixing the number of clusters *c*, and the value of *m*, and choosing any inner product induced norm metric. Then the fuzzy c-partition, U^0 is initialised (e.g. uniformly). For each step *b* (*b=0, 1, 2, ...*), the set of means V^b of the *c* clusters for the fuzzy c-partition U^b are calculated according to:

$$v_i = \frac{\sum_{k=1}^{n} (\mu_{ik})^m x_k}{\sum_{k=1}^{n} (\mu_{ik})^m} \qquad \text{Eq. 2-19}$$

Next, the c-partition U^b is updated and the memberships in U^{b+1} are calculated as follows. Let $d_{ij}=||x_k-v_i||$, where v_i is the mean of cluster *i*. Then

$$\mathit{If}\ d_{ik} \neq 0\ \mathit{then}\ \mu_{ik} = \frac{1}{\left[\sum_{j=1}^{c}\left(\frac{d_{ik}}{d_{jk}}\right)^{2/(m-1)}\right]},\ \mathit{else}\ u_{ik} = 0 \qquad \text{Eq. 2-20}$$

Finally, the two c-partitions U^b and U^{b+1} are compared, and if their difference is less than a preset threshold the process stops, otherwise it returns to the previous step of computing the new set of means V^{b+1} for the new c-partition.

A fuzzy c-means clustering algorithm has been proposed by Huntsberger, Jacobs and Cannon [81], for the segmentation of colour images. The number of clusters is predefined to four, but the pixels that are not assigned to a cluster due to a low fuzzy membership value, are fed back to the procedure, defining four more clusters. Initially the algorithm clusters a randomly chosen sample of *2400* pixels taken from the image into *c=4* clusters. The cluster means calculated from this sample are then used to calculate membership functions for all the pixels in the image. Each pixel that presents a membership value above a pre-defined threshold is assigned to the corresponding cluster. The process is repeated for all the pixels not assigned to a cluster during the first iteration, introducing four new clusters at each iteration. The number of *2400* pixels and the number of clusters were decided upon after experimentation with different values. At each iteration, the algorithm checks the cluster means to determine whether two cluster means are very close. The condition used by the authors is given in Eq. 2-21. If the difference computed for each feature is less than the set threshold, the two clusters are considered identical and are merged into one bigger cluster. The feature spaces used were the *RGB* colour space and the $I_1I_2I_3$ colour space proposed by Ohta [136], while the inner product norm metrics tested were the Euclidean and the Mahalanobis distances. The authors concluded that the differences between the *RGB* and the $I_1I_2I_3$ colour space as far as segmentation is concerned are minimal.

$$\textit{for each feature "f"},\quad \frac{2\cdot\left|f_1-f_2\right|}{f_1+f_2}<0.075 \qquad \textit{Eq. 2-21}$$

An interesting colour segmentation method based on fuzzy c-means has been proposed by Lim and Lee [103]. They use scale-space histogram thresholding to find the number of clusters to use for the fuzzy c-means clustering. First, scale-space filtering is applied to the histogram of each colour component and the optimal scale is determined. Examining the smoothed (at the determined optimal scale) histograms, they locate valleys by use of the first and the second derivative. Using this knowledge of the location of valleys in each colour component histogram, they effectively divide the colour space in a series of hexahedra. The first phase thus identifies clusters in terms of the hexahedra located, but only the ones containing above a certain number of pixels are considered good. A second step then follows that classifies the rest of the pixels into one of the found clusters. Lim and Lee tried a number of different colour spaces (*RGB*, *XYZ*, *YIQ*, *UVW*, $I_1I_2I_3$) and observed that $I_1I_2I_3$ proposed by Ohta [136] gives the best results.

Graph Partitioning

Shi and Malik [180], formulated the segmentation problem as a graph partitioning problem. Their method reduces the problem of graph partitioning to solving an eigenvector and eigenvalue problem. Let $G=(V, E)$ be a graph whose nodes are points in the measurement space and whose edges are associated with a weight representing the similarity between two nodes. This graph *G* can be partitioned into two disjoined sets *A* and *B* by removing all the edges connecting the two partitions. The total weight of the edges removed is called *cut* and is given in Eq. 2-22.

$$cut(A,B) = \sum_{u \in A, v \in B} w(u,v) \qquad \textit{Eq. 2-22}$$

$$asso(A,V) = \sum_{u \in A, t \in V} w(u,t) \qquad \textit{Eq. 2-23}$$

$$Ncut(A,B) = \frac{cut(A,B)}{asso(A,V)} + \frac{cut(A,B)}{asso(B,V)} \qquad \textit{Eq. 2-24}$$

$$Nasso(A,B) = \frac{asso(A,A)}{asso(A,V)} + \frac{asso(B,B)}{asso(B,V)} \qquad \textit{Eq. 2-25}$$

One way to recursively partition the graph would be to find the minimum cut at each iteration and split the graph in two, until the regions produced are uniform enough. Nevertheless, the minimum cut favours cutting small sets of isolated nodes, for this reason Shi and Malik proposed another criterion, the *normalized cut*, given in Eq. 2-24, based on the association (defined in Eq. 2-23) of each partition to the full node set and the definition of cut given before. Furthermore, a criterion of how tightly the nodes within a given set are connected to one another, named *normalized association* (Eq. 2-25) is defined based on the association metric.

$$w(i,j) = e^{\frac{-\|F(i)-F(j)\|_2}{\sigma I}} \cdot \begin{cases} e^{\frac{-\|X(i)-X(j)\|_2}{\sigma X}} & if \ \|X(i)-X(j)\|_2 < r \\ 0 & otherwise \end{cases} \qquad \textit{Eq. 2-26}$$

$$(D-W)y = \lambda Dy \qquad \textit{Eq. 2-27}$$

$$d(i) = \sum_{j} w(i,j) \qquad \textit{Eq. 2-28}$$

$$y = (1+x) - \frac{\sum_{x_i>0} d_i}{\sum_{x_i<0} d_i} \cdot (1-x) \text{, where x a vector with } x_i = \begin{cases} 1 & if\,(i \in A) \\ -1 & otherwise \end{cases} \qquad \text{Eq. 2-29}$$

The edge weights are given by Eq. 2-13 were *X(i)* is the spatial location of node *i*, and *F(i)* is the feature vector. Shi and Malik tested their algorithm with different features, depending on the application. For colour images, the features used are based on the *HSV* colour system, while features are also proposed for texture and grey-scale image segmentation. The algorithm suggested is based in solving the system of equations Eq. 2-27, where *D* is a diagonal matrix with *d* (Eq. 2-28) on its diagonal, and *W* a symmetrical matrix with *W(i, j)=w(i, j)*. Shi and Malik showed that the second smallest eigenvector is the real valued solution to the normalized cut problem, and is the one used in each iteration to split the graph, until the normalized cut exceeds a pre-defined threshold.

Other Clustering Algorithms

A morphological approach for 3D clustering in feature space is proposed by Park, Yun and Lee [143]. The first step of this approach is to smooth the 3D colour histogram by performing 3D Gaussian convolution with two standard deviations (σ_1 and σ_2). Then the difference of the two resulting histograms is considered and peaks and valleys are identified. After this pre-processing it is observed that non-empty bins are widely scattered in the colour space, therefore, a closing operation follows after which clusters are identified and labelled in the colour space. A dilation process comes next, enlarging the clusters in a manner so that neighbouring bins not contained to a cluster to merge with one, still preserving the clusters themselves (not combining any two of them). Finally, a post processing stage follows, where the remaining unsegmented pixels are assigned to a cluster by means of checking the colour distance (Euclidean distance in the colour space) between each unsegmented pixel and the segmented neighbouring pixels. The unsegmented pixels are assigned to one of the clusters of their neighbouring segmented pixels, according to their colour distance to it. The authors use *RGB*; nevertheless, the algorithm can be used with any colour system. Partitional algorithms (such as K-means or fuzzy c-means) partition the feature space based on a distance measure, whereas the proposed one is concerned with only the shape, connectivity and distribution of clusters.

2.2.4. Edge Detection Techniques

As mentioned in section 2.1.2, edge detection based segmentation techniques are based on the detection of discontinuities in the image. Edges are generally defined as the points where significant discontinuities occur. Discontinuities are considerably more complicated to define in colour images than in grey-scale ones. Numerous methods exist to derive edge information for each pixel when only one component is used, as in grey-scale images. The fact that more than one component (usually three) are used to describe the colour of each pixel in colour images, introduces the need of an additional step in the process of edge detection, namely the recombination of the image, which can happen in different stages in the pipeline of edge detection. Effectively, a set of operations is performed on each component and the intermediate results are then combined to a single output. According to the point at which recombination occurs, colour edge detection methods can be categorized as Output Fusion methods, Multidimensional Gradient methods, and Vector methods.

Output fusion methods work in each colour component independently, and the results are then merged to produce the final edge map. In multidimensional gradient techniques, a single estimate of the orientation and strength of an edge is computed. Finally, in vector methods, no decomposition (and therefore no recombination) of the image happens; instead, the vector nature of colour is preserved throughout the process. In the remainder of this section, approaches that fall in each of the above categories will be presented.

Output Fusion Methods

In output fusion methods, grey-scale edge detection is carried out independently in each colour component, and then the results are combined to produce the final edge map. One of the first output fusion methods was developed by Nevatia [131]. Nevatia defined a colour system of one Luminance and two Chromaticity components. The Luminance component is given as a weighted sum of the *R*, *G* and *B* components of the image, while the chromaticity components are chosen to be representative of Hue and Saturation. Hueckel’s edge detector is applied to each component separately, but although the edges in the three components are allowed to be independent, a constraint is applied, that they must have the same orientation. The author states that the edges in the Luminance component contain most of the information needed to obtain object boundaries, but does not underestimate the importance of edges in the two chromaticity components.

A method that combines edges found in the *H*, *S* and *I* components of a colour image is proposed by Weeks, Felix and Myler [202]. The colour system used is the *C-Y* system, which comprises three components, namely *Y*, *R-Y* and *B-Y*, where *Y* is a weighted sum of the *R*, *G* and *B* components representing the Luminance of the image. The *H*, *S* and *I* components are subsequently defined with the help of the *C-Y* ones. In order to avoid the 2π effect in the Hue component (imposed by the 2π modulus nature of the Hue space as explained in Appendix A), edges for the Hue component are instead computed in two derived components, $I=cos(\theta)$ and $Q=sin(\theta)$, where θ is effectively the Hue angle. This explains the decision to define the *HSI* components using the *C-Y* colour system since $R\text{-}Y=S\cdot sin(\theta)$ and $B\text{-}Y=S\cdot cos(\theta)$, which reduces the computational load of the method. Edges are computed for each component by applying the Sobel operator in the horizontal and vertical directions. Then the final edge map is produced by simply adding the edge results produced in each component. This method addresses the intrinsic 2π problem of the Hue component in an effective way based on the results reported.

Carron and Lambert [28] suggested a slightly different method to perform edge detection in colour images. The key point in their approach is the way the Hue component is treated. They defined the relevance of the Hue value of a pixel based on the Saturation value of the same pixel. When the Saturation of a pixel is low, Hue is rather unstable (if Saturation is zero, Hue is actually undefined), for this reason it should not be considered for any edge detection process. The relevance of the Hue component was defined by means of a sigmoid function, which assigns a zero value of relevance at Hues of pixels having a low Saturation and greater values to Hues of highly saturated pixels. For the edge detection process, the Sobel operator was used in the horizontal and vertical directions for each component and two ways were proposed to combine the information. Either regard the Hue information as a complement to Intensity and Saturation (simple adding the three gradients), or regarding the Hue information as more important, in which case the defined measure of Hue relevance was taken into account at the recombining stage. In further work by Carron and Lambert [29], both the Hue relevance measure and the way of combining the results of the three components were defined with the use of fuzzy inference.

Multidimensional Gradient Methods

One of the first multidimensional gradient methods proposed was by Robinson [161]. His work is focused on edge detection, but also on deciding on the proper colour system to use for it. For each pixel, Robinson applied eight directional masks to each component and computed *24* directional derivatives. Subsequently the one with the largest magnitude was chosen as the gradient at the specific pixel. Five colour systems were tested and a general conclusion according to Robinson is that edge detection should not be performed in the *RGB* colour space.

A comprehensive study on the gradient of a multi-image came later on by Di Zenzo [43]. He formulated the problem of computing the gradient in a multi-image and derived a solution based on the 1^{st} (gradient) and 2^{nd} directional derivatives of each component of the image. A *2*x*2* matrix is formed from the scalar product of the gradient vector in each component. Assuming $f:R^2 \rightarrow R^m$ is a continuous multi-image, the elements of the matrix are given by:

$$g_{hk}(x) = f_h(x) \times f_k(x) \text{, for } h,k=1,2, \text{ where} \qquad \text{Eq. 2-30}$$

$$f_h(x) = \left(\frac{\partial f^1}{\partial x^h}, \dots, \frac{\partial f^m}{\partial x^h} \right) \qquad \text{Eq. 2-31}$$

$$F(\theta) = g_{11} \cos^2 \theta + 2 \cdot g_{12} \cos\theta \sin\theta + g_{22} \sin^2 \theta \qquad \text{Eq. 2-32}$$

Then Di Zenzo suggests that the problem of finding the gradient's orientation and magnitude at *x*, can be reformulated as the problem of finding the value of θ which maximizes Eq. 2-32. This can also be expressed as the problem of finding the principal eigenvalue and eigenvector of a matrix produced by summing the *2*x*2* matrices produced over all components. Then the square root of the principal eigenvalue becomes the magnitude of the gradient and the corresponding eigenvector yields the gradient direction. Di Zenzo also considered the problem of deriving digital approximations for the gradient in a digital multi-image and suggested a straightforward application of the technique to colour images. The approach proposed by Di Zenzo has been widely used for edge detection in colour images.

A distinctly different approach has been proposed by Moghademzaddeh, Goldman and Bourbakis [120]. Their method starts with image smoothing, using an algorithm that preserves edges by checking the contrast between each pixel and pixel blocks in four directions. If the maximum and minimum contrasts are below a threshold, then there is little contrast around the centre pixel, thus the pixel is most probably not an edge and it's value is replaced by the average colour of the surrounding pixels. If only the minimum contrast is below the specified threshold, then the pixel is probably an edge pixel, and it's colour is computed as the average of the pixel colour and the block having the lower contrast to the pixel. If both the maximum and minimum contrasts are above a threshold, then the pixel is probably isolated and not similar to any of the neighbours, so it is considered noise and is removed. The edge detection part of the method is based on a set of three components, namely *h*, *i* and *s*, derived from the CIE $L^*a^*b^*$ colour system. The Hue component is considered important, since -according to the authors- it remains constant for the same objects regardless of shadows or illumination. Similarly to [29], Hue is weighted by the Saturation and Intensity components, since when any of them is small, Hue is unstable. This is defined with the help of fuzzy inference, by the creation of membership functions for the Saturation and Intensity components. As a result, a normalized Hue contrast can be calculated for each pixel in every direction. This is not adequate for edge detection, since edges caused by differences in any other component are obviously missed; therefore, the authors also compute the Euclidean distance in *RGB* for each direction. This distance is averaged with the normalized Hue contrast, and four local contrast values are computed, one for each direction. The maximum value and its direction is recorded for each pixel and if larger than a threshold, the pixel is considered an edge candidate, whereas if lesser than a smaller threshold it is discarded. If the value falls in a medium range, the contrast values are recomputed, but this time the blocks of pixels considered are moved one pixel apart from the centre pixel. Although this method addresses well the problem of Hue inconsistency when Saturation or Luminance is low, it is still an incomplete solution, since Hue is unstable or even undefined for high Luminance values as well as low ones. This is not addressed here, and is certainly not addressed in the previous approaches by Carron and Lambert [28, 29]. The authors of this approach also fail to explain the appropriateness of Euclidean distance in *RGB* as an additional measure of contrast, for edges missed by the normalized Hue contrast. One would expect that the edges missed by the Hue component, should be identified in the two remaining ones, namely the Saturation and the Luminance. Even if *RGB* is finally chosen, the Euclidean distance is not the best way to compute the contrast, for reasons explained already in section 2.2.1.

Vector Methods

An interesting edge detection method that does not perform any decomposition of the image into colour components has been suggested by Huntsberger and Descalzi [80]. The method is based on identifying clusters in the colour space, and assigning to each pixel membership values to the clusters identified. These values are subsequently used for edge detection. Specifically, the clustering part of the method is a fuzzy c-means clustering algorithm proposed by Huntsberger et al. [81] that has been reviewed in section 2.2.3. Very briefly, the algorithm starts with four clusters and introduces four new clusters at each iteration, until all pixels are assigned a membership value above a specified threshold. After this first step, each pixel has been assigned a certain membership value to each cluster. Edge pixels are then defined as the ones that equally belong to two clusters, in other words a color edge is defined as the zero-crossing of the operator $Edge_k(\mu_i,\mu_j)=\mu_i-\mu_j$, where μ_i and μ_j are the membership values of pixel k to the clusters i and j. A strong advantage of this method is that it is independent of orientation, and of any type of crisp or fuzzy threshold for the edge pixels; on the other hand, it is strongly dependant on the initial clustering and association of membership values to pixels. The authors examined different colour systems (*RGB*, *Ohta's* colour system, CIE *XYZ* and *RGB* with a Riemannian norm metric) and reported better results when using *RGB* with a Riemannian metric. They reason that this happens because such a metric induces ellipsoidal shaped clusters, which in turn match the shape of clusters derived from human colour matching experiments. Although this stands true (Mac Adam's ellipses [210]), if a colour space that presents some degree of perceptual uniformity is used, no complicated distance metric is actually needed.

A method that treats colours as vectors has been proposed by Yang and Tsai [213]. Their method is based on dimensionality reduction and moment-preserving thresholding of blocks of pixels in the image, followed by edge detection in each block. The i^{th} moment of a grey-level image is given by Eq. 2-33, where $f(x, y)$ is the grey value of pixel (x, y) and N is the number of pixels in the image. Moment-preserving bi-level thresholding is then the process of finding two thresholds

h_1 and h_2 and the associated fractions of pixels p_1 and p_2 with h_1 and h_2 respectively, in a way that the first three moments of the thresholded image are the same as the first three moments of the original one. This can be done by solving the set of Eq. 2-34. The method of Yang and Tsai starts by partitioning the image into *n*x*n* blocks of pixels. Moment-preserving bi-level thresholding is then applied to each colour plane of each block, yielding two representative colour vectors for each block: (R_1, G_1, B_1) and (R_2, G_2, B_2), the difference vector of which defines a certain axis in the colour space. All pixels of the block are subsequently projected on that axis, thus a 1D representation of each block is obtained. Edge detection is then performed in the block in question if Eq. 2-35 stands true, where τ is a preset threshold and σ_r, σ_g, σ_b are the standard deviations of the three colour planes of all image blocks. The edge detection bit is based on finding the chord of a circular window (defined so as to contain the block in question) partitioning the block in two areas and two grey levels (h_1 and h_2), so as the first three moments of the block are preserved, similarly to the bi-level thresholding described above. An edge is accepted if it observes the criterion given by Eq. 2-36, where σ_2 is the variance of the observed data.

$$m_i = \frac{1}{N}\sum_x \sum_y f^i(x,y), \; i=0,\ 1,\ 2,\ \ldots \qquad \textit{Eq. 2-33}$$

$$\begin{aligned} p_1h_1 + p_2h_2 &= m_1, & p_1h_1^2 + p_2h_2^2 &= m_2, \\ p_1h_1^3 + p_2h_2^3 &= m_3, & p_1 + p_2 &= 1 \end{aligned} \qquad \textit{Eq. 2-34}$$

$$|R_1 - R_2| + |G_1 - G_2| + |B_1 - B_2| \geq \tau(\sigma_r + \sigma_g + \sigma_b) \qquad \textit{Eq. 2-35}$$

$$|h_1 - h_2| \geq 2\sigma \qquad \textit{Eq. 2-36}$$

Dimensionality reduction of colour images is an ambiguous process in terms of whether it enables better results or not. It is a fact that it vastly reduces the computational overhead, but at the same time discards a great amount of information. Moment-preserving techniques aim to reduce this loss of information. Nevertheless, questions can be raised on the way the two colour vectors representative of each block were assembled from the thresholds that resulted for each colour plane in the method by Yang and Tsai. Specifically, no clear explanation is given to why the final vectors were *(R_1, G_1, B_1)* and *(R_2, G_2, B_2)* and not some other combination, for example *(R_1, G_2, B_1)* and *(R_2, G_1, B_2)*. Another issue in this method is the fact that only straight-line shaped edges are sought for. Unless the blocks used are small, line elements cannot be representative of edges, on the other hand, if small blocks are used, the number of pixels may not be adequate to perform statistical measurements such as computing meaningful moments. Finally, the way edge detection is performed, prohibits more that one straight edge element in each window, as for example near intersections of edges.

Edge Detection using Colour Distributions

A different approach to colour edge detection is described by Ruzon and Tomasi [168, 170]. They propose the use of the so-called compass operator, which is based on measuring the difference between the colour distributions of the pixels lying on opposite halves of a circular window taken in different orientations. A circular window is centred on each pixel, and for each orientation, it is partitioned in two hemi-circles. The distribution of pixel colours on each side is represented as a colour signature, that is a set of colour masses in a colour space. The size of each point mass is determined by a weighting function, which is defined as a function that approaches zero as we move away of the centre of the window. Vector quantization is performed before calculating the colour signatures, in order to reduce the number of colours, thus the computational cost of the algorithm. The distance between the distributions of the two halves is then computed for each orientation, using the Earth Mover's Distance (EMD) [167]. Given d_{ij} a distance measure between colours i and j, where colour i is in one hemi-circle (S_1) and colour j in the other (S_2), EMD finds a set of flows that minimizes Eq. 2-38. The distance d_{ij} between two colours ranges in [$0,1$] and is given by Eq. 2-37, where E_{ij} is the Euclidean distance of the colours computed in CIE $L^*a^*b^*$ and γ is a constant that determines the steepness of the function. The maximum and minimum values of EMD and the associated orientations are identified for each pixel. The maximum value gives the strength of the edge. The minimum value is equally important, because it can be considered a measure of the photometric symmetry of the data; when the minimum EMD is high, the edge model is violated, thus the authors call this minimum value "abnormality". Furthermore, a measure of uncertainty is proposed by the authors, related to the existence and the size of a plateau of maximum values for a range of orientations.

$$d_{ij} = 1 - e^{\left(-Eij/\gamma\right)} \qquad \text{Eq. 2-37}$$

$$a = \sum_{i \in S_1} \sum_{j \in S_2} d_{ij} f_{ij} \qquad \text{Eq. 2-38}$$

Based on the compass operator and the EMD distance between distributions, Ruzon and Tomasi extended the use of the compass operator to corner detection [169, 170]. The fact that distributions of colours are used instead of means, enables this approach to give accurate results in complicated situations based on the results presented. In a sense, the compass operator substitutes vector quantisation for smoothing, and the EMD for taking derivatives. At the same time, being a vector method, the method proposed by Ruzon and Tomasi avoids decomposing the image into separate colour planes.

2.2.5. Region Based Techniques

Generally, region based techniques for colour images can be categorized into region splitting, region growing and merging, and split and merge techniques, similarly to grey-scale images. More often than not though, region growing techniques are used. Region splitting techniques are typically based on histogram analysis and although frequently encountered, they are rarely used alone. A paradigm already reviewed in section 2.2.2 is the histogram-based method proposed by Ohlander et al. [135]. Split and merge techniques are rarely encountered in the context of colour image segmentation. A split and merge algorithm will be described next, while the rest of this section will focus on selected region growing techniques.

Region Split and Merge Techniques

A method based on recursive split and merge has been proposed by Gauch and Hsia [58]. The variance of the pixels in a region provides an indication of how homogeneous the region is. Although the variance of a region in a grey-level image is easy to compute, the problem is slightly more complicated when it comes to colour images. The trace of the covariance matrix for a region is instead used here. The mean colour of a region can be calculated by Eq. 2-39. Then the trace of the covariance matrix is given by Eq. 2-40.

$$M = (m_1, m_2, m_3) = \left(\frac{\sum_i c_{1i}}{N}, \frac{\sum_i c_{2i}}{N}, \frac{\sum_i c_{3i}}{N} \right) \qquad \textit{Eq. 2-39}$$

$$T = \frac{\sum_i (c_{1i} - m_1)^2 + \sum_i (c_{2i} - m_2)^2 + \sum_i (c_{3i} - m_3)^2}{N} \qquad \textit{Eq. 2-40}$$

where m_1, m_2, m_3 are the means of each colour component for the given region, (c_{1i}, c_{2i}, c_{3i}) is the colour of pixel i, and N is the number of pixels in the region. If this value is above a specified threshold, the corresponding region is recursively split, whereas if the value is below this threshold the region is added to a list of regions to be subsequently merged. The merging process can also be ruled by a condition based on the trace of the covariance matrix. The value is computed for the merged region and if below a threshold, the two regions can be merged. Alternatively, a comparison between the average colours of the regions could be used to rule the merging process.

Simple Region Growing Techniques

The second algorithm investigated by Gauch and Hsia [58] is a typical seed based region growing technique. They commented on a number of techniques to perform seed based region growing based on colour distance between the regions. The importance of re-computing the average colour of each region when a new pixel is added to it was stressed, since otherwise the outcome would be very much dependant on the initial selection of the seed pixels. Searching all the adjacent pixels of a given region at each iteration of the region process to identify the one with the smallest colour distance to the mean colour of the region would be preferable, but computationally expensive. Instead, the authors search for any neighbour that is within a specified colour distance to the mean colour of the region. As expected, the speed improvement is remarkable, while the authors report that no noticeable difference in the segmentation results is observed. Finally, the authors suggested that in order to obtain results independent of the initial selection of seed points, one should make sure that the seeds selected are not edge pixels. An edge detection algorithm can be used to identify which pixels to avoid. The authors concluded that the best colour space to use depends strongly on the type of the image in question, nevertheless they generally favour *RGB* and *YIQ* to $L^*a^*b^*$ and *HLS*.

A rather simple colour segmentation algorithm was proposed by Tremeau and Borel [197]. Their algorithm is based on region growing and region merging, which they perform simultaneously. Starting with a pixel as the seed, they grow a region checking the colour similarity between the pixel and its neighbours. After each region is identified, it is checked to the neighbouring regions, and if the difference of their average colours is under a specified threshold, the two regions are merged. The colour distance used in both cases is the Euclidean distance in the *RGB* colour system. Finally, the authors also propose methods to estimate the thresholds used both for pixel and for region similarity. The method proposed suffers from a number of drawbacks. First, the choice of *RGB* and the Euclidean distance to calculate colour differences is rather a disadvantage, since *RGB* is neither related to human perception of colour, nor it is perceptually uniform so as to justify the choice of Euclidean distance. The authors do mention that the method is generic, and propose the use of CIE $L^*a^*b^*$ or CIE $L^*u^*v^*$ and Mahalanobis distance instead, but take no step in that direction. Furthermore, the algorithm proposed is very dependant on the sequence of pixel and region comparisons. Although this is true for all region growing methods, the fact that the comparison of each region to the neighbouring regions takes place immediately after the region's creation (prior to having identify all the possible neighbouring regions), thus comparing with only the regions identified up to that moment, introduces an even larger dependency on the sequence of comparisons. On the other hand, this approach limits the number of tests and results in a computationally inexpensive process.

A colour segmentation method for video-conferencing type images was proposed by Ikonomakis, Plataniotis and Venetsanopoulos [82]. The colour system they use is the *HSI*, as a perceptual oriented one. The innovative point of this method is the different treatment between chromatic and achromatic pixels. An achromatic pixel is defined as having Intensity at the edges of the Intensity scale or low Saturation. For these pixels, Hue would not be indicative of their colour, since it tends to be unstable (even undefined for extreme Intensity values) at these ranges of Saturation and Intensity. For achromatic pixels, the homogeneity criterion is defined in terms of their Intensity difference, whereas for chromatic pixels, four different distant metrics were tested. These are the generalized Minkowski metric (Eq. 2-41), the Canberra metric (Eq. 2-42), and a metric defined by Tseng and Chang [198] called cylindrical distance metric (Eq. 2-43). The cylindrical distance metric computes the distance between the projections of the pixel points on a chromatic plane.

$$d_M(i,j) = \left(\left|H_i - H_j\right|^{\alpha} + \left|S_i - S_j\right|^{\beta} + \left|I_i - I_j\right|^{\gamma}\right)^{\frac{1}{\delta}} \tag{Eq. 2-41}$$

$$d_{can}(i,j) = \frac{\left|H_i - H_j\right|}{\left|H_i + H_j\right|} + \frac{\left|S_i - S_j\right|}{\left|S_i + S_j\right|} + \frac{\left|I_i - i_j\right|}{\left|I_i + I_j\right|} \tag{Eq. 2-42}$$

$$d_{cyl}(i,j) = \left(d_I^2 + d_C^2\right)^{\frac{1}{2}}, \textit{ where} \tag{Eq. 2-43}$$

$$d_I = \left|I_i - I_j\right| \textit{ and } d_C = \left(S_i^2 + S_j^2 - 2 \cdot S_i \cdot S_j \cdot \cos\theta\right)^{\frac{1}{2}} \tag{Eq. 2-44}$$

The method proposed is based on a classical region growing scheme, working in a left to right and top to bottom fashion growing regions from seed pixels by making appropriate comparisons (chromatic / achromatic) to their neighbouring pixels. According to the authors, the Cylindrical Distance metric produces better results. They also report that the Minkowski metric gives better results if they put more emphasis in the Saturation component, which is rather unexpected and directly contradicts to one of their previous statements that the Hue component has a greater discrimination power. The differentiation between chromatic and achromatic pixels can be a good enhancement. It is exploited in a different manner in the method described later on in Chapter 4.

Region Growing Techniques Based on Combined Spatial and Colour Measures

A comprehensive approach for region growing in colour images is given by Moghaddamzadeh and Bourbakis [119]. The authors define a set of procedures, which they combine in two different ways defining a method designed for coarse segmentation and one for fine segmentation. A number of pre-processing steps are used by the authors, in order to identify edges in the colour image. First, the image is smoothed using an algorithm that preserves certain pixels located at edges (same as in edge detection approach [120] by the same authors reviewed in the previous section), and then the edges are identified in the image, by the use of an algorithm explained in a bit more detail in section 2.2.4. Two conditional criteria are defined that are used extensively throughout the region growing stage of the algorithm: the homogeneity criterion and the degree of farness measure. The homogeneity criterion checks the absolute colour distance between a given pixel and a segment, and the local colour distance between the pixel and its neighbouring pixels in the direction of expansion, in order to decide whether the given pixel can be merged with the segment or not. By checking the colour distance locally, this criterion keeps the segment growing if the colour is gradually changing (due to illumination or shading). The second criterion, the degree of farness measure, combines the spatial distance of a pixel to a segment, with the colour distance between the pixel colour and the segment average colour. The two distances are multiplied to produce the degree of farness measure.

Segments are identified by scanning the image and growing any possible seed based on either the edge information (produced at pre-processing) or the homogeneity measure. Only segments having a size over a specified threshold are considered good. A segment expansion procedure is also defined, which takes into account either of the two criteria defined before. Finally, a procedure is defined that checks for unassigned pixels and decides whether they should be assigned in a neighbouring segment, or – if adequately different – it grows a segment around the pixel in question. The method defined for coarse segmentation is essentially based on repeatedly finding segments and expanding segments, calling the procedures first for large segment sizes and subsequently for smaller ones. This coarse segmentation method finishes by checking for unassigned pixels as mentioned before. The fine segmentation method differs in the way segments are identified in the first place. Instead of performing region growing, a histogram table is constructed, which contains a sorted list of all the colours and the number of occurrences of each one. Segments of the colours presenting higher occurrences are identified and expanded first, followed by colours presenting lower occurrences.

In recent work [215] the authors applied the dichromatic reflection model in addition to the two criteria described above to merge highlight and shadow areas with matte areas in the image. Although a comprehensive approach, the two methods suggested by Moghaddamzadeh and Bourbakis suffer from certain drawbacks mainly related to the two criteria defined. The homogeneity criterion works well, by allowing for gradients to be included in the segments, while at the same time checking the colour distance of the pixels examined to the colour of the segment, thus not allowing for extensive colour changes. The problem though lies on the way colour distance is computed. The colour system used throughout this approach is the *RGB*, and the colour distance employed is the Euclidean distance in the *RGB* colour space. As mentioned in section 2.2.1, *RGB* is not perceptually uniform; therefore, a distance measurement in that space does not give exact information about colour difference. A possible second drawback could be the way the spatial distance and the colour distance are combined to give the degree of farness measure. A direct multiplication is not necessarily the best way to combine the two, as it often results to crisp decisions. A fuzzy technique like the one described in Chapter 5 could possibly give better results. Nevertheless, the aforementioned approach is quite comprehensive and innovative and addresses issues often missed by other researchers, such as the need to intelligently combine spatial and colour attributes when segmenting colour images.

Exactly the fact that *RGB* is not suitable for measuring colour differences points out Schettini [177], who proposes instead the use of $L^*h^*C^*$ colour system. Initially the *RGB* values are converted to CIE $L^*u^*v^*$, which exhibits perceptual uniformity to a much higher degree than *RGB*. Subsequently, two psychological features, namely Hue (h*) and Chroma (C*) are derived from the CIE $L^*u^*v^*$ components. The method proposed by Schettini starts by performing histogram directed colour clustering, based on recursive one-dimensional histogram analysis (see section 2.2.2). The output of this first step is an over-segmented image where connected regions are identified and labelled. At this point, a region merging process takes place, based on colour similarity and spatial proximity of regions. Fisher distance is adopted to verify the spectral homogeneity of adjacent regions. This takes into account both the mean value and the variance of the colour feature. The spatial characteristic employed is a measure of connectivity based on the length of the shared boundary between two regions. A similarity function that combines the two is then defined by multiplying the two measures. The region merging strategy is based on the region adjacent graph, in which each node represents a region, and two nodes are joined by an arc when the corresponding regions are adjacent. The merging process then is carried out iteratively, by simultaneously merging all those pairs of regions mutually most similar, until all the adjacent regions have a similarity (as computed by the similarity function defined) greater than a specified threshold. This approach addressed a number of issues missed by the previous one by Moghaddamzadeh and

Bourbakis [119]. The colour system used is this time perceptually uniform, and the spatial property used is the length of the shared boundary between regions, which is superior to the Euclidean distance as will be seen in Chapter 6. In addition, the process used for merging the regions is not much dependant on the order of mergers, while it is also suitable for parallel processing, which are both great advantages.

One of the disadvantages of region growing and merging techniques is their inherently sequential nature, which often results to computationally expensive methods. Furthermore, the final regions produced by such methods are strongly dependant on the order in which regions are grown or merged. On the other hand, region growing and merging techniques allow for more flexibility during the segmentation process, since they work locally in the image. Finally, although any prior information on the image contents is useful, region based techniques generally do not need any kind of prior information to work.

2.3. Discussion

Although certain approaches for colour image segmentation date as back as in the early sixties, much progress has only been observed in the last two decades. This is mainly due to the fact that computer systems able to represent true colour information were made affordable to the average computer user only lately. On top of that, the rapid growth of the World Wide Web, and the consequent need for transferring vast amount of colour image information, gave a boost to research related to colour image processing, analysis and compression.

Trends in Colour Image Segmentation

Certain trends can be identified in regard to colour image segmentation by examining the methods presented here. Regarding the colour spaces used, a certain tendency is observed towards creating methods that respect the characteristics of human perception. To this end, one of two actions is usually taken: Perceptually uniform colour systems such as CIE $L^*a^*b^*$ or CIE $L^*u^*v^*$ are used whenever a colour distance measure is needed. Alternatively, colour systems that resemble the perceptual factors that enable humans to differentiate colours (the most prominent of which are Hue, Lightness and Saturation) are used. In certain cases, both steps are taken, that is the colour system finally used is a Hue, Lightness, Saturation one, defined directly on a perceptually uniform system instead of *RGB*.

In the context of colour image segmentation, fuzzy logic (explained in more detail in Appendix B) is used in a number of methods to describe the relative importance of the different colour components of the colour system used (e.g. [29, 120]). Although fuzzy logic can be used to define combined metrics (e.g. the farness measure [215]), it is not widely used. A successful application of fuzzy logic to define a metric combining the colour distance and the topological relation between connected components is described as part of the segmentation method suggested in Chapter 5.

Regarding the techniques employed, by examining the literature it can be seen that researchers opt at creating methods combining different techniques, aiming at exploiting the advantages of each one. A categorization of methods was attempted in this chapter therefore mostly methods representative of individual categories were reviewed. Nevertheless, some hybrid methods were also presented here, such as the method of Schettini [177], which combines recursive histogram analysis and region merging. More hybrid methods are available in the literature, for example a modified split and merge algorithm [36, 41] where splitting steps are performed with respect to the edge information and merging is based on grey value statistics of merged regions.

Colour Image Segmentation in the Context of Web Images

Most methods available in the literature for colour image segmentation focus on images of natural scenes. Undoubtedly, this encompasses a wide range of applications, such as analysis of photographs, video sequences, robot vision and many more. Techniques have been also proposed for specialized applications, such as aerial/satellite imaging, medical imaging etc. Although the range of the proposed techniques indisputably covers most areas of interest, there are still certain cases, where most methods are inapplicable and different approaches must be sought. A prominent example where this stands true is the field of artificially created images especially images created with the use of computers. Most colour images used in the World Wide Web fall into this category, and as analysed in Chapter 1 that denotes a different range of advantages and disadvantages.

Specifically, regarding the Web Images that are of interest of this research, that is the ones containing text, there are a number of facts that have to be accounted for before adapting any of the methods reviewed here. A basic problem is the type of segmentation sought in this case. The objective of the colour segmentation methods reviewed here is to partition the image in areas that exhibit similar colour properties.

Although this might be useful for Web Images, the main objective in this case is to separate the text from the background, and as mentioned before in Chapter 1, the text does not always exhibit uniform colour properties. In contrary, text can be gradient or multi-coloured, thus rendering inapplicable most of the methods. Nevertheless, it is of our understanding that whatever the variation of colour in the text and in the background, there must be adequate difference between the two, so that humans can read the text. Moreover, the text in such images is supposed to create impact, which reinforces the previous statement.

To summarize, regarding colour segmentation of Web images, the very definition of the objective of segmentation must change to reflect the desired output of separating the text from the background. Towards this end, document analysis techniques are probably most appropriate and are reviewed in the next chapter. Some of the methods reviewed here can also be used for segmentation of Web images, as long as special consideration is taken in order to produce partitions that can contain colours of a higher variation. This is needed in order to contain the cases were gradients or multicolour characters are present. Higher tolerances in respect to colour difference can be set in order to achieve that.

One more fact that might pose problems in adapting certain colour segmentation methods for use with web images is the existence of problems inherent to the file formats used. As an example, one of the most commonly used formats for images in the World Wide Web, where small sized images are easier to communicate, is *GIF*, which only supports 8-bit indexed colour information. This introduces colour quantisation that can be a problem to certain methods, especially those working in the feature space. Unwanted peaks can be produced in histograms due to that, and points of great frequency can appear in the 3D colour space impeding colour clustering. *JPEG* compression can also introduce problems due to the compression scheme used. Methods that work in each colour component separately will have problems identifying meaningful segments, because artefacts produced by discarded information are especially noticeable in certain components.

Conclusion

This chapter gave an overview of colour image segmentation techniques. Selected methods were reviewed for each category. Although particular attention was paid to review current approaches, many older methods were also included for completeness, were applicable. The theoretical background of this research concludes with the next chapter, which examines aspects of text image segmentation and classification.

Chapter 3

Text Image Segmentation and Classification

Segmentation as defined in the previous chapter, is the process of partitioning an image into a number of homogeneous disjoined regions. This definition has to be slightly altered when referring to text image segmentation. The product of text image segmentation is a higher-level description of the image in hand, in terms of regions of interest. Regions of interest typically are areas in the image where text lies. Depending on the application, other structural elements of the text image, such as tables, figures, separators etc. might also be regions of interest. After having identified a number of regions, the type of contents of each region must be established; this process is called classification. Classification of the regions can take place either independently after segmentation, or in parallel, interacting with the segmentation process.

The underlying assumption here is that text exists in the image being analysed. Consequently, specialised segmentation and classification methods make extensive use of features emanating from the existence of text in the images. Images found in the World Wide Web more often than not, contain text; therefore, they constitute a special type of text images. A review of the main text image segmentation and classification techniques will be given in this chapter. A number of bi-level page segmentation methods will be detailed in the next section, followed by methods specialised in colour documents. In Section 3.3, methods for text extraction from video sequences will be discussed and in Section 3.4 the more generic problem of finding text in real-life scenes will be addressed. Finally, Section 3.5 will detail the few existing approaches for extracting text from Web Images. A discussion on the methods presented follows in Section 3.6.

3.1. Bi-level Page Segmentation Methods

In the context of Text Image segmentation, Page segmentation has received much attention mainly due to its many applications. Page segmentation specialises in analysing document images. Typically, document images are bi-level (most often dark ink over light background). In bi-level images of document pages, the two classes, namely the foreground and the background, are already separated: pixels of one colour denote background and pixels of the other denote foreground. Regions of interest in this case would be neighbourhoods of foreground pixels.

Since the foreground and background are already separated, the way regions are described in the page can be slightly relaxed. What this means, is that strict definitions of the regions, such as connected components of foreground pixels, can be used but this level of detail is not always necessary (and many times, not easy to use). Less relaxed descriptions of the regions of interest, for example by use of bounding boxes or contours of regions, can also be used. Such descriptions though, allow part of the background to be contained in the final regions. This is not a problem for bi-level images, since the foreground and background are readily separated by means of their colour. In contrast, including parts of both the foreground and background in the region description is unacceptable for colour documents, since a separation of the two classes cannot be easily achieved afterwards.

In general, all methods devised for bi-level images assume either implicitly or explicitly that the two classes (foreground and background) are already separated. Due to this fact, most bi-level page segmentation techniques (e.g. techniques based on Projection Profiles or Analysis of the Background) are not directly applicable to colour text image segmentation. Nevertheless, there are a number of techniques (e.g. Connected Component Grouping, Segmentation by Image Transformations) that in principle can be applied to colour images.

Morphological Operations

A number of techniques based on morphological operations are commonly used either as a pre-processing step (e.g. noise reduction) or as part of the segmentation process itself [71]. The morphological operations typically used are dilation and erosion, and the combinations of them: opening and closing. Dilation is the process of converting to foreground colour the pixels in the immediate neighbourhood of foreground-coloured pixels, thus expanding foreground areas and possibly restoring lost links between them. The invert process is called erosion. A dilation process succeeded by erosion is called closing, while the opposite process is called opening.

Although morphological operations cannot be directly used with colour images, Run Length smearing (which has the same effect as a closing operation applied in one direction only) can be performed based on colour similarity between pixels. Run Length smearing connects together foreground pixel runs in a row or column of a binary image according to their distance. In the context of colour images, given a run of pixels we can check for a run of pixels of similar colour in the same row or column, and if the distance between the two is less than a predefined distance, connect the two.

Connected Components Grouping

A connected component is a set of connected pixels of the same colour. A description of a document image in terms of connected components, is the lowest level of representation possible. Connected components may be extracted in numerous ways. A labelling process is probably the approach most widely used. An initial foreground pixel is identified and assigned a tag, then all its neighbouring foreground pixels are marked with the same tag and the process continues by progressively expanding the neighbourhood to include more pixels of the same colour. Other ways to extract connected components is by contour tracing, and as the final stage of projection profile analysis. Connected components can be represented by the runs of pixels they comprise, in terms of their contours, or by their bounding box, that is the minimum enclosing rectangle of the component.

Connected components are used typically in bottom-up approaches, which group together components to form progressively higher descriptions of the regions of the document. Usually, before grouping connected components together, a classification step takes place which aims to deduce the type of each one. The most common distinguishing characteristic of connected components of different types is their size (width or height is typically used, depending on the orientation of the printed text). Another set of characteristics has to do with the overall shape of components. The Aspect Ratio (the ratio width/height or height/width depending on the particular method) is the most widely used such characteristic, while others based on the perimeter and the area of the component, are also in use. Finally, features based on the complexity of connected components such as the number of black to white, or white to black transitions, as well as structural features have been used.

Connected components are usually grouped together based on their between distance. The thresholds used are usually derived from the components themselves: either the dimensions of the most populated class of connected components [38] or analysis of the spacing between them [132]. Usually components of the same type only are grouped together [53]. Nevertheless, methods have been proposed that group together closely placed components irrespectively to type [171].

Methods based on connected component analysis can generally be used for segmenting colour documents. A connected component in this case would be a set of connected pixels of similar colour. The definition of "similar colour", depends on the application, and can strongly affect the result of segmentation.

Projection Profile Analysis

A global feature of the images used in numerous page segmentation methods is projection profiles. A projection profile is a histogram of foreground pixel occurrences along consecutive parallel lines whose indication is perpendicular to that of the profile. Projection profiles have been used for estimating the skew angle of the document image [1, 10] as well as to detect the orientation of the text lines [1, 85]. As far as it concerns page segmentation itself, projection profiles are typically used in top-down segmentation methods, in order to locate horizontal and vertical streams of background space. The assumption here is that printed regions are rectangular and separated by background space. Usually the document image is recursively split into rectangular regions [42, 128, 129]. The level of detail of the process can be very low (columns or paragraphs [199]) or very high (details of characters [127]). The very essence of calculating a profile in an image requires that the foreground and the background are already separated, or some other form of segmentation has already been performed. In this second case, projection profiles derived from the regions produced (connected components or bounding boxes [10]) can be used.

A method based on projection profile analysis which is somewhat unaffected by small skew angles of the document is suggested by Parodi and Fontana [144]. They subdivide the document into overlapping small columns of height equal to the image height. For each of them, the projection profile is computed and non-white scan lines called line elements are extracted. The line elements are ultimately combined to find the text lines. By using small columns the authors drastically reduce problems associated with skew (according to the authors, skew angles of up to *10* degrees do not pose any problem).

Analysis of the Background

The background space of the image can provide information about the layout of a document. The positions in the image of long horizontal and vertical stripes of background colour would yield the positions of separators between logical entities of the image, such as columns, paragraphs, headers and footers. Baird [11] and Spitz [187] perform an initial segmentation and subsequently analyse the space not covered by the connected components produced, in order to identify stripes of background colour. This can be seen as a post-processing step, in which a higher-level description of the image is achieved, after an initial segmentation has already been performed.

Antonacopoulos [5] on the other hand, uses background information as the means for segmenting the image. He argues that background space always surrounds the printed regions in the document image, and produces a description of the background space by means of tiles. Using this description, the contours of regions of interest are identified. In contrast to previous methods, which would only search in horizontal and vertical directions, Antonacopoulos approach is unaffected of the skew angle of the document.

Segmentation by Image Transformations

A somewhat different approach to the segmentation problem, is based on the fact that regions of different type possess different characteristics. By applying some global or local transformation to the image, it is possible to identify regions of certain characteristics. This process essentially performs segmentation and classification at the same time.

A widely used characteristic for distinguishing regions of printed text, is that character lines are linear. Based on that, methods exist that make use of the Hough transform to determine various parameters for a page. The skew angle of the document [74] and the interline spacing [52] are some of them. Srihari and Govindaraju [188], based on Hough transform determine the textural signature of a text line. Then they test individual blocks of the image to decide whether they contain text or not, based on the existence of this textural signature.

Generally, regions can be classified by their textural properties. The distinct texture produced by straight lines of characters placed in a paragraph is quite regular comparing to the more diverse texture of graphic regions. The periodicity of character lines is used by Hase and Hishino [72], who propose a method based on the Fourier transform. They apply a Fourier transform to the whole page, and the period between lines is identified. Subsequently, regions of the image are tested and if the same period is observed, they are classified as text.

Textural properties of regions can be used for classifying regions of colour documents as well. Simple measures of periodicity or co linearity are difficult to be used without some preliminary knowledge of the foreground colours, but in principle, many aspects of textural based segmentation are applicable to colour documents.

Greyscale Document Images

In the broad sense of document image segmentation, greyscale documents fall rather in the category of bi-level segmentation methods, rather than colour ones. The reason for that is that more often than not greyscale documents are first binarized, by means of some thresholding technique, and subsequently processed as bi-level documents.

Numerous methods have been suggested for binarizing greyscale documents. They range from methods using predefined fixed thresholds [88] to more complicated methods, using image features to derive the threshold values used [48, 133]. For example, Sauvola and Pietikainen [175] split the document image in rectangular windows, and compute the values of two features for each window: the average grey value, and the transient difference (which measures local changes in contrast). The two values are combined with the help of a fuzzy inference system and the output is used to select the proper binarization algorithm. Thresholds are then calculated for every nth pixel and interpolation is used for the rest in order to reduce the computational time.

It should be made clear here that there are cases where segmentation is performed directly on the greyscale images. Generally, such methods exploit the gradient information of the images, and refer to a topographic analogy where background regions (lighter colours) are like “valleys” and foreground regions (darker colours) appear as “furrows” carved on the background [178, 185]. An example can be found in Muge et al. [123] who make use of the gradient of the image, and apply a series of basic directional morphology operators to obtain the expected segmentation.

3.2. Colour Page Segmentation Methods

Segmenting colour documents is understandably a much more complicated task than segmenting bi-level or even greyscale ones. Many researchers avoid going into the field of colour document analysis, by converting the colour documents into greyscale ones, or by examining the lightness component only. They argue that text (especially when rendered in a single colour) would still stand out in the greyscale representation, as a single shade of grey, adequately different from that of the background.

For example, Goto and Aso [62] propose a method for analysing colour images of complex background, which is based on a greyscale representation of the original image. They assume that characters in a single text string are printed in a solid, uniform colour. The method starts by applying multi-level thresholding to the greyscale image to create a set of sub-images for each range of grey values as indicated from the histogram of the image. An initial pixel labelling process takes place in the sub-images, and then region growing between neighbouring (in terms of grey ranges) sub-images follows.

According to the authors, the proposed method performs much better than methods created purposely for colour documents, specifically the methods of Ohya et al. [137], and Sobottka et al. [183, 184]. The method of comparison though, is by feeding the greyscale version of the image into those methods as well, which might be unfair at least for the method of Sobottka et al. since it is created to work with full colour images. The method of Sobottka et al. (discussed in this section) specialises in book and journal covers, while the method of Ohya et al. (detailed in Section 3.4) is created for recognizing text in scene images.

Colour Reduction for Colour Document Analysis

Since full colour information can be difficult to manipulate and computationally expensive to use, many researchers suggest that some type of colour reduction be performed before processing. The necessity of colour reduction for scanned colour documents in particular, is further dictated by the nature of the scanning process. Due to the characteristics of the optical scanner, scanned documents contain more colours that the original printed document. In the specific case of colour documents, it is of great importance that no information is dropped which concerns the textual parts of the document, since that would hinder the segmentation and subsequently the recognition process.

A segmentation method based exactly on colour clustering is proposed by Worring and Todoran [208]. The document model they use, assumes that each document can be decomposed in a number of (possibly overlapping) frames of arbitrary shape, the content of which might be text or pictures. They restrict this model to documents where the background colour of each frame is uniform. The authors suggest that transitions from one colour to another (for example at the edge between two frames), would be rather smooth due to the printing and scanning process, unlike the step edges appearing in the original. In the colour space, these smooth transitions would appear as lines connecting the two (clusters of) colours. The method aims in identifying those lines in the colour space. Initially, the *N* most dominant colours are selected from the *RGB* histogram therefore *N* clusters are identified. This number of clusters is subsequently reduced by combining clusters that lie in close proximity in the colour space. This process, takes place in already reduced colour space. Lines are then identified in the colour space by means of edge detection (in the colour space). Having identified a set of lines, the colour of each pixel is checked, and the lines that present a distance to that colour less than a predefined threshold are identified. If one line is identified, the pixel is assigned to this line; otherwise, spatial characteristics are used to facilitate the decision process. Although the suggested method can produce satisfactory results in simple colour documents, the number of assumptions made does not allow for a general use.

Perroud et al. [147] examine two histogram based clustering approaches, one in the *RGB* space and a second in the *RGBY* space, where *Y* is a spatial component, which represents a quantity from the image plane. The histogram based approach, in its simplest form (*1D*), is based in analysing a quantised version of the histogram. For each cell in the histogram, the two neighbouring cells are checked and a pointer to the larger one is created. If both neighbours are equal and larger than the cell in question, a pointer to the left one is created by default, whereas if none of the neighbours is larger, no pointer is created. At the end of this process, the histogram contains chains of cells pointing to a local maximum. Clusters can then be defined with the help of those chains. When this method is extended to three dimensions, as in the case of *RGB* clustering, each cell in the quantised *3D* histogram has *26* neighbours. Chains could then be identified in a similar way, by checking all *26* neighbours each time. The suggestion of the authors was to consider spatial information as well as colour during the histogram based clustering. Therefore, a fourth dimension, namely *Y* is employed. The four-dimensional histogram contains information about the number of occurrences of a colour value dependent on its spatial position. In this case, the image row is taken as the spatial position (i.e. the *Y* component). A different quantisation step is used for the *Y* component, since it represents a quantity of the image plane. Other than that, each cell will have *80* neighbours in the *4D* space, and the process of creating chains works in the same manner as in simpler cases. If after the end of the clustering process the same line of text spreads over multiple *Y*-cells, an over-segmentation is possible. To overcome this problem, a post-processing step is used. The post-processing procedure checks if above and below the separating line of two cells there are adjacent pixels that belong to the same *RGB*-cell of the histogram. In such a case, the clusters are merged. The authors suggest that clustering in the *RGBY* space is superior to clustering in the *RGB* space since the use of spatial information allows a direct segmentation of the documents into text components. Nevertheless, this is based on the fact that text lies on horizontal lines (unless a different spatial feature is used), and is of the same colour. These two conditions are necessary for a line of text to fall in the same cluster.

Scanned Colour Documents

In the context of colour documents, an interesting approach that combines a bottom-up and a top-down segmentation process is proposed by Sobottka et al. [183, 184]. They use the previously described algorithm of Perroud et al. for colour reduction as a pre-processing step. The clustering method used is in RGB colour space (3D), while they report that the HSV colour space was also tested, and performs poorer than RGB in this case (it produces too many clusters). The top-down approach proposed is based in recursively splitting the image into regions in the horizontal and vertical directions. Regions containing text include at least two colours. Based on this knowledge, homogeneous regions are rejected as background ones. Each row or column (according to the direction of splitting) is tested to see if it contains one or more different colours. If a row (column) is found in which all pixels have the same colour, the region is split along this row (column). Splitting terminates if at least 2 colours arise for all rows and columns of the region. During segmentation, information about possible text and background colours is acquired: the colour of the rows or columns where split occurs is considered to be the background colour, while another colour that occupies a certain percentage of the region is considered to be the text colour. The top-down segmentation accurately segments small sized text, while over-segmentation of the characters is not possible. The bottom-up process acts in a complementary manner to the top-down process. A region growing is performed here: beginning with a start region of at least three horizontally or vertically aligned pixels of the same cluster, pixels within a *3x3* neighbourhood are iteratively merged if they also belong to the same cluster. This bottom-up approach presents difficulties segmenting very small characters, while it can split graphic regions into sub-regions. Regions are grouped into lines of text, based on basic text line hypotheses (distance between components, co-linearity of characters, distance between lines). This region grouping is performed independently for the results of each process, and the identification of text lines makes use of the fact that both types of analysis predicted the same output. This method assumes that both the background and the text (at least at the character level) are of uniform colour. Although the majority of documents conform to those specifications, a number of complex images where text is of gradient colour, or where text is rendered on photographic background would possible pose certain difficulties.

Another method, which is not specifically designed for colour documents, but for a broad range of colour images containing text, comes from Jain and Yu [86, 87]. They propose a method based on decomposing the given image in a number of foreground images and one background one, by using colour information. They use slightly different approaches for 8-bit palettised images and for 24-bit true colour ones. For 8-bit images the authors argue that characters occupy a sufficiently large number of pixels, so a foreground image is created for each palette entry with a number of corresponding pixels larger than a predefined threshold (*400* pixels). Therefore, each foreground image contains pixels of the same colour. Furthermore, the number of foreground images accepted is limited to eight. The colour with the largest number of pixels is considered the background colour. Also, as background colour is considered the colour with the second largest number of pixels, if that number is above a predefined threshold (*10.000* pixels). The authors further assume that if the text colour is not uniform, the surrounding background colour will be. Based on that, they produce one more foreground image, which consists of all the pixels that do not belong to the background. The process for true colour images is similar, but some pre-processing takes place first. Bit-dropping is performed to each of the RGB components, and only the highest two bits of each one are kept (this reduces the maximum number of colours to *64*). Subsequently, colour quantisation takes place, which further reduces the number of colours. From that point onwards, the extraction of foreground images is the same as with 8-bit images. In each of the foreground images produced connected components are identified. The smaller of the connected components are discarded, while the rest are combined into lines based on size similarity and alignment in the image. The components that are not part of any text line, are further checked (for the case of characters touching each other) by use of horizontal and vertical profiles. Similarly to other methods described, the method of Jain any Yu suffers from a number of assumptions that do not always stand true. The assumption that the background is of uniform colour may make sense for some colour documents, but not for other types of images such as video frames (Section 3.3), Scene Images (Section 3.4) and web images (Section 3.5) the method is supposed to cope with. In fact, the authors present a low accuracy rate for documents with complex backgrounds. The second disadvantage of this method is the fixed thresholds. Since the thresholds used refer to numbers of pixels, it would be more reasonable to be dependant on the size of the image, and not pre-defined to specific values.

Short Discussion

Unlike bi-level images, or greyscale images, where a bi-level representation of the image can be obtained by thresholding, in colour images separation of foreground and background is not a trivial task. Segmentation here involves not only a rough description of regions containing text, but actually extracting the text, and ultimately making it possible for a subsequent process to obtain a binirised version of the image for recognition purposes. The majority of segmentation methods suggested for bi-level images are not directly applicable here.

3.3. Text Extraction from Video

Text extraction from video sequences has been paid much attention lately. Extraction, in contrast to segmentation, has the broader meaning of identifying regions of interest and actually separating the foreground from the background information. Text extraction from video scenes refers mainly to extracting text superimposed on video scenes, like captions, or film credits. This task can be facilitated from the fact that this type of text either remains at the same position for a number of consecutive frames, or scrolls independently from the background scene. Therefore, the analysis of the differences between consecutive frames or motion analysis can facilitate this process enormously. Not limited to that type of textual information, methods have been proposed, which aim in extracting text from video scenes; that is from the actual content of the video sequence. These techniques are conceptually identical to extracting text from real-life scenes, which will be analysed in Section 3.4.

Extraction of Superimposed Text

Lienhart and Stuber [102] propose a method to extract text from video sequences. Their method is limited to text superimposed on video frames (captions, end credits etc). They work with greyscale instances of the video frames, and base their method in several characteristics that superimposed text possess, such as characters being rigid and of uniform colour, text being static or linearly moving in one direction (when scrolling), size restrictions etc. The method starts with a colour clustering process, based on the Split and Merge algorithm proposed by Horowitz and Pavlidis [75]. The split process begins with the whole image as a segment, and splits it in quarters checking each time the colour homogeneity of the sub-segments produced. If a sub-segment is not homogeneous, splitting continues, otherwise splitting stops. The merging process, checks the final segments and if neighbouring segments have similar average grey value they are merged. This produces a number of final regions, which initially are text candidates. A number of tests are then performed, and most of the regions are discarded. The first test discards regions of very large or very small size. Then motion analysis takes place, and the remaining regions are matched with regions of consecutive frames. They argue that text should remain unchanged in shape, rotation and colour, and it should be displaced only for a small distance (if at all). Based on that, regions without an equivalent in the subsequent frame are discarded, so are those the equivalent of which has significantly different average grey value. The next test has to do with contrast analysis. Since superimposed text is placed in an image specifically to be read by the viewers, it should be sufficiently different from the background (usually outlined text is used for that reason), which would produce strong edges. Therefore, strong edges are identified in the image and dilated. The regions that do not intersect with any dilated edge are discarded. Finally, an extra filtering is performed based on the fill factors and the aspect ratio of the remaining regions. This method is completely based on special characteristics that according to the authors superimposed text posses. The characteristics suggested are quite logical and stand true for most of the cases.

A different method is proposed by Kim [93], which works in the *RGB* colour space. The colour histogram of the image is first computed, and dominant peaks are identified in it. The colours corresponding to each of the peaks identified, are extracted in a separate colour plane, and text extraction is performed in each colour plane individually. This automatically limits this approach to characters of uniform colour (since otherwise they would have been exported in a different colour plane). First, certain components are discarded based on their shape and positioning. Those are components touching the borders of the picture, or excessively long ones. Only pixels of a certain colour (the corresponding histogram peak) lie in each colour plane; those pixels are considered the foreground ones, while the rest consist the background. Each row of the colour plane is examined, and it is determined whether it has enough foreground runs of pixels to be part of a horizontal text line. Rows having too many or too few runs of foreground pixels are discarded. If more than three consecutive rows pass this text, the stripe defined is considered as a text candidate. A similar procedure then takes place during which columns of each horizontal stripe are checked to determine whether they include any foreground pixel or not. In this way the horizontal stripe is split into a number of regions, which are subsequently tested against some more text heuristics (aspect ratio restrictions, density etc). Finally, all the remaining regions are grouped again into horizontal lines. Similarly to the method by Lienhart and Stuber [102], this one is limited to text superimposed in the image, horizontal and of uniform colour.

Quality Improvement of Extracted Characters

Sato et al. [174] address a different problem of text extraction from video sequences: the low quality of the extracted text. Based on the fact that a text line appears in a series of consecutive frames, they use this excess information to improve the quality of the characters. For the extraction of text regions a method devised by the same research group [182] is used, which works with greyscale instances of the frames. The first step towards improving the resolution of text regions is to employ a sub-pixel interpolation technique. They effectively magnify the text region by a factor of four by placing the original pixels every fourth row and fourth column, and calculating the values of the intermediate pixels by linear interpolation. More comprehensive techniques to improve the resolution of text areas are available in the literature (e.g. Thouin and Chang [193, 194]), but linear interpolation is adequate as a first step here. For the second step of this method, the authors assume that captions have high intension values (lighter colours than the background). Based on that, an enhanced image (of each sub-pixel interpolated frame) is produced from the minimum pixel value in each location that occurs during the frames containing the region. Subsequently, four filters are defined (horizontal, vertical, left diagonal and right diagonal), and each one is applied in the enhanced images produced by the previous step. The rationale behind this filtering is that characters consist of those four different diagonal elements. The results of the four filters are then integrated, and the final image is thresholded. Finally, projection profiles are used to split the regions into characters, and basic character recognition is used to select the best final segmentation. Although restricted by the fact that captions are expected to be of light colour, this method addresses a problem which is very common to video (and web image as will be seen later on) text extraction: the low quality of (small) characters.

Extraction of Scene Text

Li et al. [100, 101] extended their method to both text superimposed in the video frames, as well as text appearing as part of the scene. Their method uses a *16*x*16* window to scan the image and classify each region as text or non-text. They move the window by four pixels every time, so that the algorithm is reasonably precise and fast. The authors argue that the regions where text is printed will present higher frequencies (due to stronger edges between the text and the background) than other non-text regions. Consequently, they aim in analysing the scale space of the image to identify text regions. To do that, they decompose the image using wavelets into four sub-bands. The text regions as expected show high activity in the high-frequency bands of wavelet analysis. Having decomposed the image into four bands, for each *16*x*16* window the mean, the second and the third order moments (Eq. 3-1) are computed in the four sub-band images. The second and the third order moments of the four sub-bands are used as the features for classification of the window as either text or non-text. For the classification process, a neural network is used, trained with a big number of text and non-text samples. After each window has been classified, a number of regions (based on the pixels of text windows) are identified. The projection profile of the canny-edge map of each region is then used to extract the text elements.

$$E(I) = \frac{1}{N^2}\sum_{i=0}^{N-1}\sum_{j=0}^{N-1} I(i,j)$$

$$\mu_2(I) = \frac{1}{N^2}\sum_{i=0}^{N-1}\sum_{j=0}^{N-1}\left(I(i,j) - M(I)\right)^2 \qquad \text{Eq. 3-1}$$

$$\mu_3(I) = \frac{1}{N^2}\sum_{i=0}^{N-1}\sum_{j=0}^{N-1}\left(I(i,j) - M(I)\right)^3$$

Short Discussion

Scene-text extraction can be seen as an introduction to Section 3.4 (Text extraction from Real Scene), the only difference being that in video scenes more than one frame exists containing the same text, while in real life scenes just one image is available. In contrast to superimposed text, where the location of the text is generally static or linearly changing, scene text is moving with the scene, and is affected by camera movement. As a result, scene text is difficult to track, and research has been conducted on methods to track text regions in video sequences. Li et al. [101] based on the method discussed before propose a technique for tracking the text regions identified. Another suggestion comes from Cui and Huang [37], who propose a method for extracting characters of license plates from video sequences. They argue that as vehicles are moving along the road, the trajectories of points on the licence plates are parallel in *3D* space, and will have a common vanishing point in the image plane. They select a number of points (based on a rank value each pixel of the licence plate has been assigned) and use them to calculate the trajectory of the licence plate. By doing that, they can correct any perspective distortion the plates might have.

3.4. Text extraction from Real Scene Images

Real scene images are usually photos containing text. Characters in real scene images can suffer from a variety of noise components. They originally exist in 3D space, so they can be distorted by slant, tilt, and the shape of objects they are printed on. Moreover, uncontrolled illuminating conditions often result to text of non-uniform colour.

Methods Limited to Horizontal Text

Ohya et al. [137] present a method for the extraction and recognition of characters in real scene images. They work with grey-level images, and start by binarising the image using local thresholding. For local thresholding, the image is split into sub-blocks, and a threshold is determined for each one. Then the thresholds specified are interpolated to the whole image. For sub-blocks of a uniform grey value, performing thresholding yields a number of unwanted noise components; therefore, a bimodality check is usually employed to determine whether a sub-block should be thresholded or not. The authors though, argue that such a bimodality check would result in loosing real character segments, so they threshold every sub-block of the image instead. The characters in a real scene image are not necessarily only black or only white, so the method does not favour one situation in particular. Instead, the method identifies all components in the image, and initially selects character-like ones by assessing the grey differences between adjacent components.

Character-like components are expected to have adequate difference from their adjacent ones. Components in close proximity that have the same bi-level value (after binarizasion) or similar average grey values are identified and marked as possible parts of characters. This process addresses the problem of multi-segment characters (like "i", "j" or some Chinese characters) and aims in creating character pattern candidates. Character pattern candidates are then classified. This classification process involves checking the similarity between the character pattern candidates and a set of character categories stored in a database. High similarity patterns are then post-processed with a relaxational operation, in order to remove ambiguities. The algorithm is reported to work well with a number of different types of images: road signs, licence plates, signs of shops etc, which include a variety of characters. Due to the nature of the classification process, the characters are required to be printed horizontally.

Another approach comes form Wu et al. [209]. They propose a two-step method to segment an image into text regions, and a robust (but not very generic) way to binarize the extracted text regions. First, a texture segmentation algorithm takes advantage of certain characteristics of text to segment areas possibly containing text lines. The second phase focuses on the previously detected areas and constructs "chips" from strokes taking into account text characteristics. This phase starts by creating strokes from significant edges in the picture. Strokes that are unlikely to belong to any horizontal text string are then eliminated, while the remaining ones are aggregated to form “chips”. Next, the “chips” are filtered, and the ones unlikely to belong to any horizontal text line are eliminated. Finally, the remaining “chips” are aggregated again to describe the text lines more accurately. For these processes, the authors assume that characters of the same text line should be of similar height, and horizontally aligned. This process is repeated for a number of scaled images of the original, in order to identify text of various sizes. All the results are finally combined. After segmenting the text lines, each region identified is smoothed, and the intensity histogram of the smoothed region is computed. The text (assumed to be of adequate difference from the background) should give one peak in the histogram (left or right, depending whether it is black or white), while the background should give another peak. Based on histogram analysis, a value is selected at the valley of the two peaks, and the image is thresholded. Although the authors suggest that this method can be used for scene images, the assumptions they make are not that generic. Text is assumed horizontal, and of uniform size, which is certainly not the case for text found in scene images, where camera perspective can distort the text and the positioning of the objects carrying text can be arbitrary.

Methods Unaffected by Different Orientation Angles

A method which better addresses problems of different orientation angles of text lines in real scene images is proposed by Messelodi and Modena [117]. The method proposed, does address skew problems, but it is only described and tested on images of book covers, where acquisition takes place in a real environment, under unconstrained conditions, but the positioning is carefully adjusted so that the vertices of the book cover approximately match the corners of the image frame. Moreover, the colour contents of the image are not always representative of real scene images, since book covers generally have text printed on a uniform background. Having mentioned that, this method does address the problem of lines having different orientations in the image, as well as problems of touching characters or lines. The process starts by obtaining a number of bi-level images. For the book cover images, this is achieved by performing first intensity normalization and then analysing the histogram of the resulted (normalized) image. This histogram is strongly unimodal, since flatted regions are typically the major part of the image. Those flat regions will have been assigned similar grey values after intensity normalization, so they appear as a single peak in the histogram. Text, graphics and other non-textual components, which are usually darker or lighter from the background, make contribution to the histogram tails. Consequently, two thresholds are selected (and two bi-level images are produced), to capture the left and the right tail of the histogram. Components are then identified in the bi-level images and filtered according to several heuristics, which aim at characterizing single textual objects. Component characteristics used are dimension, convexity, local contrast and elongation (aspect ratio).

During the last step, the remaining components are grouped to produce a set of text lines. The components are clustered into text lines by means of a hierarchy divisive procedure. Initially a single cluster contains all the components, which represents the root node of a tree. Then a set of expansion rules is used to recursively expand each node. A generic node of the tree is represented by a structure with numerous fields like the direction of the block of components, the width and height of the block etc. To compute the direction of the block, a number of potential angles are generated from pairs of components, and a projection profile is generated for each angle. The projection profile with the minimum entropy corresponds to the correct angle. The expansion rules are based on a second set of heuristics concerning the characteristics of groups of components belonging to the same line of text. Such characteristics used are closeness, alignment and comparable heights. First, a “closeness segmentation” is applied once, which creates clusters of components based on their topological proximity. At this stage, areas of text printed in different angles are exported in different clusters. Then “alignment segmentation” is performed which aims in separating blocks of similarly oriented text to text lines.

Clark and Mirmehdi [35] go one step further, and suggest two approaches to the location and recovery of text in real scenes. The first method, is focused on situations where text lies on the surface of an object and the edges of the object are rectangular and aligned with the text. This is the case for paper documents, posters, signs, stickers etc. The objects on which text is printed have strong visible edges, and depending on the camera perspective and the positioning of the object in the scene, the rectangles around text regions will appear as quadrilaterals in the scene image. Such quadrilaterals are extracted from the image with the use of edge detection, Hough transform and grouping of extracted line segments. Quadrilaterals that do not refer to objects containing any text are then eliminated. A confidence measure is employed for this process, based on the histogram of edge angles for each region. Because the printed text independently of magnification has a closed shape, the histogram of the edge angles will have rotational symmetry *180°*. This property is used to define the confidence measure used for rejection of non-text regions. Having identified the quadrilateral describing a text region, the perspective of the text region can be removed.

In cases where text is not surrounded by a clearly identifiable quadrilateral, Clark and Mirmehdi suggest another approach, based on identifying the vanishing point of text lines. For this method to be used, paragraphed text (of at least three lines) is required. The text regions are initially identified using texture characteristics of text. Five measures are defined for each pixel, based on a small circular neighbourhood around it. These measures are combined with the use of a neural network, which classifies each pixel as part of a text region or not. Having identified text regions in the image, the vanishing point of the text lines has to be computed. To do that, the authors defined a circular search space for which each cell $c = (r, \theta)$ corresponds to a hypothesised vanishing point $\boldsymbol{V}(V_r, V_\theta)$ on the image plane, with scalar distance $V_r = r/(1\text{-}r)$ from the centre of the image and angle $V_\theta = \theta$. A projection profile of the text is generated for every vanishing point. Then similarly to [117], the projection profile with the minimum entropy corresponds to the correct skew angle. Having identified the vanishing point, segmentation into text lines and deskewing takes place.

Short Discussion

Many similarities exist between scene text extraction from video frames and real scene images. All approaches created for detecting text in scene images should be straightforward to use for text extraction from single video frames. Similarly, most of the methods proposed for scene text extraction from video, with the exception of methods making use of the fact that text appears in multiple frames, can be used for text extraction from scene images.

Even though a number of approaches have been suggested, there is no generic way to extract text from video frames or scene images. Almost all methods expect text to be of uniform colour (e.g. [35, 117, 137, 209]), while a number of methods expect text to be printed in a specific direction (e.g. [137, 209]). Another common assumption, either implicit or explicit to the method, is that the background of text regions is of uniform colour (e.g. [35, 117]) or that the characters have high contrast with their background (e.g. [137, 209]). Finally, most of the approaches proposed, deal with a grey-level version of the original colour image. These characteristics hinder the use of many of those methods with images taken in real environment, under unconstrained conditions.

3.5. Text Extraction from Web Images

Web Images are generally computer created. As such, they belong to a special category of images called synthetic images. As described in the introduction of this thesis, the fact that Web Images are created with the use of computers, to be shown on computer monitors, entails a number of characteristics, and inherent problems. The problems associated with Web Images, are not the typical problems one expects to find in scanned documents or video frames and scene images like skew, 3D perspective, illumination or scanning artefacts, noise etc. Instead, artefacts produced by compression or anti-aliasing are common, while text itself, as a result of the artistic expression of the creator, can appear colourful, in various orientations, outlined or shadowed etc. The main characteristic of Web Images though, is that text is rendered in colour, and most of the times, either the text or the background (or both) are multi-coloured. As can be seen, extracting text from Web images can be an extremely complicated process.

Comparing to previous categories of image text examined here, text in Web Images is created with the image. In that sense, Web Image text extraction is closer to the extraction of superimposed text on video frames, rather than scene text, or scanned documents.

The main contribution to the specific problem of Web Image text extraction comes from Lopresti and Zhou [105, 106, 218-220]. They propose two methods to locate text in images, as well as methods to recognize text, and although they make a number of assumptions that do not always hold, their approaches can produce satisfactory results to a significant sub-group of Web Images, namely images stored as GIF files (8-bit palettized colour).

The main underlying assumption to the method proposed by Lopresti and Zhou for locating text in Web Images [218], is that text is printed in uniform colour. This contradicts to the main characteristic of Web Images, that text is usually multi-coloured. This assumption automatically limits the applicability of the method to a certain subset of Web Images. Based on that assumption, the text extraction method is based on identifying in the image connected components with the same colour. First, colour quantisation is performed by clustering in the RGB colour space based on the Euclidean minimum-spanning-tree technique (EMST). According to EMST, each colour is represented as a node in a tree, and each pair of nodes is connected by a weighted edge, the weight of which equals the distance (RGB distance) between the two nodes (RGB colours). By eliminating the edges with the larger weights, the tree is partitioned in disjoined clusters. Pixels of the same colour are then grouped in connected components, which are subsequently checked to evaluate if they could be text candidates. The evaluation is performed in a similar manner to a number of previously detailed methods: based on geometrical features of the components such as size, aspect ratio, the presence of holes and branches etc.

In more recent papers [106, 220], Lopresti and Zhou changed slightly the clustering algorithm of their method to include spatial information as well as colour information. This was done mainly to address a problem inherent to images of a limited palette of colours such as GIFs. Dithering is sometimes used to create colours that are not defined in the palette used. In such a case, to create a colour, pixels of two (or more) colours are mixed in a certain way in an area of the image, so when seen from a distance, its colour appears to be the mixture of the two colours participating. If that is the case with text, the simple clustering algorithm, would separate the two colours (in different clusters), resulting in splitting the text. In order to avoid that, the authors introduced a measure for the spatial proximity of colours. Let $d_p(X)$ denote the distance between a given pixel p and the closest pixel with colour X. Denote by N an m x m neighbourhood in the image, and by $c(p)$ the colour of pixel p. Let $P_N(X)$ represent all pixels of colour X in the neighbourhood N. Then the spatial distance from colour X to colour Y in the neighbourhood N is calculated by Eq. 3-2, where $\#P_N(X)$ is the number of elements of $P_N(X)$. Because the distance defined in Eq. 3-2 is not symmetric (i.e. $D_N(X,Y) \neq D_N(Y,X)$) the distance defined in Eq. 3-3 is used.

$$D_N(X,Y) = \frac{1}{\#P_N(X)} \sum_{p \in P_N(X)} d_p(Y) \qquad \textit{Eq. 3-2}$$

$$\overline{D}_N(X,Y) = \frac{D_N(X,Y) + D_N(Y,X)}{2} \qquad \textit{Eq. 3-3}$$

The images are first divided into non-overlapping *m*x*m* blocks and each block is processed. The blocks are small enough so that each region is roughly bi-tonal. A local clustering algorithm then exports a small number of clusters from each block (one to three). Initially pixels of each block are grouped to three clusters, and then the method decides based on the distance described here, whether pairs of clusters should be merged. After this pre-processing step at the local level, the EMST algorithm runs with the already colour reduced image. The character-like connected component identification takes place as before, based on geometrical characteristics of the components, but now a second step is added. After using features that are relatively invariant to character touching (e.g. stroke width or white-to-black ratio) to identify character-like components, the elongated components are singled out, and a post-processing step splits those components based on breaks identified in their vertical profile. For this step, the method further assumes that text is printed horizontally. Finally, a layout analysis step takes place, which combines components in words, and checks the "goodness" of each grouping by use of a measure called *saliency,* based on the degree of height and the positional alignments of the characters in a word.

Apart from Web Image text segmentation, Lopresti and Zhou suggested two methods [105, 219] for performing OCR on the extracted text regions. A foreground colour is chosen for each region (based on the assumption that the text is of uniform colour). Then the method computes the difference between that colour and the colour of each pixel in the region. The result forms a 3D surface, which is used for recognition. The recognition method proposed is based on a polynomial surface fitting proposed by Wang and Pavlidis [200]. Each character is treated as a 4th degree polynomial function. This is done by least square fitting of the polynomial function on the 3D surface of the character. Features derived from that polynomial representation are then used to match the character to a database of prototypes. The problems with that approach are two. First, the polynomial surface representation method is a computationally intensive operation. Second, the polynomial representation can capture only the global shape of characters. Characters whose shapes are similar to each other (e.g. "*c*" and "*e*") may not be distinguished reliably.

Another OCR method suggested by Lopresti and Zhou, is based on n-tuples. An n-tuple is simply a set of locations in an image with specified colours. For recognition, n-tuples are superimposed onto an image and colours are compared. A degree of how much an n-tuple matches to the underlying area is then calculated. To use n-tuples, each pixel is assigned a value in the range [*0,1*] representing the certainty of it to belong to the foreground. N-tuples is an old OCR technique, which due to a number of problems it presents has received only scant attention.

In a similar manner to Lopresti and Zhou, Antonacopoulos and Delporte [7] propose a way to extract characters from Web Images. They improve on the previous suggestions in two ways. First, by providing support for full colour (JPEG) images, which represent a great percentage of Web Images. Furthermore, gradient characters extraction receives more attention. For full colour images, the authors perform a bit dropping operation, keeping only the *3* most important bits of each colour component, thus reducing the maximum number of colours to *512*. Subsequently, colour clustering is performed to further reduce the number of colours in the image. Two approaches to colour clustering are suggested. The first is based on analysing the histogram of the image, and ordering the colours according to their prominence. The most dominant colour is then taken as the centre of the first cluster, and colours presenting a distance less than a pre-defined threshold to the centre of the cluster, are assigned to it. The most important of the remaining colours is then selected and a new cluster is created. The process continues until all available colours have been assigned to a cluster. The second clustering method suggested is the Euclidean minimum-spanning-tree technique (EMST) as described previously. This second algorithm is used in more complex colour situations. Subsequently, a connected component analysis is performed, during which special attention is paid to gradient components. Subsequently, components are evaluated based on certain features, like size, aspect ratio and the number of strokes crossed by each image scanline. This last feature, essentially measures the black-to-white transitions, which cannot be more than four on a scanline (case of letter "M"), at least for the Latin character set. Then components are evaluated are grouped in words based on their colour, their alignment, and proximity in the image.

A framework for analysing Web Images is proposed by Koppen et al. [96]. The framework consists of four stages, the colour seperation stage, the information granulation-specification modules (GVMs), the task stage and the recognition stage. Each image is initially split in colour components. Five sub-images are produced, one for each component of the CMYK colour system, and a fifth one for the Saturation component of the HIS colour system. Each sub-image is then used as input to one or more GVMs. The GVMs are designed to solve specific problems, they consist of three parts: a *method maintainer*, a *parameter chooser* and a *tester*. The parameter chooser provides the method maintainer with a set of parameters for each run of the method on the input images. Then the resulting images are tested and all successfully processed images are put into a queue, the task manager then decides for the further processing of the results: it either feeds them into more GVMs or passes them to the recognition stage. The GVM of interest here, is the one specifically designed for text extraction. The input of the text location GVM is each colour component sub-image. Text regions are detected from their structural properties, while a filtering operation enhances the contrast in the image. The authors argue that the structural property indicating a sequence of characters is the repetition of nearly parallel edges in a region of the image. Parameter switching is performed, and for each set of parameters an output image is produced. Each output image is evaluated by means of topological features of the extracted components. The positively evaluated images are then fed into the recognition GVM. The authors mainly describe the framework in this paper, rather than specific modules, so specific details for the text extraction module are not available.

Short Discussion

Not many methods have been proposed specifically for extracting text from Web Images. The variability of Web Images does not allow for most methods to work but only with a minor percentage of images. For example, Zhou and Lopresti [105, 106, 218-220] only deal with 8-bit palettized images, where the text appears in uniform colour. Antonacopoulos and Delporte [7], deal with 24-bit images as well, but also require text to be of uniform colour. Moreover, all methods require text to be horizontally printed and generally not touching. As can be seen, only a limited number of Web Images conform to the specifications set here. For the rest, there is essentially no method suggested in the literature.

3.6. Discussion

A number of text segmentation and classification approaches were presented in this chapter. Although text segmentation and classification was examined in the context of various different fields, some common trends can be identified.

Trends in Text Extraction and Classification

The first such trend has to be related to the location of text in images. Most of the approaches presented use either connected component analysis, or texture analysis to identify regions of interest in the image. Connected component analysis is generally used after some pre-processing has been performed. Usually, this pre-processing aims in splitting the image in question in a number of sub-images in which pixels are separated in foreground and background ones. Another type of pre-processing, used before connected component analysis, is bit-dropping, or colour clustering (or both), in order to reduce the number of colours in the image. Then a colour is selected as the foreground one, and connected component analysis can be performed. Texture analysis on the other hand, requires much less pre-processing as is based on textual properties.

No matter what type of analysis is used to extract text from images, there is always a number of underlying assumptions for every approach. Those assumptions are usually reasonable and not many. The most common of them, is that the text is rendered in uniform colour. This is actually needed for almost all connected component based methods with few exceptions such as Antonacopoulos and Delporte [7] who provide for gradient text. On the other hand, the majority of textual analysis based methods require that the text presents sufficient contrast to the surrounding background.

Numerous more assumptions come to play during the classification process. Text is usually considered horizontally aligned. The sizes of characters of the same line are considered similar, which alone is quite reasonable, but it automatically suggests that no characters are touching, or broken in smaller components. Some methods even require that the aspect ratio between characters on the same line is similar, which is not true for all the characters of the Latin alphabet, where vertically elongated characters like “l” and “i” coexist with square characters like “x” and “o” and horizontally elongated ones like “m” and “w”.

Finally, most of the methods show a tendency towards working with grey scale versions of the images. Although it is certainly computationally efficient to work in a *1D* colour space than a *3D* or *4D* one (depending on the colour system used), a certain amount of information is discarded.

Conclusion

This chapter gave an overview of text segmentation and classification techniques. Generally, a number of good approaches exist to segment text in rather simple situations: text of uniform colour, horizontal text lines, high contrast between the text and the background etc. Nevertheless, Web Image text does not generally fall in any of those categories. The rest of this thesis will focus in novel methods for extracting text from Web Images.

Chapter 4

Split and Merge Segmentation Method

The first of the two methods investigated towards text segmentation in Web images is presented in this chapter. This method works in a split and merge fashion, aiming at constructing connected components that ultimately describe the characters in the image. The two parts of the method, namely the splitting process and the connected component aggregation (merging) process, are explained in detail. Certain aspects of the method that are of particular interest are also detailed in dedicated subsections.

4.1. Basic Concepts – Innovations

Both text segmentation methods that were produced as part of this research share a common belief about the way Web Images are constructed. As explained in Chapter 1, Web Images are created to be viewed by humans, thus it is reasonable to expect that particular colour combinations are used that enable humans to differentiate (relatively easily) between the textual content of the image and the background. Based on that observation, the common denominator both methods share is their anthropocentric character. Although approached in a distinctly different way in each method, the safe assumption that a human being should be able to read the text in any given Web Image is the foundation of both methods' reasoning.

A number of colour segmentation methods based on human perception exist in the literature as detailed in Chapter 2. Nevertheless, the field of colour document analysis lacks of perceptually oriented segmentation methods. The methods described here aim to fill this gap, by applying human perception information in the field of text segmentation in colour images. Moreover, the manner such information is exploited here, presents significant differences from perceptually oriented colour segmentation methods found in the literature, as will be shown later on.

Anthropocentric Approach

Towards addressing the observations made and implementing the anthropocentric character of the method, two stages are employed in the Split and Merge method. First, the splitting process of the method is based in the HLS colour system, which provides a colour representation much closer to the way humans understand colour than RGB. Then, the component aggregation phase utilises biological data available about human colour discrimination in order to assess the colour similarity of different image regions.

The use of HLS as a perceptually oriented system has been previously explained in the context of colour segmentation in earlier chapters. Research has been conducted in order to assess the usefulness of a number of factors towards colour discrimination [181]. The outcome of this work and related research is that wavelength separation, colour purity and lightness are the factors that influence most the perception of chromatic colour. A list of further factors that influence chromatic colour discrimination is given in Appendix A.

The Hue component of the HLS system corresponds to the perceived wavelength of each colour, thus higher differences in Hue are translated to better wavelength separation. As wavelength separation increases, the ability to discriminate between colours increases accordingly. The Saturation component corresponds to colour purity. Increases in the purity of colours maximize the perceptual distance between them, whereas colours with a low saturation value are difficult to be separated with regard to their chromatic content. Finally, changes in Lightness also influence greatly the appearance of colours. Increases in Lightness produce increases in Saturation, thus, more bright colours would appear more saturated and would be easier to distinguish. It should be noted that this increase of Saturation as the intensity of the illumination grows (or as the luminance of the emitted light is raised) reaches a different maximum for each Hue [189], beyond which further increases in intensity produce a reduction of Saturation. It should also be mentioned here that changes in Hue also occur as Lightness changes, a phenomenon known as the *Bezold-Brucke Effect* [156, 211], although such level of interaction is outside the scope of the research presented here. Based on the above information, the suitability of HLS for colour analysis of images can be well justified. This type of information about factors enabling colour discrimination and the interaction of colour attributes is taken into account in the first part of the Split and Merge method.

The next stage of component aggregation employs existing biological data on Wavelength discrimination and Lightness discrimination, involving the appropriate HLS components in accordance to the factors explained above. Each type (Wavelength, Lightness or Colour Purity) of discrimination information is used at each level of processing, working towards the final segmentation of the image.

4.2. Description of the Method

The method starts by a pre-processing step, which aims at separating the *chromatic* from the *achromatic* pixels of the image (see next section). The image is split in two layers at this point, one containing the achromatic pixels and another containing the chromatic ones.

Subsequently, the splitting process takes place. The histogram of Lightness for the pixels of achromatic layer is computed, and peaks are identified. A short analysis of the peaks identified follows, where peaks corresponding to similar Lightness values are combined. The left and right minima (see Section 4.4.2) of each peak (or combination of peaks) define a range of Lightness values. For each range of Lightness values, a new sub-layer is introduced, and the corresponding pixels are copied over.

In a similar manner, the histogram of Hues for the pixels of the chromatic layer is computed, peaks are identified and the chromatic layer is split into sub-layers of different Hue ranges. For each of the sub-layers produced from Hue histogram analysis, the Lightness histogram is computed and the process is repeated. This goes on, alternating the component used at each step until a specified number of splits are performed or until only one peak can be identified in the histogram. Following this process, a tree of layers is produced, where the original image is the root of the tree, and the layers produced are the nodes.

After the splitting process is finished, a bottom-up merging process takes place. Connected components are first identified in each of the bottom layers (leaves of the tree). Then the neighbouring pixels of each connected component are checked, and if "similar" to the component, they are flagged as a potential extension for it. Similarity depends on the type of layer, that is, if the layer in question was produced by splitting its immediate parent layer based on the Hue histogram, then Hue discrimination data are used to assess if a viewer can possibly differentiate between the Hue of the component and the Hue of the neighbouring pixel. Similarly, if the layer was produced by splitting based on the Lightness histogram, Lightness discrimination data are used. By the end of this phase, connected components have been identified in each of the bottom layers, along with potential extensions for each one of them. These potential extensions are referred to as *vexed areas* from now on.

Starting with each of the bottom layers (leaves), the overlapping of pairs of components (and their vexed areas) is computed (see Section 4.5.3) and if greater than a specified threshold, the two components and their associated vexed areas are merged into one new component (with a new vexed area). After this process finishes at the bottom layers, the resulting components are copied one level up, and their vexed areas are refined according to the type of the layer they are copied into. Then the same process of component aggregation based on overlapping is performed and the process continues, working its way up towards the root of the tree. The merging process stops when the original image layer is reached. Each of the above steps is described in detail next.

4.3. Pre-Processing

In this section, a brief anaphora to the transformations used between colour systems is given, and the pre-processing step of separating the chromatic from achromatic pixels is detailed.

Transformations between Colour Systems

The image data (RGB) is not directly available in HLS, thus a transformation is needed between the two. The RGB and HLS colour systems are interchangeable, meaning that there is a 1:1 correspondence between RGB and HLS values. The transformation used here is the one proposed by Foley et al. [55] and is detailed in Appendix A.

Although HLS is used throughout this method, biological data are given in physical units (e.g. wavelength instead of Hue) and a way to translate them into HLS is needed. This is done with the help of CIE XYZ, which acts as an intermediate system, since it is directly defined on physical data. Although the conversion to CIE XYZ depends on the hardware used, as explained in Chapter 2 and in Appendix A, the majority of monitors conform to the standard set by *ITU-R recommendation BT.709* [84] within some tolerance. The transformation suggested by *Rec.709* (which is also the standard for the sRGB colour system) is therefore used here to convert from RGB to CIE XYZ and vice versa. The transformation matrix employed is given in Appendix A.

Separation of Chromatic from Achromatic Pixels

Before any analysis begins, the separation of chromatic from achromatic pixels in the image occurs. *Chromatic colour* is any colour for which a dominant wavelength can be identified. As such, any colour that one can characterize as red, green, blue, purple etc. is a chromatic colour. On the opposite, if no dominant wavelength can be identified for a given colour, that colour is said to be *achromatic*. Any shade of grey, including black and white, is achromatic.

Separating chromatic and achromatic pixels at this stage is important for the following reason: any process that examines Hue values will fail if applied to achromatic pixels, since the Hue for those pixels is undefined or unreliable. No specific Hue value can be given to any pixel that has zero saturation (thus is grey), so by definition a Hue value of zero is assigned. If the histogram of Hues is computed at that point, all grey pixels will be counted as having Hue equal to zero, (also equivalent to being red). If saturation is not zero, but just very low, a Hue value can be defined, but still it should not be used. The reason for that is twofold. First, the Hue value assigned is very unreliable for small Saturation values, so it doesn’t carry much useful information. Second, and probably most important, the human viewer cannot actually perceive any Hue for a pixel with a low Saturation, instead they will just see it as a grey pixel.

Pixels with very high or very low values of Lightness, also lack any chromatic information (as explained before), so they should also be identified as achromatic ones.

The magnitude of the problem can be illustrated by the Hue histograms in Figure 4-1. It can be seen that the original histogram is not representative of the Hue content of the image, since a large number of pixels are actually achromatic.

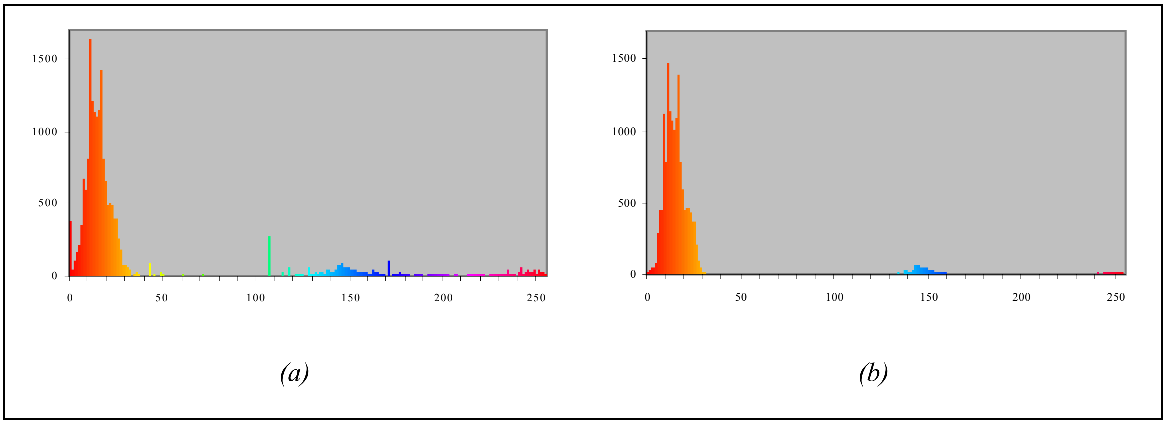


Figure 4-1 – (a) Original Hue histogram. (b) The same histogram, after removing all achromatic pixels.

The exact levels of Saturation and Lightness for which colours should be considered achromatic are difficult to set. This is due to a number of factors, mainly the interaction of different colour attributes. Biological information exists on the amount of pure Hue needed to be added to white before the Hue becomes detectable [126, 210]. This data is shown in Figure 4-2. Nevertheless, there is a major problem using this data, which can be more easily explained in the CIE XYZ colour space.

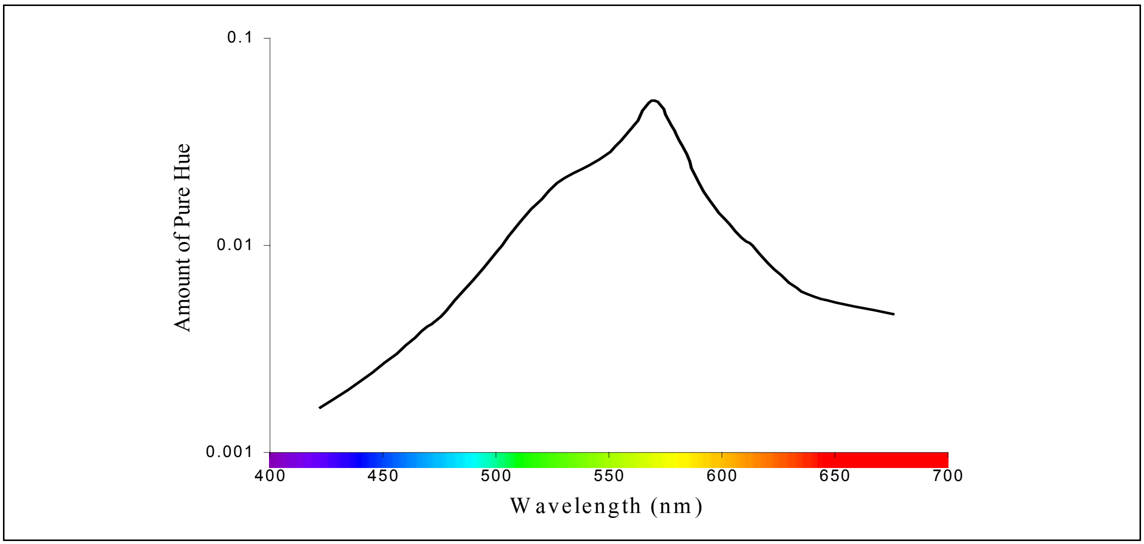


Figure 4-2 – Amount of pure hue added to neutral colour before the hue becomes detectable.

The existing biological data is given in terms of wavelength, which is the physical property corresponding to the notion of Hue. In order to find which wavelength is associated with a given Hue, a conversion to CIE XYZ must be obtained. Having the CIE XYZ values, as well as the coordinates of the White Point (the chromaticity coordinates of the reference white colour as suggested by the standard *BT.709* [84]), the computation of the wavelength is as follows. The XYZ values are normalized and a line is drawn from the White Point to the point defined by the normalized x, y, z values on the $x+y+z=1$ plane. The line is then extended until the spectrum locus (the horseshoe-shaped outline of the chromaticity diagram), and the wavelength is read by the point at the intersection of the line and the spectrum locus. For example, points A and B in Figure 4-3 intersect the spectrum locus at 504nm and 595nm respectively. Those wavelengths are called the dominant wavelengths for colours A and B. All the colours at a given straight line from the white point to a specific wavelength, have exactly the same dominant wavelength, but the closer they lay to the white point, the less saturated they are.

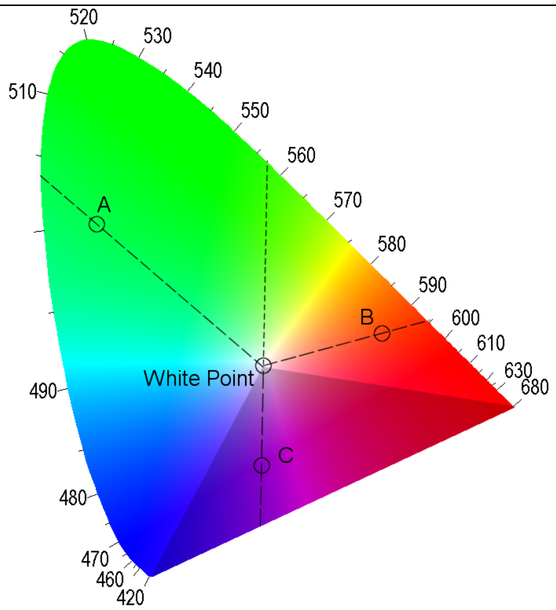


Figure 4-3 – Dominant wavelength specification with the CIE chromaticity diagram. The dominant wavelength can be read directly from the spectrum locus for spectral colours (like A and B), or specified as the complementary wavelength for non-spectral colours (colour C).

What now becomes evident is that not all Hues can be directly associated with a wavelength. As can be seen in Figure 4-3 if any point falls on the locus of “non-spectral” colours (the shadowed area in the figure), like point C, the intersection of the line connecting the white point and the point in question falls on the straight line at the bottom of the XYZ gamut, connecting blue and red, which is called the line of purples. For any point that falls in that region the line is instead extended at the opposite direction, and the dominant wavelength is considered to be the wavelength of the complementary colour. For the case of point C, the dominant wavelength is calculated as 557nm.

The physical explanation behind this is that every perceived colour is usually not monochromatic (a single wavelength), instead its spectrum comprises of a range of wavelengths. A certain wavelength may be dominant in that spectrum, in the sense that light of that wavelength has much more intensity than light of any other wavelength in the spectrum of the colour. In this case, the colour is perceived as of this wavelength, and depending of the amount of white background radiation it will appear more or less saturated. For a colour laying on the line of purples, the spectrum is somewhat different. Instead of having a background white radiation where light of a specific wavelength is dominant, it presents high white radiation from which light of a specific wavelength is missing. The colour perceived in this case is the complementary colour of the wavelength missing, for the case used before, the purple colour of point C is actually white radiation with wavelength of 557nm missing. This is illustrated in Figure 4-4.

It is now evident that the biological data given before is not complete, since it does not give any information about the Hues of the line of purples. In order to cover this range of Hues, certain experiments were conducted as part of this research, re-measuring the amount of pure Hue needed to be added to white before the Hue becomes detectable, this time including the Hues of the purple line. The results of the experiment correlate strongly with the data of Figure 4-2, enabling us to accept them as true indications. In addition, two more experiments were conducted measuring the amount of pure Hue needed to be added to black and to a mid-grey before the Hue becomes detectable. The results obtained along with details about the experimental set-up are given in Appendix A. To summarize, for each Hue three values were obtained: the amount of pure Hue added to White, the amount of pure Hue added to Black and the amount of pure Hue added to a Mid-grey before the Hue becomes detectable.

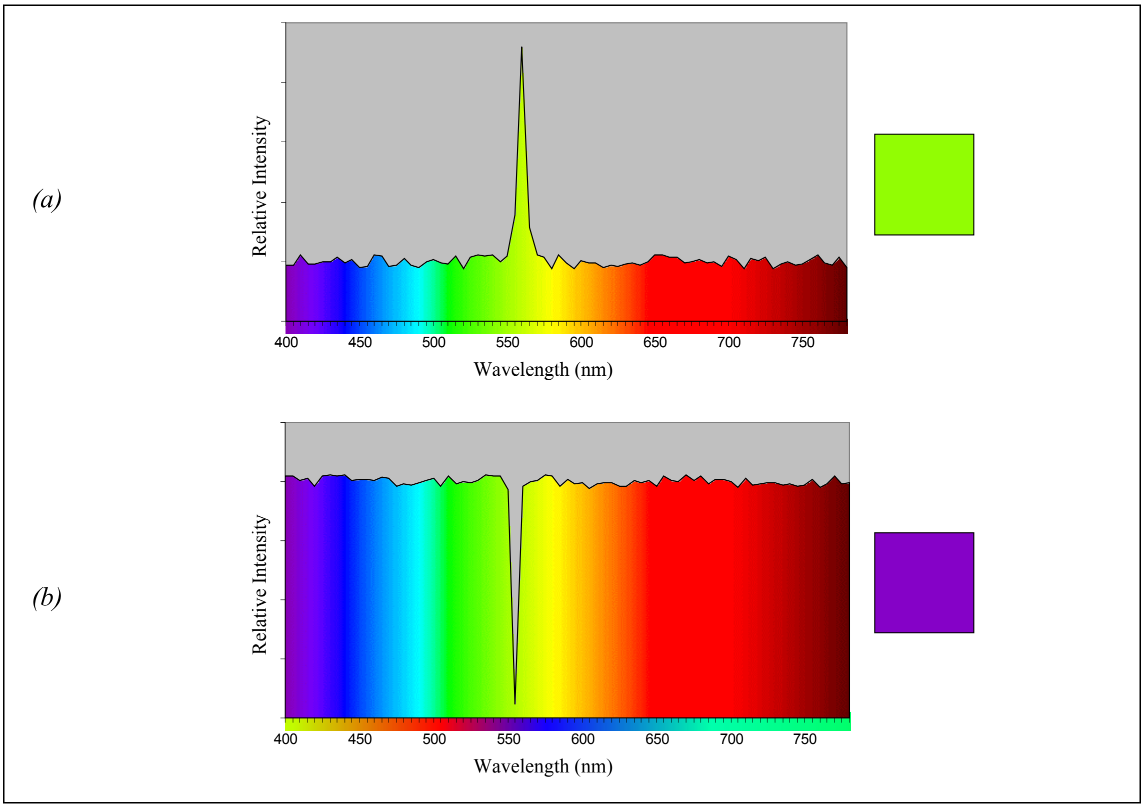


Figure 4-4 – Sample spectra of a green colour and its complementary purple one. The colour bar under spectrum (b) shows the complementary colours for each wavelength. These samples are for illustrative purposes only. A colour having a spectrum like (a) for example would not necessarily be perceived as pure green by a human.

The three sets of values (as explained above) enabled the accurate definition of the border surface between chromatic and achromatic pixels. The least detectable steps from black and white, give information about colours at the edges of the Lightness component, while the least detectable steps from mid-grey, give information about Saturation thresholds. For illustrative purposes, a vertical slice of the HLS colour space is presented in Figure 4-5. The line that separates chromatic from achromatic colours for each Hue consists of four segments, connecting white, black and three points indicated by experimental results. These three points are along the lines that connect white to pure colour (WC), mid-grey to pure colour (GC) and black to pure colour (BC).

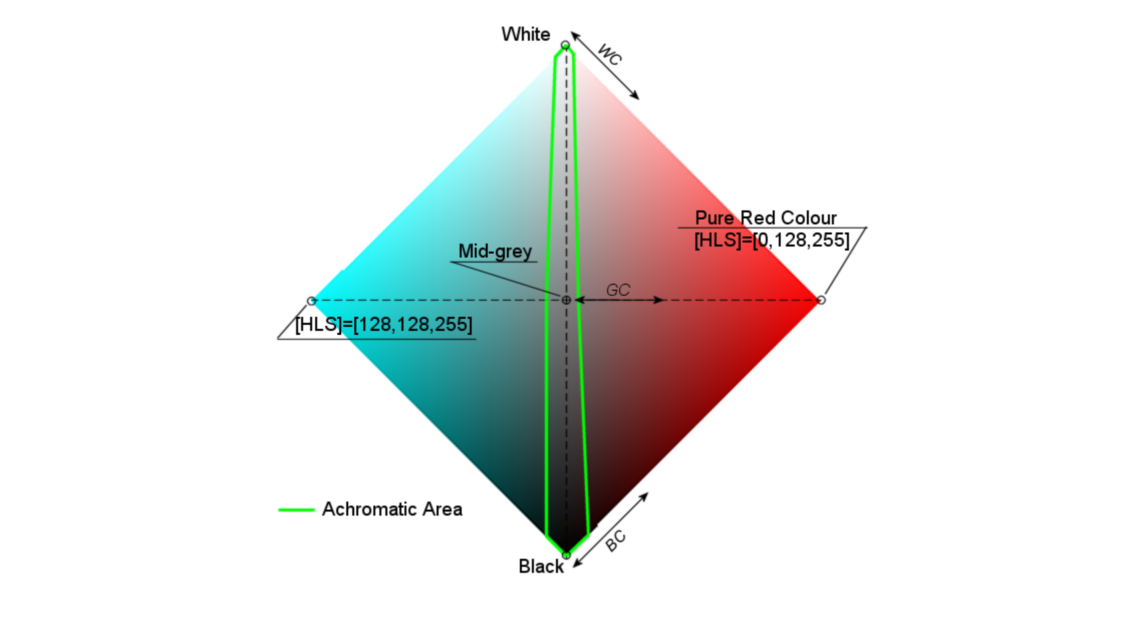


Figure 4-5 – A vertical slice of the HLS colour system. The three directions on which measurements were taken namely White to Pure Colour (WC), Mid-grey to Pure Colour (GC) and Black to Pure Colour (BC) are illustrated in the figure. The green area signifies the area of achromatic colours as defined by experimental data.

During pre-processing, this information is applied to the image, separating the achromatic pixels from the chromatic ones. The image is therefore split in two layers, one containing all the achromatic pixels (grey-levels) and the other all the chromatic ones (Figure 4-6). The achromatic layer is strictly kept out of any process involving Hue (it is processed based on the Lightness component only), whereas the chromatic one is processed based on both the Hue and the Lightness components.

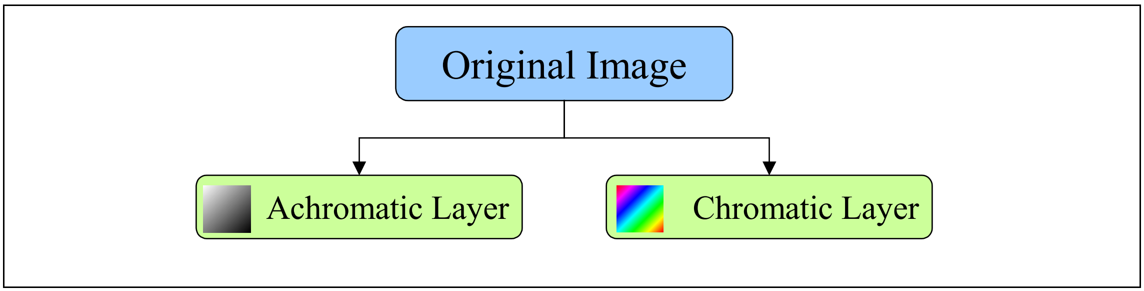


Figure 4-6 – After pre-processing, the Original Image is split in two layers: one containing the achromatic pixels and one containing the chromatic ones.

We consider using data derived by experiments in realistic situations, taking into account the effect of different Hues, superior than using a single saturation (and lightness) threshold (e.g., [28, 29, 82]) for separating chromatic and achromatic pixels. Although it is agreed that defining a fuzzy threshold based on the experimental data instead of a crisp one would be preferable, for the purposes of this application, the above process was considered adequate.

4.4. Splitting Phase

After the separation of chromatic from achromatic pixels, the splitting process begins. During this phase, histograms of each layer are analysed, and each layer is considered for further splitting into sub-layers. A detailed description of the splitting algorithm follows in the next section, while the histogram analysis process used to control splitting is described in Section 4.4.2.

4.4.1. Splitting Process

After the pre-processing step, the original image has been split in two layers as illustrated in Figure 4-6. The splitting process further splits those layers, according to global information obtained from the histograms of each layer.

Achromatic Layer

For the achromatic layer, the histogram of Lightness is computed. Since this layer contains only grey-scales, the Hue or Saturation histograms do not provide any useful information, thus they are not used. The histogram of Lightness is analysed as is described in the next section, and intervals in the histogram are identified in terms of histogram peaks. According to those intervals, the layer is split into sub-layers, each containing pixels in a range of Lightness values. The sub-layers produced cannot be split further, since the only information available for achromatic pixels is their Lightness values. Therefore, the splitting process for the achromatic layer stops at this point. The histogram analysis process (peak identification and combination of certain adjacent peaks) is detailed in Section 4.4.2.

Chromatic Layer

The layer containing chromatic pixels is treated slightly differently. There are three ways to split this layer: based on the Hue histogram, based on the Lightness histogram, or based on the Saturation histogram. Generally, Saturation is important in order to separate chromatic from achromatic pixels, but for pixels that carry some chromaticity information, the factors that play an important role in colour discrimination are mostly Hue and Lightness [181]. For this reason, the Saturation histogram is not used here; the splitting of the chromatic layer is based on the Hue and the Lightness histograms only. Splitting starts with one of the two, and continues in a recursive way, alternating between Hue and Lightness histograms in each iteration.

After experimentation, starting with the Hue histogram proved to produce more efficient splitting. Efficiency in this case translates both to the number of layers finally produced, which needs to be small, and to the actual content of the layers. Starting with the Hue histogram produces in most cases fewer and more meaningful layers. This conclusion agrees with previous research [82, 119, 203] stating that Hue has the greatest discrimination power among colour components. The appropriateness of Hue as the first splitting component can also be justified by simple observations on the images of the dataset. More often than not, humans differentiate objects according to their Hue, and then interpret differences in Lightness as shadows or highlights of the objects [119]. This claim, originally made for natural scenes, is also true to some extent for Web Images. Based on the images of the dataset, the gradient colours typically used are of almost constant Hue and variable Lightness. Therefore, splitting the image based on Hues first, has more chances to produce a split where the text is extracted in one layer, rather than scattered along different ones.

Of course, even if the text is of constant Hue, thus extracted in only one layer, parts of the background having the same Hue will also be present in that layer, necessitating the subsequent recursive splitting. If the text is not of constant Hue, it will be split along different layers, and the subsequent recursive splitting will ensure that it will finally (in subsequent layers) be separated from the background. The latter, is the worst-case scenario. Still, as long as text components exhibit higher similarity between them, than with parts of the background, the merging process that follows should be able to combine them properly.

To summarize, the chromatic layer produced by the pre-processing phase, is first split based on its Hue histogram. The Hue histogram is analysed (see Section 4.4.2) and intervals of Hue values are identified (in terms of histogram peaks, and combinations of adjacent peaks). For each interval identified, a sub-layer is produced, and the corresponding pixels are copied over. Those sub-layers (produced from the Hue histogram analysis) are further analysed based on their Lightness histograms, and split accordingly. The process continues in a recursive way, alternating between Hue and Lightness histograms in each iteration. A tree structure of layers is created by this process, an illustration of which can be seen in Figure 4-7.

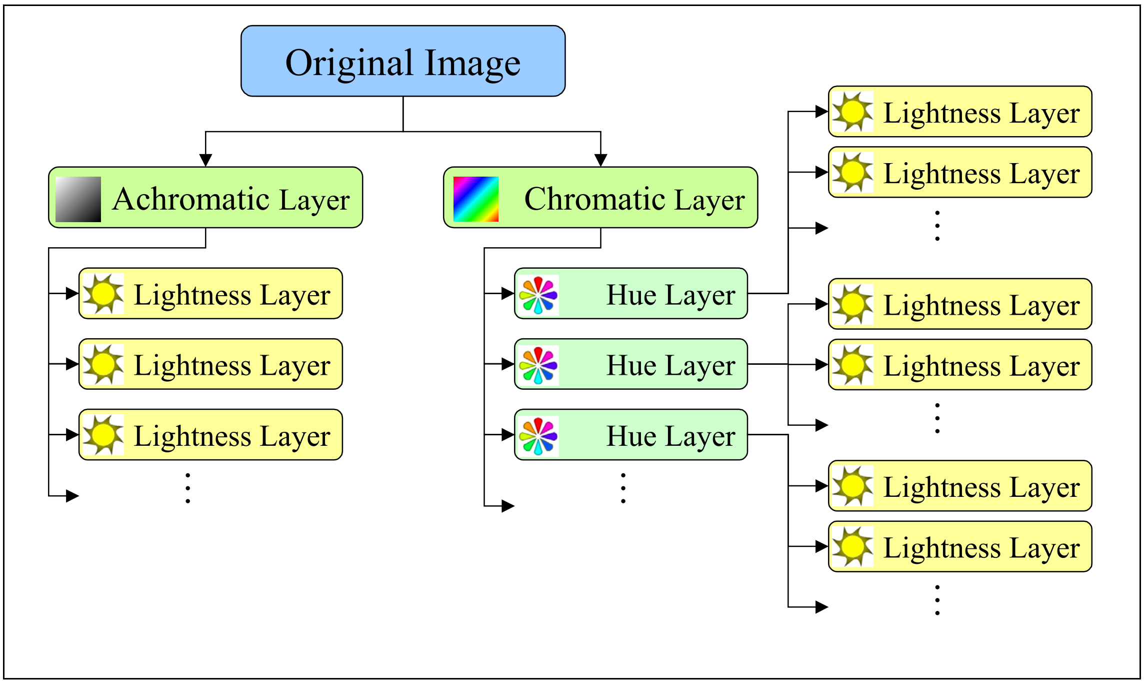


Figure 4-7 – The tree structure of layers produced by the splitting step of the Split and Merge Method. The original image is split in an achromatic and a chromatic layer. The achromatic one is split once based on its Lightness histogram. The chromatic layer is recursively split, starting with the Hue histogram and alternating between the Hue and Lightness histograms in each iteration.

The recursive splitting stops in one of the following cases: a maximum pre-defined number of splitting steps have been reached or only one peak can be found in the histogram. The maximum pre-defined number of splitting steps, was defined equal to *2*, so the initial chromatic layer is first split according to its Hue histogram, then each of the sub-layers produced is split once more according to its Lightness histogram and the process stops. Although allowing only two steps may sound at first very limiting, it was found after experimentation with different values, that if the number of steps increases, much unnecessary splitting occurs, in the sense that the two classes (text and background) have already been separated, and now components of each class start to become split across layers. Furthermore, the number of new sub-layers produced is very high and delays significantly the merging process that follows. It was experimentally found that two splitting steps were adequate to produce a good separation of the two classes, while this produces a reasonable number of layers.

The second criterion for stopping the splitting process is much more straightforward. If the histogram examined for a particular layer contains only one peak, therefore the histogram analysis returns only one possible interval for splitting, no further splitting occurs. If a split was to happen in this case, only one sub-layer would be produced from the layer examined, so effectively no further splitting is possible.

The algorithm (in pseudo code) of the above process, covering the pre-processing and the splitting step of the Split and Merge Method is given in Figure 4-8.

```
For Each Pixel in OriginalImage
{
  If PixelColour is chromatic
     Then copy Pixel to ChromaticLayer
  Else copy Pixel to AchromaticLayer
}
Split(AchromaticLayer, Lightness, 1)
Split(ChromaticLayer , Hue      , 2)

Split(Layer, ColourComponent, MaximumNumberOfSteps)
{
  If MaximumNumberOfSteps has been reached Then Exit
  Compute Histogram of ColourComponent for Layer
  Analyse Histogram
  If PeaksIdentified == 1 Then Exit
  For Each Peak identified in Histogram
  {
    Create SubLayer for the interval specified by Peak
    For Each Pixel in Layer
    {
      If ColourComponent value of PixelColour falls under Peak
         Then copy Pixel to SubLayer
    }
    If (ColourComponent == Lightness)
        Then Split(SubLayer, Hue, MaximumNumberOfSteps)
    Else If (ColourComponent == Hue)
        Then Split(SubLayer, Lightness, MaximumNumberOfSteps)
  }
}
```

Figure 4-8 – Pseudo code of the pre-processing and splitting steps of the Split and Merge Method. Commands and reserved words are typed in Bold.

4.4.2. Histogram Analysis

Histogram analysis aims at identifying interesting peaks in a histogram of the layer in question, which will be subsequently used to split the layer. Only the pixels that belong to that layer are taken into account when computing a histogram. Bearing in mind that a merging phase follows, it is preferable at this point to over-split the image, even if that entails breaking characters across different sub-layers, rather than to under-split the image. Nevertheless, if a peak in the histogram can be identified as containing only pixels belonging to text, that would be beneficial. The question that arises at this point is whether there are any "text peaks" (peaks containing only pixels belonging to text) in the histogram or not, and if yes, what are their distinguishing characteristics so that histogram analysis can positively identify them. Furthermore, the possibility of performing some kind of post-processing, such as histogram smoothing, in order to reduce the final number of peaks produced is investigated.

The general structure of a peak is illustrated in Figure 4-9. A peak is identified by a left minimum, a maximum and a right minimum. The *width* (*W*) of a peak is defined as the distance between the positions of the left and right minima. The peak width can be broken in two parts: the distance between the position of the single maximum of the peak and the position of the left minimum, denoted W_{left}, and the distance between the position of the single maximum of the peak and the position of the right minimum, denoted W_{right}. In a similar manner, two heights can be defined for a peak: the difference between the maximum and the left minimum of the peak, denoted H_{left}, and the difference between the maximum and the right minimum of the peak, denoted H_{right}.

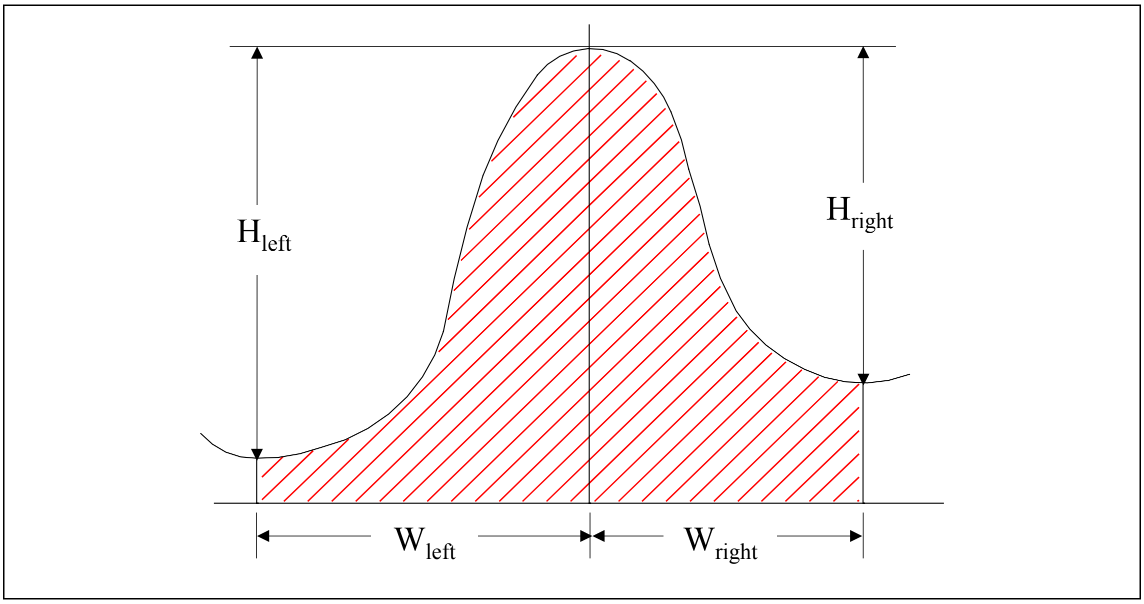


Figure 4-9 – The structure of a peak. Interesting features of the peak can be its size, its aspect ratio, the slope of ascend or descend, the width and height of the peak, etc.

Every histogram can then be decomposed to a number of such peaks. Special consideration is given to Hue histograms, due to the fact that the Hue component is 2π modulus. That is, since Hue is expressed as an angle, it presents a circular repetition where value 256 is mapped back to zero (or, value –1 is mapped to 255). For Hue histograms only, a peak is allowed to exist bridging the two ends (e.g., with a left minimum at 250 and a right minimum at 5).

Text Peak Features

Based on the general structure of a peak, certain features can be defined. Such features are discussed next in the context of identifying text peaks in a histogram.

The *size* of the peak can be defined as the number of pixels under the peak. In other words, that would be the integral of the peak structure. Size is a useful metric, if the number of pixels belonging to text is known beforehand. Although this is not generally the case, text is normally expected to comprise fewer pixels than the background. Nevertheless, because text can share some colours with the background, the size of the peak representing those colours will be significantly larger than expected (Figure 4-10). Small peaks located on top of larger ones are therefore lost. Finally, even the first assumption that text occupies a small portion of the image is not always true; sometimes the larger peak in the histogram actually corresponds to text. In conclusion, size cannot be used to sufficiently indicate the presence of a text peak.

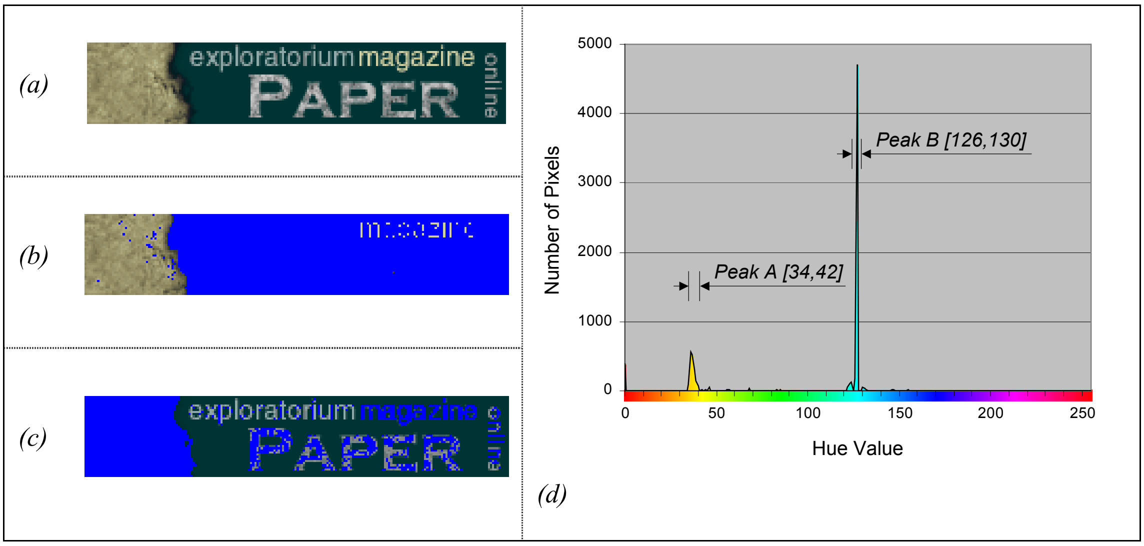


Figure 4-10 – (a) Original Image. (b) Pixels under Peak A. Peak A includes the yellow characters and part of the background. (c) Pixels under Peak B. Peak B includes the greenish characters and part of the background. (d) The Hue histogram for the image. The text and the background share colours.

The *left* or *right slope* of the peak is the rate of ascent or descent respectively of the peak. It is defined as $S_{left}=H_{left}/W_{left}$ and $S_{right}=H_{right}/W_{right}$. A high left and right slope indicates that the colours of the peak are well separated from the neighbouring colours. If the text has high contrast to the background, then the slopes of a peak representing the colours of text pixels will be relatively high. Sometimes, however, anti-aliasing is used and therefore the transition from background to text is quite smooth. Consequently, the peaks representing the colours of text will have a lower slope. Generally, no assumption can be made about the slope of text peaks, and therefore this feature cannot be used to indicate the presence of a text peak.

The *width* of a peak represents the range of colours captured by the peak. If the peak in question were a text peak, then a large width would indicate that a gradient of colours has been used for the text, while a narrow peak would indicate relatively monochrome text. However, no general assumption can be made about the width of a text peak. The *height* of the peak is representative of the number of pixels under the peak, but the size feature is more indicative of this property, and has already been discussed.

The *width ratio* and the *height ratio* [8] are defined in Eq. 4-1 below. They are metrics of the symmetry of the peak. The closer both of them are to one, the most symmetric the peak is, in the sense that it raises and falls in the same way.

$$W_{Ratio} = \begin{cases} \dfrac{W_{left}}{W_{right}} & if\ W_{left} < W_{right} \\ \dfrac{W_{right}}{W_{left}} & otherwise \end{cases} , \quad H_{Ratio} = \begin{cases} \dfrac{H_{left}}{H_{right}} & if\ H_{left} < H_{right} \\ \dfrac{H_{right}}{H_{left}} & otherwise \end{cases} \qquad Eq.\ 4\text{-}1$$

Different combinations of the above features were tried for peak selection, but none proved able to give a strong indication of a peak representing pixels of text. A scatter plot of the width and height ratios of peaks is given in Figure 4-11. Peaks from histograms of a number of pictures were examined and categorized as: peaks that correspond to pixels of the background, peaks that correspond to pixels of the text, and peaks that contain pixels of both classes. As can be seen in this example, it is difficult to define a clear decision boundary between the classes or a set of rules that separates the classes well. Different peak selection processes were tried, that involved restrains in the size, slope, width, width ratio and height ratio of peaks, but they all proved to discard some of the text peaks, while being rather time-consuming.

Complicating the process of peak selection is not beneficial for the whole segmentation process, especially since there would always be some uncertainty about the peaks finally selected. Therefore, the decision at this stage was made to avoid discarding any potentially useful information by selecting the wrong peaks, so no peak selection process is finally used. Nevertheless, creating a different layer for every single peak identified in the histogram is highly inefficient, so the possibility to combine peaks in order to reduce the final number of peaks produced was investigated.

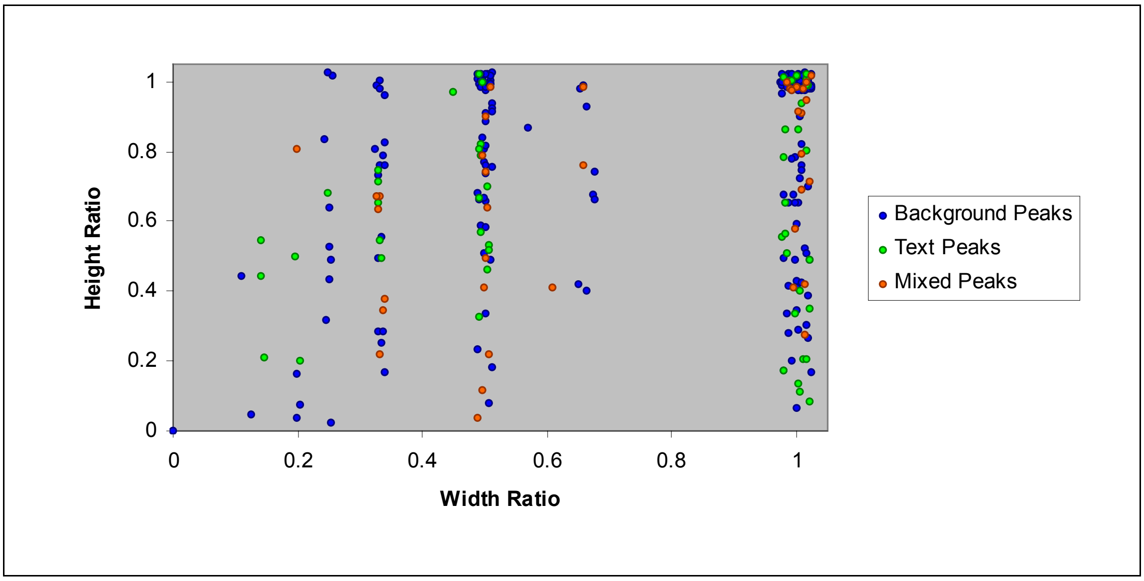


Figure 4-11 – Characteristic feature space for Width Ratio and Height Ratio of peaks. The peaks visible were collected from the Hue histograms of a number of images. (NOTE: The data in this scatter plot have been slightly jittered for easier viewing)

Reducing the Number of Exported Peaks

Towards reducing the final number of peaks produced, a number of smoothing and heuristic methods to combine peaks were tried. The criterion for deciding which method to use is the following. There are a number of mixed peaks (peaks that contain pixels of the text and of the background) in each histogram. If a peak that represents text pixels is accidentally merged during the smoothing process with a peak that contains background pixels, a mixed peak will also be produced, thus increasing the number of mixed peaks of the histogram. Therefore, a successful smoothing should decrease the overall number of distinct text and background peaks, while it should not increase the number of mixed peaks. It should be noted that special attention should be paid to the fact that if a mixed peak is merged with a background or a text one, then the number of mixed peaks will not increase, while a wrong merger will happen. For that reason, the results for each smoothing and heuristic technique examined were also visually inspected, before deciding on which technique to employ.

Weighted-averaging the histogram at different scales was initially examined. Weighted averaging reduces vastly the number of peaks in the histogram, merging a number of small peaks with bigger ones, which in most situations is not desired, since some small peaks often correspond to text. Therefore, smoothing by weighted averaging was not used.

A structural approach for smoothing noisy signals suggested by Antonacopoulos and Economou [8] was also implemented and tested. Data points are expressed in terms of *peak structures*, and peak structures are subsequently expressed in terms of *meta-peak structures*. Meta-peak structures consist of a left minimum, a maximum and a right minimum point, exactly like the initial peak structures, but the points for the meta-peaks are derived from the maxima of the initial peak structures. Certain features of peaks, such as the width and height are then analysed, and the meta-peaks are classified as either *noise* or *characteristic peaks* of the signal. Following that, the noise peaks are smoothed, while the characteristic ones are preserved. Originally devised for smoothing noisy signals, this method was slightly changed to fit the specific problem of colour histograms. The issue here is not to separate noise from characteristic peaks, but to decide which peaks can be safely combined without effectively merging text and background pixels. For this reason, absolute measures such as widths and heights of peaks were not used; instead, the width ratio and height ratio of peaks were employed. This approach produced far better results than smoothing by weighted averaging, however in a few cases, smoothing produced wrong mergers between peaks, therefore this method was not finally employed.

A different technique used was a heuristic method based on comparing the centres of gravity between peaks. A comparison between the centres of gravity of two successive peaks can give an indication of both their proximity and their relative size. The x-coordinate of the centre of gravity of a peak indicates the mean Hue or mean Lightness of the peak, depending on the type of histogram. The y-coordinate of the centre of gravity is indicative of the number of pixels in the peak. Two successive peaks are combined if both the horizontal distance and the vertical distance of their centres of gravity are below pre-defined thresholds. The rationale for this is as follows. A small horizontal distance indicates a small difference between the Hue or the Lightness of the peaks. If this is small enough, then the visual similarity between the pixels the peaks represent is adequately high for the pixels to be counted under the same peak. If the vertical distance is small, that means that the number of pixels of the successive peaks is almost the same, which in turn indicates that there is a gradient of colours in the image. After experimentation with different values, the horizontal threshold was set to 5 and the vertical threshold to $^1/_{50}$ of the maximum count in the histogram (an absolute value cannot be used for the vertical threshold, since the numbers of pixels in each image/layer differ vastly). This method produced better results than the previous ones; still there is room for improvement, so the described method was not finally selected.

The method ultimately used is based on the concept of checking horizontal and vertical distances of peaks, but improves over the previous technique in two factors. First, no crisp threshold is used for the horizontal distance, instead, information about the way humans discriminate between different Hue (or Lightness) values is employed to ensure that no combination of peaks is allowed to be formed that encompasses dissimilar Hues (or Lightness). Second, instead of checking the vertical distance between peaks against a threshold, the ratio of the maxima of the two peaks is checked, thus addressing the y-scaling problem more efficiently (rather than defining the threshold accordingly to the maximum count in the histogram). Furthermore, not using the centres of gravity of peaks relieves the method of some extra computations and slightly improves the processing time of this step.

In order to assess Hue similarity and Lightness similarity, available biological data [13, 211] about wavelength and lightness discrimination were used. Wavelength discrimination data were extended to cover Hues that are not directly associated with specific wavelengths as explained in Section 4.3. Least perceived differences in Lightness were also measured in order to verify the applicability of the theoretical model to realistic viewing situations. Those experiments are explained in detail in Appendix A. The Hue and Lightness discrimination data used in this method are presented in the next section, as they are an integral component of the merging process.

The grouping technique works as follows. For every pair of adjacent peaks, the range of Hue (or Lightness) values spanned by both peaks is examined. If the Hue (or Lightness) value at the left minimum of the left peak is similar to the Hue (or Lightness) value at the right minimum of the right peak, then the vertical distance of the peaks is tested. The ratio of the smallest maximum of the two peaks to the largest maximum is computed and if below a pre-defined threshold, the peaks are grouped together. The ratio ranges from zero to one. A value of *0.3* was selected after experimentation as an appropriate threshold, that is the smallest peak maximum of the two is not allowed to be less that *0.3* of the largest one for a grouping to take place.

To summarize, for each layer an appropriate histogram (either Hue or Lightness) is computed and expressed in terms of peaks. Every pair of subsequent peaks is considered for grouping, based on Hue or Lightness discrimination data (depending on the histogram type), and on the ratio of the maxima of the two peaks. The final peaks produced, define the intervals used by the splitting process described before.

4.5. Merging Phase

The merging process that follows the splitting phase, works the opposite way, identifying connected components in the leaf-layers of the tree structure produced, and aggregating the components following a bottom-up scheme. This process, aims to produce components that represent the characters in the image by merging components that belong to the same character according to colour similarity. Similarity is assessed based on experimentally derived discrimination data for Hue, Lightness and Saturation. The use of such experimental data ensures that the components produced after the aggregation process will consist of pixels having visually similar (as humans perceive them) colours. The experiments were conducted in order to verify and extend existing wavelength, luminance and colour purity discrimination data. The discrimination data finally used are detailed in this section, while the experiments conducted are described in Appendix A.

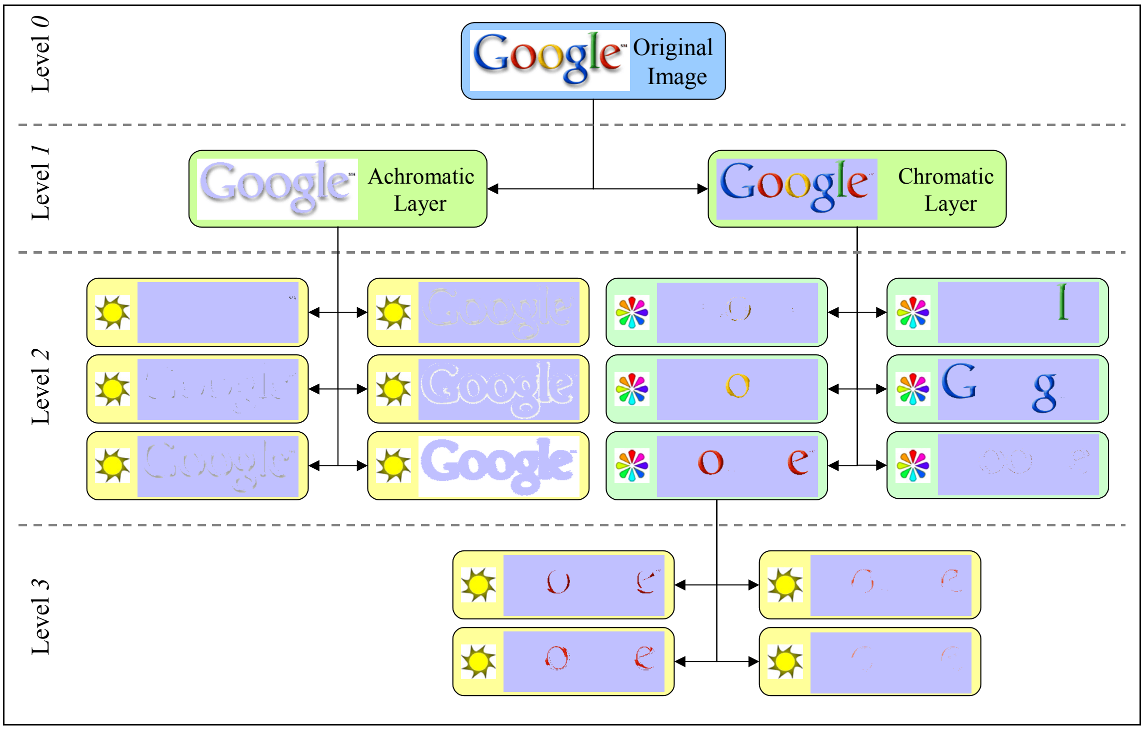


Figure 4-12 – A Web Image after the splitting phase (not all layers produced are visible in the figure). Pixels that do not belong to a layer are presented as a light-coloured background.

The different steps of the merging process will be discussed in detail next, accompanied by an example of the application of the process on a Web Image. The Web Image, which is used as an example for this section and its corresponding splitting, is shown in Figure 4-12.

4.5.1. Connected Component Identification

Each leaf-layer of the tree structure produced by the splitting process contains a subset of the pixels of the original image, as filtered through the recursive splitting. Effectively, one can associate *1* to pixels belonging to a given layer, and *0* to the rest, constructing a bi-level representation of the layer. A one-pass connected component identification algorithm [5] is applied to this bi-level representation and a number of connected components are thus detected in each of the leaf-layers. For each of the connected components detected, the average colour is calculated from the colours of the pixels belonging to it.

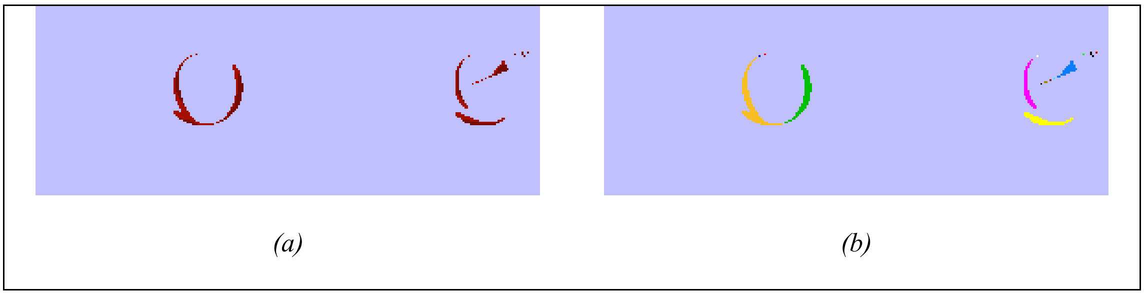


Figure 4-13 – One of the leaf-layers as illustrated in Figure 4-12 and the associated connected components identified (indicated in different colours).

Figure 4-13 illustrates one of the leaf-layers produced for the sample image of Figure 4-12, and the connected components identified in the layer. Each connected component is shown in a different colour for illustrative purposes.

4.5.2. Vexed Areas Identification

The neighbouring pixels (in the original image) of each connected component identified are subsequently examined, and if their colour is similar to the average colour of the component, they are flagged as a potential extension for the component. This potential extension will be referred to as the *vexed area* of the component for the rest of this thesis. The neighbouring pixels of each component do not necessarily belong to the layer being examined, and the method does not require them to. Instead, the colour of any pixel checked is retrieved from the original image.

Similarity is assessed according to the type of the layer, meaning that if the layer is a Lightness layer (produced by splitting based on the Lightness histogram), similarity is assessed by examining Lightness differences between the components, whereas if a Hue Layer, similarity is assessed by examining Hue differences. At this point, Hue and Lightness discrimination data are employed in order to decide whether two Hues or Lightness values are perceived as similar or not.

Hue Discrimination

The ability of humans to discriminate between two colours of different wavelength, depends on the wavelengths in question. For example, assume that two colours have a wavelength difference δ. Humans find it more difficult to differentiate between two colours if they both lie in the green band than if the two colours lie in the yellow band (with the distance remaining δ in both cases). This is because humans are more sensitive to the yellow wavelengths than they are to green ones. More specifically, a difference of only *2* nanometres would be adequate to discriminate between two colours in the yellow band, while a difference of at least *4* (relatively larger in practice) nanometres would be required for colours in the green band. Such biological information about wavelength discrimination is available, but as mentioned before it is incomplete, in the sense that information does not exist for all possible Hues. The experiments conducted in order to extend this data to Hues that are not directly associated with specific wavelengths are presented in Appendix A.

The thresholds finally used for Hue discrimination are illustrated in Figure 4-14. For each Hue in the range from *0* to *255*, a minimum and a maximum Hue value is defined. Every Hue in this range is considered similar to the Hue in question. The data shown above are derived from the strict Hue discrimination data presented in Appendix A. The thresholds have been relaxed by a factor of *2* compared to the Hue discrimination thresholds measured. This is because the measurements that were conducted aimed to identify the least noticeable difference required for two Hues to be perceived as dissimilar, while for the merging process, we need to allow for slightly greater tolerances, in order to be able to cover situations of Hue gradients or anti-aliasing. The factor of *2* was experimentally determined, so that reasonable results are obtained for a wide range of Web Images.

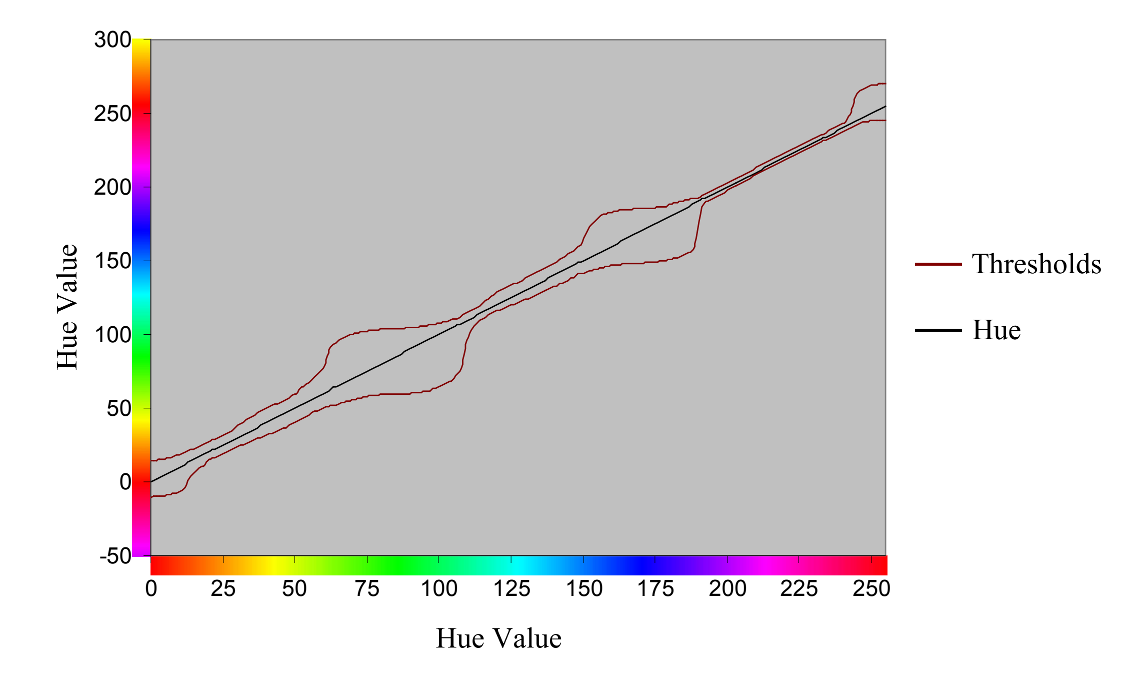


Figure 4-14 – The Hue discrimination thresholds used. For each Hue value (denoted with the black line), there is a minimum and a maximum Hue value defined (denoted with the red lines).

Lightness Discrimination

Similarly to Hue discrimination, humans have different Lightness discrimination abilities for dark and light colours. Lightness perception is roughly logarithmic. Humans cannot differentiate between two colours if the ratio of their intensities is less than approximately one percent. In the context of computer graphics, CRT monitors are inherently non-linear, that is the intensity of light reproduced on the screen of a CRT monitor is a non-linear function of its voltage input, which in turn is set by the *RGB* values of the pixels. Those *RGB* values can be corrected to compensate for this non-linearity, a process known as *gamma correction*.

An experiment was performed to measure Lightness discrimination for all *255* levels of Lightness as defined in the *HLS* colour system. Details of the experiment are given in Appendix A. Based on the results obtained, the Lightness thresholds are defined as shown in Figure 4-15.

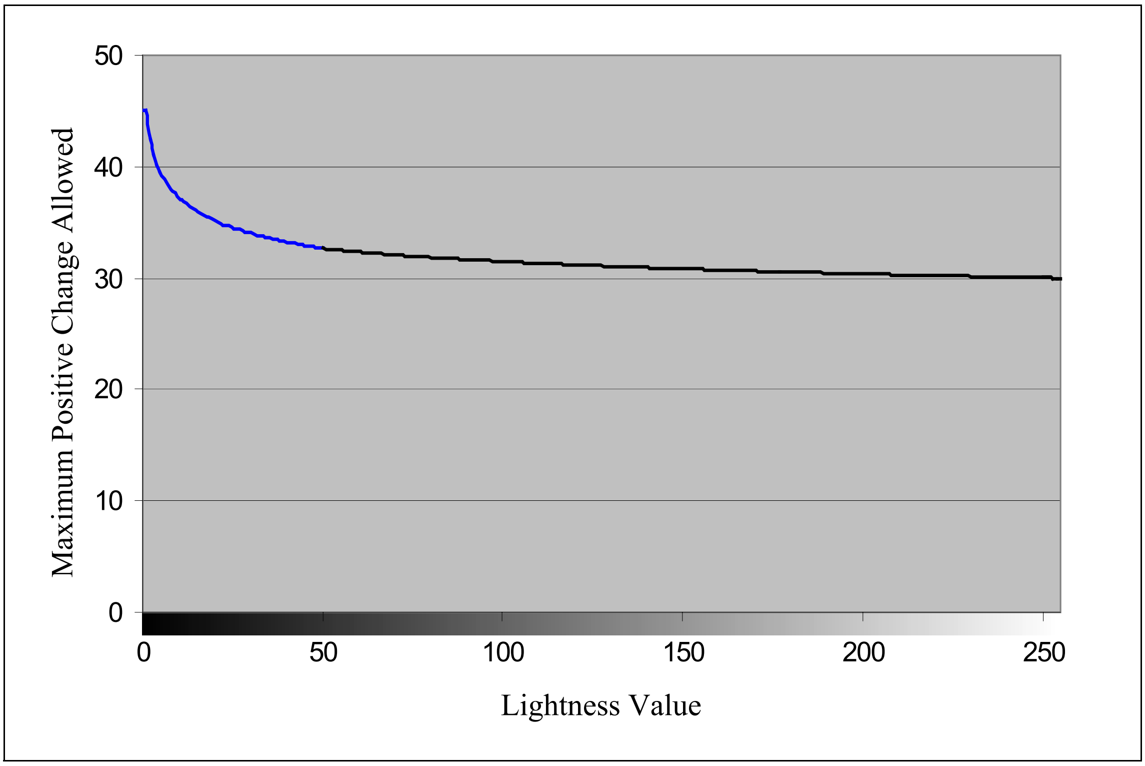
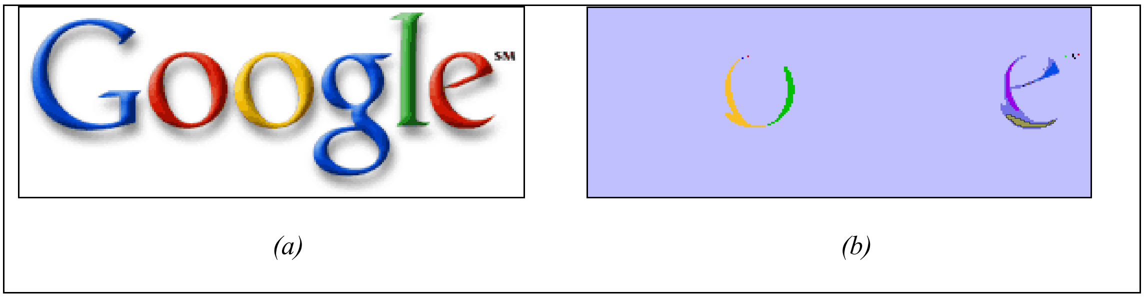


Figure 4-15 – The Lightness discrimination thresholds used. For each Lightness value, the maximum positive change for which two colours are considered similar is shown.

For each Lightness value, the minimum increase that can be made without producing any noticeable change between the two colours is shown. The thresholds used, are also relaxed compared to the measurements made, similarly to the Hue discrimination thresholds. Since the discrimination ability ranges vastly between dark and light colours, for the Lightness component, the measured values were not simply multiplied by a factor. Instead, the power function form of the distribution was preserved, but the gamma was changed to keep the thresholds strict (comparably to the measured ones) for dark colours, and much more tolerant for lighter ones. The first fifty values are approximated by a different power function in order to better match the measured ones.

As an example, the vexed area of one of the components of the layer in Figure 4-13 is shown in Figure 4-16. The neighbouring pixels are tested in respect to their values in the original image, which is also shown in the figure. As can be seen, the vexed area encompasses all pixels that have similar Lightness values (since the layer shown is a Lightness layer) that are in the 8-connected extended neighbourhood of the component. The vexed area as can be seen, overlaps with a number of other components. This fact is used next to facilitate the decision to merge two components or not.



(a) *(b)*

Figure 4-16 – (a) Original image. (b) A Lightness layer of the image with the connected components identified. The vexed area for the outlined component (component representing part of "e" character) is illustrated in blue colour.

4.5.3. Merging Process

After connected components and their associated vexed area have been identified in all leaf layers, the merging process takes place, starting from the leaf layers. The decision whether to merge or not two components is based on the amount of overlapping between the components. The idea is illustrated in Figure 4-17. The way "overlapping" is defined will be discussed next.

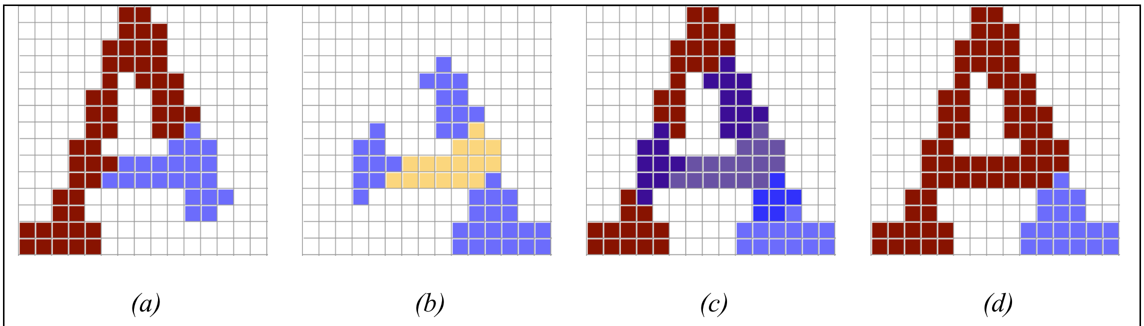


Figure 4-17 – (a) First component (red) and its vexed area (blue). (b) Second component (yellow) and its vexed area. (c) The two components overlapping. (d) The merged component and the new vexed area.

Overlapping

The most obvious, perhaps, measure one can evaluate given two components, is the number of pixels of the vexed area of one of the components that overlap with pixels belonging to the other component. Given two components *a* and *b*, and their associated vexed areas a_v and b_v, this number of pixels is denoted $NOP(a_v, b)$ and $NOP(a, b_v)$, respectively. $NOP(a_v, b)$ is the number of common pixels of the vexed area of component *a* with component *b* (effectively $NOP(a_v, b)=|a_v \cap b|$). $NOP(a, b_v)$ is the number of common pixels of the vexed area of component *b* with component *a*. The number of overlapping pixels between components *a* and *b* is then defined as the sum $NOP(a_v, b) + NOP(a, b_v)$.

The maximum possible number of overlapping pixels between two components *a* and *b*, cannot be greater than the sum of the sizes of the components. *Overlapping*, denoted *Ovl(a, b)*, can be consequently defined by Eq. 4-2.

$$Ovl(a,b) = \frac{NOP(a_v,b) + NOP(a,b_v)}{Size(a) + Size(b)} \qquad \textit{Eq. 4-2}$$

Overlapping is therefore defined in the range [*0,1*] and values near *1* would indicate some strong overlapping. Although this is a good definition, there are special cases when there would be in our interest to merge two components, where Eq. 4-2 produces a rather small value. An example of such a case is illustrated in Figure 4-18.

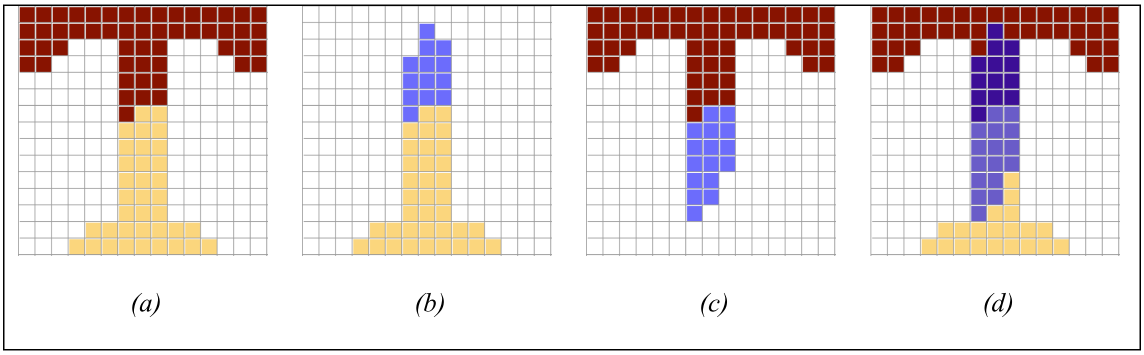


Figure 4-18 – (a) A character broken in two components. (b) Bottom component and vexed area. (c) Top component and vexed area. (d) Overlapping of components.

Suppose a character rendered in a gradient. During the splitting process, the character might be broken in different components, for example the top of the character might be one component and the bottom of the character another one. The vexed area of the top component would then cover some portion of the bottom one, and vice-versa. If, furthermore, the background is sufficiently different to the colours of the two components (that is a very simple situation for which the method has to work well), the whole of the vexed area of the top component is overlapping with the bottom one, and vice-versa.

The fact that the whole of the vexed area of one component overlaps with an other component, should give a strong indication that the components should be merged, yet, Eq. 4-2 fails to provide a value close to *1*. The reason for this is that the maximum possible number of overlapping pixels is defined to be equal to the sum of the sizes of the two components. For this case, the maximum possible number of overlapping pixels should be the sum of the sizes of the two vexed areas. This is because the vexed areas are much smaller than the components that participate in the comparison. For the example of Figure 4-18, if b is the bottom component and t the top one, then $NOP(b_v, t)=13$, $NOP(b, t_v)=16$, $Size(b)=36$, $Size(t)=53$, $Size(b_v)=13$ and $Size(t_v)=16$. To cover this case, Eq. 4-2 could be changed accordingly, replacing the denominator to the sum of the sizes of the vexed areas of the components (Eq. 4-3). Since the maximum possible number of overlapping pixels cannot be greater than this sum, overlapping will still be in the range of $[0,1]$.

$$Ovl(a,b) = \frac{NOP(a_v, b) + NOP(a, b_v)}{Size(a_v) + Size(b_v)} \qquad \textit{Eq. 4-3}$$

In real conditions though, this definition, which is based on the vexed areas rather than on the components, fails to identify some of the more obvious situations. An example is given in Figure 4-19, where the vexed areas are way too big compared to the components involved, resulting to a very small overlapping value, despite the fact that the vexed area of each component completely covers the other one.

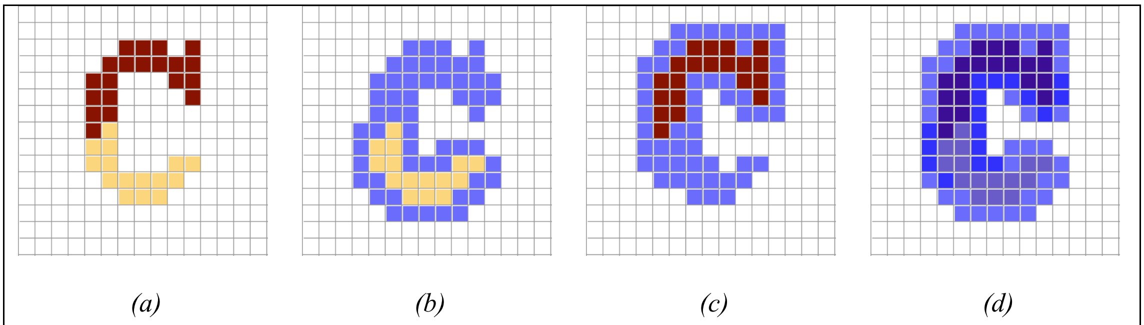


Figure 4-19 – (a) A character broken in two components. (b) Bottom component and vexed area. (c) Top component and vexed area. (d) Overlapping of components.

The remedy is to change the denominator once more, to the actual maximum possible number of overlapping pixels, which will be dependant on the given components, and the given vexed areas each time. The final definition for the overlapping, which is the one used in this method, is given in Eq. 4-4.

$$Ovl(a,b) = \frac{NOP(a_v, b) + NOP(a, b_v)}{\min\left(Size(a_v), Size(b)\right) + \min\left(Size(a), Size(b_v)\right)} \qquad \textit{Eq. 4-4}$$

It should be mentioned here, that in both Eq. 4-4 and Eq. 4-3 the denominator could be equal to zero. This happens when both components *a* and *b* have a null vexed area. In this case, the numerator will also be zero, and the overlapping is defined to be zero as well, since no pixels overlap at all.

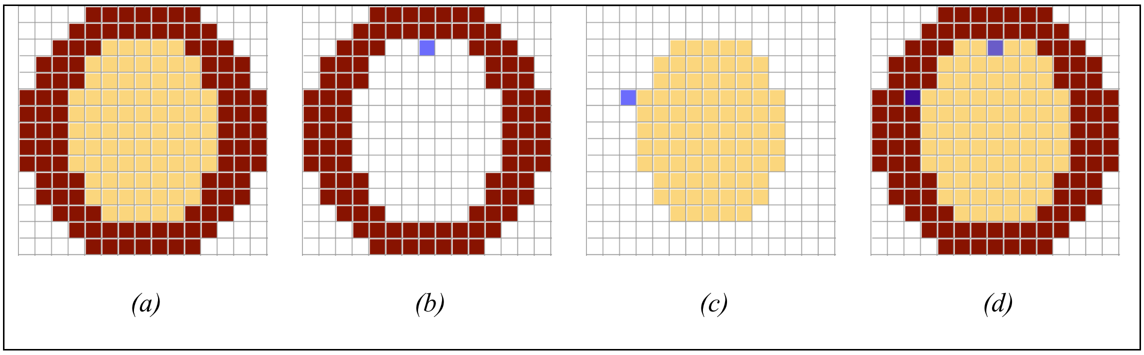


Figure 4-20 - (a) A character broken in two components. (b) Bottom component and vexed area. (c) Top component and vexed area. (d) Overlapping of components.

Although the above definition is more complete than the previous ones, there are still some special cases that should be dealt with. Such a special case is illustrated in Figure 4-20. For the situation presented in this figure, Eq. 4-4 gives an overlapping value equal to *1*. The question here is whether we feel confident to base the merging of two components on the overlapping of one or just a few pixels. This depends on the size of the components involved, if the sizes of the components involved are comparable to the number of pixels overlapping, then it is probably a good call, otherwise its probably not. This leads to the definition of a weighting function, which should reflect exactly this confidence. Such a weighting function is given in Eq. 4-5.

$$W(a,b)=\frac{NOP(a_v,b)+NOP(a,b_v)}{Size(a)+Size(b)} \qquad \textit{Eq. 4-5}$$

The above weighting function is quite comprehensive, and on a closer look, it is the same as the first definition of overlapping (Eq. 4-2) and ranges in [*0,1*]. Nevertheless, it also presents some special cases, as can be seen in Figure 4-21. Here the small component should probably be merged with the large one, and the overlapping value as computed by Eq. 4-4 is certainly large enough (equal to *1*) to indicate that, but the weight computed by Eq. 4-5 is small, due to the big size of one of the components.

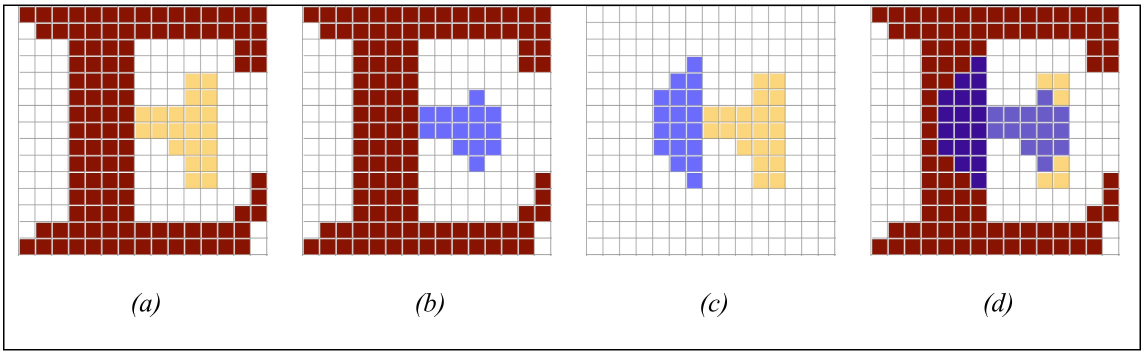


Figure 4-21 - (a) A character broken in two components. (b) Bottom component and vexed area. (c) Top component and vexed area. (d) Overlapping of components.

It proves better to base the weighting function on the smaller of the two components only, so the final weighting function is given in Eq. 4-6. The weight is no longer in the range [*0,1*], so by definition any value greater than *1* is bound to *1*.

$$W(a,b) = \frac{NOP(a_v,b) + NOP(a,b_v)}{2 \cdot \min\bigl(Size(a), Size(b)\bigr)} \qquad \text{Eq. 4-6}$$

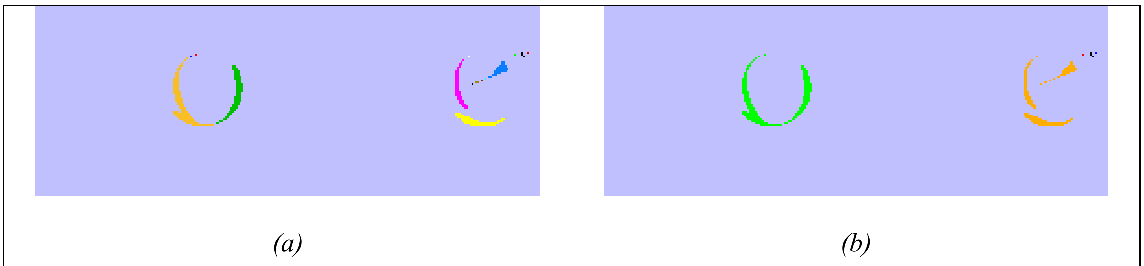
For each pair of components a and b the value $W(a,b) \cdot Ovl(a,b)$ is computed, and if above a pre-defined threshold the components are considered for merging. The value of the threshold used in the method was set equal to *0.5625* ($=0.75^2$). From now on, the term overlapping refers to weighted overlapping.

Merging in the Leaf-Layers

Merging is first performed in all the leaf layers. Subsequently, components in layers having a common parent layer (as shown in Figure 4-12) are merged by copying the components one level up (to the common parent-layer) and performing merging in the parent-layer. The merging process is the same for both the leaf and the intermediate layers, and is based on the overlapping between components as defined previously. The merging process in the leaf-layers is described next.

Every possible combination of components is checked first, and if their overlapping value is above the predefined threshold, a possible merger is identified. All identified mergers are kept in a sorted list, and merging starts with the merger having the bigger overlapping value. After each merger, the list is updated. Other mergers in the list involving either of the two components just merged are changed to refer to the new one, and the overlapping values are recomputed. In addition, possible new mergers for the newly created component are searched for, and if found added to the sorted list. The merging process stops, when all possible mergers have been made, that is, there are no mergers in the layer with an overlapping value greater than the threshold set. The new component is the union of the two initial components, and the vexed area for it, is the union of the vexed areas of the initial components. This can be seen in Figure 4-17.

It should be noted at this point, that all possible combinations of components are checked, not only pairs of components whose borders are touching. The reason for that is that due to extensive splitting, characters often split to many disjoined segments, and it is advantageous to merge these segments at this stage, as long as there is a link between them (ensured by an overlapping value greater than zero). Such an example can be seen in Figure 4-22 that follows. The initially identified components are shown on the left, and the components resulted after merging are shown on the right. As can be seen, the components that represent parts of the "*o*" character have been merged together, since their vexed areas overlap adequately. The same happens to components representing parts of the "*e*" character. As will be seen next, the final components are required to be connected, that is a situation like the one described above where a component comprises of many disjoined segments is not allowed. Nevertheless, it proved beneficial to the overall process to allow this for one step at the time, meaning that it is allowed while merging components of the leaf layers, or layers of the same level, but any disjoined components are broken before they get copied one level up in the tree structure. More about this will be given later on, in the section about component integrity.


(a)

(b)

Figure 4-22 – (a) Components identified before merging. (b) Components resulting after the merging process. Notice that some non-touching components have been merged.

Moving Up in the Tree-Structure

After all possible mergers have taken place in the leaf layers, merging between the components of layers of the same level is performed. This happens by copying the components of the leaf layers one level up, and repeating the merging process in the layer that receives the components. For example, based on Figure 4-12, after merging has finished in all *level-3* leaf layers, all the components are copied one level up, following the tree structure. In this example, the resulted components in the Lightness layers are copied in their parent *level-2* Hue layer. Consequently, after this copying, all components in the Hue layer will have similar Hues (since they were identified in children layers of this Hue layer), and will have vexed areas defined based on the Lightness thresholds (since vexed areas were identified in the Lightness layers). By performing a merger in the Hue layer at this point, effectively, we merge components of all the *level-3* Lightness layers, based on Lightness defined vexed areas.

Two components will only overlap at this stage, if their Lightness values are sufficiently similar (according to the Lightness thresholds used). The rationale behind merging at this point is to address characters of constant Hue, comprising of areas of slightly different Lightness. Examples are characters in gradient colour (in the Lightness component), or characters with shadows or highlights. These characters will have been broken across different Lightness layers, but if their consisting components are similar enough in terms of Lightness, their vexed areas will adequately overlap at this point.

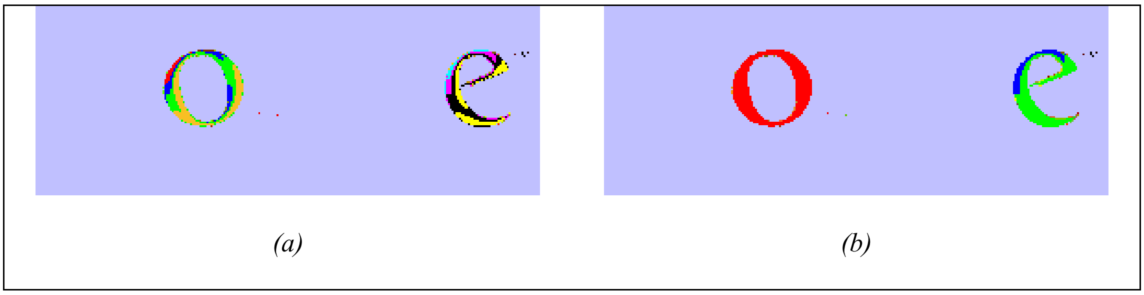

(a) *(b)*

Figure 4-23 – (a) Hue Layer with components of all its children layers copied over. (b) Components resulted after merging.

An example of the above process can be seen in Figure 4-23. All components of the children layers copied over in the shown Hue layer are illustrated in the image on the left, while the components resulted after merging are shown in the image on the right.

After all possible mergers occur in this layer, two additional processes take place: the refinement of the vexed areas of the resulting components and the examination of the integrity of the components resulting from mergers. These two processes will be explained next.

Vexed Area Refinement

The vexed areas of the components were identified in the leaf layers, according to the type of the leaf layers. After being copied one level up, and after merging has been performed, the vexed areas of the components remaining need to be refined, so that they are representative of the new layer in which they now reside. For example, after copying all the components identified in the Lightness type leaf layers to the parent Hue layer, the vexed areas must be refined so that they contain pixels not only of similar Lightness to the component, but of similar Hue as well. This is important, as merging between Hue layers of the same level will be performed next, and this merging must be based on Hue similarities.

The vexed areas are refined according to the type of layer they reside in, that is they might be refined based on Lightness similarity (for Lightness layers) Hue similarity (for Hue layers) or combined Lightness, Hue and Saturation similarity (for the Chromatic and Achromatic layers). When the process reaches the original image (the root of the tree), no refinement is necessary, since no other merging can happen.

The Hue and the Lightness similarity thresholds were defined earlier in Section 4.5.2. For the Hue and Lightness layers the process of vexed area refinement is as follows. For each component in a given layer, each pixel of the vexed area of the component is compared to the average colour of the component, and if not similar, it is removed from the vexed area. Similarity is based on the type of the layer as mentioned above. The pseudo-code of that process is shown in Figure 4-24.

```
RefineVexedAreas (Layer)
{
  For Each Component in the Layer
  {
    For Each Pixel in the VexedArea of the Component
    {
      If ( Colour(Pixel) is NOT similar to Colour(Component) )
        Then Remove Pixel from VexedArea
    }
  }
}
```

Figure 4-24 – Pseudo-code for the Refinement of Vexed Areas function of the Split and Merge Method.

The process is slightly different when it comes to the chromatic or the achromatic layer. Refining the vexed areas at this point, aims to prepare the components of the two layers (the chromatic and the achromatic) to participate in a merging process across this tree level, effectively checking the overlapping between achromatic and chromatic components. Consequently, the vexed areas at this point should represent some potential extension for each component based on the Saturation value of the component; for example, a low saturated chromatic component could potentially be merged with an achromatic one. For the chromatic layer, refining the existing vexed areas based on some kind of saturation similarity is of no real benefit. This is because the vexed areas of the components of the chromatic layer do not contain any pixels outside the range of the hues of the children hue-layers, whereas on the other hand achromatic pixels have undefined hue (conventionally set to zero). That effectively means, that if for example a very low saturated green component exists, even if the colour of the component is very low saturated indicating that it could be merged with some neighbouring grey component, its vexed area will not contain anything but green hued pixels, since it was refined as such in the children Hue layers. For this reason, instead of simply comparing the saturation of each pixel of the existing vexed areas to the saturation value of the component to which the vexed area belongs, we discard the existing vexed areas, and construct new ones, based on mixed Lightness Hue and Saturation similarity as will be described next. By doing this, we ensure that the vexed areas of the components of the chromatic layer can possibly contain some achromatic pixels, and vice versa: the vexed areas of components of the achromatic layer, can possibly contain some chromatic pixels. If this was not the case, it would make no sense to check for overlapping between components of the chromatic and the achromatic layer.

The process of finding the vexed areas for components in the chromatic and the achromatic layers is similar to the process of finding vexed areas in the leaf layers. The neighbouring pixels of each component are checked and if similar to the colour of the component they are added to the vexed area. The only difference here, is the way of examining visual similarity, between the colour of a pixel and the colour of the component. Three types of similarity tests are used: Hue, Lightness and Saturation similarity. The thresholds used for Hue and Lightness similarity are the same as defined before, while the process used for checking Saturation similarity is described below and is based on the Saturation thresholds detailed in Section 4.3 for separating chromatic from achromatic colours. The way the three types of similarity thresholds are combined is as follows. If both colours involved in the comparison are chromatic, then they are considered similar only if they are of similar Hue, Lightness (as defined by the thresholds given in Section 4.5.2) and Saturation (as is defined next). If both colours are achromatic (thus they are both low-saturated, and Hue is therefore unimportant), then only Lightness similarity is examined. Finally if one of the colours is chromatic and the other one is achromatic, then the colours are considered similar if the are of both similar Lightness and similar Saturation.

The Saturation thresholds used are based on the same thresholds used for separating chromatic from achromatic colours. The process to decide when two colours are similar in terms of saturation is as follows. If both colours are achromatic, they are considered similar (in terms of saturation), since they both have very small saturation. If both colours are chromatic, they are considered similar (in terms of saturation) if the ratio of their saturation values is less than *2*. That is, if S_1 and S_2 are the saturation values of the two colours, then the colours are similar if $\max(S_1, S_2)/\min(S_1, S_2) < 2$. If one of the colours is chromatic, and the other achromatic, then they are considered similar (in terms of Saturation) if the chromatic one is "very close" to the threshold between chromatic and achromatic colours defined for the Hue of the chromatic colour. "Very close" in this case was defined as double the threshold between chromatic and achromatic colours. Therefore, a chromatic colour with a saturation value less than double the threshold below which it is considered achromatic would be similar (in terms of Saturation) to an achromatic colour.

Component Integrity

The next operation performed is the examination of the integrity of each component. This operation aims at breaking components comprising many disjoined segments into disjoined components. There are two distinct types of such components.

The first situation is illustrated in Figure 4-25, where a component comprises two disjoined segments, connected by the vexed area of the component. This can happen as during the merging process every pair of components is considered, as long as there is a link between them (ensured by an overlapping value greater than zero). As mentioned before, this is allowed to happen, but only for one step at the time. Mergers are allowed to happen between disjoined components when merging in the leaf layers, or when merging between components of layers of the same level. Nevertheless, before copying one level up, if components still comprise of a number of disjoined segments, they are broken again, since the initial merging did not produce a connected component.

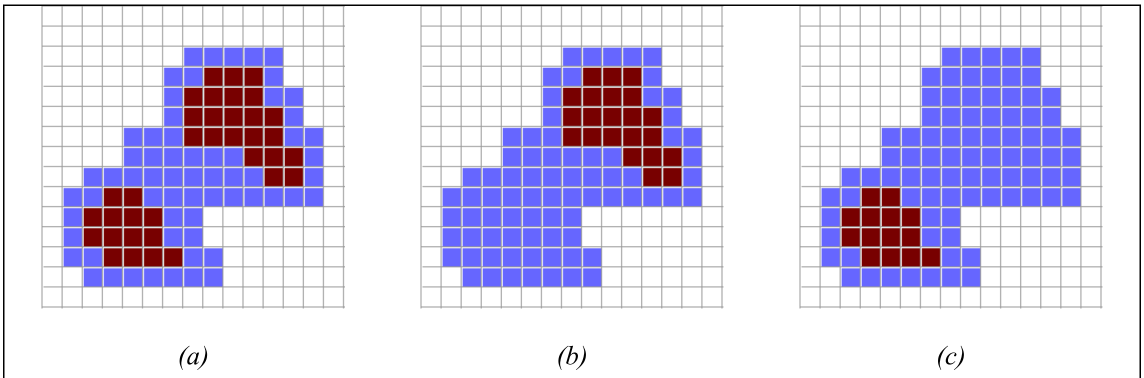


Figure 4-25 (a) Component comprising two disjoined segments (red) and its vexed area. (b) First component after breaking and its vexed area. (c) Second component after breaking and its vexed area.

One can say that it is meaningless to break the components in this situation, since the vexed areas are still overlapping, and the components will be merged again in the next level. Although this might be true, by moving up in the tree structure, more components are introduced in each step, and there is the possibility that some other merger (between one of the broken components and a new component) will present greater overlapping and should happen first, which might change the whole series of mergers.

The second type of component is illustrated in Figure 4-26. This situation occurs after merging two disjoined segments, and then refining the vexed area. Although when the components where merged some overlapping between them existed, after refining the vexed area, part of it is discarded, and the two disjoined segments are not anyhow linked anymore, thus they must be broken.

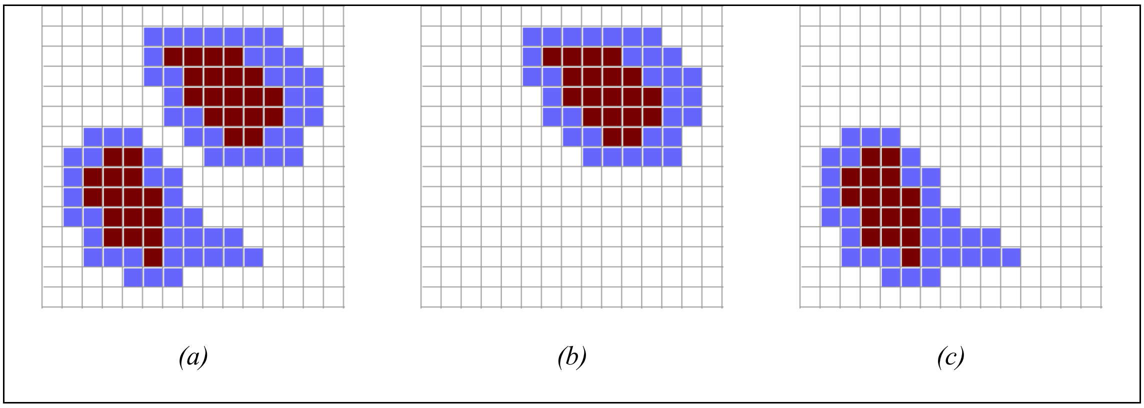


Figure 4-26 – (a) Component comprising of two disjoined segments (red) and its vexed area. (b) First component after breaking and vexed area. (c) Second component after breaking and vexed area.

Recursive Merging

The merging process is repeated level by level, moving up in the tree structure until the root of the tree (the original image) is reached. Specifically, all components created in layers of the same level in the tree structure are copied one level up. At the next level the same steps are followed: merging, refinement of the vexed areas (according to the type of the new layer), and component integrity checking, and the new components are copied one more level up. The pseudo-code for the merging process is shown in Figure 4-27. This function is called once for the parent layer, and recursively performs merging in a bottom-up manner.

```
Merge (Layer)
{
  If (Layer is leaf-layer)
  {
    Find Connected Components in the Layer
    Find Vexed Areas for the Components
    Merge Overlapping Components
  }
  Else //Layer has children layers
  {
     For Each ChildLayer
     {
       Merge(ChildLayer)
       Copy Components of ChildLayer in Layer
     }
     Merge Overlapping Components
     RefineVexedAreas(Layer)
     CheckComponentIntegrity(Layer)
  }
}
```

Figure 4-27 – Pseudo-code for the merging function of the Split and Merge Method.

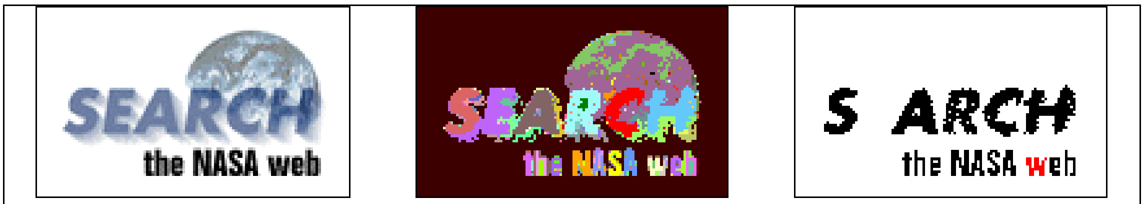
4.6. Results and Short Discussion

In this section, some representative results of the method for various images will be shown. Detailed results obtained by evaluating the method using a big number of images collected from the World Wide Web as well as details about the data set used will be given in Chapter 7. In addition, in Chapter 7 a comparison between the two methods described in this thesis is made.

The evaluation of the segmentation method was performed by visual inspection. This assessment can be subjective since the borders of the characters are not precisely defined in most cases (due to anti-aliasing or other artefacts caused by compression). Nevertheless, since no other information is available about which pixel belongs to a character and which to the background (no ground truth information is available for Web images), visual inspection is the only method of assessment that can be used. Since visual assessment is inherently subjective, in cases where it is not clear whether a character-like component contains any pixel of the background or not, the evaluator decides on the outcome based on whether by seeing the component on its own they can recognize the character or not. The foundation for this is that even if a few pixels have been misclassified, as long as the overall shape can still be recognized, the character would be identifiable by OCR software.

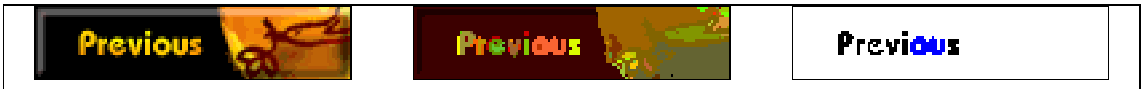
The following rules apply regarding the categorization of the results. Each character contained in the image is characterised as *identified*, *merged*, *broken* or *missed*. Identified characters are those that are described by a single component. Broken ones, are the characters described by more than one component, as long as each of these components contain only pixels of the character in question (not any background pixels). If two or more characters are described by only one component, yet no part of the background is merged in the same component, then they are characterised as merged. Finally, missed are the characters for which no component or combination of components exists that describes them completely without containing pixels of the background.

In the figures below, a number of images can be seen along with the corresponding results. For every image, the original is given along with an image of the final segmentation, and an image of the segmented characters. In the image of the final segmentation, each component is painted in a random colour. In the image of the segmented characters, the black characters denote correctly identified ones, the red characters broken ones and the blue characters merged ones. Any missed character is not illustrated at all. It should be made clear at this point, that the images of the segmented characters shown, are produced manually at this point, and are meant to serve as a better illustration of the final segmentation. Automatic classification of components will be discussed in Chapter 6.



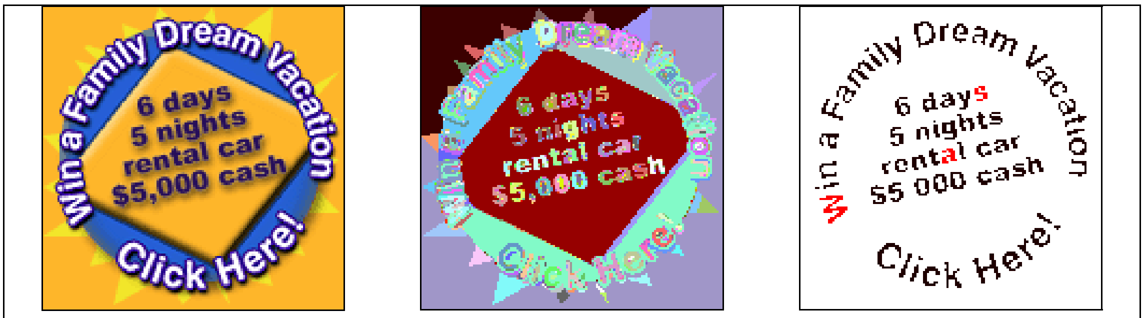
Original *Final Segmentation* *Segmented Characters*

Figure 4-28 – An image containing multi-coloured characters smoothly blending to the background and the corresponding results. Note how character "E" is erroneously merged with part of the shadow, and character "w" is split in two components. Lowering the Lightness similarity thresholds in this case would allow taking "w" as a whole component, but would produce more cases like "E".



Original *Final Segmentation* *Segmented Characters*

Figure 4-29 – Image containing single-coloured background and multi-coloured text. Anti-aliasing produces a fuzzy area between characters “o” and “u” that ultimately causes the merging of the characters.



Original *Final Segmentation* *Segmented Characters*

Figure 4-30 – An image containing characters with shadows. Most of the text is written circularly.

The method is not configured to give the optimum results in every individual situation; instead, the thresholds were selected so that reasonable results can be obtained over a range of fundamentally different images. An example can be seen in Figure 4-31. Using stricter Lightness thresholds, the method produces a better final segmentation for the same image as in Figure 4-28. Nevertheless, using stricter lightness thresholds can produce worse final segmentations in other images as can be seen in Figure 4-32. In this example, the highlights and shadows (areas of higher or lower Lightness respectively) of the characters in the word “Google”, are segmented as separate components in the case that stricter Lightness thresholds are used.

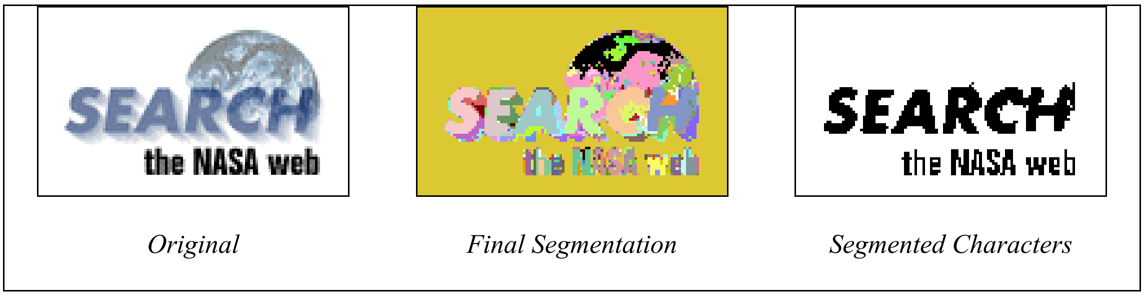


Figure 4-31 – Results for the same image as in Figure 4-28, using stricter lightness thresholds. In this case, letter "E" is identified, and letter "w" is not broken into smaller components.

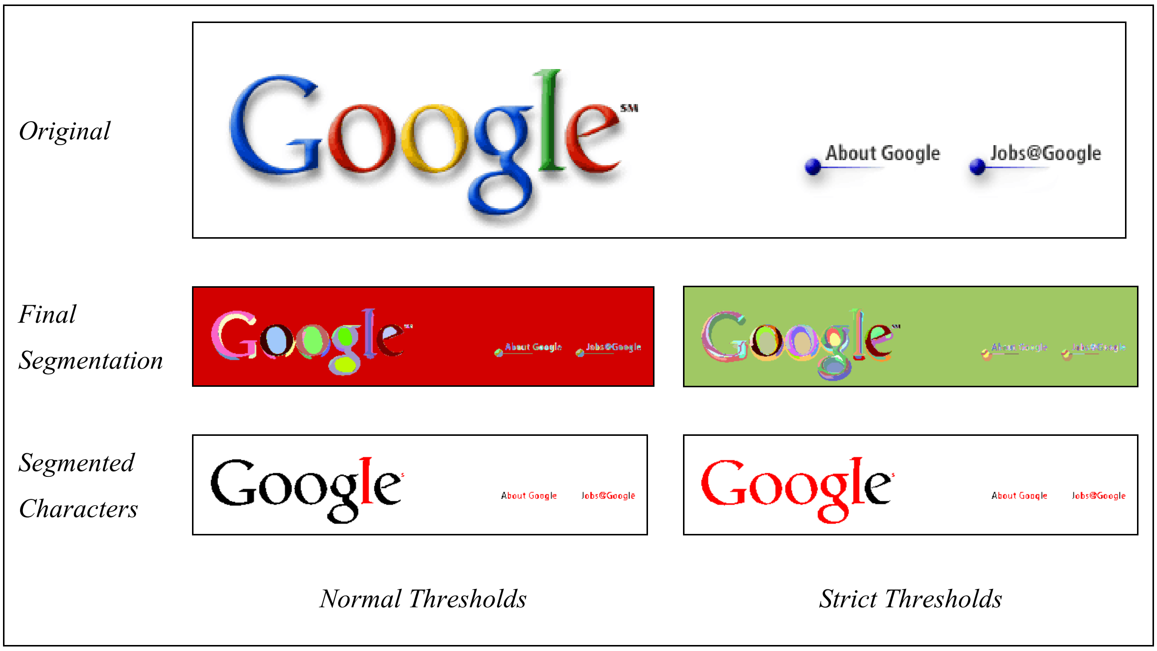


Figure 4-32 – An image with multi-coloured characters. Because a number of shades of the same colour are found in each character, using stricter lightness thresholds prevents the method from identifying each character as a whole component.

Both the Hue and the Lightness similarity thresholds used can be optimised for each individual image, so that the best segmentation is obtained each time. For example, images with well-separated Hues that at the same time have Lightness gradients would benefit from strict Hue similarity thresholds and more relaxed Lightness similarity thresholds. Nevertheless, it proved considerably difficult to automatically decide what thresholds to use, since the colour content of the image (in terms of well or not well separated Hues or Lightness) is not easy to analyse. The histograms or Hue and Lightness could be used towards analysing the colour content of the image, but as detailed in Section 4.4.2, this is not a trivial process. The thresholds used here are a trade-off, so that the method works reasonably well with a wide range of images.

4.7. Conclusion

The method described in this chapter aims to segment multi-colour Web images containing text. The method works in a split-and-merge manner and involves an innovative way to manipulate colour information. As stated in the introduction of this chapter, this is an anthropocentric approach, modelling closely the way humans perceive colour. To achieve this, certain key steps were taken, mainly regarding the use of *HLS* as an appropriate colour space and the introduction of certain similarity thresholds for each colour component as derived by experiments conducted in realistic situations.

A key point of this approach is the separation of chromatic from achromatic pixels in the image. Although this is not a new idea [28, 29, 82], we consider the use of experimentally derived data, that take into account the effect the different Hues, superior than using a single saturation (and lightness) threshold.

Furthermore, experimental data about Hue and Lightness discrimination were used to define the appropriate thresholds for Hue and Lightness similarity. The experimental data derived for colour purity are directly related to Saturation similarity, and are used as such whenever a decision based on Saturation needs to be made. This set of experimental data is used throughout the splitting and merging process, ensuring that components will finally consist of pixels having visually similar colours. Provision for the identification of gradient components and components with shadows or highlights, is supplied by allowing for some flexibility of the strict similarity thresholds derived from the experimental data.

The next open problem towards text extraction is the identification of character-like components. Results such as the above are not immediately usable by OCR methods, since there is no information regarding which components represent characters and which ones represent parts of the background. Therefore, each component must be assessed and characterized as either character-like or background. This process of component classification will be discussed in Chapter 6.

Chapter 5

Fuzzy Segmentation Method

The segmentation method described in this chapter, initially identifies connecting components of constant colour inside a given image, and subsequently merges them aiming at the creation of regions (larger components) representing the characters embedded in the image. This segmentation method is based on a metric of closeness between components called *Propinquity*, which is defined with a fuzzy inference system. The *Propinquity* can be calculated between any two components, and is defined based on the colour distance and the topological relationship between the components. The basic concepts are discussed next, followed by a detailed description of the method.

5.1. Basic Concepts – Innovations

The Split and Merge Method discussed in Chapter 4, manipulates colour by treating separately different colour attributes and using discrimination information for each attribute to decide on colour similarity. Colour discrimination in the Fuzzy Segmentation Method, is approached in a distinctly different way, not focusing on separate colour components, but on one single metric for colour similarity. The underlying assumption that Web Images are created in such a way as to enable human beings to read the textual content is common to both the Split and Merge Method and the Fuzzy Segmentation method.

The *HLS* colour system lacks a straightforward measurement method for perceived colour difference. This is due to the fact that colours having equal

(Euclidean) distances in the *HLS* colour space may not necessarily be perceived by humans as being equally dissimilar. A more suitable colour system would be one that exhibits perceptual uniformity.[1] The CIE (Commission Internationale de l'Eclairage) has standardized two colour systems: $L^*a^*b^*$ and $L^*u^*v^*$ (sometimes also referred to as CIELAB and CIELUV) based upon the CIE *XYZ* colour system [30, 116]. These colour systems offer a significant improvement over the perceptual non-uniformity of CIE *XYZ* [152, 211] and are a more appropriate choice to use in that aspect than *HLS*. Therefore, the Euclidean distance in the $L^*a^*b^*$ colour space is used here as indicative of colour similarity.

Colour is a very important attribute in identifying objects in colour images, nevertheless, is not always adequate, especially when dealing with complex colour combinations. Shape plays an equally important role, and it should be used in conjunction with colour information to achieve a correct segmentation. That extra step, of incorporating topological relationship information between components into the merging process is taken here. Towards this, a *Propinquity* measure based on both the colour distance and the topological relationship between components is defined by means of a fuzzy inference system.

5.2. Description of method

The method starts by performing an one-pass connected component identification process. Since the image is not bi-level (therefore, the components cannot be easily defined in terms of black and white), colour similarity between pixels is used for component identification.

The rationale behind this pre-processing step (expressing the image in terms of components) is that if some neighbouring pixels have colours that humans cannot differentiate, it is beneficial to treat those pixels as a single component. By doing so, the processing time of the method is substantially reduced.

The components resulted from the initial component identification step are subsequently aggregated into larger regions. Using a fuzzy inference system defined,

[1] For example, assume that two colours have HLS (Euclidean) distance δ. Humans find it more difficult to differentiate between the two colours if they both lie in the green band than if the two colours lie in the red-orange band (with the distance remaining δ in both cases). This is because humans are more sensitive to the red-orange wavelengths than they are to the green ones.

the *Propinquity* between all possible pairs of components is calculated and a sorted (in terms of *Propinquity*) list of pairs of components is produced. As long as a pair of components exists for which the propinquity value is above a pre-defined threshold, a merger takes place, and the list is updated accordingly to reflect the changes and accommodate the new component. The process finishes when no pair of components exists with propinquity above the pre-defined threshold.

The above steps of pre-processing and merging, as well as the fuzzy inference system used to define the propinquity measure, are detailed next.

5.3. Initial Connected Components Analysis

Identifying connecting components in bi-level images is a relatively trivial task, since only two classes exist, defined *a-priori*, and the pixels are already labelled. Connected component identification is merely the process of checking each black (foreground colour) pixel, and if it neighbours a connected component, assign the pixel to the connected component. If a pixel neighbours more than one component, all the components are merged into one (Figure 5-1a).

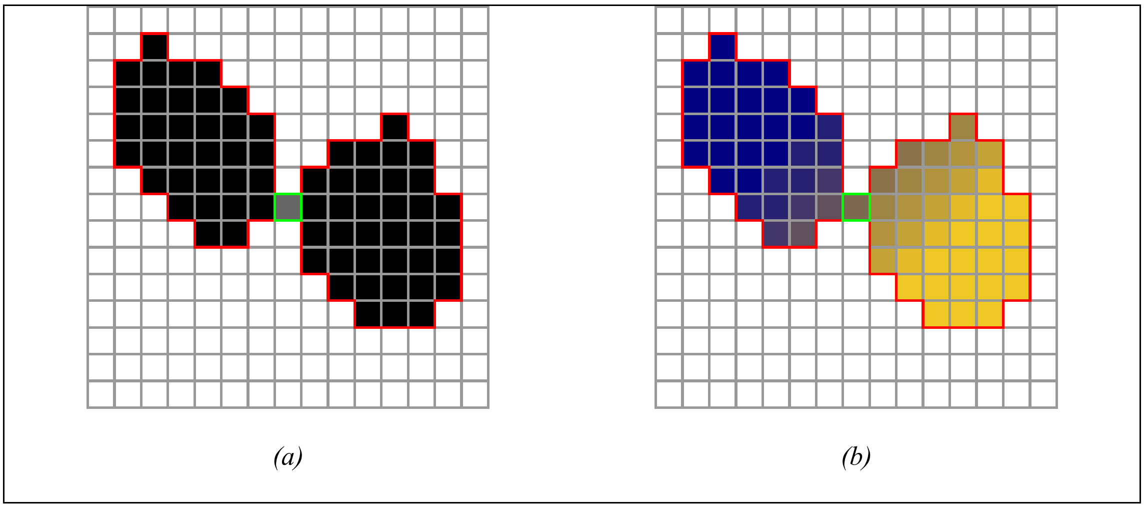


Figure 5-1 – (a) If a pixel is neighbouring more than one component, the components are always merged in bi-level images. (b) In colour images, if a pixel is similar to more than one component, it is not trivial to decide whether the components should be merged and which component the pixel should be assigned to.

In colour images, identifying connected components involves checking the similarity of the colour of a pixel to the colour of neighbouring components and deciding which one, if any, is similar enough for the pixel to be assigned to. A pixel during this process can be similar to more than one components, in which case it is not straightforward to decide whether it should be assigned to just one of them, or if the components should be merged (Figure 5-1b).

Connected Component Identification Algorithm

The connected component identification process used here, is a one-pass algorithm, which checks each pixel's similarity with four of its neighbours and decides whether the pixel should be assigned to an existing component, or not. The process starts from the top left corner of the image and works its way in a top to bottom, left to right scheme. For each pixel, the colour distance is computed between the pixel and the components to which four of its neighbouring pixels belong as shown in Figure 5-2. Only those four checks are performed at this point, because the rest of the neighbouring pixels have not been assigned to a component yet.

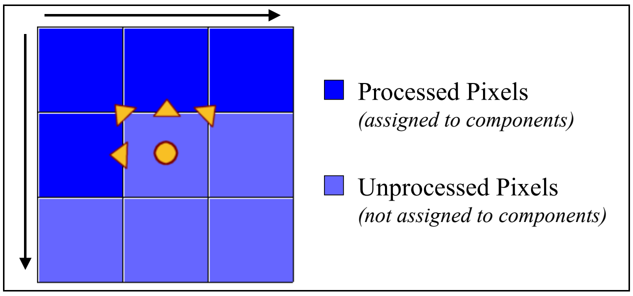


Figure 5-2 – For each pixel, the components to which their four processed neighbouring pixels belong are checked.

The smallest colour distance is identified, and if below a pre-defined threshold, the pixel is assigned to the associated component. If the pixel presents a colour distance below the pre-defined threshold with one or more of the other neighbouring components, then the colour distance between the average colours of those components and the component to which the pixel was assigned is checked. If this colour distance is also below the threshold, the two components are merged.

This process is illustrated in Figure 5-3. In the situation depicted, the pixel being processed is similar to both component *#3* (blue outline) and component *#2* (green outline). The pixel is initially merged with component *#3*, since the associated colour distance is the smallest computed. Then the colour distance between component *#3* (that was assigned the pixel) and component *#2* is computed, and if it is below the pre-defined threshold, the two components are merged.

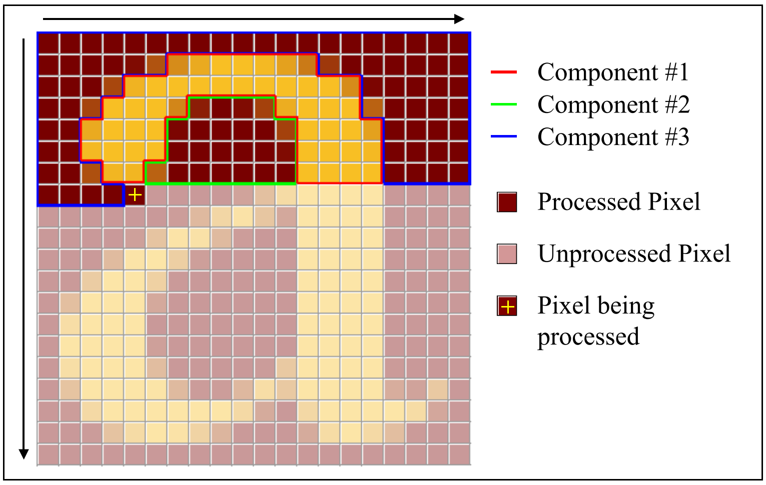


Figure 5-3 – A partially processed image. The unprocessed pixels are shown in dim colours. The next pixel to be processed is similar to both the blue and the green outlined component. The pixel will be assigned to the component with which it has the smallest colour distance, for this case the blue outlined one. Subsequently the second component (green outlined) will be compared to the one the pixel was assigned to, and if similar, they will be merged.

Colour Distance

The most important part of the above process is the definition of the colour distance metric and, subsequently, of the threshold under which two colours are considered similar.

In order for the colour distance to be related to the perceived colour difference, a colour system, which is perceptually uniform, should be used. A system is perceptually uniform if a small perturbation to a component value is approximately equally perceptible across the range of that value. Both the *RGB* and the CIE *XYZ* systems are far from exhibiting perceptual uniformity. As mentioned before, the CIE has standardised two colour systems $L^*a^*b^*$ and $L^*u^*v^*$ which greatly improve the perceptual non-uniformity of CIE *XYZ*. In both those systems, the Euclidean distance between two points can be used as a metric of colour similarity. In the current implementation of the algorithm we use the Euclidean distance in $L^*a^*b^*$ (CIE specification of 1976) as a perceptually uniform colour distance metric. Other colour systems, such as $L^*u^*v^*$ (CIE specification of 1976) and La_Lb_L (Hunter specification of 1948 and 1958) [211] have also been tried and give similar results.

	R	***G***	***B***	***White***
x	0.640	0.300	0.150	0.3127
y	0.330	0.600	0.060	0.3290
z	0.030	0.100	0.790	0.3582

Table 5-1 - Primaries and D$_{65}$ white point of Rec. 709

The image data is coded in the *RGB* colour system, so a way is needed to convert from *RGB* to $L^*a^*b^*$. A direct conversion exists between CIE *XYZ* and $L^*a^*b^*$. Still, similarly to the Split and Merge method, a conversion is needed between the *RGB* and the CIE *XYZ* colour systems. It is not feasible to convert from a device-dependant colour system to a device-independent one without any extra knowledge about the hardware used both for the creation and for displaying of the *RGB* data. The *RGB* colour system is device-dependant, however between monitors that conform to the standard *Rec. 709* [84], *RGB* colours can be considered to be unvarying. The primaries and the D_{65} white point of *Rec. 709* are displayed in Table 5-1. To convert from RGB_{709} to CIE *XYZ* and vice-versa we use the transforms:

$$\begin{bmatrix} R_{709} \\ G_{709} \\ B_{709} \end{bmatrix} = \begin{bmatrix} 3.240479 & -1.537150 & 0.498535 \\ -0.969256 & 1.875992 & 0.041556 \\ 0.055648 & -0.204043 & 1.057311 \end{bmatrix} \cdot \begin{bmatrix} X \\ Y \\ Z \end{bmatrix}$$

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} 0.412453 & 0.357580 & 0.180423 \\ 0.212671 & 0.715160 & 0.072169 \\ 0.019334 & 0.119193 & 0.950227 \end{bmatrix} \cdot \begin{bmatrix} R_{709} \\ G_{709} \\ B_{709} \end{bmatrix}$$

Eq. 5-1

Then the conversion of the CIE *XYZ* values to $L^*a^*b^*$ is given by:

$$L^* = 116 \cdot \left(\frac{Y}{Y_n}\right)^{1/3} - 16$$

Eq. 5-2

$$a^* = 500 \cdot \left[\left(\frac{X}{X_n}\right)^{\frac{1}{3}} - \left(\frac{Y}{Y_n}\right)^{\frac{1}{3}}\right]$$

Eq. 5-3

$$b^* = 200 \cdot \left[\left(\frac{Y}{Y_n} \right)^{\frac{1}{3}} - \left(\frac{Z}{Z_n} \right)^{\frac{1}{3}} \right] \qquad \textit{Eq. 5-4}$$

$$\text{for } \frac{X}{X_n}, \frac{Y}{Y_n}, \frac{Z}{Z_n} > 0.008856 \qquad \textit{Eq. 5-5}$$

where *(X_n, Y_n, Z_n)* are the coordinates of the white point [116, 211]. The colour difference formula for $L^*a^*b^*$ is given in Eq. 5-6.

$$\Delta E^* = \left[\left(\Delta L^*\right)^2 + \left(\Delta a^*\right)^2 + \left(\Delta b^*\right)^2 \right]^{1/2} \qquad \textit{Eq. 5-6}$$

Similarity Threshold

The setting of the threshold for the colour difference below which two colours can be considered similar, is dependant on the purpose of this first step of connected component analysis. The rationale behind performing this initial grouping of pixels into connected components is that having larger structural units instead of individual pixels relieves the subsequent merging algorithm from a significant computational load. The reason for that is that if the pixels are grouped into connected components, fewer comparisons will take place, as the number of components will be substantially smaller than the number of the initial pixels. Generally, the larger the components participating to the merging process are, the fewer comparisons will be necessary. Following this syllogism, it would be advantageous to come up with large components. On the other hand, the desirable outcome of the method is a correct separation of characters from the background, so any false merging at this stage would hinder the segmentation process.

Based on that, the ideal threshold should permit as many mergers as possible as long as no merger between any part of the text and the background occurs. After experimenting with different thresholds, a value of *$\Delta E^*=20$* was chosen. This value permits very similar colours to be merged but at the same time, adheres to the aforementioned requirement that no false mergers occur between parts of the text and the background.

At this point it should be mentioned that the initial connected component identification could (in theory) be totally avoided, effectively allowing the merging algorithm to work with the smallest base units available: individual pixels. Nevertheless, even the smallest amount of initial grouping proves to be significantly beneficial in terms of computational time.

5.4. The Fuzzy Inference System

In this section, the fuzzy inference system used to define *Propinquity* will be described. This fuzzy inference system takes two inputs and exports one output. The two inputs, namely *Colour Distance* and *Connections Ratio* and the *Propinquity* output are analysed and their representation in the fuzzy inference system is given next. Basic definitions for fuzzy logic can be found in Appendix B.

For the design of the fuzzy inference system, MATLAB 5.3 was used. The fuzzy inference system described here is a Mamdani-type one where the implication method employed is truncation, and the defuzzyfication method is the centroid calculation of the final output curve. The above processes are described in more detail in Appendix B. Subsequently, for the implementation of the fuzzy inference system the C libraries for fuzzy logic provided by MATLAB were employed. The use of MATLAB both for the design and implementation of the system, ensures that no incompatibilities exist between the two phases.

5.4.1. Colour Distance

The colour distance between two components is a factor that strongly influences the decision of whether the components should be merged or not. As such, colour distance participates in the definition of *Propinquity* as the first input of the fuzzy inference system.

Colour distance can be used as a metric indicating when two components are certainly different and therefore should not be merged. Nevertheless, it is not as helpful as a metric indicating when two components are similar enough to be merged, since this depends most of the cases on the colour content of their immediate neighbourhood.

One can possibly define levels of similarity between components, from absolutely identical up to definitely different, based on the colour distance between components. It is not trivial to define crisp thresholds for different levels of similarity. “Absolutely Identical” or “Definitely Different” are rather vague concepts, which can be easier described as fuzzy sets. This makes Colour Distance an excellent candidate for inclusion in a fuzzy inference system.

The colour of each component is computed as the average colour of the pixels belonging to the component. The colour difference between components is then defined as the Euclidean distance between their average colours in the $L^*a^*b^*$ space. This metric is the first input of the fuzzy inference system.

It should be mentioned at this point that computing the average of a number of different colours in the *RGB* colour system and subsequently converting the result in $L^*a^*b^*$ does not necessarily give the same value as computing the average of the colours directly in the $L^*a^*b^*$ system (there is a non-linear transformation between them). Although the difference between the two values is never practically significant, since the pixels belonging to the same component should have by definition very similar colours, there is an important reason why averaging in the *RGB* should be preferred. *RGB* is defined to be linear. This effectively means that if one mixes different amounts of two colours defined in the *RGB* colour space, the resulting colour would lie on the line connecting the initial two. As a result, the average colour of a number of colours given in *RGB* is the colour one sees if they combine the initial colours, which is not the case with the $L^*a^*b^*$ colour space. Therefore, due to the linearity of the colour space the average colour of components is computed in the *RGB* colour system.

If two components *a* and *b* are merged to a new component *(a+b)*, then the average colour of the resulting component can be easily calculated by the following equation:

$$C_{(a+b)} = \frac{S_a \cdot C_a + S_b \cdot C_b}{S_a + S_b} \qquad \text{Eq. 5-7}$$

where S_a and S_b are the sizes (in terms of number of pixels), and C_a and C_b are the average colours of components *a* and *b* respectively. This is a very important feature in terms of computational efficiency as will be seen later in Section 5.5.

Fuzzy Sets and Membership Functions

The Colour Distance needs to be incorporated into the fuzzy inference system. This involves the definition of a number of fuzzy sets and the corresponding membership functions. A membership function is a curve that defines how each point in the input space is mapped to a membership value (or a degree of membership of the associated fuzzy set) between *0* and *1*. A more detailed description of these terms is given in Appendix B. The fuzzy sets and the corresponding membership functions defined for the Colour Distance input are illustrated in Figure 5-4.

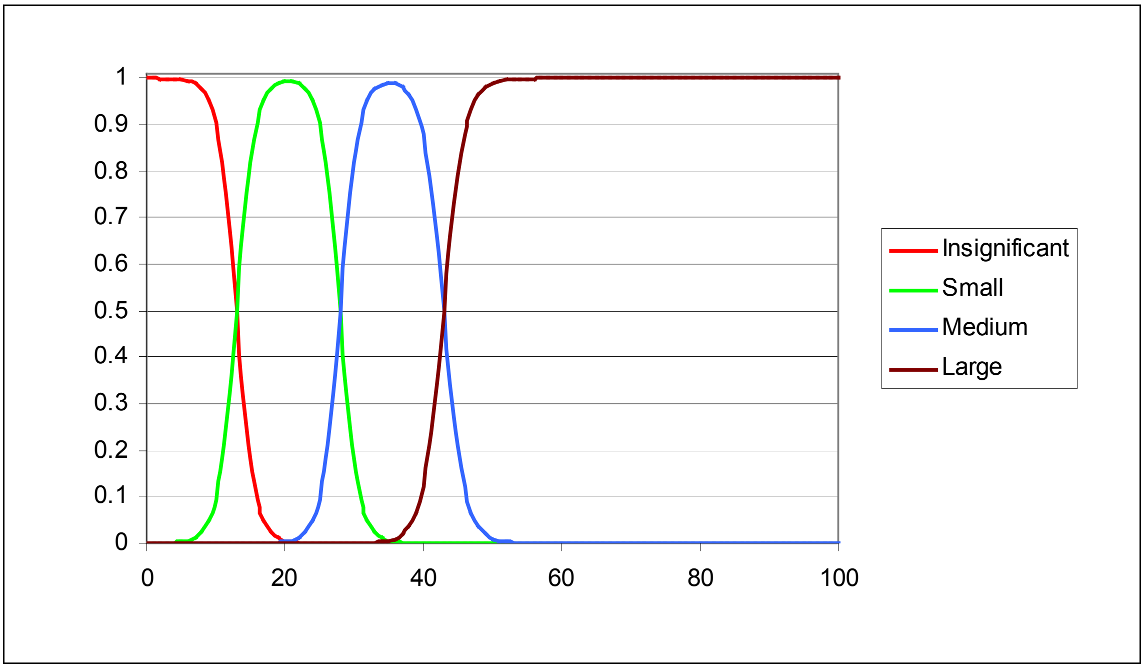


Figure 5-4 – Membership functions for the Colour Distance input.

The fuzzy set expressing the range of colour distances for which two colours would be certainly different, is defined with the "Large" membership function in Figure 5-4. The cut-off point for this membership function, was set after experimentation to $\Delta E^{*} \approx 43$.

At the other end, colours with distances less than $\Delta E^{*}=20$ should be considered similar. This conforms to the threshold used for the initial component identification process described in the previous section. Two fuzzy sets were defined for colour distances under this threshold, named "Small" and "Insignificant", the membership functions of which are shown in Figure 5-4. The rationale behind defining two fuzzy sets for this range of colour distances, instead of one, is to provide for more flexibility in designing the system. Although both "Small" and "Insignificant" colour distances are considered small enough for two components to be merged, having two fuzzy sets in this range allows for better control over the order mergers are performed. Mergers are performed hierarchically (based on the Propinquity value) as will be seen in Section 5.5. The "Small" membership function peaks at $\Delta E^{*}=20$.

For the middle range of colour distances, no decision can be made with certainty based solely on the Colour Distance input. Instead, the fuzzy inference system relies strongly on the second input as well. This middle range is represented by a "Medium" fuzzy set, defined as shown in Figure 5-4.

5.4.2. Topological Relationships between Components

The way components are arranged in the image is another factor influencing the aggregation process. The topological relationships between components can be expressed by simple metrics such as the spatial distance between them, or more complex features involving the boundaries or the shapes of the components. Two of the metrics examined as candidates for an input of the fuzzy inference system, are described below, followed by the definition of the "Connections Ratio", which is the attribute finally used.

Euclidean Distance based Attributes

The Euclidean distance between two components is the simplest topological metric that can be used. There is a number of distances one can measure between two components: the distance between their centres of gravity, the minimum distance between their borders etc. Unfortunately, Euclidean distance-based features prove unreliable when it comes to deciding whether two components should be merged or not. The main reason for this, is that Euclidean distance-based features fail to capture the shape of the components or the way components are placed relatively to each other, rather such features provide a spatial distance between components.

An example to highlight this problem for the Euclidean distance between the centres of gravity of components is shown in Figure 5-5. In this case the "c" shaped component (component *a*) presents a small Euclidean distance with the component representing the inside area (component *b*), but not with the small component on the right (blue). In the context of character extraction, it would make more sense to merge component *a* with component *b* (if the colour distance allows so) resulting to an "o" shaped component, than component *a* with component *c* as indicated by the Euclidean distance.

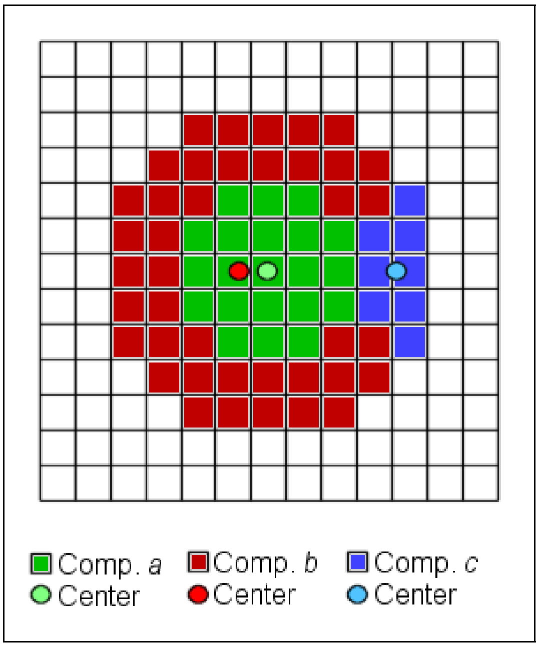


Figure 5-5 – Example of Euclidean distance between centres of gravity of components. The components with the smallest distance are not always the best to merge.

Another fundamental problem with using the Euclidean distance between the centres of gravity of components, is that it cannot give any indication about whether the two components actually share part of their boundaries of not. For that reason the Euclidean distance between points of the boundaries of two components could be used. This is much more difficult to compute, and the main problem would be that, although it shows whether two components touch or not, it does not indicate the extent of this boundary sharing.

Boundary Sharing

Subsequently, a different feature was examined, which expresses the extent to which a component's boundary is shared with another component. We define as the *Directional Boundary Sharing* ($BS_{a\to b}$) of component *a* to component *b*, the number of boundary pixels of *a* neighbouring with *b*, divided by the total number of pixels constituting the boundary of *a*:

$$BS_{a\to b} = \frac{BP_{a\to b}}{BP_a} \qquad \text{Eq. 5-8}$$

where $BP_{a\to b}$ is the number of boundary pixels of *a* located adjacent to the boundary of *b* and BP_a is the total of boundary pixels of component *a*. As such, Directional Boundary Sharing takes a value in the range [*0, 1*]. As can be seen by the definition, the Directional Boundary Sharing of component *a* to *b* is not necessarily equal to the Directional Boundary Sharing of component *b* to *a*, that is $BS_{a\to b} \neq BS_{b\to a}$. For this reason, we define as Boundary Sharing between the two components *a* and *b* the Directional Boundary Sharing from the component having the smaller boundary to the component having the larger one, that is:

$$BS_{a,b} = \begin{cases} BS_{a\to b} & , BP_a < BP_b \\ BS_{b\to a} & , BP_b > BP_a \\ \max(BS_{a\to b}, BS_{b\to a}) & , BP_a = BP_b \end{cases} \qquad \text{Eq. 5-9}$$

The Boundary Sharing feature has some strong advantages as a feature indicative of the topological relation of two components. First, it directly indicates whether two components are neighbours or not, since a value of zero would mean that the components in question are not touching at all. Furthermore, it provides a descriptive value of the degree of connectivity between two components, from zero when the components are not touching at all, to one, when one component is completely included in the other.

Since a large number of comparisons are made during the component aggregation phase, it is beneficial to select a feature that is easy to compute for the components produced after each merger, if possible without the need to read again data from the image. Unfortunately, in order to compute the Boundary Sharing after each merger, it is necessary to count again the number of Boundary Pixels for the new component, which (for the number of comparisons performed) can take a substantially large amount of time. This problem is illustrated in Figure 5-6.

In certain cases, the number of Boundary Pixels of the component resulting from a merger can be computed from known information. In most of the cases though, when a boundary pixel is neighbouring more than one component this cannot be done. Using Boundary Sharing as an input of the fuzzy inference system, made the merging algorithm too time consuming to be practicable.

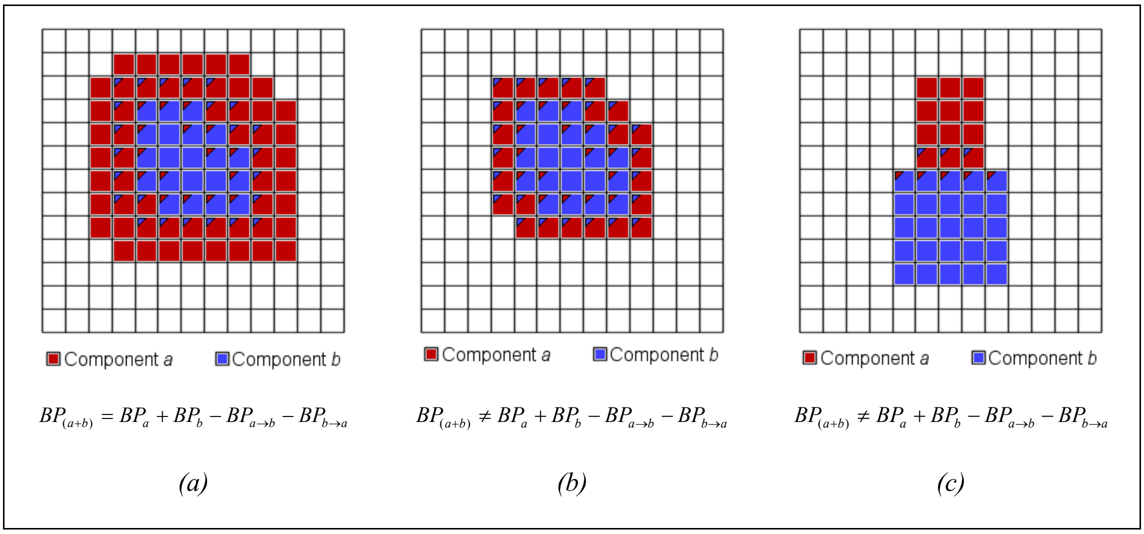

🟥 Component *a*
🟦 Component *b*
$BP_{(a+b)} = BP_a + BP_b - BP_{a \to b} - BP_{b \to a}$
(a)
🟥 Component *a*
🟦 Component *b*
$BP_{(a+b)} \neq BP_a + BP_b - BP_{a \to b} - BP_{b \to a}$
(b)
🟥 Component *a*
🟦 Component *b*
$BP_{(a+b)} \neq BP_a + BP_b - BP_{a \to b} - BP_{b \to a}$
(c)

Figure 5-6 – Different cases where the number of Boundary Pixels (BP) of the component resulting from a merger can (a) or cannot (b, c) be computed from known information. Boundary pixels are illustrated here with a triangle in the upper left corner.

5.4.3. The Connections Ratio

Another attribute was therefore sought, which also expresses the degree of connectivity between components, but at the same time is easier to calculate and use. The attribute finally used as an input in the fuzzy inference system is the *Connections Ratio*. This is defined in a way that it overcomes the problems of Boundary Sharing, by using *Connections* instead of pixels to define the boundaries of components.

Definitions

A *Connection* is defined here as a link between a pixel and any one of its 8-neighbours, each pixel thus having *8* connections. A connection can be either *internal* or *external*. A connection is called internal when both the pixel in question and the neighbouring one belong to the same component and external when the neighbouring pixel belongs to another component. Connections to pixels outside the image, are also considered external. Figure 5-7 illustrates the external and internal connections of a given component to its neighbouring components.

If C_x is the total number of connections of component x, Ci_x and Ce_x are the number of internal and external connections of component x respectively and S_x is the size of component x, where size equals the number of pixels of the component, then the following equations apply:

$$C_x = 8 \cdot S_x \qquad \textit{Eq. 5-10}$$

$$C_x = Ci_x + Ce_x \qquad \textit{Eq. 5-11}$$

Note that Eq. 5-10 applies to all components, even the ones located at the edges of the image (see Figure 5-7). By doing so, we avoid computing a smaller boundary length for those components.

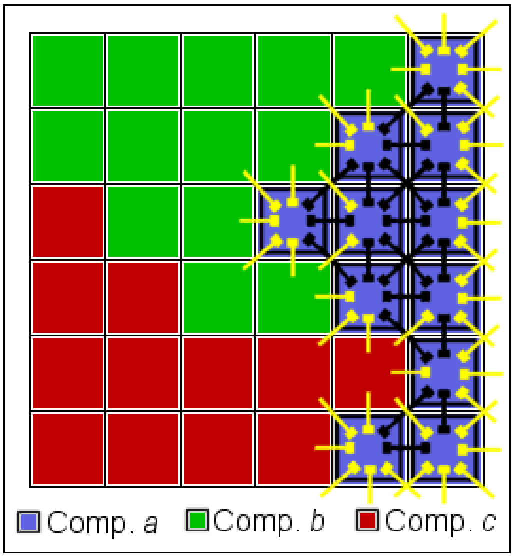


Figure 5-7 – A connected component (blue) and its internal and external connections to its neighbouring components (shown in green and red). Yellow lines indicate the external connections and black lines the internal connections.

Given any two components *a* and *b*, the Connections Ratio, denoted as $CR_{a,b}$ is defined as:

$$CR_{a,b} = \frac{Ce_{a,b}}{\min(Ce_a, Ce_b)} \qquad \textit{Eq. 5-12}$$

where $Ce_{a,b}$ is the number of (external) connections of component *a* to pixels of component *b*, and Ce_a and Ce_b refer to the total number of external connections (to all neighbouring components) of each of the components *a* and *b*, respectively. The Connections Ratio is therefore the number of connections between the two components, divided by the total number of external connections of the component with the smaller boundary. The Connections Ratio ranges from *0* to *1*.

In terms of practical significance, the Connections Ratio is far more descriptive of the topological relationship between two components than other spatial distance measures. Similarly to Boundary Sharing, a small Connections Ratio indicates loosely linked components while a large value indicates that one component is almost included in the other. In addition, in a manner similar to Boundary Sharing, the Connections Ratio provides a direct indication of whether two components are neighbouring or not in the first place, since it will equal zero if the components are disjoined.

The main advantage of the Connections Ratio over to Boundary Sharing is that it is significantly easier to compute for new components resulting from mergers between components. If we choose to merge two components *a* and *b*, then for the new component the following will hold:

(i) The total number of connections for the new component will be the sum of the total number of connections of the two components:

$$C_{a+b} = C_a + C_b \qquad \textit{Eq. 5-13}$$

(ii) The total number of internal connections for the new component will be the sum of the number of internal connections of the two components plus twice the number of connections between the two components:

$$Ci_{a+b} = Ci_a + Ci_b + 2 \cdot Ce_{a,b} \qquad \textit{Eq. 5-14}$$

(iii) The total number of external connections for the new component will be the total number of connections of the new component minus the total number of internal connections:

$$Ce_{a+b} = C_{a+b} - Ci_{a+b} \qquad \textit{Eq. 5-15}$$

which gives based on Eq. 5-11:

$$Ce_{a+b} = Ce_a + Ce_b - 2 \cdot Ce_{a,b} \qquad \textit{Eq. 5-16}$$

Given any neighbouring component *c* of the initial components *a* or *b*, for the new component *a+b* we can compute:

$$C_{a+b,c} = C_{a,c} + C_{b,c} \qquad \text{Eq. 5-17}$$

So based on Eq. 5-16 and Eq. 5-17, Eq. 5-12 becomes:

$$CR_{a+b,c} = \frac{Ce_{a+b,c}}{\min\left(Ce_{a+b}, Ce_{c}\right)} = \frac{Ce_{a,c} + Ce_{b,c}}{\min\left(Ce_{a} + Ce_{b} - 2 \cdot Ce_{a,b}, Ce_{c}\right)} \qquad \text{Eq. 5-18}$$

As shown above, the calculation of the value of the Connection Ratio feature for the component resulting from a merger requires knowledge of the individual feature values of the components involved only. Therefore, the feature values for each component need to be calculated for the components themselves only once at the beginning of the process, minimising the number of calculations after every merger.

Fuzzy Sets and Membership Functions

Characters consist of continuous strokes, therefore, if a character is split in two or more components, those components, being parts of strokes, will neighbour only partially. This is because strokes are continuous shapes of small thickness, so if a stroke is split in two components, those components will neighbour to an extent comparable to the thickness of the stroke.

The Connections Ratio input is indicative of the extent to which components neighbour each other; as a result, it can be used to indicate when components are more likely to be parts of a stroke. Components that partially neighbour, will have a Connections Ratio value in the middle range. After experimentation it was found that this middle range of Connection Ratio values indicative of components likely to be stroke parts, is between *0.05* and *0.65*. Two fuzzy sets were defined for this middle range called "Medium Low" and "Medium High", the membership functions of which are shown in Figure 5-8. The reason for defining two fuzzy sets instead of one for the middle range of Connections Ratio values is to provide for more flexibility in designing the system, in a manner similar to the fuzzy sets defined for small Colour Distance values. Specifically, although pairs of components presenting Connections Ratio in the range of either the "Medium Low" or "Medium High" fuzzy sets will be favoured at the time of component aggregation, having two fuzzy sets in this range allows for better control over the order mergers are performed. Between the two

"Medium" sets, "Medium High" is more preferable and is defined to give slightly higher Propinquity values as will be seen next. In this way, pairs of components having Connections Ratio in either the "Medium Low" or "Medium High" ranges will be merged (if the Colour Distance permits it), but those in the "Medium High" range will be aggregated first.

Connections Ratio values above *0.65* indicate pairs of components extensively neighbouring. Such pairs of components most of the times are not parts of the same character and should not be merged. Cases that fall in this range, are pairs of components representing a character and its inside area (e.g. character "o" and its inside area). Such pairs of components present Connection Ratio values in the range of the "High" fuzzy set, which is defined for values above *0.65* as seen in Figure 5-8. Connections Ratio values in this range are not favoured much and generally (depending on the Colour Distance input) give smaller Propinquity values.

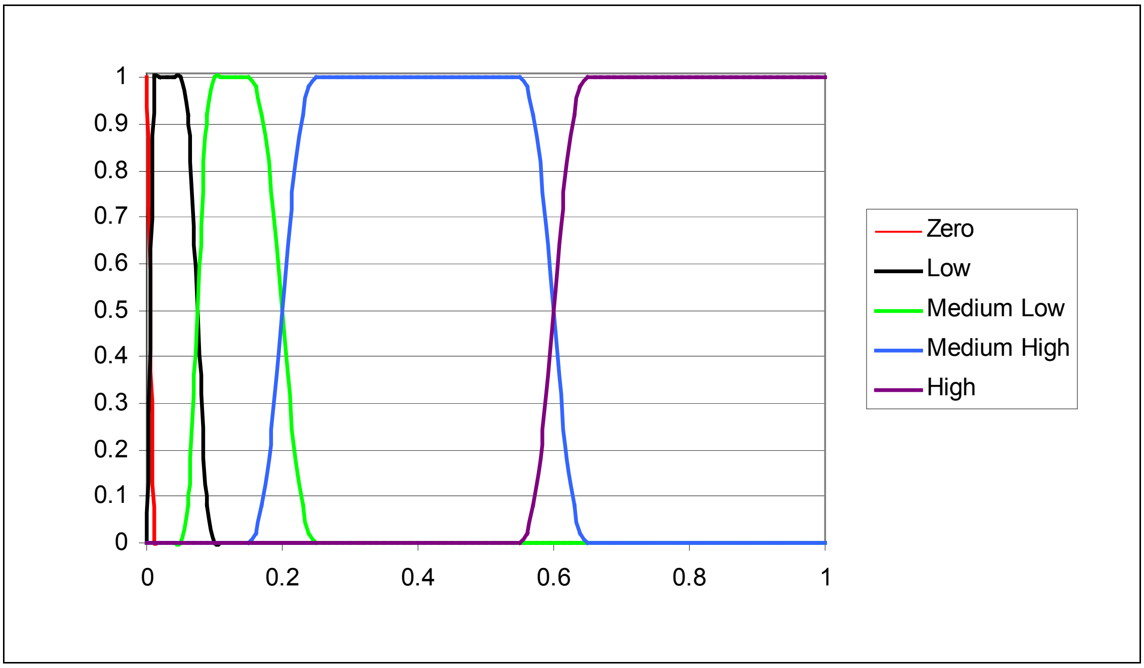


Figure 5-8 – Membership functions for the Connections Ratio input.

In a manner similar to the "High" fuzzy set, a fuzzy set called "Low" is defined for Connections Ratio values less than *0.05*. Two of components with a Connections Ratio value in that range, are generally very loosely neighbouring, and are not favoured much during the aggregation process. An important point here, is that the membership function for the "Low" fuzzy set is defined so that it steeply drops to zero when Connections Ratio values reach zero. As explained before, a Connections Ratio value of zero indicates that two components are not neighbouring, and should not be considered for merging in any case. For zero values of Connections Ratio, another fuzzy set is defined called "Zero" in order to facilitate the different handling of components that do not neighbour at all. If a pair of components presents a Connections Ratio in the "Zero" fuzzy set, it is never considered for merging, instead a Propinquity value of zero is returned independently of the Colour Distance value.

5.4.4. Combining the Inputs: Propinquity

The single output of the fuzzy inference system, the Propinquity, is defined with the help of seven fuzzy sets. Three of the fuzzy sets defined carry a special meaning. Those are the "Zero", the "Medium" and the "Definite" fuzzy sets.

The "Zero" fuzzy set is defined in such a way that a Propinquity of zero has a membership value of *1* to the set, while any other Propinquity has a membership value of *0*. This is a very crisply defined fuzzy set, which is necessary to facilitate the rejection of certain cases where components should not be merged (e.g. very large Colour Distance, or zero Connections Ratio).

On the opposite end, the "Definite" fuzzy set is defined to give high degrees of membership to Propinquity values above *0.9*. In a manner similar to the "Zero" fuzzy set, the "Definite" fuzzy set ensures that cases where components should definitely be merged (e.g. small Colour Distance and medium Connections Ratio) be awarded a high Propinquity value, therefore be placed high in the hierarchy of mergers to happen.

The Propinquity output is defined so that a value of *0.5* will be the threshold above which two components should be considered for a merger, while values less than *0.5* indicate that two components should not be merged. The "Medium" fuzzy set is defined to cover the middle range of Propinquity values (*0.4* to *0.6*) while it gives a membership value of *1* to Propinquity equal to *0.5*. This fuzzy set is used to indicate cases where it is not certain whether two components should be merged or not.

A pair of components awarded a Propinquity value above the threshold of *0.5* will effectively be merged in the subsequent component aggregation phase. Contrary, a pair of components awarded a value below that threshold, should not be considered for a merger. The range of Propinquity values between the "Medium" and the "Definite" fuzzy sets are described by two fuzzy sets: "High I" and "High II". Initially one fuzzy set was used for this range of Propinquity values, but it proved inadequate to effectively cover the full range of possible input combinations. Because *4* fuzzy sets were defined for the Colour Distance, and *5* fuzzy sets were defined for Connections Ratio, using just a single "High" fuzzy set for Propinquity did not allow for a high degree of flexibility when designing the rules of the system. The membership functions of the finally used fuzzy sets "High I" and "High II" are shown in Figure 5-9. "High I" is defined to cover the range of values from *0.5* to *0.8*, while "High II" covers the range of values from *0.75* to *0.9*.

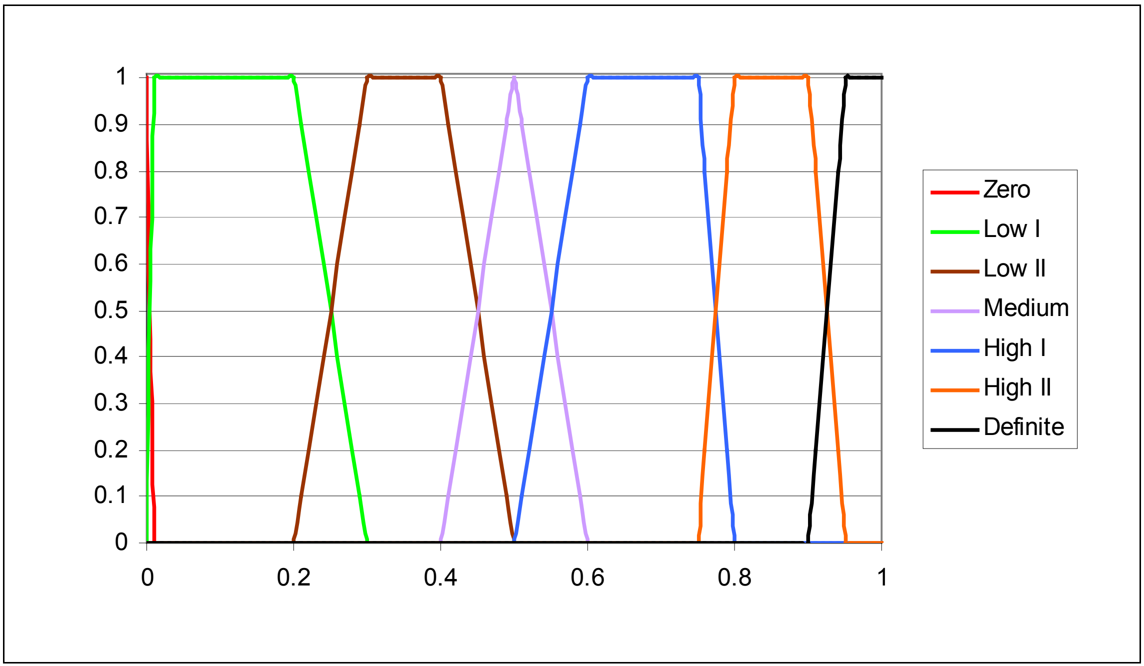


Figure 5-9 - Membership functions for the Propinquity output.

In a similar manner, the range of values between the "Zero" and "Medium" were expressed using two fuzzy sets "Low I" and "Low II". Pairs of components with a Propinquity value below *0.5* will not be considered for a merger anyway, so it is not that important (it terms of the hierarchy of mergers) to describe the range of values between *0* and *0.5* with two fuzzy sets instead of one. Nevertheless, using two fuzzy sets for this range of values, made it easier to define the rules of the fuzzy inference system. "Low I" covers the range of values between *0* and *0.3*, while "Low II" covers the range of values between *0.2* and *0.5*.

The Fuzzy Inference System

The rules of the fuzzy inference system are responsible for establishing the relations between the two inputs (Colour Distance and Connections Ratio) and the output (Propinquity). These if-then rules are simple statements, where the fuzzy sets and fuzzy operators are the subjects and verbs respectively.

The if-part of a rule is called the antecedent, while the then-part of the rule is called the consequent. Interpreting an if-then rule involves two distinct parts: first evaluating the antecedent and second applying that result to the consequent. If the antecedent is true to some degree of membership, then the consequent is also true to the same degree. The consequent of each rule specifies a fuzzy set to be assigned to the output. The output fuzzy sets for each rule of the system are then aggregated into a single output set, which is finally defuzzyfied, or resolved to a single number. The above process is described in more detail in Appendix B.

The set of rules used to define the fuzzy inference system is shown in Figure 5-10 below.

If Connections Ratio is Zero		then Propinquity is Zero
If Connections Ratio is Low	and Colour Distance is Insignificant	then Propinquity is High I
If Connections Ratio is Low	and Colour Distance is Small	then Propinquity is Medium
If Connections Ratio is Low	and Colour Distance is Medium	then Propinquity is Low II
If Connections Ratio is Low	and Colour Distance is Large	then Propinquity is Zero
If Connections Ratio is Medium Low	and Colour Distance is Insignificant	then Propinquity is High II
If Connections Ratio is Medium Low	and Colour Distance is Small	then Propinquity is High I
If Connections Ratio is Medium Low	and Colour Distance is Medium	then Propinquity is Medium
If Connections Ratio is Medium Low	and Colour Distance is Large	then Propinquity is Low I
If Connections Ratio is Medium High	and Colour Distance is Insignificant	then Propinquity is Definite
If Connections Ratio is Medium High	and Colour Distance is Small	then Propinquity is High II
If Connections Ratio is Medium High	and Colour Distance is Medium	then Propinquity is High I
If Connections Ratio is Medium High	and Colour Distance is Large	then Propinquity is Low II
If Connections Ratio is High	and Colour Distance is Insignificant	then Propinquity is High I
If Connections Ratio is High	and Colour Distance is Small	then Propinquity is Medium
If Connections Ratio is High	and Colour Distance is Medium	then Propinquity is Low II
If Connections Ratio is High	and Colour Distance is Large	then Propinquity is Zero

Figure 5-10 – The fuzzy inference system rules.

As mentioned before, a small Colour Distance indicates that the components involved are sufficiently similar (in colour) to be merged, while a large Colour

Distance informs that the components should not be considered for merging. For Colour Distance values falling at the middle range, no decision can be made with certainty. These observations manifest themselves in the rules in a specific way. If the colour distance is Small or Insignificant, the Propinquity is always set to be above medium (Medium, High I, High II or Definite). How much above medium is specified by the Connections Ratio input. For Connections Ratio values in the range of Medium Low or Medium High fuzzy sets, the Propinquity is set higher than for Connections Ratio values in the range of Small or High fuzzy sets. This conforms to the observations made before: pairs of components that correspond to parts of character strokes (partially neighbour) should be favoured during the component aggregation process.

In a similar manner, if Colour Distance is Medium of Large, the Propinquity is generally set to be below Medium (Zero, Low I, Low II, Medium) based again on the value of the second input: the Connections Ratio. Finally, an extra rule is added to handle the special case when two components are not neighbouring and should not be considered for a merger. This is the first rule shown in Figure 5-10 checking for a Connections Ratio value of zero.

The surface of Figure 5-11 represents the relationship defined by the rules of the system between the two inputs and the output. The fuzzy inference system is defined in such a way, that a propinquity of *0.5* can be used as a threshold in deciding whether two components should be considered for merging or not. We should note here that although all pairs of components with a Propinquity value above the threshold will eventually be merged, the exact value of Propinquity plays an important role during the merging process, since it dictates the order by which mergers should take place, as will be seen next. A rapid fall to zero can also be observed when Connections Ratio reaches zero, since disjoint components must not be considered for merging in any case.

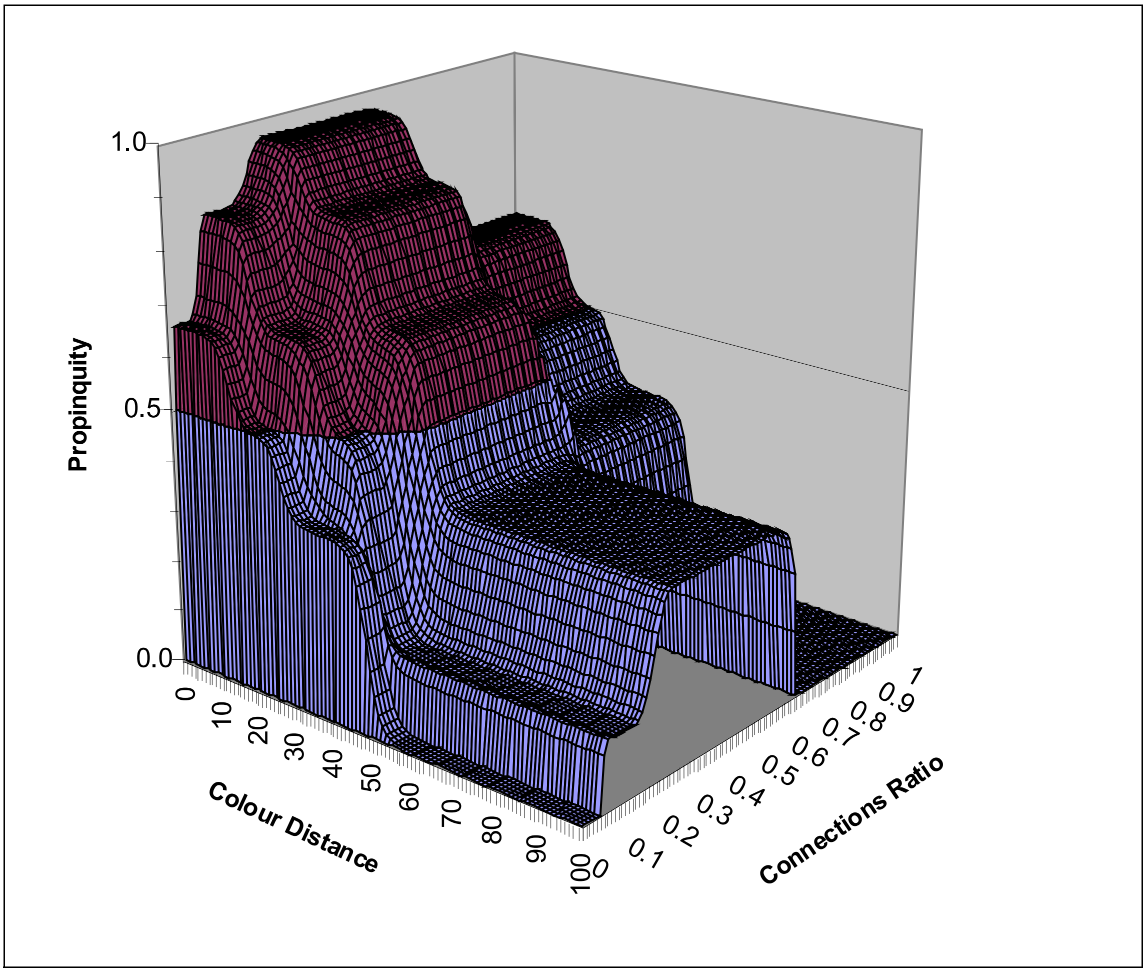


Figure 5-11 – A surface showing the mapping from Colour Distance and Connections Ratio inputs to the Propinquity output.

5.5. Aggregation of Connected Components

The components resulting from the initial connected component analysis are the base units participating in the component aggregation process, which performs mergers between pairs of components based on the Propinquity value calculated for them.

Initially, the Propinquity value for every possible pair of components is calculated. For this calculation, the average colour and the number of connections between the components need to be calculated. Those calculations are performed over the pixels of the components themselves, so they are somewhat time-consuming.

Based on the Propinquity value computed for every possible pair of components, a sorted list of possible mergers is created. As long as the Propinquity value associated with the first merger in the list is higher than a specified threshold (in this case *0.5* as derived from the definition of Propinquity), the merger in question is performed. It is essential that the sorted list is updated after each merger takes place, so that the replacement of the components involved by a new one is reflected in the sorted list of possible mergers. More specifically, if a merger is performed between components *a* and *b*, resulting in a new component *(a+b)*, then the following operations are performed:

- All mergers in the sorted list that refer to either component *a* or *b*, are changed (and the appropriate attributes recomputed) to refer to the new component *(a+b)*.
- If both components *a* and *b* have a common neighbouring component *c*, then in the list of possible mergers there would be two entries, one between *a* and *c*, and a second between *b* and *c*. After changing those mergers to refer to the newly created component, there will be two identical possible mergers in the list, between *(a+b)* and *c*. Double occurrences like these are identified and eliminated.
- The list is sorted again, to reflect the recent changes. Actually, not the whole list is sorted every time, rather the new entries are inserted in the appropriate positions, and the entries for which the propinquity value has changed, are moved so that the list remains sorted at all times.

The importance of using features that are easily recomputed after each merger, is now evident. The fact that the average colour of the new component as well as the connections feature can be calculated from the original values of the features of the components involved in the merger pays off at this stage, since there is no need to compute anything from the image data. This results in a substantial time saving.

The aggregation process continues as long as the first entry in the sorted list has a Propinquity value above the specified threshold. The pseudo code for the above process is shown in Figure 5-12.

```
AggregateComponents()
{
  For Each Component in the Image
  {
    Calculate AverageColour
    Find External Connections
  }
  For Each possible Merger
  {
    Compute ConnectionsRatio between participating Components
    Compute ColourDistace between participating Components
    Compute Propinquity
    Insert Merger in the SortedList
  }
  While (Propinquity of 1st Merger in the SortedList > 0.5)
  {
    Merge Participating Components
    Update SortedList
  }
```

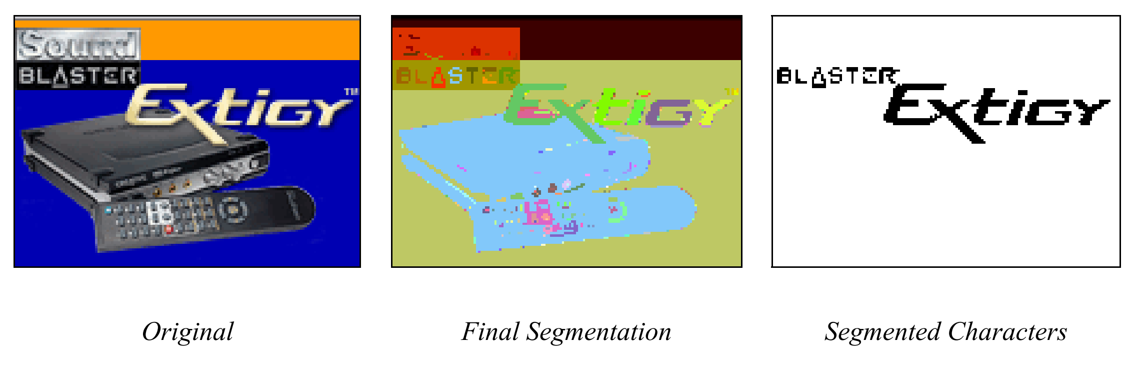
Figure 5-12 – Pseudo code for the connected component aggregation process. Commands and reserved words are typed in Bold.

5.6. Results and Short Discussion

A small sample of representative results will be shown in this section. A more complete set of results, as well as a comparison between this method and the Split-and-Merge method described in Chapter 4, are given in Chapter 7.

For the cases shown here, the original image is shown, along with an image of the final segmentation (with each component painted in different colour) and an image of the segmented characters. In the image of the segmented characters, correctly identified characters are shown in black, broken characters are shown in red, and merged characters are shown in blue.

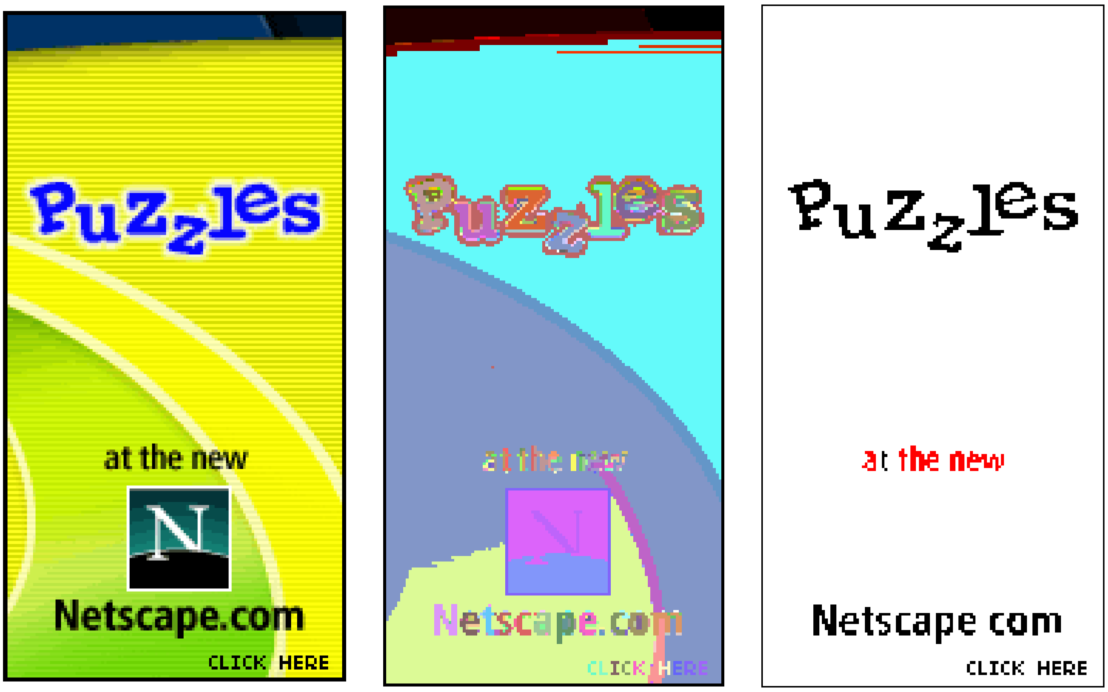
The image shown in Figure 5-13 contains split characters (character "*E*" and "*A*" in "*BLASTER*") as well as characters merged in the original (characters "*Ex*" in "*Extigy*"). The segmentation method correctly separates those characters from the surrounding background. If a character is split in the original and the segmentation method is able to identify all its parts as different components, the character is considered correctly identified. Similarly, if two or more characters are merged in the original and are segmented as a single component by the method, they are considered correctly identified. The word "*Sound*" was completely missed here due to its high similarity to the surrounding background.



Original *Final Segmentation* *Segmented Characters*

Figure 5-13 – Image with multi-coloured characters over photographic background.

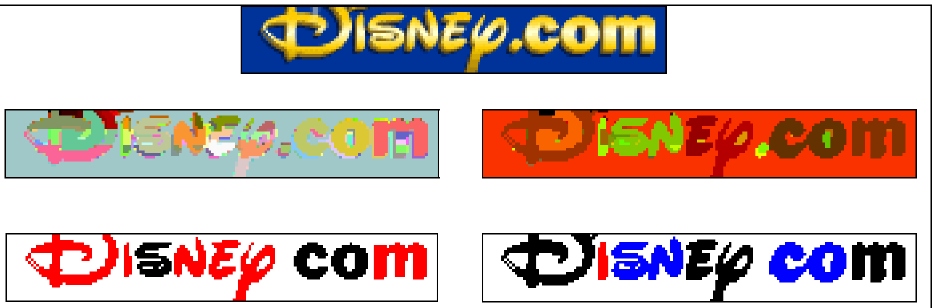
In Figure 5-14 an example of an image containing a logo (Netscape's "*N*" character), as well as one-pixel wide characters ("*CLICK HERE*" at the bottom-right corner) is given. Although the one-pixel wide characters are correctly separated from the background, the segmentation method fails to segment the logo character. What appears here to be an over-merging problem (parts of the characters merged with the background), originates from the sequence in which mergers occur (as directed by Propinquity). Parts of the gradient background of the Netscape logo (light coloured ones) are first merged with parts of the "*N*" character. The components resulting from these first mergers are subsequently merged with other parts of the background of similar colour, producing the result shown here. The dependence of the Fuzzy Segmentation method results on the sequence of mergers is an important aspect of this method. Using the Propinquity between components to define the order of the mergers in most of the cases ensures a correct separation of the characters from the background.



Original *Final Segmentation* *Segmented Characters*

Figure 5-14 – Image with single-coloured text over multi-coloured background. Most of the characters were identified correctly.

Similarly to the Split-and-Merge Method described in Chapter 4, the method presented here is configured so that reasonable results can be obtained for a range of fundamentally different images. The rules and membership functions of the fuzzy inference system can be changed to reflect stricter or more relaxed decision thresholds. Two examples illustrating this can be seen in Figure 5-15 and Figure 5-16.



Original

Final Segmentation

Segmented Characters

Normal fuzzy inference system *More relaxed fuzzy inference system*

Figure 5-15 – Image with multi-coloured characters. More relaxed membership functions for the Colour Distance input achieve better results.

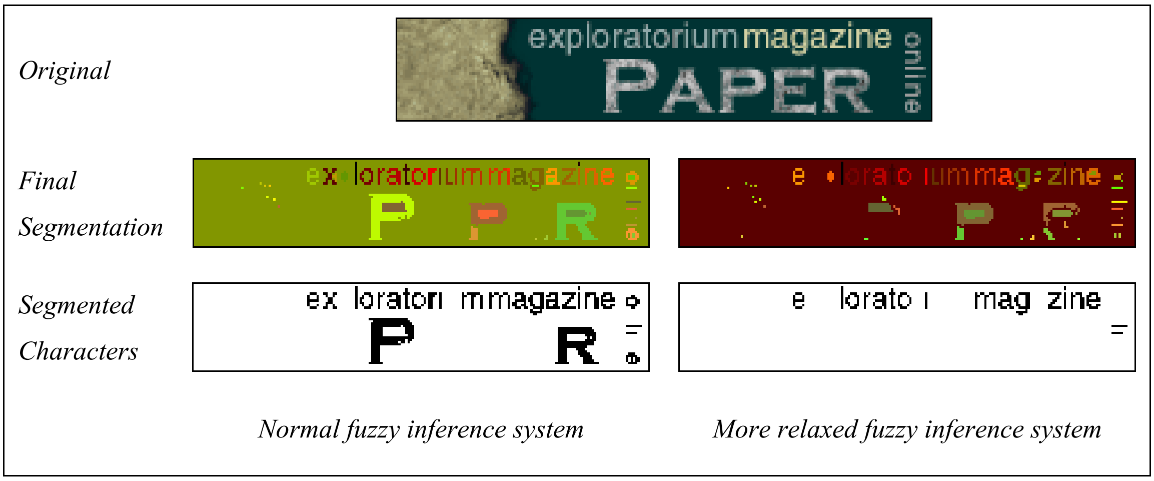

Original
Final Segmentation
Segmented Characters
Normal fuzzy inference system
More relaxed fuzzy inference system

Figure 5-16 – Image with textured text over sinlge-coloured background. Strict decision thresholds achieve far better results in this case.

A different definition of the fuzzy inference system was used here to illustrate this point. In this second definition of the fuzzy inference system, the membership functions for the Small and Medium fuzzy sets of the Colour Distance input were slightly altered. The threshold for which two components are considered similar was increased from $\Delta E^{*}=20$ to $\Delta E^{*}=\sim 25$. This results in a more relaxed definition of Propinquity, since slightly less similar (in colour) components are now allowed to merge. It is evident that different definitions of the fuzzy inference system (consequently of the Propinquity metric) affect segmentation results.

In the case illustrated in Figure 5-15 a slightly better segmentation is achieved with the more relaxed version of the fuzzy inference system. Here more characters are correctly identified (as one component) with the second version, while fewer characters are split. The opposite can be observed in Figure 5-16, where with the use of the relaxed version of the fuzzy inference system, some of the characters originally identified are now merged with the background.

5.7. Discussion

In this chapter, a new method was presented to segment text in Web images. Components of uniform colour are first identified in the image, based on the colour difference between pixels (as perceived by humans). Subsequently, the method performs mergers between the identified components towards a correct segmentation of characters (in terms of identifying individual characters as single components).

A key feature of the method, is the use of a perceptually uniform colour system (CIE $L^*a^*b^*$) to assess the colour difference between pixels and between connected components. An underlying assumption of the segmentation method is that the colour differences between the two classes (text and background) in the image, are such that human beings are able to read the text. Using a perceptually uniform colour system, allows the method to evaluate colour differences from an anthropocentric point of view, since the colour distances measured by the method are directly related to the colour dissimilarities perceived by humans.

The method described here, uses a metric called Propinquity, defined with the help of a fuzzy inference system. Propinquity combines Colour Distance and Connections Ratio (a measure of topological relation) between components to form as single closeness metric. In contrast to other methods (Lopresti and Zhou [106], Moghaddamzadeh and Bourbakis [119], Zhou et al. [220]) that employ combined colour distance and spatial distance measures, the main innovations of the approach presented here are as follows. First, the Connections Ratio is used as an indication of the topological relationship between components, in contrast to simple spatial distance metrics used by previous approaches. Second, a fuzzy inference system is used to combine the two inputs, which allows for more flexibility in contrast to other approaches (e.g. simple multiplication of colour distance and spatial distance [119]).

A number of possibilities exist for the further development the fuzzy inference system. More inputs can be introduced to the system, such as the variance of colours within a merged component. Such a measure would allow the method to eliminate mergers that result in components containing too many dissimilar colours (albeit the average colours are similar). Other inputs that could be considered are the orientation of the candidate components (if they are parts of the same stroke, the components should have similar orientation), some measure of character likeness before and after a merger etc.

Finally, techniques exist in the literature for automatic optimisation of fuzzy inference systems (by optimising the membership functions used). Such methods are based on evaluating the result (in this case the final segmentation of the image) based on some optimum solution (an optimum segmentation of the image, as suggested by ground truth data). Ground truth data for Web Images is not currently available. Automatic optimisation of the fuzzy inference system used is definitely possible in the future, on condition that ground truth data is created for a representative data set of Web Images.

Chapter 6

Connected Component Classification

Having segmented an image into regions, the next step towards text extraction is to characterize each component in terms of its probability to be either a character or part of the background. This chapter describes two different approaches taken towards identifying character-like components and compares the two in terms of their efficiency in dealing with real data.

The difference between the two approaches lies in the scale at which they work. The first approach works at the character scale, trying to classify the components according to certain features they exhibit. The second approach, works at the scale of a text line, examining similarities and differences between components aiming to identify components that could consist a line of text.

As will be seen next, the number and the variety in shape and size of the components resulting from the segmentation process does not allow any feature-based classification of individual components to be used.

6.1. Feature Based Classification

One way to decide whether a given component represents a character or not, is by checking certain features of the component that would qualify it as such. Such features can be the component's width and height, its aspect ratio, its size etc. We will examine a number of character features, and evaluate their applicability in separating character-like components from components representing parts of the background in the particular case of Web images.

6.1.1. Features of Characters

The first features to consider would be the *width* and the *height* of characters. Width and height are typically defined as the width and the height of the bounding box of a given character. Text might appear in any size in a Web image, therefore, a decision about whether a component represents a character or not cannot be based solely on its width and height. Another issue associated with width and height measurements is that they depend on the orientation of characters; for example, the width of a character would become its height if the character was rendered at an angle of *90º*. Since the orientation of text is not known *a priori*, no direct relation can be assumed between the measured width and height of a component and the expected values for a character, even if the character size was known beforehand. A feature that is slightly less dependent on the rotation of the character is its diagonal (the diagonal of its bounding box). Still, the diagonal depends on the size of the character, so it suffers from the same disadvantages as the width and the height.

A character feature, which can be derived from the width and the height, is the aspect ratio:

$$AspectRatio = \begin{cases} \dfrac{Width}{Height}, & if\ Height > Width \\ \dfrac{Height}{Width}, & if\ Width \geq Height \end{cases} \qquad \text{Eq. 6-1}$$

The aspect ratio is defined as the ratio of a character's width to its height or vice versa, and is a measure of the eccentricity of the character. Generally, with the exception of characters like "*i*" or "*l*", characters are rather square, with an aspect ratio near *1*. A component with such an aspect ratio would have a higher probability to be a character. Aspect ratio is independent of scaling, meaning that irrespective of the size of text, a given character is expected to have constant aspect ratio. The aspect ratio of each of the *26* letters of the English alphabet was measured using various fonts, and sizes ranging from *6pt* to *22pt*. The results for the Arial typeface, normal weight, non-italic characters are shown in Figure 6-1. The results for other fonts give similar results. Although the majority of characters appear to have an aspect ratio between the values of *0.5* and *0.9*, a significant number of characters appear outside this range.

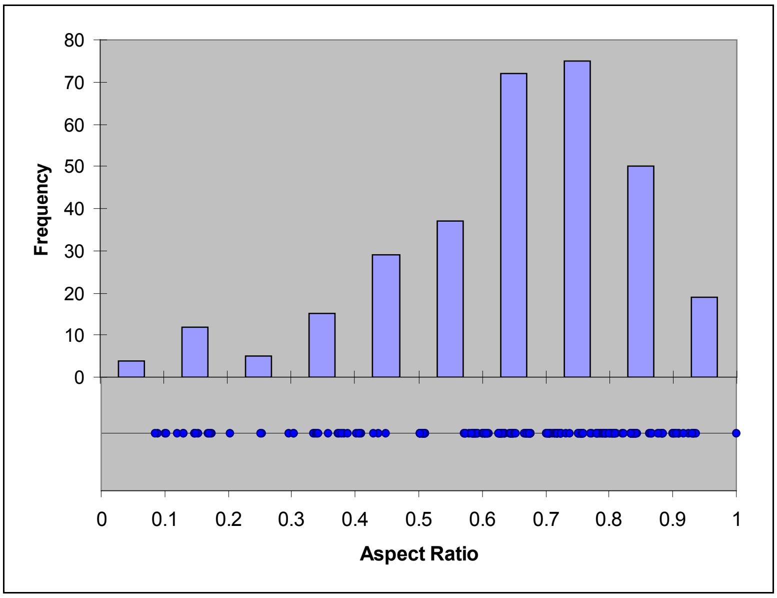


Figure 6-1 – Histogram and scatter plot of aspect ratio for the 26 letter of the English alphabet. The font used for the measurements is normal weight, non-italics Arial. Aspect ratio was measured for six sizes, ranging from 6 to 22pt. As can be seen most characters have an aspect ratio between 0.5 and 0.9.

This is only one side of the problem, since in order to decide whether a feature has any discrimination capability, we have to measure real data, and check whether the background components have adequately different distribution of aspect ratios. Such a measurement of components, obtained from real segmentation data over a number of Web images is shown in Figure 6-2. As can be seen here, although most of the background components have aspect ratios outside the range of *0.5-0.9*, a significant number of background components (in comparison to the number of character-like ones) fall inside this range. No decision boundary can be positively identified in the scatter plot that separates successfully the two classes. This renders the feature of aspect ratio ineffective in terms of identifying character-like components, at least if used alone.

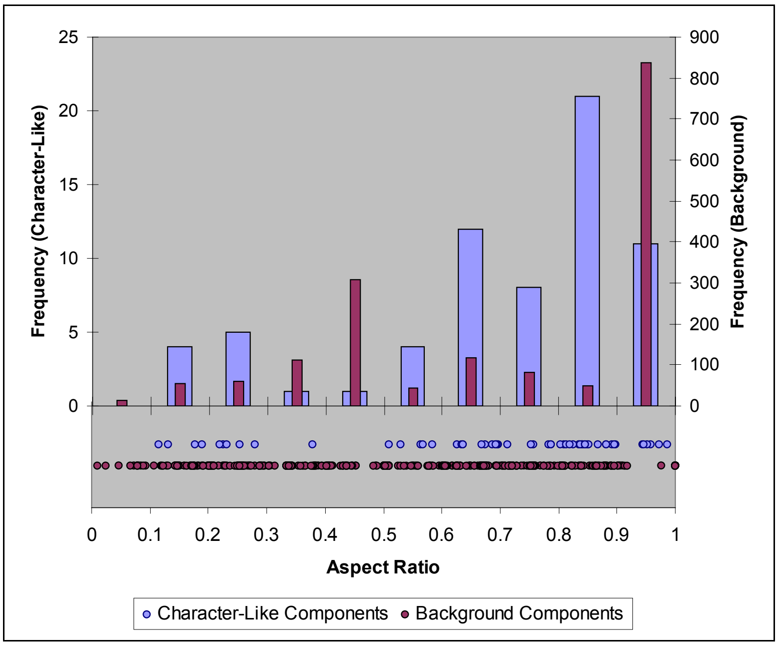


Figure 6-2 – Histogram and scatter plot of aspect ratio for real components. The Y-scale for the background components is shown on the right.

A prominent feature of characters is their size. One way to define character size is as the area of its Bounding Box, that is the product of the width and the height of the character. Measuring the area of the bounding box of a character is computationally inexpensive; nevertheless, it is strongly dependent on the orientation of the character. A better definition for the size of a character would be its *pixel size*. This refers to the actual number of pixels of the character. Computing the pixel size can be a computationally expensive process comparing to computing the area of the bounding box, still the pixel size is independent of the orientation of a character (although it can change slightly due to differences in digitisation in various rotations). Unless certain information about the size of text in the images is available beforehand or can somehow be derived, character size on its own is inadequate. Nevertheless, size can prove quite useful when used in conjunction with other attributes, that depend on it, for example different ranges of aspect ratio might be expected for characters of different size.

A feature directly derived from the above definitions, is the *compactness* of a character. The compactness of a character can be defined as the ratio of the pixel size to the bounding box size of a component:

$$Compactness = \frac{PixelSize}{BoundingBoxSize} \qquad \text{Eq. 6-2}$$

and is a measure of how full is a specific area by pixels of the character. In general, characters have a compactness value in the middle range, since they consist of a number of strokes, and therefore infrequently occupy much of the area defined by their bounding box. Certain characters, such as "*l*" or "*i*" present a much higher compactness, but these can be filtered out by the use of their aspect ratio as will be seen next.

The compactness of a number of ideal characters was measured, and the results are shown in Figure 6-3. As expected, most of the characters present a compactness value in the middle range. A number of characters such as "*i*" and "*l*" have a compactness value equal to one, which results to a peak on the right side of the histogram.

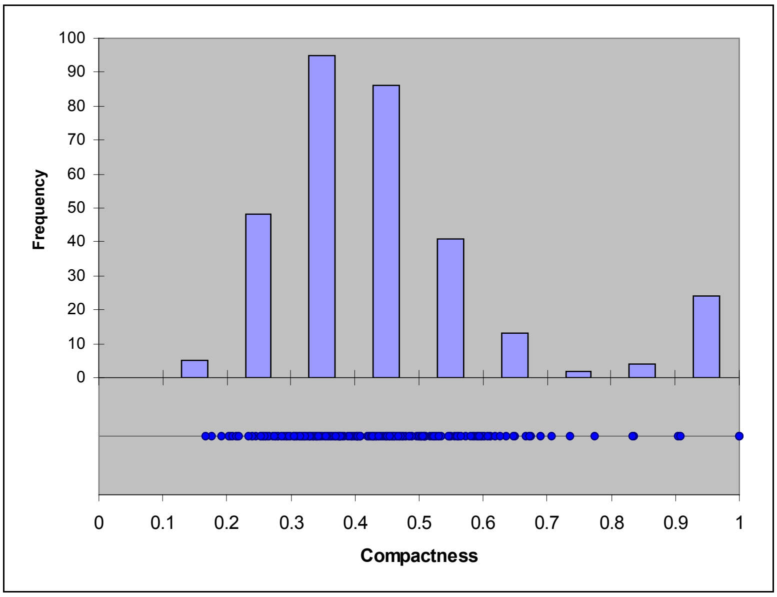


Figure 6-3 - Histogram and scatter plot of compactness for the 26 letter of the English alphabet. The font used for the measurements is normal weight, non-italics Arial. Compactness was measured for six sizes, ranging from 6 to 22pt. As can be seen most characters have an compactness value between 0.2 and 0.7.

Real data for compactness are illustrated in Figure 6-4. Components resulted by the segmentation methods discussed in this thesis were manually classified as character or background, and their compactness was measured for a number of images. The distribution of both character and background components is very similar to the distribution of Figure 6-3 for ideal characters. Most of the background components though, seem to have a compactness value near *1*.

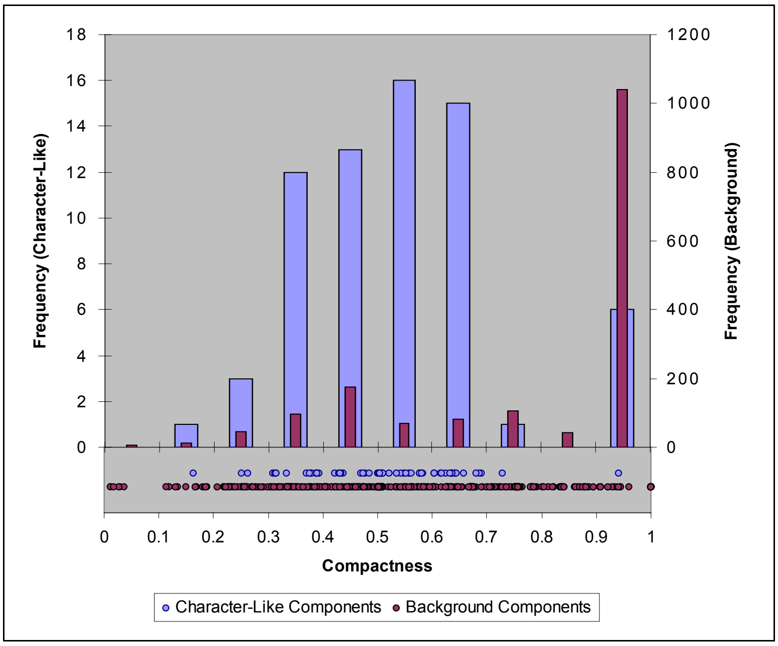


Figure 6-4 - Histogram and scatter plot of compactness for real components. The Y-scale for the background components is shown on the right.

Generally, the background components are many more than the character ones (roughly their ratio is *25:1*). Because of that, even if all the character-like components fall in a small range of values, for example between *0.2* and *0.7* regarding their compactness, the number of background components in that range is usually comparable, even bigger than the character-like ones.

Another feature that can be defined for characters (and components) is the number of the Black-to-White (or White-to-Black) transitions. If a character is digitised, and each point receives a value: *1* if it belongs to the character and *0* if not, then we can trace each scan-line of the digitised character and count the transitions from character points to background points. This number is defined as the number of transitions for the character, and is a measure of the complication of a character. Since characters consist of strokes, their number of transitions will be higher than of solid background components. Generally, the number of transitions roughly correlates with compactness, and most characters present a number of transitions at the middle range.

6.1.2. Multi-Dimensional Feature Spaces

Using just one feature to separate the two classes of components is not an option here due to the considerable variety of background components as explained before. Combinations of two and three features were tried in an attempt to see if a decision boundary could be defined in a *2D* or *3D* feature space. It is important that the features combined are as little correlated as possible, so that the best separation of the two classes is obtained.

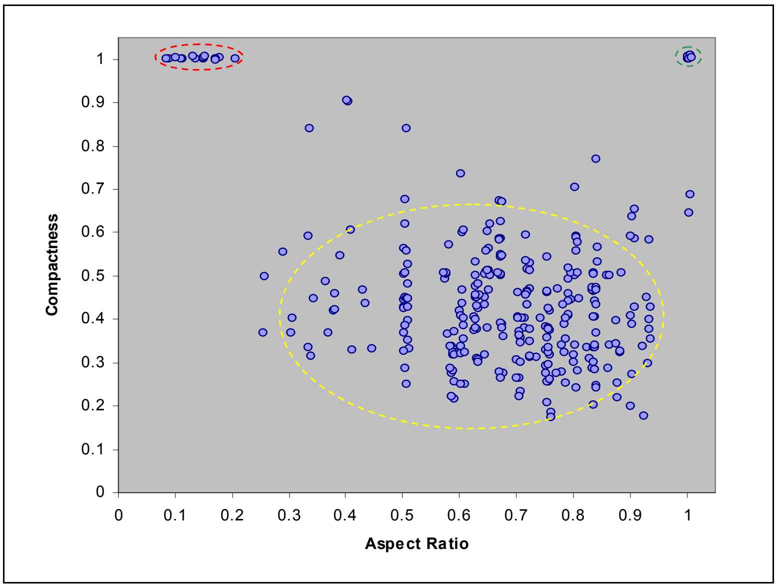


Figure 6-5 – Scatter plot of Ideal Character data in the 2D feature space of Aspect Ratio and Compactness. Areas of interest are marked.

The feature space of Aspect Ratio and Compactness was one of the 2D spaces evaluated. A scatter plot of ideal character data in the feature space is shown in Figure 6-5. As expected, most data are concentrated in the centre (yellow) marked area. Elongated characters such as "*l*" and "*i*" present significant compactness and low aspect ratio, so they are concentrated in the area (red) at the top left of the scatter plot. The third area, marked with green, at the right top of the scatter plot might appear at first mysterious, since there are no characters that are both rectangular (aspect ratio near *1*) and filled (compactness near *1*). This third set of points actually comes from the dots over characters such as "*i*" and "*j*", which are both compact and rectangular.

The way data for this plot was obtained, was to create connected components from characters and measure the features of all the connected components produced. Consequently, characters such as "*i*" and "*j*" were split in two connected components. This small cluster can be safely ignored for this process, since the dot over certain characters can be identified at a later stage, provided that the main part of the character is found.

Although a good separation is difficult to achieve, K-means clustering was performed with the ideal character data in order to help define the main two clusters of interest, namely the one containing the points in the yellow area and the ones in the red area as marked in the original plot (Figure 6-5). Clustering with *K=2*, *3* and *4* was tried and the areas of interest were better defined for *K*=3 as shown Figure 6-6(b).

The next step is to test whether real data fit into this model and can be separated using the clusters found. A scatter plot of real data collected from a number of images is shown in Figure 6-7 (page 186). As can be seen here, the vast majority of character-like components follow the expected distribution, that is they are concentrated in a middle area (aspect ratio around *0.8* and compactness around *0.5*) and also at the left top part of the scatter plot (aspect ratio near *0.1* and compactness near *1*). Unfortunately, these exact areas are also dense in background components, therefore a good separation can not be obtained. For illustrative purposes, the Voronoi diagram based on the centres of the three clusters identified before for ideal characters is also shown in the figure.

The feature space of Aspect Ratio and Compactness is one of a number of combinations of features tried. Unfortunately, no combination of features was able to give a good separation of the two classes. *3D* feature spaces were also examined, for example the space defined by Aspect Ratio, Compactness and Diagonal is shown in Figure 6-8 (page 186).

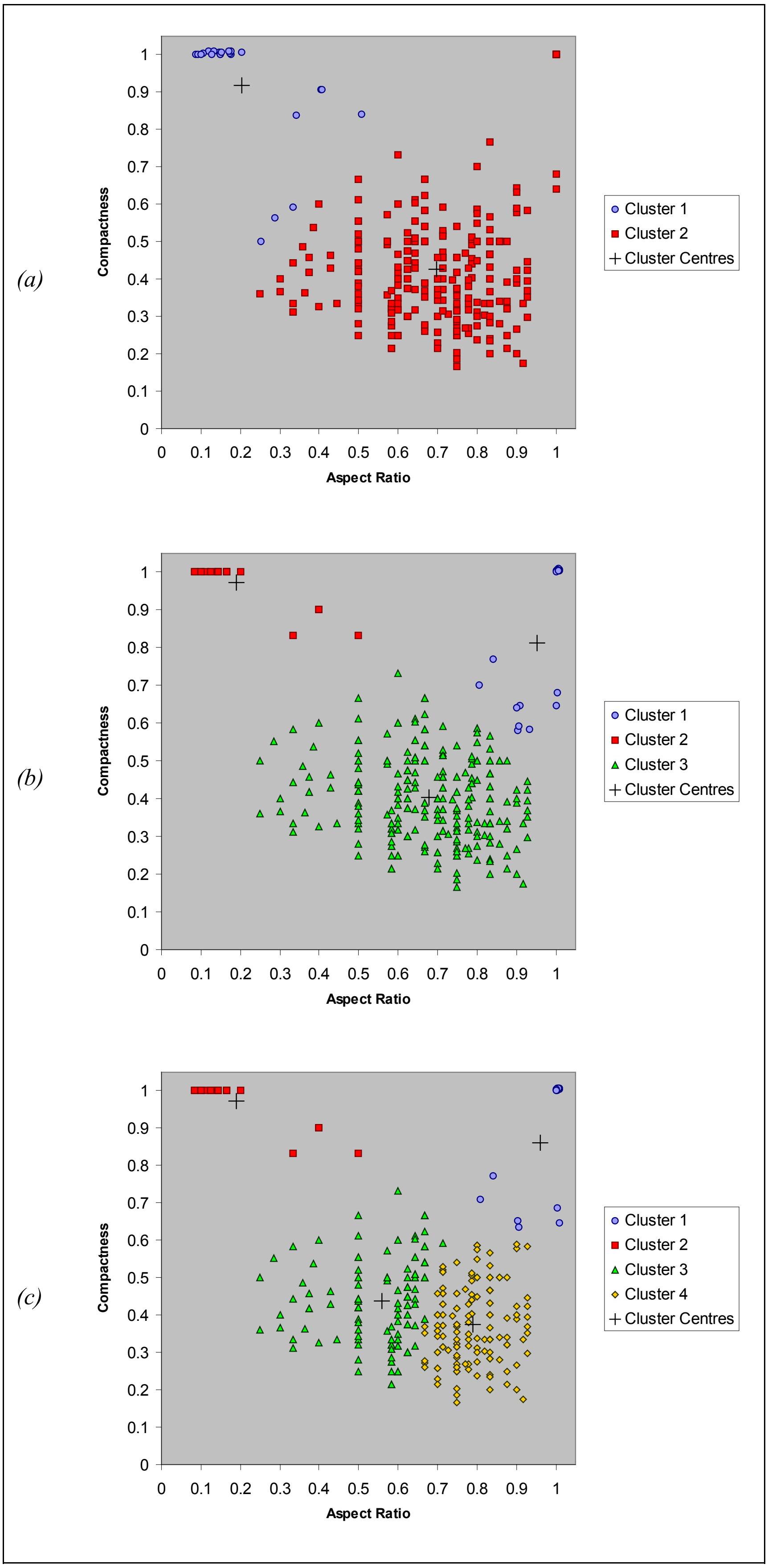


Figure 6-6 – Results of K-means clustering for K=2, K=3 and K=4. The best results were obtained in (b) for K=3.

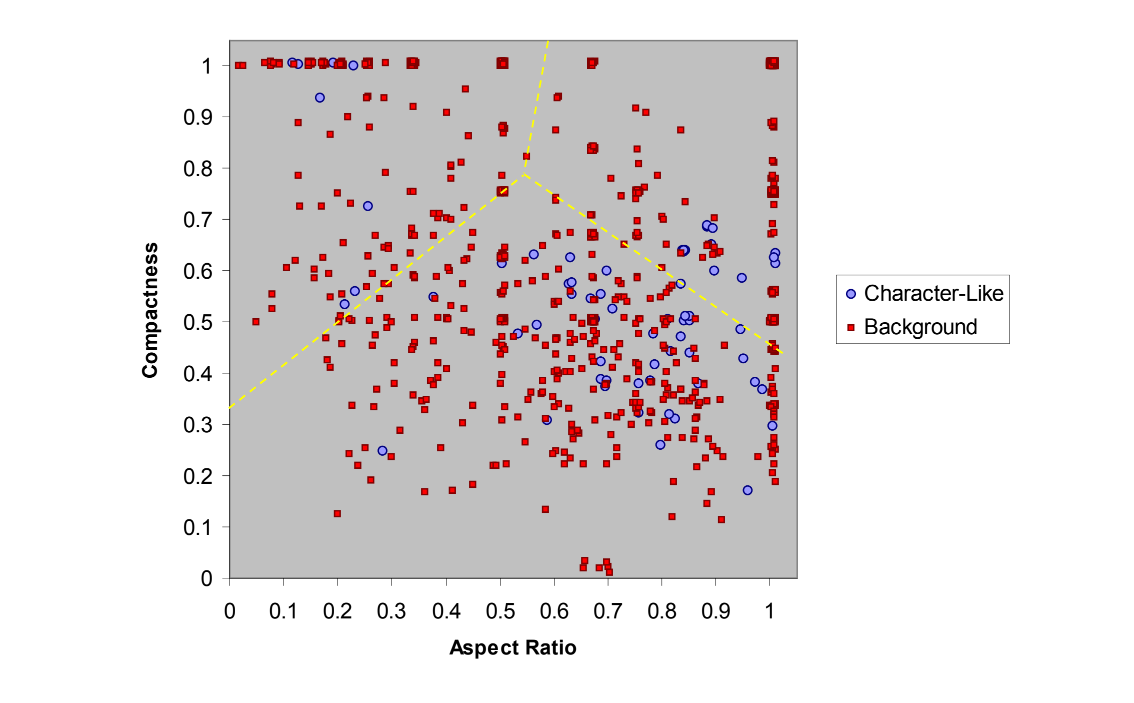


Figure 6-7 – Scatter plot of data of real components obtained by the segmentation methods. Although the distribution of character-like components follows the expected pattern, the vast amount and variability of background components hinders any effort to separate the two classes. The Voronoi diagram based on the centres of the clusters identified before is also shown (yellow lines).

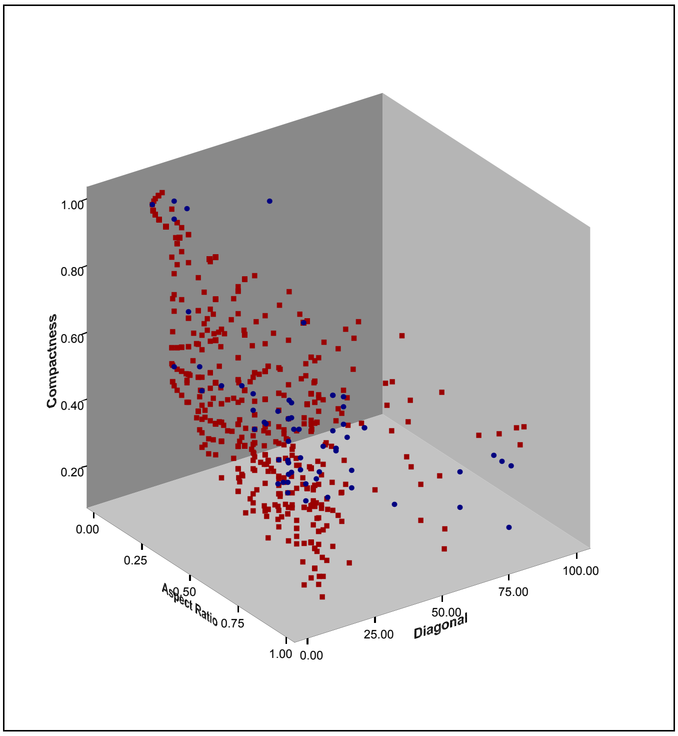


Figure 6-8 – 3D scatter plot of real data in the feature space defined by Aspect Ratio, Compactness and Diagonal.

In conclusion, due to the vast amount and the great variety of background components produced by the segmentation methods, it was decided that characterization of the components based on their distinctive features only, couldn't produce satisfactory results. Therefore, the problem of component characterization is addressed at a different level, as will be described next.

6.2. Text Line Identification

At a more macroscopic scale, when looking at the whole set of characters in an image, we usually expect them to share some common characteristics. The size of characters is probably the first such characteristic. Indeed, in the majority of cases, we expect the characters in a paragraph, or at least in a text line to have similar size. In the case of Web images, we do not expect to find whole paragraphs of text, still the assumption that characters of the same line of text have similar size stands true in the majority of cases. There are limited cases where even characters of the same word are of different size, or certain characters of a word have been replaced by other shapes for the sake of beautification, but such cases can be considered as rare.

This second attempt to separate character-like components from background ones exploits exactly the fact that characters belonging to the same line of text should share some common characteristics. Implicitly, this statement undertakes a certain assumption that the characters in a given Web Image are placed on a straight line, which is not always the case. As will be explained in the next sections, there is a certain trade-off between the number of characters required for the method to positively identify a line, and the degree of curvature a line can have.

The method starts by grouping components based on their size and tries to identify straight lines of connected components in each group. For each line identified, an assessment process follows, which indicates whether the given line is a valid text line or not. The steps of the method are detailed next.

6.2.1. Grouping of Characters

The first step towards identifying lines of characters is to group the connected components resulting from a segmentation method according to their size. Two are the prominent issues here: First, decide which metric of size we should use for the size grouping and then, given an average size value, what range of sizes should be considered acceptable for a component to have to belong in this size group.

If the characters were placed on a straight horizontal line, then the most appropriate size metric to use would be their height. This is because the height of characters of the same font varies much less than their width. Essentially, the variation in height would be from as short as the height of a typical lower case character (a metric sometimes called "*mean height*" or "*x height*"), to as long as the height of a capital letter, a descending or an ascending character. The variation of width on the other hand is much higher, from narrow letters like "*i*" to wide ones like "*w*".

In reality though, characters are not always placed on a horizontal line, so selecting the height attribute as representative of their font size is a rather unfounded decision. As mentioned before, a metric of size that is much less dependent on rotation is the diagonal of the bounding box of components. The diagonal is used here as representative of the size of components, in order to assign them to different size groups.

The second problem associated with the size-grouping of connected components is defining the range of sizes that defines each size-group. In other words, given an average size for a group, what are the size thresholds for a component to be said to have similar size to the group.

The minimum and maximum diagonals were measured for different fonts (Arial and Times typefaces, normal, bold and italics) and various sizes (from *6* to *36pt*). For each case the following equations were solved, where D_{max} and D_{min} is the maximum and minimum diagonal values, D_{av} is the average diagonal value for the size group and f is a factor defining the maximum allowed ratio of D_{max} to D_{av} and D_{av} to D_{min}.

$$D_{\max} = D_{av} \cdot f$$
$$D_{\min} = \frac{D_{av}}{f} \qquad \text{Eq. 6-3}$$

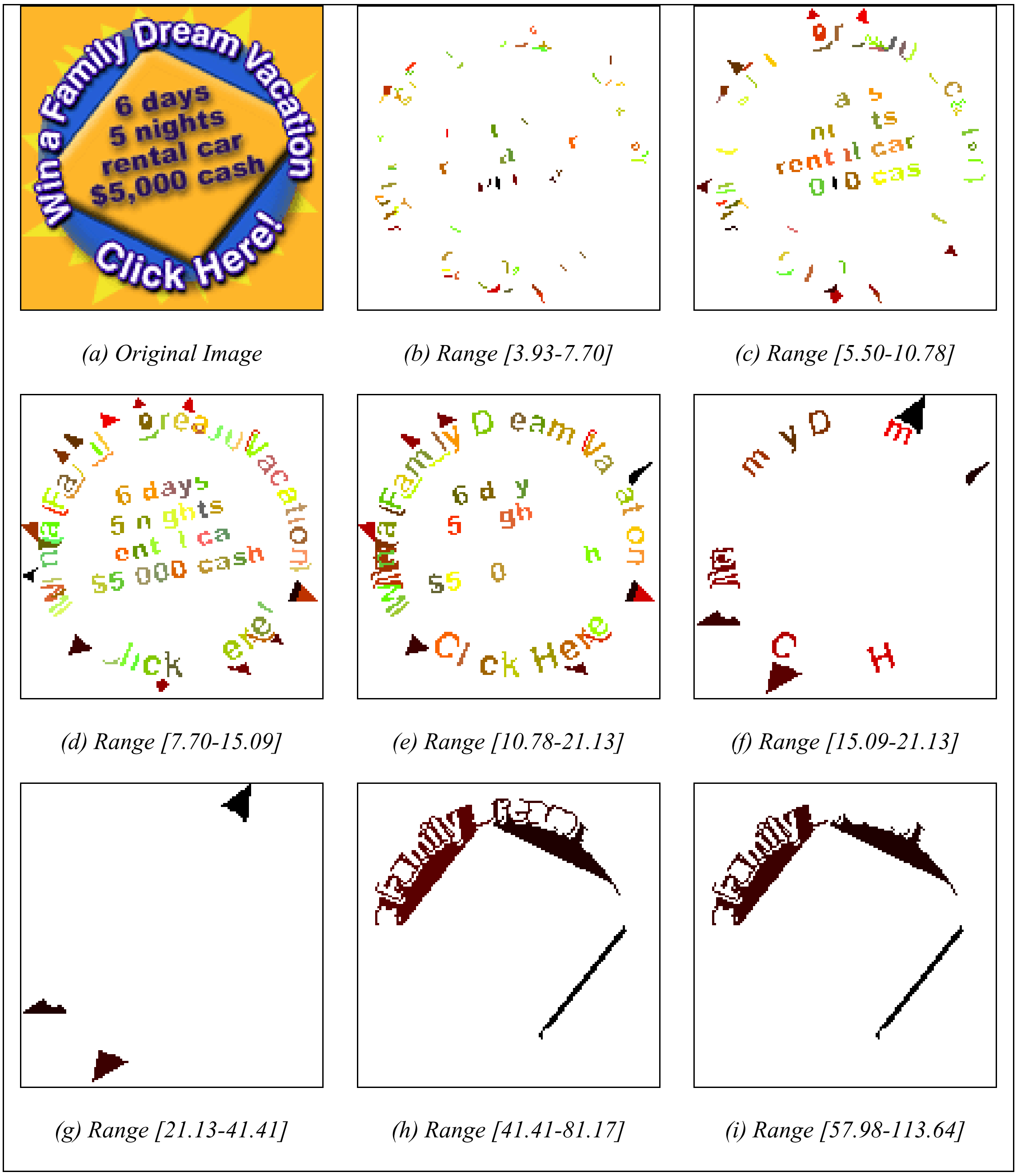
The average value obtained for f is *1.46*, subsequently for any given diagonal (D_{av}) the maximum and minimum diagonal values allowed are given by Eq. 6-3.

The grouping process creates a number of size-groups. The first group (smallest diagonal sizes) has an average diagonal size of *5.5*, meaning that components with diagonals from *~4* (*=5.5/f*) to *~7.7* (*=5.5·f*) pixels will be assigned to this group. The minimum diagonal size of around *4* pixels agrees to our requirement of a dimension (either width or height) greater than *4* pixels for a character to be classified as *readable*. Readable and non-readable characters are defined in Chapter 7, to facilitate the evaluation of the method. A minimum diagonal size of *4* ensures that even the non-readable characters will be considered. Anything with diagonal size smaller than *4* pixels is not considered as a candidate for a character-like component.

The average diagonal size for each size-group after the first one is defined as the maximum diagonal size of the previous group. For example, the second size-group would have an average diagonal of *7.7*. The maximum size-group allowed cannot have a maximum diagonal greater than the smaller dimension (width or height) of the image.

For each size-group, all the connected components of the final segmentation are considered, and if they have a diagonal in the range of the size-group, they are assigned to it. If a group is assigned less than three components at the end of the process, it is discarded. It should be made clear at this point, that since the ranges of consequent size-groups overlap, it is expected that each component can be assigned to up to two size-groups.

An example of an image and the components assigned to each size-group are shown in Figure 6-9. As can be seen, a good separation is obtained here.



(a) Original Image

(b) Range [3.93-7.70]

(c) Range [5.50-10.78]

(d) Range [7.70-15.09]

(e) Range [10.78-21.13]

(f) Range [15.09-21.13]

(g) Range [21.13-41.41]

(h) Range [41.41-81.17]

(i) Range [57.98-113.64]

Figure 6-9 – Original image (a) and the components of the size groups obtained (b-i).

6.2.2. Identification of Collinear Components

The process following size classification examines the co-linearity of components in each size-group and identifies lines of components. The way this is done is by performing a Hough transform of the centres of gravity of the components in each size-group. Certain aspects of this process will be discussed next.

A prominent issue preceding the Hough transform is to express each connected component with a single point in the (image) Cartesian space. If the text was written in a known orientation, it would make sense to try to identify the baseline of the text, so for each component an appropriate point would be chosen. Since the orientation of text is not known a-priori, the centre of gravity of each component is used instead. The use of the centre of gravity ensures that within certain tolerances a straight line between character components would be identifiable in any orientation.

A Hough transform is performed at this point. The parameter space used for the Hough transform is the θ-ρ space. The normal parameterisation is used to describe a line, where θ is the angle of the line's normal and ρ is the algebraic distance of the line from the origin. The points participating in the transform are defined in the coordinate space of the image, so lines will be described by their parameters θ and ρ as can be seen in Figure 6-10. The parameter θ is restricted to the interval $[0, \pi]$ and the quantisation step was set to 3 degrees. Similarly, the parameter ρ is restricted to the interval $[-D, D]$ where D is the diagonal of the image and the quantization step was set to half of the maximum component diagonal allowed for the given size-group. For example, the quantization step for ρ for a size-group containing components with a diagonal in the range $[3.93, 7.70]$ would be $7.70/2$.

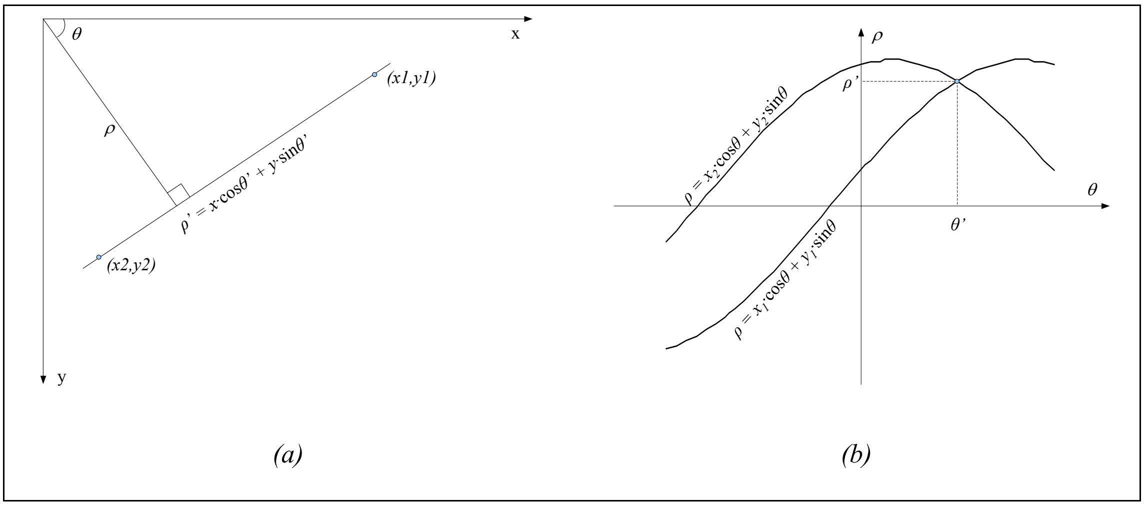


Figure 6-10 – (a) Image space. The origin is positioned at the top left corner. A line can be defined (θ', ρ') that crosses any two given points. (b) Parametric space. The point (θ', ρ') that defines the line is the crossing point curves defined by the original two points.

The accumulator array is examined after the Hough transform has taken place, and the cell (or cells) with the maximum count is identified. A possible text line is identified, having the parameters of the cell, and the associated components are recorded as part of the line. The components associated with the exported line are removed from the array of the components of this size-group, and the same process is repeated with the rest of the points each time identifying the cell (and corresponding line) with the maximum count. The process stops when no cell exists with a count of more than three. The pseudo-code for the above process is given in Figure 6-11.

```
For Each SizeGroup
{
  While (Number of Components in SizeGroup >= 3)
  {
    For Each Component in the SizeGroup
    {
       Find the Centre of Gravity and add it to the ListOfPoints
    }
    Perform Hough Transform at the ListOfPoints
    Find the MaximumCount in the accumulator Cells
    If MaximumCount < 3 Then Continue with next SizeGroup
    For Each Cell
    {
      If Count equals MaximumCount Then
      {
        Identify the components falling in the Cell as a line
        Remove components from SizeGroup
      }
    }
  }
}
```

Figure 6-11 – Pseudo-code for the line identification process.

A point worth mentioning is that if more than one cells present a count equal to the maximum count, no special selection occurs. Instead, all candidate lines are identified, covering all possibilities. By choosing the appropriate accumulator cell dimensions (quantization) for the Hough transforms, such a situation can be limited. This is because most of these cases occur as the same components identified as slightly rotated or parallel moved lines (neighbouring accumulator cells).

As mentioned before, three is the minimum number of components requested for a line to be identified. The rationale behind this decision is manyfold. First, geometrically at least three points are needed to be able to assess the co-linearity between them. The main argument against using only three points, would be that statistically three points would be just enough to give an indication, but not to define with certainty that there is a line there. A main reason why three is considered enough is because the points represent components comprising words, and words can have three or even less letters. Subsequently, if a small word were there alone in a text line (which happens very often), we would miss it if a higher threshold were chosen. The second reason for selecting a low threshold is that we need to be able to address cases where text is not actually written on straight baselines. By setting a low threshold, even if a word is written along a curve, straight lines of three characters would be relatively easy to identify, whereas larger chunks of characters would not be collinear anymore. An example of this situation is shown in Figure 6-12.

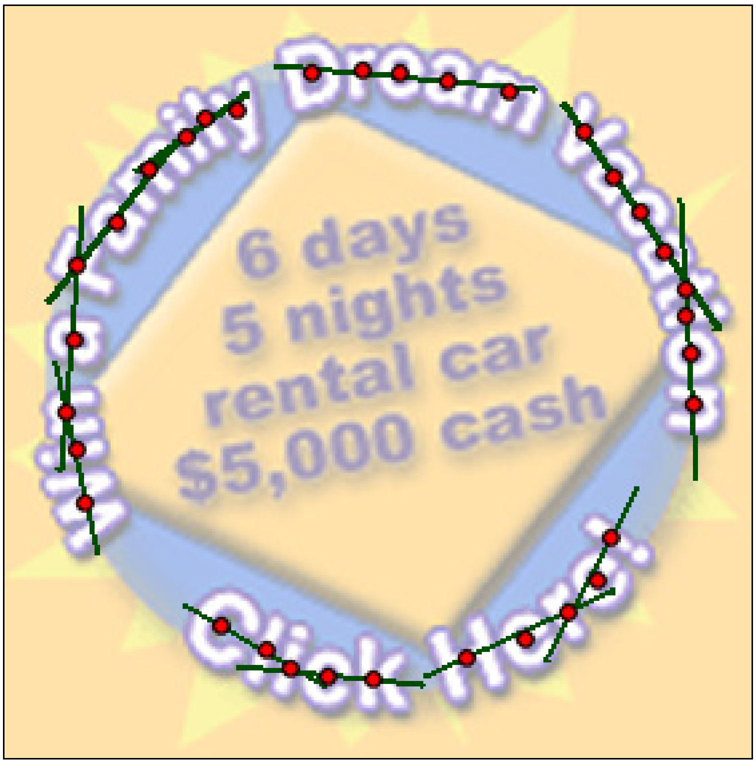


Figure 6-12 – Straight lines fitted on circularly written text. The centres of gravity of the character components are shown in red and the lines fitted in dark grey. As can be seen, many of the lines comprise a small number of components because of the way text is written.

Two facts should be kept in mind. First, the lines actually exported usually comprise more than three components; for example, if a long straight text line actually exists in an image, its components will probably be extracted in one straight line by the Hough transform, rather than broken in smaller chunks. Second, it is a fact that for lines comprising of a small number of components, any statistical operation such as the assessment process discussed in the next section, will be rather ambiguous and should be treated as such.

6.2.3. Assessment of Lines

Two mechanisms were devised for assessing the lines extracted by the previous operations. The first mechanism examines the distances between successive components in the line, and produces a higher confidence value if the components have equal distances between them. The second mechanism uses the parameter information of the line to calculate the projection of the components on the direction of the line, and examines whether the histogram produced has similarities to the histogram expected by characters. These two assessment mechanisms will be discussed next.

Distance between Components

Given the parameters of a line produced by the Hough transform and a set of associated points, we can easily find the projections of the points on the line (Figure 6-13). For each of the lines identified, the projections of the centres of gravity of the components participating are computed. For each pair of successive components the distance between their projections is calculated. For *n+1* components, we would have *n* distances computed. For *n+1* equal to three (only two distances can be computed), no statistical analysis of the distances can take place and lines are solely assessed by the second assessment mechanism, described in the next section.

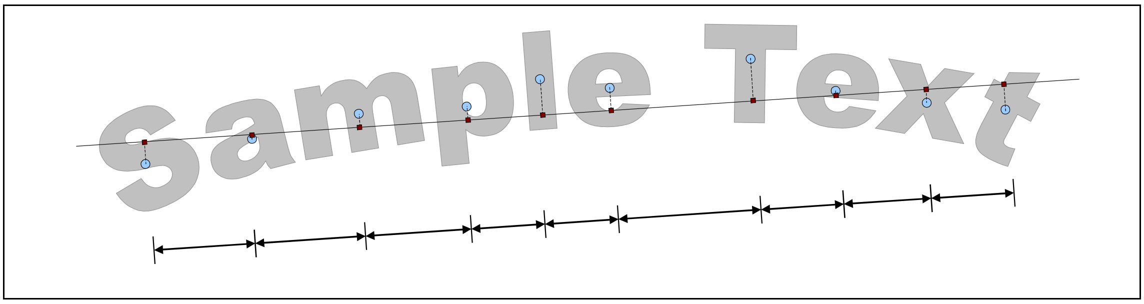


Figure 6-13 – The projection of the centre of gravity of each component is obtained and the distances are computed between successive projection points.

Successive characters in a text line are arranged in relatively equal distances. Subsequently, if the variance of the distances computed between components is comparatively small, then the line has a higher probability to be a text line.

There are many statistical distribution moments defined characterizing the variability of the distribution around its mean value. The most common is probably the variance[2], which is defined in Eq. 6-4 and the standard deviation defined in Eq. 6-5. There is also a computationally inexpensive estimator called the *average deviation* or *mean absolute deviation* defined in Eq. 6-6. For our purpose, the computational advantage of using the average deviation is minimal. The standard deviation is used here, as it gives a metric that can be directly compared to the actual units of the image.

$$Var(x_1 ... x_n) = \frac{1}{n-1}\sum_{j=1}^{n}(x_j - \overline{x})^2 \qquad \textit{Eq. 6-4}$$

$$\sigma(x_1 ... x_n) = \sqrt{Var(x_1 ... x_n)} \qquad \textit{Eq. 6-5}$$

$$ADev(x_1 ... x_n) = \frac{1}{n}\sum_{j=1}^{n}\left|x_j - \overline{x}\right| \qquad \textit{Eq. 6-6}$$

The *Standard Deviation* is a metric of the typical deviation from the mean of the values of the distribution. The value of Standard Deviation is given in image units, since the values x_i are expressed in image units. In order to associate this information to the specific size range of the components of the line, the ratio of the Standard Deviation to the Mean of the values (Eq. 6-7) was used.

$$\sigma_m(x_1 ... x_n) = \frac{\sigma(x_1 ... x_n)}{\overline{x}} \qquad \textit{Eq. 6-7}$$

[2] There is a lot of discussion about whether the denominator of Eq. 6-4 should be *n* or *n-1*. Briefly, the denominator should be *n* if the mean value of the distribution is known beforehand (it is unbiased), but it should be changed to *n-1* if the mean value is computed from the data (thus it is biased). In reality, if the difference between *n* and *n-1* actually matters, then the data set is not big enough, and we are probably trying to substantiate a questionable hypothesis with marginal data. Unfortunately, this is frequently true for our case where lines contain only a few characters each.

After experimenting with real data, it was derived that text lines usually have an σ_m value below *1.00* and lines of background components have a value above *1.35*. Lines presenting a value in the intermediate range usually comprise components of both classes. Based on the above, a line passes this first test if it has a value of σ_m less than *1.00*.

A scoring scheme was devised for the lines, according to which, a line is awarded *3* points if it has a value of σ_m less than *1.00* and *1* point if it falls in the intermediate range. The score of a line can be interpreted as a confidence value for it to be a text line, and it is also used to characterize individual components at the end of the assessment process as will be explained in Section 6.3.

The above metric gives a value for the typical deviation relative to the mean value of the distribution. Nevertheless, there is no association yet to the size of the components that participate in the line. The components of the line might be placed in equal intervals (successive components would have equal distances), but if the mean distance between successive components is not comparable to the size of the components themselves, it's probably not a text line. This is illustrated in Figure 6-14 below.

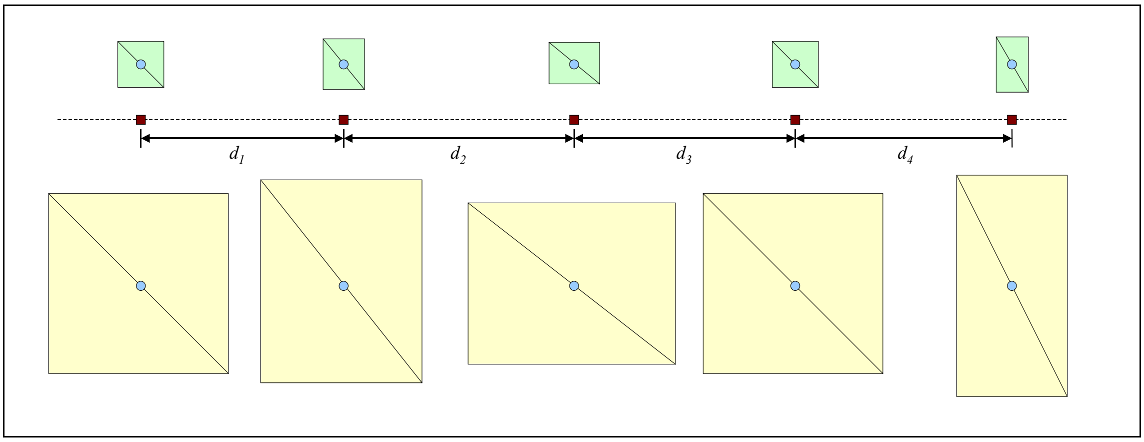


Figure 6-14 – Two lines of components, each comprising components of a different size-group. Although the distances between the components are the same in both cases, the components in yellow are more probable to be text, because the mean distance is comparable to the size of the components themselves. The other line comprises of smaller components, and the same mean distance is in this case rather large to assess that this is a text line.

Each size-group is defined by a minimum and maximum allowed value for the sizes of diagonals of the components it contains. Given this range for the diagonals of components, a line is said to present a mean distance between components comparable to the size of components if:

$$MinimumDiagonal \cdot 0.75 \leq \bar{x} \leq MaximumDiagonal \cdot 1.50 \qquad \textit{Eq. 6-8}$$

The values *0.75* and *1.50* were experimentally determined. The line passes this second test if the distance mean is comparable to the size of the components. Similarly to the previous scoring scheme, the line is awarded *3* points if the distance mean is comparable to the size of the components.

Based on the scoring scheme introduced, if a line produces a score above *5*, it is characterized as a text line and the assessment process finishes here. If it produces a score below *2*, it is characterized as a non-text line. Finally, if the line produces a score between *2* and *4*, then it is uncertain at this point whether the line comprises text or background components, so the second assessment mechanism, described next, is invoked.

Directional Projection

It is common in document segmentation methods, where black text is typed over white background, to obtain the horizontal and vertical projections of the image, in order to separate first lines of text, and then individual characters in the lines. The horizontal or vertical projection is nothing more than a histogram of the count of black pixels in each row or column of the image respectively.

The above technique can give quite good results if the orientation of the text is known (vertical and horizontal directions are defined according to the orientation of the text) and a statistically adequate number of characters exist in each line, enough to produce a meaningful histogram.

Having identified a number of possible text lines, thus knowing the orientation of text for each one, it is possible to calculate the horizontal projection of the components. Horizontal in this case, would be the direction of the line as shown in Figure 6-15. Vertical projections would not be of any use in this case, since the characters are already expressed as individual components.

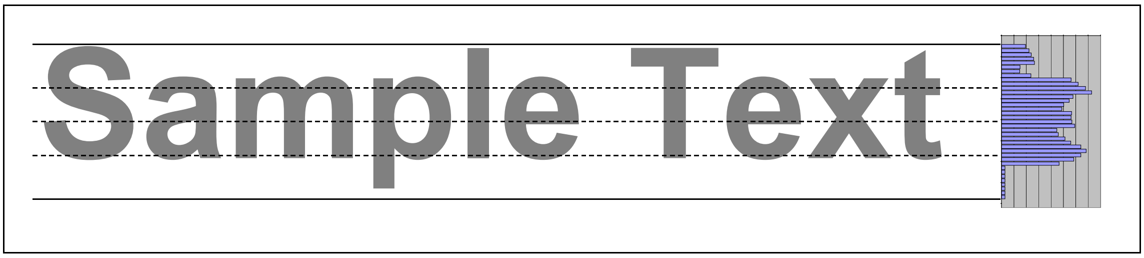


Figure 6-15 – Horizontal projection of a text line. Certain characteristics of text line projections such as higher peaks at the base line, mid-line and top-line, and trailing caused by descending characters are illustrated.

Assessing the similarity of the projections to the projection expected from ideal characters, a decision can be made as to whether the given line resembles a text line or not. Such an assessment involves a number of operations, such as obtaining the projections, normalizing the values and performing some kind of similarity check. These steps will be briefly described next.

Given the parameters of a line *(ρ, θ)*, and a point *(x, y)*, the vertical distance *d* between the point and the line is given by Eq. 6-9. The sign of the distance indicates whether the point is above of below the line.

$$d = \rho - \sqrt{x^2 + y^2} \cdot \cos\left(\theta - \arctan\left(\frac{y}{x}\right)\right) \qquad \textit{Eq. 6-9}$$

In terms of the identified lines of components, the distance of each pixel of the components assigned to the line can be computed. The maximum distance a pixel may have from the line can be calculated from known information about the quantisation of the accumulator array of the Hough transform (related to the size-group the line belongs to). The distance of each pixel of the components assigned to the line is calculated and the histogram of distances is obtained. This is the projection of the line's components on the direction of the line.

Ideal Character Projections

The projections of ideal images of long sentences were obtained and sampled. Various fonts were used (Arial and Times typefaces, various sizes). Furthermore, projections of all the small and all the capital letters of the English alphabet were separately obtained. Although this second type of projections would not be very relevant for paper documents, lines with solely capital and solely small letters frequently appear in Web images, therefore it was considered vital to examine the differences in their projections. A small sample of the projections obtained is shown in Figure 6-16.

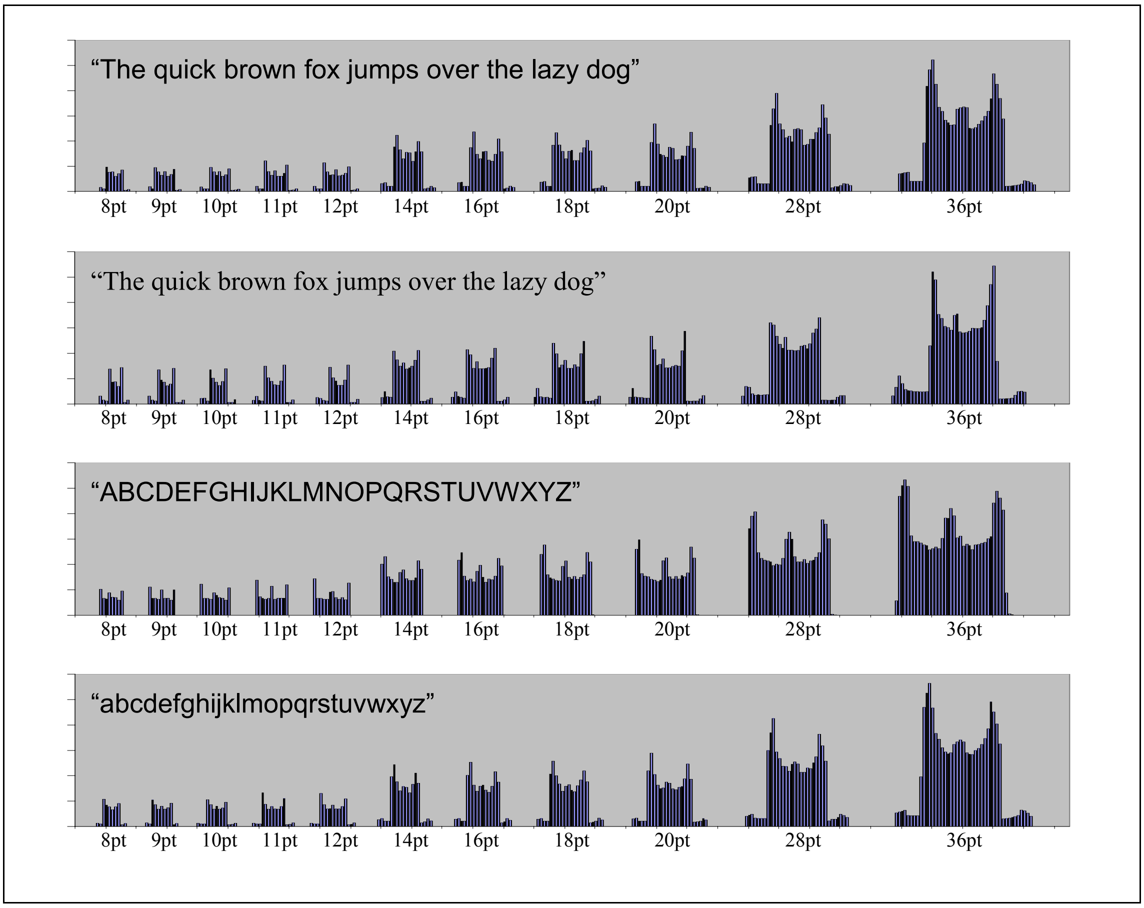


Figure 6-16 –Projections of ideal characters.

As can be seen in the figure, the projections have a well-defined distribution, with two peaks of almost equal strength one at the height of the baseline, and the other at the height of the top of the small characters. Trails are also visible on the left and right on the main body of each projection, which occur from the descending, ascending and capital characters. A visible peak also exists at the height of the strike-through line (the middle of lowercase characters).

When only capital letters are present in the text line, the left and right trails do not exist anymore. The two prominent peaks in this histogram are at the height of the baseline and the height of the top of the capital characters. The characteristics of this histogram, as well as the shifted positions of the dominant peaks indicate that a different approach should be taken if such a line is encountered.

Comparison of Histograms

A straightforward way to compare two histograms is to subtract one from the other and measure the differences. This method is only meaningful if the histograms are very well-defined in terms of their ranges so that a correct matching between values can be obtained. Any slight shifting or scaling can produce misleading results. Due to its high sensitivity, this comparison method is not used here. Instead, a very basic rank-order comparison is employed.

The main characteristics of the ideal projections (both in the normal projections and the capital characters ones) are the two prominent peaks of almost equal strengths and a well-defined smaller peak between them. If a measured projection possesses these characteristic peaks, at the right positions, then it is considered a true match.

Before any comparison can be made, the histogram of the projection must be normalized in both axes. The normalization in the vertical axis is trivial: each value is divided by the maximum count in the histogram. The normalization in the horizontal axis is slightly more complicated.

The histogram initially obtained is centred at the given component line as shown in Figure 6-17. It is generally preferable to cut the left and right trails of zeros (red range in figure) otherwise the range of the measured histogram will not match the range of the ideal one. Although this is frequently good practice, in some cases it produces a shifted histogram. This can happen because text lines in Web images usually contain just a few characters. It is therefore possible that there will be no descending or no ascending characters in a line, causing the elimination of the left or right low count areas in the projection. Another approach examined is to keep the histogram centred at the given line and check the left and right distance from that point until the first zero count is reached. Then define the range of the histogram based on the larger of the two (green range in the histogram). After experimentation with real data, the first method was selected as it gives better results in most cases.

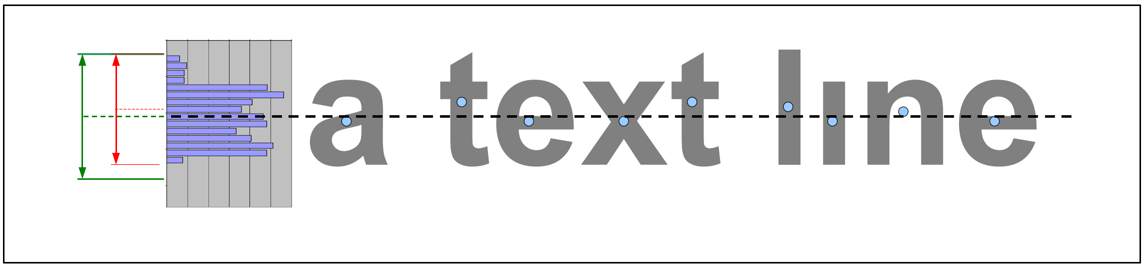


Figure 6-17 – Different ranges for normalizing in the horizontal axis of the projection. The sample text used here contains no descending characters. In order to account for their effect in the histogram range, the range illustrated in green should preferably be used.

Knowing the horizontal range, the normalization in the horizontal direction is now possible. If the centre is assigned a value of zero, and the range of values is normalized in [*-1, 1*], then the expected peaks should occur as follows. The two prominent peaks are expected in the intervals of [*-0.75, -0.25*] and [*0.3, 0.75*] while their values should be above *0.8*. The smaller peak should occur in the interval of [*-0.25, 0.3*] and have a value in the range [*0.5, 0.8*]. These values were determined from the projections of the ideal characters.

Real Projections

Well-defined lines of big and clear components, indeed produce projections very similar to the ones expected, and do not pose any problem in their identification. Unfortunately, a number of lines identified prove to produce significantly different projections than the ideal ones. There are many reasons why this happens, but mainly three factors play an important role: the size of the participating components, the presence of some background components in the line and the slightly curved lines.

When lines of small-sized components (diagonal less than *7-9* pixels) are examined, the resolution of the histogram turns to be too small for fine details to appear, and the characteristic peaks (if there) are not easily recognized. Statistically, both the number of bins of the histogram (horizontal axis) and the average count of pixels for each value (vertical axis) are very small to allow for a safe comparison. An example of such a line is shown in Figure 6-18. As can be seen the projection is only five pixels wide, therefore there is no way for the three prominent peaks to appear.

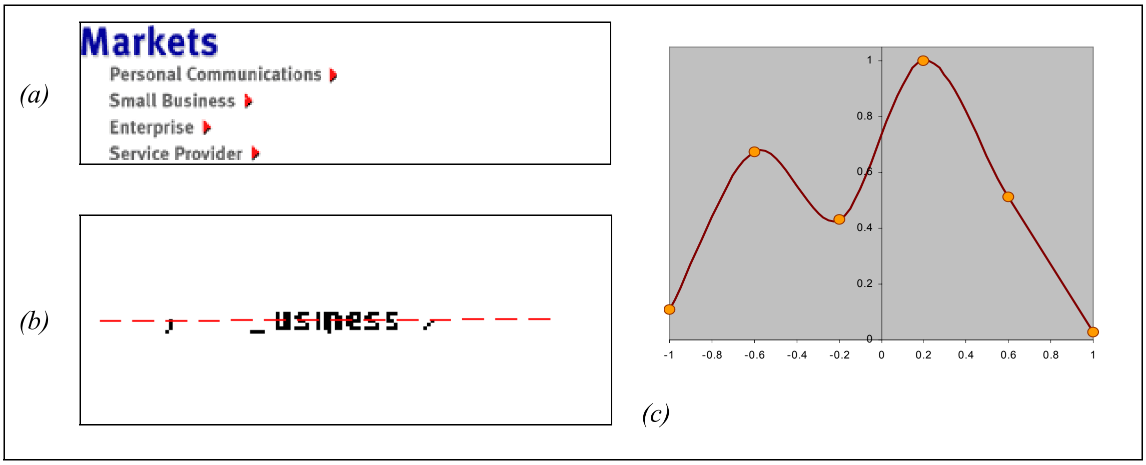


Figure 6-18 – (a) Original image. (b) Identified line of components (belonging to a small size-group). (c) Projection of the line. As can be seen, the projection is only five pixels wide, which does not allow the main characteristics to appear.

A second factor that plays an important role is the existence of curved lines. The projection of a curved line may be slightly shifted and/or horizontally stretched or shrunk. The positions of the characteristic peaks, if visible, are effectively shifted, while the peaks themselves do not have the expected strengths. An example of such a case, where the characteristic peaks do not have the expected strengths, is illustrated in Figure 6-19.

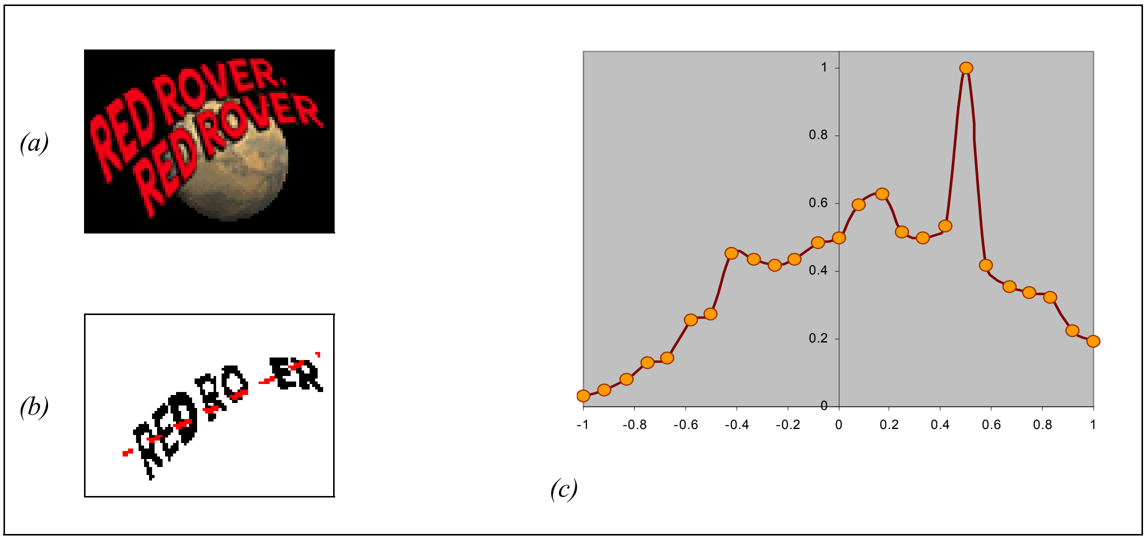


Figure 6-19 – (a) Original image. (b) Identified line of components, (components follow a curved line). (c) Projection of the line.

Finally, the presence of non-character components can introduce more peaks and change completely the expected profile of a projection. Although most of the non-character components should have been eliminated by this point (by size-grouping), there are still some components of similar size of the text that also fall on the same line with it, and are therefore accounted for in the calculation of the projection. Usually, such non-character components would be the inside areas of closed characters (such as the inside of an "*o*"), or shadows (that tend to have the same sizes with their associated characters). Nevertheless, the most difficult case is when irrelevant components of similar size just happen to fall on the direction of the line. An example is illustrated in Figure 6-20.

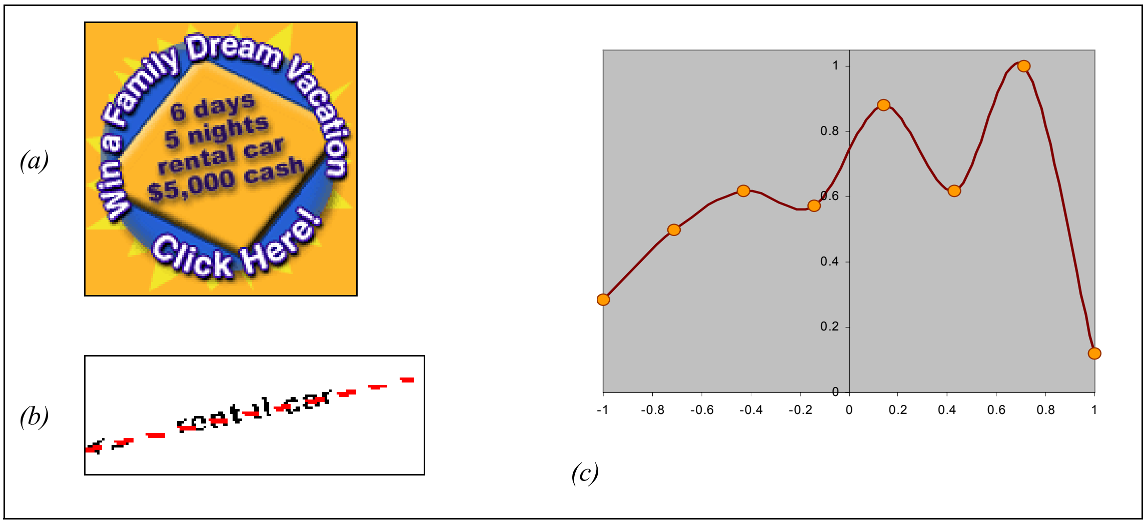


Figure 6-20 – (a) Original image. (b) Identified line of components (containing some non-character ones at the left end). (c) Projection of the line. As can be seen, there are a number of irrelevant peaks introduced in the histogram.

Although the three characteristic peaks are present in most of the cases (even if one or more of the above situations occur), they rarely appear to have the strengths expected. For example in Figure 6-19 and Figure 6-20 above, the three higher peaks are indeed in the positions expected, but they do not follow the identified pattern of two high peaks at the sides and a lower one between them.

To cover such cases, the "rank" of the peaks is not checked during this assessment mechanism, but only their position. The three higher peaks are identified and, if they have a maximum at the correct positions, an indication is given that the projection came from a text line.

To summarize, given a line projection, there are a number of different cases encountered. If the resolution of the histogram is very small (less than 7 bins), then nothing can be said for the line by examining its projection, and the score of the line remains unchanged. If the resolution of the histogram is adequate, and fewer than three peaks are present, it is considered an indication that the line does not contain characters. It is still uncertain whether this is caused by non-character components or by some other factor that would smooth the projection (a curved line for example), therefore, only *1* point is deduced by the score of the line. If adequate peaks can be identified and they are positioned in the expected places, then the line is considered to consist of character components, and is awarded *3* points. In every other case, the line is probably not a text line, and *2* points are deduced from its score.

Final Verdict

The final score of a line serves as a confidence value for whether the line consists of character-like components or not. The scoring scheme is devised in such a way so that a line with a score greater or equal to *5* is considered a text line, and a line with a score less or equal to *1* is not. For a line with a score between *2* and *4* a decision cannot be made.

The main assessment mechanism used is the one described in the beginning of this section, which examined the distances between subsequent components. The second mechanism, based on the projections of the lines identified, is only employed when the first assessment gives an uncertain verdict. The reason for restricting the use of this second mechanism is that it is sensitive to various factors as detailed before, whereas the first mechanism, even if simplistic, is not.

The output of this method can be either a list of the lines identified, along with the corresponding confidence values, or a list of the components themselves along with a confidence value for being characters. The confidence value of individual components is defined as the sum of the scores of the lines in which a given component was assigned, divided by the number of lines it was assigned to. Reporting the individual lines is generally better, since it also gives information about the orientation of the line.

6.3. Conclusion

In this chapter, two methods were examined to classify the connected components resulting from a segmentation process into two classes: text or background. The difference of the two methods is the scale at which the problem was addressed. Specifically, the first attempt tries to classify components looking into features of individual components, whereas the second attempt, works at a different scale, trying to identify lines of components that share some common features.

It proves that for the specific problem of Web Images, a classification method based on individual components is unable to provide good results. The second approach, based on identifying lines of text, works better in a variety of situations, and is therefore the one used here for classifying components after segmentation.

It should be mentioned at this point, that any character-like component identification process, is strongly dependent on a correct segmentation. Slightly inaccurately segmented characters, which is a rather frequent situation for Web images, may result in wrong assessments.

An evaluation of the connected component classification method based on text line identification will be presented in Chapter 7.

Chapter 7

Overall Results and Discussion

In this chapter, the results for the two segmentation methods (Split-and-Merge, Chapter 4 and Fuzzy, Chapter 5) and the character component classification method (see Chapter 6) are presented and critically appraised. The evaluation of the methods was based on a dataset of images collected from numerous Web pages. A description of the dataset used, is given in Section 7.1. The two segmentation methods presented in Chapters 4 and 5 are evaluated on the same dataset and statistical results are given for each one, as well as characteristic examples (Sections 7.2.1 and 7.2.2). Then a comparison between the two segmentation methods is made in Section 7.2.3. The text line identification method is subsequently evaluated in Section 7.3. Finally, an overall discussion is given in Section 7.4.

7.1. Description of the Dataset

In order to evaluate the methods described in this thesis, a dataset of images collected from a variety of Web pages was used. To achieve a representative sampling of the images, the images of the dataset, originate from Web pages that the average user would be interested to browse. Sites of newspapers, companies, academic sites, e-commerce sites, search engines etc were included in the sample. All the images in the data set contain text. The ability of the text extraction method to decide whether an image contains text or not, is a desired property, but considered to be out of the scope of this research, so images that do not contain text were not included in the data set. The function of the images in the Web page (header, menu item, logo, equation etc.) was also considered when creating the data set, so that the resulting collection reflects well the use of images in Web pages. The images contained in the dataset can be seen in Appendix C. This section will summarize the details of the images in the data set, and describe the way those images were categorized.

The data set comprises *115* images, *33* of which are *JPEG*s and *82* *GIF*s. Although *JPEG* is the most commonly used format for images in Web as mentioned previously, when it comes to images containing text, *GIF* seems to be the default choice. This is partly because *GIF* does not distort characters as much as *JPEG*. As will be seen later, *JPEG* is used for larger images and when more complex colour schemes are used, whereas *GIF* is preferred for smaller images and less complex colour schemes, such as company logos. A summary of the basic properties of the images in the dataset is given in Table 7-1.

	Minimum	***Maximum***	***Average***
Width	23	770	190
Height	15	354	77
Spatial Resolution (Dots Per Inch)	72	300	75
Colour Resolution (Bits Per Pixel)	8	24	12
Number of Colours Used	2	29,440	2,341
Number of Characters in Image	2	83	20
Number of Readable Characters	0	79	16
Number of Non-readable Characters	0	46	4

Table 7-1 – Summary of image properties for the images in the dataset.

The resolution of *110* out of the *115* images of the data set is *72dpi*, as expected for images created to be viewed on computer monitors. Only one image had a resolution of *300dpi*, while the remaining four images had a resolution in the middle range. The width of the images ranges from *23* to *770* pixels and the height from *15* to *354* pixels. The average size of the images in the data set is *190x77* pixels. The number of colours used in the images range from *2* up to *29,440*, with an average of *2,341* colours. The average number of colours of *GIF* images only is *218*, whereas the average number of colours for *JPEG* images is *7,850*.

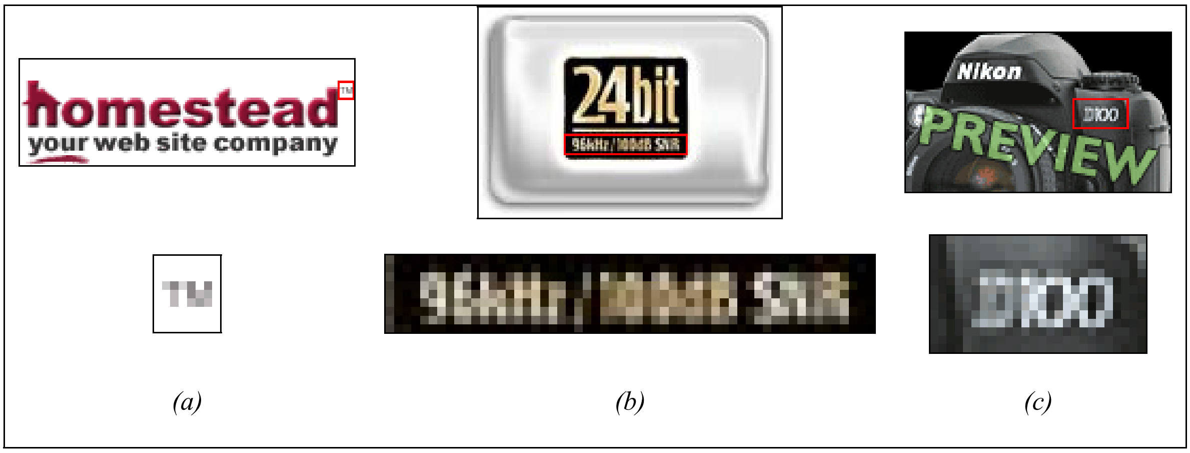
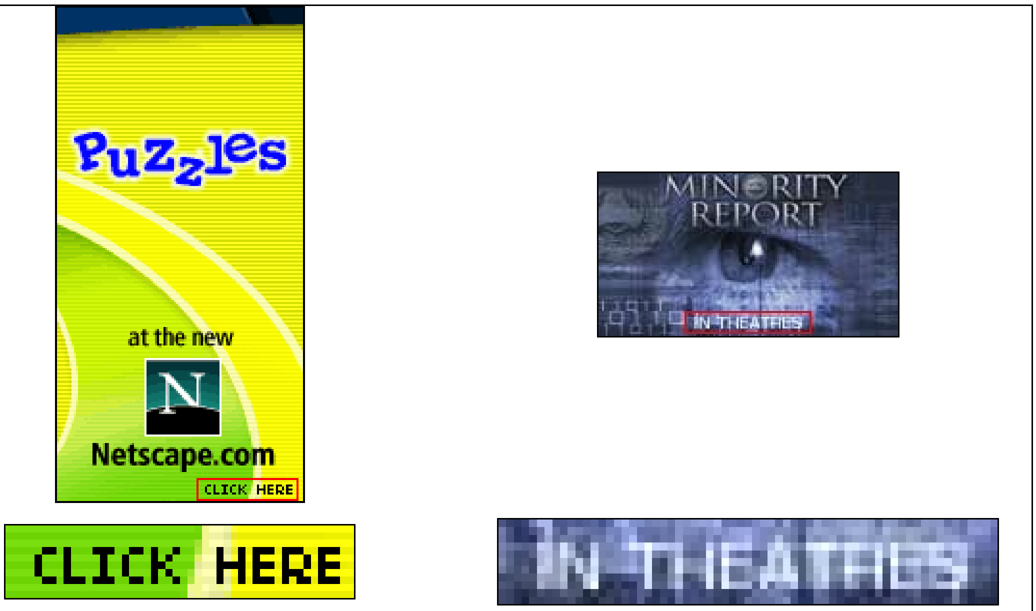


Figure 7-1 – (a) Image containing text, which is too small to be read. (b) Image containing badly designed text, which the viewer cannot read. (c) Image containing text as part of the background.

Not all of the text present in a Web image is readable by humans. There are cases where characters exist and are either very small to be read (Figure 7-1a), badly designed in terms of the selection of colours or the use of anti-aliasing (Figure 7-1b), or belong (semantically) to the background (Figure 7-1c). For this reason, text in the images of the dataset is characterized as either Readable or Non-readable. A minimum size was defined for a character to be considered readable, in terms of its width and height. The minimum width was chosen to be *4* pixels and the minimum height *6* pixels, since even humans have difficulties recognizing characters smaller than this threshold. Although the above rule works quite well in most of the cases, there are some exceptions, mainly due to the extent of use of antialiasing. In a few cases, characters were smaller than the thresholds decided still no antialiasing was used. In such cases, as long as the viewer was able to read the characters, they were characterized as readable (e.g. Figure 7-2a). In a similar manner, there were cases were the characters were larger than the thresholds set, yet due to extensive antialiasing, they were difficult to read (e.g. Figure 7-2b); thus, they were characterized as non-readable. The ultimate decision was made by a human operator, on a case by case basis.



(a) *(b)*

Figure 7-2 – (a) Image containing small, but readable text. (b) Image containing large but non-readable text.

The number of characters in the images of the data set ranges from *2* to *83*. An average image was found to have around *20* characters, out of which around *16* are readable. In total, the images in the dataset contain *2,404* characters, out of which *1,852* are classified as readable and *552* are classified as non-readable.

The images in the data set were grouped into four categories according to the colour combinations used. Category A holds images that contain multicolour characters over multicolour background. Category B contains images that have multicolour characters over single-colour background. Category C has images with single-colour characters over multicolour background. Finally, Category D holds images with single-colour characters rendered over single-colour background. The grouping of images into the four categories is shown in Figure 7-3, while the number of characters per category is shown in Figure 7-4.

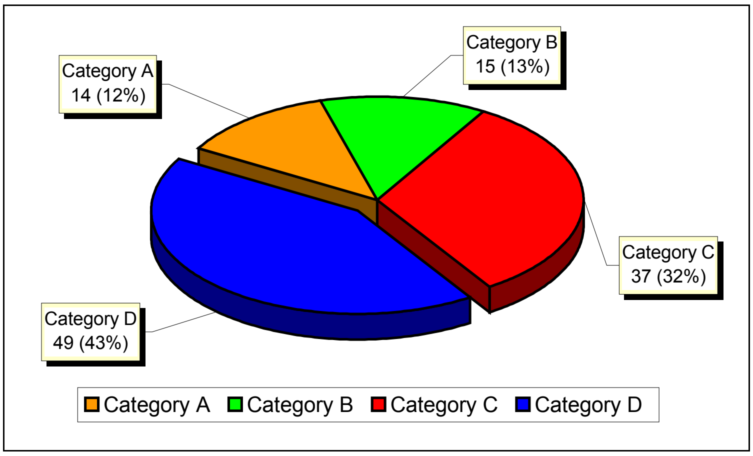


Figure 7-3 – Grouping of the images of the dataset into the four categories.

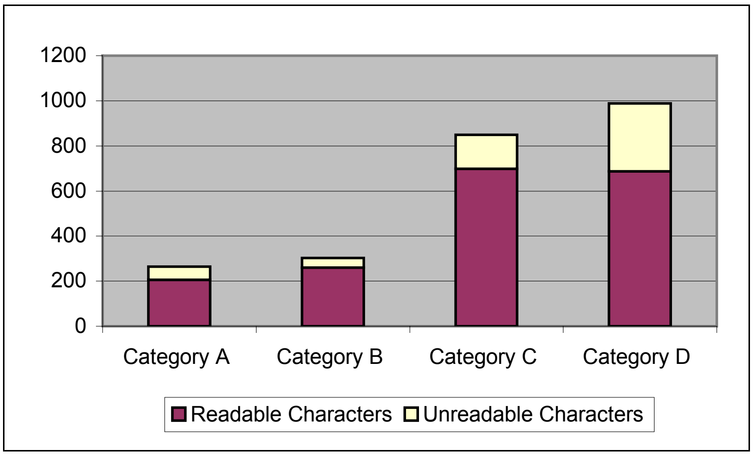


Figure 7-4 – Number of characters per category

Whether the text (or background) is single-coloured or multi-coloured is assessed by visual inspection. If the text (or the background) appears single-coloured, it is classified as such, even if it consists (in reality) of pixels of slightly different colours. This is in accordance to the methods being anthropocentric: if a human cannot discriminate between two colours, the methods shouldn't. Furthermore, antialiasing is not taken into account for the grouping of images. For example, if single-coloured text is antialiased over single-coloured background (thus producing a multi-coloured edge: a smooth transition between the foreground and background colour), the image is still considered to be of single-colour text and single-colour background.

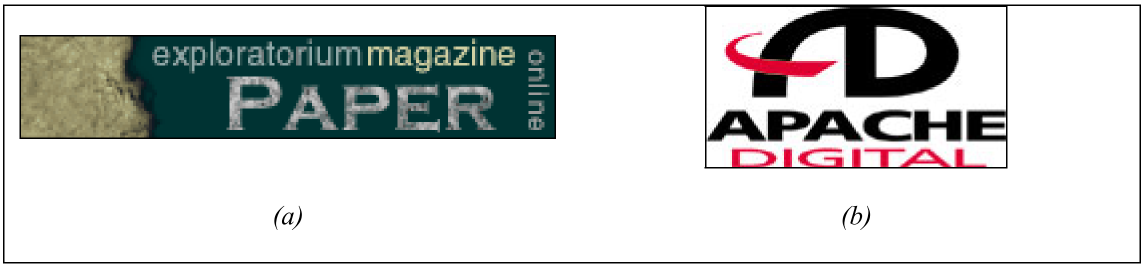
Images of category A are the most difficult to deal with since they have the most complex combination of colours. There are *14* images in this category, *8* of which are in *JPEG* format. *JPEG* is widely used for images of this class because of their complex colour composition. Gradient or textured characters and backgrounds are common in this category, as well as photographic backgrounds.

Images of category B are equally difficult to segment. Category B contains *15* images. Characters are usually rendered in gradient colours or textures, and their edges are typically antialiased.

Categories C and D contain images that are relatively easier to analyse. The colour of the characters in the images of both categories is either constant or slightly variable (perceived as constant). Category C contains *37* images with multicolour background. Category D holds *49* images, the most of any category. Most logos and equations fall in this category. Only *9* images in this class are in the *JPEG* format due to the (usually) small size of the images and the simple colour (combinations) used. The almost constant colour of characters and background gives well-defined edges in the majority of the cases, although in some images antialiasing is evident.

The distribution of images in the four categories reflects the occurrence of images of those types on Web Pages.

A text extraction method should ideally be able to cope with images of every category. It would be beneficial if an automatic decision could be made about the class of a given image, as this would allow different segmentation approaches (fine tuned for the specific class of images) to be used. Nevertheless, such a decision proves rather difficult since there are no distinctive features for each category. For example, the number of colours cannot be used as a distinctive feature since category D (single-colour characters over single-colour background) contains images with up to *22,000* colours, while at the same time category A (multicolour characters over multicolour background) holds images with as little as *31* colours (e.g. Figure 7-5).



(a)

(b)

Figure 7-5 – (a) An image belonging to category A (multicoloured text and background). The number of colours used in this image is only 64. (b) An image belonging to category D (single-colour text and background). The number of colours in this image is 1,973, mainly due to antialiasing.

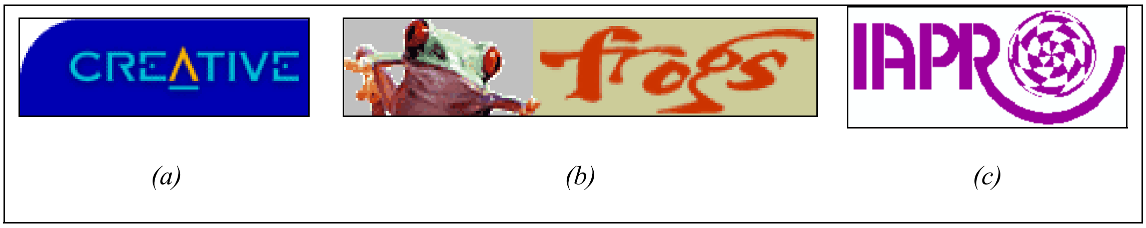
7.2. Segmentation Results

Both segmentation methods described in Chapters 4 and 5 were evaluated based on all the images contained in the dataset. The aim of the segmentation process is to partition the image into disjoined regions, in such a way that the text is separated from the background, and characters are not split in sub-regions or merged together.

The evaluation of the segmentation methods was performed by visual inspection. This assessment can be subjective since the borders of the characters are not precisely defined in most cases (due to anti-aliasing or other artefacts caused by compression). Nevertheless, since no other information is available about which pixel belongs to a character and which to the background (no ground truth information is available for web images), visual inspection is the only method of assessment that can be used. This method of assessment is in agreement with previous research on web image text extraction. Lopresti and Zhou [106], for instance, evaluate their segmentation results in the same way. Since visual assessment is inherently subjective, in cases where it is not clear whether a character-like component contains any pixel of the background or not, the evaluator decides on the outcome based on whether by seeing the component on its own they can recognize the character or not. The foundation for this is that even if a few pixels have been misclassified, as long as the overall shape can still be recognized, the character will be recognisable by OCR software. Even though in many cases a human could still read the text in question, even if some pixels are missed (or added), OCR processes tend to be much more sensitive; hence, we err on the side of being conservative.

The following rules apply regarding the categorization of the results. Each character contained in the image is characterised as *identified*, *merged*, *broken* or *missed*. Identified characters are those that are described by a single component. Broken ones, are the characters described by more than one component, as long as each of those components contain only pixels of the character in question (not any background pixels). If two or more characters are described by only one component, yet no part of the background is merged in the same component, then they are characterised as merged. Finally, missed are the characters for which no component or combination of components exists that describes them completely without containing pixels of the background.

It should be made clear at this point, that in cases where characters are merged or broken in the original image, the segmentation method is expected to segment them as a single (if merged) or multiple (if broken) components. In those cases, the characters are considered identified if they are segmented as single or multiple components since they appear as such in the original. An example can be seen in Figure 7-6.



(a) *(b)* *(c)*

Figure 7-6 – (a) An image containing broken characters. (b) An image containing merged characters. (c) An image containing both split and merged characters.

In terms of importance, separation of the text from the background comes first. Given that separation has been achieved, in terms of successful segmentation, identified characters come first, merged characters come second, and broken characters come last. Identified characters are ready for further processing (binarization, resolution enhancement, recognition). If two or more characters are merged, despite it being a difficult situation, they can still be processed as a word (possibly separated with the use of projection profiles). Broken characters, on the other hand, are very difficult to recombine, especially when the text (and/or background) is multi-coloured.

In the results presented here, the original image is shown in each case, along with an image of the final segmentation (with each component painted in different colour) and an image of the segmented characters. In the image of the segmented characters, correctly identified characters are shown in black, broken characters in red, and merged characters are shown in blue.

7.2.1. Split and Merge Method

The overall results for the Split and Merged method can be seen in Table 7-2. In total, the method was able to correctly identify *1,290* (*69.65%*) out of *1,852* readable characters, while *151* (*8.15%*) characters were merged, *271* (*14.63%*) were broken and *140* (*7.56%*) characters were missed. In addition, out of the *552* non-readable characters, the method was able to identify *184* (*33.33%*).

		Number of Characters	***Identified***	***Merged***	***Broken***	***Missed***
All Categories	Readable	1,852	1,290 (69.65%)	151 (8.15%)	271 14.63%)	140 (7.56%)
	Non-readable	552	184 (33.33%)	22 (3.99%)	160 (28.99)	186 (33.70%)
Category A	Readable	206	115 (55.83%)	0 (0.00%)	60 (29.13%)	31 (15.05%)
	Non-readable	58	12 (20.69%)	2 (3.45%)	15 (25.86%)	29 (50.00%)
Category B	Readable	260	135 (51.92%)	48 (18.46%)	67 (25.77%)	10 (3.85%)
	Non-readable	42	6 (14.29%)	3 (7.14%)	3 (7.14%)	30 (71.43%)
Category C	Readable	699	530 (75.82%)	48 (6.87%)	64 (9.16%)	57 (8.15%)
	Non-readable	150	55 (36.67%)	11 (7.33%)	45 (30.00%)	39 (26.00%)
Category D	Readable	687	510 (74.24%)	55 (8.01%)	80 (11.64%)	42 (6.11%)
	Non-readable	302	111 (36.75%)	6 (1.99%)	97 (32.12%)	88 (29.14%)

Table 7-2 – Results of the Split and Merge method over the images of the dataset.

Generally, the Split and Merge method was able to identify characters in more than *50%* of the cases for categories A and B, containing multi-coloured characters. At the less difficult categories C and D, the method is able to identify characters in more than *70%* of the cases.

For the images of category A (multi-colour characters over multi-coloured background), the method was able to correctly identify *115* (*55.83%*) of the *206* readable characters. No characters were merged here, while *60* (*29.13%*) characters were broken and *31* (*10.05%*) were missed. The fact that characters are multi-coloured in this category is reflected in the results, as characters tend to be broken than merged. The high difficulty associated with the images of this category is also suggested by the high ratio of missed characters. With *15.05%*, category A has the highest percentage of missed characters of all four. As far as it concerns the non-readable characters, *12* (*20.69%*) out of *58* characters were identified and *29* (*50.00%*) are missed. Given the special characteristics of those characters (small size, strong antialiasing etc.) this can be considered an encouraging result. It should be kept in mind that non-readable characters in most of the cases are not meant to be read (e.g. copyright and registered symbols, trademark symbols etc.).

Category B (multi-coloured characters over single-coloured background) presents similar results as far as it concerns the percentage of correctly identified characters. In this category, *135* (*51.92%*) out of *260* readable characters are identified, *48* (*18.46%*) are merged, *67* (*25.77%*) are broken and *10* (*3.85%*) are missed. As far as it concerns non-readable characters, *6* (*14.29%*) out of *42* are identified and *30* (*70.43%*) are missed.

A number of observations can be made for this category. First, with *51.92%*, category B is the one with the smallest percentage of identified characters. The percentage of correctly identified characters for both categories A and B is under *60%*, verifying the higher difficulty multi-coloured characters pose to the Split and Merge method. A second point worth noticing is that category B yields a much smaller percentage of missed characters than category A, while at the same time, it presents the highest percentage of merged characters of all four categories. This can be awarded to the fact that the background in this case is single-coloured. Due to the way the Split and Merge method works, if the background is single-coloured, it is more probable to get it as a single component during the merging phase at an early stage, before mergers between the background and the characters (or parts of them) are considered. If the background is correctly segmented early in the process (and if single-coloured, it will be a larger component than the characters), it becomes more difficult to merge smaller components (like parts of character) with it. As a result, fewer components are merged with the background (small percentage of missed components) and more probabilities exist for character parts to be merged between them.

For the images of Category C (single-coloured characters over multi-coloured background), *530* out of *699* (*75.82%*) characters are identified, *48* (*6.87%*) are merged and *64* (*9.16%*) are broken. The number of missed characters is *57* (*8.15%*). As can be easily observed, the percentage of identified characters is much higher than before, while merged, broken and missed characters only account for less than 10% of the characters each. The higher identification rate was expected, since the text in images of this category is single-coloured. Equally interesting results are obtained for the non-readable characters of this category. Out of *150* non-readable characters, *55* (*36.67%*) are identified, *11* (*7.33%*) are merged and *45* (*30.00%*) are broken, while only *39* (*26.00%*) are missed.

Finally, for the images in Category D (single-coloured characters over single-coloured background), out of *687* readable characters, *510* (*74.24%*) were identified, *55* (*8.01%*) were merged and *80* (*11.64%*) were broken, while *42* (*6.11%*) were missed. Similarly to category C, we obtain quite high percentages of identified characters, due to the characters being single-coloured. As far as it regards the background being single-coloured, the same pattern can be observed like between categories A and B.

Specifically, in category D we get a smaller percentage of missed characters than category C, while at the same time the percentage of merged ones is slightly higher. Although the magnitude of this effect is not the same as between categories A and B, it is possibly caused by the same reason: the background being single-coloured in categories B and D.

The results mentioned above reflect the increasing difficulty in categories where the text and/or the background are multi-coloured. Categories A and B, which contain multi-coloured text yield lower percentages of identified characters (less than *60%*), while categories C and D, which contain single-coloured text give substantially higher identification rates, around *75%*. The background plays a secondary role here, as it affects more the percentages of missed characters. As can be observed, categories B and D (images with single-coloured background) yield lower percentages of missed characters than categories A and B (images with multi-coloured background) respectively. Figure 7-7 is a comparative chart between the categories.

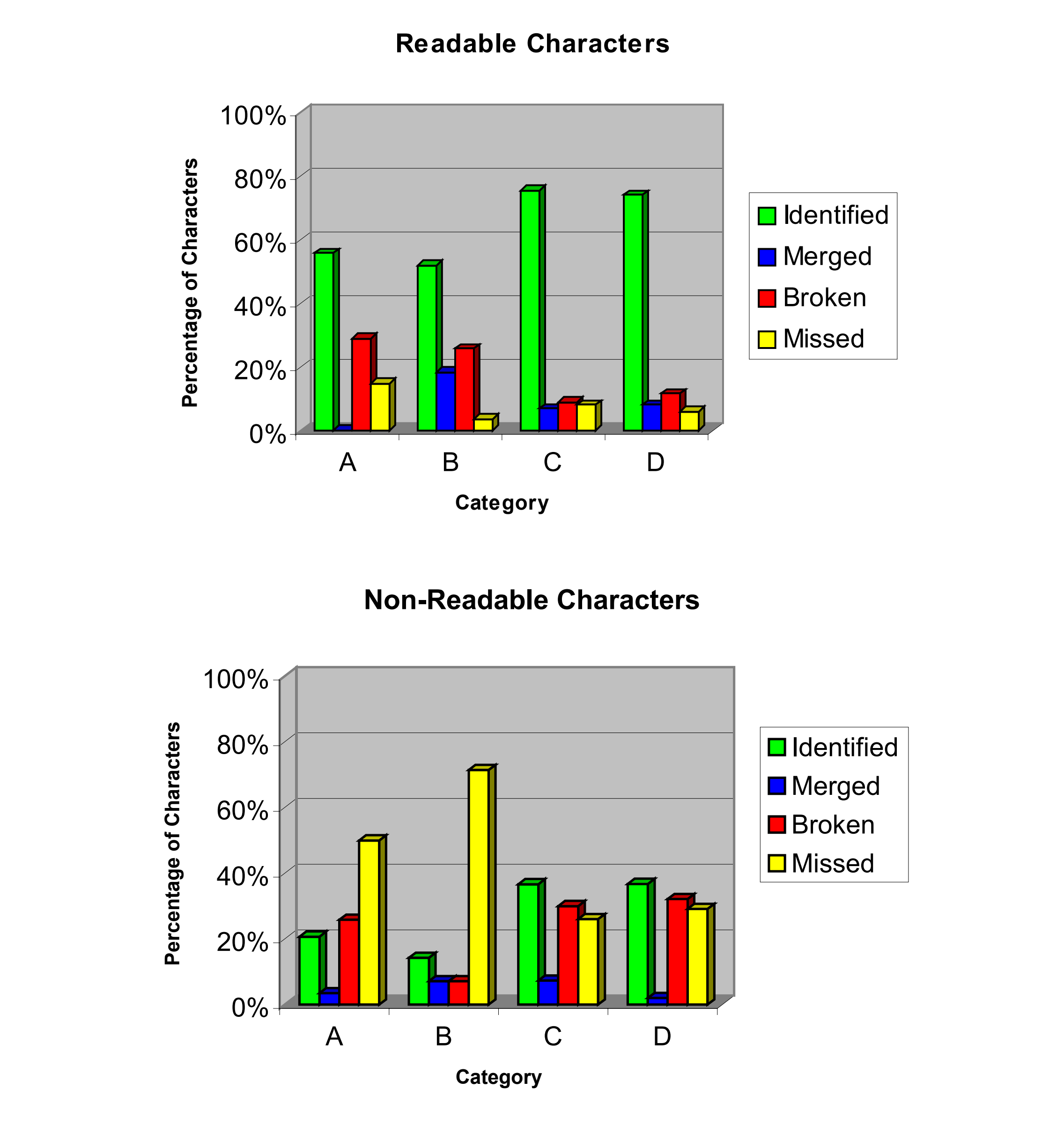


Figure 7-7 – Comparative charts between the categories. Percentages of identified, merged, broken and missed characters for all categories.

A different visualisation of the results can be seen in Figure 7-8. In *38* (*33%*) of the images, the Split and Merge segmentation method was able to identify *100%* of the characters.

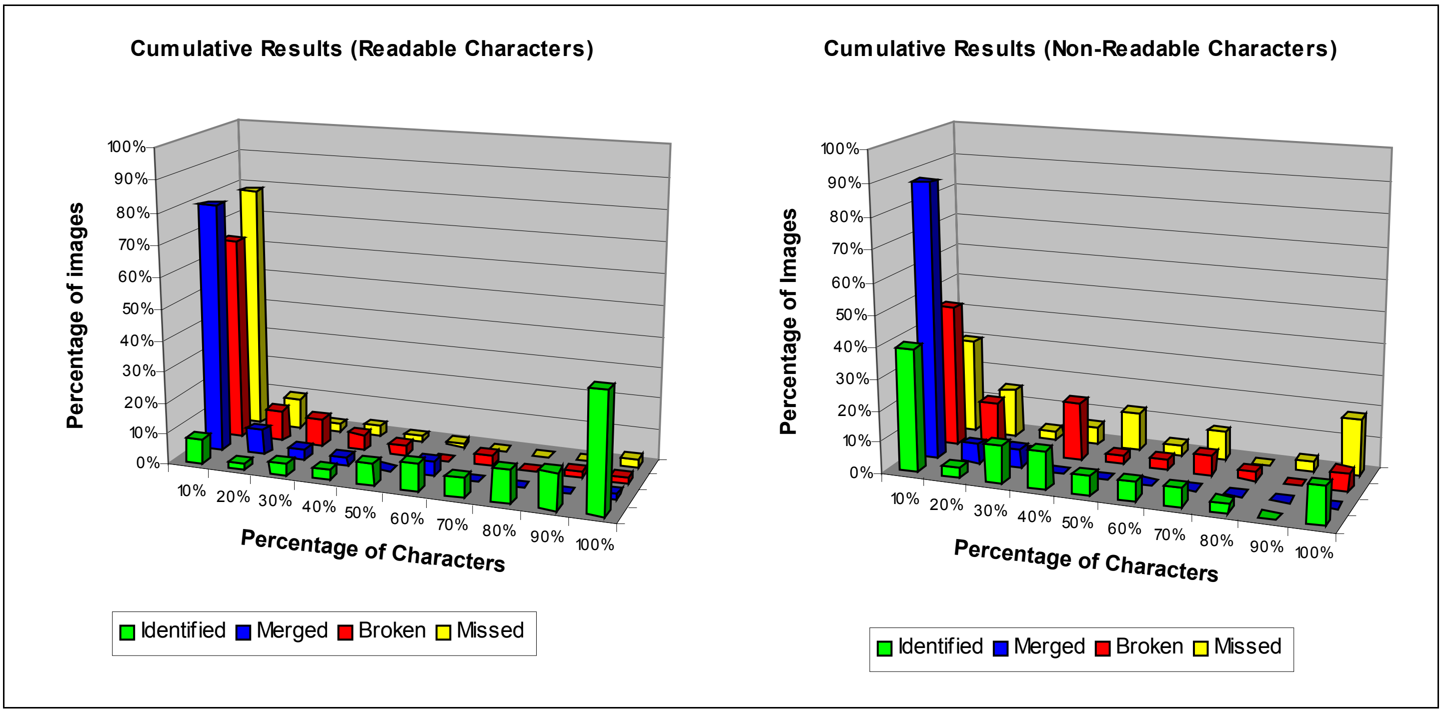


Figure 7-8 – Split and Merge Segmentation method performance. Cumulative Results.

In terms of execution time, the Split and Merge method performs rather poorly as can be seen in Table 7-3. All the results reported here are based on a Pentium IV computer running at 1.8GHz. The Splitting stage is fast; there are only three cases where it takes slightly more than 1 second to run. Most of the time spent for the splitting stage (around *80%*) is actually consumed by the user interface in order to visualise the tree of the layers produced. The most time of a run of the method is consumed to the Merging stage. Depending on the image, the Merging stage can take from less than a second, up to *15 – 20* minutes, producing an overall average of around *5* minutes. The most time consuming part of the method is the computation of the overlapping between components, and the actual merging of them. The main reason for that is the internal representation used for the connected components: a list of runs of pixels. This representation proves to be quite robust for a number of other processes (e.g. the Fuzzy method, user interface actions), but not good for the Split and Merge method. It should be made clear at this point that the code used for these measurements is un-optimised in many aspects, and further improvement of run times is definitely possible.

	Time to Split (sec)	*Time to Merge (sec)*	*Total Time (sec)*
Category A	0.445	414.701	415.146
Category B	0.313	558.649	558.962
Category C	0.366	609.529	609.895
Category D	0.182	33.774	33.956
All Categories	0.286	320.396	320.683

Table 7-3 – Average execution times for the Split and Merge segmentation method per category.

Two factors play an important role in terms of execution time: the size of the image in question, and the complexity of its colour scheme. The first factor is self-explanatory, the bigger an image is, the more time is required for each operation. The second factor, namely the colour content of the image, or else the total number of colours it contains, affects the method in a different way. First, the more colours an image has, the more peaks will be identified in the histograms and the more layers will be produced. This affects both the splitting and the merging process. The number of components produced also increases as the number of colours (and layers) of the image grows. Based on those observations, the role of those two factors was studied.

The images of the dataset were broken into different size groups and colour content groups, depending on the total number of pixels (*width* x *height*) and the total number of colours they have. The time required for the splitting and the merging phase for each image in the dataset was recorded and the results can be seen in the figures below. Two remarks can be made here: the execution time increases roughly exponential as the size and colour complexity of the images rise, and the splitting stage takes negligible time comparing to the merging stage.

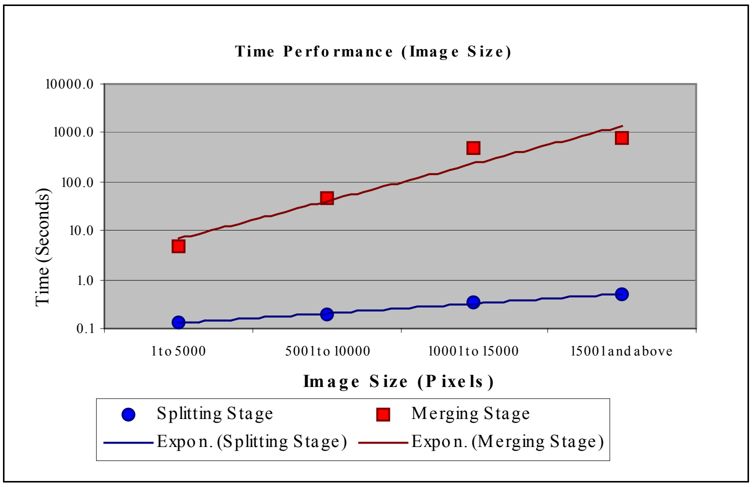


Figure 7-9 – Time performance of the Split and Merge segmentation method, for images of increasing size.

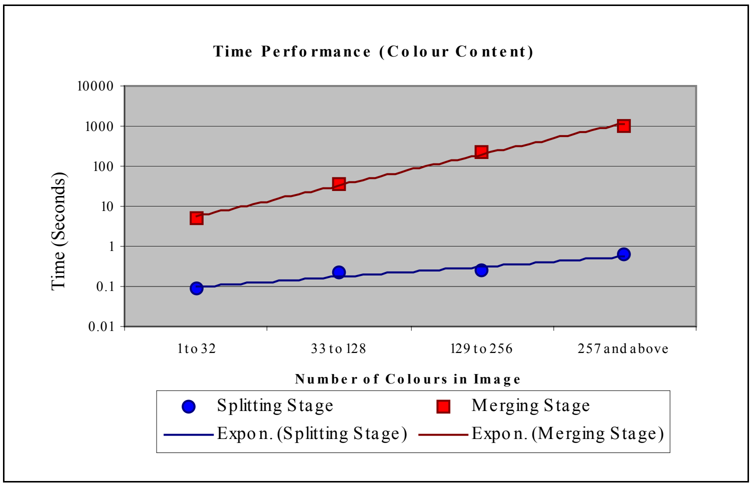
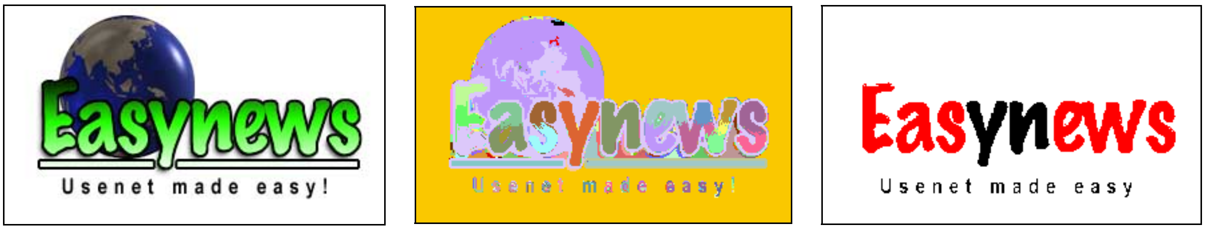


Figure 7-10 – Time performance of the Split and Merge segmentation method, for images of increasing colour content.

In figures 1.2-6 to 1.2-11, a number of special cases representative of the difficulties encountered can be seen along with the corresponding results. For every image, the original is given along with an image of the final segmentation, and an image of the segmented characters.

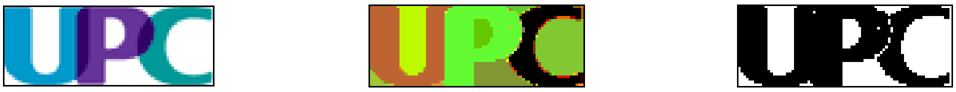
A typical problem with multi-coloured characters is that in many cases they are painted in some gradient colour. In the case of Figure 7-11 the characters are green with a gradient intensity. To correctly segment those characters, more relaxed thresholds have to be used for the colour similarity measurements. Nevertheless, as mentioned in Chapter 4, this can cause a problem to numerous images of complex colour schemes that the text is of similar colour to the surrounding background. Figure 7-11 illustrates one of the difficult cases that the currently used thresholds cannot cope with. Most of the characters here are split in two components.



Original *Final Segmentation* *Segmented Characters*

Figure 7-11 – An image containing characters of gradient colour.

A different problem appears when characters are overlapping. In the case of Figure 7-12 the colours of the characters are also mixed. As can be seen in the final segmentation image, the three characters are well separated into three connected components, each complete enough to recognize. In this case, parts of "*U*" and "*C*" have been assigned to "*P*". In order to get all three characters right those common areas should belong to more than one character, which is not allowed with the current segmentation requirements for disjoined, non-overlapping areas. Nevertheless, although not visible in Figure 7-12, the vexed areas (possible extensions) of the components representing "*U*" and "*C*" actually consist of exactly the missing parts of the characters. This could be used as an extra feature for recognition, as will be discussed next.

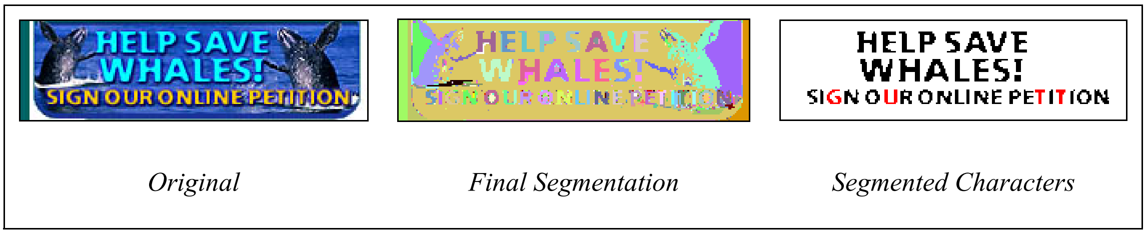


Original *Final Segmentation* *Segmented Characters*

Figure 7-12 – An image containing overlapping characters.

Antialiased characters can be a significant problem during segmentation. If the characters are thick enough, antialiasing will create a soft edge, but will leave enough part of the character unaffected, which the segmentation process can identify. Smaller characters though, tend to present problems with antialiasing, since the main part of the character is now affected.

In Figure 7-13, the big characters on top are not really affected, although they are antialiased. The shape of the smaller characters though is slightly altered as can be seen in the magnified region of Figure 7-13 (shown in Figure 7-14). A close inspection reveals that the upper parts of the "*E*" and "*T*" characters are blended strongly with the background. This hinders the segmentation process, which is able to correctly identify only some of the characters (like "*E*") but misses a number of others. It should be made clear at this point, that the characters shown here are comparably large, still affected by strong antialiasing. The problem is even larger with smaller or thinner characters.



Original *Final Segmentation* *Segmented Characters*

Figure 7-13 – An image of single-coloured characters over multi-coloured background. Strong antialiasing results in slightly changing the colour of thin characters.

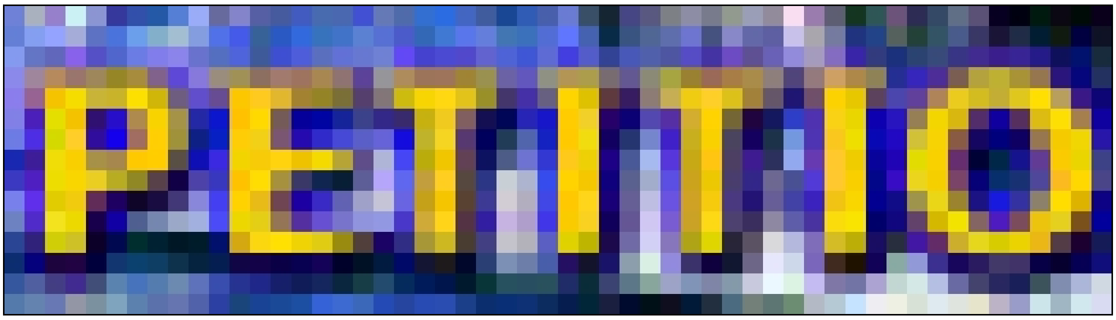
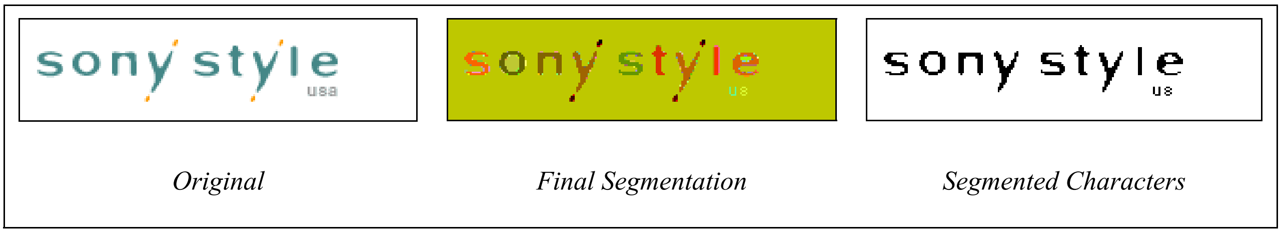


Figure 7-14 – Magnified region of Figure 7-13.

Another situation worth commenting on, is the existence of extremely small characters in an image. Figure 7-15 is an example of such an image. Small characters such as the ones of the word "*usa*" in the image are usually considered non-readable. A magnification of the area with the characters in question is shown in Figure 7-16. The problems with correctly segmenting such characters can be easily identified.

Small characters are the ones affected most by antialiasing, compression, low contrast colour schemes etc. Still, the Split and Merge method can identify such characters in numerous cases. One such case is the one presented here.



Original *Final Segmentation* *Segmented Characters*

Figure 7-15 – An image containing small (non-readable) characters.

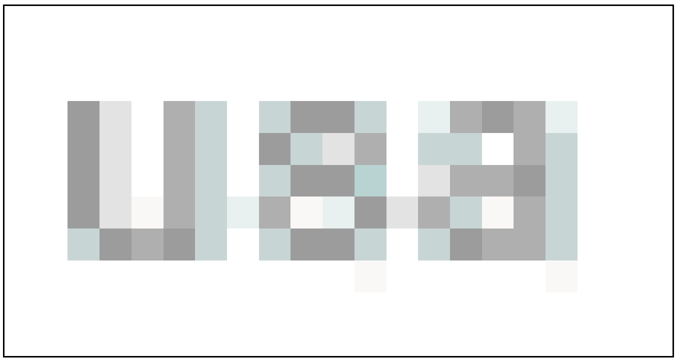


Figure 7-16 – Magnified region of Figure 7-15

Overall, the Split and Merge method proved to be versatile when evaluated with the images of the dataset. As expected, the images of categories A and B (containing multi-coloured characters) proved to be more difficult to segment, while the images of categories C and D (containing single-coloured text) were substantially easier. Generally, a trade-off must be made so that the method works efficiently in a range of fundamentally different images. This trade-off lies exactly at the selection of the similarity thresholds used. The average correct identification rate of around *70%* is a promising result for the Split and Merge method.

7.2.2. Fuzzy Method

Table 7-4 shows the overall results for the fuzzy segmentation method. Out of *1,852* readable characters, *1,288* (*69.55%*) were identified, *150* (*8.10%*) were merged, *169* (*9.13%*) were broken and *245* (*13.23%*) were missed. As far as it concerns non-readable characters, *223* (*40.40%*) out of *552* were identified, *71* (*12.86%*) were merged, *92* (*16.67%*) were broken and *166* (*30.07%*) were missed. Although based on these results the Fuzzy method performs at the same levels like the Split and Merge method, it will be seen next, that they perform significantly different at the level of individual categories.

		Number of Characters	***Identified***	***Merged***	***Broken***	***Missed***
All Categories	Readable	1,852	1,288 (69.55%)	150 (8.10%)	169 (9.13%)	245 (13.23%)
	Non-readable	552	223 (40.40%)	71 (12.86%)	92 (16.67)	166 (30.07%)
Category A	Readable	206	122 (59.22%)	8 (3.88%)	24 (11.65%)	52 (25.24%)
	Non-readable	58	15 (25.86%)	5 (8.62%)	8 (13.79%)	30 (51.72%)
Category B	Readable	260	180 (69.23%)	16 (6.15%)	36 (13.85%)	28 (10.77%)
	Non-readable	42	9 (21.43%)	5 (11.90%)	1 (2.38%)	27 (64.29%)
Category C	Readable	699	494 (70.67%)	66 (9.44%)	40 (5.72%)	99 (14.16%)
	Non-readable	150	66 (44.00%)	30 (20.00%)	26 (17.33%)	28 (18.67%)
Category D	Readable	687	492 (71.62%)	60 (8.73%)	69 (10.04%)	66 (9.61%)
	Non-readable	302	133 (44.04%)	31 (10.26%)	57 (18.87%)	81 (26.82%)

Table 7-4 – Results of the Split and Merge method over the images of the dataset.

Generally, the method performs better (around *70%* characters identified) in the easier categories C and D and a bit poorer in the more difficult ones A and B. Nevertheless, the gap in performance between the categories containing multi-colour characters and the ones containing single-coloured characters is much smaller than in the Split and Merge method. Here the performance in category A (the most difficult one) reaches *60%*, while the performance in category B touches *70%*.

For the images of category A (multi-coloured characters over multi-coloured background) out of *206* readable characters, *122* (*59.22%*) were identified, *8* (*3.88%*) were merged and *24* (*11.65%*) were broken, while *52* (*25.24%*) were missed. As expected, the performance of the method on this category is the poorer of all. Still with an identification rate of nearly *60%* the method performs reasonably well taking in mind the complication of those images. As far as it regards the non-readable characters, the Fuzzy method was able to identify *15* (*25.86%*) out of *58* characters, while *5* (*8.62%*) were merged, *8* (*13.79%*) were broken and *30* (*51.72%*) were missed.

In category B (multi-colour characters over single-colour background), out of *260* readable characters *180* (*69.23%*) were identified, *16* (*6.15%*) were merged, *36* (*13.85%*) were broken and *28* (*10.77%*) were missed. In contrast to the Split and Merge method, where the performance in categories A and B is similar, here a vast improvement of the identification ratio can be observed. At the same time, the percentage of missed characters drops to *10%*. This is a strong indication that the Fuzzy method is affected more by the type of the background. Specifically, single-coloured background allows for better identification rates as well as less missed characters in the Fuzzy method, while for the Split and Merge method it effectively affected only the rate of missed characters and not the rate of the identified ones. Concerning the non-readable characters, *9* (*21.43%*) out of *42* characters were identified and *27* (*64.29%*) were missed. For this type of characters, the Fuzzy method performs roughly at the same levels for categories A and B.

Of the *699* readable characters of the images of category C (single-coloured characters over multi-coloured background), *494* (*70.67%*) were identified, *66* (*9.44%*) were merged, *40* (*5.72%*) were broken and *99* (*14.16%*) were missed. With an identification rate around *70%* the Fuzzy method does not perform much better in this category comparing to category B. Although the identification rate is here slightly higher, the difference of *1*% does not allow for any safe assumptions to be made. On the other hand, the method performs much better on non-readable characters in this category. Comparing to categories A and B where the identification rate of non-readable characters range around *25%*, the rate for category C is up to *44%*, while at the same time the rate for missed non-readable characters drops from above *50%* for categories A and B to *18.67%*. This is an indication that the type of the characters (single-coloured vs. multi-coloured) affects the Fuzzy method as far as it concerns small characters (which are the main representatives of non-readable ones), while it does not make a big difference for the rest (readable) characters.

Similarly to category C, the performance for category D (single-coloured characters over single-coloured background) is around *70%*. Specifically, out of *687* readable characters, *492* (*71.62%*) were identified, *60* (*8.73%*) were merged, *69* (*10.04%*) were broken and *66* (*9.61%*) were missed. Again, there is not a large difference between this category and categories B and C as far as it concerns identification rates. Concerning the non-readable characters, the method was able to identify *133* (*44.04%*) out of *302* and missed *81* (*26.82%*). Similarly to category C, a significant percentage of non-readable characters was identified.

As can be seen, the Fuzzy segmentation method yields a slightly lower identification rate for the images of category A, around *60%*, while the identification rates for the other three categories are around *70%*. The identification rates of *~60%* for category A and *~70%* for category B are quite high taking into account the difficulty that these categories present. At the much easier categories C and D though, the identification rate of around *70%* is acceptable but not much higher than the other categories.

The fact that all categories having either single-coloured text or single-coloured background yielded identification rates around *70%* is an indication that the Fuzzy method actually benefits from having single-coloured areas, no matter whether that is the text (Category C) the background (Category B) or both (Category D). At the same time, the fact that at the easier categories C and D the identification rates stay around *70%* (the same like category B) indicates that the type of the characters specifically does not play an important role here: as long as either the characters or the background is single-coloured, the method benefits without any preference between the two.

Where the type of the characters plays an important role is at the segmentation of the non-readable characters. For this special group of characters (consisting mainly of very small ones), the identification rates for Categories A and B, where characters are multi-coloured, is *~25%* while the rates of missed characters are well above *50%*. As long as the characters become single-coloured, in categories C and D, the identification rates double to around *44%* while the rates of the missed non-readable characters halve to around *25%*. A comparative chart between the performances of the categories can be seen in Figure 7-17.

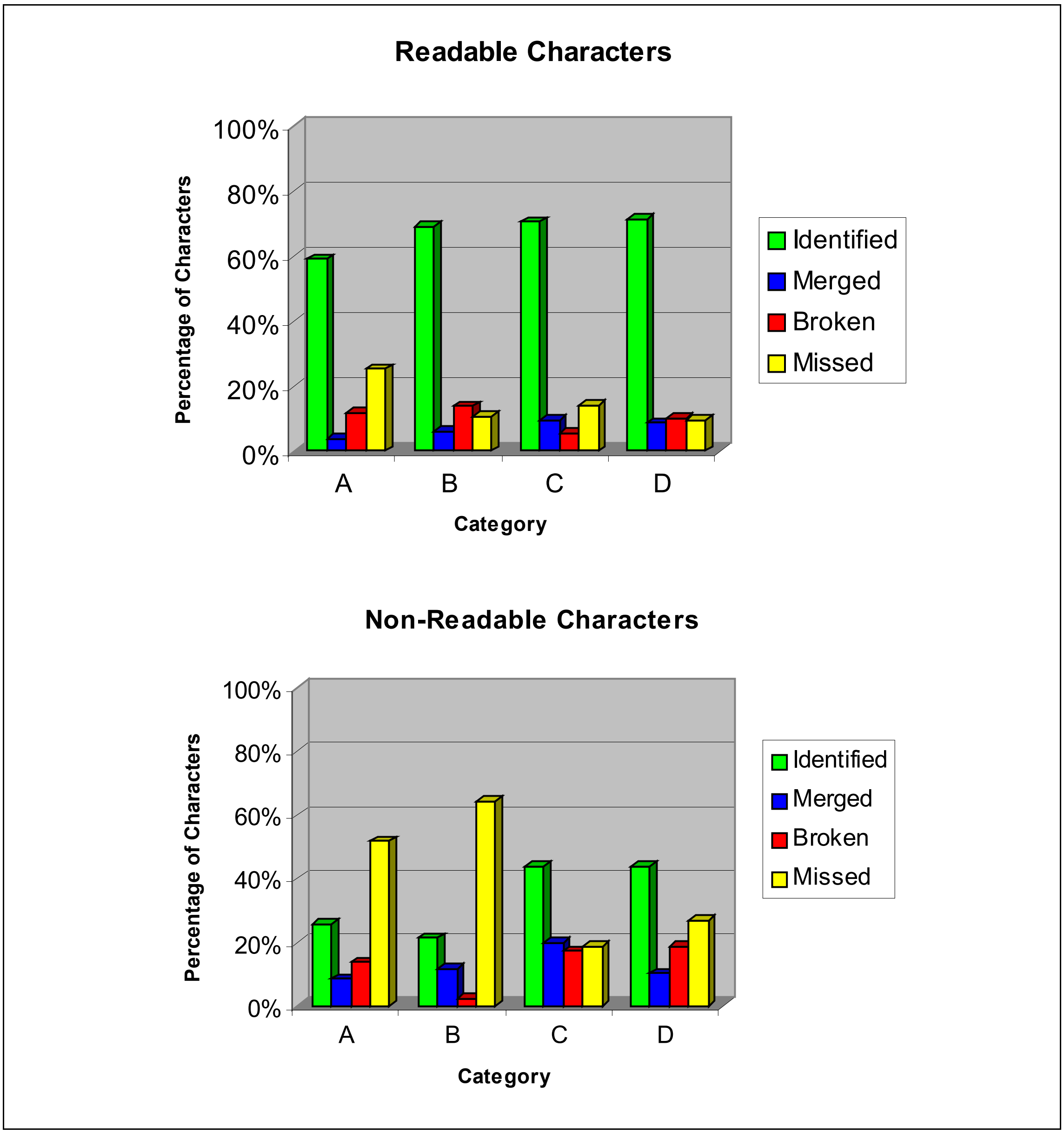


Figure 7-17 – Percentages of identified, merged, broken and missed characters for all categories.

A different visualisation of the results can be seen in Figure 7-18. In *30* (*26%)* of the images, the Fuzzy segmentation method was able to identify *100%* of the readable characters.

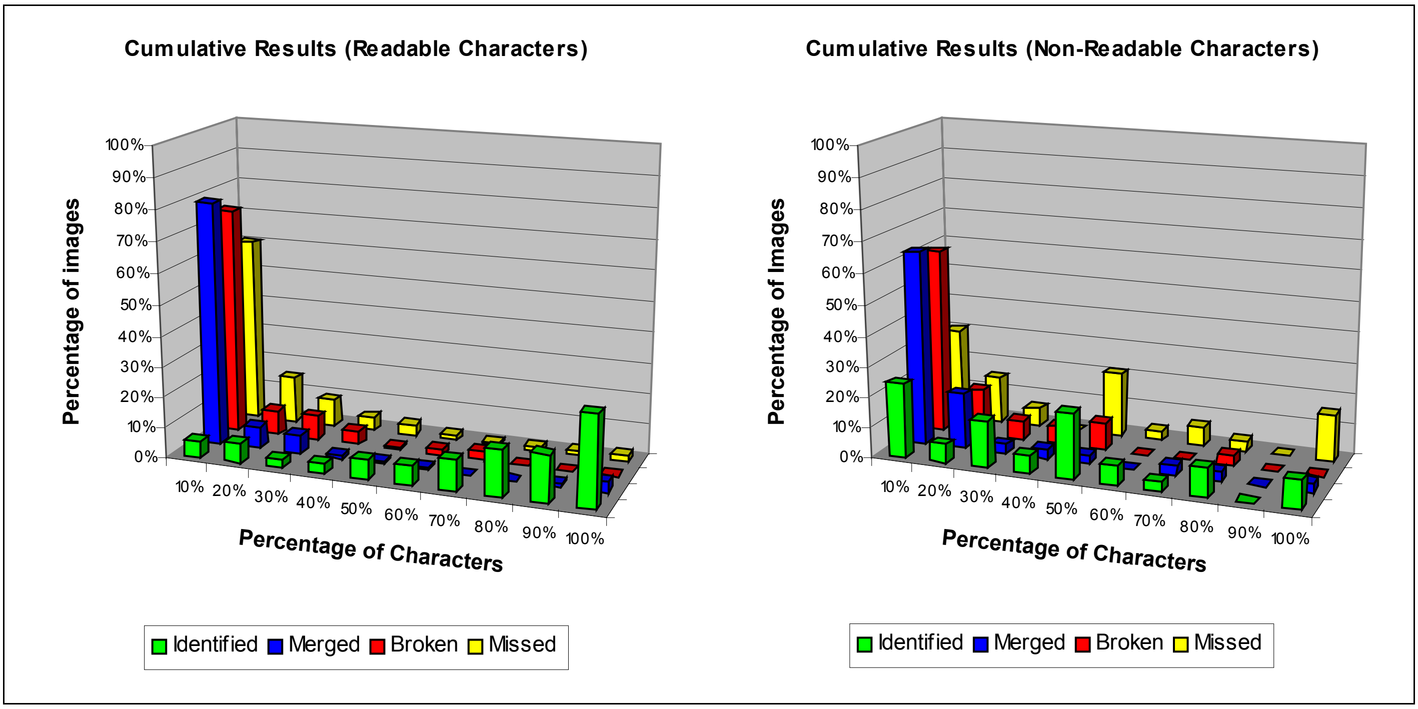


Figure 7-18 – Fuzzy Segmentation method performance. Cumulative Results.

In terms of the execution times of the Fuzzy segmentation method, two stages were measured: the initial connected component analysis stage, and the component aggregation stage. The measurements for each category of images in the dataset can be seen in Table 7-5. The Fuzzy segmentation method is quite fast. The initial connected component analysis stage never takes more than a second to complete, while the component aggregation phase takes on average around *15* seconds.

	Time of Initial CC Analysis (sec)	***Time of Component Aggregation (sec)***	***Total Time (sec)***
Category A	0.326	10.817	11.143
Category B	0.377	25.270	25.648
Category C	0.524	17.463	17.986
Category D	0.271	13.106	13.378
All Categories	0.373	15.816	16.189

Table 7-5 – Average execution times for the Fuzzy segmentation method per category.

Similarly to the Split and Merge method, an analysis was performed to establish the way different sizes and colour contents affect the Fuzzy segmentation method's run time. The same size and colour content groups were used, and the results are shown in Figure 7-19 and Figure 7-20.

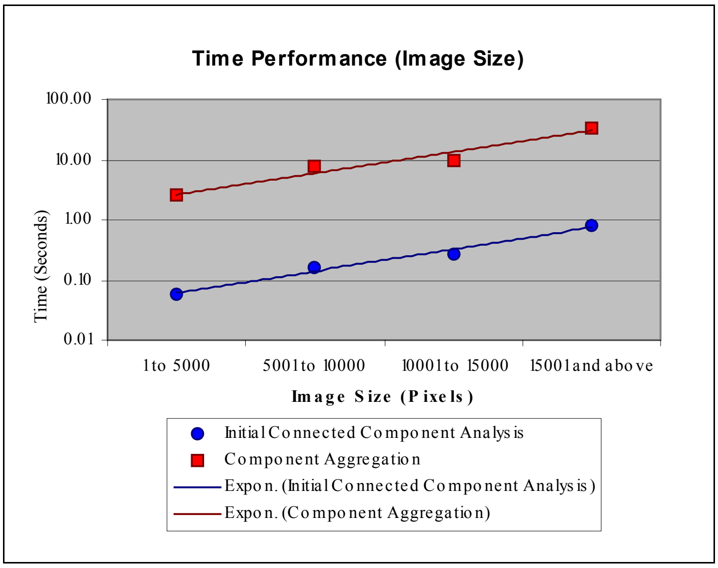


Figure 7-19 – Time performance of the Fuzzy segmentation method, for images of varying size and varying colour content.

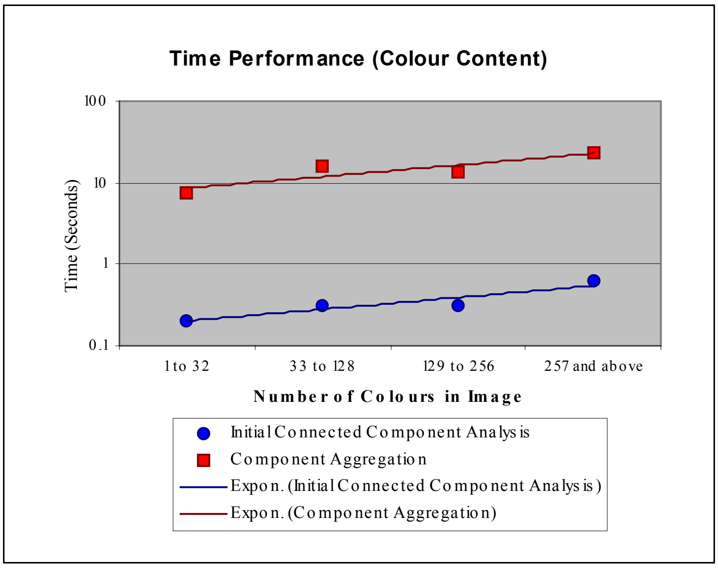
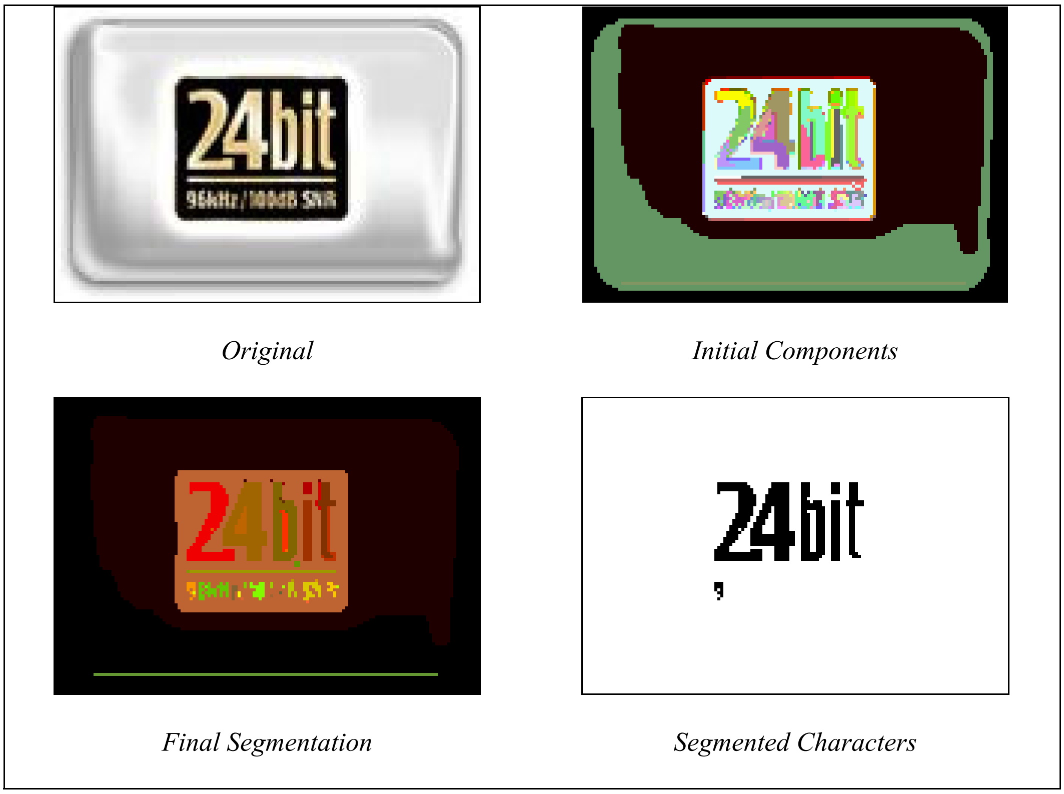


Figure 7-20 – Time performance of the Fuzzy segmentation method, for images of varying size and varying colour content.

Some characteristic examples of the difficulties encountered will be discussed next. This time, an image of the initial components identified during the pre-processing step (prior to the component aggregation process) is also given.

As far as in concerns multi-coloured characters, the method performs well in most of the cases. An example can be seen in Figure 7-21. The characters of this image range in colour from yellow to white, giving a highlighting effect to the text. The method was able to identify all of the characters correctly, mainly due to the fact that the surrounding background is of uniform colour. Still it is supporting to the method that the parts of the characters, which were first identified as smaller regions, were combined in the right way resulting in whole characters to be taken as single components. As far as it concerns the extremely complicated small characters in the image (characterized as non-readable), the method was able to identify the only one that is somewhat clearly rendered. For the rest of the small characters, one could claim that the method merged groups of them in single components. Nevertheless, since ground truth information is non-existent, it is extremely difficult for the person assessing the final segmentation to decide whether the rest of the components contain character pixels only. As stated before, in such cases we err on the side of being conservative, and classify such characters as missed.



Original

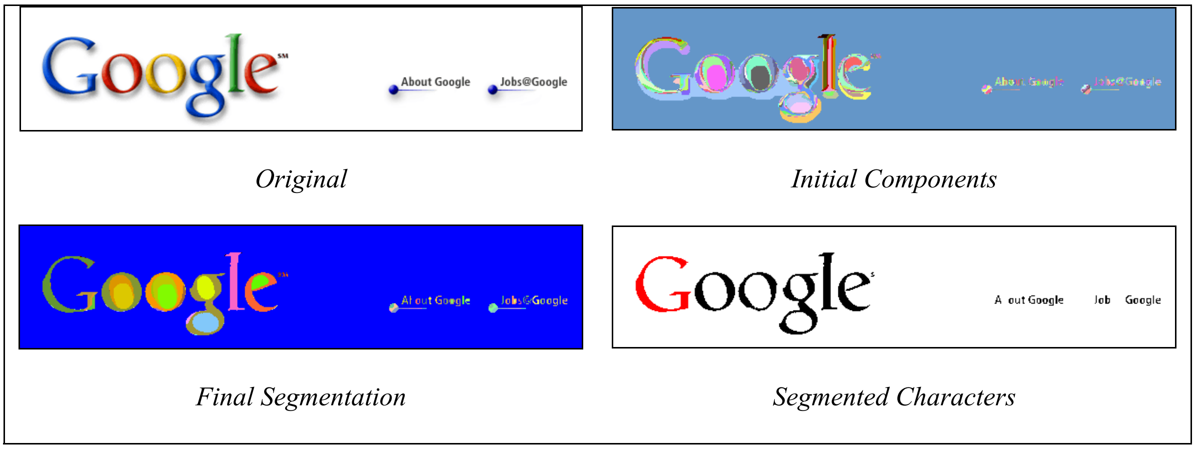
Initial Components

Final Segmentation

Segmented Characters

Figure 7-21 – An image containing multi-colour characters over uniform coloured background.

One more example of multi-coloured characters can be seen in Figure 7-22. Similarly to the previous example, the originally split (after the initial connected component analysis) characters are finally correctly identified, leaving the background and the shadow out of the final character components. The only exception here is character “*G*”, which is finally broken in two components. This is because the letter gets too thin at its lower right side, and subsequently the two parts of the character do not touch adequately for the algorithm to combine them. The small characters on the right present high contrast to the background and are thus identified correctly to a great extend. Of the non-readable characters, only one is correctly identified.



Original

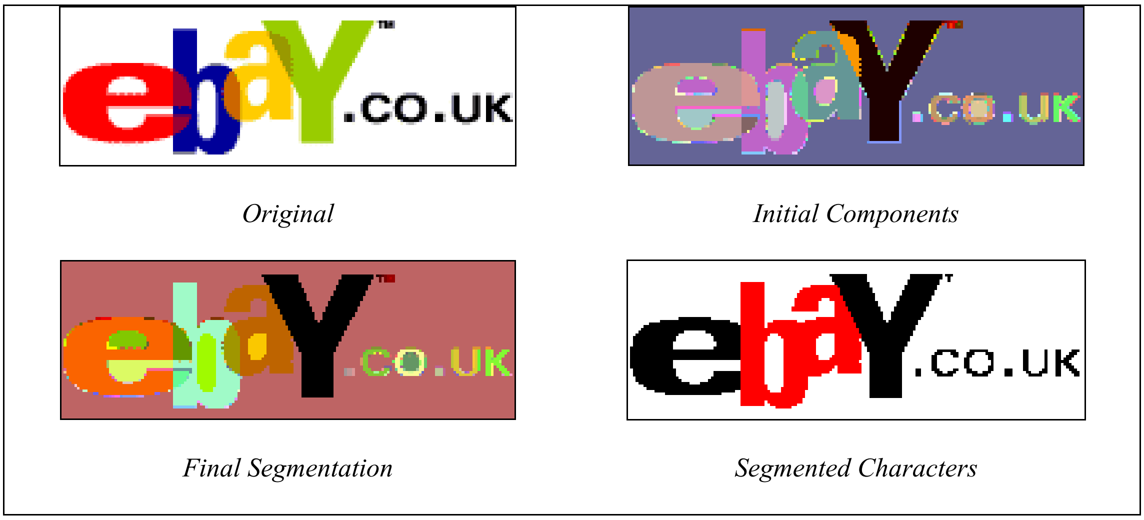
Initial Components

Final Segmentation

Segmented Characters

Figure 7-22 – An image containing multi-coloured characters (each “Google” character contains a number of different shades of its main colour) with shadows. The small characters on the right are classified as readable, while the “^{TM}” characters at the upper-right side of “Google” are classified as non-readable ones.

An image with overlapping characters can be seen in Figure 7-23. The fuzzy method is able to segment half of the characters in this method. It finds the common parts of the characters equally different in colour from both characters they belong to, thus it does not assign them to any character specifically. In numerous cases, the resulting characters maintain their overall shape (e.g. character “*e*” above), and are thus considered identified for recognition purposes. In contrast, characters like “*b*” and “*a*” above, whose overall shape is not maintained in a single component, are not considered identified. As for the non-readable characters “^{TM}” at the upper right side of the logo, one out of the two is correctly identified.



Original

Initial Components

Final Segmentation

Segmented Characters

Figure 7-23 – An image with overlapping characters.

Overall, the Fuzzy segmentation method, proves to perform very well in the difficult categories A and B, while its performance in the easier categories C and D is not substantially higher. The distinction between difficult and easy to segment images is not very clear here, where the method performs around *70%* for three of the categories, while category A does not fall much behind with *60%*.

7.2.3. Comparison of the two Segmentation Methods

In this section, a comparison between the two segmentation methods will be attempted. The results of both methods will be compared, while specific examples will be given in terms of individual images of the data set.

Two key differences exist in terms of the performance of the two segmentation methods. The first difference lies to the types of images (categories) each method seems to be strong at, while the second major difference is the time of execution of the methods.

Specifically, while the Split and Merge segmentation method performs poorly in the difficult categories A and B, the Fuzzy method performs much better. On the other hand, the Split and Merge method’s performance rises substantially when it comes to the easier categories C and D, while the Fuzzy segmentation method maintains a rather stable performance. In Figure 7-24, an immediate comparison is made between the identification rates of the methods for each category.

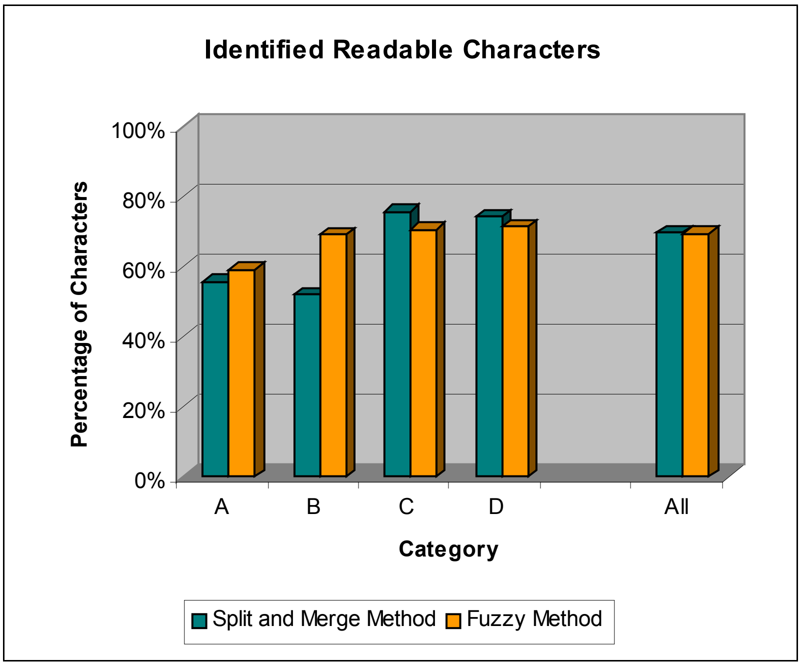


Figure 7-24 – Comparison between the identification rates of the two segmentation methods. The Split and Merge method performs poorly at the first two categories, while its performance rises at the other two. The Fuzzy method presents the opposite behaviour.

Time wise, the Fuzzy segmentation method is far more efficient with average execution time ~*15* seconds, in comparison to ~*5* minutes for the Split and Merge method. Both methods perform quite fast (less than *10* seconds) with small images (less than *5000* pixels), which represent a great percentage of the images found in the Web. A comparison between the execution times of the two segmentation methods is shown in Figure 7-25.

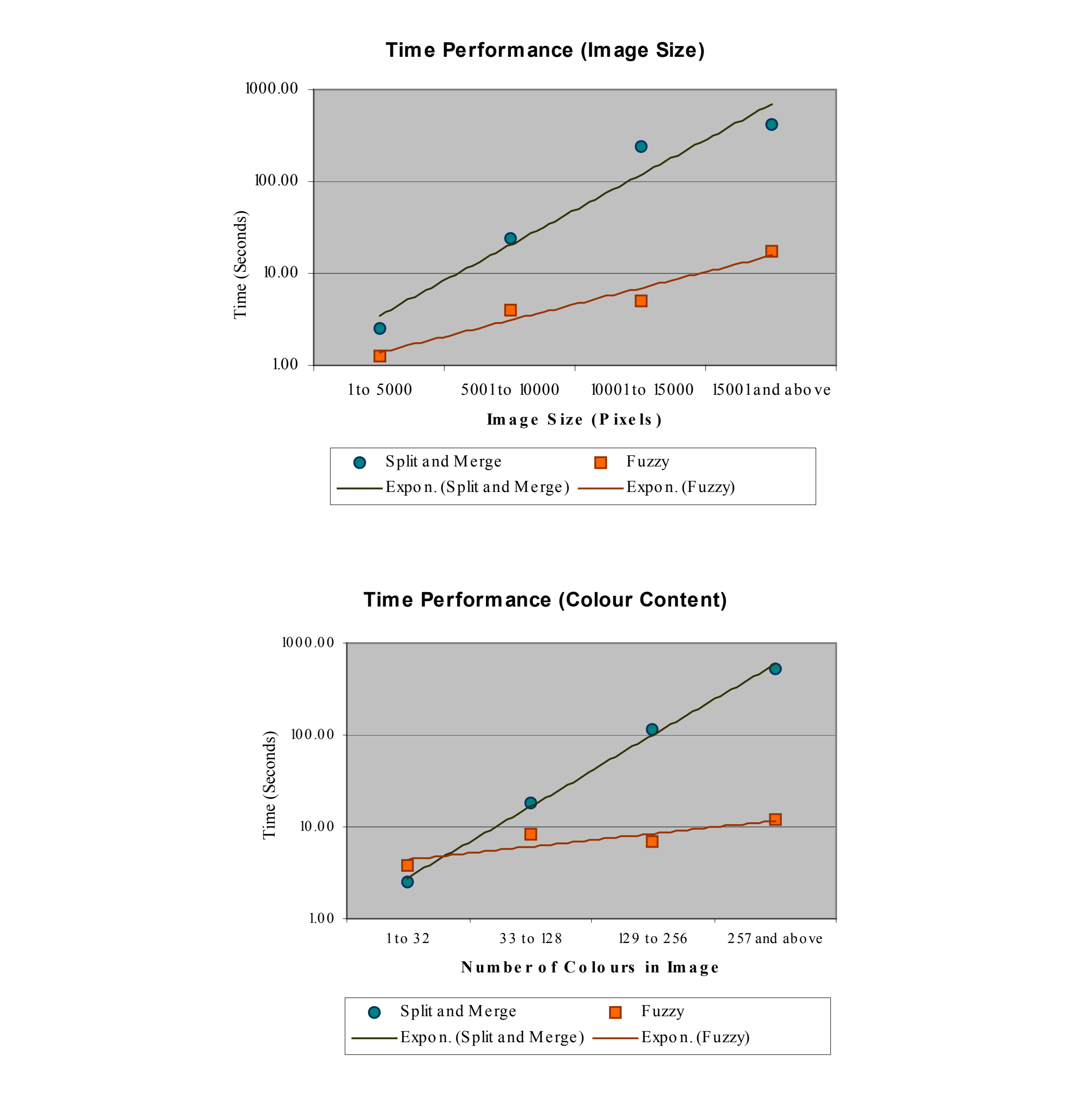


Figure 7-25 – Comparison between the execution times of the two segmentation methods. The effect of different image sizes and different colour contents is demonstrated.

Concerning other characteristics of the methods, we can say that overall, the Split and Merge method shows a tendency to over-merge medium sized components, comparably to the Fuzzy method which is more preservative. An example can be seen in Figure 7-26.

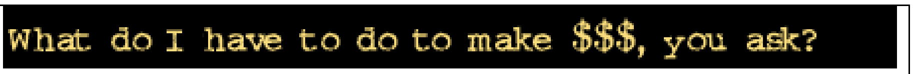
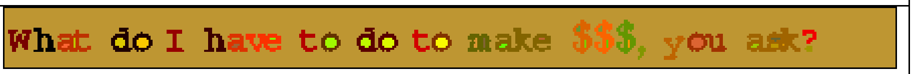
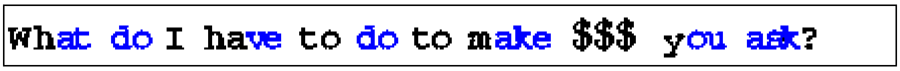
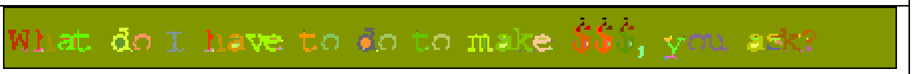
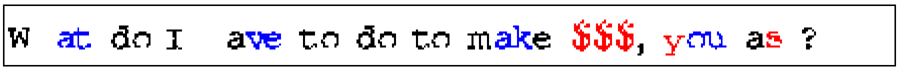
Original Image	
Split and Merge	 
Fuzzy Method	 

Figure 7-26 – An example of an image where the merging tendency of the Split and Merge method is evident. In this case, this results in a better final segmentation.

In many cases like in Figure 7-26, the over-merging tendency of the Split and Merge method results to a better final segmentation. Nevertheless, there are cases such as the one presented in Figure 7-27 where unwanted mergers between characters take place. Even worse, in cases like the one depicted in Figure 7-28, parts of the background can be merged with the characters. In most of those cases, the Fuzzy segmentation method gives far better final segmentations.

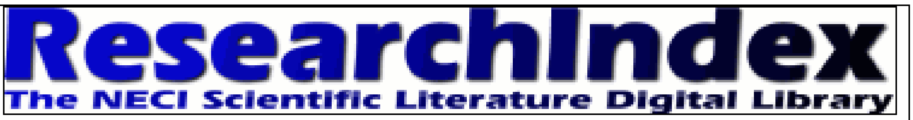
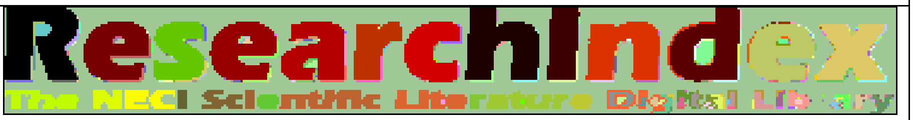
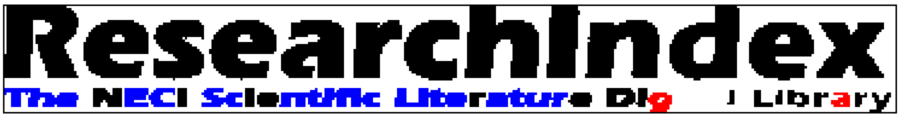
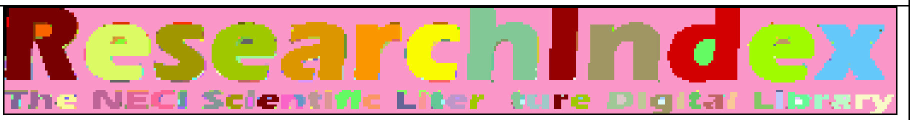
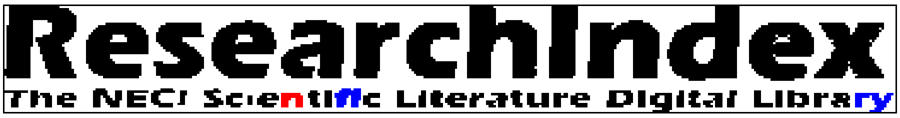
Original Image	
Split and Merge	 
Fuzzy Method	 

Figure 7-27 – Another example of an image where the merging tendency of the Split and Merge method is evident. In this case, the Split and Merge method erroneously merges characters together.

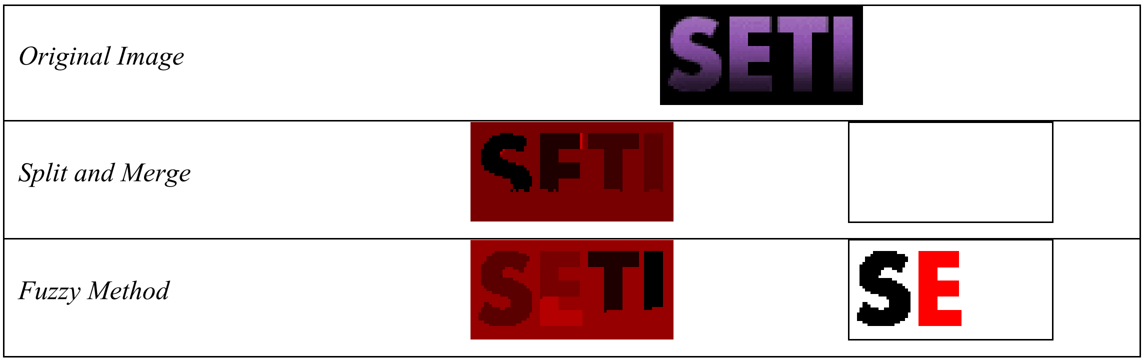


Figure 7-28 – An example of an image with gradient text. The Split and Merge method merges parts of the character with the background.

Generally, the Fuzzy segmentation method produces much "cleaner" segmentations. The reason for that is that the Fuzzy segmentation method appears to deal much better with the small components of the image. While the Split and Merge method shows a tendency to merge medium sized components, the Fuzzy method has a tendency to merge the small ones. By doing so, two aspects of the final segmentation are affected. First, the small components surrounding characters (like parts of the outlines, the antialiasing areas or the shadows) are merged with either the character or the background, leaving in many cases thinner characters (e.g. Figure 7-26 and Figure 7-29), but also producing a much cleaner output. The fact that the final segmentations produced by the Fuzzy method are much cleaner is reflected at the results of the Character Component Classification process as will be discussed next.

The second, and most important effect of this tendency of the Fuzzy method, is that in most of the cases the non-readable (extremely small) characters of the images are much better segmented by the Fuzzy method, than the Split and Merge one. An example of this can be seen in Figure 7-29. Further proof for this comes by comparing the performance of the two methods for the non-readable characters of the images. As can be seen in Figure 7-30 (page 239), the rates for the correct identification of non-readable components are always higher for the Fuzzy segmentation method, rather than the Split and Merge one.

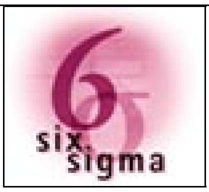
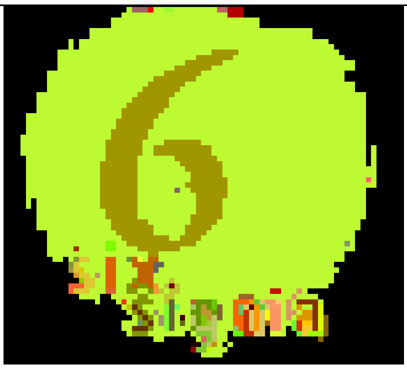
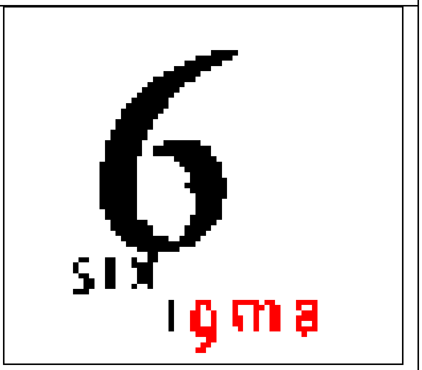
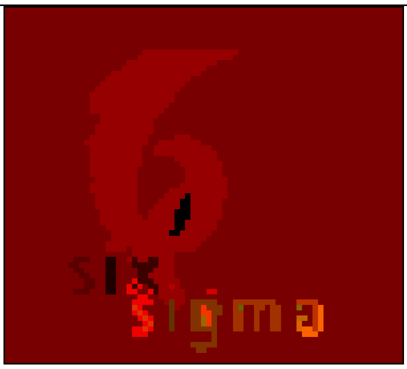
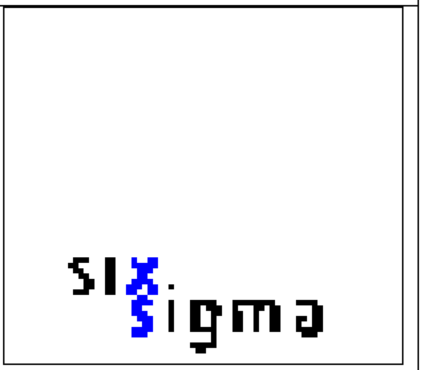
Original Image		
Split and Merge		
Fuzzy Method		

Figure 7-29 – An example of an image with multi-coloured text and background (The results are presented magnified). The Split and Merge method better identified the number "6", whereas the Fuzzy method is more effective in identifying the small components.

Overall, both methods are able to correctly identify around *70%* of the readable characters in the images of the dataset. The main advantage of the Split and Merge segmentation method is its performance with the images containing single-coloured text. The Fuzzy segmentation method on the other hand is better with images containing multi-coloured text. If one method had to be singled out as the best of the two, that has to be the Fuzzy segmentation method mainly due to its much better execution time.

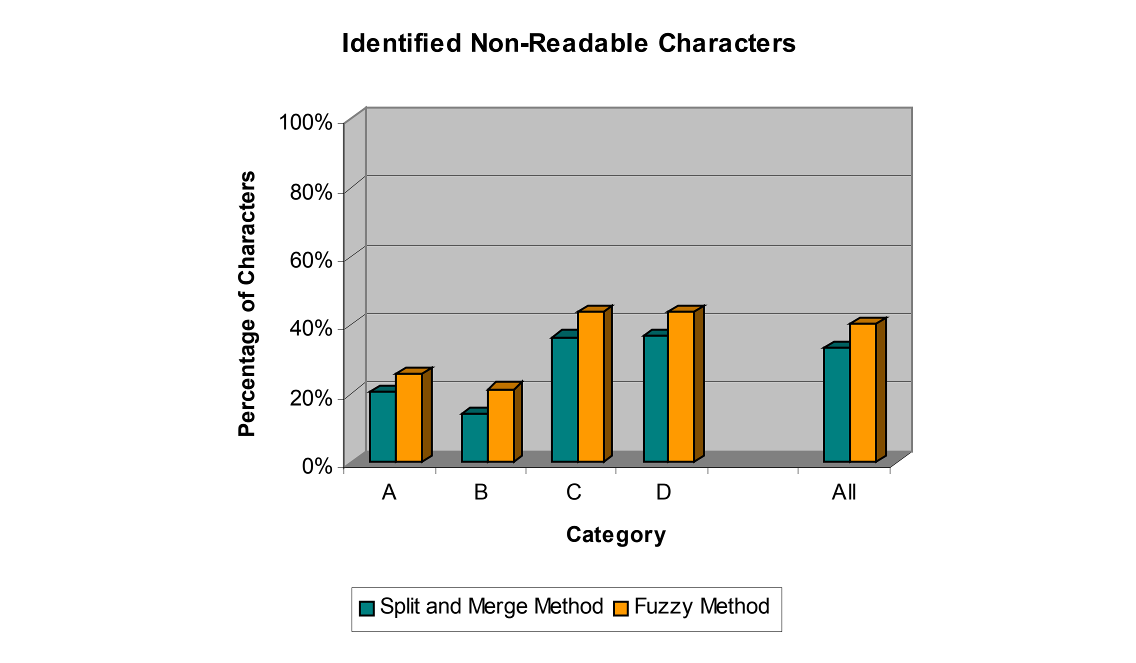


Figure 7-30 – Comparison between the identification rates for the non-readable components of the two segmentation methods.

7.3. Character Component Classification Results

The Character Component Classification (CCC) process, automatically classifies components as character or non-character ones. As described in Chapter 6, the CCC process analyses the image components by first separating them into size groups, and then identifying lines of components that resemble lines of text in each size group. It is evident, that due to the way the CCC process works, it is essential that it starts with a good segmentation of an image. If during the segmentation a character is split or merged with neighbouring ones, the CCC process will have difficulties identifying it for a number of reasons. First, the size of the components associated with the character will be either smaller (if a character is broken) or larger (if it is merged with other characters) than the size of the rest of the characters on the same line. A second issue is that even if all the components associated with characters were exported in the same size group, the distances between them will vary too much to be considered parts of a text line (see Section 6.2.3).

For these reasons, in order to assess the CCC process on its own without reporting accumulated errors from the segmentation process, only a subset of the images was used. The images used are the ones for which the segmentation process was able to correctly identify *100%* of the readable characters. Furthermore, an extra restriction was set: that the images contain at least *3* characters. This lower limit of *3* characters is a restriction set by the CCC process itself. The total number of images that conform to these specifications is *58* out of which *31* were segmented using the Split and Merge method and *27* using the Fuzzy method. All four categories are represented in this set of images.

For this set of images, the CCC process was run, and a number of lines were exported. Each exported line was assessed as either a text or non-text one. The total number of components classified as characters were counted, as well as the number of them that actually represented characters. The CCC process can identify a single component as part of more than one text line. Special attention was paid so that each component is counted only once, no matter how many lines it participates in. Measures for recall and precision for the CCC method were calculated based on Eq. 7-1 and Eq. 7-2, where C is the set of connected components that correspond to characters and I is the set of connected components classified as characters by the CCC process.

$$P = \frac{|C \cap I|}{|I|} \qquad \textit{Eq. 7-1}$$

$$R = \frac{|C \cap I|}{|C|} \qquad \textit{Eq. 7-2}$$

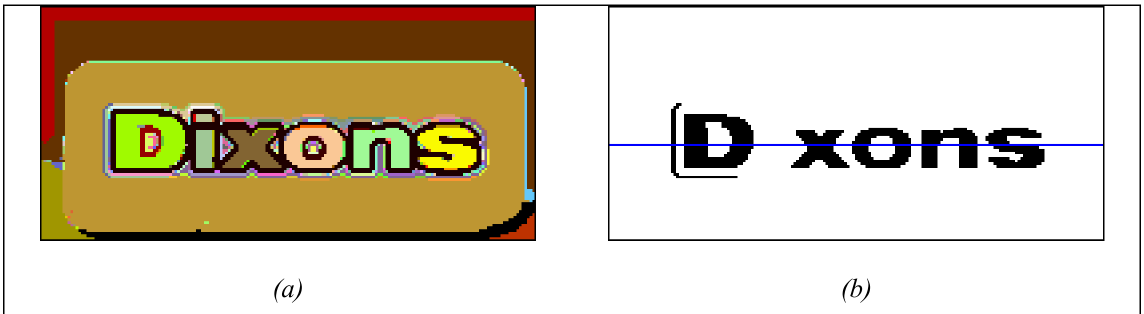
The overall precision and recall was measured, as well as separate measures for the segmentations produced with the Split and Merge method, and for the segmentations produced with the Fuzzy method. The results are shown in Table 7-6.

Segmentations produced with	***Precision***	***Recall***
Split and Merge method	53.37%	87.94%
Fuzzy method	82.66%	84.36%
All methods	62.84%	86.38%

Table 7-6 – Precision / Recall results for segmentations produced by the two segmentation methods.

As can be seen, the recall is above *80%* for segmentations produced by either segmentation method. The precision on the other hand, is quite affected by the method used for segmentation. Specifically very low precision measures are obtained for segmentations produced with the Split and Merge method, while precision above 80% is obtained for segmentations produced by the Fuzzy method. The main reason for that is that the Fuzzy segmentation method gives much cleaner results than the Split and Merge segmentation method.

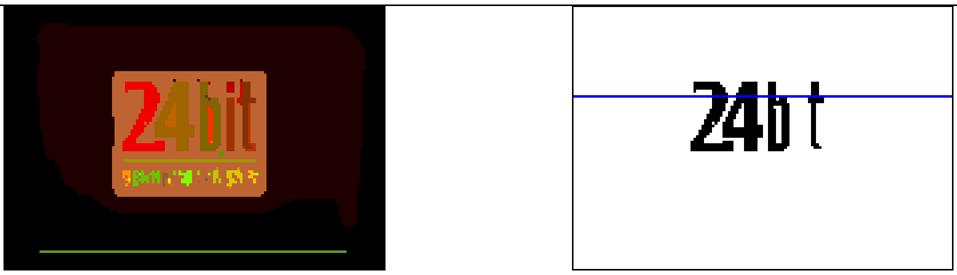
A low precision measure indicates that a great percentage of the components classified as character-like, are actually parts of the background. Most of such background components that were falsely classified as characters are actually parts of the shadows of the characters, the outlines of the characters, or finally internal parts of closed characters (the holes of letters such as "o" or "B"). As far as it concerns the latter, they appear at the same degree in both segmentations produced by the Split and Merge method, and the Fuzzy method. The components representing parts of the shadows or outlines of characters on the other hand, tend to be suppressed by the Fuzzy segmentation method, but not by the Split and Merge segmentation method (discussed in Section 7.2.3). An example of a line where such components have been falsely classified as characters is shown in Figure 7-31.



(a) *(b)*

Figure 7-31 – (a) A segmentation produced with the Split and Merge method. A number of components associated with the outlines of the characters can be seen. (b) One of the text lines extracted. A non-character component is erroneously classified as text.

A second less important problem of the CCC method is the handling of characters having much smaller (or larger) diagonal size than the average size of the rest of the characters on the same line. Such characters, would not be considered in the same size group as the rest of the characters on the text line, and therefore would be missed when the text line is identified. The character missed most of the times is character “i”, as illustrated in Figure 7-32. Although the size ranges for the diagonals were selected so that all characters of the same font would be grouped together, there are some cases where very small (or very large) characters are not grouped correctly.



(a) *(b)*

Figure 7-32 – (a) A segmentation produced by the Fuzzy method. (b) One of the extracted lines identified as text. Character “i” is missing here because it was not considered as part of the same size group as the rest of the characters on the line.

It should be mentioned at this point, that two-part characters (like “i” and “j”) are split in two components by the segmentation methods. The CCC method, aims to classify the bigger part of such characters, without any special consideration for the dots above them. The dots of such characters can be retrieved at a later post-processing stage of the CCC method. Such a post-processing stage is not currently implemented.

Concerning the time performance of the CCC method, it never takes above *0.007* seconds on a Pentium IV running at 1.8GHz.

7.4. Discussion

In this chapter, the results of both segmentation methods and the Connected Component Classification method were presented and critically appraised. The dataset of images used for the assessment was also described and the categorization used for the images was explained. Finally, possible applications of the methods to domains other than Web Text Extraction were studied.

Overall, the segmentation methods yield character identification rates around *70%*, which is a very promising result. The Split and Merge method works better for easier images containing single-coloured characters, while the Fuzzy method performs better in the most difficult categories containing multi-colour characters. The main advantage of the Fuzzy method is the low execution time it takes.

The connected component classification method gives a rather low precision rate of around *60%*, mainly caused by the existence of too many background components in segmentations produced by the Split and Merge segmentation method. The recall on the other hand was quite high, with an average above *80%*.

A comparison of the results presented here with other text extraction methods is rather difficult. Most methods existent for extracting text from colour images, are not specifically designed for Web Images, but for other types of colour documents: real life scenes, video frames etc. As a consequence, there are but limited reported results concerning the application of such methods to web images specifically.

Of the very few methods dedicated in Text Extraction from Web Images, concrete results are given in the work of Lopresti and Zhou [106, 220]. Nevertheless, their method is focused (and consequently optimised) to GIF images, containing a maximum of *256* colours. Moreover, they limit the assessment of their method to a subgroup of Web Images which meet the assumptions set for their method: they contain homogeneously coloured text and have horizontal text layouts. Finally, they only report cumulative results obtained after both segmentation and component classification stages are completed. The average character detection rate reported in [106] is *78.8%*, which (taking into account the restricted type of images they used), is not much higher than the results of the methods presented here.

It is rather ungrounded to perform a direct comparison between the two methods, since the segmentation methods presented here are optimised in such a way so that reasonable results are obtained in a number of fundamentally different images, in contrast to the specific type of images tested in [106]. However, for completeness, a comparative graph between the results reported by Lopresti and Zhou, and the results of our segmentation methods is shown in Figure 7-33.

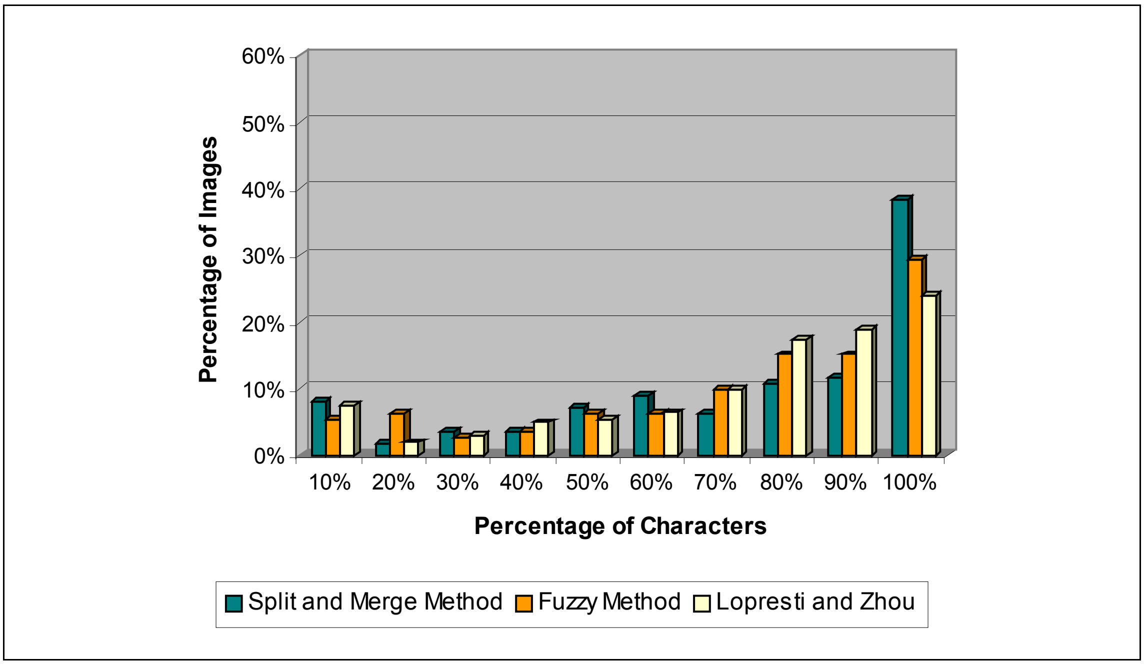


Figure 7-33 – Comparison chart between the two segmentation methods presented and the Text Extraction method of Lopresti and Zhou.

In terms of future possibilities and possible extensions to the segmentation and component classification methods, there are a number of points that can be addressed. The Split and Merge method is susceptible of a number of optimisations that can improve vastly the execution time of the method. Most optimisations are related to the connected component internal representation, and the calculations of the new components and associated vexed areas after merging. Apart from that, further research can be done in colour discrimination thresholds for a number of colour components and subsequently better thresholds can be defined. Moreover, other colour systems except HLS can be tried for the implementation of the method.

An idea introduced for the Split and Merge method, that could prove useful to a variety of other applications is the concept of vexed areas. Vexed areas are defined as possible extensions of connected components based on colour similarity. The existence of such possible extensions can be useful during the recognition phase, as they can provide extra information for a given component. An example of such a case was discussed previously (Section 7.2.1), when the common part of two characters was identified as a vexed area for both the corresponding components. In addition, for complicated images (e.g. where the borders of the characters are not precise), vexed areas could be used for the definition of ground truth information. The vexed area of a component would be a possible part of it, but it should not be considered wrong if a method did not identify it as such. For example, the outline of a character can be part of the vexed area of the component representing the character, as well as the part of the vexed area of the component representing the background, indicating that this area (the outline) can be assigned to either.

As far as it concerns the Fuzzy segmentation method, the majority of possible extensions and optimisations are focused to the fuzzy inference system used, and the definition of propinquity. A number of connected component features other that connections ratio and colour distance can be incorporated to the definition of propinquity. Given two components, features indicative of the validity of a merger between them are sought. Such features would be the colour variance of the resulting component, their orientation, some measure of character likeness before and after a merger etc. Furthermore, the fuzzy inference system can be optimised by evaluating it with different combinations of membership functions for each of the input features used. An example of such an optimisation procedure based on genetic algorithms is given in [17]. Of course, for such automatic optimisation methods to be used, ground truth information and a method of assessing the result are necessary in order to produce the necessary feedback for the optimisation process. A number of methods are available for assessing document segmentations, most based on bounding box representations [6]; nevertheless, the construction of ground truth information is an open problem.

As far as it concerns the Connected Component Classification method, a number of optimisations can be identified. The main problem in using most connected component features for classification in the case of Web Images, is the plethora and variety of components associated to the background, that resemble character components. Due to that, features used in a number of other methods failed to give good results in this case. However, in combination to the existent approach (which produces quite a high recall rate), and given the existing information about the direction of each identified text line, connected component features can be used at a post-processing stage to further eliminate components, or to include more components to each text line. The need for further evaluation of the components exported is established by the low precision rate and the high recall rate of the method. Examples of features that could be used are the number of black-to-white or white-to-black transitions, the number of strokes per line (given the direction of the text line, a maximum of *4* strokes should be found -characters “M” and “W”- in any component line), the number of holes in a component etc. Furthermore, vertical profiling of the text line could provide an extra method of checking for non-character components erroneously exported.

An overall view of the problem of text extraction form Web Images, reveals that most of the cases that present a problem to the current implementations are associated with complicated colour schemes and text renderings (small characters, antialiasing etc). Better handling of complicated colour schemes can be achieved by employing non colour oriented information. This was attempted here by the use of topological properties between components to further facilitate connected component aggregation. Problems associated with small or antialiased text, can be possibly addressed by improving the resolution of certain parts of the image (effectively enlarging the characters). Some methods for doing this were described in Chapter 3.

Chapter 8

Conclusion

This thesis examined the problem of text extraction from Web Images. A summary of the contents of this thesis will be given next, followed by a discussion based on the aims and objectives set at the beginning. The main contributions and limitations of this research are detailed in the last section along with a summary of future possibilities suggested and new directions identified. Finally, the applicability of the methods presented to other domains, except Web Image text segmentation, is investigated.

8.1. Summary

This thesis was divided in three parts. First, the theoretical background of the problem was presented. The description of the new methods developed followed, and the thesis closed with an evaluation of the methods presented.

After a brief introduction to the problem, and the establishment of the aims and objectives of this research in Chapter 1, the theoretical background for this research was given in Chapters 2 and 3. Chapter 2 gave a literature review of segmentation methods, focused mostly on methods created for colour images. Chapter 3 specialised in text image segmentation and classification of the results. Previous work on web text extraction was also presented in Chapter 3.

The segmentation and classification methods were described in Chapters 4 to 6. Chapter 4 detailed the first segmentation method, which works in a split and merge fashion, based on human colour discrimination information. In Chapter 5 the second segmentation method developed was described. This segmentation method is based on a fuzzy defined propinquity value, which is a metric of closeness between components that takes into account both colour distance and topological information between them. Chapter 6 addressed the problem of classifying the connected components produced by the segmentation method as character or non-character ones. The classification method developed assesses lines of similar sized components, and decides whether they resemble lines of text or not.

Finally, an evaluation of the two segmentation methods and the connected component classification method was performed in Chapter 7. A dataset of images collected from the numerous web pages was introduced, and the segmentation methods were tested against the images in the dataset. The connected component classification method was subsequently tested on segmentations resulted from both the split and merge and the fuzzy segmentation methods.

8.2. Aims and objectives revisited

This research aimed in identifying novel ways to extract text from images used in the World Wide Web. Towards this, two segmentation methods and one classification method were developed. The methods developed were built on observations made over a number of web images.

The segmentation methods presented are generally able to cope with complicated colour schemes, including gradient text and background, and various instances of antialiasing. Both segmentation methods are able to correctly segment the characters contained in the images at around *70%* of the cases. At the same time, the classification method, presents a high recall (around *80%*) and an average precision rate (around *60%*).

The assumptions of the method were kept to an absolute minimum. The only assumption made for the segmentation method was that the text in every image is written in such a way that a human being is able to read it. This mainly entails that the contrast between the text and the background, should be higher than the contrast between pixels of the same class. The anthropocentric character of both segmentation methods ensures that human based colour discrimination information is employed at all times (either explicitly by the use of appropriate colour spaces, or implicitly by the use of experimentally derived colour discrimination information).

The assumptions made for the connected component classification method are also kept to a minimum. The only assumptions made here, is that characters are placed on a straight line and that characters on a line are of similar size. Nevertheless, the minimum number of characters required for a line to be identified is set to three. By being able to identify lines of that few characters, the classification method can effectively identify curved lines and lines containing characters of varying size, as long as they can be split in sub-lines of at least three characters each. Most curved lines can be identified in this way, as well as most characters of varying character size, making the classification method effectively unaffected to size and co-linearity constraints.

The main negative point of the segmentation methods developed is the execution times they yield. The Fuzzy segmentation method is the faster of the two, with an average time of around *15* seconds per image, while the Split and Merge method performs poorly with an average of *5* minutes per image (Pentium IV at 1.8GHz). Generally, both methods take less than *10* seconds for small images (less than *5000* pixels), which represent a great percentage of the images in the Web. Furthermore, the code used here is not optimised, and a number of optimisations have been described that could improve vastly the execution times of both methods. In any case, execution time is the biggest disadvantage of the segmentation methods developed. The classification method on the other hand is quite fast, with execution times that never rise above *0.007* seconds.

8.3. Application to Different Domains

In addition to the evaluation of the described segmentation methods to Web Images, a number of tests were performed in order to obtain a view of the strengths and difficulties those methods present when applying in other fields. For this purpose, a variety of images was used: images of book covers, photographs containing text (real life scenes) and video frames containing captions. A number of images were segmented with the use of the Fuzzy segmentation method.

The main problem that the method presents when used to segment coloured scanned book and magazine covers is the handling of dithered areas. Due to the way coloured publications are printed, dithering is extensively used to produce colours. The segmentation method cannot handle correctly dithered areas, because the dots used to create the dither are of dissimilar colour as can be seen in Figure 8-1. In such cases, the initial connected component analysis produces a vast number of components, of inadequately similar colour for the aggregation stage to combine, and the segmentation fails. A solution to that problem would be to downscale the image by averaging over neighbours of pixels. That would produce areas of similar colour, but also smaller characters.

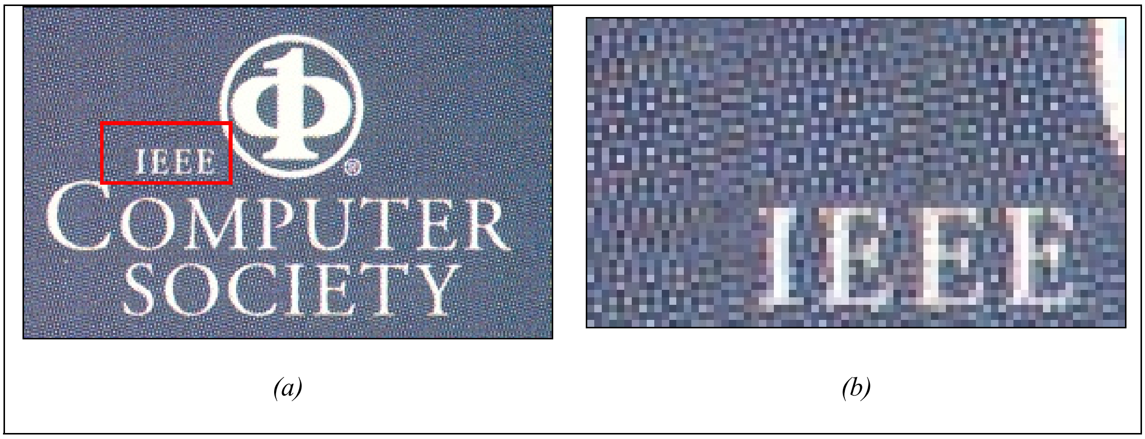


Figure 8-1 – (a) Part of a scanned magazine cover. (b) Magnified region, where dithering is visible.

Instead of scanning and downscaling a number of book covers, we opted at testing the segmentation method with already downscaled images of book covers obtained from “amazon.com”. This online business displays small-sized (*~350* x *475* pixels) versions of the scanned covers for each book offered, so a huge dataset of downscaled book covers is readily available. The segmentation method was able to correctly segment an average of around *80%* of the characters in those images, while the execution time was on average *40* seconds for each image.

Another test performed was with video frames containing superimposed text. Video frames are very similar to web text in many aspects. The segmentation method was able to correctly identify an average of *70%* of the characters in such images. The average execution time for video frames (*352* x *240*) was *8* seconds.

Finally, a few real scene photos were segmented, which contained signs and car plates. The photos tested were of size *800* x *600* or larger. On average, the method was able to identify around *60%* of the characters, while the execution time for the method was on average *5* minutes.

The most significant problem of segmenting photos and scanned documents is the size of the document. In its current state the implementation of the method is rather slow for documents of sizes above *500* x *500* pixels. Apart from that, the method seems able to cope with a variety of images, at performances similar to the ones obtained for web images. The only significant problems identified were associated with the use of dithering in scanned images. It has to be mentioned that the results presented in this section are by no means precise, since only *5* to *10* images of each category were tested, and the statistical errors are therefore high.

8.4. Contributions of this Research

The main contributions of this research to the document analysis field are listed below.

- The problem of excessive amount of text being embedded in images in the context of the World Wide Web was analysed. This thesis offers a detailed description of the problem.
- The special characteristics of Web Images were detailed, and a comprehensive dataset of Web Images was created for the evaluation of the method. This dataset can prove useful as a first step towards creating ground truth information for Web Image analysis. The statistical information obtained from the dataset is representative of the current state of the World Wide Web, as far as it concerns the use of images in Web Pages.
- Two anthropocentric segmentation methods were proposed, specifically developed for text extraction from Web Images. Both segmentation methods developed are able to cope with complicated colour schemes: an improvement over a number of existing colour text extraction methods.

- A connected component classification method was also developed, which is optimised to the problem of Web Image analysis. A vast number of connected components are generally produced for Web Images as a result of the complicated colour schemes used. The classification method is designed to cope with this situation by assessing lines of components, instead of individual ones.
- Future possibilities and new directions for research were identified. Possible optimisations for the existing methods were suggested, and the main requirements for similar research were clearly laid out.

To summarize, the text extraction (segmentation and classification) methods presented in this thesis address a number of problems associated with Web Images. Nevertheless, there are certain cases which the methods developed cannot cope with. Most of these cases are associated with complicated colour schemes and text renderings. Future work should focus on those extreme situations.

Equally important for future text extraction methods developed, is the enlargement of the dataset used and the creation of associated ground truth information. This will prove an important tool for improving and optimising existing and future methods.

This thesis presented novel methods for extracting text from Web Images. A number of key issues were successfully addressed, although some problems still exist. The methods discussed here make a significant contribution to the field of Web Document Analysis, by suggesting new ways for making the text embedded in Web Images accessible for further analysis.

Appendix A

Human Vision and Colorimetry, A discussion on Colour Systems

This appendix gives an overview of Human Vision and Colour Perception and presents a number of colour systems widely used. Basic definitions and concepts of colorimetry are also given here. Finally, a detailed description of the experiments conducted to determine the discrimination thresholds for Lightness, Hue and Colour Purity (Saturation) is described and results are presented.

A.1. Human Vision and Colorimetry

Colour is the perceptual result of light having wavelength from *400* to *700nm* that is incident upon the retina. To be more precise, when referring to the perceptual result, the term *Perceived Colour* should be used, while when referring to the characteristic of the radiant energy causing this perceptual result the term *Psychophysical Colour* should be used.

A.1.1. Human Vision

The human retina has two types of photoreceptor cells: the *cones* and the *rods*. The cones are effective in daylight conditions (photopic vision), whereas the rods are effective only at extremely low light intensities (scotopic vision). There are three types of colour photoreceptor cone cells, which respond to incident radiation with different spectral response curves. Because there are exactly three types of colour photoreceptors, three components are necessary and sufficient to describe a colour, providing that appropriate *spectral weighting functions* are used.

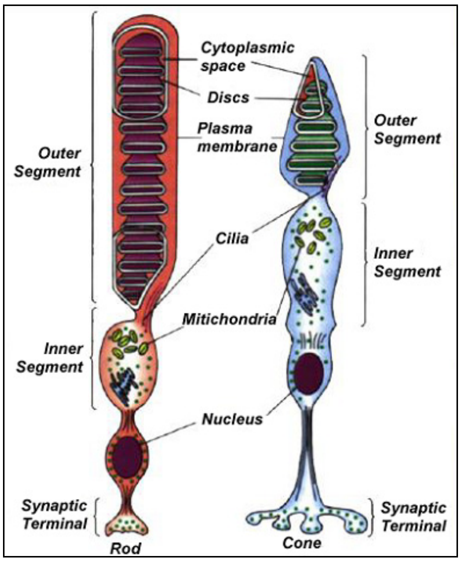


Figure A-1 – The two types of photoreceptor cells found in the human eye: rods and cones.

As far as it concerns the rods, as there is only one type of rod cell, (chromatic) colour is not perceived during "night vision". The number of rods in the human eye is approximately $110 – 125 \cdot 10^6$, while the number of cones is approximately $4 – 6.8 \cdot 10^6$.

A.1.2. Colorimetry

Colorimetry is the branch of colour science concerned, in the first instance, with specifying numerically the colour of a physically defined visual stimulus in such a manner that:

When viewed by an observer with normal colour vision, under the same observing conditions, stimuli with the same specification look alike,

Stimuli that look alike have the same specification and

The numbers comprising the specification are continuous functions of the physical parameters defining the spectral power distribution of the stimulus.

Colour matching is the process of finding the amounts of *n* reference lights required to give by additive mixture a match with the colour considered. There are three central questions introduced by the above sentence: how many reference lights are needed, which exactly reference lights should be used, and what amounts of them need to be mixed to produce a given colour.

The answer to the first question is straightforward: colour vision is inherently trichromatic; therefore, a set of three appropriate spectral weighting functions is sufficient for colour matching. The reference lights mentioned before are directly defined upon the spectral weighting functions. The objective here is to choose the spectral weighting functions in such a way so that different colours are represented by different sets of values (denoting the amounts of the reference lights needed to be mixed to produce the colours).

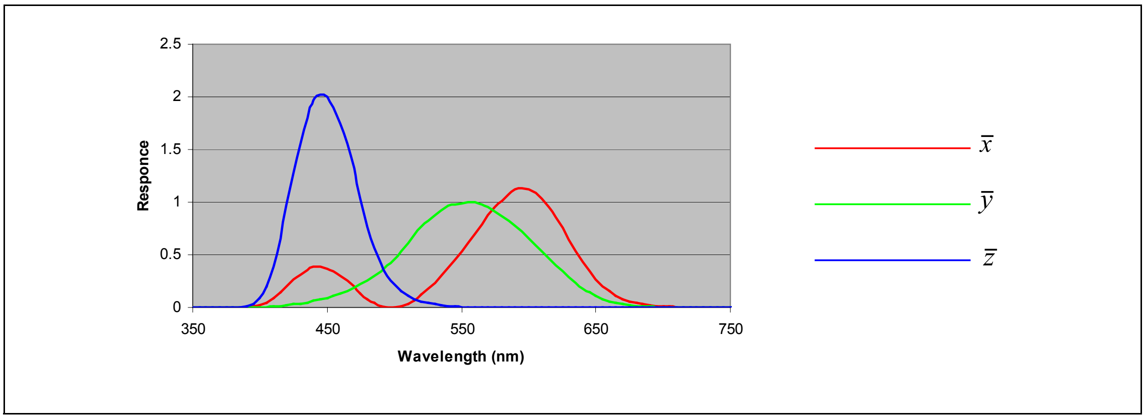


Figure A-2 – The CIE's colour matching functions for the standard observer.

The CIE (Commission Internationale de l'Eclairage or International Commission on Illumination) standardized a set of spectral weighting functions that models the perception of colour. These curves, defined numerically, are referred to as the $\bar{x}$, $\bar{y}$, and $\bar{z}$ colour matching functions for the CIE standard Observer. Given a continuous spectral power distribution, CIE's *X*, *Y* and *Z tristimulus values* are computed by integrating the spectral power distribution using the $\bar{x}$, $\bar{y}$, and $\bar{z}$ weighting functions. The CIE designed their system so that the *Y* value has a spectral sensitivity that corresponds to the lightness sensitivity of human vision.

A.1.3. Colour Discrimination

The ability to distinguish between colours is affected by numerous factors. The principal factors influencing colour discrimination are summarized in Table A-1 [181].

Factor	***Change***	***Ability to Discriminate Colours***
Wavelength Separation	▲	▲
Colour Purity	▲	▲
Brightness	▲	▲
Colour Stimulus Size	▲	▲
Brightness Adaptation Level	▲	▲
Number of Colours	▲	▼
Display Background		
Light		▲
Dark		▼
Colour Stimulus Location		
Central		▲
Peripheral		▼
Type of Discrimination Required		
Relative (comparative)		▲
Absolute (identification)		▼

Table A-1 – Principal factors affecting the ability to distinguish between colours.

A.2. Colour Systems

Throughout history, different ways to systematically order colours were sought by scientists, artists and philosophers in various contexts. Aristotle was probably the first to investigate colour mixtures. Aristotle's linear sequence of colours is the first attempt to construct a system of colours. It can be observed during the course of day: the white light of noon becomes tinged with yellow, and changes gradually to orange and red. After sunset, it becomes purple, then dark blue, and finally black.

Other attempts to organize colours in a colour system were made by numerous scientists, artists and philosophers, including Plato, Pythagoras, Leonardo da Vinci, Newton, Goethe, Maxwell and more recently by Munsel, the CIE (Rösch, MacAdam, Stiles) etc. The main function of a colour system is to code a given colour in such a way that it can be accurately reproduced at a later time.

The vast majority of colour reproductions do not attempt to reconstruct the spectral composition of the original colours, but only to elicit in the retina's three different types of cones the same or similar responses.

A thorough presentation and comparison of colour systems is out of the scope of this appendix. The colour systems presented here are only the ones used in this thesis and limited information is given: the transformations used, and main characteristics.

A.2.1. RGB

The *RGB* colour system (Figure A-3) is used for colour reproduction in Cathode Ray Tube (CRT) displays. Colours in CRTs are produced by mixing three lights of different colours (red, green and blue), produced by the phosphors of the screen. The three components of the *RGB* colour system denote the amount of light of each colour that needs to be mixed. Depending on the technical and physical characteristics of the CRT display, only a certain gamut of colours can be produced. The largest range of colours will be produced with primaries that appear red, green and blue, and that is the reason why phosphors producing colour stimulus with exactly those primaries are employed. Nevertheless, since there are no standard primaries and no standard white point *RGB* information alone is not adequate to reproduce the original colours if the hardware changes.

	R	***G***	***B***	***White***
x	0.640	0.300	0.150	0.3127
y	0.330	0.600	0.060	0.3290
z	0.030	0.100	0.790	0.3582

Table A-2 - Primaries and D_{65} white point of Rec. 709

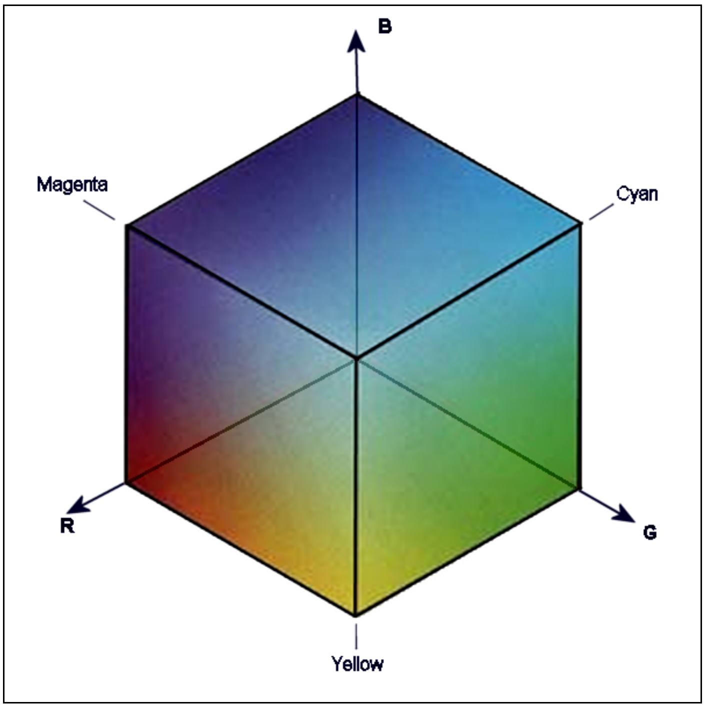


Figure A-3 – The RGB Colour Space.

Although contemporary CRT displays have slightly different standards around the world, international agreement has been obtained on primaries for high definition television (HDTV), and these primaries are closely representative of contemporary displays. The standard is formally denoted *ITU-R Recommendation BT.709* (Table A-2). The *sRGB* colour system, is defined as part of this standard, as a colour independent extension of the *RGB* colour system. Use of hardware information ensures the accurate reproduction of colours across different hardware configurations. If hardware information is not available, one can use the *RGB* values as *sRGB* ones, based on the assumption that the hardware used conforms to *Rec. 709* within some tolerance (which is usually the case). This provides us with a standard way to transform between *RGB* values and CIE *XYZ*, by using the transformations defined for *sRGB* (Eq. A-1, Eq. A-2).

$$\begin{bmatrix} R_{709} \\ G_{709} \\ B_{709} \end{bmatrix} = \begin{bmatrix} 3.240479 & -1.537150 & 0.498535 \\ -0.969256 & 1.875992 & 0.041556 \\ 0.055648 & -0.204043 & 1.057311 \end{bmatrix} \cdot \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} \qquad \textit{Eq. A-1}$$

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} 0.412453 & 0.357580 & 0.180423 \\ 0.212671 & 0.715160 & 0.072169 \\ 0.019334 & 0.119193 & 0.950227 \end{bmatrix} \cdot \begin{bmatrix} R_{709} \\ G_{709} \\ B_{709} \end{bmatrix} \qquad \textit{Eq. A-2}$$

A.2.2. HLS

The *RGB* colour system is hardware oriented. Numerous other colour systems exist which are user-oriented, meaning that they are based on more human oriented concepts. Colour systems like *HSV*, *HLS*, *HSI* etc fall in this category. Such systems usually employ three components, namely *Hue*, *Saturation* and *Lightness* (or *Value*, or *Intensity*, or *Brightness*), corresponding to the notions of tint, shade and tone. Specifically Hue represents the “redness”, “greenness”, “blueness” etc of the colour and corresponds to the colorimetric term *Dominant Wavelength*. Lightness denotes the perceived *Luminance* (as a matter of fact, for self-luminous objects the correct term is Brightness). Saturation denotes the purity of the colour, corresponding to the colorimetric term Excitation Purity. The lower the Saturation, the closer the colour is to grey.

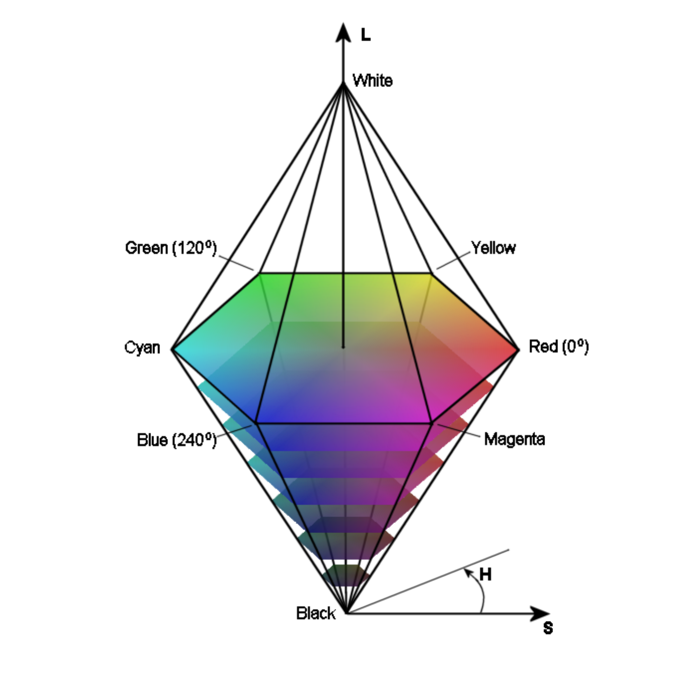


Figure A-4 – The HLS double hexcone colour space..

The *HLS* colour system (Figure A-4) is one such system used in this thesis. It is defined in the double-hexcone subset of a cylindrical space. Hue is the angle around the vertical axis of double hexcone, with red at *0^o* and colours occurring counter clockwise in the order: red, yellow, green, cyan, blue and magenta. The complement of any Hue in this system is located *180^o* farther around the double hexcone. Saturation is measured radially from the vertical axis (value of *0*) to the surface (value of *1*). Lightness is *0* for black, which occurs at the lower tip of the double hexcone, and *1* for white, at the upper tip.

All greys in the *HLS* system have Saturation equal to zero, while the maximum saturated colours occur for Saturation equal to *1* and Lightness equal to *0.5*. The Hue component of the *HLS* colour system is expressed as an angle, so it presents a circular repetition. The *2π*-modulus nature of the Hue component can be an advantage or a disadvantage in different properties. The principle problems appear during computations with the Hue components. Any type of clustering or histogram analysis has to take this circular repetition in account to correctly process *HLS* data.

The transformations from *RGB* to *HLS* are given below:

$$L = \frac{\max(R,G,B) + \min(R,G,B)}{2} \qquad \textit{Eq. A-3}$$

$$S = \begin{cases} 0, & \textit{if} \max(R,G,B) = \min(R,G,B) \\ \dfrac{\max(R,G,B) - \min(R,G,B)}{\max(R,G,B) + \min(R,G,B)} & \textit{if } L \le 1/2 \\ \dfrac{\max(R,G,B) - \min(R,G,B)}{2 - \max(R,G,B) - \min(R,G,B)} & \textit{if } L > 1/2 \end{cases} \qquad \textit{Eq. A-4}$$

$$H = \begin{cases} \textit{undefined}, & \textit{if } R = G = B \\ \left(\dfrac{G - B}{\max(R,G,B) - \min(R,G,B)}\right) \cdot \dfrac{\pi}{3} & \textit{if } R = \max(R,G,B) \\ \left(2 + \dfrac{B - R}{\max(R,G,B) - \min(R,G,B)}\right) \cdot \dfrac{\pi}{3} & \textit{if } G = \max(R,G,B) \\ \left(4 + \dfrac{R - G}{\max(R,G,B) - \min(R,G,B)}\right) \cdot \dfrac{\pi}{3} & \textit{if } B = \max(R,G,B) \end{cases} \qquad \textit{Eq. A-5}$$

The inverse transformations are as follows:

$$\alpha = \begin{cases} L \cdot (1+S), & \mathit{if}\ L \leq 1/2 \\ L + S - L \cdot S, & \mathit{if}\ L > 1/2 \end{cases}$$

$$\beta = 2 \cdot L - \alpha$$

Eq. A-6

If $S = 0$, then $R = G = B = L$.

If $0 < H \leq \pi/3$, then

$$R = \alpha, \qquad G = \beta + (\alpha - \beta) \cdot \left(\frac{H}{\pi/3} \right), \qquad B = \beta$$

If $\pi/3 < H \leq 2\pi/3$, then

$$R = \beta + (\alpha - \beta) \cdot \left(\frac{2\pi/3 - H}{\pi/3} \right), \qquad G = \alpha, \qquad B = \beta$$

If $2\pi/3 < H \leq \pi$, then

$$R = \beta, \qquad G = \alpha, \qquad B = \beta + (\alpha - \beta) \cdot \left(\frac{H - 2\pi/3}{\pi/3} \right)$$

If $\pi < H \leq 4\pi/3$, then

$$R = \beta, \qquad G = \beta + (\alpha - \beta) \cdot \left(\frac{2\pi - H}{\pi/3} \right), \quad B = \alpha$$

If $4\pi/3 < H \leq 5\pi/3$, then

$$B = \beta + (\alpha - \beta) \cdot \left(\frac{H + 2\pi/3}{\pi/3} \right), \qquad G = \beta, \qquad B = \alpha$$

If $5\pi/3 < H \leq 2\pi$, then

$$R = \alpha, \qquad G = \beta, \qquad G = \beta + (\alpha - \beta) \cdot \left(\frac{2\pi - H}{\pi/3} \right)$$

To transform between the *HLS* and CIE *XYZ* systems, an intermediate transformation to *RGB* (*sRGB*) is needed, then we use the matrix transformation given in section A.2.1.

A.2.3. CIE XYZ

The CIE *XYZ* system has already been described in the previous section. The three components of this system are the tristimulus values computed to match any given colour based of the $\bar{x}$, $\bar{y}$, and $\bar{z}$ colour matching functions for the CIE standard Observer. CIE *XYZ* colour space is illustrated in Figure A-5.

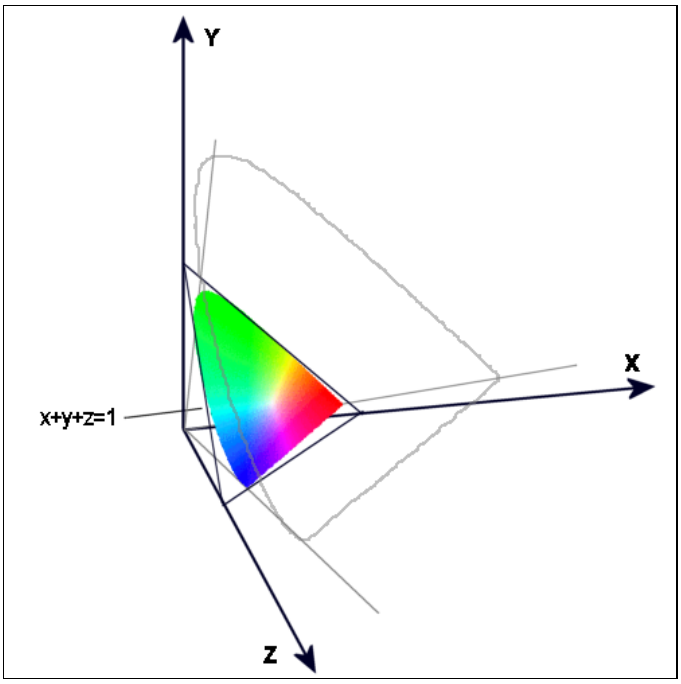


Figure A-5 – The CIE XYZ Colour Space.

It is convenient for both conceptual understanding and computation, to have a representation of "pure" colour in the absence of luminance. The CIE standardized a procedure for normalizing *XYZ* tristimulus values to obtain two *chromaticity* values *x* and *y* (Eq. A-7). A third chromaticity value, *z*, can be computed similarly, however it is redundant due to the identity $x + y + z = 1$.

$$x = \frac{X}{X+Y+Z},\ y = \frac{Y}{X+Y+Z},\ z = \frac{Z}{X+Y+Z} \qquad \textit{Eq. A-7}$$

Essentially the chromaticity diagram is the slice produced by the intersection of the CIE *XYZ* colour space and the plane: $x + y + z = 1$. The chromaticity diagram can be seen in Figure A-6.

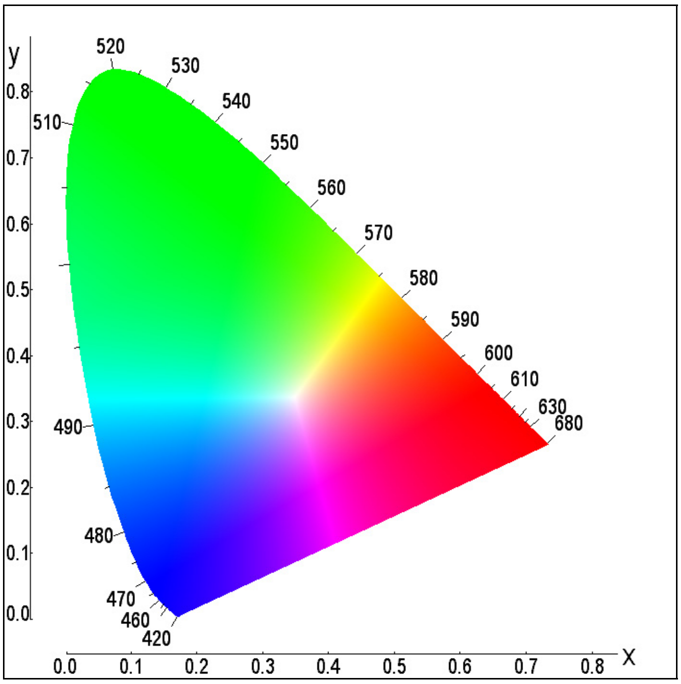


Figure A-6 – The Chromaticity Diagram (CIE 1931)

A.2.4. CIE LAB and CIE LUV

The CIE *XYZ* presents a problem: it is not perceptually uniform. Consider the distance of colour $C_1\,(X_1, Y_1, Z_1)$ to colour $C_1 + \Delta C$, and the distance from colour $C_2\,(X_2, Y_2, Z_2)$ to $C_2 + \Delta C$, where ΔC is the same in both cases and equal to $(\Delta X, \Delta Y, \Delta Z)$. Although the distance is the same in both cases, the perceived distance between the colours of each pair will not generally be the same. To address this problem, the CIE introduced in 1976 two perceptually uniform colour systems: CIE LAB (or CIE 1976 $L^*a^*b^*$) and CIE LUV (or CIE 1976 $L^*u^*v^*$).

The CIE LAB colour system is defined by Eq. A-8 with the constraint that $X/X_n,\ Y/Y_n,\ Z/Z_n > 0.008856$.

$$L^* = 116 \cdot \left(\frac{Y}{Y_n} \right)^{1/3} - 16$$

$$a^* = 500 \cdot \left[\left(\frac{X}{X_n} \right)^{1/3} - \left(\frac{Y}{Y_n} \right)^{1/3} \right] \qquad \textit{Eq. A-8}$$

$$b^* = 200 \cdot \left[\left(\frac{Y}{Y_n} \right)^{1/3} - \left(\frac{Z}{Z_n} \right)^{1/3} \right]$$

If any of X/X_n, Y/Y_n, Z/Z_n are smaller than *0.008856*, Eq. A-9 - Eq. A-11 can be used:

$$L_m^* = 903.3 \cdot \frac{Y}{Y_n}, \mathit{for}\ \frac{Y}{Y_n} \le 0.008856 \qquad \textit{Eq. A-9}$$

$$a_m^* = 500 \cdot \left[f\left(\frac{X}{X_n} \right) - f\left(\frac{Y}{Y_n} \right) \right] \qquad \textit{Eq. A-10}$$

$$b_m^* = 200 \cdot \left[f\left(\frac{Y}{Y_n} \right) - f\left(\frac{Z}{Z_n} \right) \right] \qquad \textit{Eq. A-11}$$

Where

$$f(x) = x^{1/3}, \quad x > 0.008856$$

$$f(x) = 7.787 \cdot x + \frac{16}{116}, \quad x \le 0.008856 \qquad \textit{Eq. A-12}$$

The tristimulus values X_n, Y_n and Z_n are those of the nominally white object-colour stimulus. (*Rec. 709* defines as standard white CIE's D_{65} standard illuminant, see Table A-2). The colour difference between two colour stimuli is calculated from:

$$\Delta \mathrm{E}_{ab}^* = \left[(\Delta L^*)^2 + (\Delta a^*)^2 + (\Delta b^*)^2 \right]^{1/2} \qquad \textit{Eq. A-13}$$

The CIE LUV colour system is defined by Eq. A-14.

$$L^* = 116 \cdot \left(\frac{Y}{Y_n} \right)^{1/3} - 16$$

$$u^* = 13 \cdot L^* \left(u' - u'_n \right) \qquad \textit{Eq. A-14}$$

$$v^* = 13 \cdot L^* \left(v' - v'_n \right)$$

As can be seen the definition of L* is the same here as in Eq. A-8 and the same constraint applies, namely if Y/Yn is less than 0.008856 Eq. A-9 should be used. The quantities u', v' and u'_n, v'_n are calculated from:

$$u' = \frac{4 \cdot X}{X + 15 \cdot Y + 3 \cdot Z} \qquad v' = \frac{9 \cdot Y}{X + 15 \cdot Y + 3 \cdot Z}$$

$$u'_n = \frac{4 \cdot X_n}{X_n + 15 \cdot Y_n + 3 \cdot Z_n} \qquad v'_n = \frac{9 \cdot Y_n}{X_n + 15 \cdot Y_n + 3 \cdot Z_n} \qquad \textit{Eq. A-15}$$

The tristimulus values X_n, Y_n and Z_n are similarly to CIE LAB those of the nominally white object-colour stimulus. The colour difference between two colour stimuli is calculated from:

$$\Delta \mathrm{E}^*_{uv} = \left[\left(\Delta L^* \right)^2 + \left(\Delta u^* \right)^2 + \left(\Delta v^* \right)^2 \right]^{1/2} \qquad \textit{Eq. A-16}$$

If two coloured lights C_1 and C_2 are mixed additively to produce a third colour, C_3, and the three of them are plotted on a chromaticity diagram, then C_3 should ideally lie on the straight line joining C_1 and C_2 at a position that can be calculated from the relative amounts of C_1 and C_2 in the mixture. The CIE 1932 (x, y) Chromaticity diagram (Figure A-6) and the CIE LUV derived (CIE 1976 UCS) chromaticity diagram (if u^* is plotted against v^* for constant L^*) exhibit this property, but not the CIE LAB based one. If L^* is constant, straight lines in the CIE 1931 (x, y) chromaticity diagram or in the CIE 1976 UCS diagram become, in general, curved lines in the (a^*, b^*) diagram.

In applications concerned with self-luminous objects, CIE LUV is more often used. However, in applications concerned with small colour differences between object colours CIE LAB has been used more than CIE LUV. Furthermore, studies indicate that the degree to which the luminance information is separated from the colour information in CIE LAB is greater than in other colour systems, such as CIE LUV, YIQ, YUV, YCC, XYZ [91].

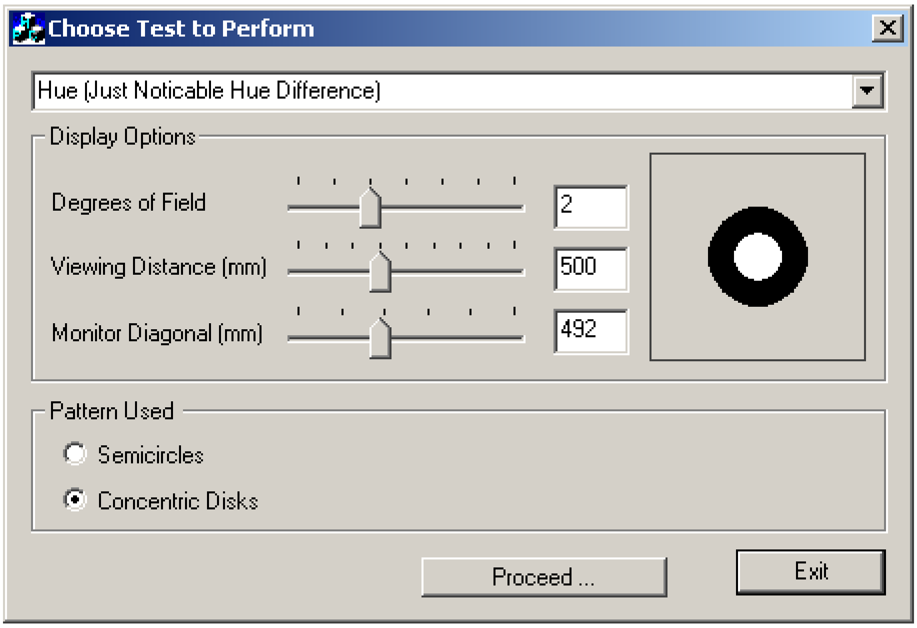
A.3. Colour Discrimination Experiments

This section details the experiments conducted to measure the least noticeable difference between colours based on the three components of the *HLS* colour system. The results presented here were used as the basis to select the appropriate thresholds for the aforementioned components in the Split and Merge segmentation method described in Chapter 4 of this thesis.

	Property	*Value*
Monitor		
	Temperature	9300K
	Phosphors	P22
	Make	HP D2846
	Gamma	2.4
	Diagonal	492mm (~19.4" viewable)
Graphics Card		
	Gamma	1.0
Apparatus		
	Field of View	2º
	Pattern Used	Concentric Disks
	Viewing Distance	500 mm

Table A-3 – Technical Specification of the Hardware Used

These experiments aim to measure discrimination thresholds for those components in realistic computing situations, and are always compared and assessed with corresponding biological information when available. Details on the hardware used (considered to be typical to the vast majority of computer systems used nowadays) are given in Table A-3. The Lightness discrimination experiments were repeated for a monitor temperature of *6500 ºK* as well as *9300 ºK*.



Hue (Just Noticable Hue Difference)

Degrees of Field 2

Viewing Distance (mm) 500

Monitor Diagonal (mm) 492

Pattern Used

○ Semicircles

● Concentric Disks

Figure A-7 – The configuration dialog for the discrimination experiments. The user has to select the type of test, set up the geometry and positioning of the computer/observer, and decide on the type of the field of view to be used.

Colour discrimination has been studied in various special investigations by presenting to the observer a bipartite visual field, the two parts of which are adjusted to be just perceptibly different in some special quantity such as Hue or Brightness. This practice is adopted here. The typical visual fields used are either a circular bipartite field with a vertical dividing line (two semicircles), or two concentric circular disks. The latter pattern is used for these experiments. The size of the outer disk is set to *2* degrees, while the size of the inner disk is set to *1* degree. It is noted that decreasing the field size generally results in poorer discrimination. Sizes close to *2* degrees have been used to obtain the biological data to which the measured thresholds are compared, so the selection of this size is also necessary to enable comparison.

Three observers participated in each of the experiments described here. The number of observers is not large, but is enough to give results that correlate to the existing biological data. Having mentioned that, in some of the experiments in the literature (e.g. wavelength separation [13]) less than three observers were involved.

A.3.1. Colour Purity (Saturation) Discrimination

This experiment aims to measure the threshold above which a purely achromatic colour becomes chromatic. Starting with black, white or a mid-grey achromatic colour and a fully saturated chromatic colour, the observer is requested to slide a bar representing the amount of the fully saturated colour mixed with the achromatic one, until a perceivable tint appears (Figure A-8).

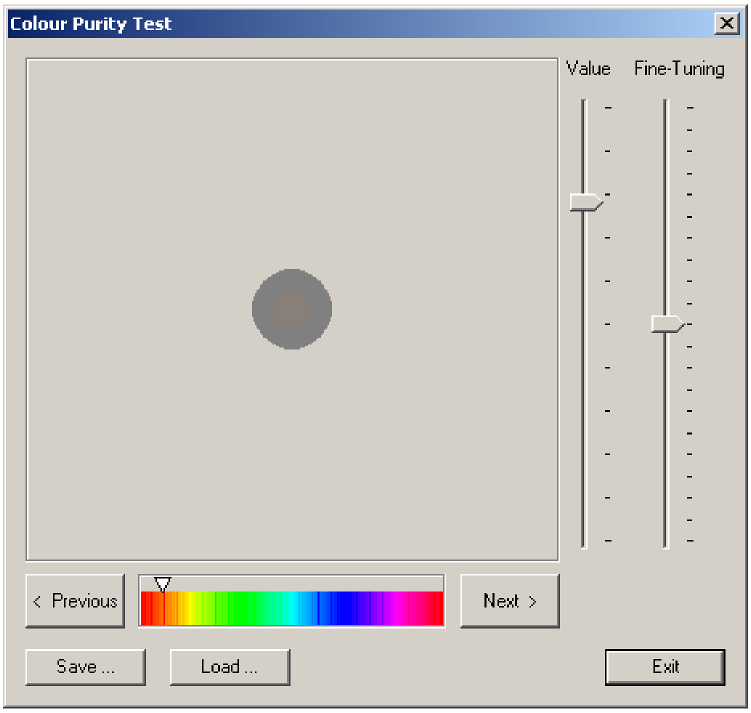


Figure A-8 – The colour purity test dialog. A 2-degree field is presented to the observer. The inside colour changes with the positioning of the sliders on the right, while the outside colour is either black, white or mid-grey (depending on the test).

The threshold where a tint appears depends both on the achromatic colour and on the Hue of the fully saturated colour used. For this experiment, the three achromatic colours mentioned before were tested: black, white and grey (Lightness = *128*, Saturation = *0*). Each of those colours was mixed with all (*256* values of Hue) possible fully saturated colours, and the threshold where the Hue became detectable was measured every time. The initial achromatic colour is presented to the observer as a two-degree field, while the new colour (created by sliding a bar) occupies the inner one-degree of the field. In other words, the observer is presented with two circles, the outer one maintaining the initial achromatic colour and the inner one showing each time the colour created by the movement of the bar. Since the change here is logarithmic, a second bar is also included for fine adjustment of the amount added.

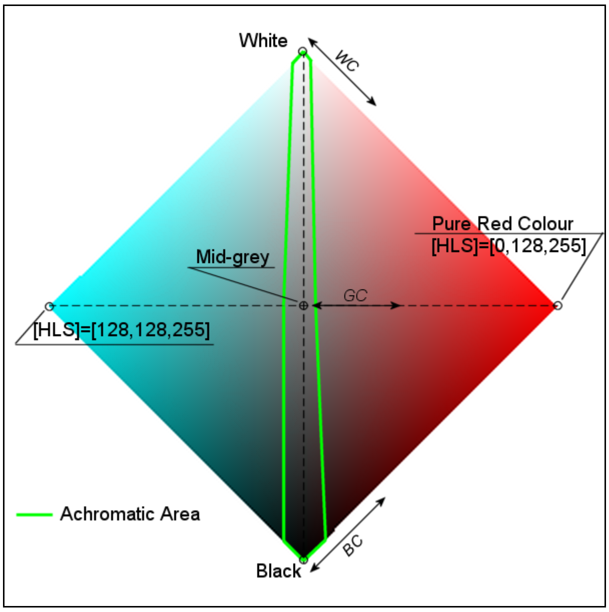


Figure A-9 – A vertical slice of the HLS colour system. The three directions on which measurements were taken namely White to Pure Colour (WC), Mid-grey to Pure Colour (GC) and Black to Pure Colour (BC) are illustrated in the figure. The green area signifies the area of achromatic colours as defined by experimental data.

Three sets of values are therefore obtained, which enable us to define the border surface between chromatic and achromatic pixels. The amount of pure Hue added to black and white for that Hue to become detectable, give information about colours at the edges of the Lightness component, while the amount added to grey, give information about the Saturation component. For illustrative purposes, a vertical slice of the *HLS* colour space is presented in Figure 4-5. The line that separates chromatic from achromatic colours for each Hue consists of four segments, connecting white, black and three points indicated by experimental results. These three points are along the lines that connect white to pure colour (WC), mid-grey to pure colour (GC) and black to pure colour (BC).

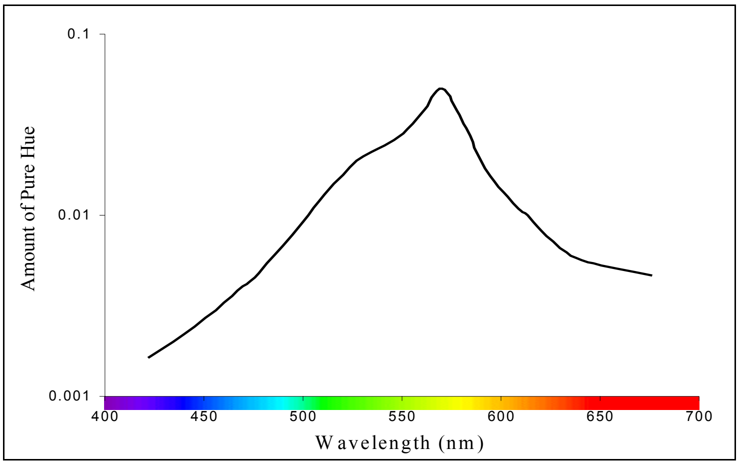


Figure A-10 – Amount of pure hue added to neutral colour before the hue becomes detectable.

Existing biological data describes the amount of pure Hue needed to be added to white before the Hue becomes detectable [126, 210]. This data is shown in Figure 4-2 and later on is compared to the measured thresholds. As explained in Chapter 4, these data only cover part of the range of Hues available. Furthermore, the experiments described here measure the amount of pure Hue needed to be added to black and to mid-grey before the Hue becomes detectable.

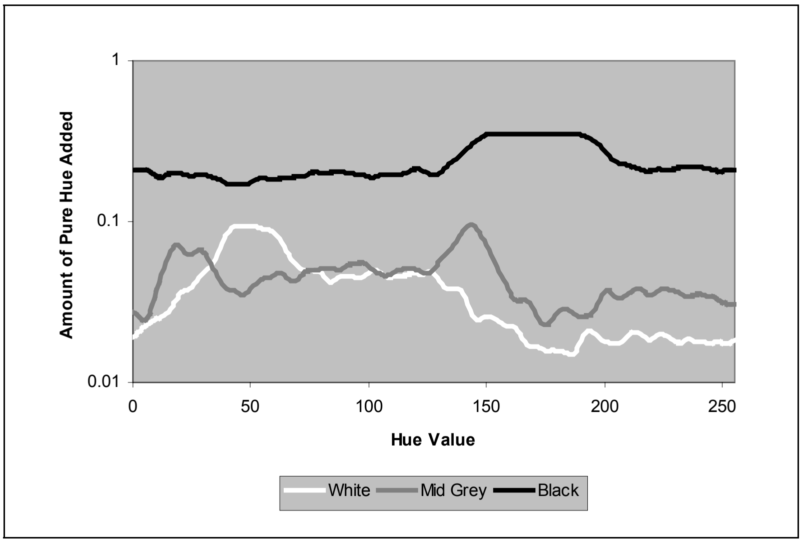


Figure A-11 – Amount of pure hue (fully saturated colour) added to white, mid-grey and black, before that hue becomes detectable.

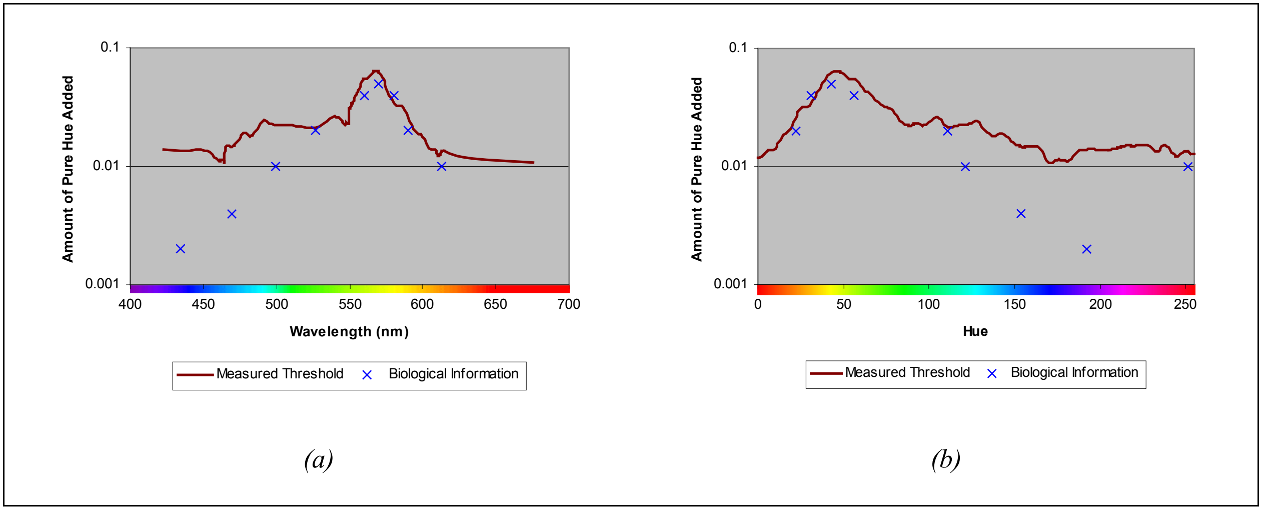


Figure A-12 – A comparison of the measured data (amount of pure hue added to white colour before that hue becomes detectable) to the biological data available. (a) X-axis is expressed wavelengths (only a portion of the measured data is visible). (b) X-axis is expressed in Hue values (biological data is not available for the range 194-240). Good correlation with the biological data can be seen for the range of wavelengths 525 to 650, while for wavelengths smaller than ~525nm biological data suggest stricter thresholds.

Three observers participated in the experiments. The average of the thresholds measured are presented here. The graphs shown here have been smoothed by use of the moving average. Figure A-11 shows the amount of pure Hue needed to be added to each of the achromatic colours tested before the Hue becomes detectable. In Figure A-12, a comparison is made between the biological data available and the thresholds measured here (the amount of pure Hue added to white for the Hue to become detectable). Finally, Figure A-13 shows a horizontal slice of the *HLS* system, for *L = 128* (at the level of the mid-grey colour tested), showing the thresholds measured.

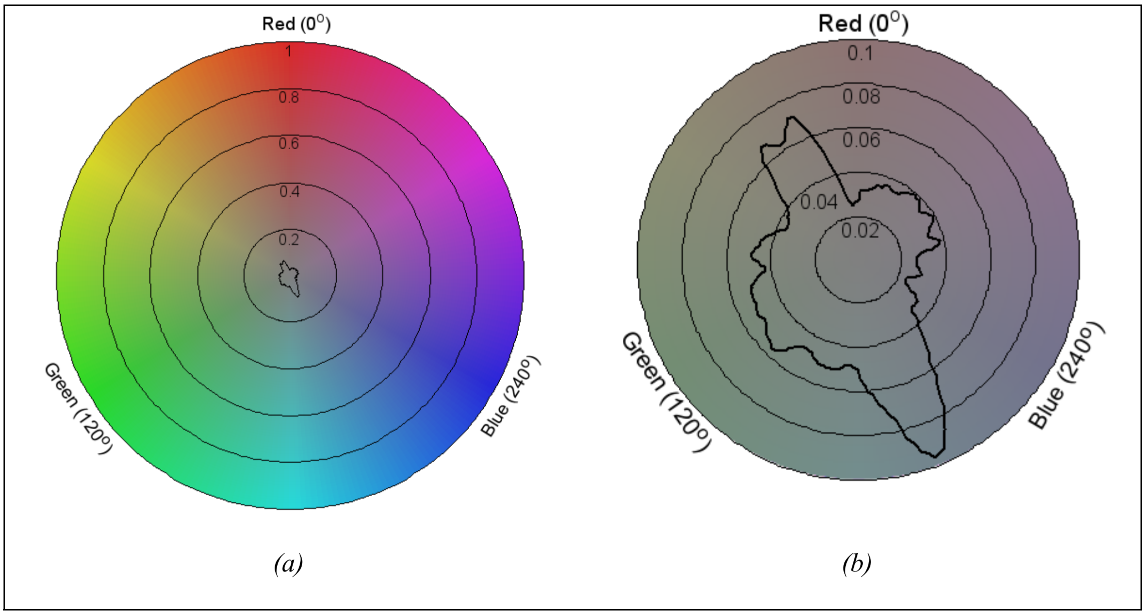


Figure A-13 – (a) A horizontal slice of the HLS colour space at L = 128. The centre area marks the colours considered achromatic based on the measurements made. A magnification of the centre area is shown in (b). The slices appear cyclical instead of hexagonal here for illustrative purposes.

A.3.2. Hue Discrimination

In a similar manner to the colour purity discrimination experiment, we measured the minimum change of Hue required for a detectable difference in Hue to appear. For each of the *256* possible fully saturated colours, the observer is presented with *2* two-degree concentric disk fields (Figure A-14). The outer disc of both shows the initial colour, while the inner discs show the colours created by the moving two sliding bars. The left pattern is used to judge when a detectable change occurs by reducing the Hue value, while the other by increasing the Hue value. Therefore, for each Hue value, two thresholds are measured: the positive and the negative change needed for a detectable change to occur. In the figures below, the average values for the thresholds are reported.

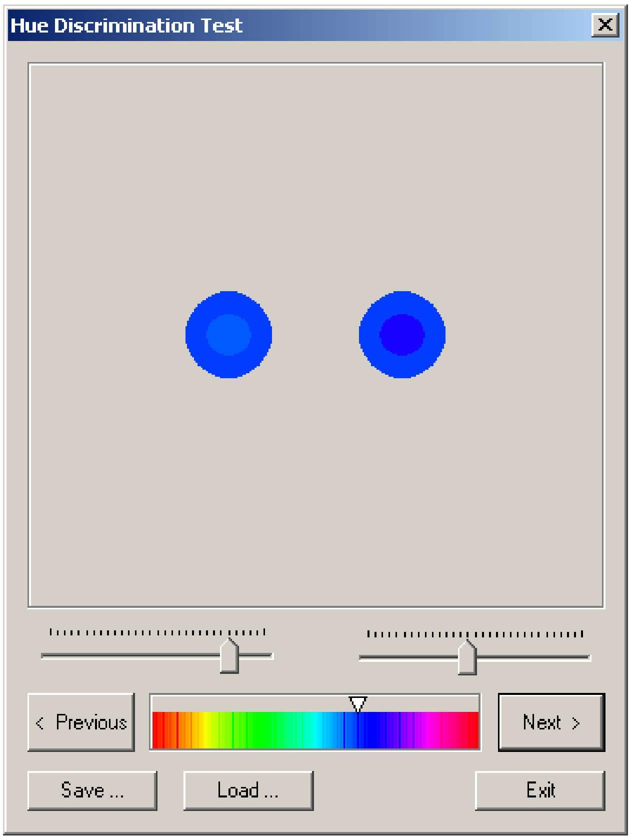


Figure A-14 – The Hue discrimination test dialog. Two 2-degree fields are presented to the observer. The inside colours change with the positioning of the sliders below the patterns, while the outside colour is the fully saturated colour for the Hue value selected from the colour bar.

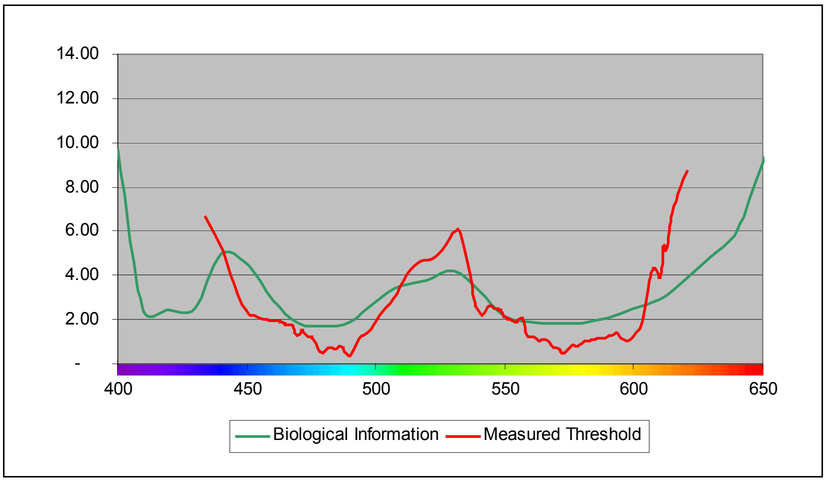


Figure A-15 – The average difference required for a detectable difference in Hue to appear.

The biological data available for wavelength discrimination in comparison to the measured Hue discrimination thresholds appear in Figure A-15 and Figure A-16. The strong correlation to the existing data is evident.

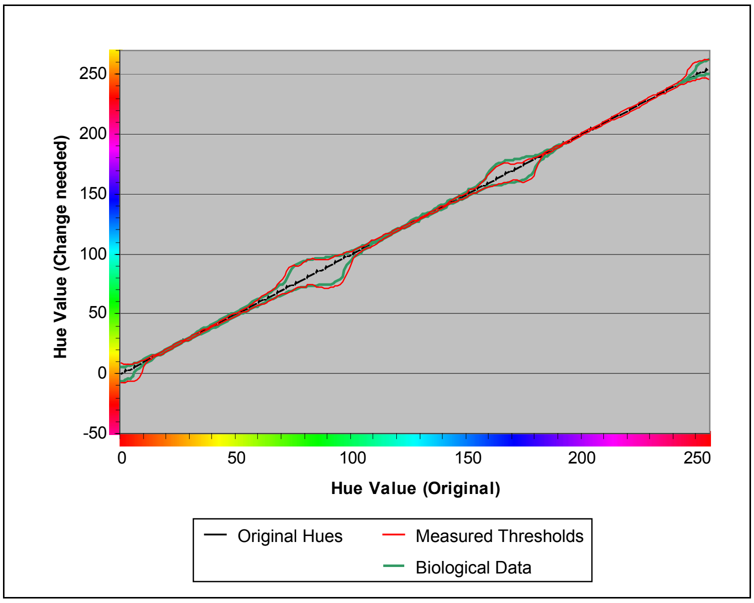


Figure A-16 – The positive and negative changes required for each Hue for a detectable difference in Hue to appear. Biological data do not exist for Hues in the range ~ 194-240.

A.3.3. Lightness Discrimination

The last experiment conducted aims to measure the increase in Lightness required for a detectable difference in Lightness to occur. For every shade of grey, the observer is presented with a 2-degree field similarly to the previous experiments. The outer disk appears in the shade of grey currently being tested, while the inner disk presents the colour selected by moving a slider (Figure A-17). The observer is asked to move the slider to the point that a detectable difference between the two disks first appears.

Before presenting the results of this experiment, it is useful to provide some definitions for Luminance, Brightness and Lightness. *Luminance* is a quantity used to describe the luminous power (flux) in a specified direction and at a specified point on a surface element of the source, or on a surface element of a detector placed in the specified direction at some distance away from the source. Luminance is a physical quantity measured in cd·m^{-2}.

According to Wyszecki and Stiles [211] "*Brightness* is defined as the attribute of a visual sensation according to which a given visual stimulus appears to be more or less intense, or according to which, the area in which the visual stimulus is presented appears to emit more or less light". The magnitude of Brightness can be estimated for *unrelated* visual stimuli, as well as for *related* visual stimuli. Unrelated visual stimuli are stimuli presented individually in isolation (darkness). Related visual stimuli are presented in a simple or complex display of other visual stimuli. The relation between brightness and luminance is described by a power law in the form $B = a \cdot L^{0.31} - B_0$. The values a and B_0 depend on the observing conditions (filed size, luminance of surround etc.).

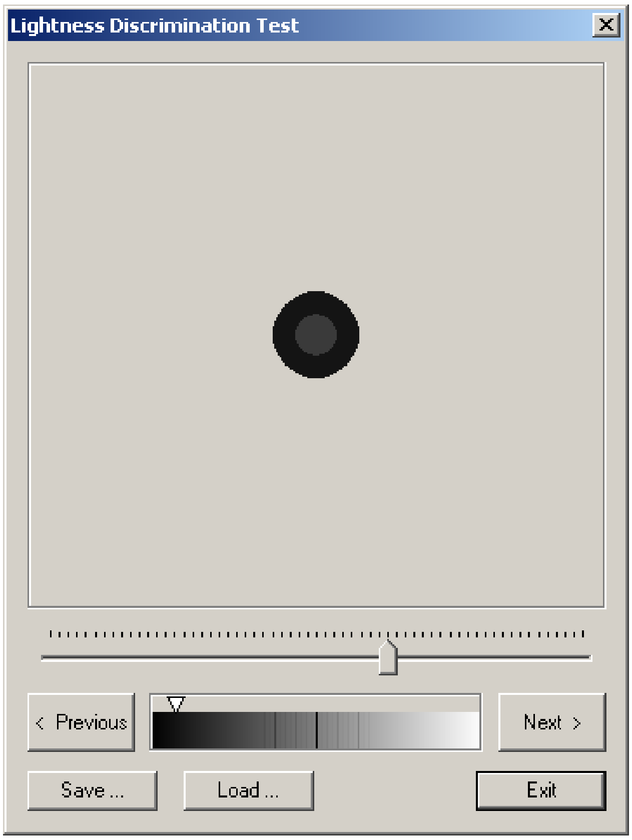


Figure A-17 - The Lightness discrimination test dialog. A 2-degree field is presented to the observer. The inside colour change with the positioning of the slider below the pattern, while the outside colour is the grey scale (Saturation=0) for the Lightness value selected from the grey-scale bar.

Lightness is defined as the attribute of a visual sensation according to which the area in which the visual stimulus is presented appears to emit more or less light in proportion to that emitted by a similarly illuminated area perceived as a "white" stimulus. Lightness is thus an attribute of visual sensation that has a meaning only for related visual stimuli. As Lightness is judged with reference to the Brightness of a "white" stimulus, it may be considered a special form of Brightness measure that could be referred to as relative Brightness. If a uniform Lightness scale is created (where the perceived difference between consecutive Lightness steps is the same), then the relationship between V (the ordinal value of the step, starting form zero for black) and luminance can be expressed by means of an empirical formula in the form of a power law.

On top of that, the luminance generated by a physical device is generally not a linear function of the applied signal. A conventional CRT has a power law response to voltage: luminance produced at the face of the display is approximately proportional to the applied voltage raised to the *2.4* (*2.1*, *2.4*, *2.5* are typical values for CRT monitors). The numerical value of the exponent of this power function is colloquially known as *gamma*.

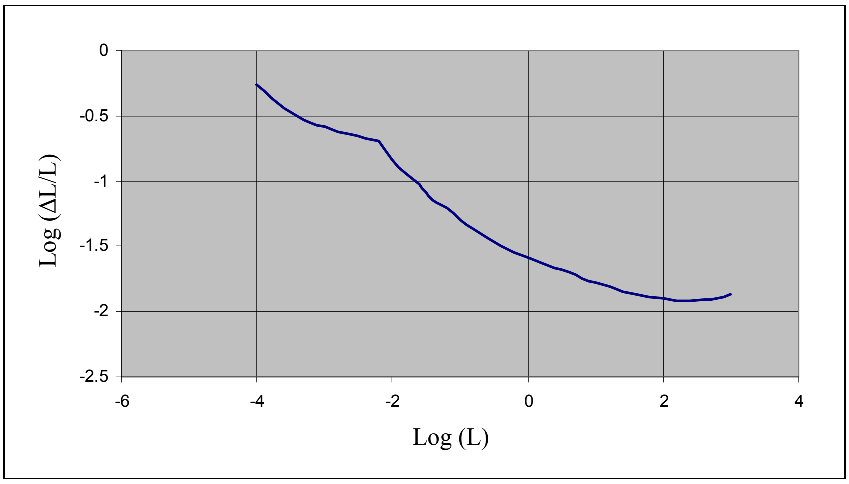


Figure A-18 – Discrimination of luminance differences.

The biological information available on discrimination is based on Luminance. Specifically, for each luminance level, the least change in Luminance was measured for which a detectable change occurred. Those biological results are presented in Figure A-18. The curve shown exhibits a sharp bend at a Luminance of roughly $5 \cdot 10^{-3}\ cd \cdot m^{-2}$. This break signifies that at Luminance levels below that value, judgements of luminance differences were effectively made by using parafoveal vision, strongly introducing the rod mechanism (“night vision” photoreceptor cells). Above this level, foveal vision is dominant, while for levels higher than $\sim 100\ cd \cdot m^{-2}$ the ratio tends to be constant and equal to *0.01*.

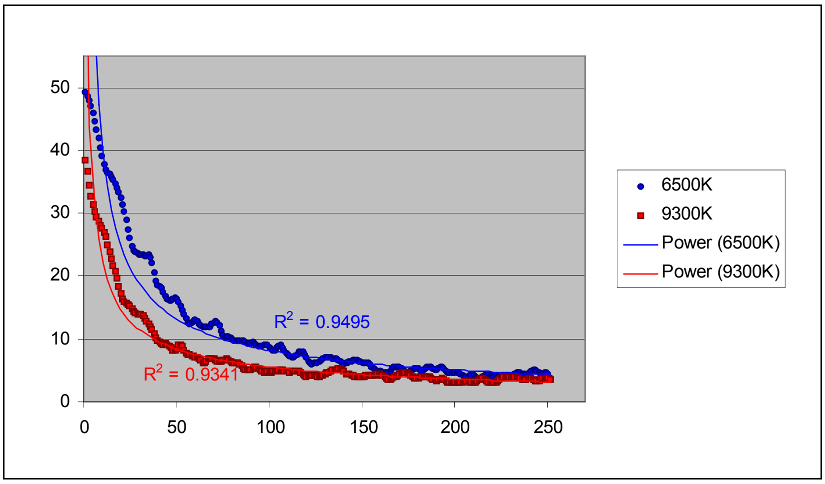


Figure A-19 – Lightness discrimination thresholds for two temperatures of the CRT monitor used. As can be seen power functions fit well on the measured data.

The experiments performed here, are actually measuring Lightness least noticeable differences and not Luminance ones. The conversion between the two is not straightforward, since it depends on a number of factors, related not only to the hardware used, but also to the environment the experiments took place. The curve of the thresholds obtained is expected to follow a power law, with larger least-noticeable steps at low levels of Lightness, and smaller ones for higher levels. The results obtained are illustrated in Figure A-19. A power function can be fitted well on the curve as shown in the figure.

Appendix B

Fuzzy Inference Systems

Fuzzy logic is all about the relative importance of precision. You are approaching a red light and must advise a student when to apply the brakes, would you say: “begin braking *65* feet from the crosswalk”? or would your advice be more like “apply the brakes pretty soon”? The latter, is not a precise statement, but it delivers the necessary information in a way human beings understand and is much closer to everyday statements. Fuzzy logic provides an easy way to describe such situations.

B.1. Fuzzy Sets and Membership Functions

The point of fuzzy logic is to map an input space to an output space, and the primary mechanism for doing this is a list of if-then statements called rules. All rules are evaluated in parallel, and the order of the rules is unimportant. The rules themselves are useful because they refer to variables and the adjectives that describe those variables. All those structural elements of the rules have to be defined prior to using them.

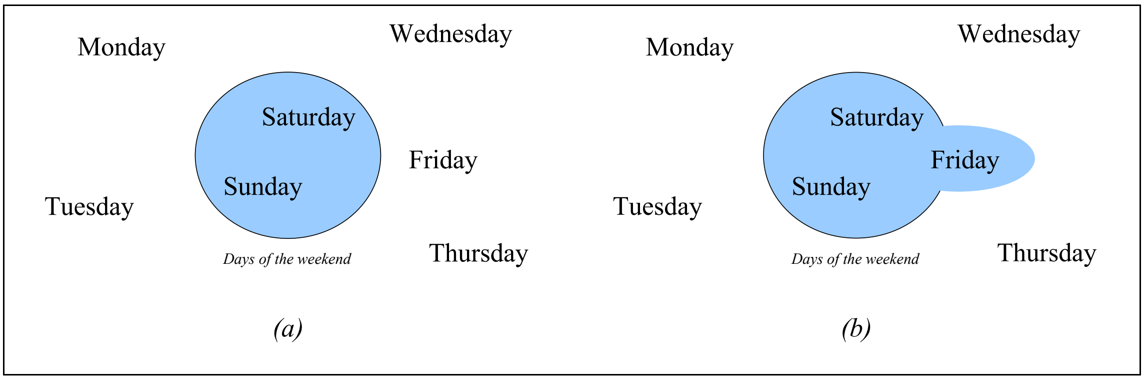


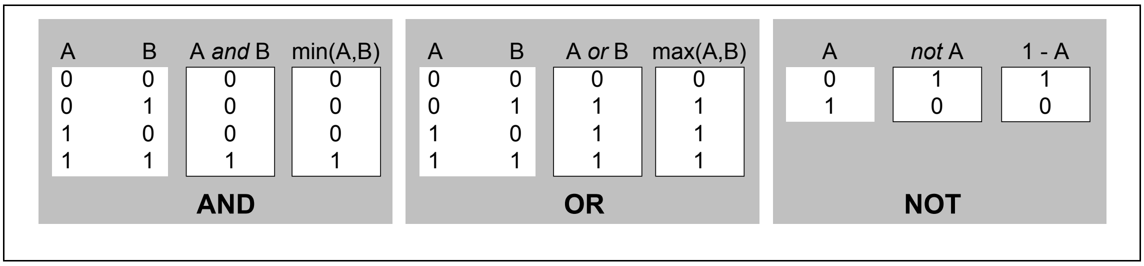
Figure B-1 – (a) The definition of "weekend days" using a classical set. (b) The definition of "weekend days" using a fuzzy set. Although technically Friday should be excluded from the set, it feels like part of the weekend, so in the latter case it is assigned a partial membership to the set.

The concept of the Fuzzy Set is of great importance to fuzzy logic. A *Fuzzy Set* is a set without a clearly defined boundary. In a classical definition, a set is a container that wholly includes or wholly excludes any given element. There is not such a thing as an element being both part of the set and not part of the set, consequently, of any subject, one thing must be either asserted or denied, must be either true or false. A fuzzy set on the other hand, allows elements to be partially members of it. For example, trying to define a set of days comprising the weekend one would agree that Saturday and Sunday belong to the set, but what about Friday. Although it should be technically excluded, it "feels" like part of the weekend (Figure B-1). Fuzzy sets are used to describe vague concepts (weekend days, hot weather, tall people).

In fuzzy logic, the truth of any statement becomes a matter of degree. Friday can be part of the weekend to a certain degree. So, instead of assigning true (*1*) or false (*0*) membership values to each element, fuzzy logic allows for in-between degrees of membership to be defined (Friday is sort of a weekend day, the weather is rather hot). Everyday life suggests that this second approach is much closer to the way people think. For the above example, one could give a value representing the degree of membership of each day of the week to the fuzzy set called "weekend". A *Membership Function* is a curve that defines how each point in the input space is mapped to a membership value (or degree of membership) between *0* and *1*. The input space is sometimes referred to as the *Universe of Discourse*.

B.2. Logical Operations

Since we allowed for partial memberships to fuzzy sets (degrees of membership between true and false), the logical reasoning has to slightly change, to accommodate this. Fuzzy logical reasoning is a superset of standard Boolean Logic. If we keep the fuzzy values at their extremes (*1* and *0*) standard logic operations should hold. Consequently, the *AND*, *OR* and *NOT* operators of standard Boolean Logic have to be replaced by operations that behave in the same way if the fuzzy values are at their extremes. One widely used set of operations used is illustrated in Figure B-2. A *AND* B becomes *min*(A, B), A *OR* B becomes *max*(A, B) and *NOT*(A) becomes *1*-A.



A	B	A *and* B	min(A,B)
0	0	0	0
0	1	0	0
1	0	0	0
1	1	1	1

AND

A	B	A *or* B	max(A,B)
0	0	0	0
0	1	1	1
1	0	1	1
1	1	1	1

OR

A	*not* A	1 - A
0	1	1
1	0	0

NOT

Figure B-2 – Boolean Logic and Fuzzy Logic operations. If we keep the fuzzy values at their extremes (1 and 0) standard Boolean Logic operations hold.

B.3. If-Then Rules

Fuzzy sets and fuzzy operators are the verbs and subjects of fuzzy logic. A set of if-then statements called rules is used to formulate the conditional statements that comprise fuzzy logic. A single fuzzy rule assumes the form: *"if x is A then y is B*", where A and B are linguistic values defined by fuzzy sets. The if-part of the rule is called the antecedent, while the then-part of the rule is called the consequent. An example of a fuzzy rule might be "if distance from crosswalk is short, then braking is hard". Here "short" is represented as a number between 0 and 1, therefore the antecedent is an interpretation that returns a single number between 0 and 1. On the other hand, "hard" is represented as a fuzzy set, so the consequent is an assignment that assigns the entire fuzzy set "hard" to the output variable "braking".

The use of the word "is" here is twofold, depending on whether it appears in the antecedent or the consequent. In the antecedent "is" has the meaning of a relational test (similar to "= =" of C++), while in the consequent is used as an assignment operator (like "=" of C++).

Interpreting a fuzzy rule involves different parts. First, the antecedent has to be evaluated, that is the input must be *fuzzyfied* and any fuzzy operators must be applied. Then the result must be applied to the consequent (a process known as *implication*). If the antecedent is true to some degree of membership, then the consequent is also true to the same degree. The most common ways to modify the output fuzzy set are truncation, or scaling.

In general, more than one rules are used, each one outputting a fuzzy set. The output fuzzy sets are the aggregated into a single output fuzzy set, which is ultimately *defuzzyfied*, or resolved to a single number.

Fuzzy Inference is the process of formulating the mapping from a given input to an output using fuzzy logic (Figure B-3). The mapping then provides a basis from which decisions can be made, or patterns discerned. The process of fuzzy inference involves all of the pieces described here. There are two main types of fuzzy inference systems: Mamdani-type and Sugeno-type, mainly differing in the way the outputs are determined. The fuzzy inference system used in this thesis is a Mamdani-type one, where the implication method employed is truncation, and the defuzzyfication method is the centroid calculation of the final output curve.

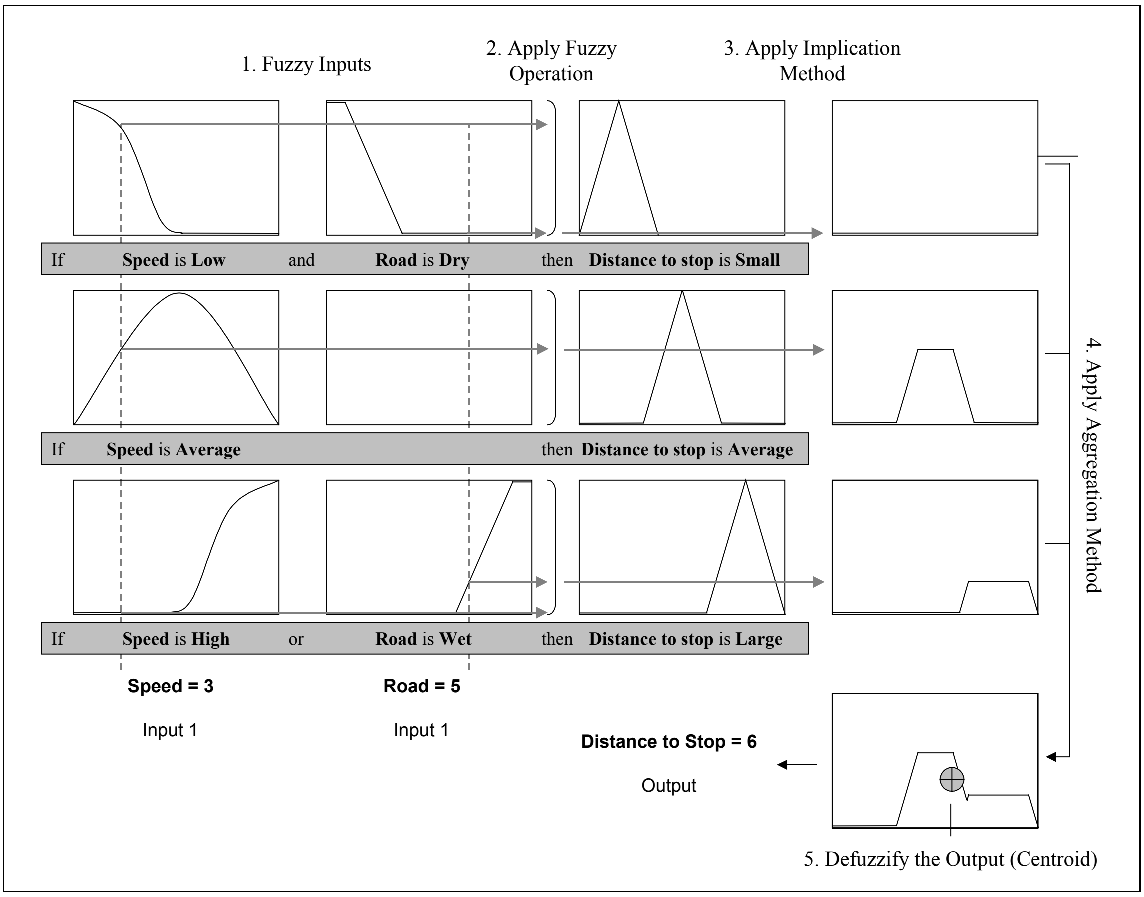

1. Fuzzy Inputs
2. Apply Fuzzy Operation
3. Apply Implication Method
If Speed is Low and Road is Dry then Distance to stop is Small
If Speed is Average then Distance to stop is Average
If Speed is High or Road is Wet then Distance to stop is Large
4. Apply Aggregation Method
Speed = 3
Input 1
Road = 5
Input 1
Distance to Stop = 6
Output
5. Defuzzify the Output (Centroid)

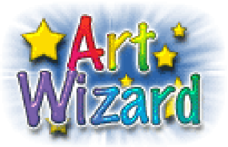
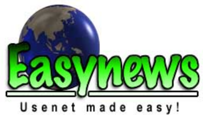
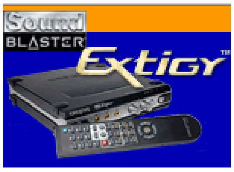
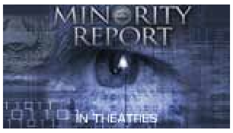
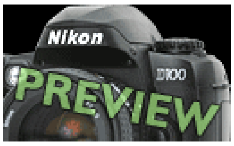
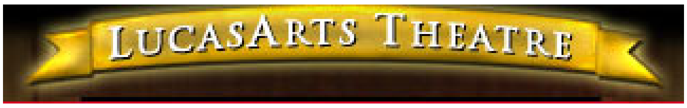
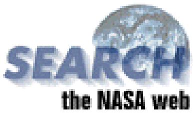
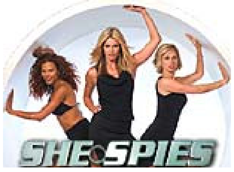
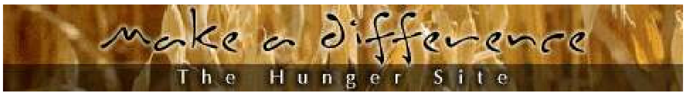
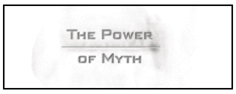
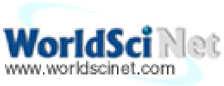
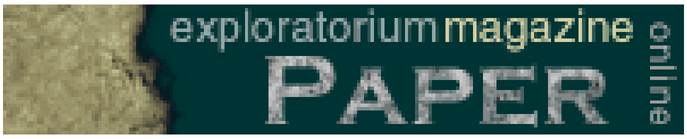
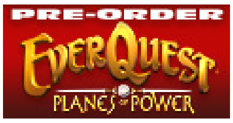
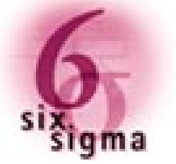
Figure B-3 – A complete Fuzzy Inference System. The five steps of mapping the input values to the output are depicted here.

Appendix C

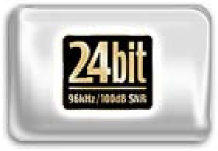
The Data Set

In this appendix, the images of the data set are presented, along with statistical data for the data set, and per image results for the two segmentation methods. The Images are presented by category.

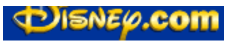
C.1. Multi-Coloured Text over Multi-Coloured Background

 ArtWizard	 EasyNews	 Extigy	 MinorityReport	 NikonPreview
 LucasTheatre		 SearchNASA	 ShwSpies	
 MakeADifference		 ThePowerOfMyth	 WorlSciNet	
 PaperBanner		 PlanesOfPower	 SixSigma	

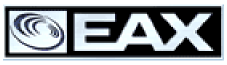
C.2. Multi-Coloured Text over Single-Coloured Background



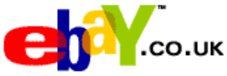
24bit



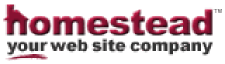
DisneyLogo



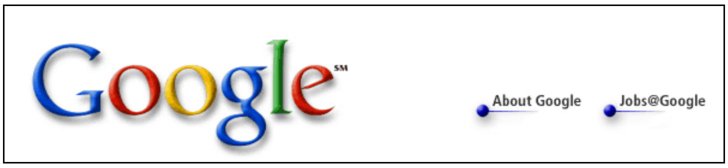
Eax



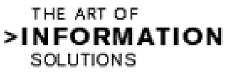
EBayLogo



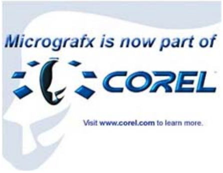
Homestead



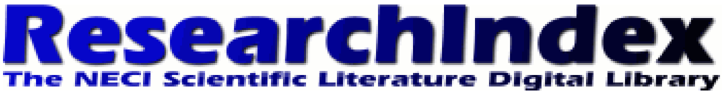
Google



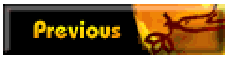
InformationSolutions



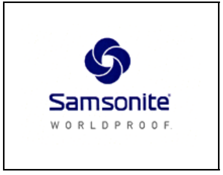
Micrografx



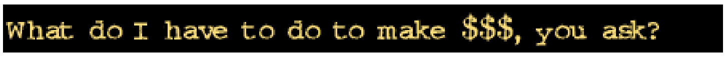
ResearchIndex



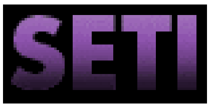
PreviousConnect



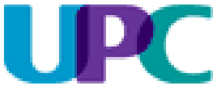
Samsonite



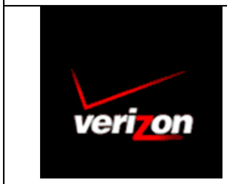
WhatDoIDo



SmallSetiLogo

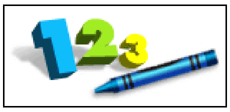


UPCLogo



VerizonLogo

C.3. Single-Coloured Text over Multi-Coloured Background



123

Advertised
SPECIALS!

AdvertisedSpecials

BARNES&NOBLE.com
www.bn.com

BarnesNoble

Dixons

HomeGrey

LIFE ALONG THE
FAULTLINE

Faultline

High

Princess Cruises' Call Center
Sets Sail

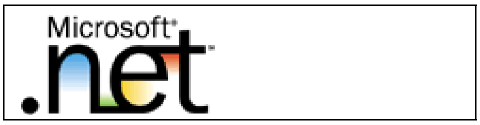
CruisesAdvertisment

Littlewoods

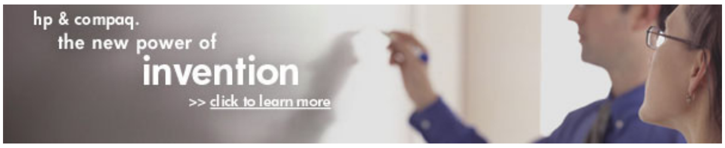
Littlewoods

FORUM
The U.S.
in Panama:
Boon or Bane?

Forum



DotNet



HPAdvert1

Lycos

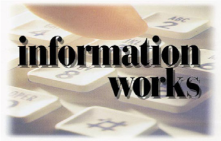
MZ

THE REAL WITCH PROJECT
SALEM
ALSO: EERIE E-GREETINGS
HALLOWEEN PHOTOS

Salem

RED ROVER,
RED ROVER

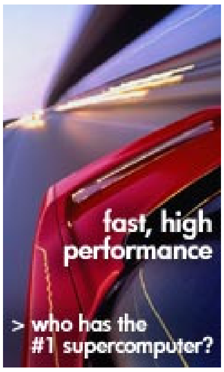
RedRover



InformationWorks

Microsoft
Windows
Strategy

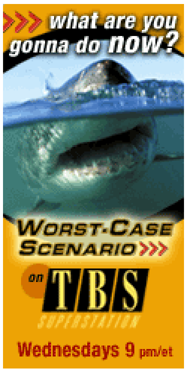
MSWindows



HPAdvert2

Puzzles
at the new
N
Netscape.com
CLICK HERE

Puzzles



TBS

The Wir
Opera

WindowsOperating

Openwave

SELECTED
MJ
SITE

SelectedSite

REUTERS
HEADLINES

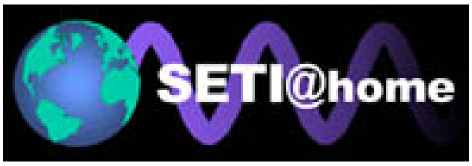
Reuters

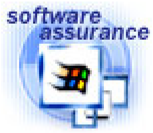
SOAP CITY RADIO
ENTER NOW!

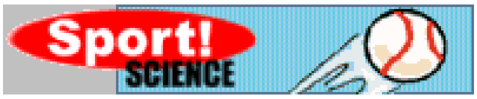
SoapCityRadio

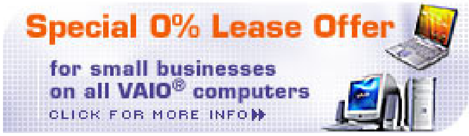
SAVING THE
AMAZON

SavingTheAmazon

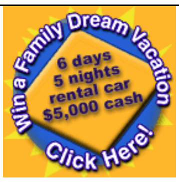
<table>
<tr>
<td>

SetiLogo</td>
<td>

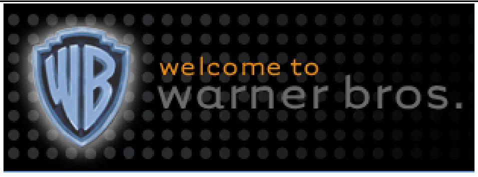
SoftwareAssurance</td>
<td>

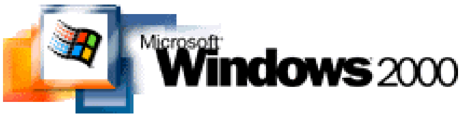
SportScience</td>
</tr>
<tr>
<td>

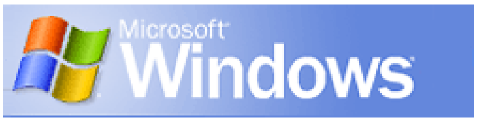
Special 0% Lease Offer
for small businesses
on all VAIO® computers
CLICK FOR MORE INFO

VaioAdvert</td>
<td>

Win a Family Dream Vacation
6 days
5 nights
rental car
$5,000 cash
Click Here!

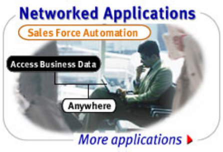
WinVacation</td>
<td>

Warner</td>
</tr>
<tr>
<td colspan="2">

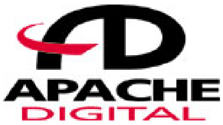
Windows2000</td>
<td>

WindowsLogo</td>
</tr>
</table>

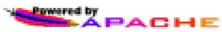
C.4. Single-Coloured Text over Single-Coloured Background



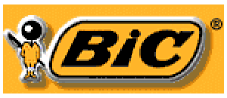
3ComApplications



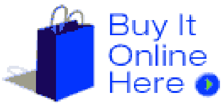
ApacheDigital



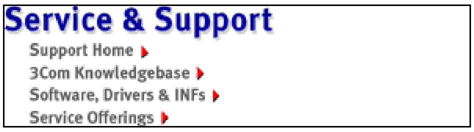
Apache



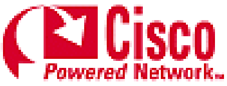
BICLogo



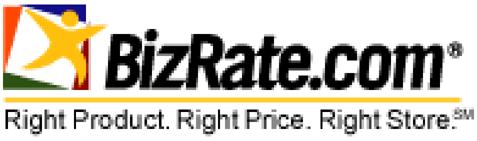
BuyItOnLine



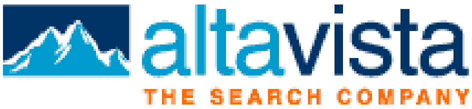
3ComServices



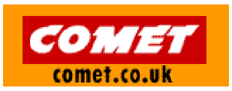
Cisco



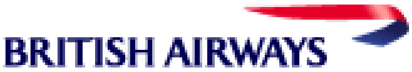
Bizrate



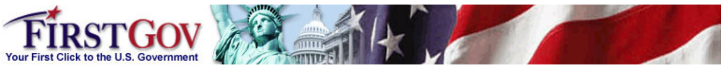
AltaVista



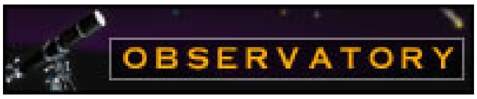
Comet



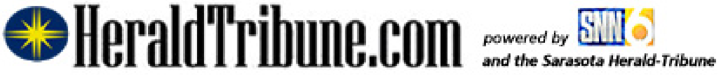
BritishAirways



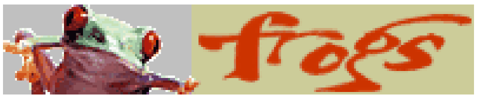
FirstGov



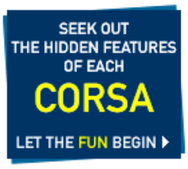
Observatory



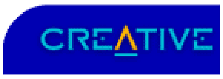
HeraldTribune



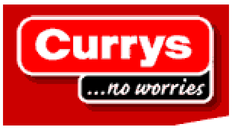
Frogs



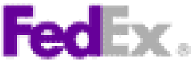
Corsa



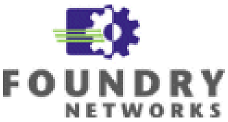
Creative



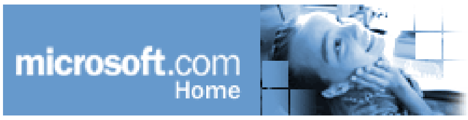
Currys



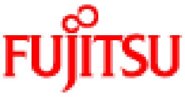
FedEx



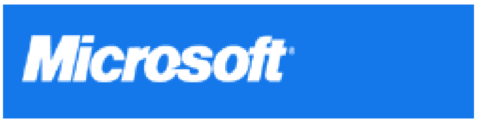
Foundry



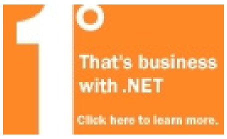
MicrosoftHome



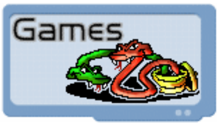
Fujitsu



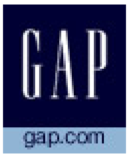
MicrosoftLogo



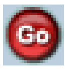
FullHeightCharacter



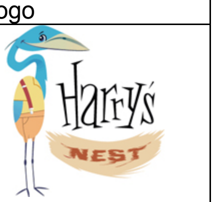
Games



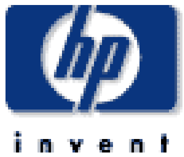
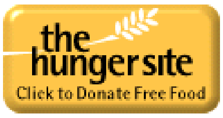
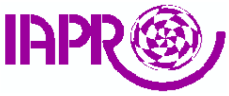
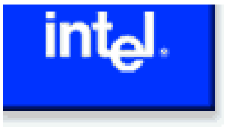
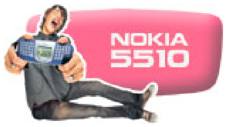
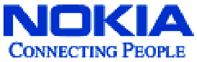
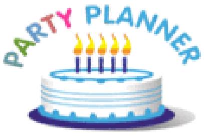
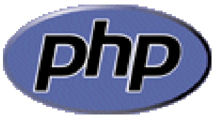
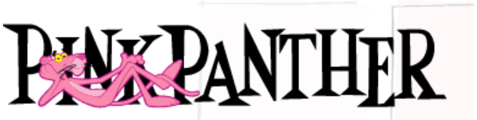
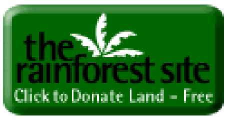
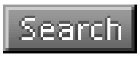
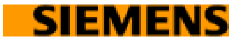
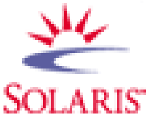
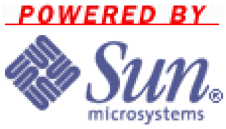
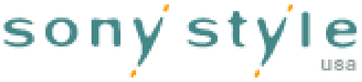
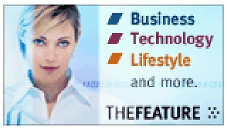
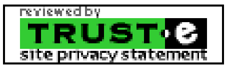
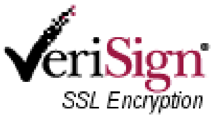
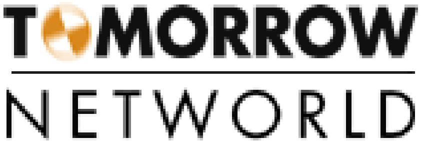
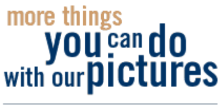
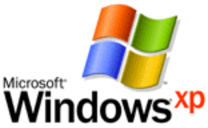
GapLogo



Go_Button



HarrysNest

 HPInvent	 HungerSite_Button	 IARP	 IntelLogo	 Nokia5510
 NokiaLogo	 PartyPlanner	 php	 PinkPanther	
 RainForest_Button	 Search_Button	 Siemens_Logo	 Solaris	 Sun
 SonyStyle		 TheFeature	 Truste	 VeriSign
 TomorrowNetworld		 WhatYouCanDo	 WindowsXP	

C.5. Additional Information on the Data Set

Image Name	Width	Height	DPI	BPP	File Type	Num of Colours
ArtWizard	125	80	73	8	GIF	128
EasyNews	283	163	72	24	JPEG	11860
Extigy	150	108	72	8	GIF	128
LucasArtsTheatre	322	47	72	24	JPEG	10267
MakeADifference	480	60	72	24	JPEG	19689
MinorityReport	156	85	100	24	JPEG	6216
NikonPreview	127	76	73	8	GIF	128
PaperBanner	200	38	72	8	GIF	64
PlanesOfPower	120	60	72	24	JPEG	5020
SearchNASA	120	73	73	8	GIF	108
SheSpies	153	117	72	8	JPEG	10417
SixSigma	75	67	72	24	JPEG	2386
ThePowerOfMyth	226	83	72	8	GIF	31
WordSciNet	132	50	72	24	JPEG	1510
24bit	142	98	72	24	JPEG	2630
DisneyLogo	162	25	73	8	GIF	60
Eax	138	35	72	8	GIF	32
eBayLogo_Large	170	59	73	8	GIF	32
Google	600	130	72	8	GIF	128
Homestead	158	50	72	8	GIF	64
InformationSolutions	132	37	72	8	GIF	16
Micrografx	250	198	72	24	JPEG	8738
PreviousConnect	118	27	72	8	GIF	255
ResearchIndex	370	44	73	8	GIF	256
Samsonite	212	163	72	8	GIF	208
SmallSetiLogo	76	37	72	8	GIF	63
UPCLogo	80	30	73	8	GIF	37
VerizonLogo	147	141	73	8	GIF	32
WhatDoIDo	450	30	73	8	GIF	22
123	154	70	73	8	GIF	128
AdvertisedSpecials	150	93	72	8	GIF	128
BarnesNoble	244	62	72	8	GIF	20
CruisesAdvertisment	244	77	72	8	GIF	256
Dixons	185	87	73	8	GIF	64
dotNet	250	60	73	8	GIF	128
Faultline	200	38	72	24	JPEG	3225
Forum	104	52	72	8	GIF	56
HelpWhales	216	62	72	24	JPEG	11170
High	111	27	72	24	JPEG	2591
HomeGray	112	64	300	24	JPEG	1274
HPAdvert1	602	116	72	24	JPEG	9656
HPAdvert2	120	201	72	24	JPEG	18029
InformationWorks	545	354	150	24	JPEG	29440
InternetBanking	400	40	73	8	GIF	57
Littlewoods	142	30	72	8	GIF	32
Lycos	319	50	72	8	GIF	31
MSWindows	273	120	72	8	GIF	94

Image Name	Width	Height	DPI	BPP	File Type	Num of Colours
MZ	120	60	72	24	JPEG	4682
Openwave	124	30	73	8	GIF	205
Puzzles	120	240	73	8	GIF	111
RedRover	93	68	72	8	GIF	233
Reuters	171	20	72	8	GIF	32
Salem	310	190	72	24	JPEG	15146
SavingTheAmazon	202	62	72	24	JPEG	10277
SelectedSite	77	77	72	8	GIF	214
SetiLogo	196	66	72	24	JPEG	5236
SoapCityRadio	120	60	72	24	JPEG	4484
SoftwareAssurance	75	65	72	24	JPEG	3112
SportScience	200	38	72	8	GIF	64
TBS	120	240	72	8	GIF	128
VaioAdvert	270	89	72	24	JPEG	8292
Warner	310	111	73	8	GIF	127
Windows2000	250	60	72	8	GIF	128
WindowsLogo	250	60	72	8	GIF	256
WindowsOperatingSystem	181	225	72	8	GIF	216
WinVacation	140	141	72	8	GIF	126
3ComApplications	244	167	72	24	JPEG	14509
3ComServices	297	77	72	8	GIF	62
AltaVista	211	45	72	8	GIF	16
Apache	120	15	73	8	GIF	223
ApacheDigital	150	80	72	24	JPEG	1973
BICLogo	149	60	72	8	GIF	41
BizRate	193	54	72	8	GIF	64
BritishAirways	189	48	73	8	GIF	124
BuyItOnLine	131	61	72	8	GIF	15
Cisco	100	35	73	8	GIF	62
Comet	160	58	72	8	GIF	16
Corsa	142	130	73	8	GIF	89
Creative	141	45	72	8	GIF	32
Currys	148	80	73	8	GIF	16
FedEx	75	24	72	8	GIF	16
FirstGov	770	63	72	24	JPEG	21998
Foundry	120	61	73	8	GIF	116
Frogs	200	38	72	8	GIF	64
Fujitsu	95	64	72	8	GIF	17
FullHeightCharacter	131	77	72	24	JPEG	1164
Games	156	86	73	8	GIF	249
GapLogo	61	74	72	24	JPEG	1053
Go_Button	23	23	73	8	GIF	64
HarrysNest	160	174	72	24	JPEG	3883
HeraldTribune	413	60	72	24	JPEG	2198
HPInvent	66	55	72	8	GIF	12
HungerSite_Button1	120	60	73	8	GIF	121
IARP	240	96	72	8	GIF	2
IntelLogo	98	54	72	8	GIF	16
MicrosoftHome	250	60	72	8	GIF	253
MicrosoftLogo	250	60	72	8	GIF	248

Image Name	Width	Height	DPI	BPP	File Type	Num of Colours
Nokia5510	156	85	92	24	JPEG	6225
NokiaLogo	135	50	73	8	GIF	12
Observatory	200	38	72	24	JPEG	3272
PartyPlanner	125	80	73	8	GIF	127
php	120	64	73	8	GIF	158
PinkPanther	385	101	72	8	GIF	32
RainForest_Button1	122	61	73	8	GIF	49
Search_Button	50	18	72	8	GIF	28
Siemens_Logo	128	18	72	8	GIF	16
Solaris	57	44	73	8	GIF	185
SonyStyle	211	54	72	8	GIF	24
Sun	112	62	73	8	GIF	73
TheFeature	156	75	83	8	GIF	128
TomorrowNetworld	164	60	72	8	GIF	78
Truste	116	31	73	8	GIF	9
VeriSign	113	60	73	8	GIF	53
WhatYouCanDo	215	103	72	8	GIF	10
WindowsXP	166	111	72	8	GIF	128

C.6. Per Image Results for the Split and Merge Segmentation Method

Image Name	Number of Characters In Image	Number of Readable Characters	Number of Non-Readable Characters	Readable Characters Correctly Identified	Readable Characters Merged	Readable Characters Split	Readable Characters Missed
ArtWizard	9	9	0	3	0	6	0
EasyNews	22	22	0	16	0	6	0
Extigy	18	6	12	1	0	5	0
LucasArtsTheatre	16	16	0	9	0	6	1
MakeADifference	28	28	0	13	0	12	3
MinorityReport	24	14	10	4	0	2	8
NikonPreview	12	12	0	11	0	1	0
PaperBanner	32	32	0	14	0	8	10
PlanesOfPower	28	17	11	6	0	11	0
SearchNASA	16	16	0	14	0	1	1
SheSpies	8	8	0	3	0	1	4
SixSigma	9	1	8	1	0	0	0
ThePowerOfMyth	14	14	0	12	0	1	1
WordSciNet	28	11	17	8	0	0	3
24bit	18	5	13	3	0	2	0
DisneyLogo	9	9	0	0	0	9	0
Eax	3	3	0	3	0	0	0
eBayLogo_Large	8	8	0	6	0	2	0
Google	30	28	2	9	0	19	0
Homestead	27	27	0	18	9	0	0
InformationSolutions	28	28	0	19	0	8	1
Micrografx	53	26	27	3	0	22	1
PreviousConnect	8	8	0	6	2	0	0
ResearchIndex	54	54	0	28	21	2	3
Samsonite	19	19	0	15	0	3	1
SmallSetiLogo	4	4	0	0	0	0	4
UPCLogo	3	3	0	3	0	0	0
VerizonLogo	7	7	0	7	0	0	0
WhatDoIDo	31	31	0	15	16	0	0
123	3	3	0	3	0	0	0
AdvertisedSpecials	19	19	0	16	2	0	1
BarnesNoble	23	23	0	23	0	0	0
CruisesAdvertisment	33	33	0	33	0	0	0
Dixons	6	6	0	6	0	0	0
dotNet	12	12	0	8	0	4	0
Faultline	21	21	0	19	0	0	2
Forum	29	29	0	15	0	10	4
HelpWhales	35	35	0	31	0	4	0
High	4	4	0	0	0	4	0
HomeGray	4	4	0	4	0	0	0
HPAdvert1	47	31	16	28	2	1	0
HPAdvert2	44	19	25	11	0	8	0
InformationWorks	16	16	0	7	1	5	3
InternetBanking	15	15	0	15	0	0	0

Image Name	Number of Characters In Image	Number of Readable Characters	Number of Non-Readable Characters	Readable Characters Correctly Identified	Readable Characters Merged	Readable Characters Split	Readable Characters Missed
Littlewoods	11	11	0	0	11	0	0
Lycos	4	4	0	4	0	0	0
MSWindows	24	24	0	6	4	5	9
MZ	7	7	0	6	0	1	0
Openwave	8	8	0	4	0	4	0
Puzzles	35	18	17	14	4	0	0
RedRover	16	16	0	13	3	0	0
Reuters	16	16	0	14	0	0	2
Salem	58	24	34	19	0	2	3
SavingTheAmazon	15	15	0	11	2	0	2
SelectedSite	14	14	0	12	0	0	2
SetiLogo	9	9	0	8	0	1	0
SoapCityRadio	21	21	0	11	3	4	3
SoftwareAssurance	17	0	17	0	0	0	0
SportScience	13	13	0	12	0	0	1
TBS	70	64	6	46	0	4	14
VaioAdvert	72	55	17	45	6	2	2
Warner	21	21	0	11	0	1	9
Windows2000	20	11	9	1	10	0	0
WindowsLogo	16	7	9	7	0	0	0
WindowsOperatingSystem	9	9	0	9	0	0	0
WinVacation	62	62	0	58	0	4	0
3ComApplications	83	37	46	31	0	6	0
3ComServices	79	79	0	55	0	19	5
AltaVista	25	25	0	21	0	4	0
Apache	15	6	9	4	0	1	1
ApacheDigital	13	13	0	13	0	0	0
BICLogo	4	3	1	3	0	0	0
BizRate	42	10	32	10	0	0	0
BritishAirways	14	14	0	14	0	0	0
BuyItOnLine	15	15	0	10	0	5	0
Cisco	19	19	0	11	5	0	3
Comet	14	14	0	14	0	0	0
Corsa	49	49	0	37	4	8	0
Creative	8	8	0	8	0	0	0
Currys	15	6	9	6	0	0	0
FedEx	5	5	0	0	3	0	2
FirstGov	39	8	31	8	0	0	0
Foundry	15	15	0	15	0	0	0
Frogs	5	5	0	5	0	0	0
Fujitsu	7	7	0	7	0	0	0
FullHeightCharacter	42	2	40	2	0	0	0
Games	5	5	0	5	0	0	0
GapLogo	9	9	0	0	5	4	0
Go_Button	2	0	2	0	0	0	0
HarrysNest	10	10	0	10	0	0	0
HeraldTribune	55	19	36	11	3	5	0
HPInvent	8	2	6	0	0	0	2
HungerSite_Button1	34	13	21	10	0	3	0

Image Name	Number of Characters In Image	Number of Readable Characters	Number of Non-Readable Characters	Readable Characters Correctly Identified	Readable Characters Merged	Readable Characters Split	Readable Characters Missed
IARP	4	4	0	4	0	0	0
IntelLogo	5	5	0	5	0	0	0
MicrosoftHome	16	16	0	8	2	0	6
MicrosoftLogo	9	9	0	4	5	0	0
Nokia5510	9	9	0	7	0	0	2
NokiaLogo	21	21	0	9	11	1	0
Observatory	11	11	0	11	0	0	0
PartyPlanner	12	12	0	8	4	0	0
php	3	3	0	3	0	0	0
PinkPanther	6	6	0	6	0	0	0
RainForest_Button1	38	17	21	0	0	0	17
Search_Button	6	0	6	0	0	0	0
Siemens_Logo	7	7	0	7	0	0	0
Solaris	7	7	0	2	2	2	1
SonyStyle	12	9	3	9	0	0	0
Sun	25	25	0	22	2	0	1
TheFeature	44	44	0	26	7	9	2
TomorrowNetworld	15	15	0	15	0	0	0
Truste	36	6	30	6	0	0	0
VeriSign	21	21	0	6	2	13	0
WhatYouCanDo	33	33	0	33	0	0	0
WindowsXP	18	9	9	9	0	0	0

C.7. Per Image Results for the Fuzzy Segmentation Method

Image Name	Number of Characters In Image	Number of Readable Characters	Number of Non-Readable Characters	Readable Characters Correctly Identified	Readable Characters Merged	Number of Readable Characters Split	Number of Readable Characters Missed
ArtWizard	9	9	0	2	0	6	1
EasyNews	22	22	0	15	2	5	0
Extigy	18	6	12	6	0	0	0
LucasArtsTheatre	16	16	0	12	0	2	2
MakeADifference	28	28	0	14	0	8	6
MinorityReport	24	14	10	2	2	0	10
NikonPreview	12	12	0	10	2	0	0
PaperBanner	32	32	0	25	0	0	7
PlanesOfPower	28	17	11	7	0	2	8
SearchNASA	16	16	0	10	2	1	3
SheSpies	8	8	0	1	0	0	7
SixSigma	9	1	8	0	0	0	1
ThePowerOfMyth	14	14	0	10	0	0	4
WordSciNet	28	11	17	8	0	0	3
24bit	18	5	13	5	0	0	0
DisneyLogo	9	9	0	3	0	6	0
Eax	3	3	0	3	0	0	0
eBayLogo_Large	8	8	0	7	0	1	0
Google	30	28	2	24	0	1	3
Homestead	27	27	0	27	0	0	0
InformationSolutions	28	28	0	20	0	8	0
Micrografx	53	26	27	3	0	9	14
PreviousConnect	8	8	0	5	0	3	0
ResearchIndex	54	54	0	49	4	1	0
Samsonite	19	19	0	13	0	1	5
SmallSetiLogo	4	4	0	1	0	1	2
UPCLogo	3	3	0	2	0	0	1
VerizonLogo	7	7	0	1	6	0	0
WhatDoIDo	31	31	0	17	6	5	3
123	3	3	0	2	0	0	1
AdvertisedSpecials	19	19	0	16	0	0	3
BarnesNoble	23	23	0	17	5	1	0
CruisesAdvertisment	33	33	0	28	0	0	5
Dixons	6	6	0	6	0	0	0
dotNet	12	12	0	8	0	4	0
Faultline	21	21	0	11	0	0	10
Forum	29	29	0	17	0	8	4
HelpWhales	35	35	0	27	0	8	0
High	4	4	0	4	0	0	0
HomeGray	4	4	0	4	0	0	0
HPAdvert1	47	31	16	25	0	0	6
HPAdvert2	44	19	25	8	11	0	0
InformationWorks	16	16	0	3	0	0	13
InternetBanking	15	15	0	12	0	0	3

Image Name	Number of Characters In Image	Number of Readable Characters	Number of Non-Readable Characters	Readable Characters Correctly Identified	Readable Characters Merged	Number of Readable Characters Split	Number of Readable Characters Missed
Littlewoods	11	11	0	0	11	0	0
Lycos	4	4	0	4	0	0	0
MSWindows	24	24	0	8	0	7	9
MZ	7	7	0	7	0	0	0
Openwave	8	8	0	6	0	0	2
Puzzles	35	18	17	18	0	0	0
RedRover	16	16	0	12	4	0	0
Reuters	16	16	0	16	0	0	0
Salem	58	24	34	13	0	7	4
SavingTheAmazon	15	15	0	12	3	0	0
SelectedSite	14	14	0	12	0	0	2
SetiLogo	9	9	0	9	0	0	0
SoapCityRadio	21	21	0	13	6	2	0
SoftwareAssurance	17	0	17	0	0	0	0
SportScience	13	13	0	5	0	0	8
TBS	70	64	6	40	6	1	17
VaioAdvert	72	55	17	50	3	1	1
Warner	21	21	0	12	0	0	9
Windows2000	20	11	9	1	10	0	0
WindowsLogo	16	7	9	5	0	0	2
WindowsOperatingSystem	9	9	0	9	0	0	0
WinVacation	62	62	0	54	7	1	0
3ComApplications	83	37	46	21	0	16	0
3ComServices	79	79	0	53	4	0	22
AltaVista	25	25	0	21	0	2	2
Apache	15	6	9	5	0	1	0
ApacheDigital	13	13	0	11	0	2	0
BICLogo	4	3	1	3	0	0	0
BizRate	42	10	32	5	5	0	0
BritishAirways	14	14	0	12	0	2	0
BuyItOnLine	15	15	0	4	0	10	1
Cisco	19	19	0	14	4	1	0
Comet	14	14	0	14	0	0	0
Corsa	49	49	0	38	10	1	0
Creative	8	8	0	8	0	0	0
Currys	15	6	9	6	0	0	0
FedEx	5	5	0	0	5	0	0
FirstGov	39	8	31	6	0	2	0
Foundry	15	15	0	13	0	0	2
Frogs	5	5	0	5	0	0	0
Fujitsu	7	7	0	7	0	0	0
FullHeightCharacter	42	2	40	0	0	0	2
Games	5	5	0	5	0	0	0
GapLogo	9	9	0	1	2	3	3
Go_Button	2	0	2	0	0	0	0
HarrysNest	10	10	0	9	0	0	1
HeraldTribune	55	19	36	9	7	0	3
HPInvent	8	2	6	2	0	0	0
HungerSite_Button1	34	13	21	12	0	1	0

Image Name	Number of Characters In Image	Number of Readable Characters	Number of Non-Readable Characters	Readable Characters Correctly Identified	Readable Characters Merged	Number of Readable Characters Split	Number of Readable Characters Missed
IARP	4	4	0	4	0	0	0
IntelLogo	5	5	0	5	0	0	0
MicrosoftHome	16	16	0	9	4	0	3
MicrosoftLogo	9	9	0	0	9	0	0
Nokia5510	9	9	0	7	0	1	1
NokiaLogo	21	21	0	8	4	8	1
Observatory	11	11	0	9	0	2	0
PartyPlanner	12	12	0	5	0	0	7
php	3	3	0	3	0	0	0
PinkPanther	6	6	0	6	0	0	0
RainForest_Button1	38	17	21	15	2	0	0
Search_Button	6	0	6	0	0	0	0
Siemens_Logo	7	7	0	7	0	0	0
Solaris	7	7	0	3	0	4	0
SonyStyle	12	9	3	6	0	0	3
Sun	25	25	0	22	0	0	3
TheFeature	44	44	0	32	2	1	9
TomorrowNetworld	15	15	0	15	0	0	0
Truste	36	6	30	6	0	0	0
VeriSign	21	21	0	4	2	12	3
WhatYouCanDo	33	33	0	33	0	0	0
WindowsXP	18	9	9	9	0	0	0

References

1. T. Akiyama and N. Hagita, "Automated Entry System for Printed Documents," *Pattern Recognition*, vol. 23, pp. 1141-1154, 1990.

2. M. d. B. Al-Daoud and S. A. Roberts, "New Methods for the Initialisation of Clusters," *Pattern Recognition Letters*, vol. 17, pp. 451-455, 1996.

3. M. Ali, W. N. Martin, and J. K. Aggarwal, "Color-Based Computer Analysis of Aerial Photographs," *Computer Graphics and Image Processing*, vol. 9, pp. 282-293, 1979.

4. D. Amor, *The E-Business (R)evolution: Living and Working in an Interconnected World*, 2nd ed. New Jersey: Prentice Hall, 2001.

5. A. Antonacopoulos, "Page Segmentation Using the Description of the Background," *Computer Vision and Image Understanding*, vol. 70, pp. 350-369, 1998.

6. A. Antonacopoulos and A. Brough, "Methodology for Flexible and Efficient Analysis of the Performance of Page Segmentation Algorithms," Proc. of 5th International Conference on Document Analysis and Recognition, Bangalore, India, 20-22 September 1999.

7. A. Antonacopoulos and F. Delporte, "Automated Interpretation of Visual Representations: Extracting Textual Information from WWW Images," in *Visual Representations and Interpretations*, R. Paton and I. Neilson, Eds. London: Springer, 1999.

8. A. Antonacopoulos and A. Economou, "A Structural Approach for Smoothing Noisy Peak-shaped Analytical Signals," *Chemometrics and Intelligent Laboratory Systems*, vol. 41, pp. 31-42, 1998.

9. A. Antonacopoulos, D. Karatzas, and J. Ortiz Lopez, "Accessing Textual Information Embedded in Internet Images," Proc. of SPIE Internet Imaging II, San Jose, USA, 24-26 January 2001, pp. 198-205.

10. H. S. Baird, "The Skew Angle of Printed Documents," Proc. of SPSE 40th Conference & Symposium on Hybrid Imaging Systems, Rochester, N.Y., May 1987, pp. 21-24.

11. H. S. Baird, S. E. Jones, and S. J. Fortune, "Image Segmentation by Shape-Directed Covers," Proc. of 10th International Conference on Pattern Recognition, Atlantic City, New Jerey, U.S.A., June 16-21 1990, pp. 820-825.

12. D. H. Ballard, "Generalizing the Hough Transform to Detect Arbitrary Shapes," *Pattern Recognition*, vol. 13, pp. 111-122, 1981.

13. R. E. Bedford and G. Wyszecki, "Wavelength Discrimination for Point Sources," *Journal of the Optical Society of America*, vol. 48, pp. 129, 1958.

14. F. Bergholm, "Edge Focusing," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 9, pp. 726-741, 1987.

15. J. C. Bezdek, *Pattern Recognition with Fuzzy Objective Function Algorithms*. New York: Plenum Press, 1981.

16. J. C. Bezdek and S. K. Pal, *Fuzzy Models for Pattern Recognition - Methods that Search for Structures in Data*. New York: Institute of Electrical and Electronics Engineers, 1992.

17. U. Bodenhofer and E. P. Klement, "Genetic Optimization of Fuzzy Classification Systems," in *Computational Intelligence in Theory and Practice*, B. Reusch and K. H. Temme, Eds. Heidelberg: Physica-Verlag, 2001, pp. 183-200.

18. P. S. Bradley and U. M. Fayyad, "Refining Initial Points for K-Means Clustering," Proc. of 15th International Conference on Machine Learning, San Francisco, CA, 1998, pp. 91-99.

19. G. Braudaway, "A Procedure for Optimum Choice of a Small Number of Colors from a Large Color Palette for Color Imaging," Proc. of Electronic Imaging, San Francisco, CA, 1987.

20. C. Brice and C. Fennema, "Scene Analysis Using Regions," *Artificial Intelligence*, vol. 1, pp. 205-226, 1970.

21. M. K. Brown, S. C. Glinski, and B. C. Schmult, "Web Page Analysis for Voice Browsing," Proc. of 1st International Workshop on Web Document Analysis, Seattle, USA, September 8 2001, pp. 59-61.

22. J. D. Browning and S. L. Tanimoto, "Segmentation of Pictures into Regions with a Tile-by-Tile Method," *Pattern Recognition*, vol. 15, pp. 1-10, 1982.

23. J. M. Buhmann, D. W. Fellner, M. Held, J. Kettener, and J. Puzicha, "Dithered Color Quantization," Proc. of EUROGRAPHICS 1998, Computer Graphics Forum, Lisboa, Portugal, 1998, pp. 219-231.

24. R. L. Cannon, R. L. Dave, and J. C. Bezdek, "Efficient Implementation of Fuzzy c-means Clustering Algorithms," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 8, 1986.

25. J. Canny, "A Computational Approach to Edge Detection," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 8, pp. 679-698, 1986.

26. M. J. Carlotto, "Histogram Analysis Using a Scale-Space Approach," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. PAMI-9, pp. 121-129, 1987.

27. S. G. Carlton and O. R. Mitchell, "Image Segmentation Using Texture and Gray Level," Proc. of IEEE Conf. Pattern Recognition and Image Processing, Rot, New York, 6-8 June 1977, pp. 387-391.

28. T. Carron and P. Lambert, "Color Edge Detection Using Jointly Hue, Saturation and Intensity," Proc. of IEEE International Conference on Image Processing, ICIP'94, Austin, USA, 1994, pp. 977-981.

29. T. Carron and P. Lambert, "Fuzzy Color Edge Extraction by Inference Rules Quantitative Study and Evaluation of Performance," Proc. of IEEE International Conference on Image Processing, ICIP'95, Washington DC, USA, 23-26 October 1995, pp. 181-184.

30. R. C. Carter and E. C. Carter, "CIE $L^*u^*v^*$ Color-Difference Equations for Self-Luminous Displays," *Color Research and Application*, vol. 8, pp. 252-253, 1983.

31. M. Celenk, "A Color Clustering Technique for Image Segmentation," *Computing Vision Graphics Image Processing*, vol. 52, 1990.

32. S. Y. Chen, W. C. Lin, and C. T. Chen, "Split-and-Merge Image Segmentation Based on Localized Feature Analysis and Statistical Tests," *Graphical Models and Image Processing*, vol. 53, pp. 457-475, 1991.

33. Y. P. Chien and K. S. Fu, "Preprocessing and Feature Extraction of Picture Patterns," Purdue University, West Lafayette, Indiana TR-EE 74-20, 1974.

34. C. K. Chow and T. Kaneko, "Automatic Boundary Detection of the Left-ventircle from Cineangiograms," *Comput. Biomed. Res.*, vol. 5, pp. 388-410, 1972.

35. P. Clark and M. Mirmehdi, "Recognising Text in Real Scenes," *International Journal on Document Analysis and Recognition*, vol. 4, pp. 243-257, 2002.

36. J. Cornelis, J. De Becker, M. Bister, C. Vanhove, G. Demonceau, and A. Cornelis, "Techniques for Cardiac Image Segmentation," Proc. of 14th IEEE EMBS Conference, Paris, France, 1992, pp. 1906-1908.

37. Y. Cui and Q. Huang, "Extacting Characters of Licence Plates from Video Sequences," *Machine Vision and Applications*, vol. 10, pp. 308-320, 1998.

38. J. F. Cullen and K. Ejiri, "Weak Model-Dependent Page Segmentation and Skew Correction for Processing Document Images," Proc. of 2nd International Conference on Document Analysis and Recognition, Tsukuba, Japan, October 20-22 1993, pp. 757-760.

39. L. S. Davis, "A Survey of Edge Detection Techniques," *Computer Graphics and Image Processing*, vol. 4, pp. 248-270, 1975.

40. L. S. Davis, "Hierarchical Generalized Hough Transforms and Line Segment Based Hough Transforms," *Pattern Recognition*, vol. 15, pp. 277-285, 1982.

41. J. De Becker, M. Bister, N. Langloh, C. Vanhove, G. Demonceau, and J. Cornelis, "A Split-and-merge Algorithm for the Segmentation of 2-d, 3-d, 4-d cardiac images," Proc. of IEEE Satelite Symposium on 3D Advanced Image Processing in Medicine, Rennes, France, 1992, pp. 185-189.

42. A. Dengel and G. Barth, "Document Description and Analysis by Cuts," Proc. of Conference on User-Oriented Content-Based Text and Image Handling, MIT, Cambridge MA, March 21-24 1988, pp. 940-952.

43. S. Di Zenzo, "A Note on the Gradient of a Multi-Image," *Computer Vision, Graphics, And Image Processing*, vol. 33, pp. 116-125, 1986.

44. W. Doyle, "Operations useful for similarity-invariant pattern recognition," *J. Ass. Comput. Mach.*, vol. 9, pp. 259-267, 1962.

45. R. O. Duda and P. E. Hart, *Pattern Classification and Scene Analysis*. New York: Wiley, 1970.

46. R. O. Duda and P. E. Hart, "Use of the Hough Transform to Detect Lines and Curves in Pictures," *Communs ACM*, vol. 15, pp. 11-15, 1972.

47. H. J. Durrett, *Color and the Computer*. Orlando, Florida, USA: Academic Press, Inc., 1987.

48. L. Eikvil, T. Taxt, and K. Moen, "A Fast Adaptive Method for Binarization of Document Images," Proc. of International Conference on Document Analysis and Recognition, ICDAR'91, France, 1991, pp. 435-443.

49. J. H. Elder and S. W. Zucker, "Local Scale Control for Edge Detection and Blur Estimation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 20, pp. 699-716, 1998.

50. J. A. Feldman and Y. Yakimovsky, "Decision Theory and Artificial Intelligence: A Semantic-based Region Analyser," *Artificial Intelligence*, vol. 5, pp. 349-371, 1974.

51. M. Ferraro, G. Boccignone, and T. Caelli, "On the Representation of Image Structures via Scale Space Entropy Conditions," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 21, pp. 1199-1203, 1999.

52. J. L. Fisher, S. C. Hinds, and D. P. D'Amato, "A Rule-Based System for Document Image Segmentation," Proc. of 10th International Conference on Pattern Recognition, Atlantic City, New Jersey, U.S.A., 16-21 June 1990, pp. 567-572.

53. L. A. Fletcher and R. Kasturi, "A Robust Algorithm for Text String Separation from Mixed Text/Graphics Images," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 10, pp. 910-918, 1988.

54. L. M. J. Florack, B. M. ter Haar Romeny, J. J. Koenderink, and M. A. Viergever, "Scale and Differential Structure of Images," *Image and Vision Computing*, vol. 10, pp. 376-388, 1992.

55. J. D. Foley, A. van Dam, S. K. Feiner, and J. F. Hughes, *Computer Graphics: Principles and Practice*, 2nd ed: Addison-Wesley, 1996.

56. K. S. Fu and J. K. Mui, "A Survey on Image Segmentation," *Pattern Recognition*, vol. 13, pp. 3-16, 1981.

57. A. Gagneux, V. Eglin, and H. Emptoz, "Quality Approach of Web Documents by an Evaluation of Structure Relevance," Proc. of 1st International Workshop on Web Document Analysis, Seatlle, USA, September 8 2001.

58. J. Gauch and C. W. Hsia, "A Comparison of Three Color Image Segmentation Algorithms in Four Color Spaces," Proc. of Visual Communications and Image Processing, 1992, pp. 1168-1181.

59. M. Goldberg and S. Shlien, "A Cluster Scheme for Multi-spectral Images," *IEEE Trans. Systems, Man, Cybernet.*, vol. SMC-8, pp. 86-92, 1978.

60. R. G. Gonzalez and R. E. Woods, *Digital Image processing*, 3 ed: Addison-Wesley Publishing, 1993.

61. R. G. Gonzalez and R. E. Woods, *Digital Image processing*, 2 ed: Addison-Wesley Publishing, 2001.

62. H. Goto and H. Asi, "Character Pattern Extraction from Documents with Complex Backgrounds," *International Journal on Document Analysis and Recognition*, pp. 258-268, 2002.

63. J. N. Gupta and P. A. Winzt, "Computer Processing Algorithm for Locating Boundaries in Digital Pictures," Proc. of Second International Joint Conference on Pattern Recognition, 1974, pp. 155-156.

64. J. N. Gupta and P. A. Winzt, "Multi-image Modelling," School of Electrical Engineering, Purdue University, Technical Report TR-EE 74-24, Sept. 1974.

65. E. R. Hancock and J. Kittler, "Edge-labelling Using Dictionary-based Relaxation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 12, pp. 165-181, 1990.

66. A. R. Hanson and E. M. Riseman, "Segmentation of Natural Scenes," in *Computer Vision Systems*, A. R. Hanson and E. M. Riseman, Eds. New York: Academic Press, 1978, pp. 129-164.

67. R. M. Haralick, "Zero Crossing of Second Directional Derivative Edge Operator," Proc. of the Society of Photo-Optical Instrumentation Engineers Technical Symposium East, Arlington, Virginia, May 3-7 1982.

68. R. M. Haralick, "Digital Step Edges From Zero Crossing of Second Directional Derivative," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. PAMI-6, pp. 58-68, 1984.

69. R. M. Haralick, "Image Segmentation Techniques," *Computer Vision, Graphics, And Image Processing*, vol. 29, pp. 100-132, 1985.

70. R. M. Haralick, "Glossary of Computer Vision Terms," *Pattern Recognition*, vol. 24, pp. 69-93, 1991.

71. R. M. Haralick, I. Phillips, S. Chen, and J. Ha, "Document Zone Hierarchy and Classification," Proc. of IAPR International Workshop on Structural and Syntactic Pattern Recognition (SSPR'94), Nahariya, Israel, October 4-6 1994.

72. M. Hase and Y. Hoshino, "Segmentation Method of Document Images by Two-Dimensional Fourier Transformation," *Systems and Computers in Japan*, vol. 3, pp. 38-45, 1985.

73. P. Heckbert, "Color Image Quantization for Frame Buffer Display," *Comput. Graph.*, vol. 16, pp. 297-307, 1982.

74. S. C. Hinds, J. L. Fisher, and D. P. D'Amato, "A Document Skew Detection Method Using Run-Length Encoding and the Hough Transform," Proc. of 10th International Conference on Pattern Recognition, Atlantic City, New Jersey, U.S.A., 16-21 June 1990, pp. 464-468.

75. S. L. Horowitz and T. Pavlidis, "Picture Segmentation by a Traversal Algorithm," *Computer Graphics and Image Processing*, pp. 360-372, 1972.

76. S. L. Horowitz and T. Pavlidis, "Picture Segmentation by a Directed Split-and-merge Procedure," Proc. of 2nd Int. Joint Conference on Pattern Recognition, Copenhagen, Denmark, 1974, pp. 424-433.

77. P. V. C. Hough, "Method and Means for Recognizing Complex Patterns," U.S. Patent 3,069,654, 18 Dec 1962

78. M. Hueckel, "An Operator Which Locates Edges in Digital Pictures," *J. Ass. Comput. Mach.*, vol. 18, pp. 113-125, 1971.

79. R. W. G. Hunt, *Measuring Colour*. West Sussex, England: Ellis Horwood Limited, 1987.

80. T. L. Huntsberger and M. F. Descalzi, "Color Edge Detection," *Pattern Recognition Letters*, vol. 3, pp. 205-209, 1985.

81. T. L. Huntsberger, C. L. Jacobs, and R. L. Cannon, "Iterative Fuzzy Image Segmentation," *Pattern Recognition*, vol. 18, pp. 131-138, 1985.

82. N. Ikonomakis, K. N. Plataniotis, and A. N. Venetsanopoulos, "A Region-based Color Image Segmentation Scheme," Proc. of SPIE Visual Communication and Image Processing, 1999, pp. 1202-1209.

83. J. Illingworth and J. Kittler, "The Adaptive Hough Transform," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 9, pp. 690-698, 1987.

84. ITU-BT709, "Basic Parameter Values for the HDTV Standard for the Studio and for International Programme Exchange," International Telecommunications Union, ITU-R Recommendation BT.709 [formely CCIR Rec.709] Geneva, Switzerland: ITU 1990.

85. O. Iwaki, H. Kida, and H. Arakawa, "A Segmentation Method Based on Document Hierarchical Structure," Proc. of IEEE International Conference on Systems, Man and Cybernetics, Alexandria, VA, Oct. 20-23 1987, pp. 759-763.

86. A. K. Jain and B. Yu, "Automatic Text Location in Images and Video Frames," PRIP Lab, Department of Computer Science, Michigan State University, Technical Report MSU-CPS-97-33, 1997.

87. A. K. Jain and B. Yu, "Automatic Text Location in Images and Video Frames," *Pattern Recognition*, vol. 31, pp. 2055-2076, 1998.

88. A. K. Jain and B. Yu, "Document Representation and Its Application to Page Decomposition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 20, pp. 294-308, 1998.

89. T. Kanade, "Region Segmentation: Signal vs Semantics," *Computer Graphics and Image Processing*, vol. 13, pp. 279-297, 1980.

90. T. Kanungo, C. H. Lee, and R. Bradford, "What Fraction of Images on the Web Contain Text?," Proc. of 1st International Workshop on Web Document Analysis, Seattle, USA, September 8 2001.

91. J. Kasson and W. Plouffe, "An Analysis of Selected Computer Interchange Color Spaces," *ACM Transactions on Graphics*, vol. 11, pp. 373-405, 1992.

92. M. Kelly, "Edge Detection by Computer Using Planning," in *Machine Intelligence*, vol. VI. Edinburgh: Edinburgh University Press, 1971, pp. 397-409.

93. H.-K. Kim, "Efficient Automatic Text Location Method and Content-Based Indexing and Structuring of Video Database," *Journal of Visual Communication and Image Representation*, vol. 7, pp. 336-344, 1996.

94. R. Kirsch, "Computer Determination of the Constituent Structure of Biological Images," *Comput. Biomed. Res.*, vol. 4, pp. 315-328, 1971.

95. R. Kohler, "A Segmentation System Based on Thresholding," *Computer Graphics and Image Processing*, vol. 15, pp. 319-338, 1981.

96. M. Koppen, L. Lohmann, and B. Kickolay, "An Image Consulting Framework for Document Analysis of Internet Graphics," Proc. of 4th International Conference on Document Analysis and Recognition, Ulm, Germany, 18-20 August 1997, pp. 819-822.

97. J. J. Kulikowski, V. Walsh, and I. J. Murray, *Limits of Vision*, vol. 5. Boca Raton, USA: Macmillan Press Ltd, 1991.

98. R. S. Ledley, M. Buas, and T. J. Golab, "Fundamentals of True-Color Image Processing," Proc. of 10th International Conference on Pattern Recognition, 1990, pp. 791-795.

99. Y. Leung, J. S. Zhang, and Z. B. Xu, "Clustering by Scale-Space Filtering," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, pp. 1396-1410, 2000.

100. H. Li, D. Doerman, and O. Kia, "Text Extraction, Enhancement and OCR in Digital Video," in *Document Analysis Systems: Theory and Practice*, vol. 1655, *Lecture Notes in Computer Science*, Y. Nakano and S.-W. Lee, Eds.: Springer, 1999, pp. 363-377.

101. H. Li, D. Doerman, and O. Kia, "Automatic Text Detection and Tracking in Digital Video," *IEEE Transactions on Image Processing*, vol. 9, pp. 147-156, 2000.

102. R. Lienhart and F. Stuber, "Automatic Text Recognition in Digital Videos," Proc. of SPIE Volume: 2666 - Image and Video Processing IV, 1996, pp. 180-188.

103. Y. W. Lim and S. U. Lee, "On the Color Image Segmentation Algorithm Based on the Thresholding and the Fuzzy c-means Techniques," *Pattern Recognition*, vol. 23, pp. 935-952, 1990.

104. T. Lindeberg, "Scale-space for Discrete Signals," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 12, pp. 234-254, 1990.

105. D. Lopresti and J. Zhou, "Document Analysis and the World Wide Web," Proc. of the Workshop on Document Analysis Systems, Marven, Pennsylvania, October 1996, pp. 417-424.

106. D. Lopresti and J. Zhou, "Locating and Recognizing Text in WWW Images," *Information Retrieval*, vol. 2, pp. 177-206, 2000.

107. D. G. Lowe, "Three-dimensional Object Recognition from Single Two-dimensional Images," *Artificial Intelligence*, vol. 31, pp. 355-395, 1987.

108. Y. Lu and R. C. Jain, "Behavior of Edges in Scale Space," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 11, pp. 337-356, 1989.

109. E. P. Lyvers and O. R. Mitchell, "Precision Edge Constrast and Orientation Estimation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 10, pp. 927-937, 1988.

110. L. W. MacDonald and M. R. Luo, *Colour Imaging*. West Sussex, England: John Wiley & Sons, 1999.

111. J. MacQueen, "Some Methods for Classification and Analysis of Multivariate Observations," Proc. of 5th Berkeley Symposium on Mathematical Statistics and Probability, 1967, pp. 281-297.

112. E. H. Mamdani and S. S., "An Experiment in Linguistic Synthesis with a Fuzzy Logic Controller," *International Journal of Man-Machine Studies*, vol. 7, pp. 1-13, 1975.

113. D. Marr and E. Hildreth, "Theory of Edge Detection," *Proc. R. Soc. Lond.*, vol. B 207, pp. 187-217, 1980.

114. A. Martelli, "Edge Detection Using Heuristic Search Methods," *Computer Graphics and Image Processing*, vol. 1, pp. 169-182, 1972.

115. A. Martelli, "An Application of Heuristic Search Methods to Edge and Contour Detection," *Communs ACM*, vol. 19, pp. 73-83, 1976.

116. K. McLaren, "The Development of the CIE 1976 ($L^*a^*b^*$) Uniform Colour Space and Colour-difference Formula," *Journal of the Society of Dyers and Colorists*, vol. 92, pp. 338-341, 1976.

117. S. Messelodi and C. M. Modena, "Automatic Identification and Skew Estimation of Text Lines in Real Scene Images," *Pattern Recognition*, vol. 32, pp. 791-810, 1999.

118. M. Mirmehdi and M. Petrou, "Segmentation of Color Textures," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, pp. 142-159, 2000.

119. A. Moghaddamzadeh and N. Bourbakis, "A Fuzzy Region Growing Approach for Segmentation of Color Images," *Pattern Recognition*, vol. 30, pp. 867-881, 1997.

120. A. Moghaddamzadeh, D. Goldman, and N. Bourbakis, "A Fuzzy-Like Approach for Smoothing and Edge Detection in Color Images," *International Journal of Pattern Recognition and Artificial Intelligence*, vol. 12, pp. 801-816, 1998.

121. U. Montanari, "On the Optimal Detection of Curves in Noisy Pictures," *Communs ACM*, vol. 14, pp. 335-345, 1971.

122. J. L. Muerle and D. C. Allen, "Experimental Evaluation of Techniques for Automatic Segmentation of Objects in a Complex Scene," in *Pictorial Pattern Recognition*, G. C. Cheng and e. al., Eds. Washington: Thompson, 1968, pp. 3-13.

123. F. Muge, I. Granado, M. Mengucci, P. Pina, V. Ramos, N. Sirakov, J. R. Caldas Pinto, A. Marcolino, M. Ramalho, P. Vieira, and A. Maia do Amaral, "Automatic Feature Extraction and Recognition for Digial Access of Books of the Renaissance," in *Research and Advanced Technology for Digital Libraries*, vol. 1923, *Lecture Notes in Computer Science*, J. Borbinha and T. Baker, Eds. Berlin Heidelberg: Springer-Verlag, 2000, pp. 1-13.

124. J. K. Mui, J. W. Bacus, and F. K.S., "A Scene Segmentation Technique for Microscopic Cell Images," Proc. of Symp. Computer Aided Diagnosis of Medical Images, San Diego, CA, 1976, pp. 99-106.

125. E. V. Munson and Y. Tsymbalenko, "To Search for Images on the Web, Look at the Text, Then Look at The Images," Proc. of 1st International Workshop on Web Document Analysis, Seattle, USA, September 8 2001, pp. 39-42.

126. G. Murch, "Color Displays and Color Science," in *Color and the Computer*, J. Durrett, H., Ed. Orlando, Florida: Academic Press INC., 1987, pp. 1-25.

127. G. Nagy, J. Kanai, M. Krishnamoorty, M. Thomas, and M. Viswanathan, "Two Complementary Techniques for Digitized Document Analysis," Proc. of ACM Conference on Document Processing Systems, Santa Fe, New Mexico, Dec. 5-9 1988, pp. 169-176.

128. G. Nagy and S. Seth, "Hierarchical Representation of Optically Scanned Documents," Proc. of 7th International Conference on Pattern Recognition, Montreal, Canada, 1984, pp. 347-349.

129. G. Nagy, S. Seth, and S. D. Stoddard, "Document Analysis with an Expert System," in *Pattern Recognition in Practice II*: North-Holland, 1986, pp. 149-159.

130. Y. Nakagawa and A. Rosenfeld, "Some Experiments on Variable Thresholding," *Pattern Recognition*, vol. 11, pp. 191-204, 1979.

131. R. Nevatia, "A Color Edge Detector and Its Use in Scene Segmentation," *IEEE Transactions on Systems, Man and Cybernetics*, vol. 7, pp. 820-826, 1977.

132. L. O'Gorman, "The Document Spectrum for Bottom-Up Page Layout Analysis," in *Advances in Structural and Syntactic Pattern Recognition*, H. Bunke, Ed.: World Scientific, 1992, pp. 270-279.

133. L. O'Gorman, "Binarization and Multithresholding of Document Images using Connectivity," *CVGIP: Graphical Models and Imaage Processing*, vol. 56, pp. 496-506, 1994.

134. R. B. Ohlander, "Analysis of Natural Scenes," Ph.D. Dissertation, Department of Computer Science, Carnegie Mellon University, Pittsburgh, Pensylvania, 1975

135. R. B. Ohlander, K. Price, and D. R. Reddy, "Picture Segmentation Using a Recursive Region Splitting Method," *Computer Graphics and Image Processing*, vol. 8, pp. 313-333, 1978.

136. Y. Ohta, T. Kanade, and T. Sakai, "Color Information for Region Segmentation," *Computer Graphics and Image Processing*, vol. 13, pp. 222-241, 1980.

137. J. Ohya, A. Shio, and S. Akamatsu, "Recognizing Characters in Scene Images," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 16, pp. 214-218, 1994.

138. M. T. Orchard and A. Bouman, "Color Quantization Techniques," *IEEE Transactions on Signal Processing*, vol. 39, pp. 2677-2690, 1991.

139. N. R. Pal and D. Bhandari, "On Object-Background Classification," *Int. J. Syst. Sci.*, vol. 23, pp. 1903-1920, 1992.

140. N. R. Pal and S. K. Pal, "A Review on Image Segmentation Techniques," *Pattern Recognition*, vol. 26, pp. 1277-1294, 1993.

141. S. K. Pal, "Image Segmentation Using Fuzzy Correlation," *Information Science*, vol. 62, pp. 223-250, 1992.

142. N. Papamarkos, A. E. Atsalakis, and C. P. Strouthopoulos, "Adaptive Color Reduction," *IEEE Transactions on Systems, Man and Cybernetics*, vol. 32, pp. 44-56, 2002.

143. S. H. Park, I. D. Yun, and S. U. Lee, "Color Image Segmentation based on 3-D Clustering: Morphological Approach," *Pattern Recognition*, vol. 31, pp. 1060-1076, 1998.

144. P. Parodi and R. Fontana, "Efficient and Flexible Text Extraction from Document Pages," *International Journal on Document ANalysis and Recognition*, pp. 67-69, 1999.

145. T. Pavlidis, "Segmentation of Pictures and Maps through Functional Approximations," *Computer Graphics and Image Processing*, vol. 1, pp. 360-372, 1972.

146. T. Pavlidis, *Structural Pattern Recognition*. Berlin: Springer Verlag, 1977.

147. T. Perroud, K. Sobottka, H. Bunke, and L. O. Hall, "Text Extraction from Color Documents-Clustering Approaches in Three and Four Dimensions," Proc. of 6th International Conference on Document Analysis and Recognition, 2001, pp. 937-941.

148. K. P. Philip, "Automatic Detection of Myocardial Contours in Cine Computed Tomographic Images," PhD Thesis, University of Iowa, 1991

149. M. Pietikainen and A. Rosenfeld, "Image Segmentation by Texture Using Pyramid Node Linking," *IEEE Transactions on Systems, Man and Cybernetics*, vol. 11, pp. 822-825, 1981.

150. M. Pietikainen and A. Rosenfeld, "Gray Level Pyramid Linking as an Aid in Texture Analysis," *IEEE Transactions on Systems, Man and Cybernetics*, vol. 12, pp. 422-429, 1982.

151. M. Pietikainen, A. Rosenfeld, and I. Walter, "Split-and-Link Algorithms for Image Segmentation," *Pattern Recognition*, vol. 15, pp. 287-298, 1982.

152. C. Poynton, *A Technical Introduction to Digital Video*. New York: John Willey & Sons, 1996.

153. J. M. Prager, "Extracting and Labelling Boundary Segments in Natural Scenes," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 2, pp. 16-27, 1980.

154. J. M. S. Prewitt, "Object Enhancement and Extraction," in *Picture Processing and Psychopictorics*, B. S. Lipkin and A. Rosenfeld, Eds. New York: Academic Press, 1970, pp. 75-149.

155. J. M. S. Prewitt and M. L. Mendelsohn, "The Analysis of Cell Images," *Transactions of New York Academy of Science*, vol. 128, pp. 1035-1053, 1966.

156. D. M. L. Purdy, "On the Saturations and Chromatic Thresholds of the Spectral Colours," *British Journal of Phsycology*, vol. 21, pp. 283, 1931.

157. E. M. Riseman and M. A. Arbib, "Computational Techniques in the Visual Segmentation of Static Scenes," *Computer Graphics and Image Processing*, vol. 6, pp. 221-276, 1977.

158. E. M. Riseman and M. A. Arbib, "Segmentation of Static Scenes," *Computer Graphics and Image Processing*, vol. 6, pp. 221-276, 1977.

159. A. L. Robertson, "The CIE 1976 Color Difference Formulae," *Color Research and Application*, vol. 2, pp. 7-11, 1977.

160. T. V. Robertson, P. H. Swain, and K. S. Fu, "Multispectral Image Partitioning," School of Electrical Engineering, Purdue University TR-EE 73-26, August 1973 1973.

161. G. S. Robinson, "Color Edge Detection," *Optical Engineering*, vol. 16, pp. 479-484, 1977.

162. A. Rosenfeld, *Picture Processing by Computer*. New York: Academic Press, 1969.

163. A. Rosenfeld, "Iterative Methods in Image Analysis," Proc. of IEEE Conf. Pattern Recognition and Image Processing, Troy, Ney York, June 1977 1977, pp. 14-20.

164. A. Rosenfeld, R. A. Hummel, and S. W. Zucker, "Scene Labelling by Relaxation Operations," *IEEE Trans. on Systems, Man and Cybernetics*, vol. 6, pp. 420-433, 1976.

165. A. Rosenfeld and A. C. Kak, *Digital Picture Processing*. New York: Academic Press, 1976.

166. P. L. Rosin and G. A. W. West, "Segmentation of Edges into Lines and Arcs," *Image and Vision Computing*, vol. 7, pp. 109-114, 1989.

167. Y. Rubner, C. Tomasi, and L. J. Guibas, "A Metric for Distributions with Application to Image Databases," Proc. of International Conference on Computer Vision, Bombay, India, 5-8 January 1998, pp. 59-66.

168. M. A. Ruzon and C. Tomasi, "Color Edge Detection with the Compass Operator," Proc. of IEEE Conference on Computer Vision and Pattern Recognition, June 1999, pp. 160-166.

169. M. A. Ruzon and C. Tomasi, "Corner Detection in Textured Color Images," Proc. of IEEE International Conference on Computer Vision, September 1999, pp. 1039-1045.

170. M. A. Ruzon and C. Tomasi, "Edge, Junction, and Corner Detection Using Color Distributions," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 23, pp. 1281-1295, 2001.

171. A. Saheed and I. H. Witten, "Processing Textual Images," *New Zealand Journal of Computing*, vol. 4, pp. 57-66, 1993.

172. P. K. Sahoo, S. Soltani, A. K. C. Wong, and Y. C. Chen, "Survey of Thresholding Techniques," *Computer Vision, Graphics, And Image Processing*, vol. 41, pp. 233-260, 1988.

173. A. Sarabi and J. K. Aggarwal, "Segmentation of Chromatic Images," *Pattern Recognition*, vol. 13, pp. 417-427, 1981.

174. T. Sato, T. Kanade, E. K. Hughes, and M. Smith, A., "Video OCR for Digital News Archive," Proc. of IEEE International Workshop on Content Based Access of Image and Video Database, 1998, pp. 52-60.

175. J. Sauvola and M. Pietikainen, "Adaptive Document Image Binarization," *Pattern Recognition*, vol. 33, pp. 225-236, 2000.

176. B. J. Schacter, L. S. Davis, and A. Rosenfeld, "Scene Segmentation by Cluster Detection in Color Space," Computer Science Center, University of Maryland, Technical Report 424 1975.

177. R. Schettini, "A Segmentation Algorithm for Color Images," *Pattern Recognition Letters*, vol. 14, pp. 499-506, 1993.

178. J. Serra, *Image Analysis and Mathematical Morphology*. London: Academic Press, 1982.

179. L. G. Shapiro and G. C. Stockman, *Computer Vision*. Upper Saddle River, New Jersey: Prentice-Hall, Inc., 2001.

180. J. Shi and J. Malik, "Normalized Cuts and Image Segmentation," Proc. of 16th IEEE Conference on Computer Vision and Pattern Recognition (CVPR'97), Puerto Rico, 17-19 June 1997, pp. 731-737.

181. L. D. Silverstein, "Human Factors for Color Display Systems: Concepts, Methods, and Research," in *Color And The Computer*, J. Durrett, H., Ed. Orlando, Florida: Academic Press INC., 1987, pp. 27-61.

182. M. Smith, A. and T. Kanade, "Video Skimming and Characterization through the Combination of Image and Language Understanding Technique," Proc. of IEEE Conference on Computer Vision and Pattern Recognition, 1997, pp. 775-781.

183. K. Sobottka, H. Bunke, and H. Kronenberg, "Identification of Text on Colored Book and Journal Covers," Proc. of 5th International Conference on Document Analysis and Recognition (ICDAR), September 1999.

184. K. Sobottka, H. Kronenberg, T. Perroud, and H. Bunke, "Text Extraction from Colored Book and Journal Covers," *International Journal on Document ANalysis and Recognition*, pp. 163-176, 2000.

185. P. Soille, *Morphological Image Analysis*. Berlin: Springer, 1999.

186. M. Sonka, V. Hlavec, and R. Boyle, *Image Processing, Analysis and Machine Vision*. London: Chapman & Hall Computing, 1993.

187. A. L. Spitz, "Recognition Processing for Multilingual Documents," Proc. of International Conference on Electronic Publishing, Document Manipulation and Typography, Gaithersburg, Maryland, September 1990, pp. 193-205.

188. S. N. Srihari and Govindaraju, "Analysis of Textual Images Using the Hough Transform," *Machine Vision and Applications*, vol. 2, pp. 141-153, 1989.

189. S. S. Stevens, *Psychophysics: Introduction to Its Perceptual, Neutral and Social Prospects*. New York: Wiley, John & Sons, Incorporated, 1975.

190. M. Stokes, M. Anderson, S. Chandrasekar, and R. Motta, "A Standard Default Color Space for the Internet -sRGB," 1.10 ed. 1996, http://www.w3.org/Graphics/Color/sRGB.html

191. M. Suk and S. M. Chunk, "A New Image Segmentation Technique Based on Partition Mode Test," *Pattern Recognition*, vol. 16, pp. 469-480, 1983.

192. T. Taxt, P. J. Flynn, and A. K. Jain, "Segmentation of Document Images," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. PAMI-11, pp. 1322-1329, 1989.

193. P. D. Thouin and C.-I. Chang, "A Method for Restoration of Low-resolution Document Images," *International Journal on Document Analysis and Recognition*, pp. 200-210, 2000.

194. P. D. Thouin and C.-I. Chang, "Automated System for Restoration of Low-resolution Document and Text Images," *Journal of Electronic Imaging*, vol. 10, pp. 535-547, 2001.

195. S. Tominaga, "Color Image Segmentation Using Three Perceptual Attributes," Proc. of Conference Computer Vision and Pattern Recognition, 1986, pp. 628-630.

196. S. Tominaga, "A Color Classification Method for Color Images Using a Uniform Color Space," Proc. of 10th International Conference on Pattern Recognition, Atlantic City, New Jersey, 16-21 June 1990, pp. 803-807.

197. A. Tremeau and N. Borel, "A Region Growing and Merging Algorithm to Color Segmentation," *Pattern Recognition*, vol. 30, pp. 1191-1203, 1997.

198. D. C. Tseng and C. H. Chang, "Color Segmentation Using Perceptual Attributes," Proc. of 11th International Conference on Pattern Recognition, 1992, pp. 228-231.

199. D. Wang and S. N. Srihari, "Classification of Newspaper Image Blocks Using Texture Analysis," *Pattern Recognition*, vol. 47, pp. 327-352, 1989.

200. L. Wang and T. Pavlidis, "Detection of Curved and Straight Segments from Gray Scale Topography," Proc. of SPIE Symposium on Character Recognition Technologies, San Jose, California, 1993, pp. 10-20.

201. S. Watanabe and et al., "An Automated Apparatus for Cancer Prescreening," *Computer Graphics and Image Processing*, vol. 3, pp. 350-358, 1974.

202. A. R. Weeks, C. E. Felix, and H. R. Myler, "Edge Detection of Color Images Using the HSL Color Space," Proc. of SPIE Non Linear Image Processing VI, February 1995, pp. 291-301.

203. A. R. Weeks and G. E. Hague, "Color Segmentation in the HSI Color Space Using the K-means Algorithm," Proc. of Nonlinear Image Processing VIII, April 1997, pp. 143-154.

204. J. S. Weszka, "A Survey of Threshold Selection Techniques," *Computer Graphics and Image Processing*, vol. 7, pp. 259-265, 1978.

205. J. S. Weszka, R. N. Nagel, and A. Rosenfeld, "A Threshold Selection Technique," *IEEE Trans. Comput.*, vol. C-23, pp. 1322-1326, 1974.

206. J. S. Weszka and A. Rosenfeld, "Threshold Evaluation Techniques," *IEEE Trans. Systems, Man, Cybernet.*, vol. SMC-8, pp. 622-629, 1978.

207. A. P. Witkin, "Scale-space Filtering," Proc. of IEEE Int. Conf. on Acoustic & Signal Processing, 1984, pp. 39A.1.1-39A.1.4.

208. M. Worring and L. Todoran, "Segmentation of Color Documents by Line Oriented Clustering using Spatial Information," Proc. of 5th International Conference on Document Analysis and Recognition ICDAR'99, Bangalore, India, September 1999, pp. 67-69.

209. V. Wu, R. Manmatha, and E. M. Riseman, "Finding Text in Images," Proc. of 2nd ACM International Conference on Digital Libraries, Philadephia, PA, 1997, pp. 23-26.

210. G. Wyszecki and W. S. Stiles, *Color Science - Concepts and Methods, Quantitative Data Formulas*. New York: John Wiley, 1967.

211. G. Wyszecki and W. S. Stiles, *Color Science, Concepts and Methods, Quantitative Data and Formulae*, 2nd ed. New York: John Wiley & sons, 2000.

212. Y. Yakimovsky and J. A. Feldman, "A Semantics-based Desicion Theory Region Analyser," Proc. of Third International Joint Conference on Artificial Intelligence, 1973, pp. 580-588.

213. C. K. Yang and W. H. Tsai, "Reduction of Color Space Dimensionality by Moment-preserving Thresholding and its Application for Edge Detection in Color Images," *Pattern Recognition Letters*, vol. 17, pp. 481-490, 1996.

214. S. D. Yanowizt and A. M. Bruckstein, "A New Method for image Segmentation," *Computer Vision, Graphics, And Image Processing*, vol. 46, pp. 82-95, 1989.

215. X. Yuan, D. Goldman, M. A., and N. Bourbakis, "Segmentation of Colour Images with Highlights and Shadows using Fuzzy-like Reasoning," *Pattern Analysis and Applications*, vol. 4, pp. 272-282, 2001.

216. L. A. Zadeh, "Outline of a New Approach to the Analysis of Complex SYstems and Decision Processes," *IEEE Transactions on Systems, Man and Cybernetics*, vol. 3, pp. 28-44, 1973.

217. X. Zhang and B. A. Wandell, "A Spatial Extension of CIELAB for Digital Color-Image Reproduction," *Journal of the Society for Information Display*, vol. 5, pp. 61-63, 1997.

218. J. Zhou and D. Lopresti, "Extracting Text from WWW Images," Proc. of the 4th International Conference on Document Analysis and Recognition, Ulm, Germany, August 1997.

219. J. Zhou, D. Lopresti, and Z. Lei, "OCR for World Wide Web Images," Proc. of IS&T/SPIE International Symposium on Electronic Imaging, San Jose, California, 1997, pp. 58-66.

220. J. Zhou, D. Lopresti, and T. Tasdizen, "Finding Text in Color Images," Proc. of IS&T/SPIE Symposium on Electronic Imaging, San Jose, California, 1998, pp. 130-140.

221. S. W. Zucker, "Region Growing: Childhood and Adolescence," *Computer Graphics and Image Processing*, vol. 5, pp. 382-399, 1976.

222. S. W. Zucker, "Relaxation Labelling, Local Ambiguity and Low-level Vision," in *Pattern Recognition and Artificial Intelligence*, C. H. Chen, Ed. New York: Academic Press, 1976, pp. 593-616.