UNIVERSITY OF SOUTHAMPTON

Faculty of Engineering, Science and Mathematics
School of Electronics and Computer Science

Analysis of Craquelure Patterns for Content-Based Retrieval

by

Fazly Salleh Abas

A thesis submitted in partial fulfilment for the degree of Doctor of Philosophy

August 2004

UNIVERSITY OF SOUTHAMPTON

ABSTRACT

FACULTY OF ENGINEERING, SCIENCE AND MATHEMATICS
SCHOOL OF ELECTRONICS AND COMPUTER SCIENCE

Doctor of Philosophy

ANALYSIS OF CRAQUELURE PATTERNS FOR CONTENT-BASED RETRIEVAL

by Fazly Salleh Abas

The advent of multimedia technology has offered a new dimension in computerised applications. Art-based applications are among those which have and will continue to benefit from this advancement. Content-based image retrieval (CBIR) and analysis is attracting attention from museums and art institutions. One of the image-based requirements from museums is to automatically classify craquelure (cracks) in paintings for the purpose of aiding damage assessment using non-destructive monitoring and testing. Craquelure in paintings can be an important element in judging authenticity, use of material as well as environmental and physical impact, which these can contribute to different craquelure patterns. Mass screening of craquelure patterns will help to establish a better platform for conservators to identify cause of damage and a content-based approach is seen as an appropriate path.

This thesis covers the issues of crack enhancement and detection, using a mathematical morphology technique, namely the top-hat operator and also a grid-based automatic thresholding. Craquelure representation aids the processes of craquelure pattern analysis in which the Freeman chain-code is used as a basis for converting the image-based representation into a hierarchically structured numerical form. This hierarchical representation offers several advantages in the sense that detected craquelure patterns can be pruned, according to a certain rule for eliminating suspected noise and insignificant structures. Information can be retrieved in a flexible way, given multi-level access into structural detail. A grouping technique determines 'objects-of-interest' and structured craquelure patterns, named crack-networks are grouped using proximity and characteristic rules. Craquelure patterns are generalised by utilising conservative approximations based on the minimum bounding rectangle (MBR) and rotated minimum bounding rectangle (RMBR). Meaningful features based on orientation histograms and structural statistics are extracted to distinguish between craquelure patterns. The resultant features are used as inputs for a three-stage average distance k-nearest neighbour (k-NN) classifier with fuzzy outputs where the goal is to produce class memberships. A prototype architecture of a craquelure retrieval system is also discussed.

Contents

List of Figures

List of Tables

List of Acronyms

TBIR	Text-based Image Retrieval
CBIR	Content-based Image Retrieval
AM	Application Module
PM	Processing Module
CTH	Closing Top-hat
OTH	Opening Top-hat
SIS	Simple Image Statistics
MBR	Minimum Bounding Rectangle
RMBR	Rotated Minimum Bounding Rectangle
MBC	Minimum Bounding Circle
MBMC	Minimum Bounding m-Corner
MBE	Minimum Bounding Ellipse
CH	Convex Hull
L&M	Label and Merge
M&E	Merge and Expend
QGH	Quantised Gradient Histogram

Acknowledgements

First and foremost, I would like to give sincere tribute to my supervisor, Dr. Kirk Martinez for his guidance and excellent support in this research. He provided me with the proper tools and motivation throughout the three years. My work was also made easier with the help of research colleagues within the IAM Group, namely Dr. Paul Lewis, Dr. Stephen Chan, Dr. David Dupplaw, Mohd. Faizal Ahmad Fauzi and Mike Westmacott. As far as the research is concerned, my contact with Dr. Spike Bucklow of the Hamilton Kerr Institute, Cambridge was a major breakthrough. He generously provided me with necessary literature and materials at a time when the research was in its infancy, which I am most grateful.

Thanks also to my friends within the IAM Group, especially my bay-mates, in alphabetical order, Junaidi Abdullah, Nor Aniza Abdullah, Eric Cooke, Jonathan Hare, Paul Kitchin, Norliza Mohamad Zaini, Dr. Muan Hong Ng, Aniza Othman, Talal Rahwan, Wasara Rodhetbhai, Dr. Victor Tan, Mark Thompson, Dr. Yee W. Sim, Arouna Woukeu and many more, including the Head of IAM Group, Professor Nick Jennings. It has been a pleasure being part of this research group and memories of my time here in Southampton will be cherished for a long time to come.

Outside the research circle, I was encouraged by my friendships, in particular, with Zulfadzli, Yusoff, Ridzuan, Dr. Hafizal, Dr. Pauzi (and his family), Dr. Ahmad Kamsani, Nawal, Jeffry, Nurul Nadia, Wan Noor Shahida and many more too numerous to mention.

I wish to express my gratitude to my family for their unconditional love, support, encouragement and all the beautiful things they have done for me. My beloved parents, Abas Salleh and Che Yom Yahya, as always, have been wonderful. I understand the responsibility of raising me from the day I was born until I become the person I am today and for that, I thank you.

My appreciation also goes to Tuan Haji Nordin Nasir and Badrul of the Sultan Iskandar Foundation, Malaysia, for providing me with the highly important financial support.

Last but not least, thanks to Mr. Phil Myles for the time and effort he put into polishing my written English, correcting many of my bad writing habits.

To my beloved parents...

Chapter 1

Introduction

Every day, a huge amount of data is generated, thanks to the rapidly increasing size of digital storage space as well as the advent of the *World Wide Web* (WWW). Storage and retrieval of digital information is now possible at phenomenal speed, almost unimaginable just a decade ago. Information may contain pictorial data such as images or video sequences, as well as synthetic illustrations, diagrams, charts or computer aided graphics. In short, the huge amount of data is increasingly diverse and can be more than just small chunks of several kilobytes, extending to gigabytes and even terabytes of storage space.

Not only is multimedia information generated at an ever increasing rate, it is also transmitted all over the world due to the expansion of the WWW. Accordingly, the issue of efficient information storage and retrieval is one that requires thorough attention. Information will be less meaningful to users unless it is organized in a systematic way to allow efficient browsing, searching and retrieval.

With the unprecedented amount of information available, it is sometimes hard to find the precise piece of information needed. In a simple scenario where a user issues a simple text query, the one relevant document may be buried within 10000 irrelevant ones. Accordingly, the user has to issue several more queries to cut the list down to a small size. When looking for non-textual information, digital images are a convenient media for describing and storing spatial, temporal, spectral and physical components of information contained in a variety of domains (e.g aerial/satellite images in remote sensing, medical images in telemedicine, fingerprints in forensics, museum collections in art history, and registration of trademarks and logos) [1]. Large volumes of these images make it difficult for users to quickly browse through the entire database.

In an era where multimedia information has a great deal of influence on the human lives,

the quest for efficient information retrieval is inevitable. Compared to text, multimedia information imposes greater demands and challenges than text alone, especially in terms of storage and computation. Computation of information from an image is highly complex compared to information from text. Text is a discrete, symbolic medium; images are non-discrete and involve huge amounts of perception ambiguities, thus requiring significant computational work to gain sufficient information. Image retrieval has been an active area of research for approximately three decades.

Conceptually, image retrieval can be divided into text-based (TBIR) and content-based (CBIR). Text-based image retrieval existed since the 1970s, while content-based image retrieval emerged in the early 1990s. Traditionally, textual features such as filenames, captions and keywords are used to annotate and retrieve images. However, there are several problems with these approaches. First of all, human involvement is required to describe and tag the contents of the images in terms of selected captions and keywords. Beyond that, the spatial relationships among the various objects in an image have to be expressed to understand its content. With the increasing size of the database, the use of keywords becomes not only cumbersome but also inadequate to represent the image content. Keywords are inherently subjective and not unique. The linguistic barrier is also seen as a major problem if the database is to be shared globally. Keywords will be ineffective. Another problem is the inadequacy of uniform textual descriptions of attributes such as shape, colour, texture, etc.

These problems trigger the need for CBIR, where instead of manually annotating the images with keywords, images would be indexed by their own visual content. Some noticeable problems do arise in this approach, such as segmenting images into meaningful regions, extracting features that capture the perceptual meanings and matching images. However, research communities are working actively together to tackle such problems. To date, a vast amount of techniques and frameworks have been developed in this particular direction and further advancements are anticipated in the coming years.

1.1 Content-Based Image Retrieval (CBIR)

The rapid advancement in CBIR-related research over the past few years has been due to the rapid increase in the size of digital image collections [2]. This advancement allows efficient browsing, searching and retrieval. Since the early 1990s, content-based image retrieval (CBIR) has become a very active research area. Many retrieval systems, both commercial and research have been built. There are three fundamental components of a

CBIR, namely *visual feature extraction*, *multi-dimensional indexing* and *retrieval system design* [2].

Feature extraction is the basis of any CBIR system. Within the scope of a visual content, features can be classified as general features and domain-specific features. General features can include shape, colour and texture, while examples of domain-specific features are faces and fingerprints. Due to subjective perception, no single best representation for a given object or image exists. Different individuals have different thoughts about object similarities and representations. Interestingly, even an individual sometimes has different opinions about an object on separate occasions. The use of features varies with applications and user-demands. Thus, a highly flexible CBIR system which can vary its feature selection based on demand is always desired.

In large-scale CBIR systems, an efficient multi-dimensional indexing technique is highly desirable. This is to make the CBIR system truly scalable to large size image collections. A CBIR system must be prepared to overcome two problems: the high-dimensionality of feature vectors and non-Euclidean similarity measures since they may not effectively emulate human perception [2]. To solve the problems, a popular first step is to perform dimension reduction and then use appropriate multi-dimensional indexing techniques, which are capable of supporting non-Euclidean similarity measures.

Most CBIR systems support at least one of the following methods [3]:

- random browsing
- search by example
- search by sketch
- search by text (including keyword or speech)
- navigation with customised image categories

Among the notable CBIR systems are QBIC [4], MARS [5], NeTra [6], Virage [7], Retrieval-Ware [8], Photobook [9], VisualSeek [10], Webseek [11], ART MUSEUM [12], Blob-world [13, 14], etc. Figure 1.1 illustrates the typical architecture of a CBIR system.

1.2 Computer Vision for Art

Following a revolutionary trend, museums and art galleries are beginning to digitise their collections, not just to make them publicly available on the web, but also for internal

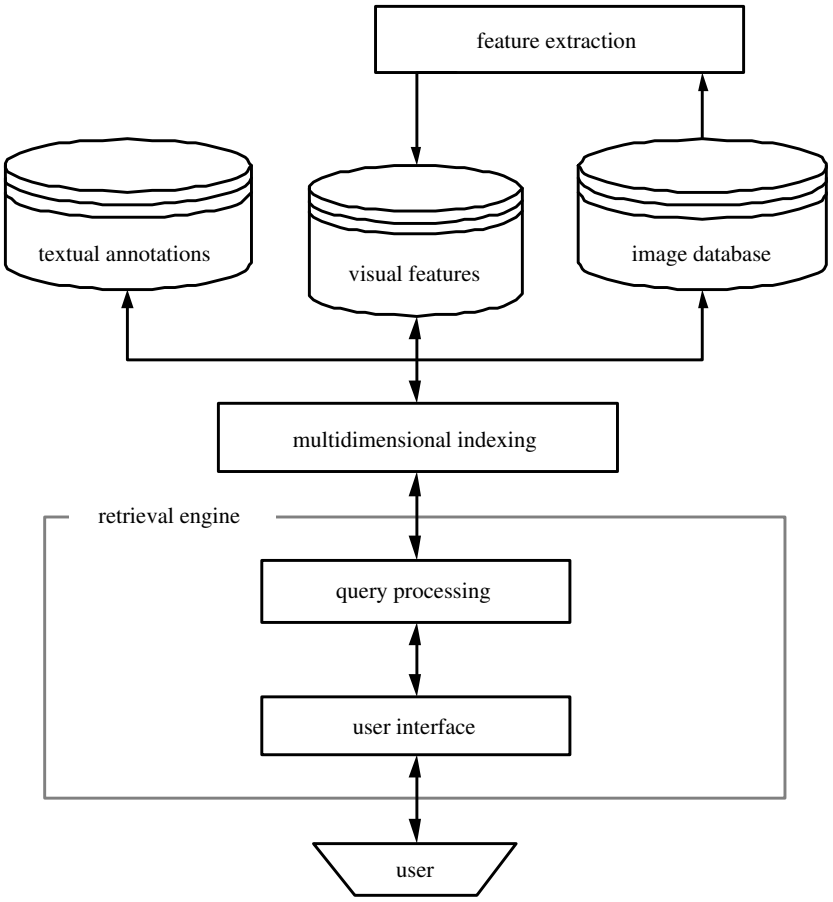


Figure 1.1: Typical CBIR architecture.

use within the museums' or galleries' own environment. Digitising the collection means providing a faster and more efficient way of recording what is available, thus giving a new dimension to methods of information retrieval within the environment itself. Instead of storing the information in a traditional manner, the ability to store it digitally opened the path for further manipulation of the technology, where digital preservation and restoration can play their part. Many paintings and artefacts were created centuries ago and are in need of preservation and restoration, to make sure that their physical appearance is maintained. Manual recording and detection of aging seem less efficient, given the increasing number of collections and electronic-based approaches seems to be the best choice.

Driven by the availability of effective electronic imaging tools and computer vision advancement, significant growth has occurred in research on computer vision relating to art, including quality evaluation of art images [15, 16], image processing tools for art analysis [17, 18], virtual enhancement as well as restoration [19, 20, 21, 22, 23, 24, 25], image retrieval [26, 27, 28, 29, 30, 31, 32] and as an aid for conservation [33, 34, 35, 36, 37, 38, 39].

As regards quality evaluation of art images, Barni et al. [21] presented a new approach

for modelling the human visual system (HVS) which is then implemented in an automatic system for quality estimation of art images. Eastaugh [16] discussed the types of information that can be directly obtained from the scientific study of paintings. The first is a localised analysis of composition that provides information about the materials that the artist chose, while the other is the imaging of these materials across the painting, showing how they were used. Eastaugh concentrated more on the latter in most of the paper, looking into how imaging techniques (i.e. X-ray, infra-red etc.) can be used to give an idea of what can be discovered on and inside a painting.

Research has also been conducted to produce image processing tool for art analysis. Bonachi et al. [17] introduced the ArtShop software, which operates in two directions: quality improvement and virtual restoration. Among the algorithms available in this software are filters for removing noise and crack filling. Müller [18] concentrated on the use of image processing techniques for the interpretation of paintings, the detection of degradations caused by transport and the characterisation of inks in the near infrared. The significance of the work is that Müller implemented a system that allows detection of alterations or damage of a painting after transportion. Using resampling and line detection algorithms and with some interactive steps, changes in cracks are evaluated for possible damage. Climatic conditions, shock and vibration are also taken into consideration in determining the causes of damage.

A significant amount of art-related research has focused on the virtual enhancement and restoration of paintings. Basically, image processing techniques are used to produce a digitally restored version of the original physical artwork (virtual restoration) [21, 24], which is then used for digital archiving. de Polo [20] touched the general issues behind painting restoration and enhancement. One of the issues mentioned is the fact that digital image editing allows old and damaged images to be repaired, restored, preserved, archived and protected. In comparison with traditional retouching techniques, digital technology is better, faster and less expensive. de Polo also explained simple techniques using filters to enhance images for archiving purposes. Smolka and Szezapanski [19] proposed a new efficient algorithm of noise suppression in colour images using a multichannel image enhancement capable of reducing impulsive and Gaussian noise, while Pappas and Pitas [23] presented ways of digitally restoring colour in old paintings. Giakoumis and Pitas [22] performed digital removal of cracks in paintings by using morphological high-pass operators [40, 41, 42], called the top-hat transformation [43], followed by a thresholding process. They also performed a crack filling procedure to fill the empty areas left behind by the cracks. One problem they found was that brush strokes may be detected as cracks, because of their

structural similarities, and to tackle this, they applied a separation technique using hue and saturation information in the HSV (hue, saturation, value) and HSI (hue, saturation, intensity) colour space. A semi-automatic approach is also explained.

Huge collections of art-related information is already available in most museums across the world, and in conjunction with the recent advancements in CBIR technology, image retrieval approaches have been used to perform efficient searches from large databases. The Artiste (Integrated Art Analysis and Navigation Environment) project [26] aimed to provide access across museum collections using metadata as well as content-based retrieval of image data. One aspect of the project involved navigation facilities. A particular objective was to provide a facility for retrieving images from a collection or for navigating to related information in the database, given a query image, which may be only a part of a particular image in the collection. Chan et al. [30] described part of the work done for the ARTISTE project [26] as aiming towards developing a distributed database of art images from European art galleries. A particular objective of their work concentrated on retrieving an image from a collection or navigating to related information in the database, given a query image which may be only a part of a particular image in the collection. More specifically, their work is based on using colour coherence vectors (CCV) extracted from image patches for the query and target images at a range of scales with multiple vector matching in order to find the best sub-image matches. Westmacott et al. [29] demonstrate the use of colour patches to retrieve images from a large archive using colour as the cue for similarity.

Another of the image-based requirements of the ARTISTE project, which originated from the Uffizi Gallery in Florence, is to automatically classify craquelure in paintings to assist damage assessment. Craquelure in paintings can also be used for other research [44], and can be a very important element in judging authenticity and use of material as well as environmental and physical impact, because these can lead to different craquelure patterns. Although most conservation of fine artwork relies on manual inspection for deterioration, the ability to screen the whole collection semi-automatically is believed to be a useful contribution to preservation. Crack formations are influenced by factors including aging and physical impacts, which also relate to the wooden framework of the paintings. It is hoped that the mass screening of craquelure patterns will help to establish a better platform for conservators to identify the cause of damage.

With the advancement of acquisition technologies, paintings can be digitised in various ways. Besides normal photographs, X-radiography (X-ray) [16] and Xeroradiography [1] [45]

[1] A process in which X-radiation forms a latent electrostatic image, usually on a selenium-coated plate, the charged image areas attracting and holding a fine powder (toner) which can be transferred to a paper surface.

introduced a new dimension into the conservation of fine artwork. Details such as brush strokes are not clearly visible in an X-ray image, allowing easy detection of cracks. A similarly important property is the ability of X-rays to capture the physical structures of a painting framework [25] (e.g. stretcher bars, nails, wedges). This will enable conservators to directly relate crack patterns with the presence of these physical structures, allowing more efficient analysis and monitoring. Their presence is also believed to contribute to different crack patterns. The ability to identify these patterns and relate them to the various physical structures is an area which draws quite a lot of attention from art conservators [44, 46]. A content-based analysis of craquelure patterns will introduce more flexibility in performing the task.

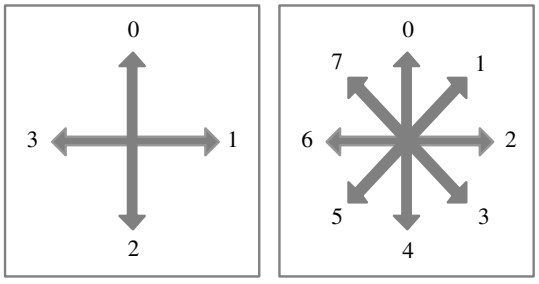
1.3 Shape Analysis

Description of visual data in general benefits from the use of salient features such as shapes, texture and colour. Shapes have long been one of the important cues used for content-based retrieval of images. Shape is a concept that is widely understood yet difficult to define [47]. Among all common features (i.e. shape, texture and colour), retrieval and indexing of the information is a highly challenging task, due to the complexity of the feature description and the difficulties in deriving a suitable similarity matching function [48]. Many contributions have been made by researchers on this particular area with wide a range of applications, namely character recognition, ECG analysis, EEG analysis, cell classification, chromosome recognition, automatic inspection, technical diagnostics, content-based image retrieval, image and video coding.

With the diverse nature of applications and techniques, shape analysis can be classified into two major approaches, the first being a global approach based on global statistics derived from the whole shape of an object (i.e. the global approach), while the second approach concentrates on primitives which are computed from a shape's local regions (i.e. the local approach). Another popular classification of shape representation is as described by Sonka et al. [49], where it is divided into two: contour-based and region-based.

Chain-codes describe an object by a sequence of unit-size line segments with a given orientation. The first information in the chain-code must contain details about its position, to allow the region to be reconstructed. The process results in a sequence of numbers, which are arranged in a chain-like formation. A chain-code C in an ordered sequence of links is written in the form $C = c_1c_2c_3...c_{n-1}$ where n is the number of pixels in the boundary of the object. Omitting the position information results in a position invariant feature.

However, chain-codes are very sensitive to rotation, scaling and noise. A slight change in orientation will cause a major error in object matching using chain-codes. The Freeman chain-code [50, 51] has been used for various kinds of image processing, including finding features of curves/lines [52, 53]. Chain-codes can be implemented as either 4-connectivity (mod 4) or 8-connectivity (mod 8), depending on application suitability (Figures 1.2(a) and 1.2(b)). For a complex contour, a mod 8 chain-code stores a more accurate description of a boundary compared to mod 4. Figure 4.2 shows an example of a contour described by the Freeman chain-code. There are some variations to the implementation of the Freeman chain-code such as the generalised chain-code [54] and the use of the chain-code to extract critical points which can be used to generate a shape description that is invariant to translation, rotation and scaling [55].



(a) 4-connectivity chain-code (mod 4)

(b) 8-connectivity chain-code (mod 8)

Figure 1.2: The Freeman chain-code connectivities.

Another popular form of the chain-code is the primitives chain-code [56] (PCC), which is an extension of the Freeman chain-code, designed to preserve information on branching and junction topology. The fundamental difference between PCC and the Freeman chain-code is in terms of how the contours are coded. PCC introduces codewords in the form of a packed representation of zero to three single-connection direction codes. In other words, contours which comprise one to three pixels can uniquely be represented by PCC codes. Longer chains are represented by a sequence of PCC codes. There are 241 possible configurations for zero to three pixel connections. Unlike the Freeman chain-code, PCC contains additional representative codes such as those for junction and end-points. PCC requires ∼10% to 20% less memory compared to the Freeman chain-code due to the richer syntax of features [57]. A disadvantage of PCC is it is more complex to code and decode compared to the Freeman chain-code. Chain-codes will be discussed further in Chapter 4.

Representing shapes using Fourier coefficients [49, 58, 59, 60, 61] has been common over the last few decades. This methodology uses the Fourier Transform of the 1-dimensional boundary representation to characterise the shape.

The Zahn and Roskies [62] method or more widely known as the Fourier descriptors, uses the tangent angle vs. arc length shape boundary representation. Fourier descriptors present a useful set of shape descriptors that are determined by a spectral analysis of contour deviations from a suitable reference state. It provides a tool (Fourier Transform) to obtain a spatial frequency representation of a particular shape represented by contours.

The Fourier descriptor starts with a polygonal contour represented by a sequence of vertices: $P \equiv \{(x_k, y_k);\ 0 \leq k \leq N-1\}$, where N is the number of samples taken for a boundary. Each coordinate pair are represented as a complex number, $z_k = x_k + iy_k$, in order to convert the 2-dimensional coordinate points into a 1-dimensional representation of complex numbers. From here, the Fourier Transform of the 1-dimensional sequence can be determined using the equation

$$a(u) = \frac{1}{N} \sum_{k=0}^{N-1} z_k e^{-i2\pi uk/N} \qquad u = 0, 1, ..., N-1. \tag{1.1}$$

The resulting sequence of complex coordinates $a(u)$ yields the Fourier descriptors in the form of a power spectrum $|a(u)|^2$. Low-order terms describe the low frequency (slow varying) parts of the shape. Higher terms represent high frequency (fine) components of the shape. The resulting descriptors can be made invariant to rotation, scale and translation. Ignoring the descriptor phase by only taking the magnitude makes them rotation invariant. Normalising the descriptors to the size of the second descriptor $a(1)$ makes them scale independent, and ignoring the first descriptor ($a(0)$) makes them invariant towards translation.

The major advantage of Fourier descriptors is that they are relatively easy to implement and based on a well developed theory of Fourier analysis. The disadvantage is that they do not provide local information about the analysed shape. Shape information is distributed to all Fourier coefficients and not localised in the frequency domain [63].

Moment analysis [64] is also a popular technique for describing shapes. The evaluation of moments represents a systematic method of shape analysis. Moments can be used for binary or grey level region description. In general, a moment of order $(p + q)$ can be manipulated to be invariant to scaling, translation and rotation. For a 2-dimensional signal $f(x, y)$, it is given as [49]

$$m_{pq} = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} x^p y^q f(x, y) \quad dxdy. \tag{1.2}$$

For digital images, the integrals are substituted by summations, and m_{pq} is now calculated as

$$m_{pq} = \sum_{\infty}^{\infty} \sum_{-\infty}^{-\infty} i^p j^q f(i,j) \tag{1.3}$$

where x, y, i and j are coordinate points. For binary images, $f(i,j) \in \{0,1\}$. The first step involves computing the centre of mass or centroid of an object denoted by $(\bar{y}, \bar{x})$ and calculated as

$$\bar{x} = \frac{m_{10}}{m_{00}}, \qquad \bar{y} = \frac{m_{01}}{m_{00}}. \tag{1.4}$$

Translation invariance is achieved with central moments,

$$\mu_{pq} = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} (x - \bar{x})^p (y - \bar{y})^q f(x,y) \quad dxdy \tag{1.5}$$

and for digital images, μ_{pq} is calculated as

$$\mu_{pq} = \sum_{\infty}^{\infty} \sum_{-\infty}^{-\infty} (i - \bar{x})^p (j - \bar{y})^q f(i,j). \tag{1.6}$$

Scale invariance can be achieved by normalising the central moments with respect to the zeroth moment computed as

$$\eta_{pq} = \frac{\mu_{pq}}{(\mu_{00})^\gamma} \tag{1.7}$$

where

$$\gamma = \frac{p+q}{2} + 1. \tag{1.8}$$

If an object doubles in size, then the value of μ increases in proportion to its area and to the powers of the two moment terms. Thus, the denominator of the scale invariant function η increases by the same amount as μ and hence the ratio is scale invariant.

The most commonly used normalised central moment is η_{11}. This provides a measure of deviation from a circular region shape. A value close to zero indicates a region that is nearly circular and a value closer to one indicates a tendency towards non-circularity. There are more properties and manipulations of moments which are not discussed in this section. Features such as the principal major and minor axis (axes of maximum and minimum inertia) are discussed in Section 4.6.2.

Moment analysis can be used for both boundary and contour representation of shapes. No extra processing is needed for spiral and concave contours [49]. It can also be used as descriptors in open boundary cases [65].

Hu introduced seven nonlinear functions defined on regular moments [66], which are translation, scale and rotation invariance. The seven so called moment invariants are based on second and third order moments, used in numerous pattern recognition problems. The moments are given by

$$\varphi_1 = \eta_{20} + \eta_{02}, \tag{1.9}$$

$$\varphi_2 = (\eta_{20} - \eta_{02})^2 + 4\eta_{11}^2, \tag{1.10}$$

$$\varphi_3 = (\eta_{30} - 3\eta_{12})^2 + (3\eta_{12})^2 + (3\eta_{21} - \eta_{03})^2, \tag{1.11}$$

$$\varphi_4 = (\eta_{30} + \eta_{12})^2 + (\eta_{21} + \eta_{03})^2, \tag{1.12}$$

$$\begin{aligned}\varphi_5 = {} & (\eta_{30} - 3\eta_{12})(\eta_{30} + \eta_{12})[(\eta_{30} + \eta_{12})^2 - 3(\eta_{21} + \eta_{03})^2] + \\ & (3\eta_{21} - \eta_{03})(\eta_{21} + \eta_{03})[3(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2],\end{aligned} \tag{1.13}$$

$$\varphi_6 = (\eta_{20} - \eta_{02})[(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{12})^2] + 4\eta_{11}(\eta_{30} + \eta_{12})(\eta_{21} + \eta_{03}), \tag{1.14}$$

$$\begin{aligned}\varphi_7 = {} & (3\eta_{21} - \eta_{03})(\eta_{30} + \eta_{12})[(\eta_{30} + \eta_{12})^2 - 3(\eta_{12} + \eta_{03})^2] - \\ & (\eta_{30} - 3\eta_{12})(\eta_{21} + \eta_{03})[3(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2],\end{aligned} \tag{1.15}$$

where η_{pq} are scaled centralised moments which can be computed from Equation 1.7. The first six orders of the moment invariants, φ_1, φ_2, φ_3, φ_4, φ_5 and φ_6 are invariant under rotation, scale, translation and reflection, but the seventh order moment φ_7 is sensitive to reflection. Instead of using the actual values, logarithms of the seven invariant moments are taken to reduce the dynamic range.

1.4 Description of the Problem

Figure 1.3 shows an example of cracks in a painting as an introduction to the type of image being discussed. As can be seen, the crack patterns are clearly visible and can be easily

identified by the human vision system. As in the figure, the cracks are represented by dark pixels, while the background is represented by the brighter ones. Obviously, as far as the human visual system is concern, segmenting the appropriate crack affected regions does not pose a problem, as long as the image is not highly distorted by noise. To automatically segment the image within a certain tolerable error range is quite problematic, especially when dealing with a large collection of cracked images with different levels of illumination, contrast, noise and intensity. In large collection, it is difficult to monitor every single painting to spot cracks. Digitisation of painting collections, including X-radiographs of paintings, serves the need for a content-based approach in analysing crack patterns. X-ray images in this case tend to show cracks very well as the details in the paint layer are suppressed.

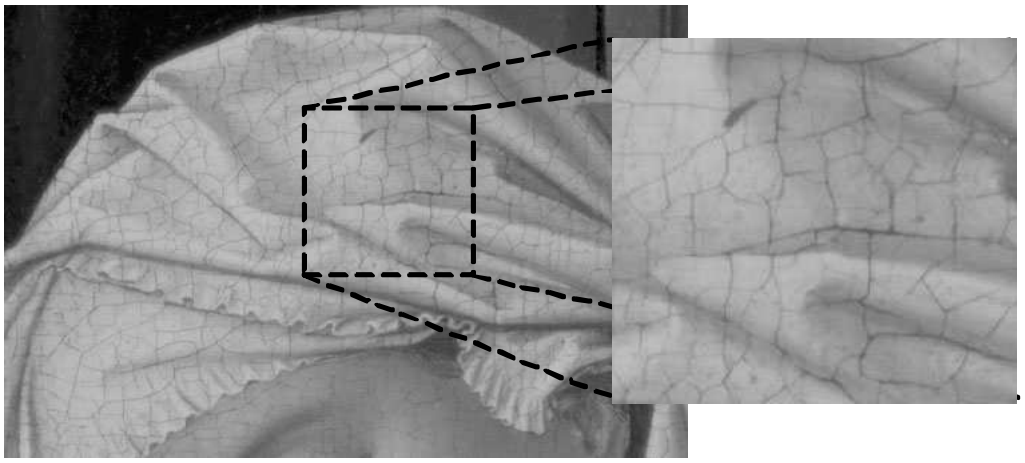


Figure 1.3: Example of a painting with cracks.

A bigger challenge after the detection stage is to extract meaningful features from the detected cracks for classification purposes. "Meaningful" in this case refers to the capability of each feature to distinguish a particular type of crack from another. Generally speaking, cracks can be classified into many types, depending on how they are observed. As mentioned before, cracks can be used to judge authenticity and that is the case in [44], where cracks are classified into 4 different categories namely, French, Flemish, Dutch and Italian. The relevant classes introduced in this research take into account probable effects and physical or environmental factors perhaps imposed upon the paint layer. Support structures such as wooden stretcher bars, nails, wedges and wooden joints are believed to introduce different patterns of cracks. The formation of cracks due to those physical factors introduces a problem that they remain undetected for a long period of time. Although the support structures are meant to facilitate restoration, their purpose is not fully justified, since they contribute to the formation of cracks on the paint layer.

Based on observations, different crack patterns are caused by the type of support structure used by conservators. The common cracks are *circular*, *rectangular*, *unidirectional*, *spider-web* and *random*. A large portion of this project is dedicated to the highly challenging task of automatic classification of crack patterns into these 5 classes.

1.5 Thesis Overview

The research presented here generally falls into 5 main parts: craquelure detection, craquelure representation, pattern grouping, high-level feature extraction, pattern classification and system implementation.

Chapter 2 describes craquelure from a conservator's point of view, looking at its appearance, how it forms and its relation with artwork conservation. The conceptual background of the project is explained in this chapter. Questions are also asked about the possibility of implementing CBIR functionalities for the purpose of assisting conservation work. A content-based approach towards analyzing craquelure is seen as the vision of the project looking into possible application scenarios: *query for similar pattern* and *craquelure information query*. Initial architecture modules are also planned as guidelines for the work. The architecture comprises two modules, namely the *application module* and the *processing module*.

Chapter 3 touches on the technical side of the project by first reviewing common line detectors. Further in the chapter, the algorithm used to enhance crack patterns using morphological top-hat operators is explained. Then the technique used to segment the crack patterns is described using a variable thresholding technique followed by a grid-based automatic thresholding approach with the aid of the Otsu thresholding algorithm, before experimental results are presented and discussed in the later parts of the chapter.

Chapter 4 explores the approaches taken to create a structured craquelure model in what is called a crack-network. A method based on a chain-code *crack following* technique is explained in detail. Features are organised in a hierarchical structure to allow efficient manipulation. A crack approximation scheme, is also described using the minimum bounding rectangle and the rotated minimum bounding rectangle. A pruning process is presented at the end of this chapter, which aims to reduce the amount of detection error and also to remove insignificant crack patterns. Results are presented to show the effectiveness of the algorithm.

Chapter 5 then covers issues regarding object-of-interest determination. The tasks under-

taken to group crack-networks which are suspected to belong to the same object-of-interest are explained. This stage is important in the sense that it prepares the analysed images for content-based analysis. In order to perform query, matching and result presentation in a content-based format, the objects-of-interest must be identified. In order to achieve this objective, a two-stage approach is applied, whereby crack-networks are first grouped using proximity rules, while the second stage utilises descriptive features as cues for hierarchical clustering. In order to separate patterns into objects-of-interest, a technique is developed, which determines an optimum number of clusters out of a cluster hierarchy. Results are presented at the end of the chapter.

Chapter 6 discusses the strategies taken to extract meaningful features from the grouped crack patterns. At this stage, it is assumed that each resulting object represents a meaningful crack pattern. Features extracted from these objects are used for classification. Experiments were carried out with several potential features based on orientation histograms and also statistical measures. Separate experiments were also conducted in order to look at the discriminating power of each feature. This chapter also touches on a more challenging issue of classification. A background review on classification techniques is presented and critical issues are outlined. A three-stage classification approach using the average distance k-nearest neighbour rule is employed for classification and experiments are conducted to evaluate the performance of the chosen classifier.

Chapter 7 consists of approaches used in designing the architecture for a content-based craquelure retrieval. This chapter integrates the separate stages explained in the earlier chapters into a functional system prototype which in the end aims at providing classification of craquelure patterns in various ways. Specifically discussed are the implementations of *query using feature matching*, *query using fuzzy set matching* and *query using class matching* as a prototype result interface for craquelure pattern analysis.

Chapter 8 concludes the dissertation by outlining the contributions of this work and discussing potentials for the future.

Chapter 2

Content-based Analysis of Craquelure Patterns

2.1 Introduction

This chapter highlights the conceptual motivation of the project, generally touching on issues from the physical structure of a painting framework to the potential relationship between craquelure and support structures. The issues behind the implementation of a CBIR approach for the analysis of painting cracks are also discussed. Finally, probable application environments, scenarios and proposed system architecture are explained.

2.2 Physical Structure

Eastaugh [16] mentioned two types of information that can be directly obtained from the scientific study of paintings, which are the localised analysis of composition, which informs us about the materials the artist used, and imaging of these materials across the painting, showing how they were used. Eastaugh concentrates on the latter and further divides it into two more sub-sections, which are image interpretation and image formation. Image formation is how different types of radiation (i.e. infra-red, X-ray, ultra-violet etc.) interact with the picture's structure. This, according to Eastaugh, helps to determine what the picture is made of and how those materials are assembled. Basically, in most cases, a picture may have 5 elements arranged in layers:

1) A support (such as a canvas or wood panel).

2) A preparation layer (often referred to as a ground).

3) A preparatory design, such as drawing.

4) The paint layer(s).

5) Transparent surface coating (varnish).

2.3 Crack Formation

As mentioned by Müller [18], the paint layer is protected by the varnish and problems occur when the originally transparent varnish becomes yellowish or greenish or simply loses its transparency during ageing. As the paint layer ages, the solvent is no longer capable of keeping the paint layer intact, thus cracks begin to form. This is the most common reason behind the forming of cracks. Other known reasons include physical tensions within the structure of the painting and external impacts such as in human handling. Depending on acquisition resolution, a typical crack pattern possesses width of 1 to 4 pixels for a resolution of 20 pixel/mm [18].

These faults often occur during transportation of paintings when on loan for exhibitions. Figure 2.1 shows the steps taken to transport paintings and where *condition report* is established [18]. During transportation, the painting is exposed to external shock or vibration. The *condition report* stage is performed by assessing differences between two events (as an example, before packing and leaving the museum and after returning and being unpacked). The drawback of having frequent *condition reports* lies in the increasing number of acquisitions, and will be difficult to achieve or even impossible if a mobile acquisition system is not made available [18].

Another important factor influencing painting is climate change. Variation of temperature, humidity and pressurisation are elements which cause cracks to form. In transportation via air for instance, problems usually arise when the pressure in the cargo compartment is lower than in the cabin [18].

According to Bucklow in [46], the pattern of cracks is a visible record of the physical tensions within the structure of the painting. The ways in which tensions are generated and dissipated are dependent upon the choice of materials and methods of construction employed by the artist. It is stressed that auxiliary supports[1] contribute to patterns in the craquelure. Bucklow also mentioned in [44] that on canvas paintings, there is perceived interaction between the canvas and the stretcher, which creates *keying out*, *tacking edge* and *stretcher bar* cracks. Interactions between the canvas and the environment are also

[1] The stretcher upon which the canvas is held.

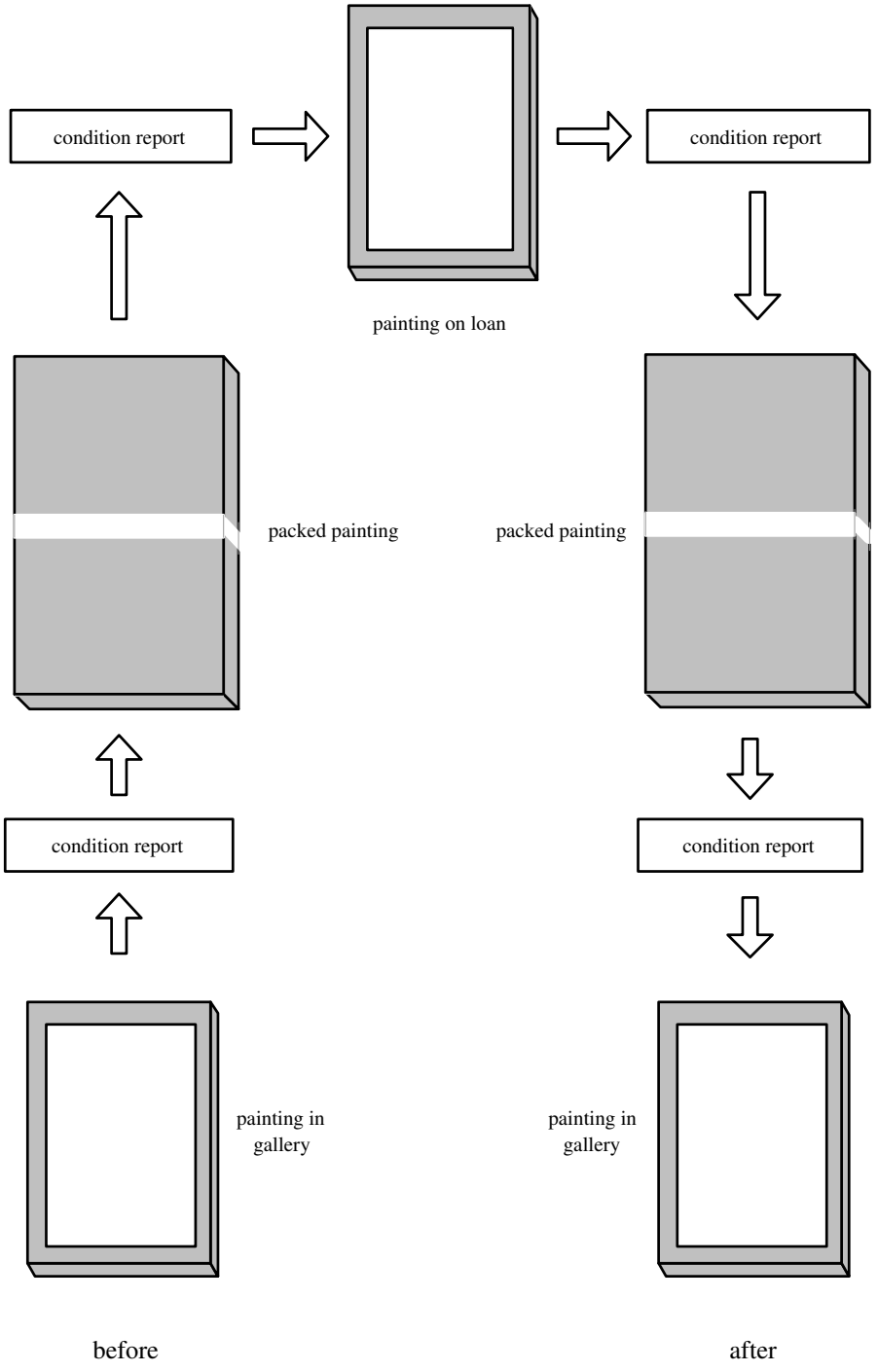


Figure 2.1: The important steps during transportation of paintings.

evident in *circular* and *herring-bone* cracks, the results of mechanical impacts and scrapes respectively; it is asserted by some that circular cracks can also be caused by points of tension, such as knots in the canvas.

2.4 Condition Reports

Muller reported in [18], the technique in which conservators report painting condition relies heavily on written descriptions supported by appropriate photographs and diagrams. Even if the best conservators are up to the task, it is still difficult to produce a perfect *condition report* and it can also be easily misinterpreted by another conservator [67].

There are some early examples of how traditional conservators in the yesteryears performed *condition reports*. Stolow and Wennberg [68] reported that in 1968, twenty-five paintings were transported and their conditions surveyed. This was done by taking photographs of five details for each painting. Each of these details cover an area of 4cm^2. One is situated in the centre of the painting and the remaining four near the corners. These are the locations at which mechanical load is expected to be among the highest. Several pieces of damage were detected, and it was concluded that where one or two new cracks were observed, it might be assumed that in reality perhaps more cracks had developed in the picture. At the end of the experiment, Stolow and Wennberg came to the conclusion that much larger areas of a painting should be photographed, and the idea was later extended to the whole painting by Muller [18].

Muller's work was motivated by the necessity to photograph the whole painting. This could be done by taking many photos and comparing results before and after transportation. However, in practice, it is not easy, since there are problems of different viewing angles, lighting effects and image quality. In addition, it is almost impossible for a human observer to compare images and spot the differences. Bearing in mind the large dimensions of some paintings, efficient (accurate and fast) monitoring is very unlikely to be achieved.

Whenever a painting is monitored the issues are subtle and damage may easily be overlooked. Cracking often occurs in the ground-layer first and the paint layer may not be penetrated until years later [69]. Normal photography is known to fail in detecting ground-layer cracking and this is where X-radiography [16] and Xeroradiography [45] can be useful. As reported by Murray et al. [70], structures of 0.15cm or, at certain angle, less than 0.02cm can be achieved.

The VASARI (Visual Art System for Archiving and Retrieval of Images) project [71] aimed

to establish an acquisition system for high resolution digital images of paintings. The problem of colorimetric acquisitions and systems for computer aided learning for art historians have been looked into and general image processing software has been developed. The important issue behind the VASARI project was for it to be user driven which made it possible to respect museums' needs as far as possible, and not, as in many other applications, to find compromises in adapting systems designed for completely different applications.

2.5 The Link Between Craquelure and Conservation

Bucklow [46] highlights the potential of craquelure as a non-destructive means of identifying the structural components of paintings, where the pattern features can reveal the nature of the support (canvas or wood), species of tree (poplar or oak), type of panel preparation (gypsum or chalk, thickness of ground layer, and size of particles), and type of canvas preparation (glue or oil based grounds). Bucklow also mentioned how craquelure can provide potential non-destructive means of accessing the mechanical strengths of the structural components of paintings, which can be of very significant value to conservators [44].

Cracks can serve great deal of assistantance to painting conservators in terms of providing them clues to degradation on the paint surface. Based on the proof given in the previous section, it is clear that craquelures do in some cases originate from physical impact to the structural part of a painting. As far as conservation is concerned, these clues can be utilised for two main purposes: continuous paint layer degradation monitoring and a more in-depth study of how physical and environmental factors contribute to the existence of cracks, so that steps can be taken to reduce them.

Due to the huge amount of digitised painting related images, it is far from efficient for conservators to monitor cracks by manually searching for them. Most digitised images in well-known collections, as in the National Gallery of London, the Louvre Museum in Paris and the Uffizi Museum in Florence are of very high resolution. It is just impractical for a conservator to do a manual search on such large collections. Even checking over a single painting for a particular feature can be a very exhausting work. As tools for artwork restoration, image processing techniques serve different purposes.

Most of the previous research on fine art restoration has concentrated on the virtual restoration of digitised paintings. Giakoumis and Pitas [22] developed a method for virtual restoration of cracks on paintings using a morphological top-hat operator to detect crack patterns

followed by an implementation of the MRBF neural network to separate brush strokes from cracks. Later they propose two crack-filling methods based on order statistics and anistropic diffusion.

Barni et al. developed a virtual restoration system to remove cracks on digital images of paintings [21]. Their method is based on a semi-automatic crack detection procedure, where users need to specify a point (pixel) believed to belong to a crack-network. The algorithm will then track other suspected crack points based on two main features, absolute grey-level and crack uniformity. Once the algorithm has completely detected cracks, the user can decide to erase the cracks by interpolation.

An area which is thought to be of a great use to artwork conservation is content-based analysis. Bucklow in [46] mentioned the possibility of utilising craquelure for information storage and retrievals where methods of converting visual structures into machine-manipulable representations are necessary for the content-based retrieval of information from image databases. A fine example of such a system is the ARTISTE (An Integrated Art Analysis and Navigation Environment) project [26, 72], a European project developing a cross-collection search system for art galleries and museums. It combines image content retrieval with text based retrieval, and uses RDF (Resource Description Framework) [73] mappings in order to integrate diverse databases.

ARTISTE developed a sophisticated image retrieval system, which analyses and cross-links artworks in four of Europe's greatest museums, namely London's National Gallery, London's Victoria Albert Museum, the Louvre in Paris and Florence's Uffizi Gallery.

The core feature to many of the objectives laid out in the ARTISTE project is image content analysis. This feature enables users to submit an image to the system and request "similar" images to be returned as results. "Similar" in this sense means homogeneous in terms of either colour, texture, shape or any other attribute considered as important. A sub-image can also be used as a query with a request to search for the parent image.

With the advancement of CBIR techniques and the growing number of applications deriving direct benefit from it, historians and conservators can expect a big leap from traditional methods to state-of-the-art conservation approaches, if CBIR technology can be fully utilised.

This dissertation attempts to underline the issues related to the implementation of the CBIR approach to the analysis of craquelure patterns in paintings for the purpose of conservation.

2.6 CBIR Implementation Issues

The design of a content-based retrieval system depends on requirements driven by users. However, in some cases, what machines are capable of doing does not fully conform to these needs. This section briefly explains the main issues in designing a simple system capable of performing content-based analysis of craquelure patterns. Obstacles and implementation issues can be drawn out from the computer vision point of view and also from the requirements of potential users.

2.6.1 Detection Accuracy

From the computer vision point of view, the process of identifying crack patterns is believed to be a very critical step. The relative difficulty in detecting cracks depends on whether their shape and typical orientation are known *a priori*, whether they start from the edge of the object, and whether the texture is periodic or random. A key problem is the typically very small transverse dimensions and poor contrast of cracks. The human visual system may easily detect them, but they may actually consist of "chains" of non-adjacent single pixels in the image. In some of the worst cases, the surface is highly textured (with brush stroke patterns) and this will certainly pose a problem for the detection stage. However, details may not be too important since in classifying craquelure patterns the key features to detect are the dominant ones.

2.6.2 Feature Selection

Feature selection is very important, since good pattern discrimination can only be achieved if highly distinguishable features are used. Selecting relevant features is not an easy task, firstly since there is no clear grammar or language that can explain precisely how a particular crack pattern differs from another. The scenario certainly does not mimic that of the optical character recognition (OCR) problem [74, 75, 76, 77] where each alphabetical/numerical character has unique structural features allowing relatively easy classification, assuming ideal block-based characters. This is not the case for cracks, since the perception towards a particular pattern varies with respect to the observer and the scope of view (shape and scale).

2.6.3 Objects-of-interest

Content-based processing does not assume everything within an image to belong to the same object-of-interest. An image contains objects with different shapes, sizes and appearance.

Taking an easy example, a car is a parent object to sub-objects such as the wheels, doors and body. Each of these objects is distinguishable by their colour, texture or shape. Looking at a slightly more complicated scenario, a human face can be further segmented into several obvious sub-objects, namely the ears, nose, eyes and mouth. These sub-objects are quite similar in colour but differ in terms of their shape and also position on the face.

The scenario is much more challenging when attempting to segment crack patterns which, to begin with, do not contain any colour information, possess similar texture properties and offer no prior knowledge of any positional attributes. The only feature that can be used to extract the objects-of-interest is the shape of the cracks and how these structures combine.

Another observed complication stems from the issue of subjective perception, where crack patterns change with varying viewpoints, i.e. the scope of view. General observers might have different opinions about the likelihood of the objects-of-interest.

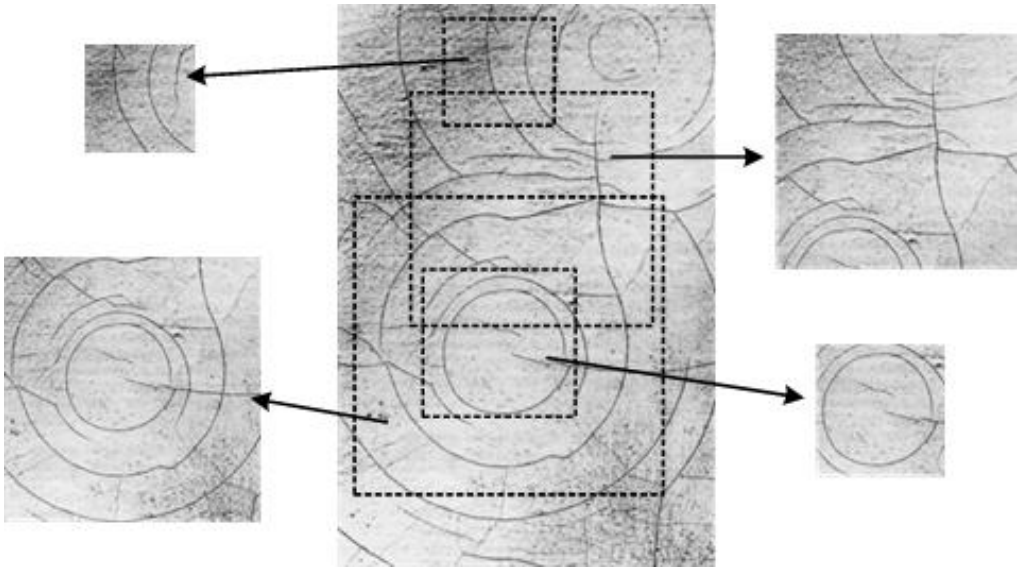


Figure 2.2: Subjective perception regarding crack pattern change with different viewpoints.

This is seen as a very big challenge. As shown in Figure 2.2, the view in which crack patterns are observed affects the perceptive notion as to how they are classified. Furthermore, crack patterns consist of multiple line segments either connected or separated.

2.6.4 Hard versus Fuzzy Class Assignment

Another computer vision related issue concerns the classification of crack patterns. As mentioned earlier, there is no existing standard grammar or language to describe crack patterns. It is not quite true to say that a certain crack pattern exclusively belongs to a particular class. Unless a crack pattern totally agrees to a particular class descriptor, while contradicting to descriptors of the classes, it can be assumed that each crack pattern is a member of every crack class but with a varying confidence value. Experiments are conducted in the later stages of the dissertation to show the significance of the claim.

2.6.5 Retrieval Efficiency

To be able to serve potential users with retrieval functionalities, the system should be able to process and access data in a considerably short time. Hence speed is a highly desirable design element, which can directly result from efficient handling of data queries and quick database look-up. Museum image collections which include X-ray and visible images are very large in size (some exceeding 10000x10000 in pixel resolution), thus requiring efficient algorithm executions. The system's approach towards the implementation of content-based functionalities also poses some questions, one of these being the type of query and whether or not it is useful for the end-users, bearing in mind that such applications might only be useful for fine artwork conservators. Relevant query types in this context are query by example and query by type (text-based query). Examples of these queries are further elaborated in Section 2.7.

2.7 Application Scenario

The capabilities of image analysis and processing for conservators can prove to be very useful if the process can provide users with the functionalities that manual processes fail to serve them. Among the common problems of manual defect screening are time consumption and the risk of further damage. Desired information should be retrieved correctly (i.e. as close as possible to the users' individual perception) and consuming considerably less time and effort. The following sections briefly outline potential application scenarios from a user point of view.

2.7.1 Scenario 1 : Query for Similar Pattern

The main requirement to a content-based analysis of cracks will be to query for a particular region or image for a similar type of crack patterns. This scenario can attract special attention from users who are interested in investigating the relations between cracks of near-similar patterns. As mentioned in the earlier chapters, crack patterns can be judged from two points of view: authenticity and damage assessment. This type of scenario would definitely suit both, given that good type descriptors are used to efficiently describe crack patterns. Scenario 1 is as follows:

A user issues a query on the possible forms/actions listed below.

(a) Uploading an example of a crack image from an external or an internal source.

(b) Selecting a particular region from a large image.

(c) Specifying a particular type/class of crack pattern from a list.

Results of the query can be of the following forms.

(a) Highlighted regions of the best matches and some associated data (i.e. location of bounding rectangle, dimension, statistical values).

(b) List of image regions containing the specified crack patterns.

Figure 2.3 shows an example of a scenario where a user uploads an image from a database (query type (a)) and results are shown in the form of a list (result type (b)).

Figure 2.4 illustrates an example of the scenario where a user specifies a class type (query type (c)) and results are listed down (query type (b)).

2.7.2 Scenario 2 : Craquelure Pattern Analysis

The second scenario needs less processing time compared to the first since it does not perform pattern matching. It focuses on providing users with the ability to interactively obtain information about a particular region of an image. Conservators in particular are interested in statistical values of crack patterns in a particular area including the type/class into which they fall. This functionality considered useful for pattern learning or even for random inspection of painting surfaces. Scenario 2 is as follows:

A user requests for information by doing either one of the below.

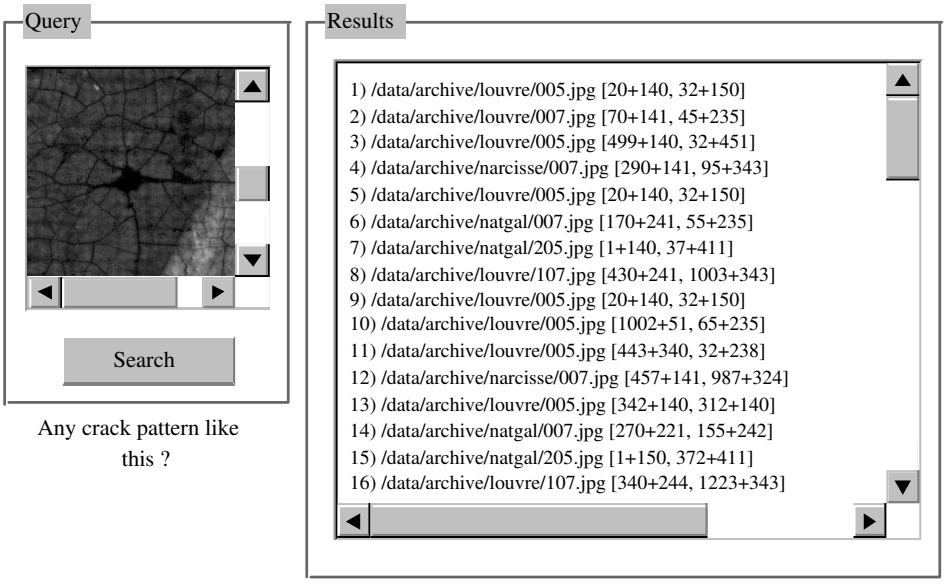


Figure 2.3: An example of *query for similar pattern* using query type (a) and result type (b).

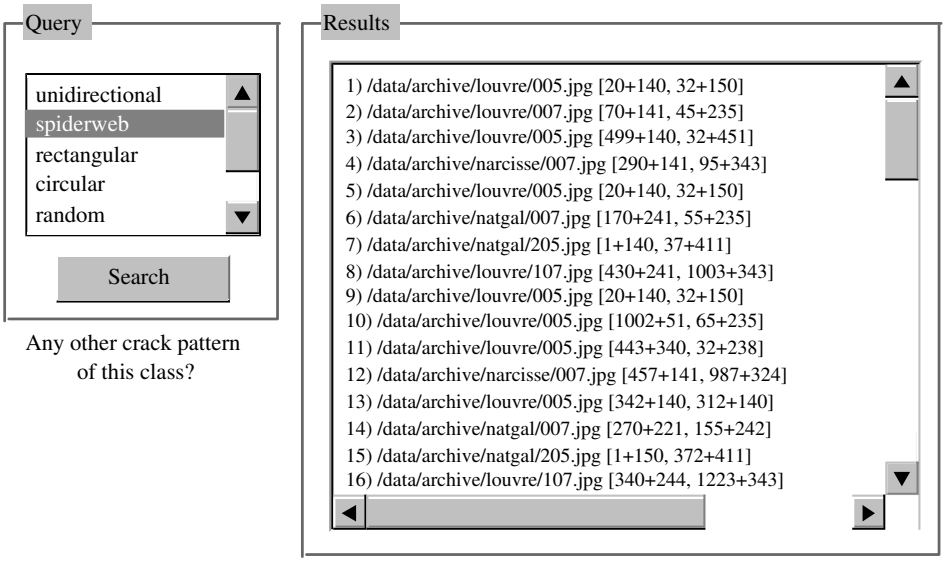


Figure 2.4: An example of *query for similar pattern* using query type (c) and result type (b).

(a) Uploading an example of a crack image from an external or an internal source.

(b) Selecting a region from a crack image.

Results of the user request can be of the following forms.

(a) Highlighted area or feature of interest (i.e. detected cracks, bounding rectangle).

(b) List of information (i.e. class membership, mean crack length, crack density etc.).

Figure 2.5 graphically shows an example of the scenario where a user requests information about an image uploaded from a database (query type (a)) and results are listed down (query type (b)).

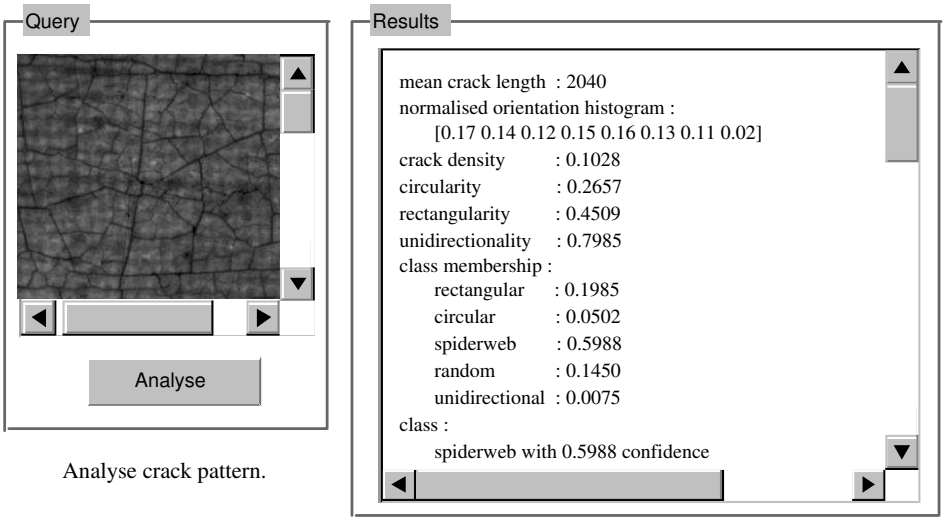


Figure 2.5: An example of *craquelure pattern analysis* using query type (a) and result type (b).

2.8 System Overview

For the purpose of integrating CBIR functionalities in the system, a general outline of the system architecture was designed, which consists of two main modules, namely the *application module* (AM) and the *processing module* (PM). AM deals with information retrieval issues such as querying, feature storage and management, pattern matching and sub-image search, while PM looks at the problem from a more technical point of view, by providing the core computer vision processes. Nevertheless, the prime tasks of the whole system, as in all other CBIR systems, are to accept queries, process data, retrieve information and

present them to users. Figures 2.6 and 2.7 illustrate the proposed architectures of the system corresponding to scenario 1 and scenario 2 respectively.

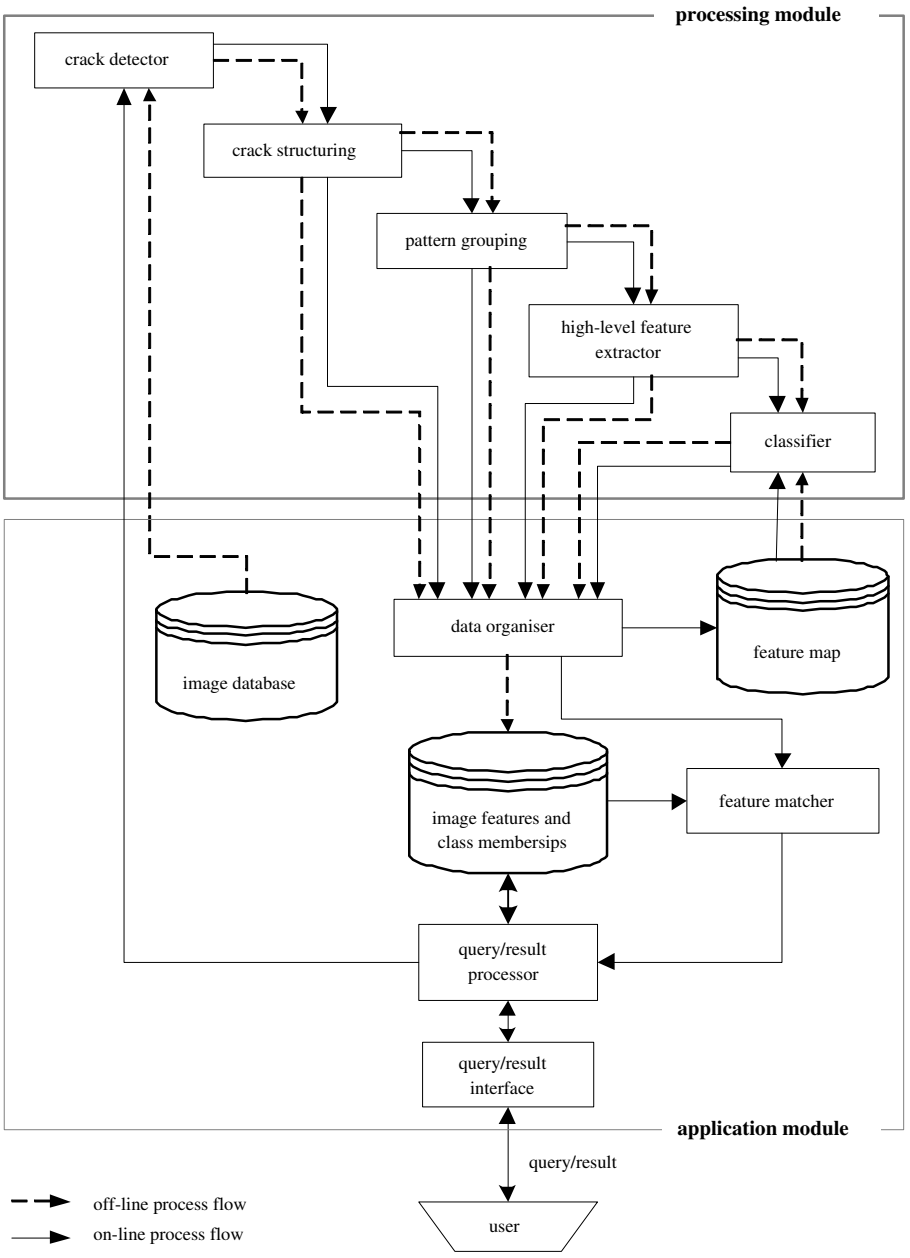


Figure 2.6: General architecture of the proposed system corresponding to scenario 1.

2.8.1 The Application Module

The *application module* makes sure queries are processed correctly. Queries, as mentioned in Section 2.7, can be made in two ways, either by example or by textual description.

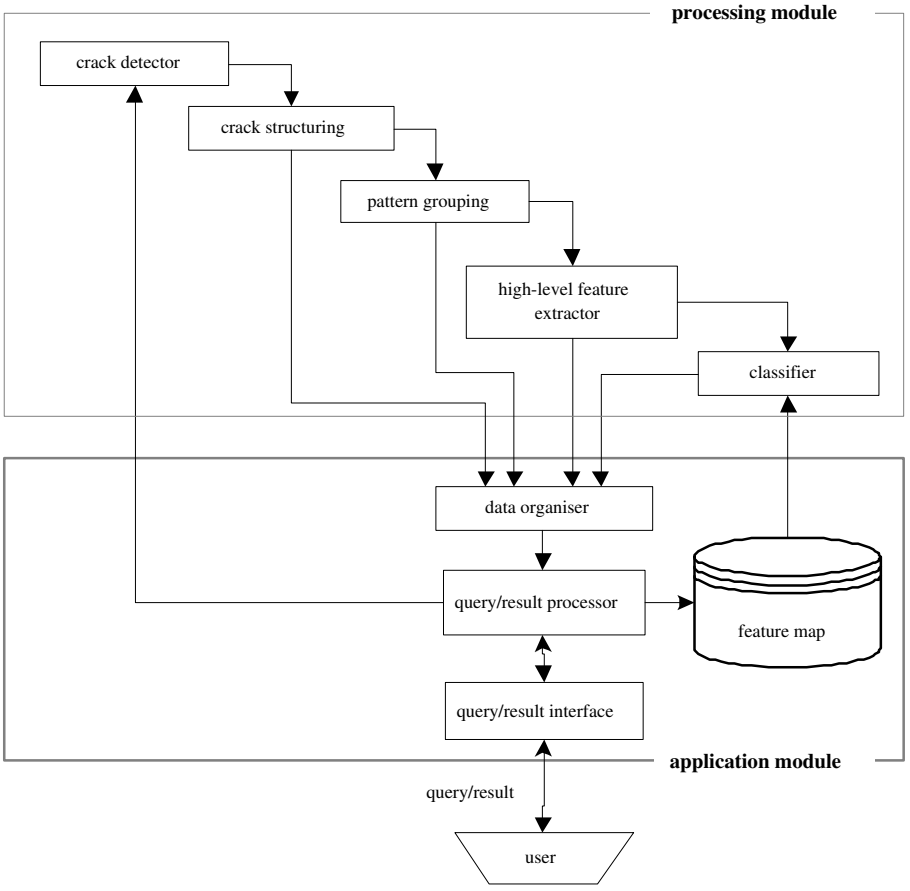


Figure 2.7: General architecture of the proposed system corresponding to scenario 2.

One of the intended tasks of AM is to invoke the correct processes based on the type of query. This task is quite straightforward, and is dealt with by the *query/result processor*. The *query/result processor* acts as a mediator between the *query/result interface*[2] and the databases. It controls the flow of information based on queries and results. For instance, a query based on scenario 1 will cause the *query/result processor* to invoke the databases, while a query described by scenario 2 will involve getting information directly from PM. The three main databases are the *image database*, *feature and classification database* and *feature map database*[3].

The *data organiser* organises the structured data, features and classification results before they are stored in the *feature database*. The *feature matcher* compares features extracted from queried images with those in the database and retrieves the relevant ones for display or presentation to the user.

[2] A command line or graphical user interface to interact with users.

[3] The parameters are the updated values of certain processes such as the trained feature map of the classification sub-module.

2.8.2 The Processing Module

The *processing module* executes image processing and pattern recognition algorithms when invoked by the *application module*. Technically, this module is more difficult to design and implement compared to the *application module*. Theoretically, the role of this module is to process inputs either off-line or on-line from sources which can be from user queries or images in a database. It consists of low-level and high-level computer vision algorithms which act as sub-modules. The sub-modules are crack detection, statistical crack pattern structuring, crack pattern grouping, high-level feature extraction and classification. A large portion of this dissertation is devoted to this module.

2.9 Summary

This chapter has shown how conservation is a main concern for museums as well as art historians and conservators. Paintings are exposed to uncontrolled environment and are under huge threat of degradation. Cracking is one of the major problems which occur mainly because of factors such as ageing, physical impact and acclimatisation. Traditional monitoring techniques, which rely heavily on human effort have proven in the past to be inefficient, in the sense that it takes too much time and tends to introduce inconsistent evaluation.

Craquelure has also been attracting growing interest in the research community, particularly in relation to areas such as virtual craquelure removal, authenticity, etc. With the rapid advancement of computer vision and CBIR approaches, computer-aided systems may be the answer to more efficient artwork monitoring. Acquisition systems have been developed to perform high-resolution scanning of paintings, including X-radiography and Xeroradiography scans, which offer a new dimension to artwork monitoring, particularly on the analysis of craquelure.

Several obstacles were listed as a guide to the advanced stages of this work. Some early strategies were also outlined in designing a CBIR approach for analysing craquelure patterns, which included possible application scenarios, followed by proposed modules for the system.

Chapter 3

Craquelure Detection

3.1 Introduction

Chapter 2 discussed the basis for this research and the common expected obstacles were listed. Before any information can be extracted from a crack pattern, first and foremost, the cracks have to be detected. In other words, the cracks have to be segmented from the background.

Crack-like pattern detection, better known as ridge-valley structure extraction in some literature [78, 79, 80, 81], has been a matter of high concern among researchers, mostly for its potentially useful contribution to a variety of applications. The results presented here have a much wider set of applications than just the analysis of paintings. Many images contain similar patterns, such as biological images of veins and tissues [82, 83, 84], images of fingerprints [85] and multi-spectral satellite photos of rivers or roads [86, 87]. All these examples are current areas of research in the modelling and classification fields for which the results in this dissertation can find an application.

The aim at this stage is to extract or in other words, segment suspected cracks patterns from the background. By noting that cracks are usually considerably darker than the background, and that they are characterised by a uniform grey-level which have an elongated structure, detection can be accomplished on the basis of two main features: absolute grey-level and crack uniformity [21, 22].

Varley [38], assumes that crack patterns in paintings can be segmented by just thresholding the image at a certain manually-selected threshold level. Although the assumption is true, unfortunately the outcome is not quite encouraging when applied. As shown in Figures

3.1 and 3.2, thresholding does not always work. Factors such as inconsistent illumination, noisy presence and low contrast add to the difficulty of obtaining fairly accurate detection.

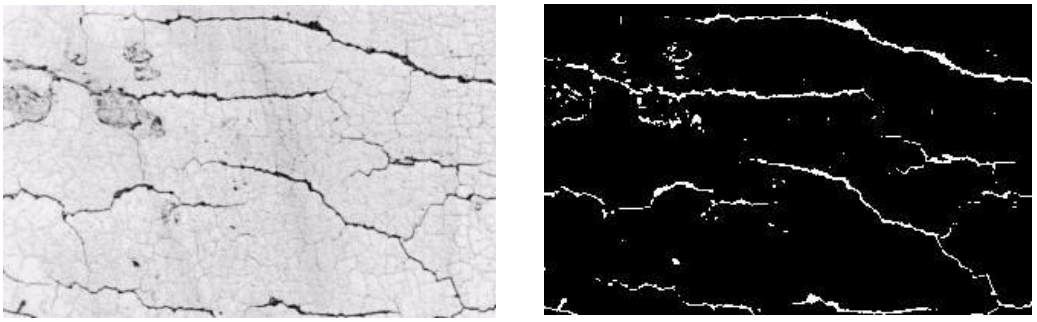


Figure 3.1: An image segmented at threshold pixel value 167.

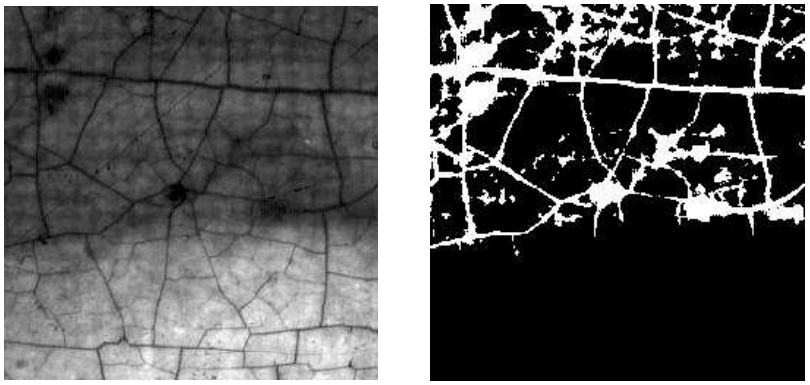


Figure 3.2: An image segmented at threshold value 52.

This chapter is dedicated to the approaches taken to deal with this important step of crack detection.

3.2 Common Line Detectors

A line segment on an image can be characterised as an elongated rectangular region, having grey-level bounded on both its longer sides by homogeneous regions of a different grey-level. For a typical painting crack, grey-levels of the side level have higher values than the centre elongated region containing the dark line. The width along the same line segment may also vary. A general line detector should be able to detect lines in a range of widths.

Among early work on line detection, Vanderbrug suggested a semi-linear line detector created by a step edge on either side of the line [88]. Vanderbrug also developed an algorithm which is originally designed for road detection in satellite images [89]. The method is based

on the response of the image to different masks, allowing estimation of the local variations of the grey-levels. This algorithm, which is also employed by Müller [18], uses 14 masks. Good results are reported if the image is not too affected by noise. However, in the presence of noise, these techniques do not perform well. They are also seen to be very poor in terms of flexibility in order to cope with the unpredictable nature of crack patterns. A very in-depth review of line structure and vessel detection techniques is reported by Kirbas and Quek [90].

3.3 Mathematical Morphology

Mathematical morphology is a part of digital image processing that is concerned with image filtering and geometric analysis by structuring elements. Originally, the theory and application of mathematical morphology was developed for binary images. Its main protagonists were Matheron [91] and Serra [92]. Afterwards, the theory was extended to grey-scale images by Sternberg [40] and later by Haralick, Sternberg and Zhuang [41, 42].

Since the early days, morphological operations and techniques have been applied from low-level, to intermediate, to high-level vision problems. These operations are mainly used for noise reduction and feature detection, with the objective that noise be reduced as much as possible without eliminating essential features.

3.3.1 Basic Morphology Operators

Dilation and erosion are the two fundamental operations that define the algebra of mathematical morphology. These two operations can be implemented in different combinations in order to obtain more sophisticated operations.

Binary dilation is the morphological transformation that combines two sets of pixels by using vector addition of set elements. Binary dilation was first used by Minkowski, and, in the mathematics literature, it is called *Minkowski addition*. If A and B are sets in N-space(E^N), with elements a and b respectively, a=$\{a_1, ..., a_N\}$ and b=$\{b_1, ..., b_N\}$ being N-tuples of element coordinates, then the dilation of A by B is the set of all possible vector sums of pairs of elements, one coming from A and one coming from B. More formally, the dilation of A by B is denoted by $A \oplus B$ and is defined by

$$A \oplus B = \{c \in E^N | c = a + b \text{ for every } a \in A \text{ and } b \in B\}. \tag{3.1}$$

The dilation operation can also be represented as a union of translations of the structuring element:

$$A \oplus B = \bigcup_{b \in B} A_b. \tag{3.2}$$

Erosion is the morphological dual of dilation. The morphological transformation combines two sets by using containment as its basic set. If A and B are sets in Euclidean N-space, then the erosion of A and B is the set of all elements x for which $x + b \in A$ for every $b \in B$. The erosion of A by B is defined by

$$A \ominus B = \{x \in E^N | x + b \in A \text{ for every } b \in B\}. \tag{3.3}$$

The erosion operation can also be represented as an intersection of the negative translations:

$$A \ominus B = \bigcap_{b \in B} A_{-b}. \tag{3.4}$$

Dilations and erosions are usually employed in pairs: a dilation of an image is usually followed by erosion of the dilated result or vice versa. In either case, the result of successively applied dilations and erosions results in the elimination of specific image detail smaller than the structuring element without the global geometric distortion.

The opening of an image is obtained by first eroding the image with a structuring element and then dilating the result using the same structuring element. The closing of an image is obtained by first dilating the image with a structuring element and then eroding the result using the same structuring element.

The opening of A by B is denoted by $A \circ B$ and is defined as:

$$A \circ B = (A \ominus B) \oplus B \tag{3.5}$$

while the closing of A by B is denoted by $A \bullet B$ and is defined as:

$$A \bullet B = (A \oplus B) \ominus B. \tag{3.6}$$

For in-depth mathematical analysis of binary morphological operators, see [92], [93] and [94].

3.3.2 Grey-scale Morphology Operators

Although binary morphological operations serve useful analytical tool for image analysis and classification, they play only a limited role in the processing and analysis of grey-level images. To overcome this limitation, Sternberg and Serra extended binary morphology in the early 1980s to grey-scale images via the notion of an umbra [40, 92]. Similar to the binary case, dilations and erosions are the basic operations that define the algebra of grey-scale morphology. They are combined to produce the grey-scale opening and closing operations which are very useful and effective sets of operations for various computer vision applications. The following paragraphs briefly describe some of the basic definitions of grey-scale morphological operators that can be useful for this research [82].

A discrete grey-level image, A, is defined as a finite subset of Euclidean 2-dimensional (2-D) space, $\mathbb{R}^2$, whose range is $[N_{min},N_{max}]$, $A : \mathbb{R}^2 \rightarrow [N_{min},N_{max}]$, while a 2-D structuring element is defined as a function $S : \mathbb{R}^2 \rightarrow \boldsymbol{S}$, where $\boldsymbol{S}$ is the set of neighbourhoods of the origin. Grey-scale morphological dilation and erosion can be visualised as working with two images, namely the image being processed, I, and the structuring element, S. Each structuring element has a specific shape that acts as a probe. The four basic grey-scale morphological operators are defined with respect to the structuring element S, the scaling factor e, image A and point $M_o \in \mathbb{R}^2$. Grey-scale erosion and dilation are defined as:

$$\text{erosion} \quad : \quad \epsilon_S^e(A)(M_o) = MIN_{M \in M_o + e \cdot S(M_o)}(A(M)), \tag{3.7}$$

$$\text{dilation} \quad : \quad \delta_S^e(A)(M_o) = MAX_{M \in M_o + e \cdot S(M_o)}(A(M)). \tag{3.8}$$

Conceptually similar to the binary case, dilation followed by erosion is a closing transformation, while erosion followed by dilation is an opening transformation, and they are defined as:

$$\text{opening} \quad : \quad \gamma_S^e(A) = \delta_S^e(\epsilon_S^e(A)), \tag{3.9}$$

$$\text{closing} \quad : \quad \varphi_S^e(A) = \epsilon_S^e(\delta_S^e(A)). \tag{3.10}$$

Originating also from grey-scale opening and closing, an operation such as the top-hat operator can be derived.

3.3.3 The Top-hat Transformation

Cracks can be detected with the implementation of a very useful morphological filter known as the top-hat transformation developed by Meyer [43]. These details can be lines or areas with particular sizes. Top-hat operators can function as a closing or opening operator based on the features to extract from an image. Opening top-hat operators (OTH) will detect bright details in an image while closing top-hat operators (CTH) (or also sometimes known as the bottom-hat operator) are designed to detect dark details. Formulation for OTH and CTH are shown denoted by Equations (3.11) and (3.12) respectively.

$$OTH_S^e = A - \gamma_S^e(A) \tag{3.11}$$

$$CTH_S^e = \varphi_S^e(A) - A \tag{3.12}$$

The top-hat operator can be tuned for detection of specific features by modifying two important parameters [22]:

- The shape and the size of the structuring element S. A square-shaped or disk-shaped structuring element may be used. The size must be carefully selected according to the thickness of the crack to be detected.
- The number of times in which erosion or dilation is performed.

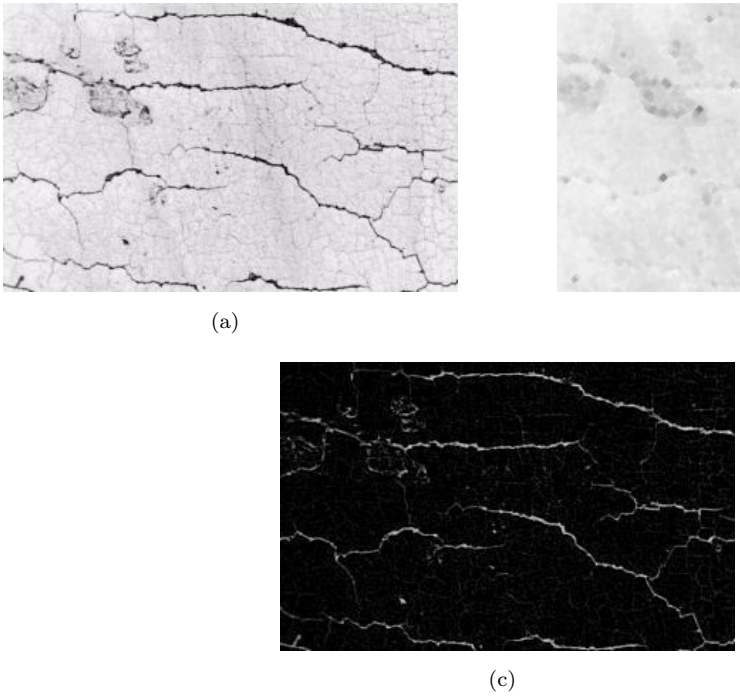
The transformation produces a grey-scale image with the desired features enhanced to a certain level. A thresholding operation is needed to separate the features from the background.

Figures 3.3 and 3.4 show intermediate and final outcomes from the CTH and the OTH operations respectively.

Figure 3.6 shows some results of crack enhancement using the CTH. The images are enhanced using a disk-shaped structuring element of size 5x5, as shown in Figure 3.5.

An attempt is also made to use structuring elements of various sizes in order to properly capture cracks with varying size. Figure 3.7 shows some results from using different structuring element sizes.

Much has been written about detection of edges or vessel-like structures. Zana and Klein [82], described the use of a top-hat operation on retinal vessels in the human eye. The



(a) (b)

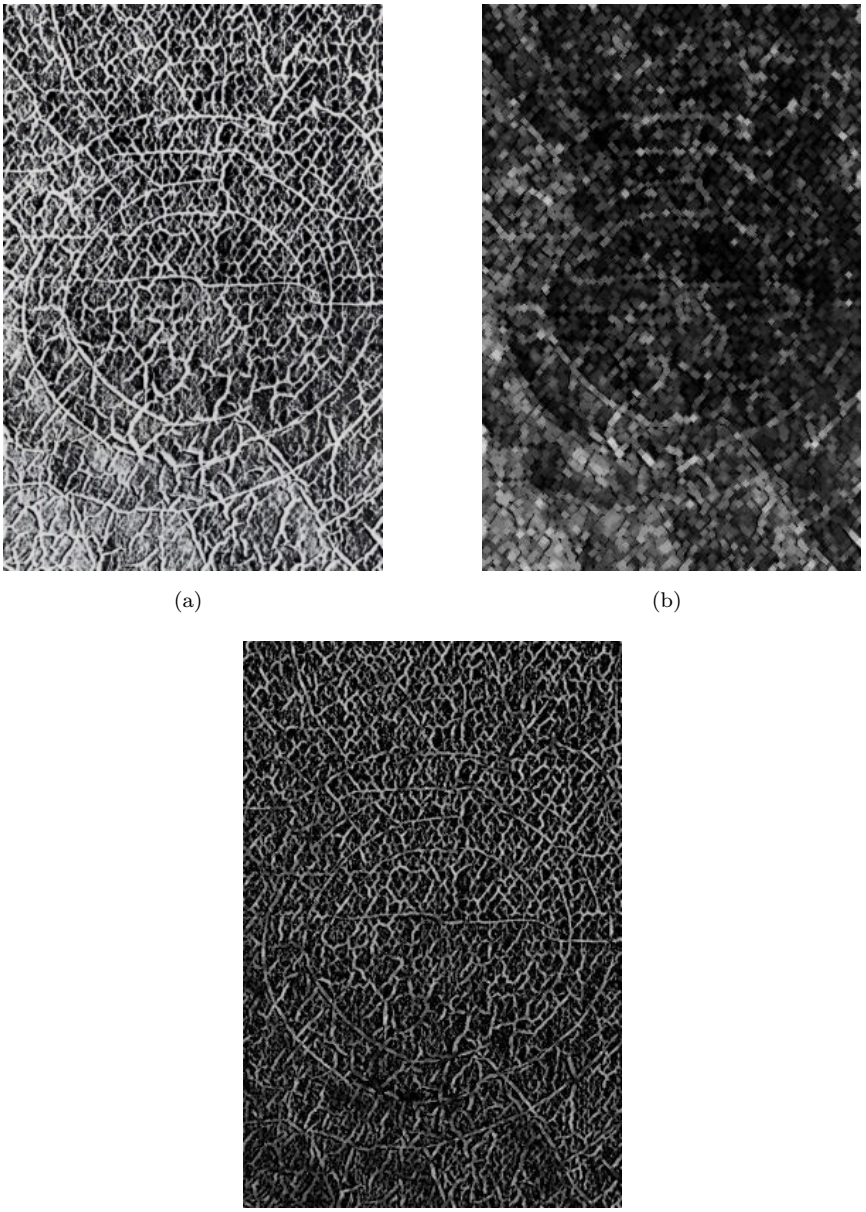
(c)

Figure 3.3: A closing top-hat operation: a) the original image with dark cracks; b) after closing operation; c) after closing top-hat operation.

algorithm used structuring elements (every 15^o) 15-pixels long and 1-pixel wide and as many as twelve structuring elements are used in order to enhance vessel structures at different angles. The sum of top-hats are then taken as the output of the process. There are many other examples of algorithms which use multi-orientation filters, such as the work by Jain et al. [85], which uses rotated matching filters characterised by four differently oriented Gabor filters (every 45^o) to capture fingerprint patterns.

Experiments have been performed using multi-orientation structuring elements for the top-hat transformation as well as using Gabor filters for crack enhancement [33]. Figure 3.8 shows an example of cracks enhanced using horizontal and vertical rectangular structuring elements.

The technique can be implemented for two reasons: to enhance only cracks from a specified orientation, and as a part of a bigger detection algorithm which employs combined filter approach. The combined filter approach accumulates corresponding pixel values from n numbers of filtered images and takes the average value as the final outcome for each pixel (mean pixel value). Another approach takes the maximum pixel value among the filtered images for each pixel location. Taking the mean pixel value will cause a smoothing effect, thus details are suppressed.



(a) (b)

(c)

Figure 3.4: An opening top-hat operation: a) the original image with bright cracks; b) after opening operation; c) after opening top-hat operation.

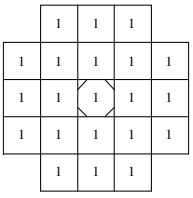
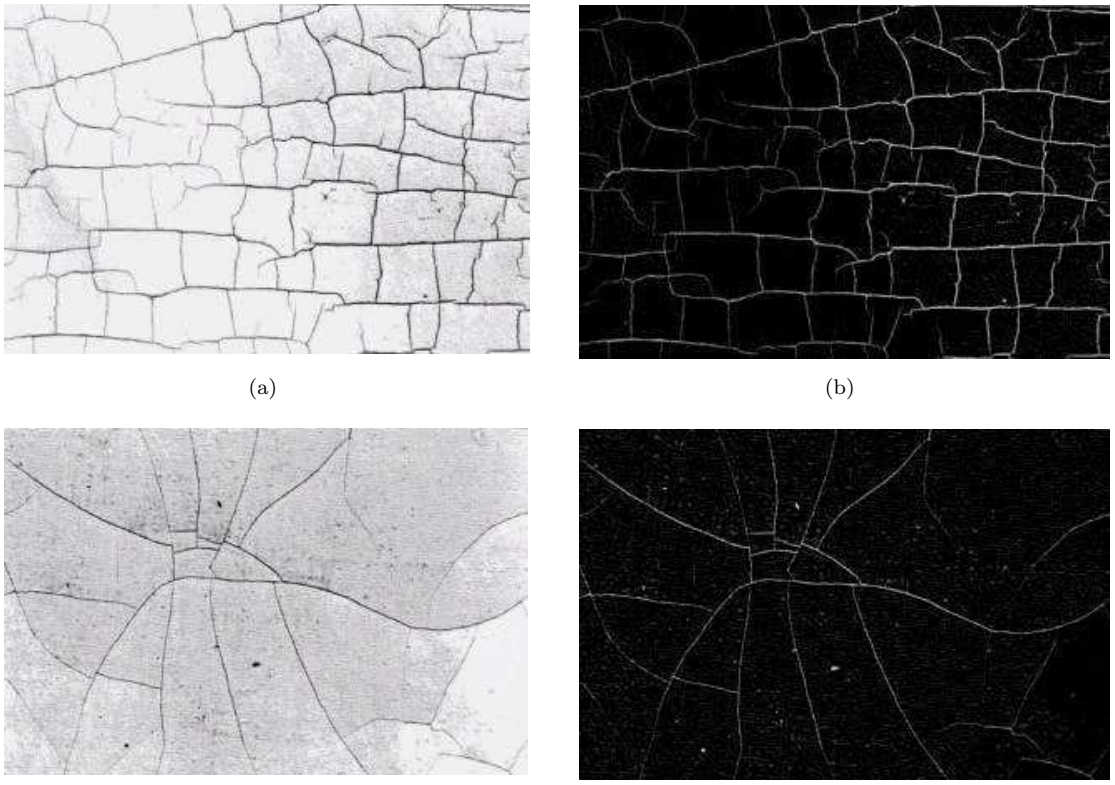


Figure 3.5: A 5x5 disk-shaped structuring element.



(a) (b)

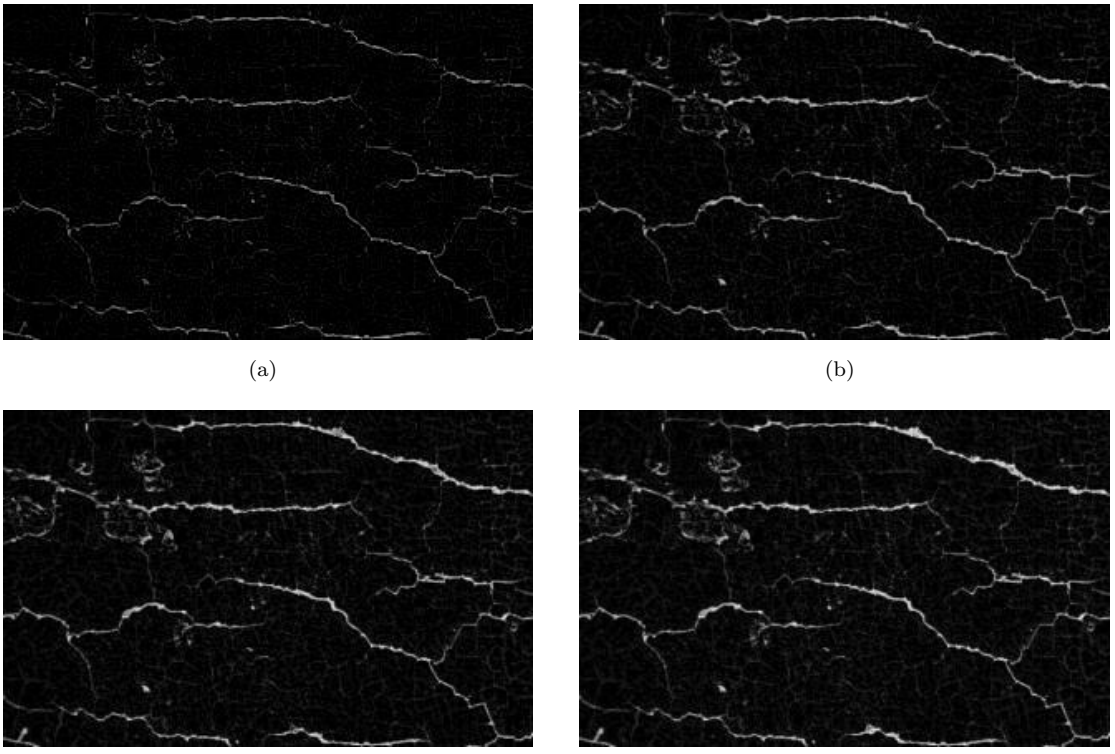
(c) (d)

Figure 3.6: Cracks enhanced by a close top-hat operator using a 5x5 disk-shaped structuring element.

The number of orientations used may vary according to the desired enhancement accuracy. However, the more filters (structuring elements) used, the higher the computational load is. Each structuring element accounts for a single execution of the top-hat operation on the image and computational load also depends on the size of the structuring element. Thus, to use four structuring elements, the workload will be at least four times more than using a single top-hat operation using a square-shaped structuring element. Computing the average value of each pixel adds up to the load.

For more accurate detection, the combined filter approach is seen as a good technique, since it captures lines better than when using square-shaped structuring elements, which also capture dark blobs of similar size, but still, for large images, this can be a very computationally expensive process.

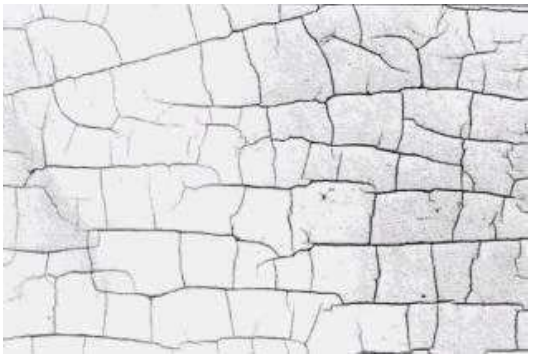
In order to effectively use a multi-orientation filtering approach, the angle difference between each subsequent structuring element must be low, in order to represent more crack directions. An angle resolution of $\pi/12$, for instance, will require twelve structuring elements, while an angle resolution of $\pi/2$ requires two structuring elements. In short, the computation of multi-orientation filtering becomes more expensive as the resolution gets



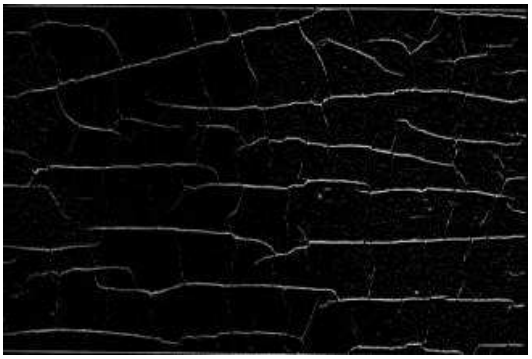
(a) (b)

(c) (d)

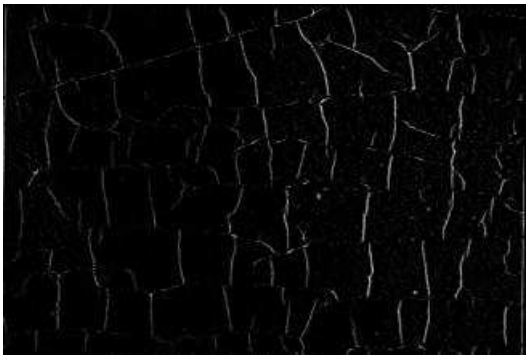
Figure 3.7: Cracks enhanced using disk-shaped structuring element of different sizes: a) 3x3; b) 5x5; c) 7x7; d) 9x9.



(a) Original image.



(b) Horizontally enhanced crack patterns.



(c) Vertically enhanced crack patterns.

Figure 3.8: Multi-orientation filtering.

finer.

The nature of the cracks itself in some sense makes multi-orientation filtering ineffective. Take for instance the images shown in Figure 3.9, where the crack patterns are somewhat circular or curved in nature. Figures 3.9(b) and (c) show the enhanced crack patterns in horizontal and vertical orientations respectively (angle resolution of $\pi/2$) using structuring elements with dimension 15x1. As can be seen the difference is not visually clear between Figures 3.9(b) and (c).

As opposed to this, multi-orientation filtering with similar structuring elements on a seemingly rectangularly-arranged crack pattern produces two different outputs. Figures 3.10(b) and (c) show the crack patterns of Figure 3.10(a) enhanced in horizontal and vertical directions respectively. When the two enhanced images are combined (Figure 3.10(d)) and compared with an image enhanced using a single 5x5 disk-shaped structuring element (Figure 3.10(e)), there is no clear difference, apart from the high level of noise in Figure 3.10(d), maybe because of over-enhancement.

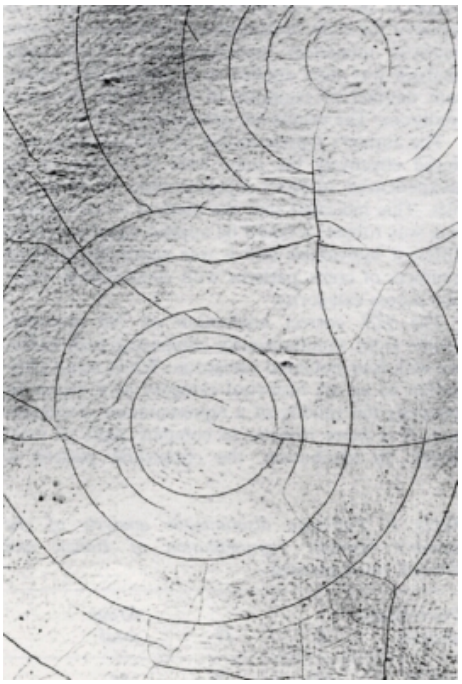
While the issue of choosing the correct structuring element type and size is still widely open, the question of which enhancement strategy to use is highly dependent on the level of accuracy it can offer and the computational load to endure. This issue will be discussed later in the chapter.

3.4 Automatic Thresholding

Enhanced images need to be segmented to separate the crack patterns from the background. This can be done in several ways, thresholding being one of the simplest and most widely used techniques.

The goal of thresholding is to segment an image into regions of interest, and to remove all other regions deemed inessential. The simplest thresholding methods use a single threshold in order to isolate objects-of-interest. In many cases however, especially in the case of crack images, no single threshold provides an excellent segmentation result over the entire image (as shown in Figure 3.2). In such cases, variable and multilevel threshold techniques, based on various statistical measures are used.

To assist this work, two most commonly used automatic threshold selection methods are considered, namely the Otsu technique and the *simple image statistic* technique.



(a)

(b)

(c)

Figure 3.9: A somewhat circular and curved crack pattern enhanced using multi-orientation structuring elements: a) the original image; b) enhancement in horizontal direction; c) enhancement in vertical direction.

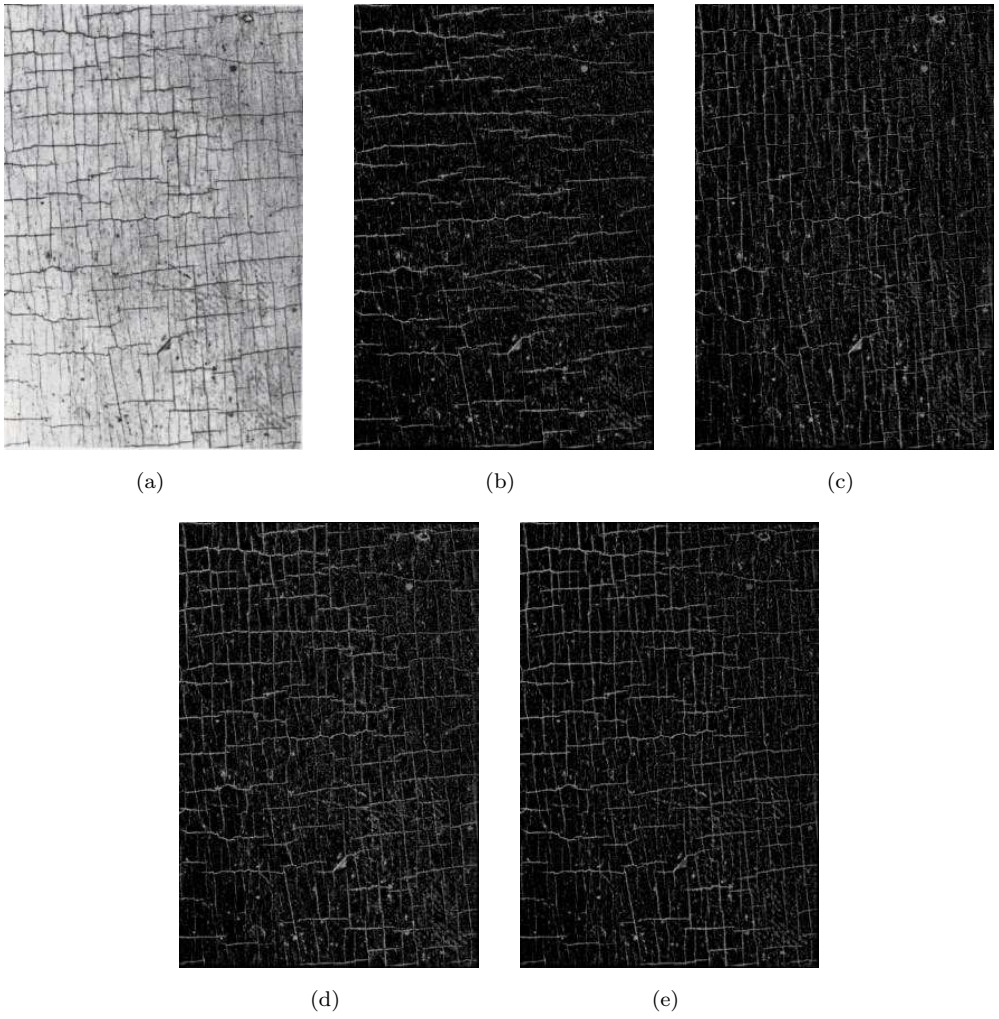


Figure 3.10: A rectangularly-arranged crack pattern: a) the original image; b) enhancement in horizontal direction; c) enhancement in vertical direction (d) (b) and (c) combined; e) enhancement using a 5x5 disk-shaped structuring element.

where n_i and n are the number of pixels with grey-level i and the total number of pixels in the image respectively. For more on the subject, see [94].

3.4.1 The Otsu Method

The Otsu method [95, 96] is based on discriminant analysis. The threshold operation is regarded as the partitioning of the grey-level distribution of an image into two classes $C_0 = \{0, 1..., t\}$ and $C_1 = \{t + 1, t + 2..., l\}$ at grey-level $t \in G = \{0, 1, ..., l\}$. It uses the histogram information derived from the source image. The optimal threshold, t_h, can be determined by maximising the criterion function:

$$t_h = \max_{t \in G} \left(\sigma_b^2\right) \tag{3.13}$$

with

$$\sigma_b^2 = \frac{\left(\left(\sum_{i=0}^{l} ip_i\right)\left(\sum_{i=0}^{t} p_i\right) - \sum_{i=0}^{t} ip_i\right)^2}{\left(\sum_{i=0}^{t} p_i\right)\left(1 - \sum_{i=0}^{t} p_i\right)} \quad \forall \quad t \in G \tag{3.14}$$

and

$$p_i = \frac{n_i}{n}. \tag{3.15}$$

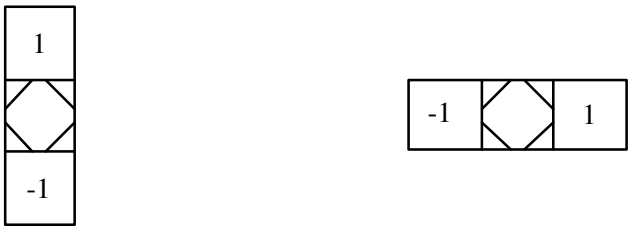
n_i and n are the number of pixels with grey-level i and the total number of pixels in the image respectively. For more on the subject, see [94].

3.4.2 The Simple Image Statistic Technique

The *simple image statistic* (SIS) algorithm [97] is an automatic bi-level threshold selection technique. Unlike the Otsu technique, SIS algorithm does not require computing a histogram for the image. Let $\mathbf{A} \in \mathbb{R}^{\mathbf{X}}$ be the source image over an m x n array $\mathbf{X}$. The SIS algorithm assumes that $\mathbf{A}$ is an imperfect representation of an object and its background. The ideal object is composed of pixels with grey-level a and the ideal background has grey-level b.

Let $\mathbf{e}(i,j)$ be the maximum on the absolute sense of the gradient masks $\mathbf{s}$ and $\mathbf{t}$ (see Figure 3.11) applied to $\mathbf{A}$ and centred at $(i,j) \in \mathbf{X}$. $\mathbf{a}(i,j)$ represents the source image $\mathbf{A}$. It is shown in [97] that

$$\frac{\sum_{i=1}^{n}\sum_{j=1}^{m} |\mathbf{a}(i,j) \cdot \mathbf{e}(i,j)|}{\sum_{i=1}^{n}\sum_{j=1}^{m} |\mathbf{e}(i,j)|} = \frac{a+b}{2}. \tag{3.16}$$



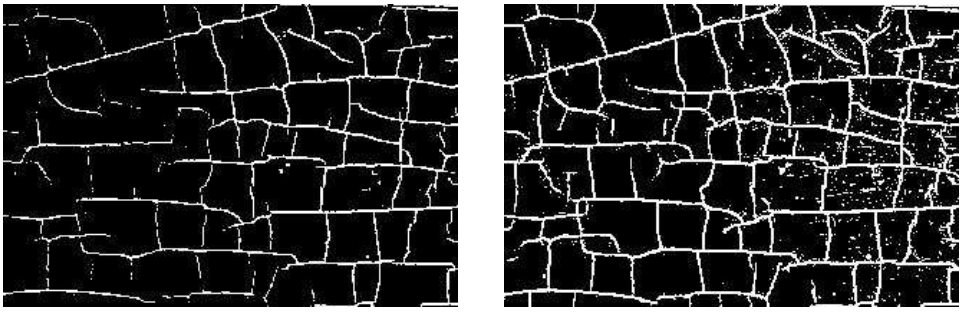
(a) s. (b) t.

Figure 3.11: Gradient masks $\mathbf{s}$ and $\mathbf{t}$ used in the SIS technique.

The fraction on the right-hand side of the equation is the midpoint between the grey-level of the object and background. Intuitively, the midpoint is an appealing level for thresholding. Thus, the threshold t_h set by the SIS algorithm is equal to the left-hand side of Equation 3.16.

3.4.3 Performance Evaluation

Experiments were conducted with several images to demonstrate the effectiveness of these algorithms. Figure 3.12 shows some results obtained for threshold values generated by the Otsu technique and the SIS algorithm. The enhanced image is thresholded using values from Otsu and SIS algorithm (48 and 30 respectively).



(a) (b)

Figure 3.12: Enhanced image segmented using automatically determined threshold values.

Based on observations, the Otsu technique is chosen due to the fact that its outcome is more consistent compared to the SIS technique in terms of segmented crack patterns. The Otsu technique produces threshold values higher than the SIS method; thus there is less noise in the output image. There is also a similar technique used for the detection of vehicle license plates as reported in [98].

3.5 Variable Thresholding

Specifically, in the case of large crack images, a single threshold value over the whole image is not a good strategy for segmenting the cracks. The main problem is due to uneven enhancement levels in suspected cracks as a result of inconsistencies in illumination. Although the top-hat operation to some extent is not affected by illumination inconsistency, in some cases it does. One simple strategy to overcome this limitation is to perform a technique known as variable thresholding [99]. Variable thresholding allows different threshold levels to be applied to different regions of an image.

The image is first divided into smaller pre-specified regions of similar size. In the current implementation, the image is divided into regions of even dimension, as shown in Figure 3.13, where G is the grid dimension. The image is first zero-padded, if its dimension is not a multiple of the grid size. The threshold value is then established for each region separately, using either the Otsu algorithm or the SIS algorithm, and thresholding is performed locally on every sub-image.

The exact methodology is as follows. Let $\mathbf{a} \in \mathbb{R}^{\mathbf{X}}$ be the source image, and let image $\mathbf{d} \in \mathbb{R}^{\mathbf{X}}$ denote the region threshold value associated with each point in $\mathbf{X}$, that is, $\mathbf{d}(\mathbf{x})$ is the threshold value associated with the region in which point $\mathbf{x}$ lies. The thresholded image $\mathbf{b} \in \{0,1\}^{X}$ is defined by

$$\mathbf{b}(\mathbf{x}) = \begin{cases} 1 & \text{if} \quad \mathbf{a}(\mathbf{x}) \geq \mathbf{d}(\mathbf{x}) \\ 0 & \text{if} \quad \mathbf{a}(\mathbf{x}) < \mathbf{d}(\mathbf{x}). \end{cases} \tag{3.17}$$

Instead of being globally thresholded, the enhanced crack image is locally processed; thus weak cracks are better detected. However, dimension of the grids must be chosen carefully. Very small grids will result in the emergence of unwanted noise in the thresholded image, since the assumption made prior to the process is that every grid should contain cracks (the object and the background). Even if a region in reality does not contain cracks, the algorithm will "force" cracks to appear, since the threshold will emerge as being extremely low in these regions.

3.6 Crack Thinning

In the later stages, it is more convenient and less time consuming to work with one-pixel wide cracks rather than those of variable width. Although width is seen as an important element in characterising a crack pattern [44], it is ignored for the time being to concentrate on other characteristics.

Crack patterns are then thinned using the *hit-or-miss* algorithm [92]. To make sure the cracks have been thinned to one-pixel wide, 10 iterations are performed. A *hit-or-miss* cleaning algorithm is then applied to remove isolated pixels. A "thinned" and "cleaned" image (see Figure 3.14) is the final outcome of the *crack detection* process, which will be the input to the next stage.

It is extremely difficult to achieve high accuracy in this detection stage. However, due to the large portion of cracks detected over the portion of noise, the outcome of the algorithm

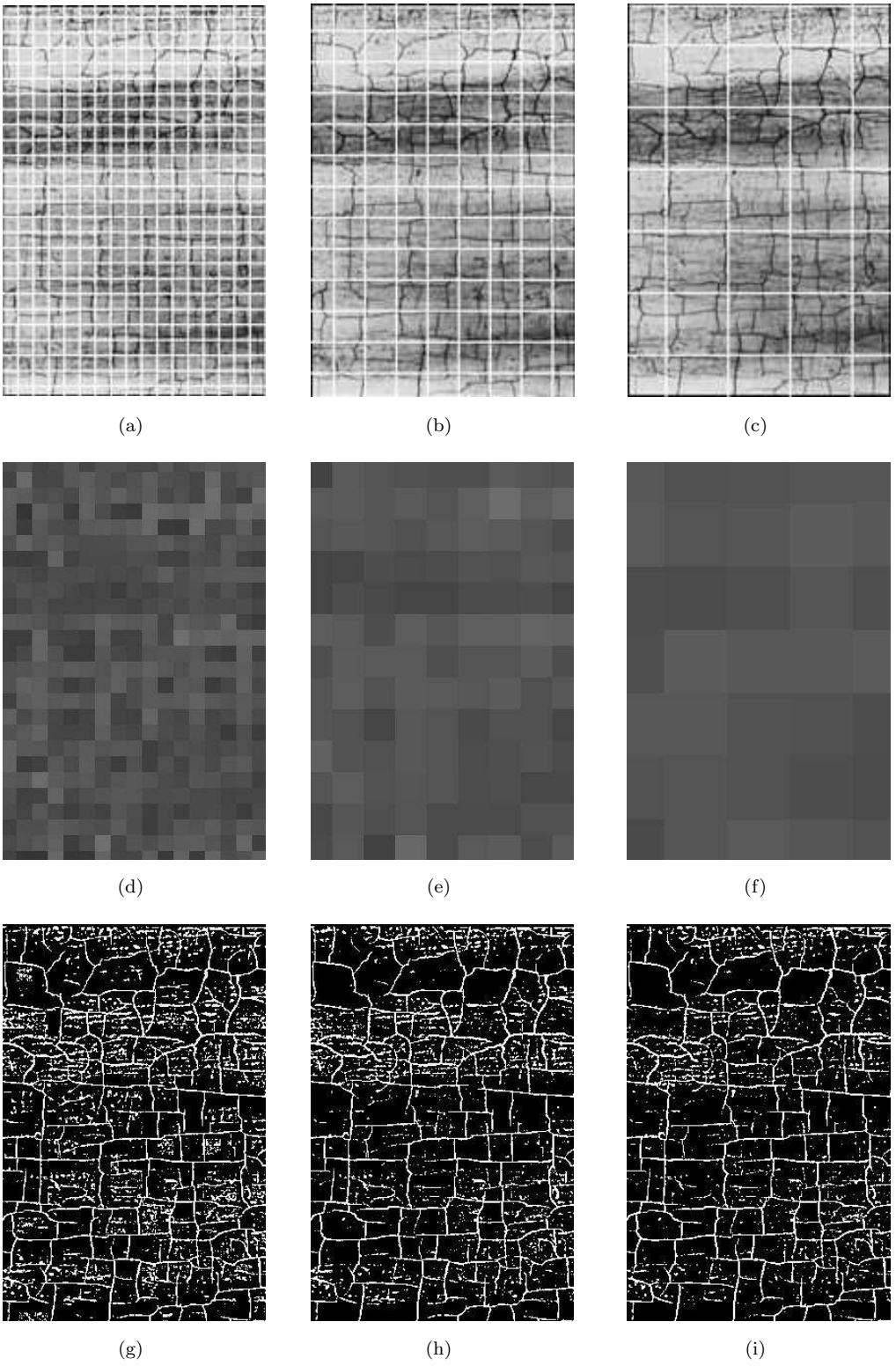


Figure 3.13: The first row showing grids overlaid on images ((a), (b) and (c)). The second row showing square regions of different threshold value displayed as grey values ((d), (e), (f)) while the third row displays the results of automatic thresholding on the images ((g), (h), (i)).

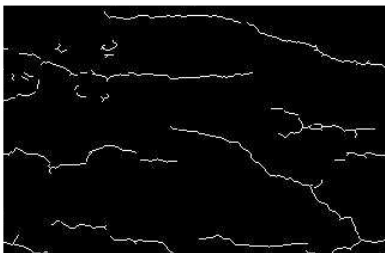


Figure 3.14: Thinned and cleaned cracks.

is considered acceptable and can be used for the later stages.

3.7 Results and Discussion

The algorithm has been tested on real crack images of various sizes. In additions to results shown earlier, other results are presented here.

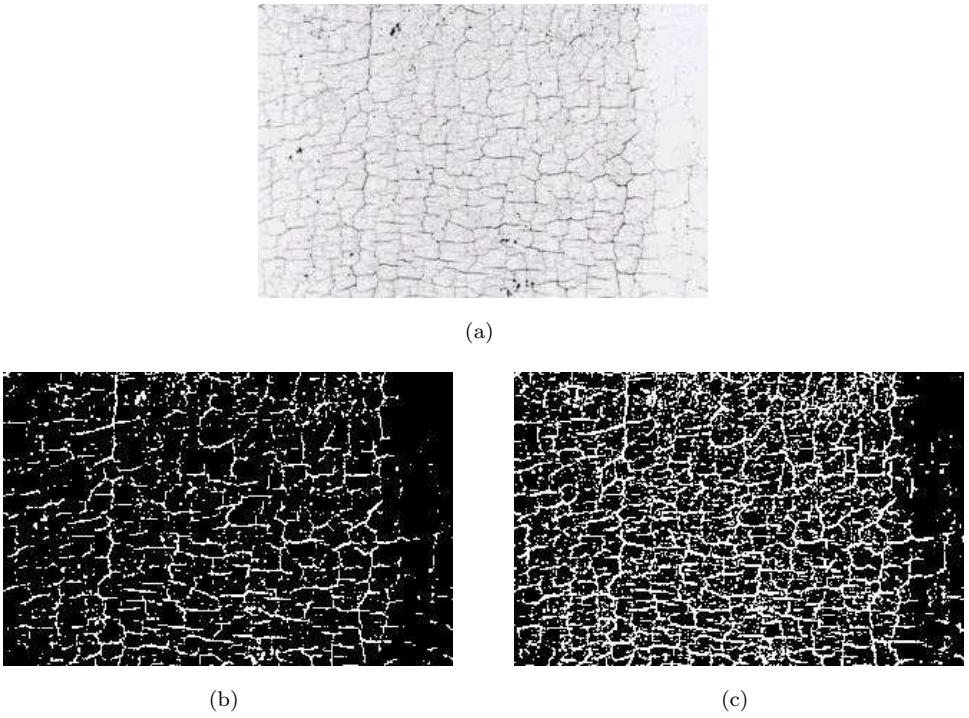
The questions still needing to be answered at this point are: 1) Should multi-orientation filtering be used or a single structuring element is sufficient to enhance the crack patterns? 2) Which shape and size of the structuring element should be used? 3) Which automatic thresholding algorithm is more suitable? 4) Should variable thresholding be used and, if it should, what should be the grid size? Difficulties occur in answering these questions on the basis that the process is entirely dependent on human observation (i.e. no strong analytical method).

Refering to Figure 3.8, it can be seen that multi-orientation filtering is effective when the crack patterns possess near-similar orientation properties as the structuring element. However, the vast variety of crack patterns makes this technique less practical, since, in order to represent more orientations, more structuring elements have to be used and this increases the computational load. To make matters more difficult, the enhancing ability of the multi-orientation filtering approach does not really overcome the problem of the single structuring filtering approach when applied on somewhat circular crack patterns (see Figure 3.9) and the same goes for rectangularly-arranged crack patterns (see Figure 3.10). As the crack patterns are not intended to be enhanced orientation-wise, the multi-orientation filtering approach offers no clear advantage over single structuring element filtering.

As for the shape size/dimension of the structuring element, the disk-shaped structuring element with a dimension of 5x5 is used solely based on continuous observations. Getting near-perfect accuracy is not a high priority, since the main target in the end is to classify

patterns and it is believed, as long as the stronger features of the crack patterns are detected, there will not be significant loss of information.

Going into the segmentation stage, the significant difference between the Otsu and the SIS technique is that the former consistently produces higher threshold values compared to the latter on the majority of occasions. Consequently, this raises concerns as to whether it is better to have a high threshold or a low threshold. In principle, high threshold might remove some information while low threshold can produce unwanted noise in the segmented image. Again the choice of technique is based on observations and Figures 3.15, 3.16 and 3.17 visualise some of the detected crack patterns using the Otsu and SIS techniques.



(a)

(b) (c)

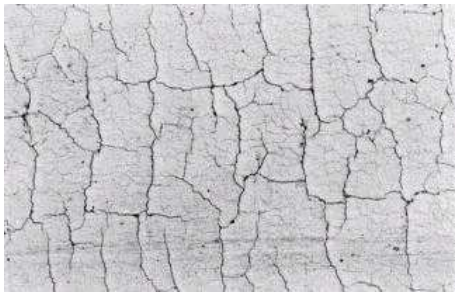
Figure 3.15: First sample comparison between crack detected using automatic threshold techniques: (a) original image; (b) Otsu technique; (c) SIS technique.

Inconsistent illumination and contrast can cause major difficulties in segmenting crack patterns if appropriate strategies are not taken as visualised in Figure 3.2. Four different approaches are compared and Figures 3.18, 3.19 and 3.20 visualise the comparisons for some selected images. The approaches are:

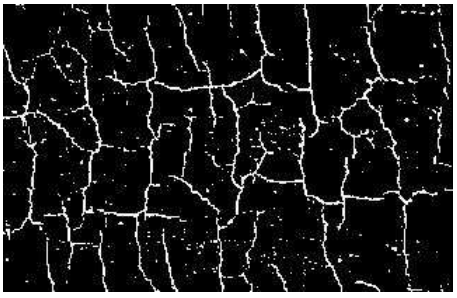
i. straightforward thresholding using Otsu technique,

ii. CTH operation followed by segmentation using the Otsu technique,

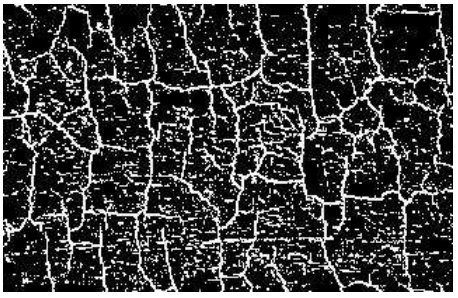
iii. variable thresholding using Otsu technique with grid size 64, and



(a)



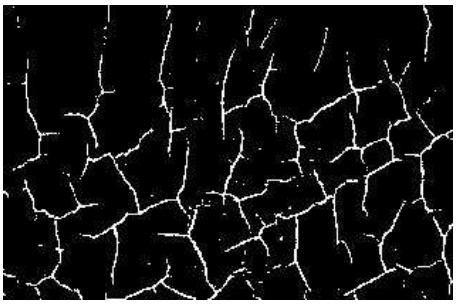
(b)



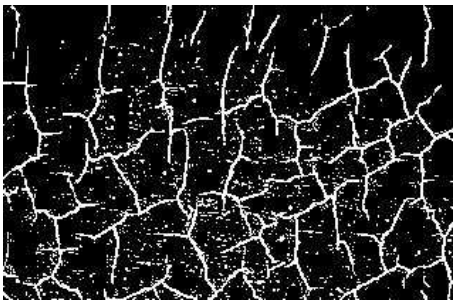
(c)

Figure 3.16: Second sample comparison between crack detected using automatic threshold techniques: (a) original image; (b) Otsu technique; (c) SIS technique.

(a)



(b)



(c)

Figure 3.17: Third sample comparison between crack detected using automatic threshold techniques: (a) original image; (b) Otsu technique; (c) SIS technique.

iv. CTH operation followed by a variable thresholding using Otsu technique with grid size 64.

Although in principle, thresholding alone in some cases can be used to segment crack patterns, it has been proven that the CTH operation contributes significantly in enhancing the suspected crack patterns, as evident in the sample results, with approaches (ii) and (iv) producing the better results. There is no doubt at this point that the CTH operation will be a vital component in the crack detection approach in this thesis. However, it is still necessary to determine how effective variable thresholding is.

The variable thresholding approach is seen as a good step in dealing with illumination inconsistency as demonstrated in Figure 3.21, using Otsu technique with grid size (G) 64. The dotted rectangle highlights the region which global thresholding failed to segment properly, but which was dealt with effectively by the variable thresholding approach.

However, variable thresholding can cause oversegmentation in areas where there are no cracks, especially if the grid size is too small. In this case, "cracks" will be forced to appear, as can be seen in Figure 3.22.

These circumstances raised a dilemma in terms of choosing between global thresholding and variable thresholding. The tradeoff between the two approaches is the amount of noise. Global thresholding works well for small images but in very large images, some features will be left out, while variable thresholding tends to introduce unwanted noise especially in areas not affected by cracks.

3.8 Summary

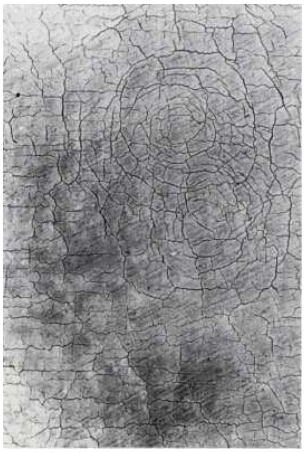
This chapter has shown how crack patterns can be detected using the morphological top-hat operation. It is a widely used technique for extracting ridge-valley structures. The importance of the technique in detecting cracks has been shown and the top-hat operation is considered as a vital component.

An automatic segmentation strategy has also been presented, using the Otsu technique, which has been demonstrated and proven to be useful in separating enhanced crack patterns. Several known obstacles in performing the task have been also outlined particularly those related to contrast and illumination inconsistency within a crack image. The issue regarding the size of the crack patterns has also been raised. This matter is not discussed in detail,

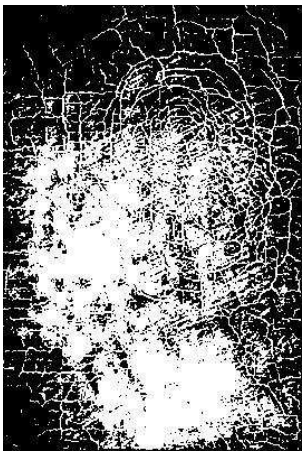
since it is less important. The main interest here is to capture the dominant features in a crack pattern and the approach undertaken is seen to be sufficient.

Multi-orientation filtering was also attempted to see the effect it has in enhancing crack patterns. Cracks are elongated structures which in a way match the shape of a rectangular (one-dimensional) structuring element. Experiments were conducted to demonstrate the effectiveness. The key point in multi-orientation filtering is angle resolution where in order to represent more angles, more structuring elements must be used. The drawback is higher computational load caused not only by a bigger structuring element size, but also by the extra number of structuring elements used. Furthermore, based on the experiments, multi-orientation filtering does not offer a significant advantage over the single structuring element approach.

Another critical point raised is the use of variable thresholding in segmenting the enhanced crack patterns. Oversegmentation occurs when variable thresholding is used particularly affecting areas without cracks. A portion of Chapter 4 will discuss a technique to further eliminate unwanted elements within the detected crack image.

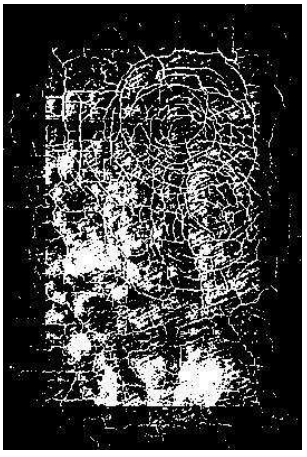


(a) Original image.



(b) Method (i).

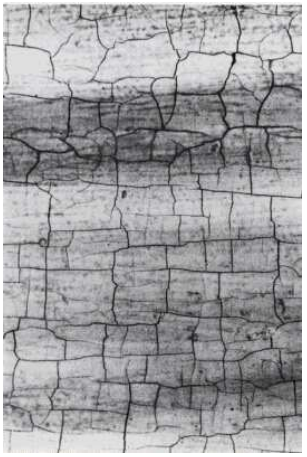
(c) Method (ii).



(d) Method (iii).

(e) Method (iv).

Figure 3.18: First example of crack patterns detected using four methods, namely (i) straightforward thresholding, (ii) CTH operation followed by thresholding, (iii) variable thresholding using grid size 64, and (iv) CTH operation followed by variable thresholding with grid size 64.



(a) Original image.

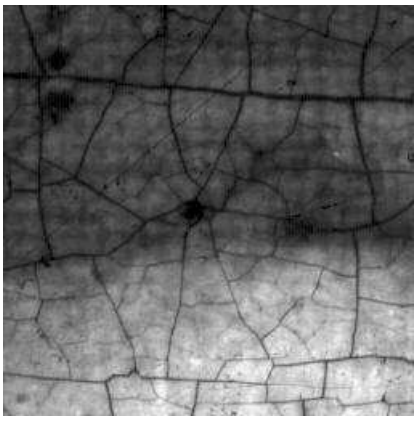
(b) Method (i).

(c) Method (ii).

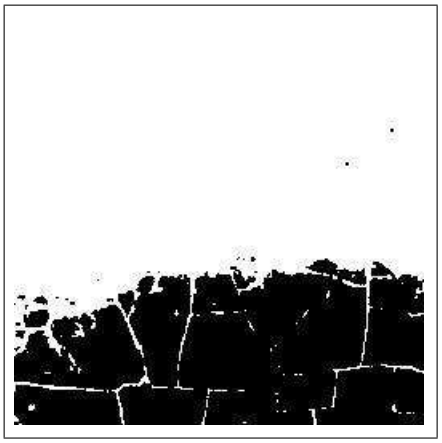
(d) Method (iii).

(e) Method (iv).

Figure 3.19: Second example of crack patterns detected using four methods, namely (i) straightforward thresholding, (ii) CTH operation followed by thresholding, (iii) variable thresholding using grid size 64, and (iv) CTH operation followed by variable thresholding with grid size 64.

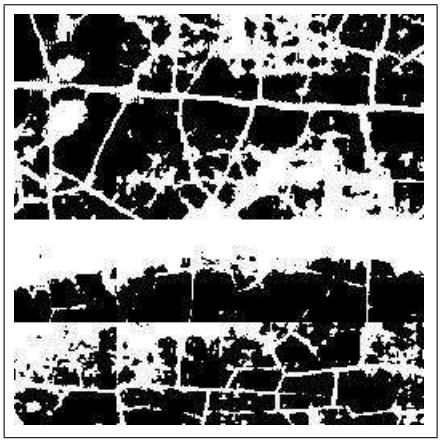


(a) Original image.

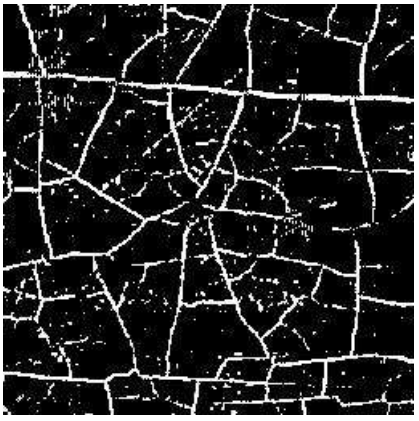


(b) Method (i).

(c) Method (ii).

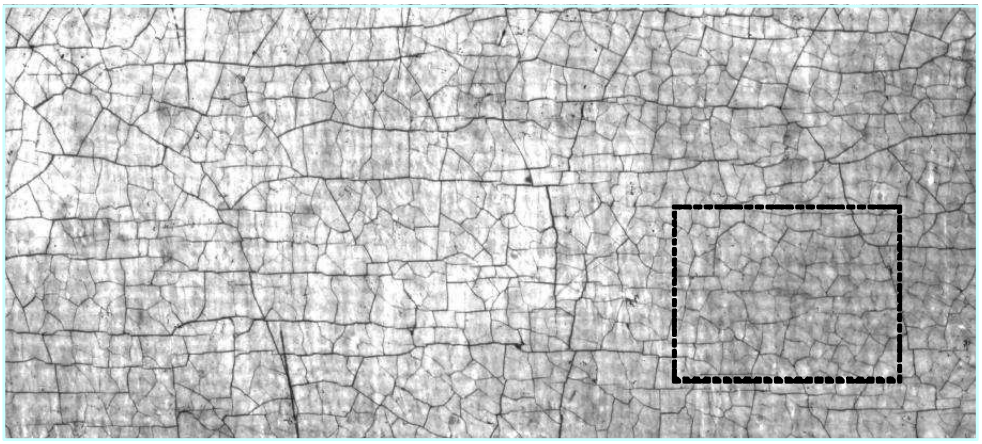


(d) Method (iii).

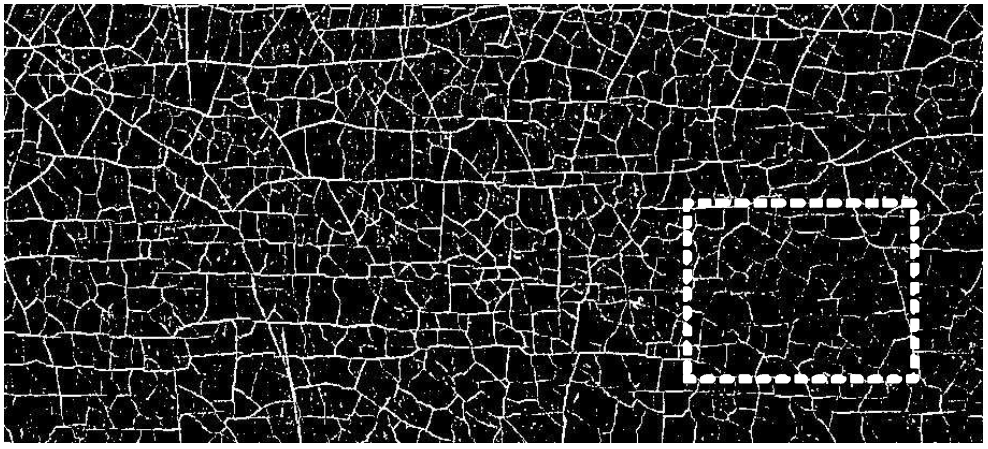


(e) Method (iv).

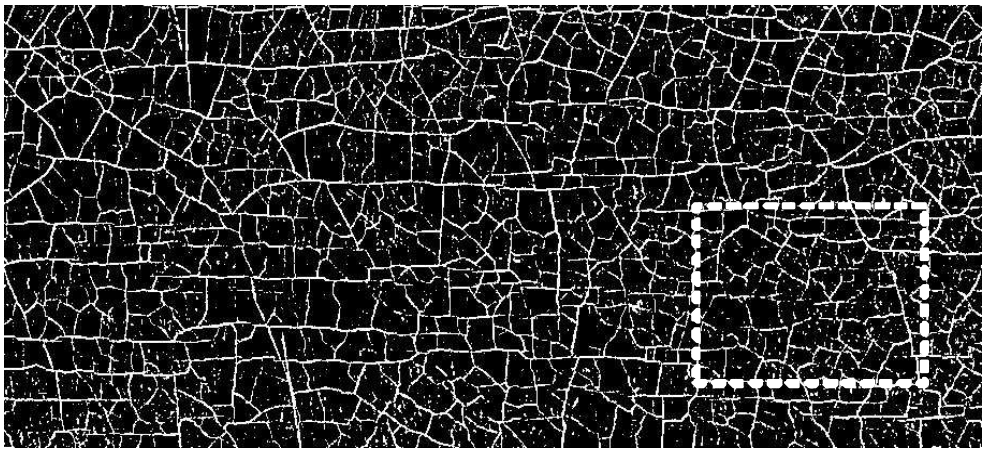
Figure 3.20: Third example of crack patterns detected using four methods, namely (i) straightforward thresholding, (ii) CTH operation followed by thresholding, (iii) variable thresholding using grid size 64, and (iv) CTH operation followed by variable thresholding with grid size 64.



(a) Original image.

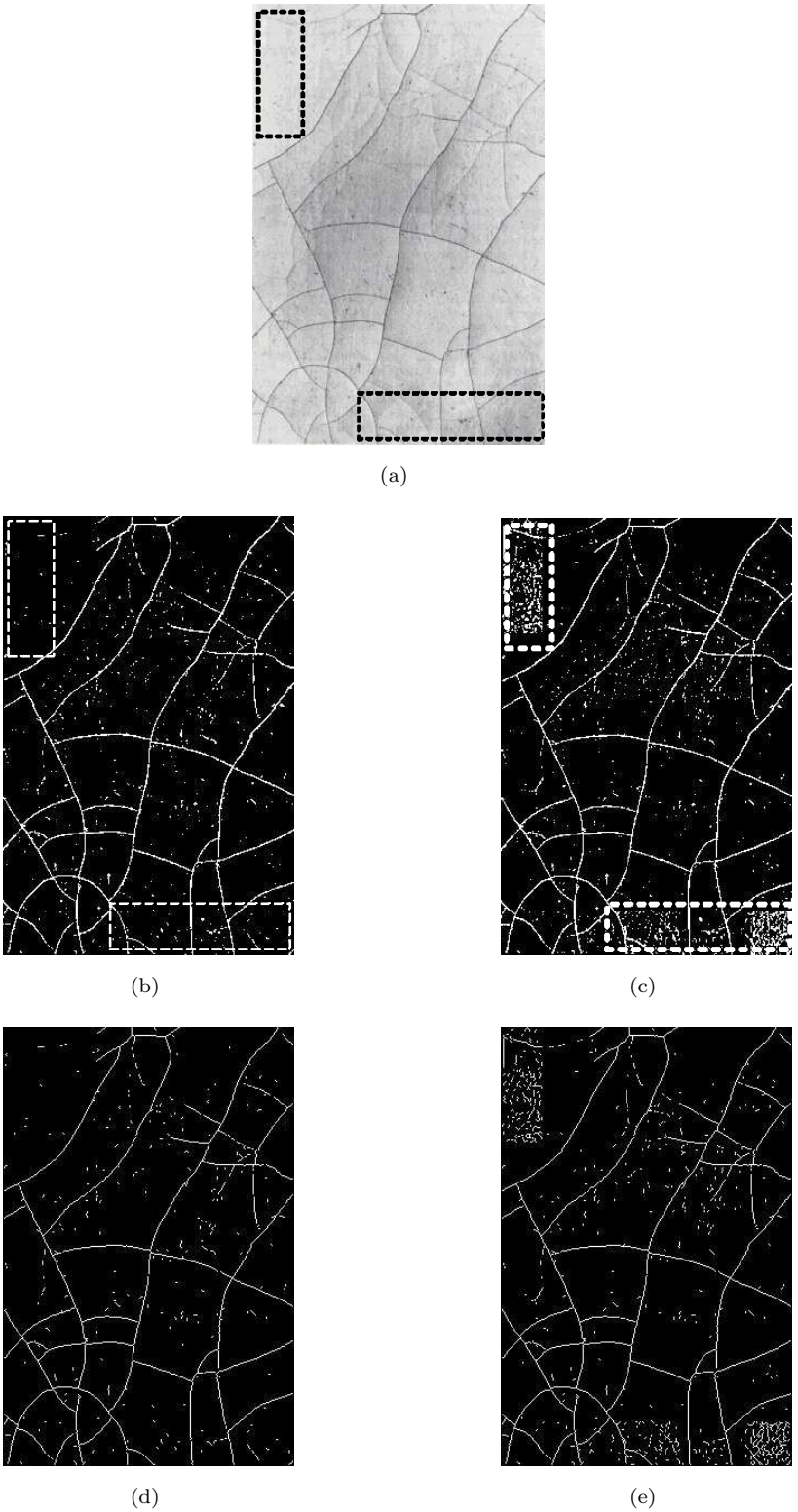


(b) Cracks segmented using manual thresholding.



(c) Cracks segmented using variable thresholding.

Figure 3.21: Dealing with illumination inconsistency using variable thresholding.



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

Figure 3.22: Figure showing the oversegmentation effect when "cracks" are forced to appear, as the variable thresholding technique assumes the existence of cracks in each square region. The rectangles highlight the regions affected by oversegmentation, clearly spotted in the image produced from approach (iv): (a) the original image; (b) cracks detected using approach (ii); (c) cracks detected using approach (iv); (d) the thinned version of (b); (e) the thinned version of (c).

Chapter 4

Craquelure Representation

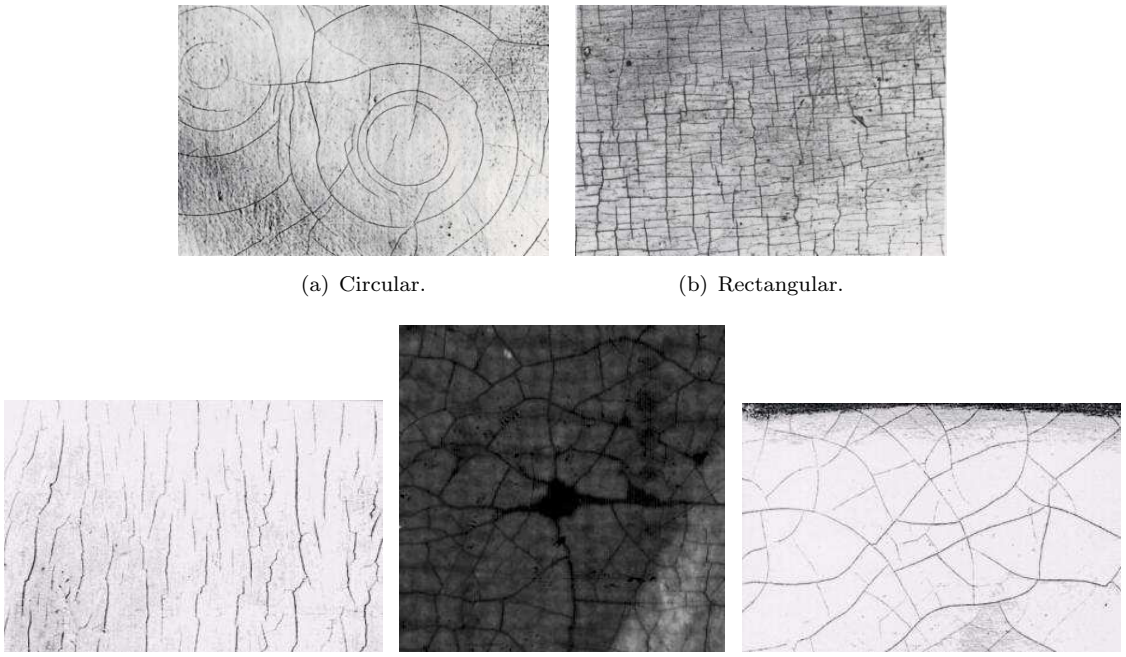
4.1 Introduction

Having segmented the crack contours from the background, the next stage involves representing the contours in a different form such that analysis can be made effectively. This chapter presents a stage-by-stage process of converting the crack contours from an image-based representation into a numerically structured representation, with the main aim to provide a platform from simple, effective and flexible data manipulation for use in the later stages. An approximation scheme using conservative shapes is also introduced to facilitate both feature extraction and area of interest determination. Finally, a crack pruning stage is presented which aims at removing noise and insignificant crack patterns.

4.2 Description of Crack Patterns

The main aim of this stage of the research is to find a way of describing crack patterns numerically. Prior to that, clear distinctions must be made between the classes mentioned in Section 1.4 generally stating the criteria and characteristics of each one. Figure 4.1 shows typical examples of the 5 crack classes.

Throughout most of the remaining chapters, the images in Figure 4.1 will be used as sample images to represent the five crack classes. Through observations based on the structural appearance of these images, it is clear that line direction is one of the most discriminating features. The crack patterns in Figures 4.1(a) and 4.1(e) can easily be distinguished from those in Figures 4.1(b), 4.1(c) and 4.1(d), where the shapes of the line segments are used as features. The line segments in Figures 4.1(a) and 4.1(e) are altogether more curvy than



(a) Circular. (b) Rectangular.

(c) Unidirectional. (d) Spiderweb. (e) Random.

Figure 4.1: Typical classes of cracks related to support structures and physical impact.

those in the other images. However, globally there is no clear distinction between Figures 4.1(b), 4.1(c) and 4.1(d) in terms of the structural shape of each of the line segments. Other features should be used to distinguish them.

Another feature that seems to be quite useful is the length of the line segments. The lengths of line segments in Figures 4.1(b), 4.1(c) and 4.1(d) are more consistent compared to those of Figures 4.1(a) and 4.1(e). From a statistical view point, the standard deviation of the line segment lengths might be a potentially beneficial measure to classify crack patterns.

Given the general observations of the differences, the next question is how to describe the cracks numerically.

Varley's, [38], model of cracks uses a variable number of Bézier curves [36]. The parameters of the Bézier curve model are sampled using the Reversible Markov Chain Monte Carlo (MCMC) technique [37, 100]. Varley explains in brief the methodology in which the parameters of the Bézier curve are used as features for classification. In total, twenty-eight features are extracted from the Bézier curve.

The fact that the MCMC technique is a global process makes it very inefficient in terms of its computational cost. Varley reported in [37], a total iteration of 10^7 to model a simple crack image, which is extremely time consuming. Being a global process makes it less suitable for a content-based analysis since processes are made on an image basis instead of

a regional or pixel basis.

A simple technique is used as the basis for analysis in this research, involving a chain-code based crack pattern structuring approach. It is a pixel-based process which allows analysis down to the pixel level. It also allows analysis at multiple structural levels of a crack pattern. The next section explains the approaches.

4.3 Structural Representation of Crack Patterns

A *crack following* algorithm is applied on a crack detected image, such as the one shown in Figure 3.14. Statistical data are collected as it "runs" along the lines. This feature extraction approach collates statistical information, while marking important points such as junctions and end points.

The Freeman chain-code [50, 51] has been used for various image processing applications, including finding features of curves/lines [52, 53]. A similar scheme is employed to record the direction of the crack pixels. The 8-connected Freeman chain-code uses a 3-bit code $0 \leq c \leq 7$ for each boundary point. The integer c indicates the direction in which the next crack pixel is located. The 8-connectivity scheme is as shown in Figure 4.2. Boundary chain-codes can be determined using contour following [101], which is a traversing process to identify the boundary of a binary object. The algorithm requires operations of $O(N)$. A conceptually similar method to that of [101] is employed, except that it is implemented on open boundary line segments instead of closed boundary objects.

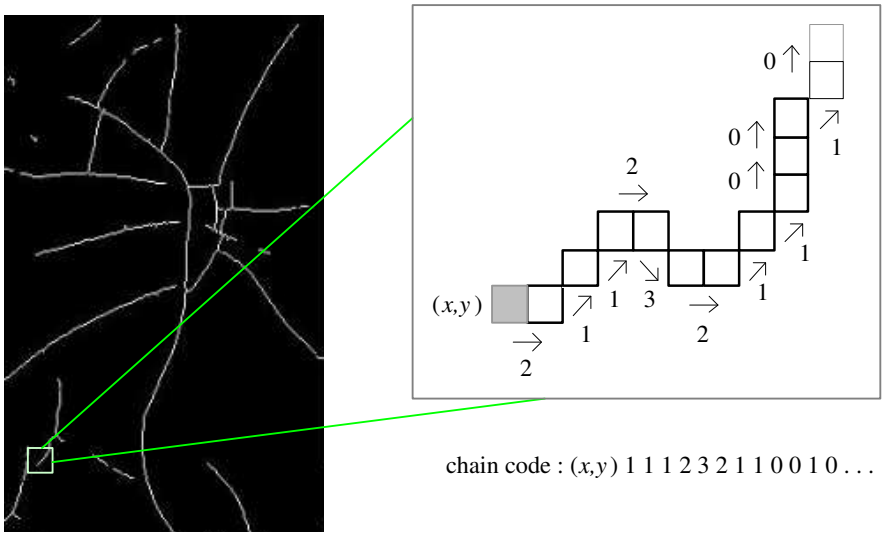


Figure 4.2: Example of an 8-connectivity chain-code of a small portion of crack pattern.

This approach serves two main purposes, namely post-detection filtering/pruning and high-level feature extraction. However, unlike most of the applications which implement chain-codes, the approach employed does not use the 8-connected direction as means of storing and reconstructing a crack structure. It is utilised for the sole purpose of building a structured representation of statistical data, to allow efficient data access and manipulation. With that, high-level feature extraction and crack pruning can be done easily. The extracted high-level features are the actual information stored as signatures describing the nature of an analysed crack pattern.

Figure 4.3 shows the first 250 chain-codes for the crack patterns in Figures 4.1 while Figure 4.4 shows the first 1000 chain-codes.

4.3.1 Hierarchical Structuring of a Crack-Network

In implementation, a pixel-based representation of crack patterns is converted into an alternative representation based on statistical measures and computational values. This should be a structured and organised representation that will allow easy manipulation and access to information at different resolution levels. Terminologies are defined beforehand, so that elements of a transformed crack pattern representation can be consistently recognised. The important terminologies are defined below.

- Edge point : A point at the edge of a line segment.
- Line segment : Collection of pixels connected in chain form between two edge points.
- Node : A point which connects 3 or more line segments.
- Crack-network : A collection of inter-connected line segments.
- Object-of-interest : A "meaningful" collection of crack-networks.

In graphical form, Figure 4.5 visualises the definition of the terminologies.

4.3.1.1 The Crack Following Routine

The *crack following* routine collates information on the basis that pixels are 8-connected. The process starts from a pixel $x = (x_1, x_2)$ in the image and a 3 by 3 neighbourhood $M(x)$ is defined by $M(x) = \{y : y = (x_1 \pm j, x_2 \pm k), j, k \in \{0, 1\}\}$. The initial pixel must be selected on the basis that it is the first with a single neighbour. At this instance, this

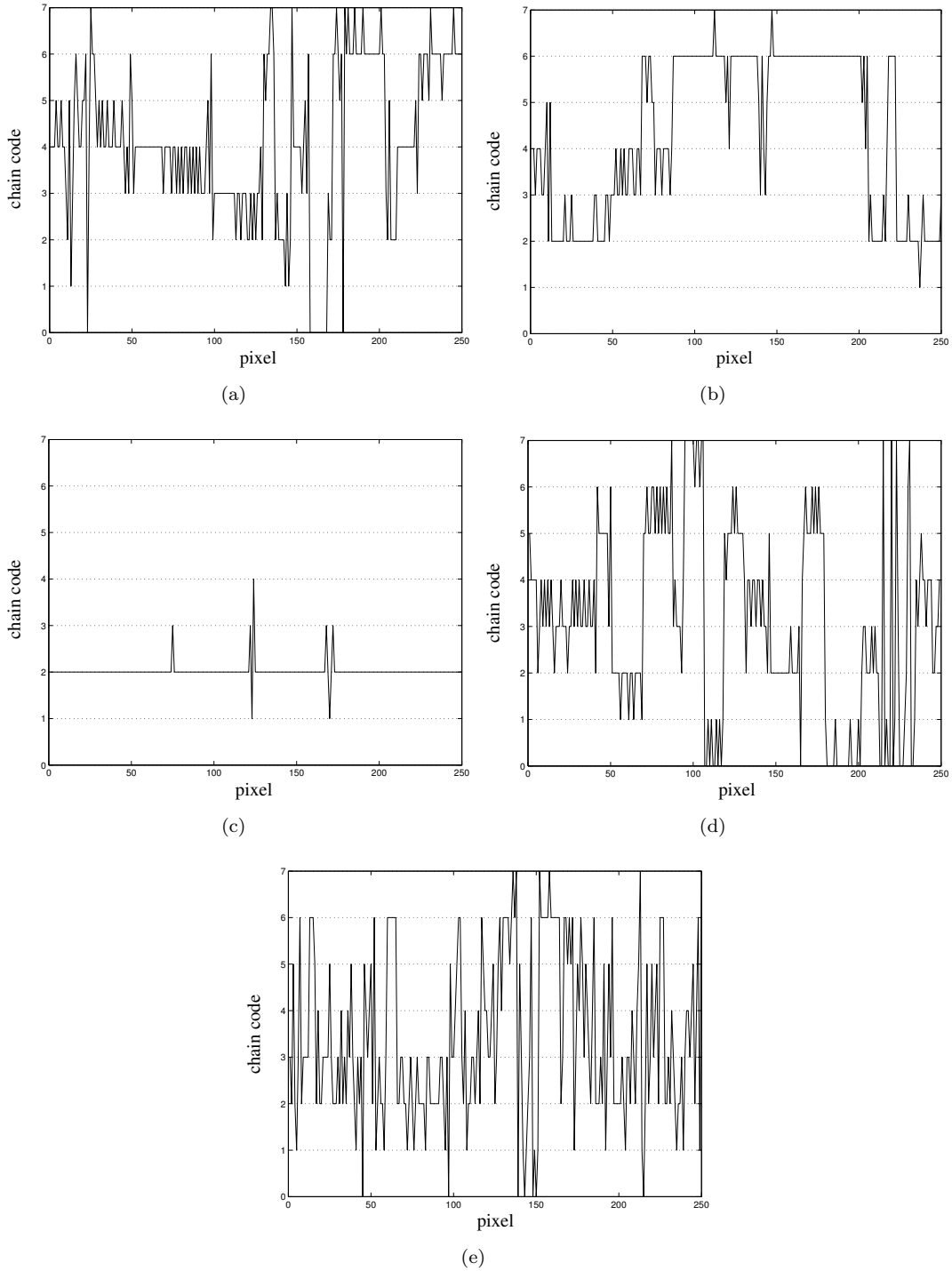


Figure 4.3: The first 250 chain-codes for the crack patterns of Figures 4.1(a), 4.1(b), 4.1(c), 4.1(d) and 4.1(e) respectively.

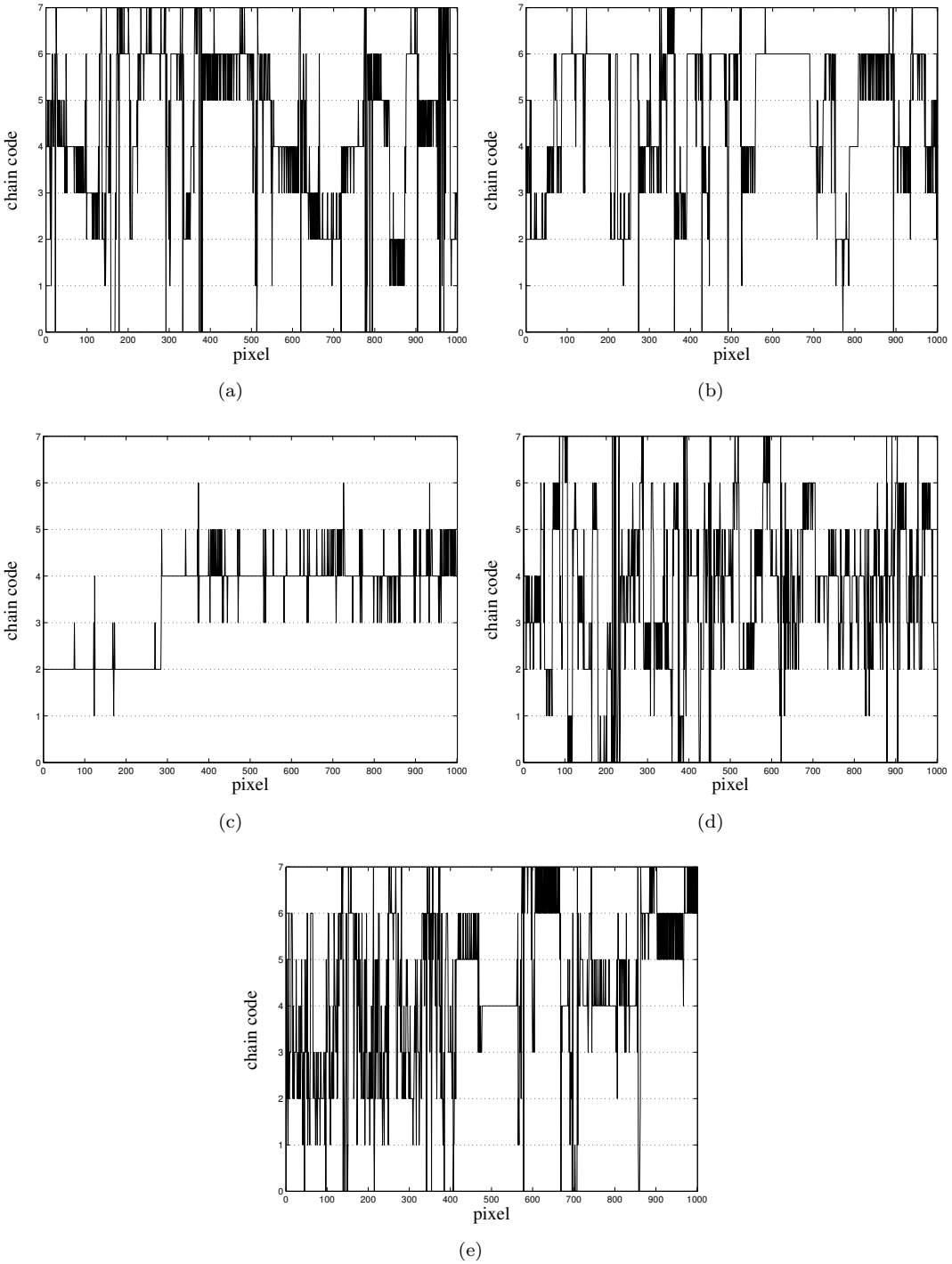


Figure 4.4: The first 1000 chain-codes for the crack patterns of Figures 4.1(a), 4.1(b), 4.1(c), 4.1(d) and 4.1(e) respectively.

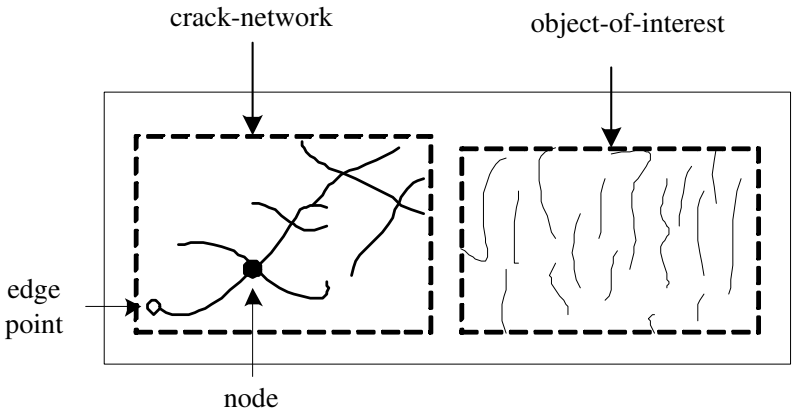


Figure 4.5: Terminologies related to crack pattern structuring defined graphically.

point is registered as a line segment and its starting point (x_1,x_2) is noted. If a point without a neigbour is encountered, that point will be registered both as a line segment and a crack-network. In a situation where the algorithm fails to find any single neighbour pixels, a random point is selected as a starting point.

Using $M(x)$ to monitor each neighbour for pixel x, the algorithm then searches for the location of the adjacent pixel. x is assigned to the adjacent pixel and the chain-code corresponding to the direction from the previous point to the current point is registered. All points that have been processed are labelled as *marked*. The *crack following* process continues until either one of the following events occur; a point without any *unmarked* neighbours is encountered or a point with more than one *unmarked* neighbour is encountered. In the former, the point is registered as an *edge point* for the respective line segment while for the latter a separate decision making procedure is implemented in order to decide whether the point should be assigned as a node or another point succeeding it.

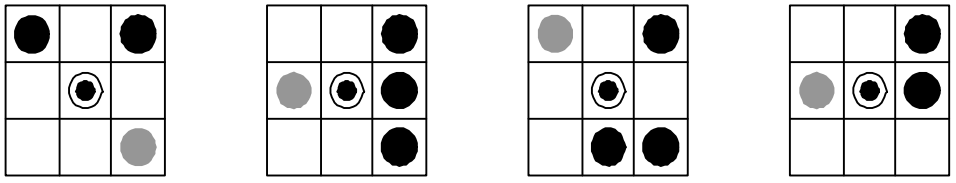
The decision making procedure is explained in Section 4.3.1.2. The node is registered as a substructure to the current data structure of the crack-network. Its location as well as the locations of its neighbours are copied into the data structure. In the next step, the algorithm selects one of the unmarked neighbours of the node as a start point and repeats the *crack following* procedure. The process continues until it fails to find pixels with at least one neighbour. At this point, a complete crack-network has been "followed". All the pixels belonging to the crack-network have been labelled as *marked* at this point.

The algorithm then performs another scan on the image to find a starting point for another crack-network. If it detects a point with a single neighbour, the same steps are repeated. The algorithm stops when there are no *unmarked* pixels left in the image.

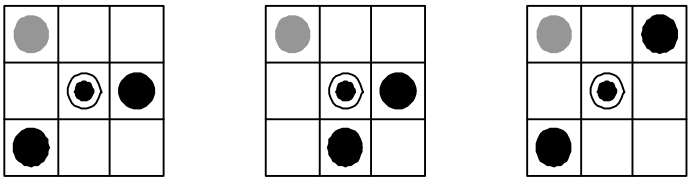
4.3.1.2 Node Determination

One of the challenging tasks within the *crack following* routine is to determine which pixel should be assigned as a node. The problem exists because of the nature of the process, which follows a route and determines feature points as it goes through the cracks. When the algorithm comes across a point with more than one neighbour, there are two possibilities: whether that point is a node or whether it is not. If it is not, then one of its neighbours should be. This section addresses a method to determine the exact node points of a crack-network.

The complication in deciding whether a pixel is a node or not depends on the arrangements of its neighbouring points. As an example, Figure 4.6 shows valid and indeterminate representations of nodes shown in a 3 by 3 neighbourhood.



(a) Valid. (b) Indeterminate. (c) Valid. (d) Valid.



(e) Valid. (f) Valid. (g) Valid.

Figure 4.6: Valid and indeterminate representations of nodes. The black coloured circles represent neighbours to the middle pixel which is located in the middle of the 3 by 3 masks, while *marked* pixels are shown as grey circles.

Out of all the examples, the status of the middle pixel of Figure 4.6(b) cannot be determined at this stage, although it has more than one neighbour whereas all the remaining middle pixels are considered valid nodes. Figure 4.6(b) can be a node only in a particular situation.

The next step determines whether the middle pixel of Figure 4.6(b) can be regarded as a node point. With the aid of a 5x5 neighbourhood mask, count the number of neighbouring crack pixels in the directions of the immediate neighbouring pixels (see Figure 4.7(a)). If all the sums equal two, assign the middle pixel as a node (Figure 4.7(b)). On the other hand, if all the sums equal 1, assign the centre immediate pixel of the middle pixel as a node as shown in Figure 4.7(a). If these two conditions are not satisfied, a further step is then employed to determine which pixel is the next pixel out of all the neighbours. In solving

this, two rules are defined, which takes into account the previous and potential paths taken by the *crack following* algorithm. The rules are as listed below according to the highest priority.

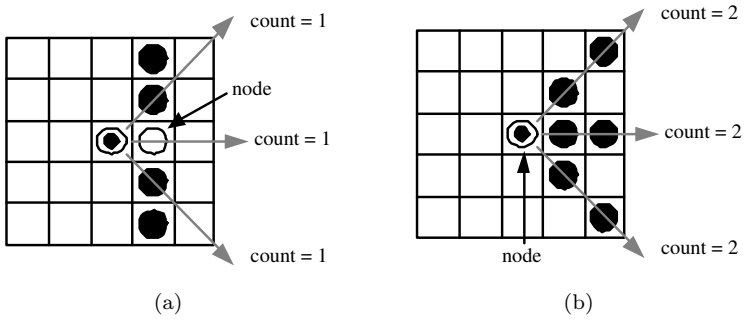


Figure 4.7: Conditions for assignment of node points in a 5x5 neighbourhood: (a) condition where count=1 in all directions and the centre immediate pixel is assigned as the node; (b) condition where count=2 and the middle pixel is assigned as the node point.

Rule A : With the aid of a 5x5 mask and with the middle pixel being the current pixel under consideration, count the number of neighbouring crack pixels in the directions of the immediate neighbouring pixels (see Figure 4.8). If there is a single maximum count, choose the corresponding path as the next path or in other words, pixel P as the next pixel. If no single maximum count exists, then Rule B applies.

Rule B : Choose the current path as the path taken for the next pixel (see Figure 4.9).

If the above procedures are not followed, a slight deviation to the actual location of the node will occur (one pixel deviation). Besides that, for a certain node point, there will also exist several "dummy" node points. Although the exact locations of the nodes are not really crucial, the number of assigned nodes can be an important cue for feature computation related to the spatial distribution of node points.

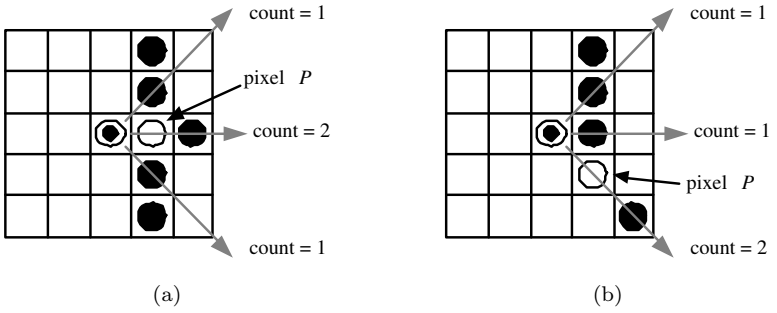


Figure 4.8: Two samples of a condition that satisfy Rule A. The black coloured circles represent neighbouring pixels, the black and white coloured circles represent middle pixels. The newly assigned nodes are shown by the black arrows.

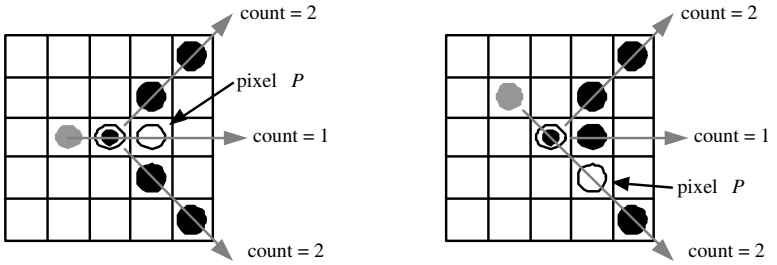


Figure 4.9: Two samples of conditions that do not satisfy Rule A but satisfy Rule B. The black coloured circles represent neighbouring pixels, the black and white coloured circles represent middle pixels and the grey circles represent previous middle pixels. The newly assigned nodes are shown by the black arrows.

4.3.1.3 Hierarchical Data Structuring of Important Features

The output of the *crack following* algorithm is structured data concerning all the existing crack-networks in the image called the *network tree*. Each crack-network has its own substructures, which hold information related to nodes and line segments. The number of substructures depends on how many of these are detected. The locations of the nodes are stored. As for the line segments, statistical information such as the total number of pixels, total number of nodes and the chain-code histogram are generated. The statistical data corresponding to each line segment is then collated to produce a representation for a crack-network. Thus, each crack-network contains details about the total number of pixels, number of nodes, total crack length (see Section 6.2.1), chain-code histogram (see Section 6.2.1) and details about its components (i.e. nodes and line segments).

With reference to Figure 4.10, the structured crack patterns are divided into two parts: crack-network features and local features. Simply defined, the former refers to statistical values which represent a single crack-network, while the later numerically represents local entities such as nodes and line segments.

This pre-supposes a definition of crack pattern, which may be regarded as a single crack-network or as cracks bounded in a particular image or region, and this in CBIR terms becomes the issue of object-of-interest.

4.4 Crack Pattern Approximation

Some features are left undetected in the craquelure detection stage, and this tends to produce disconnected curves. In order to extract high-level features for content-based

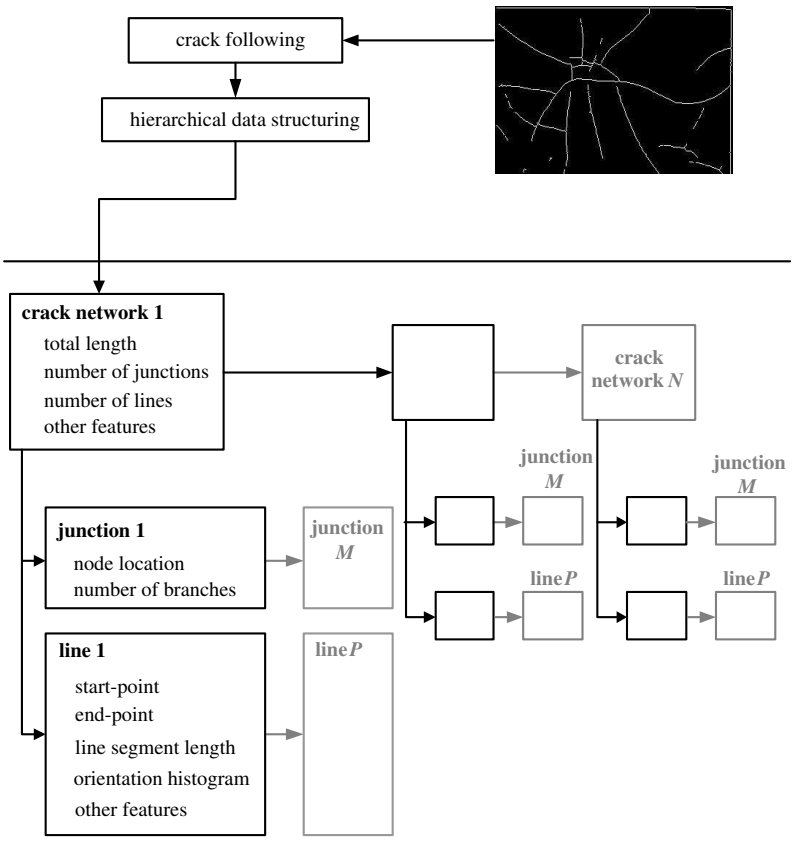


Figure 4.10: The *network tree*, a hierarchically structured data concerning crack-networks.

application, these supposedly connected curves have to be grouped together. It is one of the many steps needed to produce a content-based platform for digital analysis of crack patterns in paintings, particularly for classification purpose. The prime objective of the grouping algorithm is to segment or partition areas of an image to produce reliable representations of object-of-interest. The first stage of the algorithm utilises the minimum bounding rectangle (MBR) of a crack-network as a means to deciding on merging, using a proximity rule. The use of both the rotated and the un-rotated MBR are demonstrated. In the second stage, characteristics represented by the rotated MBR are used as features for an N-dimensional clustering.

4.5 Object Interpretation in Craquelure Analysis

Crack patterns can be interpreted in several ways. From a content-based view-point, the question to be asked is how to define a single crack pattern. Based on Varley's work [38], classification is made based on an image-to-image basis. In other words, all crack patterns in a single image are assumed to belong to the same crack class. This is fully understood,

since Varley's work is not at all concerned with content-based issues.

This undefined interpretation of the object-of-interest is an important matter. Real application of content-based crack analysis needs processes on very large images with hundreds (even thousands) of crack-networks and thus the interpretation of object-of-interest must be addressed.

For object-of-interest extraction, processing and storing the definition is seen as an important element. For instance, a single crack-network can be acknowledged to represent an object-of-interest. However, this assumption might be less reasonable in a shorter crack-network. In another case, an *a priori* determined square grid can be assumed to contain a single object-of-interest, but yet again this assumption is not particularly valid since cracks are thin structures, and they tend to "overflow" into neighbouring grids. Furthermore, patterns of cracks are dependent on the way they are viewed. The approach explained in Section 4.3 and visualised in Figure 4.10 processes cracks assuming that each pair of adjacent pixels belong to the same entity, which is a crack-network. Post-processing must be done to further elaborate the definition of object-of-interest using more complex criteria.

4.6 Conservative Shape Approximation

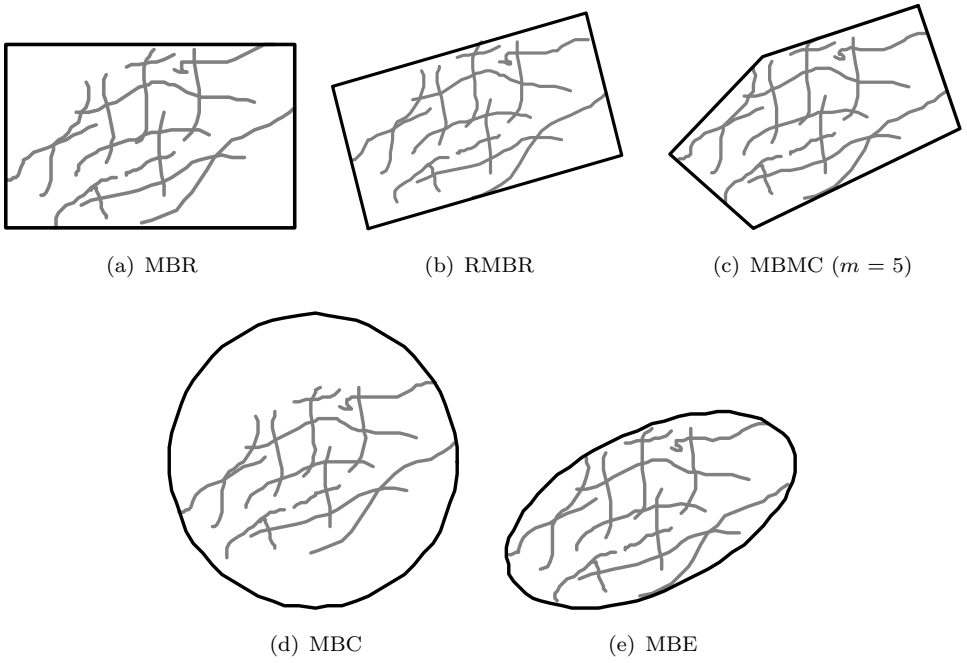
If reliable image segmentation is available, a popular approach to object classification is based on analysing the boundaries of the extracted regions, which offers two main benefits. Firstly, it allows simple and efficient computation of descriptors, and secondly it offers a wide choice of techniques for classification based on a vector of properties [102]. Another approach for shape representation is to define a set of standard shapes such as rectangles, circles or ellipses against which input regions are compared. These representations are known as *conservative approximations* [103].

An approximation is considered conservative if, and only if, each point inside the contour of the original object is also in the *conservative approximation*. Other known conservative representations besides the MBR and the RMBR, are the *convex hull* (CH), the *minimum bounding m-corner* (MBMC), the *minimum bounding circle* (MBC) and the *minimum bounding ellipse* (MBE) [103, 49]. These approximations differ in terms of their accuracy, the area they cover and the number of required parameters. Figure 4.11 visualises these conservative shape approximations while Table 4.1 compares them in terms of the number of required parameters. The CH has on average the highest storage requirement and the highest accuracy, while the MBC needs the least space for parameter storage [104].

This approach allows a more general characterisation of descriptors, since detailed information about a shape has been translated into a more simplified representation. Despite a reduction in shape information, it serves well in high-volume applications, where the spatial objects display a very complex structure. Computation of spatial operators is very time-intensive, and therefore a simplified shape representation will allow faster computation.

conservative approx.	MBR	RMBR	CH
no. of parameters	4	5	var.
conservative approx.	MBMC	MBC	MBE
no. of parameters	$2m$	3	5

Table 4.1: Number of parameters for *conservative approximation*.



(a) MBR (b) RMBR (c) MBMC ($m = 5$)

(d) MBC (e) MBE

Figure 4.11: Sample *conservative approximations* of a crack-network using the *minimum bounding rectangle* (MBR), *rotated minimum bounding rectangle* (RMBR), *minimum bounding m-corner* (MBMC), *minimum bounding circle* (MBC) and *minimum bounding ellipse* (MBE) respectively.

"Bounding boxes" is a term commonly used in the literature. Although the word box can be used in a two-dimensional format, any possible suggestion of three dimensions is removed by calling them "bounding rectangles".

Bounding rectangles are most commonly used in computer graphics to improve the performance of algorithms, which should process only intersecting objects. Due to their simpler shape, checking intersections among bounding rectangles is almost always more efficient

than among complex objects. Thus, bounding rectangles allow an algorithm to quickly perform a task and avoid costly processing in unnecessary cases. Traditionally, computer programs have dealt with on-screen objects such as images and characters by placing them in an invisible rectangle, which appears whenever it is clicked. Word processing tools, like *Microsoft Word*^{TM} for instance, allows images bounded by rectangles. Another fine example is the bounding box information stored with *encapsulated postscript* (EPS) files, where a larger document that embeds these files can place and compose them properly.

The heuristic of bounding rectangles is used in rendering algorithms, including traditional visible-surface determination [105] as well as image-based techniques to reconstruct new images from the re-projected pixels of reference images [106]. Bounding rectangle heuristics is also commonly used in algorithms for modelling, such as techniques that define complex shapes as Boolean combinations of simpler shapes [107] and techniques to verify the clearance of parts in an assembly [108]. Bounding rectangles are also useful in animation algorithms, especially in collision detection algorithms for path planning [109, 110] and the simulation of physically based motion [111, 112]. Bounding rectangles have also been extensively used in spatial database systems [108, 113].

Another significant motivating factor in the use of bounding rectangles is the increase in computational speed. Arrebola et al. [114] use a coarse shape representation, such as bounding rectangles of objects in a scene in order to minimise computational load and time requirements in a foveal active vision system. Meier and Ade [115], use bounding rectangles to track objects such as cars, in an image sequence in the development of an automatic traffic scene analysis system that would avoid collisions.

Bounding rectangles are also extensively employed in document analysis systems. Chen and Bloomberg [116] use bounding rectangles to highlight imaged documents where bounding rectangles of words, sentences and paragraphs are extracted. Wu et al. [77], describe an algorithm that detects text strings in an image and puts bounding rectangles around them for further processing.

In other applications, Chang and Lee [117] extract a frame-difference sequence from video and subsequently segment the video content via a box segmentation mechanism. Using characteristics and prominent points that accompany the bounding rectangles, they perform video content indexing. In 1-D signal analysis, bounding rectangles are used to speed up speech recognition, as described in [118]. Paquet et al. [119] use bounding rectangles as a means for describing 2-D objects, using coarse description of what belongs to the bounding rectangles. The representation is invariant to resolution, translation and rotation and targeted for use in the MPEG-7 [120] standard description of audio-visual (AV) data.

4.6.1 The Minimum Bounding Rectangle (MBR)

Computation of the MBR is simple and straightforward as shown by Freeman et al. [121]. It begins by enclosing an object (crack-network) in a rectangle with sides parallel to the y and x axes of a cartesian coordinate system. The crack-network is represented in chain form, $C = c_0c_1c_2...c_n$ where c_j are octal-valued chain links computed over $j = 0, 1..., n$. h_{min} and h_{max} are the minimum and maximum pixel coordinates of the object along the y-axis while w_{min} and w_{max} are the minimum and maximum pixel coordinates of the object along the x-axis. The MBR is constructed by lines $y = h_{min}$, $y = h_{max}$, $x = w_{min}$ and $x = w_{max}$ which satisfy $h_{min} \leq y \leq h_{max}$ and $w_{min} \leq x \leq w_{max}$.

4.6.2 The Rotated Minimum Bounding Rectangle (RMBR)

The computation of RMBR benefits from the concept of moments [64]. A moment of order $(p + q)$, μ_{pq} is as computed in Equation 1.6. A parameter which is crucially important in this work is the direction or *axis of minimum inertia*, θ [64] computed as

$$\theta = \frac{1}{2}\tan^{-1}\left(\frac{2\mu_{11}}{\mu_{20} - \mu_{02}}\right) \tag{4.1}$$

where $0 \leq |\theta| \leq \frac{\pi}{2}$. The *axis of minimum inertia* is a property which makes more sense for elongated objects. However, it is a prerequisite for computing the RMBR. Let L be the number of non-zero pixels in the crack-network, $y(l)$ and $x(l)$ are coordinates of y and x for $0 \leq l \leq L$. Once θ is known, the following transformations are used [122]:

$$\alpha = x(l)\cos\theta + y(l)\sin\theta, \tag{4.2}$$

$$\beta = -x(l)\sin\theta + y(l)\cos\theta. \tag{4.3}$$

The maximum and minimum of α and β are then computed to reveal α_{min}, α_{max}, β_{min} and β_{max}. The sides of the RMBR (i.e the height and the width) h_R and w_R can be calculated as

$$h_R = \beta_{max} - \beta_{min}, \tag{4.4}$$

$$w_R = \alpha_{max} - \alpha_{min}. \tag{4.5}$$

In order to reconstruct the RMBR, the following information is needed: a centre point (centroid), an *axis of minimum inertia* θ, RMBR width w_R and RMBR height h_R.

The centre of the RMBR is computed using some geometrical properties, first taking into account the corresponding x and y coordinates of α_{min}, α_{max}, β_{min} and β_{max}. Knowing the *axis of minimum inertia* and the axis perpendicular to it, $\theta+\pi/2$, the coordinates are then used as points to construct four straight lines, A_1, A_2, A_3 and A_4 corresponding to the sides of the RMBR. Solving simultaneous equations, $A_1 = A_2$, $A_2 = A_3$, $A_3 = A_4$ and $A_1 = A_4$, the corner points of the RMBR, I_1, I_2, I_3 and I_4 can be detected. From I_2 and I_4, the straight line B_1 can be obtained and the same applies for I_1 and I_3, which produce B_2. From this point, finding the centre of RMBR is a straightforward process. In doing so, the simultaneous equation, $B_1 = B_2$ is solved to reveal the centre points, (x_c, y_c). Using x_c, y_c, h_R, w_R and θ the RMBR can be reconstructed.

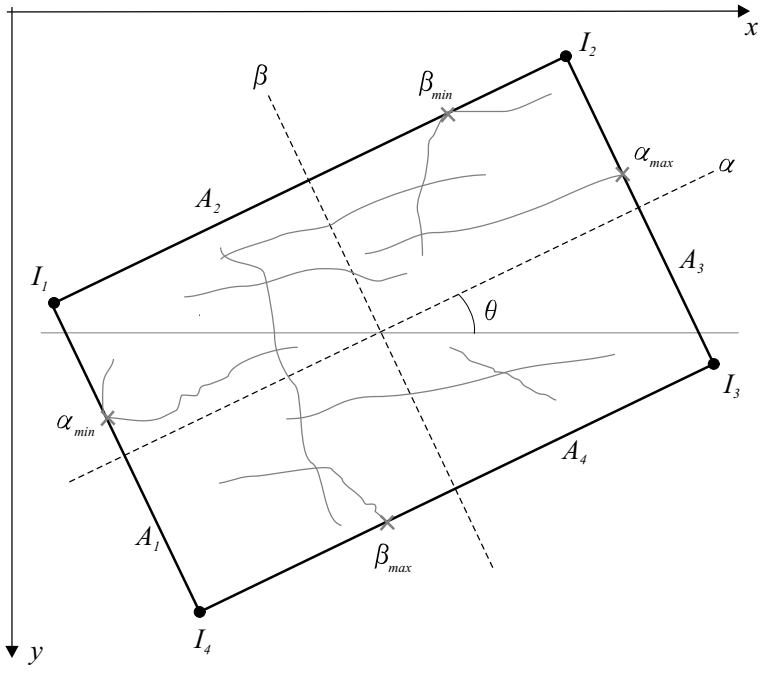
RMBR can also be reconstructed using a different centroid and, in fact, any point within the image can be used. Using the centre of mass (see Equation 1.4) to reconstruct an RMBR results in a translated version. This version possesses the same properties as the genuine RMBR, except that it is tuned to shift itself towards an area of high pixel concentration in the crack-network. In cases where the concentration is not constant, a bounding rectangle that does not enclose the whole of the crack pixels is produced and this of course contradicts with the definition of conservative approximation (see Section 4.6). However, in order to open more options for analysis, the definition is not followed strictly.

4.7 Crack-network Pruning

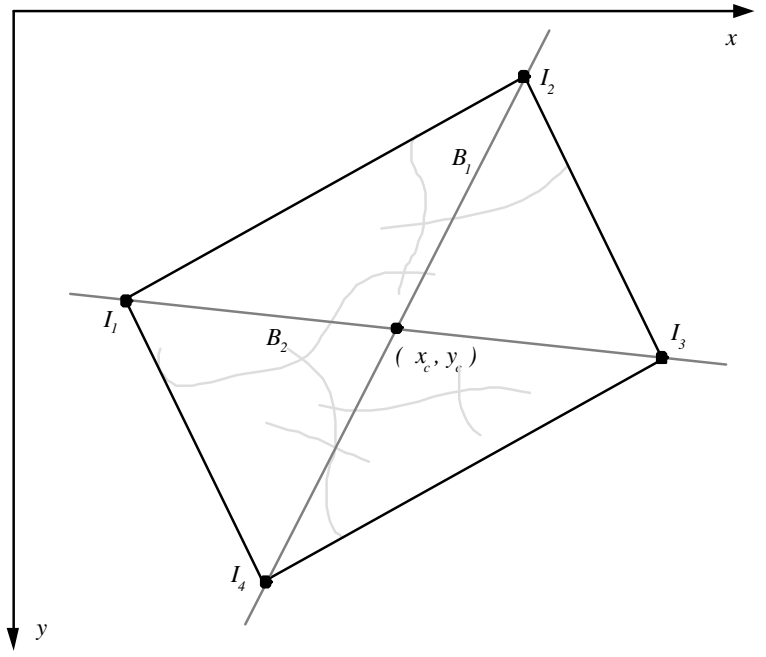
One available option once crack patterns have been structurally represented by a *network tree*, is to perform pruning in order to eliminate any unwanted crack-network. "Unwanted" in this case refers to the significance of a crack-network or the level in which it influences the outcome of a certain process. From a different perspective, it can also be any elements that are suspected to be noise originating from the output of the crack detection stage.

Among the criteria that can be used as an indication of the significance level is total length, number of nodes and number of line segments. Among these, total length is the most reasonable criterion. As an example, it can be assumed that a "valid" crack-network in some cases can appear without any nodes or with only a single line segment. On the contrary, short cracks will certainly pose minimal significance if the rest of the crack-network is lengthy.

Noise in most cases in the analysis possesses three characteristics; it is short in length, very localised and densely populated. The density of crack pixels in an area can also be used as



(a)



(b)

Figure 4.12: Computing the centre of RMBR using geometrical techniques: (a) knowing θ, straight lines A_1, A_2, A_3 and A_4 are computed from the knowledge about maximum and minimum pixel points α_{min}, α_{max}, β_{min} and β_{max} which are then used to reveal the corner points I_1, I_2, I_3 and I_4; (b) straight lines B_1 and B_2 are used to find the centre of RMBR (x_c, y_c).

a cue to determine if the crack-network should be deleted from the list. The crack density is computed by taking the total number of pixels over an area covered by the RMBR of that particular crack-network (refer Section 4.4). The noise populated areas are modelled from the ratio of the RMBR dimensions, computed as shown by Equation 5.6.1.3. The dimension ratio attempts to model the elongatedness of a crack-network.

Noise is differentiated from the actual crack pixels by setting three threshold limits. Let L_{th} be the threshold for the minimum allowable length of a crack-network, D_{th} the threshold for the maximum allowable crack density and R_{th} the threshold for the maximum allowable dimension ratio. Short crack-networks are suspected to be noise and the same goes for crack-networks which possess high population density with high dimension ratio. Crack-networks with high population density can either belong to noise or unidirectional crack patterns. In order to model the behaviour of noise, the dimension ratio is used. Thus, pruning is imposed on a crack-network only if one of the following conditions is satisfied:

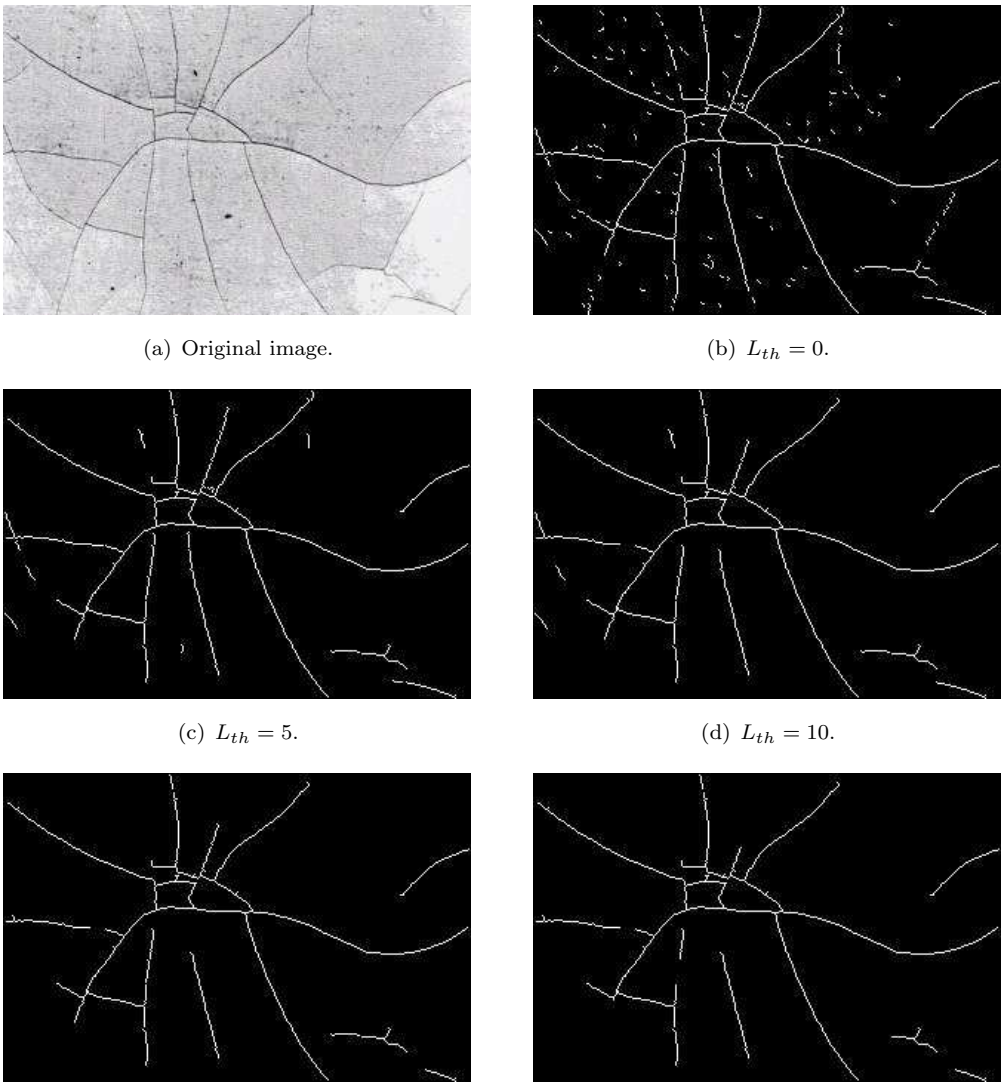
- (crack-network length $< L_{th}$)
- (crack-network density $> D_{th}$) AND (crack-network dimension ratio $> R_{th}$).

In the implementation, the *network tree* is traversed to search for networks that satisfy the pruning condition. Once these networks have been detected, they are deleted from the *network tree* leaving only "significant" networks behind.

Figure 4.13 shows cracks for several levels of pruning using network length, L_{th} as a pruning criterion, while Figures 4.14 and 4.15 show examples of a crack-network pruned using D_{th} and R_{th} respectively. The cracks are reconstructed after pruning for display purposes.

Based on continuous observations, for most cases, the values chosen for L_{th}, D_{th} and R_{th} are 15, 0.15 and 0.5 respectively. Figure 4.16 shows an example of pruning performed on crack images using these values. This action behaves as a filtering mechanism to eliminate any element of a crack-network which does not fulfil certain criteria. Not only can this functionality be extended to prune a whole crack-network, but it can also be tuned to eliminate short line segments.

Performing the technique with the same parameter values in Figures 3.22(c) and (d) yields the result as shown in Figures 4.17(a) and (b) respectively.



(a) Original image. (b) $L_{th} = 0$.

(c) $L_{th} = 5$. (d) $L_{th} = 10$.

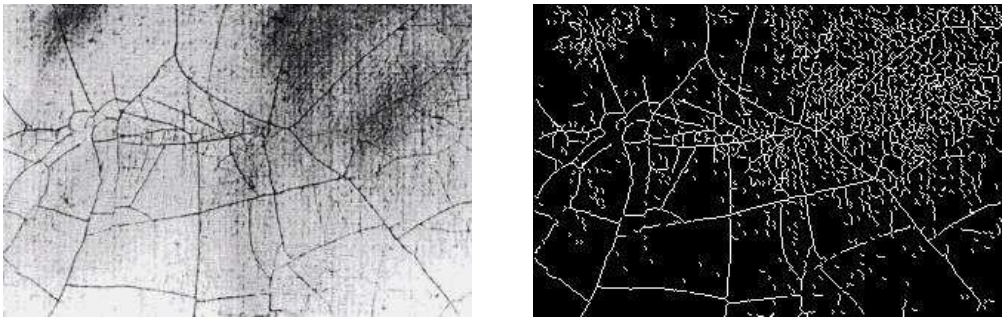
(e) $L_{th} = 15$. (f) $L_{th} = 20$.

Figure 4.13: Crack pruning as a tool to eliminate noise using crack-network length as a cue.

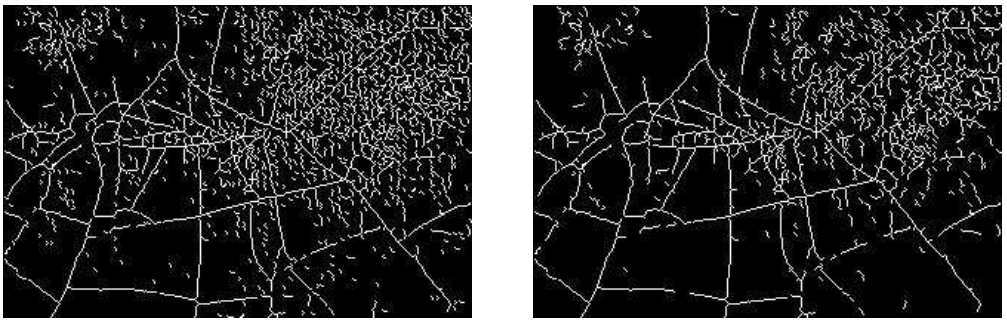
4.8 Summary

This chapter has shown how thinned cracks are represented in a hierarchically structured manner based on the Freeman chain-code. Crack contours are first coded as chains and a structured representation is then built in a hierarchical way, taking into account the fine details (i.e line segments) first and combined into a more global representation (i.e crack-network). Important features are collated as the cracks are "followed". This notion of hierarchical structuring allows a more flexible and effective manner of extracting features and data manipulation in the later stages of the whole system.

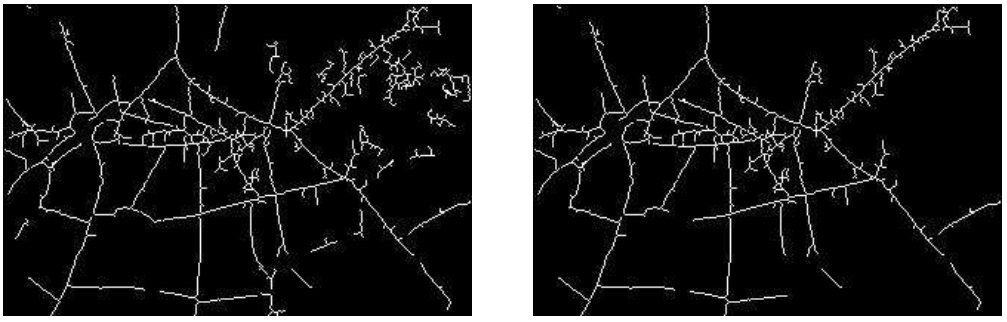
The steps taken to approximate crack-networks using the minimum bounding rectangle



(a) Original image. (b) $D_{th} = 1$



(c) $D_{th} = 0.8$ (d) $D_{th} = 0.6$

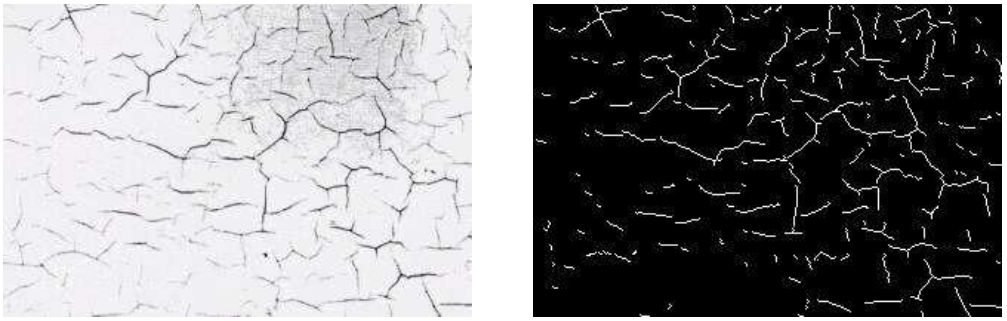


(e) $D_{th} = 0.4$ (f) $D_{th} = 0.2$.

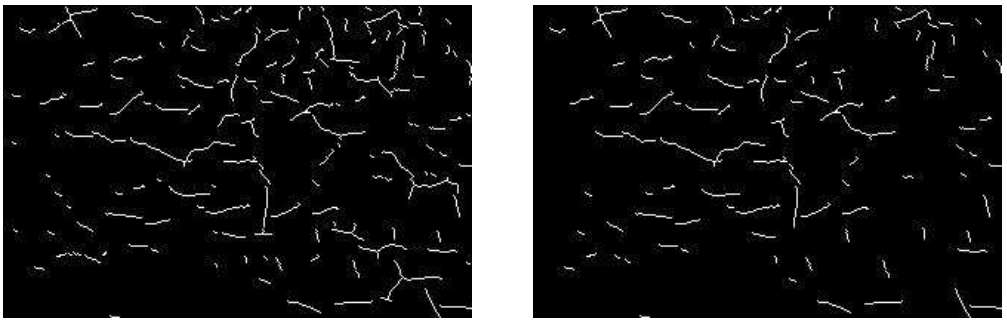
Figure 4.14: Crack pruning using crack-network density as a cue.

(MBR) and the rotated minimum bounding rectangle (RMBR) are also presented. These approximations are used to evaluate the regions of interests, for feature extraction and also for visualisation purposes. It can be concluded that the MBR does not possess any information about orientation, which is not good if orientation is important. However, compared to the RMBR, it is less computationally expensive for parameter computation, intersection detection and boundary reconstruction. On the other hand, as opposed to the MBR, the RMBR holds information about orientation. Thus, more computation power is needed for parameter computation, intersection detection and boundary reconstruction. However, this drawback is tolerable with powerful processing capabilities. A more extensive usage of the approximation is explained in Chapter 5.

Finally, it is demonstrated how the hierarchically structured cracks can be put into use



(a) Original image. (b) $D_{th} = 1$.



(c) $D_{th} = 0.5$ (d) $D_{th} = 0.3$

(e) $D_{th} = 0.2$ (f) $D_{th} = 0.1$

Figure 4.15: Crack pruning using crack-network dimension ratio as a cue.

through crack pruning. Noise and insignificant crack-networks can be eliminated by setting conditions based on crack-network length, crack density and dimension ratio (based on the dimensions of the corresponding RMBR).

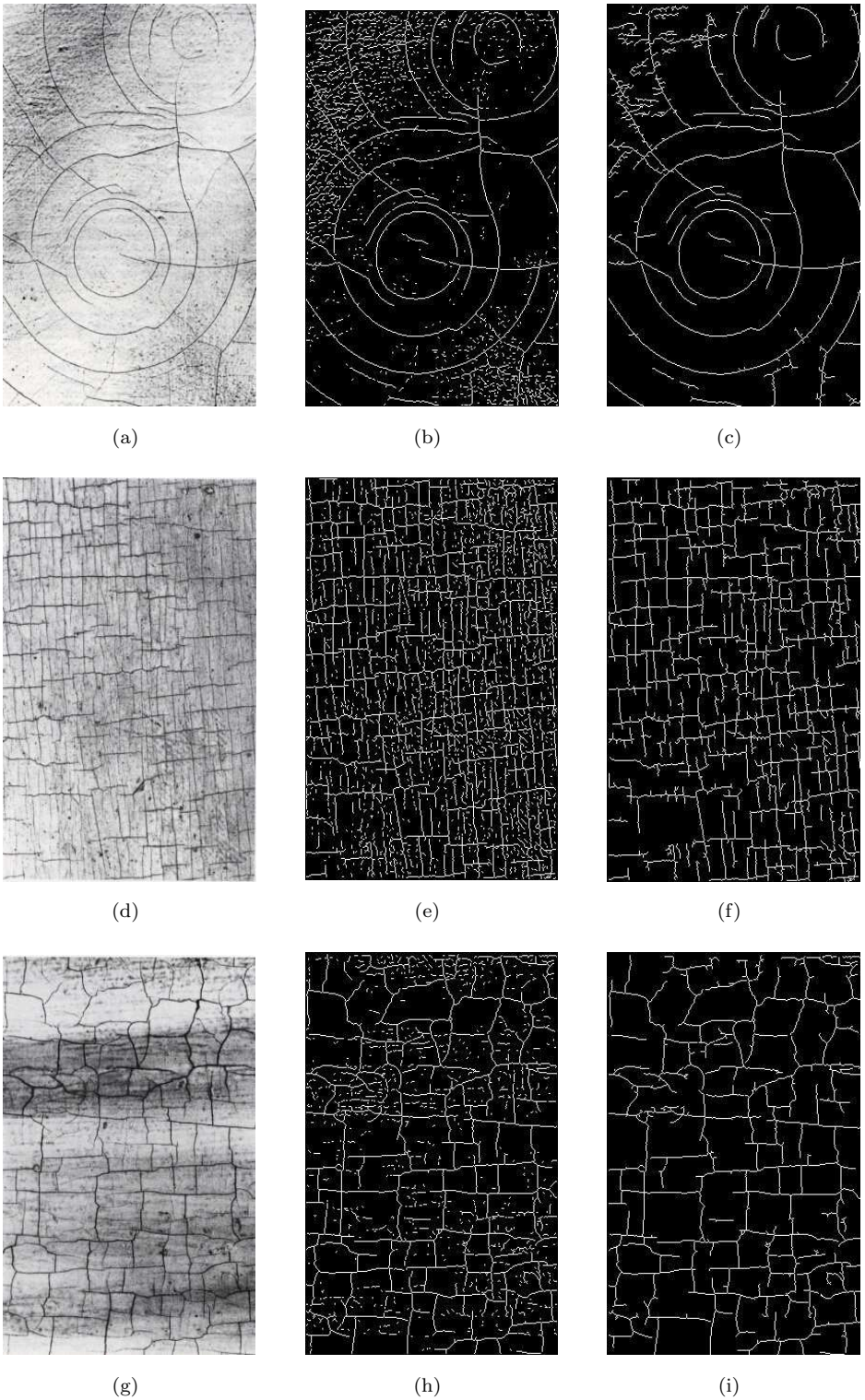
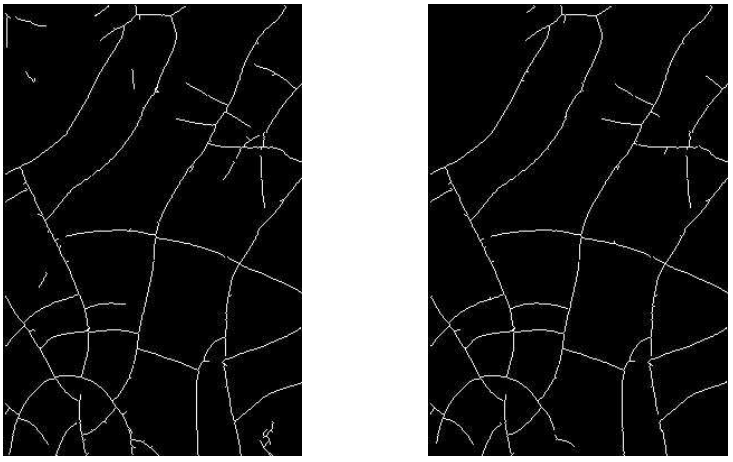


Figure 4.16: Crack-network pruning using $L_{th} = 15$, $D_{th} = 0.15$ and $R_{th} = 0.5$, where (a), (d) and (g) represent the original image while (b), (e) and (h) correspond to their detected cracks. The corresponding pruned version of the cracks are as shown in (c), (f) and (i).



(a) (b)

Figure 4.17: Results of crack-network pruning on images of Figure 3.22(c) (shown in (a)) and 3.22(d) (shown in (b)) using $L_{th} = 15$, $D_{th} = 0.15$ and $R_{th} = 0.5$.

Chapter 5

Content Interpretation and Merging

5.1 Introduction

Query, matching and result representation require objects-of-interest to represent the sub-images. In typical content-based retrieval applications, these areas of interest are either segmented manually or automatically, according to common characteristics such as shape, texture or colour. Segmented subsets of the image are used as objects-of-interest and their features or descriptors are calculated independently. Features computed from user-specified queries are matched with features of these objects-of-interest to obtain the list of most similar matches.

This chapter covers the issues related to the areas of interest in a crack image. The questions surrounding the issue are the manner of segmenting an image containing crack contours into meaningful regions and the type of characteristics used to perform the task. The basics of pattern grouping are investigated first. Then a two-stage crack pattern grouping approach is presented which uses proximity and characteristic rules to merge crack-networks into meaningful objects-of-interest.

5.2 Pattern Grouping

At this point, each crack-network (connected curve) is assumed to represent a meaningful object-of-interest as far as content-based analysis is concerned and this representation is

called a sub-object. A question at this point is whether the sub-object is sufficient to describe a meaningful pattern. Using perceptual means, one crack-network is not sufficient. The reasons for this are two-fold. Firstly, the crack detection process is an inherently unreliable process which results in segmentation errors such as line fragmentation. Secondly, crack patterns should be thought of as a combination of connected curves rather than just single connected curves. Furthermore, the regions covered by a sub-object are too small to offer meaningful features for the purpose of crack classification, information query and result representation.

Line grouping is a crucial stage in the intermediate level of computer vision, to close the gap between what is produced by state-of-the-art low-level algorithms and what is desired as input to high level algorithms. Processes such as edge detection for instance, produce imperfect contours and fragmentation. A line grouping algorithm is an approach that is meant to compensate for this weakness. Robbles Kelly et al. [123] produced groupings of straight line segments using eigenclustering. They were interested in locating groups of straight line segments that exhibit strong geometric affinity to one another. Guy and Medioni [124] demonstrated the use of a threshold-free and non-iterative technique which in a way resembles the Hough Transform [49, 125, 126] in terms of its voting approach. The paper also summarised significant work on line grouping, namely that of Lowe [127], Ahuja & Tuceryan [128], Dolan & Weiss [129], Mohan & Nevatia [130], Sha'ashua & Ullman [131], Parent & Zucker [132] and Heitger & von der Heydt [133].

While almost all work on line grouping has concentrated on grouping line segments and/or curves, this research is interested in grouping patterns which in most cases consist of combinations of line segments, as the end result of the crack network structuring stage as described in Chapter 4. A representation of a crack-network, may evolve from a straight line, a curve, a combination of connected straight lines, a combination of curves and even a mixture of straight lines and curves. Grouping these crack-networks is a different kind of problem compared to the line grouping problem described in almost all the line grouping literature. The dependability of the solution on human perception is a fact that cannot be denied. The research on *perceptual grouping* [134] and *perceptual organization* [135] gains attention for its importance in computer vision. However, the subject is not straightforward. Taking a quote from Sarkar and Boyer [136], it is clear how grouping crack networks on the basis of perception is a highly challenging task:

> "Perception is not a mere passive recording of information impressed upon my sensory organs by the environment. Rather, it consists of an active construction by means of which sensory data are selected, analyzed, and integrated with

> properties not directly noticeable but only hypothesized, deduced, or anticipated, according to available information and intellectual capacities".

Perceptual organization, going back through history, is research that looks for underlying principles which would unify the various grouping abilities of human perception [135]. In the early twenties, Wertheimer [113], Koffka and Köhler founded the *Gestalt School of Psychology*, and demonstrated the importance of perceptual organization to human visual perception. The first categorisation of perceptual organization rules, made by Max Wertheimer, is known as *Wertheimer's Laws of Grouping*. Some of the rules are as denoted below [134, 135, 137].

- Proximity - elements that are close.
- Similarity - elements which have similar shape, colour, orientation, size, etc.
- Continuation - elements that lie on a line or a smooth curve.
- Closure - elements, like lines or curves, that form a closed shape.
- Symmetry - elements which are placed in symmetric order.
- Familiarity - elements which are used to be seen together.

In most literature, these rules are usually denoted as *grouping cues* or just *cues*. The *Gestalt* psychologists [113] were among the first to address the issue of pre-attentive perception [124]. Many "laws of grouping" were formulated, but none in any algorithmic language. The choice of *cues* varies heavily with the anticipated outcome of the grouping process. For instance, to group fragments of line edges from an edge detection process, *proximity* and *continuation* are the most appropriate criteria. On the other hand, in texture analysis, *cues* such as *similarity* and *familiarity* are the most likely to succeed, if used to group or segment regular texture patterns. According to Sonka et al. [49], there are mutually related elements, which, in the literature on texture analysis, are known as *primitives* or *texels*.

The current problem is viewed in a similar way. As discussed in Chapter 4, crack patterns originated from the most primitive form, a pixel, then combine with other primitives to form a line segment, eventually forming a crack-network which is a group of connected line-segments. This hierarchical structure is formed through a grouping process which is based on *connectivity*. Grouping processes vary tremendously in terms of level of difficulty from quick tasks of proximity and co-linearity based groupings to the difficult knowledge-based approach.

In the problem in hand, and looking from a general perspective, the *cues* that are appropriate in grouping crack-networks are *proximity* and *similarity*. The crack-network grouping

scheme implemented considers a crack-network as the primitive. Bearing in mind the diverse structural form of a crack-network, a typical *cue* used for line-segments such as *continuation* cannot successfully group crack-networks.

In order to "measure" proximity and similarity, a two-stage technique is implemented using conservative shape approximations characterised by the minimum bounding rectangle (MBR) and the rotated minimum bounding rectangle (RMBR) which were discussed in Section 4.6.

The concerns in the analysis are time consumption and simplicity. Working on a very large image requires a large amount of computational load and the complex mathematical calculations will significantly slow down the process. Thus, simple approximations are chosen for a crack-network such as the MBR and the RMBR. They require a small number of parameters for object approximation as opposed to the convex hull [49], which needs a variable number of parameters. With a huge amount of crack-network in an image, the task of representing each of them by a simple approximation is highly desirable.

5.3 Merging Algorithms

From a crack-network approximation, a measure of homogeneity has to be established in order to group the objects together. In doing this, several criteria or characteristics can be taken into consideration. Prior knowledge as to how crack-networks should be grouped contributes towards deciding which criteria to employ. At this stage, it cannot be assumed that RMBRs of similar orientation should be grouped together, and at the same time it cannot be ruled out the possibility that they should not be. It depends on the anticipated end result. If the end result is expected to be a unidirectional pattern, it is highly desirable to group them. On the other hand, it is less desirable if a different pattern is expected. For example, an area with unidirectional pattern implies using different grouping criteria. This dilemma makes pattern characteristics less effective as a grouping *cue* at this stage of the process and this is where a 2-stage process becomes appropriate.

A criterion based on proximity and object location is expected to produce visually better results compared to a characteristic criterion at this first grouping stage. By referring to Figure 5.1, B is more likely to be grouped with A compared to C although B and C are closer in resemblance. From observations, it is more appropriate for merging at this point to assume that two RMBRs be evaluated in terms of the distance between them rather than their appearance.

Implementation-wise, the crack-networks are organised as a linked list, as explained in Chapter 4. This enables structured data manipulation approaches. In order to group the crack-networks based on the *proximity cue*, two data merging algorithms are experimented with, namely the *merge and expand* (M&E) and the *label and merge* (L&M) approaches.

5.3.1 The Merge and Expand Approach

Let $CN = [\lambda_1, \lambda_2, ..., \lambda_n]$ be a list of crack-networks of structure λ where n is the total number of networks in the list. The first algorithm which called the *merge and expand* (M&E) is an iterative technique visualised in Figure 5.1 and explained as follows.

1. $i = 1, 2, ..., n$ and $k = i+1$, compare the shape approximations of λ_i and λ_{i+k} to check for merging.
2. If λ_1 and λ_{i+k} satisfy a merging rule.
   - 2.1. Compute combined features and reconstruct shape approximation for λ_i and λ_{i+k}.
   - 2.2. Rearrange crack-network list: $CN = [\lambda_1, \lambda_2, ..., \lambda_{n-1}]$.
   - 2.3. $n = n - 1$.
3. $k = k + 1$.
4. Repeat steps (2) and (3) until $i + k = n$.
5. $i = i + 1$.
6. Repeat steps (4) and (5) until $i = n - 1$.

Refering to Figure 5.1, A and B are first merged, since they partially overlap one another. A new bounding rectangle is formed to represent a combination of A and B. The algorithm then checks if the newly formed bounding rectangle overlaps with another bounding rectangle. In the case of Figure 5.1, it overlaps C, and, as a result, a new bounding rectangle is formed. The same process is then repeated until no overlaps are detected. This approach can be seen as a "lenient" approach, since it easily merges bounding rectangles.

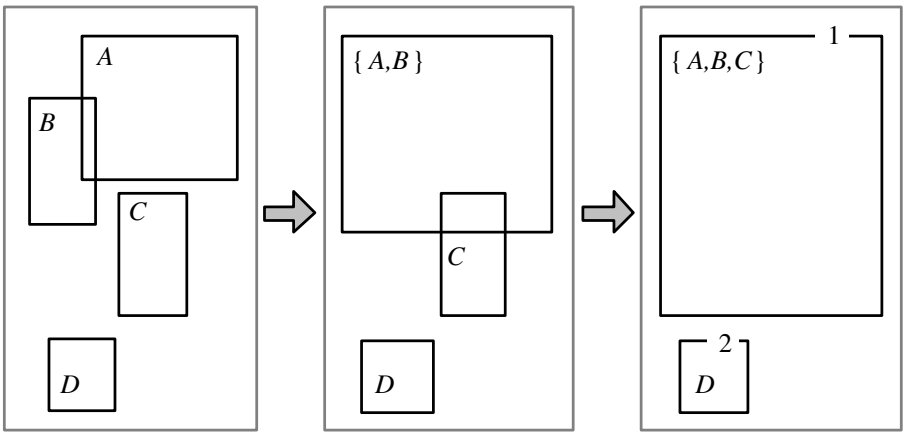


Figure 5.1: The *merge and expand* approach.

5.3.2 The Label and Merge Approach

The second approach is different from the M&E in the sense that it labels every "connected" RMBR at a first run and merges their properties in a second run. This approach is named the *label and merge* (L&M) technique as shown visually in Figure 5.2 and as explained by the following algorithm.

1. $i = 1, 2, ..., n$ and $k = i+1$, compare the shape approximations of λ_i and λ_{i+k} to check for merging.
2. If λ_1 and λ_{i+k} satisfy a merging rule.
   2.1. Label λ_1 and λ_{i+k} with the same label.
3. $k = k + 1$.
4. Repeat steps (2) and (3) until $i + k = n$.
5. $i = i + 1$.
6. Repeat steps (4) and (5) until $i = n - 1$.
7. $i = 1, 2, ..., n$ and $k = i + 1$, compare the labels of λ_i and λ_{i+k}.
8. If λ_1 and λ_{i+k} have the same label.
   8.1. Compute combined features and reconstruct shape approximation for λ_i and λ_{i+k}.
   8.2. Rearrange crack-network list: $CN = [\lambda_1, \lambda_2, ..., \lambda_{n-1}]$.
   8.3. $n = n - 1$.
9. $k = k + 1$.
10. Repeat steps (8) and (9) until $i + k = n$.
11. $i = i + 1$.
12. Repeat steps (10) and (11) until $i = n - 1$.

This approach attempts to label each connected crack-network first and merge them when all the crack-networks have been labelled. Referring to Figure 5.2, A, B, C, G and E are given similar labels since they are connected. D and F are similarly labelled while H stands on its own. The final outcome are three approximations. As opposed to M&E, this approach can be regarded as a "strict" merging approach, due to the fact that it does not merge crack-networks as easily as the M&E approach.

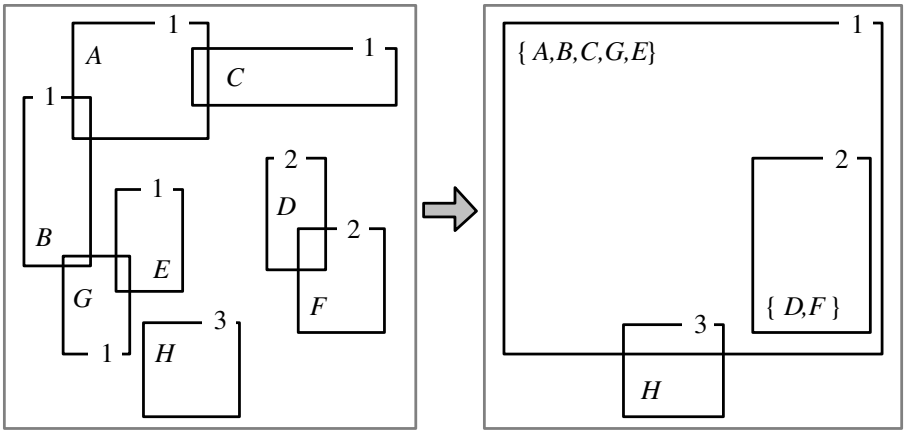


Figure 5.2: The *label and merge* approach.

5.4 Object Merging Using Proximity Rules

In the scope of the current work, proximity can be defined in several ways. To decide whether two approximations should be merged, there are several decision rules that can be considered. The first being a measure of distance between two approximations and the second being an assessment of logical operations between two approximations based on intersection.

Ideally, two approximations should be merged only if the distance between them is below a permissible degree. There are two complications on using proximity based on calculated distance, which are:

1. Deciding which two points (or more) between the approximations are to be used as representative points, and
2. Finding the optimum distance between approximations or to set a threshold for the distance.

The practical scenario for this problem is complicated by the existence of a variable number of suspected objects-of-interest in a crack image. There can be a minimum of one or any number of objects-of-interest in a crack image. The distances are the elements that can be used to define "relatedness" among the disconnected crack-networks. However, as mentioned in the previous paragraph, one of the trickiest dilemmas is in assigning a representative point to a crack-network.

Robbles Kelly et al. [123] used end points of a line segment as representatives to the object-of-interest. This is an appropriate approach due to the fact that they are dealing

with single line segments rather than a "network" of lines. Furthermore, they are not concerned with connecting lines which are parallel in location but rather with those which are serially connected (end point to end point).

Heitger and von der Heydt [133] made use of anistropic selective filters which are combined pair-wise to recover occluding contours. The scheme takes as input end points and T-junctions, and results in the most natural connections of those. The advantage of assigning representative points (assuming the reliable and appropriate representations are chosen), is accuracy, because the higher the number of representation points, the more accurate the representation becomes. However, this is at the expense of processing time and information processing complexity, which are not desirable in real-time applications.

Another means to define proximity is through logical assessment, by assessing intersections between approximations [103]. Two approximations will only be merged if they overlap or intersect. In mathematical terms, refering to Figure 5.3, A and B are merged only if $A \cap B \neq \emptyset$. The key phrase which best describes this approach is the generalisation of information, which, is also worth stressing, can be an advantage for certain applications and a disadvantage for others. The details of an object seem to be made more general or vague when it comes to this type of object representation. However, in a situation where it is relatively difficult to determine the representative points, and pixel-to-pixel accuracy is not a major concern, this approach might be the most appropriate. The core advantage of it is that it avoids threshold determination. Depending on how efficiently intersections are calculated and determined, faster processing speed can also be an advantage of this approach.

Looking again at the crack-network grouping issue, the number of crack-networks is directly proportional to the number of fragmentations caused by segmentation errors. On the other hand, the mean crack-network length in an image is inversely proportional to the number of fragmentations. Considering the structural diversity of a crack-network, it is not straightforward to select reliable end points as representative points. A crack-network can have any number of end points and to determine the relation between end points of adjacent crack-networks is a cumbersome and complicated process.

Furthermore, as mentioned before, selecting representative points to compute distances between objects needs high-level postprocessing to determine the optimum distance between objects. This is seen as a huge obstacle since there is no literature related to crack analysis in paintings that covers the subject of content-based analysis of craquelure and so the perceptual model of craquelure is still non-existent. Thus, a generalised approach is chosen by utilising the MBR and the RMBR as a model of proximity by looking for intersections.

5.4.1 The MBR Overlap Test

Theoretically, the intersection between two MBRs (see Figure 5.3) is determined by evaluating whether they overlap, which is a straightforward process. In mathematical terms, the overlap test for an MBR is done as explained in this sub-section. Let P_a and P_b be two sets of corner points for A and B respectively where

- $P_a = \{h_{min_a}, w_{min_a}, h_{max_a}, w_{max_a}\}$,
- $P_b = \{h_{min_b}, w_{min_b}, h_{max_b}, w_{max_b}\}$.

Eight conditions are then defined; r_1,r_2,...,r_8 where

- $r_1 : h_{min_b} \leq h_{min_a} \leq h_{max_b}$,
- $r_2 : w_{min_b} \leq w_{min_a} \leq w_{max_b}$,
- $r_3 : h_{min_b} \leq h_{max_a} \leq h_{max_b}$,
- $r_4 : w_{min_b} \leq w_{max_a} \leq w_{max_b}$,
- $r_5 : h_{min_a} \leq h_{min_b} \leq h_{max_a}$,
- $r_6 : w_{min_a} \leq w_{min_b} \leq w_{max_a}$,
- $r_7 : h_{min_a} \leq h_{max_b} \leq h_{max_a}$,
- $r_8 : w_{min_a} \leq w_{max_b} \leq w_{max_a}$.

A and B are merged if one or more of the following conditions are met. The conditions are $r_1 \wedge r_2$=1, $r_1 \wedge r_3$=1, $r_4 \wedge r_2$=1, $r_4 \wedge r_3$=1, $r_5 \wedge r_6$=1, $r_5 \wedge r_7$=1, $r_8 \wedge r_6$=1 and $r_8 \wedge r_7$=1.

5.4.2 The RMBR Overlap Test

A more complicated computation is required to test overlapping for an RMBR (see Figure 5.4) due to the fact that the sides are not always in the same orientation as the y and x axes. Mathematically, overlapping is detected as the following. Let R_a and R_b be two sets of required parameters of an RMBR, where

- $R_a = \{\theta_a, \breve{y}_a, \bar{x}_a, h_a, w_a\}$, and
- $R_b = \{\theta_b, \breve{y}_b, \bar{x}_b, h_b, w_b\}$,

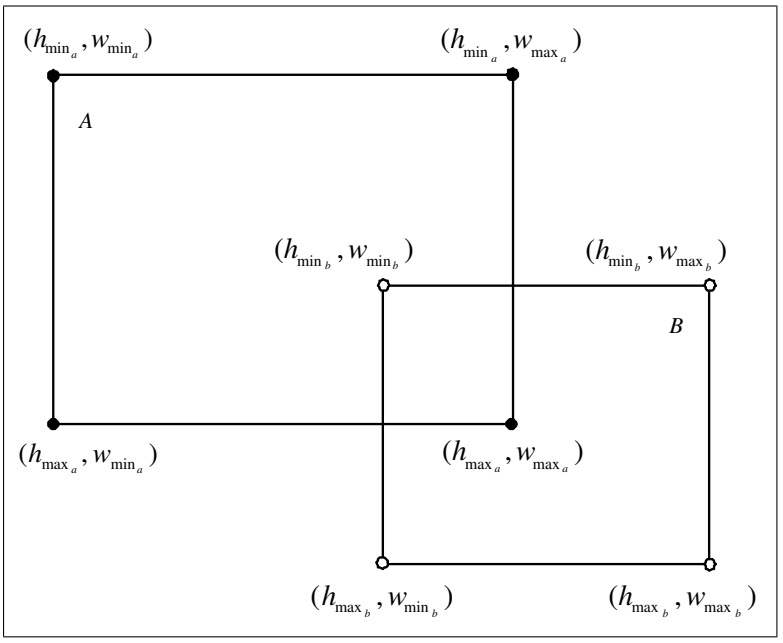


Figure 5.3: MBR A and MBR B overlap each other.

with θ, $\bar{y}$, $\bar{x}$, h and w being the *axis of minimum inertia*, y-centroid, x-centroid, RMBR height and RMBR width respectively. Line intersections and corner points can be calculated from the five parameters using trigonometric computations.

From the 10 parameters of the two RMBRs, 8 corner points and 16 possible line intersections are computed. Let C_a and C_b be two sets of corner coordinate points for A and B respectively, where

- $C_a = \{(y_{a_1}, x_{a_1}), (y_{a_2}, x_{a_2}), ..., (y_{a_4}, x_{a_4})\}$,
- $C_b = \{(y_{b_1}, x_{b_1}), (y_{b_2}, x_{b_2}), ..., (y_{b_4}, x_{b_4})\}$.

Similarly, let I be a set of all possible intersection coordinate points between A and B, where

- $I = \{(\hat{y}_1, \hat{x}_1), (\hat{y}_2, \hat{x}_2), ..., (\hat{y}_{16}, \hat{x}_{16})\}$.

Using C_a, C_b and I, it can be determined whether A intersects with B by underlining some logical rules.

5.5 Results and Discussion

In testing the algorithm, an image consisting of all the five crack pattern types is constructed. This is to show how the algorithm will react to an image consisting of crack

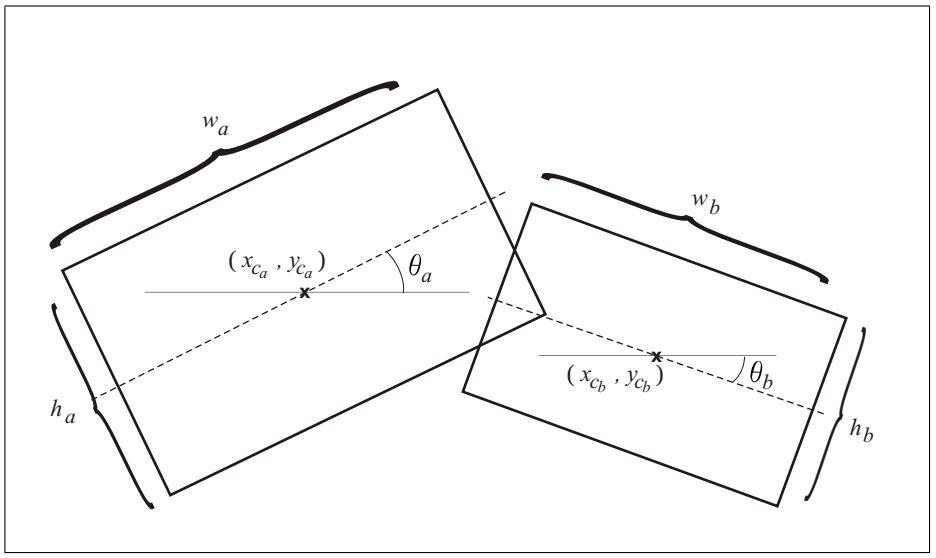


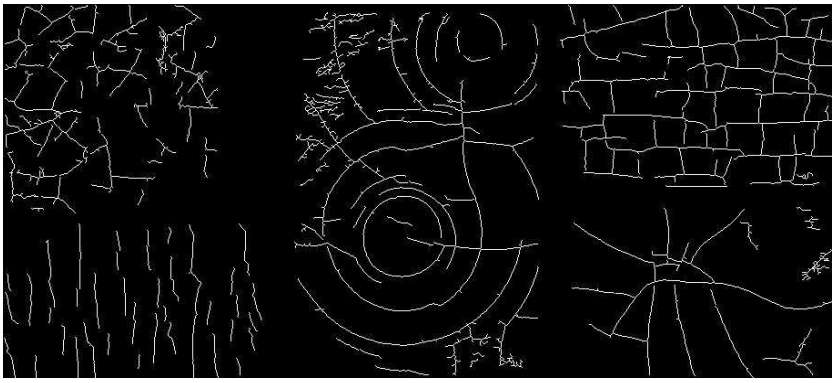
Figure 5.4: RMBR A and RMBR B overlap each other.

patterns of various types. To simplify the situation, the patterns are kept at a specific distance. The crack detected representation of the image is shown in Figure 5.5(a) as well as their crack-network approximations using MBR and RMBR, as shown in Figures 5.5(b) and (c) respectively.

The first stage of merging is attempted on several images to look at the effectiveness of the algorithm. Bearing in mind that no quantitative computation can be made to the resulting outcome in order to evaluate the results, the evaluation is based on observations. Figure 5.6 illustrates the results for this first grouping stage.

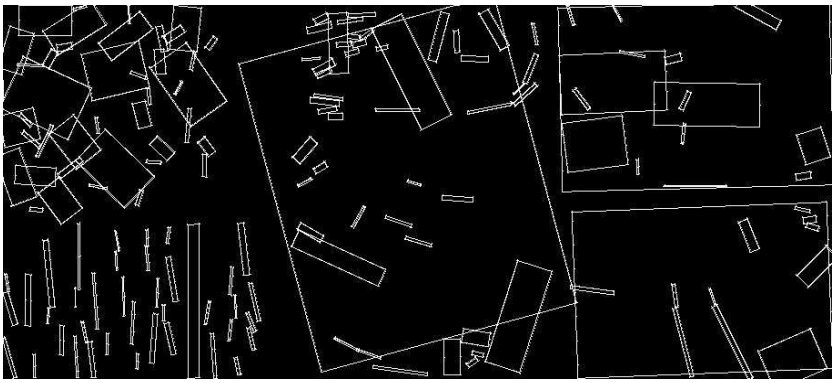
As can be seen, Figure 5.6(a) produces visually good results, the closest to the expected outcome. It is important to note that although it produces the best result, it does not mean that the same technique will work similarly well on other images. The reason why M&E/MBR produces the best result is because the patterns are arranged in a manner and distance which prevents unwanted MBR intersections.

As can be seen from Figure 5.6(c), the small RMBRs at the bottom left of the image are supposed to be merged. If grouped, they will form a unidirectional pattern. However, since their RMBRs do not intersect, they are not grouped at this point. The next section explains an approach taken to deal with this situation.



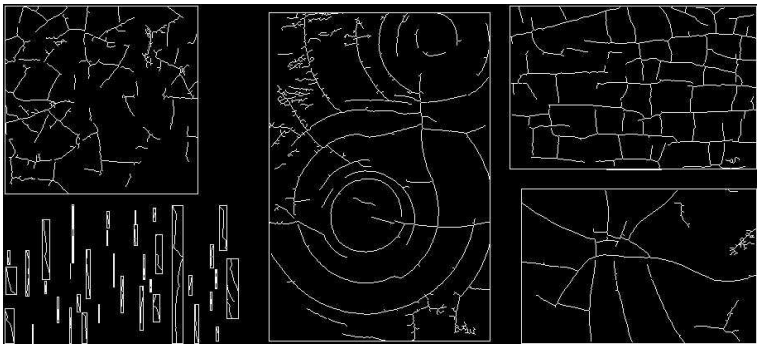
(a)

(b)

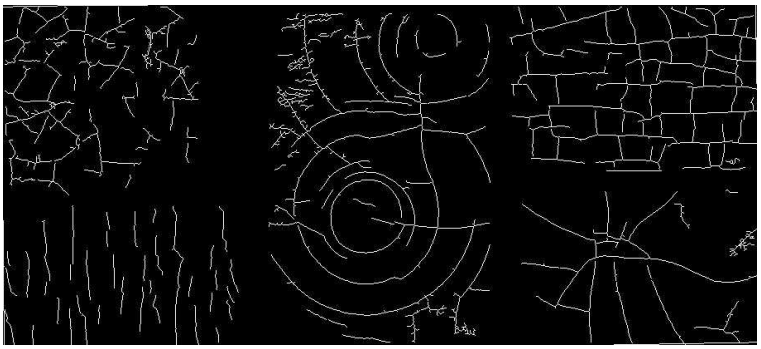


(c)

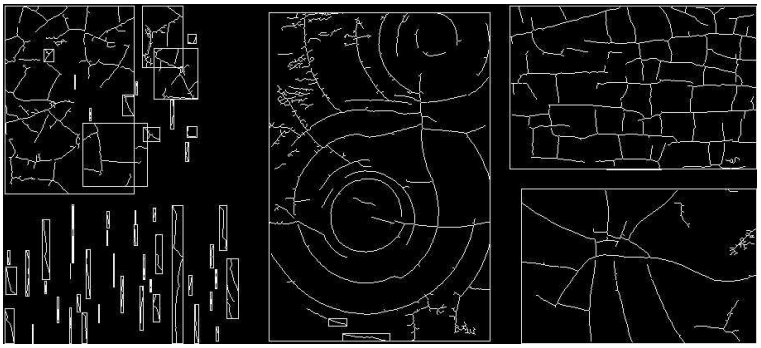
Figure 5.5: A test image consisting of all five crack pattern types: a) the detected crack pattern; b) MBR representation; c) RMBR representation.



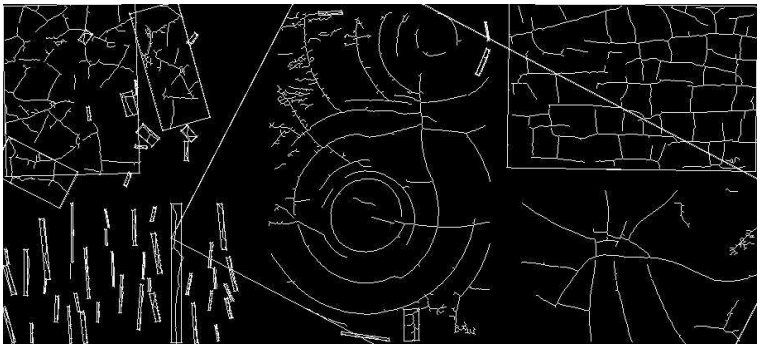
(a) M&E/MBR.



(b) M&E/RMBR



(c) L&M/MBR



(d) L&M/RMBR

Figure 5.6: Grouping of crack patterns using proximity rules, where results are shown for the combinations of the L&M and M&E techniques using the MBR and RMBR shape approximations.

5.6 Object Merging Using Characteristic Rules

Based on the results shown in Section 5.5, RMBR pairs which are in close proximity with each other have been merged. However, a second stage is still needed in order to group RMBRs which do not conform to the merging rule but still possess a great level of similarity with their neighbours. In principle, crack-networks will be considered to belong to the group if they are within a certain distance apart and possess tolerable similarities. In some cases, the first level grouping alone is sufficiently successful in grouping the supposedly "related" crack-networks.

However, in defining a meaningful object-of-interest for a query, a reliable combination of crack-networks over a considerably spacious area in an image is desirable, and this includes short and isolated crack-networks. In some cases, short and isolated crack-networks are left isolated because their shape approximations do not intersect with one another. This is the situation where crack-network grouping using a *proximity cue* failed to perform and the suitable *cue* here is *similarity*. Figure 5.6(c) shows crack-networks which are supposed to be part of the same object-of-interest, but are not merged because they do not satisfy the *proximity* rule. A *similarity* rule will group these isolated crack-networks. Simple features are used as characteristic representations for each object. The features are calculated according to values obtained from the conservative approximations (i.e. the MBR and RMBR) or/and the crack itself. In the ensuing discussions, the resultant objects of grouping from the first stage are addressed as sub-objects to indicate the difference between them and the objects-of-interest.

5.6.1 Object Characteristics

Based on early experiments, several features are identified as potentially good descriptors for further grouping. These features cover the aspects of orientation, density, location, and size of either the objects themselves or their shape approximation.

5.6.1.1 Object Centroid Relative to Image Size

The object centroid is denoted by the y-centroid ($\bar{y}$) and the x-centroid ($\bar{x}$). In principle, these two features are used to represent the location of the object. Let the image height and width be m and n respectively. The object centroid relative to the image size is computed as

$$\tilde{y} = \frac{\bar{y}}{m} \qquad \text{and} \qquad \tilde{x} = \frac{\bar{x}}{n} \quad . \tag{5.1}$$

5.6.1.2 Size of The Object

The size of the object can be modelled in three ways, first using the number of pixels that formed the cracks, secondly by modelling it from the square root of the area of its RMBR given as

$$a = \sqrt{h_R \cdot w_R} \quad , \tag{5.2}$$

and thirdly by computing the perimeter of its RMBR given as

$$p = 2(h_R + w_R) \quad . \tag{5.3}$$

5.6.1.3 Dimension Ratio

The rotation invariant *dimension ratio* D_R is the ratio between the height h_R and the width w_R or vice versa of an RMBR, expressed as

$$D_R = \begin{cases} h_R/w_R & \text{if } w_R > h_R \\ w_R/h_R & \text{if } w_R < h_R \\ 1 & \text{if } w_R = h_R \end{cases} \quad . \tag{5.4}$$

5.6.1.4 Axis of Minimum Inertia

The axis of minimum inertia (see Equation 4.1) attempts to model the dominating orientation of an object. A slight modification to this measure is needed in order to perform valid angle comparisons. Angular distance cannot be computed by directly assigning representative angles and subtracting two values. More of this will be explained later in the chapter.

The relevance of this feature increases as the *dimension ratio* decreases.

5.6.1.5 Node Density

Assuming that in an object, there are x nodes and y number of pixels, node density can be defined as $\delta = x/y$. The *node density* tells how dense a node distribution is in an object. Another way of measuring node density is by relating the number of nodes with respect to the size of the RMBR, mathematically described as $\delta = x/(h_R \cdot w_R)$. However, this approach is not so good in the sense that it produces very small values. A better way of measuring node density is by taking $\sqrt{\delta}$ as a representation.

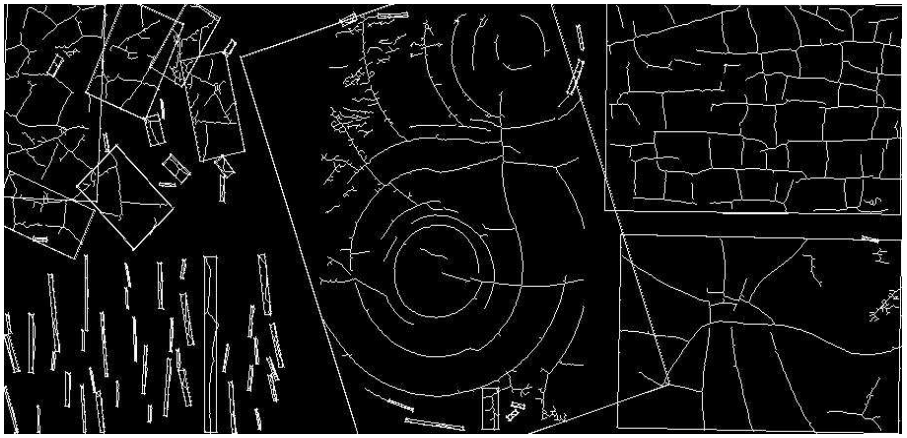
These features are arranged as a feature vector, $\mathbf{v} = (v_1, v_2, ..., v_n)$, with n being the number of features. At this point, each merged (at the first level of merging) crack-network can be imagined being represented by the feature vector $\mathbf{v}$. Thus, an image contains a set of feature vectors $I = (\mathbf{v_1}, \mathbf{v_2}, ..., \mathbf{v_m})^T$ where m is the number of objects. This set of feature vectors is used for clustering, in order to reveal similarities among the points/objects represented by the set members.

5.6.2 Cluster Analysis

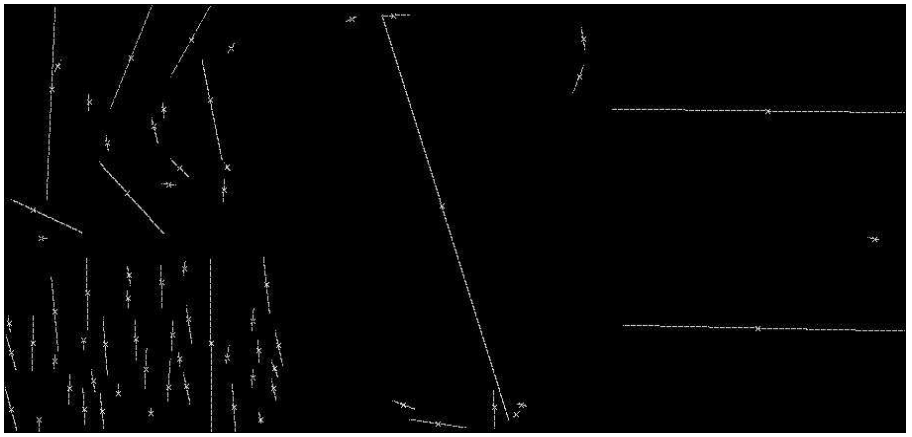
Cluster analysis is the organisation of a collection of patterns (usually represented as a vector of measurements, or a point in a multidimensional space) into clusters based on similarity and, patterns within a cluster are intuitively more similar to each other than there are to pattern belonging to a different cluster [138].

It is crucial to understand the difference between clustering and discriminant analysis, also respectively known as unsupervised and supervised classification [138, 139]. In discriminant analysis, a collection of labelled patterns are provided and a learning/training procedure is conducted to produce decision boundaries [139]. The main task is to label a newly encountered, yet unlabelled pattern based on the decision boundaries. In the case of clustering, the main issue is to group a set of unlabeled patterns into clusters with a degree of similarity among their cluster members. In a sense, labels are also associated with clusters, but these category labels are data driven, which means that they are obtained solely from the data.

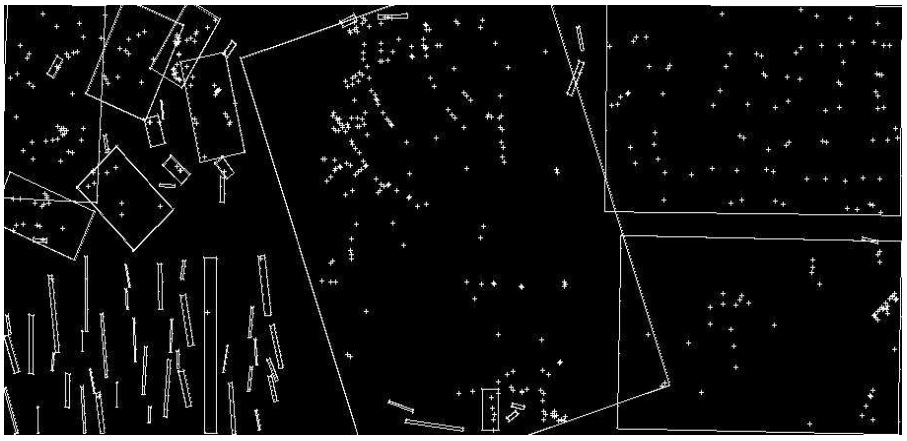
Humans perform competitively in a clustering process in two dimensional space. However, many clustering problems in real applications deal with a higher dimensional feature space. Most patterns are better described using more features. The more features used to describe a pattern, the higher the dimension of a feature space becomes and this is where humans fail, because it is difficult for humans to obtain an intuitive interpretation of data embedded in high-dimensional space. Different approaches to data clustering are also described in



(a)



(b)



(c)

Figure 5.7: Images visualising the features used to characterise a crack-network: a) the ratio of the shorter side against the longer side of an RMBR and the square root of the RMBR area can be used as features; b) the centroid of the crack-networks shown by the "x" sign and the axis of minimum inertia as shown by the dotted lines; c) a node as denoted by the "+" sign is also useful as a feature when their density with respect to either the number of crack pixels or size of the respective RMBR is taken.

[138, 140].

5.6.2.1 The Flexibility of Clustering

There is a vast choice of clustering methodologies with flexibility in their implementation. The choice of a clustering algorithm depends on preferences related to the user and the problem in hand, which might involve issues such as simplicity, speed, flexibility and reliability. The issues surrounding cluster analysis are as follows [138]:

- *Agglomerative* versus *Divisive*: An agglomerative technique begins with each pattern considered as a cluster and successively merges clusters until a stop criterion is satisfied. On the other hand, a divisive technique begins with all patterns in a single cluster and splitting is performed until a stopping rule is achieved.

- *Monothetic* versus *Polythetic*: This issue relates to the sequential or simultaneous use of features in a clustering process. The majority of techniques are *polythetic*, where all features are used to compute distances between patterns and decisions are based on these distances. *Monothetic* approaches consider features sequentially rather than simultaneously. A *monothetic* clustering approach is explained by Anderberg [141], where a dataset is divided into two by using feature f_1. Each of the resultant clusters are further divided independently using feature f_2. The main problem with this approach is that it creates 2^d clusters in a d-dimensional space. In a high dimension feature space, the number of clusters will be so large that the end result will be uninterestingly small and fragmented.

- *Hard* versus *Fuzzy*: A *hard* clustering approach allocates each pattern to a single cluster during operation and as an end result. A fuzzy clustering approach [142] assigns a pattern membership to all clusters in the dataset. Hence, each pattern is associated with every cluster. The patterns will have membership values in $[0, 1]$ for each cluster.

- *Deterministic* versus *Stochastic*: This issue is the most relevant to partitional approaches designed to optimise a squared error function. This optimisation can be accomplished using traditional techniques or through a random search of the state space consisting of all possible labellings.

- *Incremental* versus *Non-incremental*: When a dataset to be clustered is large, execution time or memory space constraints affect the architecture of the algorithm. The advent of data mining has nurtured the development of clustering techniques that

minimise the number of patterns examined during execution, or reduce the size of the data structures used in the process.

5.6.2.2 The Hierarchical Agglomerative Clustering Algorithm

Hierarchical clustering methods are categorised into agglomerative (bottom-up) and divisive (top-down) [140, 143]. As mentioned in Section 5.6.2.1, an agglomerative clustering starts with one point (singleton) clusters and recursively merges two or more of the most appropriate clusters. Divisive clustering starts with one cluster of all data points and recursively splits the most appropriate clusters. The process continues until a stopping criterion is achieved. The advantages of hierarchical clustering include [144]:

- Embedded flexibility regarding the level of granularity.
- Ease of handling of any forms of similarity or distance.
- Consequently, applicability to any attribute types.

The disadvantages of hierarchical clustering relate to:

- Vagueness of termination criteria.
- The fact that most hierarchical algorithms do not revisit intermediate clusters, once constructed for the purpose of their improvement.

A hierarchical algorithm yields a *dendrogram* representing the nested grouping of patterns and similarity levels at which groupings change. A *dendrogram* (see Figure 5.11(b)) is an n-tree [145] with the additional property that a height h is associated with each of the internal nodes. For each pair of objects, (i,j), h_{ij} is defined as the height of the internal node, specifying the smallest class to which both object i and j belong; the smaller the value of h_{ij}, the more similar objects i and j are regarded to be.

The algorithm of a hierarchical agglomerative clustering is as follows:

1. Compute the proximity matrix containing the distance between each pair of patterns and treat each pattern as a cluster.
2. Find the most similar pair of clusters using the proximity matrix.
3. Merge these two clusters into a single cluster.
4. Update the proximity matrix.
5. If all patterns are in one cluster, stop. Otherwise go to step 2.

5.6.2.3 Linkage Metric

Hierarchical agglomerative clustering initialises a cluster system as a set of singleton clusters and proceeds iteratively with merging or splitting of the most appropriate clusters until a stopping criterion is achieved. The appropriateness of clusters for merging depends on the (dis)similarity of cluster elements. This reflects a general presumption that clusters consist of similar points. An important example of (dis)similarity between two points is the distance between them.

To merge subsets of points rather than individual points, the distance between individual points has to be generalised to the distance between subsets. Such a derived proximity measure is called a *linkage metric*. The type of linkage metric used significantly affects hierarchical algorithms, since it reflects the particular concept of *closeness* and *connectivity*. The main inter-cluster linkages [146] include single linkage, complete linkage, average linkage, centroid linkage and incremental sum of squares (ward) linkage [145]. Among these, the single linkage and the complete linkage are the most widely used [138].

In the single linkage scheme, the distance between two clusters is taken as the minimum distance between all pairs of patterns drawn from the two clusters. In the complete linkage scheme, the distance between two clusters is the maximum of all possible distances between points of the two clusters.

Implementation-wise, the choice of linkage scheme does effect the outcome of the hierarchical process. The complete linkage algorithm produces tightly bound or compact clusters [147]. The single linkage on the other hand suffers from a chaining effect [148] and has a tendency to produce clusters that are straggly or elongated. However, in most applications, it has been observed that complete linkage produces more useful hierarchies compared to the single linkage [140].

Let n_r be the number of objects in cluster r and n_s be the number of objects in cluster s. x_{ri} is the ith object in cluster r and x_{sj} is the jth object in cluster s. The distance between two clusters $d(r, s)$ using the linkage metrics are as follows:

- Single Linkage: smallest distance between objects in the two clusters.

$$d(r, s) = \min(d\left(x_{ri}, x_{sj}\right)), \quad i \in (1, ..., n_r), \quad j \in (1, ..., n_s) \tag{5.5}$$

- Complete Linkage: largest distance between objects in the two clusters.

$$d(r, s) = \max(d\left(x_{ri}, x_{sj}\right)), \quad i \in (1, ..., n_r), \quad j \in (1, ..., n_s) \tag{5.6}$$

- Average Linkage: average distance between all pairs of objects in the two clusters.

$$d(r,s) = \frac{1}{n_r n_s} \sum_{i=1}^{n_r} \sum_{j=1}^{n_s} d\left(x_{ri}, x_{sj}\right) \tag{5.7}$$

- Centroid Linkage: distance between centroids of the two clusters.

$$d(r,s) = d(\bar{x_r}, \bar{x_s}) \tag{5.8}$$

$$\text{where} \quad \bar{x_r} = \frac{1}{n_r} \sum_{i=1}^{n_r} x_{ri} \quad \text{and} \quad \bar{x_s} = \frac{1}{n_s} \sum_{j=1}^{n_s} x_{sj} \tag{5.9}$$

- Ward Linkage: incremental sum of squares, which is the increase in the total within-cluster sum of squares as a result of joining clusters r and s.

$$d(r,s) = \left(\frac{n_r n_s}{n_r + n_s} \right) d_{rs}^2 \tag{5.10}$$

where d_{rs}^2 is the distance between the centroids of clusters r and s.

5.6.2.4 Distance Metric

The distance between 2 objects can be calculated using various metrics. Among the common ones are the Euclidean distance, the Manhattan distance (special cases of the Minkowski distance) and the Mahalanobis distance. They are as explained below.

Given two points $\mathbf{x} = (x_1, ..., x_n)$ and $\mathbf{y} = (y_1, ..., y_n)$, the Minkowski metric for calculating the distance between $\mathbf{x}$ and $\mathbf{y}$ (d_{xy}) in Euclidean n-dimensional space $\mathbb{R}^n$ is defined as

$$d_{xy} = \left\{ \sum_{i=1}^{n} \mid x_i - y_i \mid^p \right\}^{1/p}. \tag{5.11}$$

In a special case where $p = 1$, the Minkowski metric gives the Manhattan distance defined in Equation 5.12, and when p=2, the metric represents the Euclidean distance described by Equations 5.13 and 5.14.

The Manhattan distance function $(p = 1)$ which is also known as the City-Block distance, computes the distance that would be travelled to get from one data point to the other if a grid-like path is followed. The Manhattan distance between x and y is the sum of the differences of their corresponding components computed as

$$d_{xy} = \sum_{i=1}^{n} \mid x_i - y_i \mid . \tag{5.12}$$

The Euclidean distance ($p = 2$) between x and y is computed as

$$d_{xy} = \sqrt{\mid x_1 - y_1 \mid^2 + \mid x_2 - y_2 \mid^2 + ... + \mid x_n - y_n \mid^2} = \sqrt{\sum_{i=1}^{n} \mid x_i - y_i \mid^2}. \tag{5.13}$$

or in matrix form written as

$$d_{xy} = \sqrt{(x_1 - y_1)^T (x_1 - y_1)}. \tag{5.14}$$

The statistical distance or Mahalanobis distance between x and y is defined as

$$d_{xy} = \sqrt{(x_1 - y_1)^T C^{-1} (x_1 - y_1)}. \tag{5.15}$$

where C^{-1} is the covariance matrix of the dataset.

5.6.3 Feature Normalisation

Experimentally, every image tested has a variable number of RMBRs. The number of feature vectors associated with each image depends on the number of RMBRs. In general, the numerical value for a feature v depends on the units used, i.e. on the scale. If v is multiplied by a scale factor a, then both the mean and the standard deviation are multiplied by a. (The variance is multiplied by a^2). It is desirable to scale the data so that the resulting standard deviation is unity. Traditionally, this is done by dividing v by the standard deviation s. Similarly, in measuring the distance from v to μ (μ is the mean of the respective feature), it often makes sense to measure it relative to the standard deviation. The mean μ and the standard deviation s of a feature over all input samples are computed as in Equations 5.16 and 5.17.

$$\mu = \frac{1}{M} \sum_{i=1}^{M} x_i \tag{5.16}$$

$$\sigma = \sqrt{\frac{1}{M-1} \sum_{i=1}^{M} (x_i - \mu)^2} \tag{5.17}$$

where M is the number of input samples. The calculation is performed for all features resulting in N number of means and standard deviations, where N is the number of features used. From a set of feature vectors $I = (V_1, V_2, ..., V_M)$, with $V = (v_1, v_2, ..., v_N)^T$ representing a feature vector, feature values are normalised according to

$$\tilde{v_i} = \frac{v_i - \mu}{s} \qquad \forall \quad i = 1, 2, ..., M. \tag{5.18}$$

The same procedure is carried out on all features. An N-dimensional hierarchical clustering is performed using $\mathbf{v} = (v_1,v_2,...,v_n)$ as the features of interest. Ideally, the features should be first normalised over all RMBRs to produce a mean of zero and standard deviation of unity for each feature element. However, the usual step cannot be performed here for all features considering that θ is represented in polar coordinates (see Figure 5.8(b)) with r=1. All the other features used are represented in Cartesian coordinates (see Figure 5.8(a)). These angular values do not represent the "true location" of the orientation. Taking the distance between two points in a polar coordinate and calculating the distance may in some cases result in false distance.

Looking at the polar value representations in Figure 5.8(b), the distance between $\theta_2 = \pi/16$ and $\theta_3 = -\pi/16$, which is $\pi/8$, represents the true account of distance between two radial points. However, the difference between points $\theta_1 = 7\pi/16$ and $\theta_4 = -7\pi/16$ tells an opposite story. The distance between θ_1 and θ_4 should be $\pi/8$, similar to the one between $\theta_2 = \pi/16$ and $\theta_3 = -\pi/16$. The maximum distance d between any two points in this version of polar coordinate representation is $\pi/2$. A result which exceeds this maximum distance is produced if the absolute difference is taken between θ_1 and θ_4, which results in $7\pi/8$. Thus, a different approach towards finding the distance is needed.

In finding the correct distance between two θ values, π and -π translated versions of either one of the two orientation features are computed (see Figures 5.9(b) and (c)). This is clearly necessary, since two orientations in exactly opposite directions (difference of π) should be considered the same. For instance, $\pi/8$ and $-7\pi/8$ are two different values, but they represent the same orientation as far as the object-of-interest is concerned.

In order to compute the actual orientation distance d_θ, three distances are calculated. The first distance is an absolute value difference between θ_1 and θ_2. The second distance is an absolute value difference between θ_1 and θ_2, translated by π, while the remaining distance is an absolute value difference between θ_1 and θ_2, translated by -π. These three distances are visually explained in Figure 5.9. From the three values, the minimum value is taken to represent the absolute distance $|d_\theta|$ between the two orientations under consideration, θ_1 and θ_2. However, for the calculation of a distance matrix, the actual value d_θ is used. Equation 5.19 further explains the method.

$$|d_\theta| = \min\left(|\theta_1 - \theta_2|, |\theta_1 - (\theta_2 + \pi)|, |\theta_1 - (\theta_2 - \pi)|\right) \tag{5.19}$$

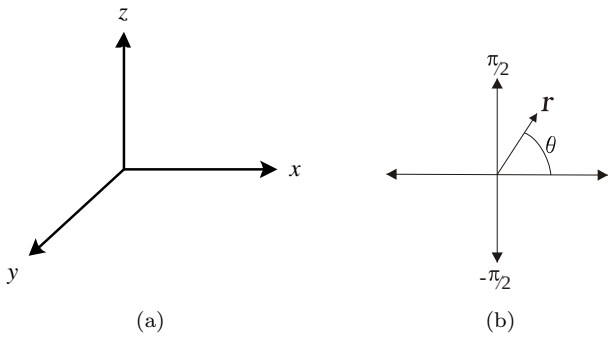


Figure 5.8: Coordinate systems: a) cartesian and b) polar.

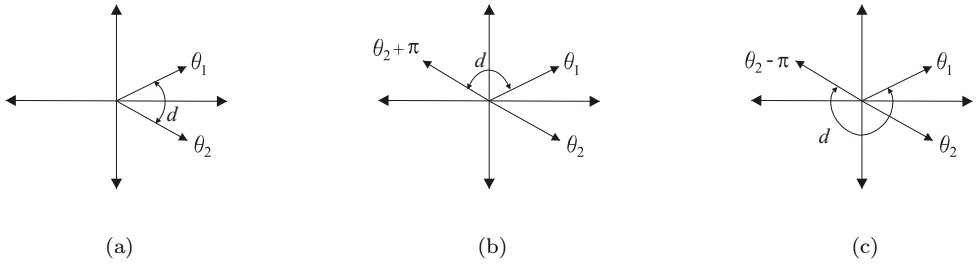


Figure 5.9: The three possibilities of actual orientation distance between θ_1 and θ_2: a) difference between θ_1 and θ_2, b) difference between θ_1 and π translated version of θ_2 and c) difference between θ_1 and -π translated version of θ_2.

A slight complication occurs at this point, because of the fact that, in traditional techniques, as well as this initial approach, features are normalised first before distance is calculated for clustering. The unusual way in which distance between orientations is evaluated requires the orientation features normalised after difference is computed. Implementation-wise, distance calculation is performed before the distance is normalised with respect to the standard deviation of the respective features. Let $\{\boldsymbol{x}_1, \boldsymbol{x}_2, \boldsymbol{x}_3, ..., \boldsymbol{x}_m\}$ be a vector of data representing feature values, where m is the number of data and $\boldsymbol{x} = (x_1,x_2,...,x_n)$, with n being the number of features. The traditional approach in finding the normalised distance for feature i is performed by first normalising with regard to the mean μ_i and standard deviation σ_i, calculated as

$$\left\{ \frac{x_{(1,i)} - \mu_i}{\sigma_i}, \frac{x_{(2,i)} - \mu_i}{\sigma_i}, ..., \frac{x_{(m,i)} - \mu_i}{\sigma_i} \right\} \tag{5.20}$$

for all i = 1,2,...,n which results in a vector of normalised data $\{\tilde{\boldsymbol{x}}_1, \tilde{\boldsymbol{x}}_2, ..., \tilde{\boldsymbol{x}}_m\}$. Next, all possible distances are computed as

$$\{|\tilde{x}_{(1,i)} - \tilde{x}_{(2,i)}|, ..., |\tilde{x}_{(1,i)} - \tilde{x}_{(m,i)}|, |\tilde{x}_{(2,i)} - \tilde{x}_{(3,i)}|, ..., |\tilde{x}_{(m-1,i)} - \tilde{x}_{(m,i)}|\} \tag{5.21}$$

for all i = 1,2,...,n which reveals the normalised distance matrix represented by $\{\tilde{\boldsymbol{d}}_1$, $\tilde{\boldsymbol{d}}_2$,...,$\tilde{\boldsymbol{d}}_{\frac{m(m-1)}{2}}\}$ with $\tilde{\boldsymbol{d}} = (d_1,d_2,...,d_n)$.

Using any distance metric, a distance vector representing all possible distance pairs can be computed by using the schemas explained in Section 5.6.2.4.

In the modified version, the distance between all possible data pairs are computed and the distances are normalised.

The distances between all possible data pairs are calculated to reveal $(m(m-1)/2)$ x n matrix $\boldsymbol{d} = \{d_{(1,i)}, d_{(2,i)}, ..., d_{(m(m-1)/2,i)}\}$. Normalising a feature requires subtraction of the mean and then dividing by the standard deviation of the feature (see Equation 5.18). By taking the difference between a data pair, it is realised that the translation factor, which is the mean, does not contribute to the end result, since it can be cancelled out. Thus, in this case, only the scaling factor is required, which is the standard deviation of the feature in the computation of the normalised distance. Equation 5.22 shows mathematically how this is done for the first normalised distance $\tilde{d}_1$.

$$\tilde{d}_{(1,i)} = \left|\frac{x_{(1,i)} - \mu_i}{\sigma_i} - \frac{x_{(2,i)} - \mu_i}{\sigma_i}\right| = \left|\frac{x_{(1,i)} - x_{(2,i)}}{\sigma_i}\right| = \frac{d_{(1,i)}}{\sigma_i} \tag{5.22}$$

Thus, to obtain a normalised distance from a vector of distances, the members are just divided by the standard deviation of the vector set such as $\{\frac{d_{(1,i)}}{\sigma_i}, \frac{d_{(2,i)}}{\sigma_i}, ..., \frac{d_{(m(m-1)/2,i)}}{\sigma_i}\} = \{\tilde{d}_{(1,i)}, \tilde{d}_{(2,i)}, ..., \tilde{d}_{(m(m-1)/2,i)}\}$.

5.6.4 Automatically Determining The Number of Clusters

By looking at the crack images of Figure 5.6, the number of objects-of-interest in each of them can be manually determined, but is this possible automatically? By "cutting" through a dendrogram, the number of clusters desired can be manually chosen. A challenge at this point is to estimate an optimum number of clusters from a hierarchically organised data by "cutting" through a correct layer in a dendrogram.

5.6.4.1 Second Order Differential of The Minimum Distance Variance

In the implementation, the standard deviation of distances in the cluster hierarchy is used. With an initial number of m clusters (data points), let t be the number of iterations needed by a clustering process to cluster m until only one cluster remains. For all cases, the number of iterations is $t = m - 1$, since points are compared in pair-wise form. For each iteration, the standard deviation of minimum distances between data points is calculated, regardless of whether these points are the initial points or clustered points. The values for the distances depend on the linkage metric used (single-linkage, complete-linkage, average-linkage, centroid-linkage or ward-linkage). Given m data points, the distance vector is a set of all possible distances between data points. Thus, the distance vector is a vector of length $z = \left((m-1) \cdot \frac{m}{2}\right)$. Let the distance vector $\mathbf{d} = (d_1, d_2, ..., d_z)$ be a set of distances for a single clustering iteration.

Let A, B, C, D, E and F be sample points and the distance between points represented by d_{ij}, where i is the first point and j the second point. In the first iteration of the hierarchical clustering process, ζ distances are obtained, namely $(d_{AB}, d_{AC}, d_{AD}, d_{AE}, d_{AF})$, $(d_{BC}, d_{BD}, d_{BE}, d_{BF})$, (d_{CD}, d_{CE}, d_{CF}), (d_{DE}, d_{DF}) and d_{EF}. The minimum distances among these pairwise distances are taken to construct a vector of minimum distances $\mathbf{f} = (f_1, f_2, ..., f_t)$. The number of members in the vector increases along with the iteration. For each iteration, the standard deviation of $\mathbf{f}$ is calculated and a vector of standard deviation $\mathbf{s}$ is updated. After this first iteration, the whole process is repeated for the second iteration, and at the end of the hierarchical clustering process, complete vector of standard deviations $\mathbf{s} = (s_1, s_2, ..., s_t)$ is obtained.

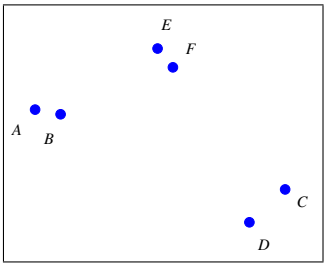


Figure 5.10: Example of scattered data points.

The first derivative of $\mathbf{s}$ is then computed, denoted as $\mathbf{s}'$ which is simply the difference between two successive values computed as

$$\mathbf{s}' : s'_i = s_{i+1} - s_i \qquad \forall \quad i = 1, 2, ..., t-1 \tag{5.23}$$

to produce $\mathbf{s}' = (s'_1, s'_2, ..., s'_{t-1})$. The differential of $\mathbf{s}$ signifies the rate of change of its elements. A positive ($+ve$) value of $\mathbf{s}'$ indicates an increase between two successive elements of $\mathbf{s}$, while a negative ($-ve$) value accounts for a drop. The second derivative $\mathbf{s}''$ from $\mathbf{s}'$ is then computed as

$$\mathbf{s}'' : s''_i = s'_{i+1} - s'_i \qquad \forall \quad i = 1, 2, ..., t-2 \tag{5.24}$$

to obtain $\mathbf{s}'' = (s''_1, s''_2, ..., s''_{t-2})$. $\mathbf{s}''$ represents the changing rate between two successive elements of $\mathbf{s}'$ which, in other terms, detects corner points in $\mathbf{s}$. Taking the maximum and the minimum of $\mathbf{s}''$, the extreme $+ve$ and $-ve$ changes in $\mathbf{s}'$ can be detected. They also indicate points in $\mathbf{s}$ with the sharpest corners. The maximum point in $\mathbf{s}''$ is interesting in the sense that it highlights the highest increase between two successive points in $\mathbf{s}'$ and the sharpest valley (downward corner) among three adjacent points in $\mathbf{s}$.

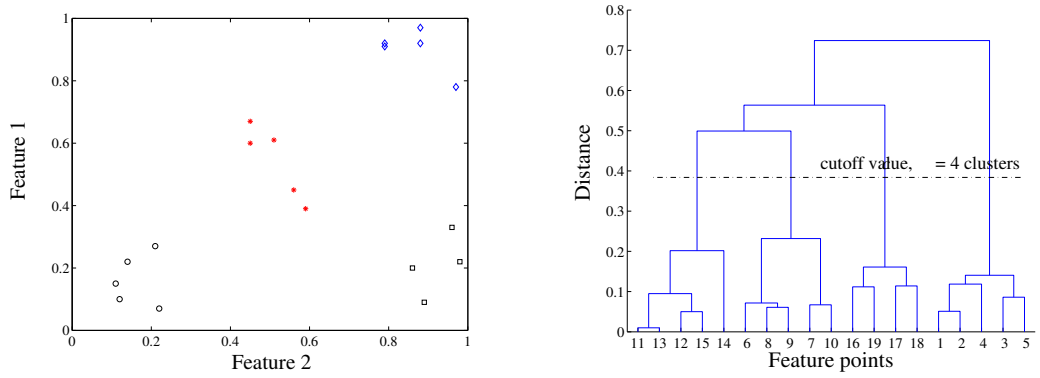
It is understood that the least number of optimum clusters that can be detected from this technique is two and this is represented by a maximum value at the (t-2) point of the $\mathbf{s}''$ plot. On the other hand, the maximum number of clusters that can be achieved is (t-1) at the 1^{st} point of the $\mathbf{s}''$ plot. Thus, in order to computationally reveal the optimum number of clusters, the maximum value in the vector $\mathbf{s}''$ is subtracted from the total number of iterations t to produce a cutoff value c where

$$c = t - \max(\mathbf{s}''). \tag{5.25}$$

As a simple example, Figure 5.11(a) shows 2-dimensional scattered feature points which are quite well clustered. The technique is attempted on the points in order to extract the optimum number of clusters based on the cluster hierarchy, as can be seen in Figure 5.11(b). The technique starts by extracting $\mathbf{s}$ from iterative clustering of the points. Consequently, $\mathbf{s}''$ is then calculated, and, by detecting the maximum in the $\mathbf{s}''$ plot, the optimum number of clusters can be estimated using Equation 5.25. Figures 5.12(a), (b) and (c) show graphical plots of $\mathbf{s}$, $\mathbf{s}'$ and $\mathbf{s}''$ respectively.

Demonstrating the technique on more complicated data, several examples are taken to represent data of different complexities, particularly in terms of cluster separations. The results are shown in Figure 5.13.

It is quite important to note that the technique is sensitive to the linkage metric used (i.e. single, complete, average, centroid and ward) and Figures 5.14 and 5.15 prove this statement. Thus, it is really useful if the "correct" linkage scheme can be chosen. However,



(a) (b)

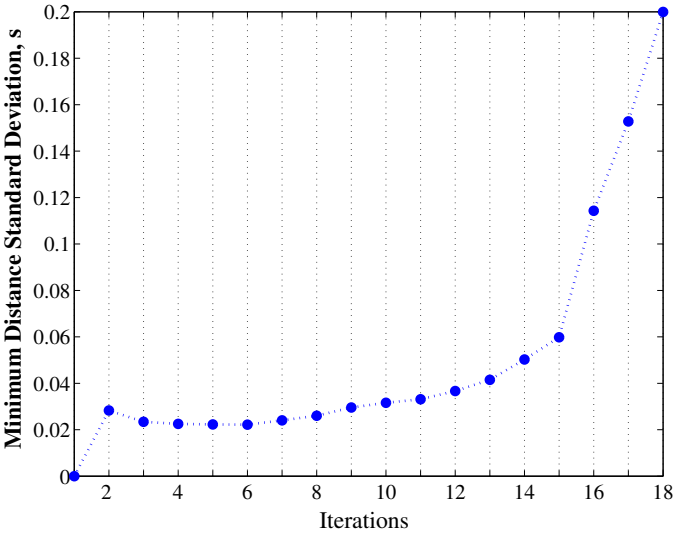
Figure 5.11: The figure shows (a) scattered feature points separated into 4 clusters and (b) the dendrogram of the cluster hierarchy with 4 clusters identified as the optimum number of clusters c based on the procedure explained in Figure 5.12.

measure of goodness among linkage metrics is not easily evaluated, since the basic interpretation of cluster separation is itself a subjective matter, with no concrete ground truth. As a means of choosing the preferred linkage metric, continuous experimentations and observations were performed. Centroid linkage in this case turns out as the most consistently satisfying metric.

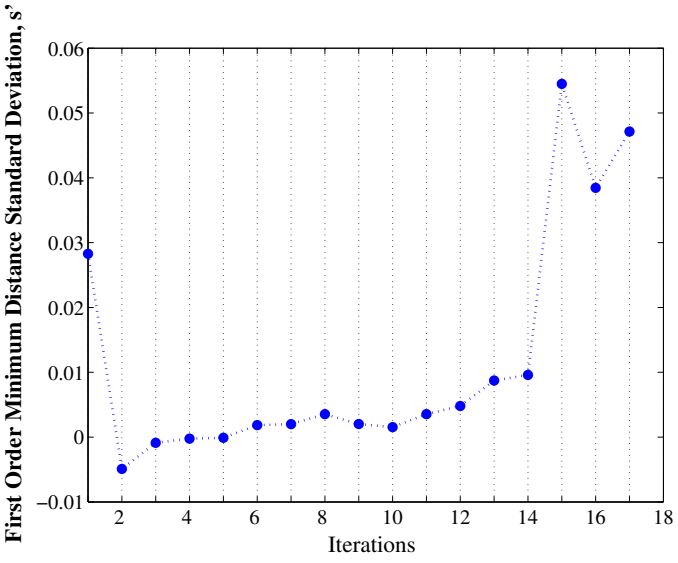
A noticeable drawback of this optimum cluster approach is its failure in making a decision in a one-cluster case and (t-1)-cluster case. Thus, the range of optimum clusters in a data set spans from 2 clusters to (t-1) clusters. This happens because of the amount of information taken off the standard deviation vector from the second order differential operation, which reduces the vector length by as much as two. However, the issue is not seen as a serious drawback and will be ignored at this point.

The features explained in Section 5.6.1 and further summarised in Table 5.1 are calculated for each sub-object (see Section 4.3.1 for definitions). The multidimensional feature space is transformed into a distance matrix $\mathbf{d}$ and normalised (mean of zero and standard deviation of one) using the approach discussed in Section 5.6.3. Hierarchical clustering is then performed, where $\mathbf{s}$ can then be determined and the optimum number of clusters is finally computed. Figure 5.16 illustrates the results.

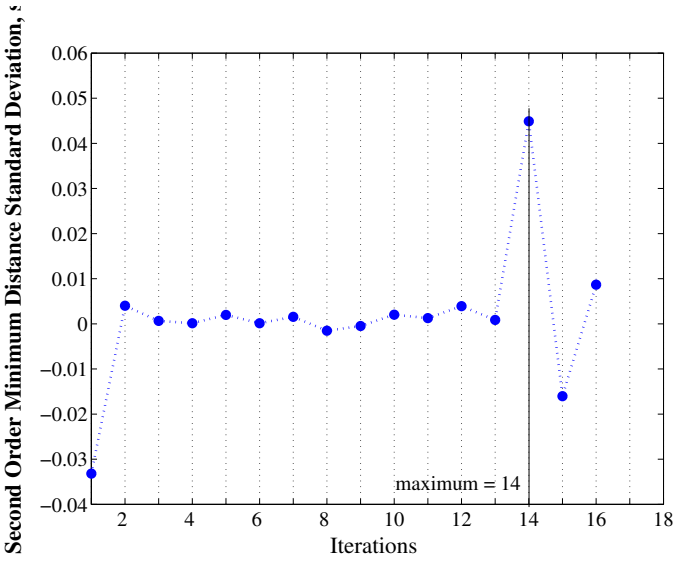
The combination of features is very important in making sure the patterns are correctly grouped. The centroids $\tilde{y}$ and $\tilde{x}$ attempt to model the location of each pattern, thus representing distance between a pair of crack patterns. As explained earlier, the axis of minimum inertia θ is important in grouping patterns with near-similar orientations (unidirectional patterns). However, the use of θ is not constantly applicable among all crack patterns. In other words, θ has a different influence on patterns based on the pattern itself. Since θ



(a)

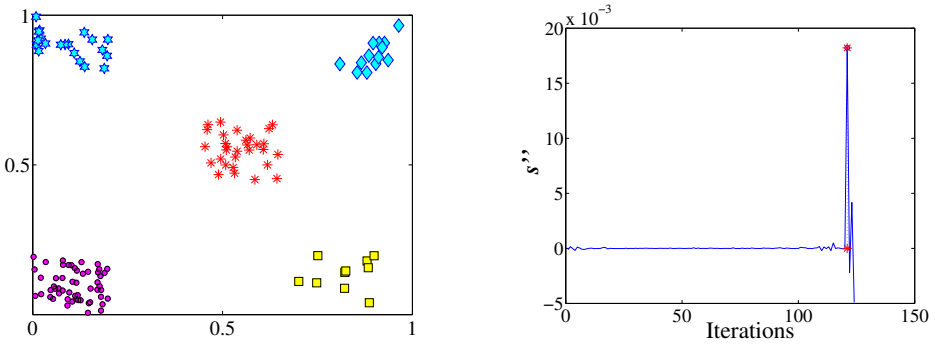


(b)

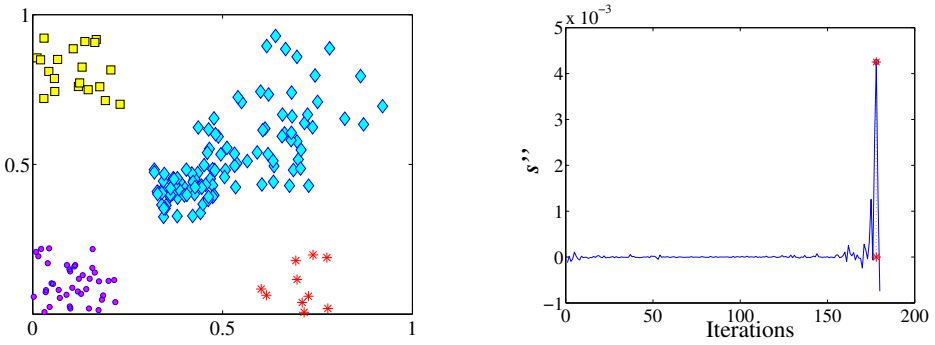


(c)

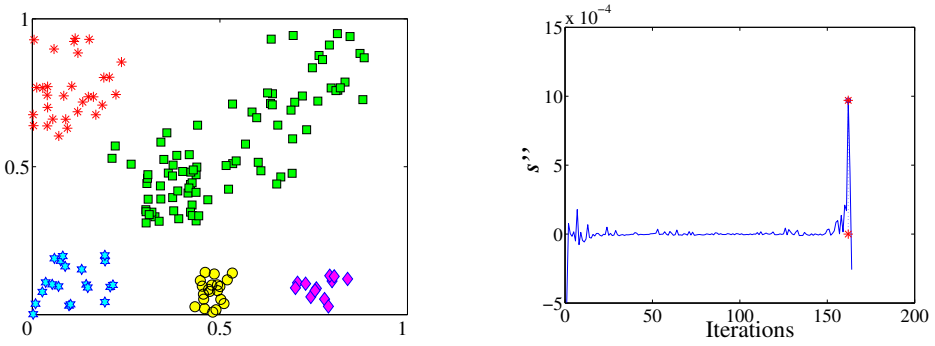
Figure 5.12: Plots of (a) $\mathbf{s}$, (b) $\mathbf{s}'$ and (c) $\mathbf{s}''$. By detecting the maximum in the $\mathbf{s}''$ plot, the optimum number of clusters can be calculated using Equation 5.25.



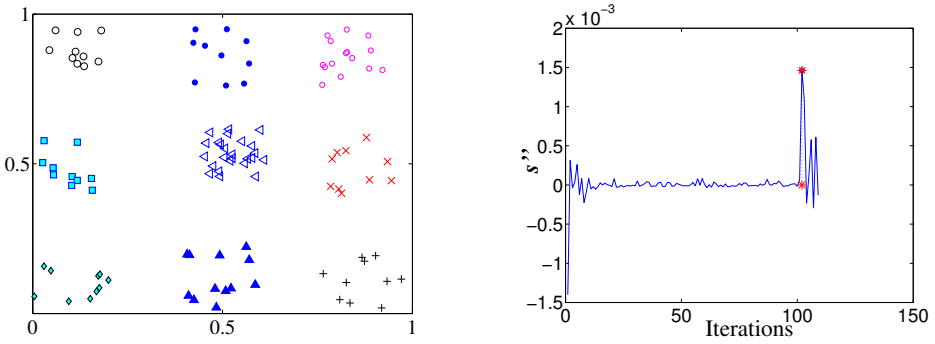
(a) $c = t$ - max(s'') = 126 - 121 = 5.



(b) $c = t$ - max(s'') = 182 - 178 = 4.

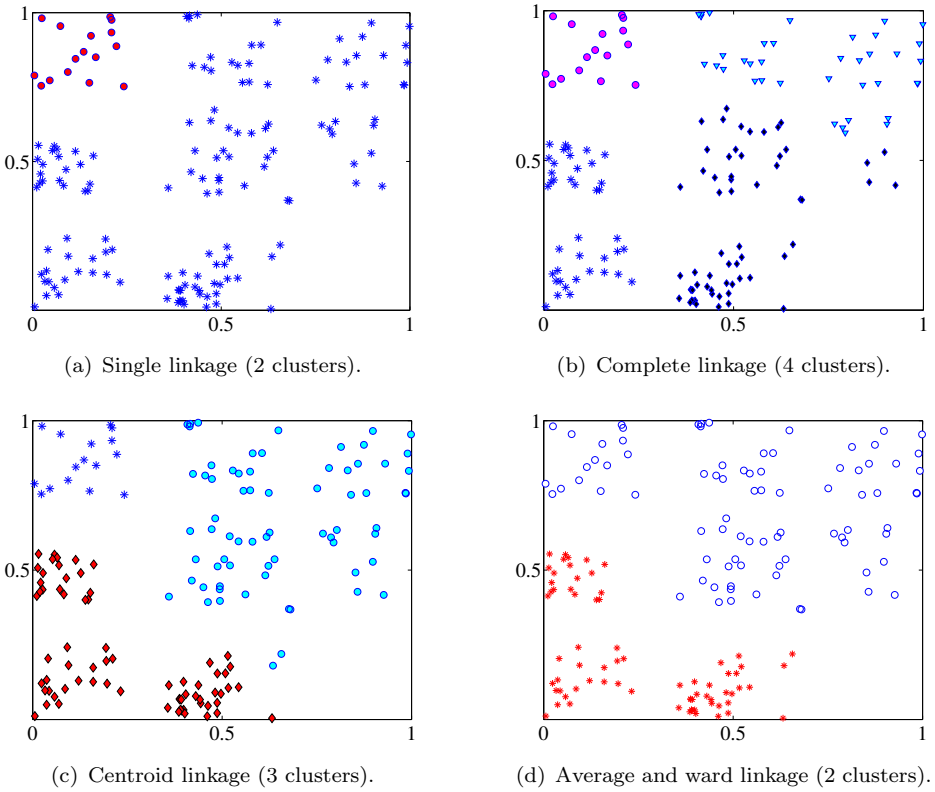


(c) $c = t$ - max(s'') = 166 - 161 = 5.



(d) $c = t$ - max(s'') = 111 - 102 = 9.

Figure 5.13: Figure showing clustered feature points (as shown by the images on the left hand side) and their respective $\mathbf{s}''$ plots on the right. All the results are obtained using the centroid metric.



(a) Single linkage (2 clusters).

(b) Complete linkage (4 clusters).

(c) Centroid linkage (3 clusters).

(d) Average and ward linkage (2 clusters).

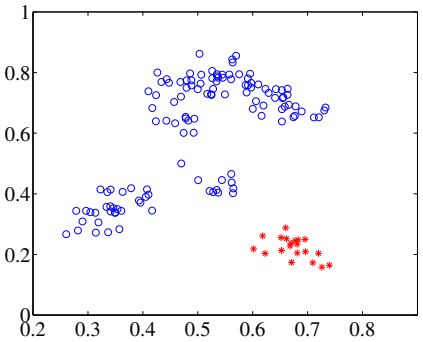
Figure 5.14: Figure demonstrating the sensitivity of using a different linkage metric for clustering somewhat random patterns. The complete linkage tends to produce the most clusters.

Symbol	Feature
$\tilde{y}$	Pattern centroid in the y-axis
$\tilde{x}$	Pattern centroid in the x-axis
θ	Axis of minimum inertia
p	RMBR perimeter
D_R	RMBR dimension ratio
δ	Square root of the node density with respect to the number of pixels

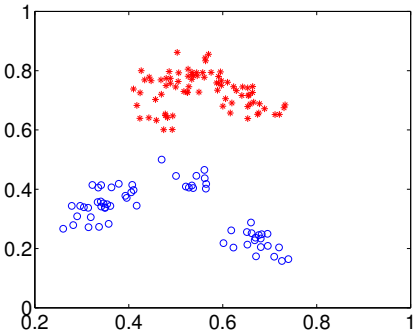
Table 5.1: Features considered for the second stage crack pattern grouping.

itself originates from an approximation of a crack pattern based on the RMBR, it is safe to say that RMBRs with low dimension ratio, (D_R) possess a more "meaningful" θ compared to RMBRs with high D_R. In computative terms, a weighted θ is a better representation of orientation in an approximation of crack pattern. For clustering, D_R is used as a weighting for θ. Letting D_{R_A} and D_{R_B} be dimension ratios of pattern A and B respectively, the weighted distance between A and B as far as orientation is concerned is defined as

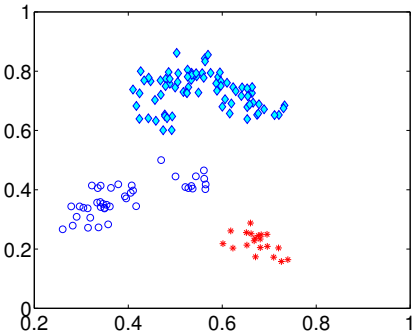
$$d_w(\theta_A, \theta_B) = \begin{cases} d_{\theta_1}(D_{R_A} + D_{R_B}) & \text{if} \quad \min\left(d_{\theta_1}, d_{\theta_2}, d_{\theta_3}\right) = d_{\theta_1} \\ d_{\theta_2}(D_{R_A} + D_{R_B}) & \text{if} \quad \min\left(d_{\theta_1}, d_{\theta_2}, d_{\theta_3}\right) = d_{\theta_2} \\ d_{\theta_1}(D_{R_A} + D_{R_B}) & \text{if} \quad \min\left(d_{\theta_1}, d_{\theta_2}, d_{\theta_3}\right) = d_{\theta_3} \end{cases}, \tag{5.26}$$



(a) Single linkage (2 clusters).



(b) Complete, average and ward linkage (2 clusters).



(c) Centroid linkage (3 clusters).

Figure 5.15: Figure demonstrating the sensitivity of using different linkage metrics for clustering patterns of various sizes and shapes. The centroid linkage in this case produces the highest number of clusters.

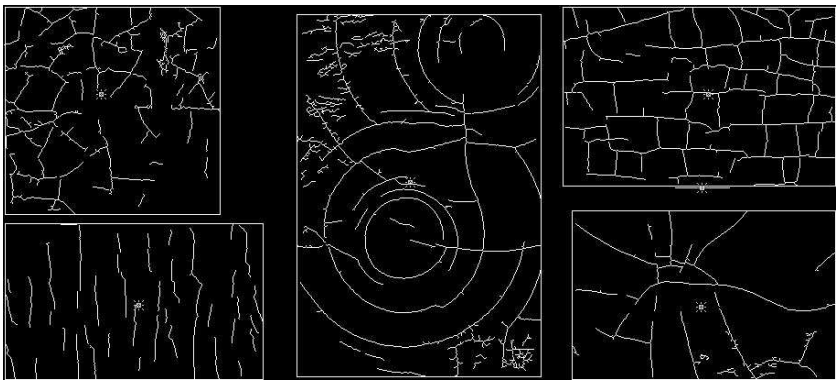
$$d_{\theta_1} = |\theta_A - \theta_B|, \tag{5.27}$$

$$d_{\theta_2} = |\theta_A - (\theta_B + \pi)| \quad and \tag{5.28}$$

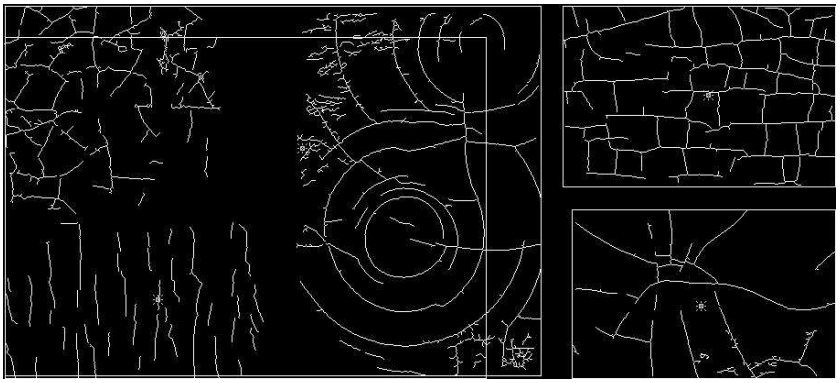
$$d_{\theta_3} = |\theta_A - (\theta_B - \pi)|. \tag{5.29}$$

The square root of the node density δ on the other hand is used to differentiate between structures belonging to unidirectional cracks and the rest of the pattern types. Unfortunately, there is no clear and obvious way of uniquely identifying the four remaining pattern types in a lower level of the crack network tree. Ironically, a circular type pattern may consist of crack-networks with low node density and so do rectangular, spiderweb and random type cracks. The same goes for D_R and p. At the lower level of the hierarchy, only θ presents a clear distinction between classes; unidirectional versus the other four crack types. As a result, it is expected that in a situation where no suspected unidirectional patterns exist, the algorithm differentiates crack-networks based on their centroids. Comparing centroids indirectly gives two types of information; how far the two patterns are apart and how large their combination is.

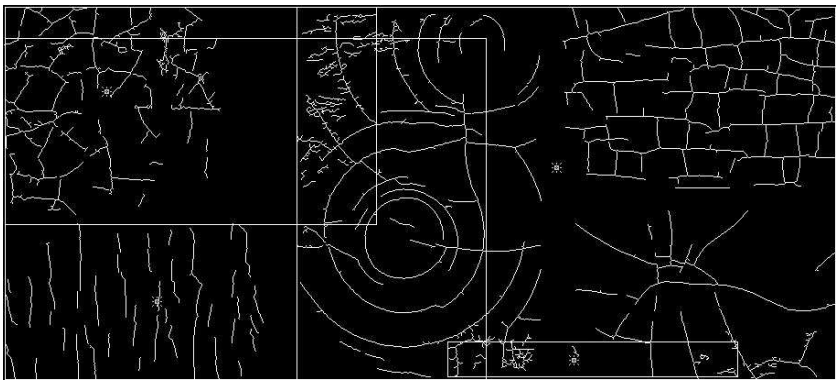
In implementation, the distances used for clustering are $d(\tilde{y})$, $d(\tilde{x})$, $d_w(\theta)$ and $d(\delta)$. The technique is then performed on real craquelure data represented in multi-dimensional feature space using the centroid linkage. The results from Section 5.5 are used, i.e. Figures 5.6(a), (c) and (d). Figure 5.6(b) is not used since the crack patterns have been grouped altogether and thus are not in need of a second grouping stage.



(a) M&E/MBR.



(b) L&M/MBR



(c) L&M/RMBR

Figure 5.16: Crack patterns grouped using proximity rules are later merged using hierarchical agglomerative clustering based on pattern characteristics/features. This figure shows the results of this second stage crack grouping process on images of Figures 5.6(a), (c) and (d).

5.7 Techniques for Crack Pattern Grouping

At this point, choices on the techniques mentioned earlier are left open, in particular, the use of MBR and RMBR in the first crack pattern grouping stage, which does not lead to convincing answers in terms of the advantage each one has to offer over the other.

MBR is clearly better compared to RMBR when computational speed comes into consideration. However, the issue is not a major concern when computing power is available and the number of parameters needed by each means of crack pattern approximation does not hugely differ (refer Table 4.1).

The major difference between the effectiveness of each approach can be seen from how they are influenced by the location of the suspected sub-objects. MBRs are better to use when the suspected sub-objects are somewhat horizontally or vertically aligned (as in the crack patterns of Figure 5.5). However, RMBR is most effective in approximating single-line and elongated crack patterns, where orientation plays an important role. In the case where an elongated-shaped crack pattern is more diagonally aligned, RMBR is more suitable since the orientation information is well-preserved by the RMBR (refer Figure 5.17). Aside from the two aforementioned crack types, orientation does not impose any clear influence on the grouping outcome.

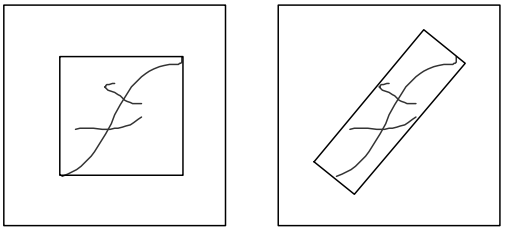


Figure 5.17: Figure showing a comparison between MBR and RMBR in approximating elongated-shaped crack-networks where orientation plays an important role: a) MBR-approximated crack-network; b) RMBR-approximated crack-network.

There are ways to blend MBR and RMBR into a single decision making procedure, without having to decide on the one that offers more advantage, since in grouping crack patterns, the level of randomness and uncertainty is relatively high. The use of MBR and RMBR in deciding whether or not to merge approximated crack patterns are combined together based on area they cover and also on logical rules. These techniques are considered in order to avoid restricted usage of only one medium of crack pattern approximation throughout a single image grouping process.

5.7.1 Merging Based on Area of Approximation

Manual determination introduces inflexibility when merging two crack-networks. An adaptive approach dependent on predetermined rules can suit the task better. One of the merging techniques implemented adapts with respect to the area covered by two crack-network approximations. Let us denote the area covered by MBR_A, MBR_B, RMBR_A and RMBR_B as **area**(MBR_A), **area**(MBR_B), **area**(RMBR_A) and **area**(RMBR_B) respectively. Two approximations produce combined areas **area**(MBR) and **area**(RMBR), computed as

$$\textbf{area}(\text{MBR}) = \textbf{area}(\text{MBR}_A) + \textbf{area}(\text{MBR}_B), \tag{5.30}$$

$$\textbf{area}(\text{RMBR}) = \textbf{area}(\text{RMBR}_A) + \textbf{area}(\text{RMBR}_B). \tag{5.31}$$

From here, a *strict area-based rule* can be applied which employs the approximation using the smaller area **min**(**area**(MBR), **area**(MBR)), computed as

$$\textbf{approximation} = \begin{cases} \text{MBR} & \text{if } \textbf{area}(\text{MBR}) \leq \textbf{area}(\text{RMBR}) \\ \text{RMBR} & \text{if } \textbf{area}(\text{MBR}) > \textbf{area}(\text{RMBR}). \end{cases} \tag{5.32}$$

On the contrary, by taking **max**(**area**(MBR), **area**(MBR)), the *lenient area-based rule* is employed. It is shown as

$$\textbf{approximation} = \begin{cases} \text{RMBR} & \text{if } \textbf{area}(\text{MBR}) \leq \textbf{area}(\text{RMBR}) \\ \text{MBR} & \text{if } \textbf{area}(\text{MBR}) > \textbf{area}(\text{RMBR}). \end{cases} \tag{5.33}$$

At the end of the process, this approach offers a choice of crack pattern approximation by taking into consideration the total area covered by both the approximations under scrutiny.

5.7.2 Merging Based on Logical Rules

Another approach to automatically setting a rule for merging is by using both intersection schemes, i.e. MBR and RMBR. Intersection is checked for both MBR and RMBR approximations. Let **intr**(MBR) be the action of intersection checking between MBR_A and MBR_B and **intr**(RMBR) the intersection checking between RMBR_A and RMBR_B. The actions will produce an output of 1 if true (**intr**() = TRUE) and 0 if false (**intr**() = FALSE). Similar to area-based merging, the strict and lenient rules are defined based on the logical relation between **intr**(MBR) and **intr**(RMBR).

For the *lenient logic-based rule*, two crack patterns are merged if either one or both of **intr**(MBR) and **intr**(RMBR) is/are true,

$$\textbf{merge}(A, B) = \begin{cases} 1 & \text{if } \textbf{intr}(\text{MBR}) \vee \textbf{intr}(\text{RMBR}) = 1 \\ 0 & \text{if } \textbf{intr}(\text{MBR}) \vee \textbf{intr}(\text{RMBR}) = 0. \end{cases} \tag{5.34}$$

On the other hand, for the *strict logic-based rule*, two crack patterns are merged if both **intr**(MBR) and **intr**(RMBR) are true,

$$\textbf{merge}(A, B) = \begin{cases} 1 & \text{if } \textbf{intr}(\text{MBR}) \wedge \textbf{intr}(\text{RMBR}) = 1 \\ 0 & \text{if } \textbf{intr}(\text{MBR}) \wedge \textbf{intr}(\text{RMBR}) = 0. \end{cases} \tag{5.35}$$

This approach is more flexible in coping with the challenging task of merging crack patterns.

In real-life, paintings with craquelure patterns as clearly separated as Figure 5.5 are seldom found . However, for analysis and testing, it is better to use crack patterns with obvious type separations to make evaluations of results easier. In order to observe the effectiveness of the algorithm, the two automatic merging approaches were applied on the same crack patterns as in Figure 5.16. In the experimentation, two "lenient" and two "strict" approaches were used in order to demonstrate the outcome of each process. The general objective is to observe whether the automatic technique can outperform the result obtained by the M&E/MBR approach (see Figure 5.16(a)). Table 5.2 summarises the results, while Figures 5.18 and 5.19 visualise the two outcomes of the "strict" approaches.

Algorithm	Number of cluster(s)
M&E with the *lenient area-based rule*	1
M&E with the *lenient logic-based rule*	1
L&M with the *strict area-based rule*	4
L&M with the *strict logic-based rule*	4

Table 5.2: Results of "lenient" and "strict" automatic merging techniques.

From the results, it can be said in general that the "strict" techniques worked better than the "lenient" techniques in the sense that crack patterns are not easily grouped in the "strict" approach. The test image used draws quite distinctive boundaries between patterns of different type which is quite unlikely to occur in real cases.

5.8 Summary

In this chapter, issues related to the interpretation of objects-of-interest were discussed. The importance of identifying objects-of-interest in content-based analysis lies in the need to specify regions with meaningful patterns, so as to allow sub-image query and retrieval. This requirement is because sub-objects are in most cases not sufficient to represent a meaningful pattern. The reasons for this are two-fold. Firstly, the crack detection process is an inherently unreliable process, which results in segmentation errors such as line fragmentation. Secondly, crack patterns should be thought of as combinations of connected curves rather than just single connected curves. Furthermore, the regions covered by a sub-object are too small to offer meaningful features for the purpose of crack classification, information query and result representation.

Selected literature on pattern grouping and perceptual grouping was reviewed and the key elements for tackling the problem of grouping crack patterns were discussed. Crack pattern approximation using the minimum bounding rectangle (MBR) and the rotated minimum bounding rectangle (RMBR) were used as a basis for further analysis.

A two-stage technique for merging crack patterns is the main contribution of this chapter. The first stage uses proximity as a cue for merging while the second stage uses pattern characteristics as a means for pattern relativity.

Two simple approaches in merging two pattern approximations were introduced as a basis for the first-stage grouping, namely the *merge and expand* (M&E) and the *label and merge* (L&M) techniques. One of the drawbacks of the M&E approach is its high sensitivity to the starting point, i.e. the starting crack-network from which the algorithm starts. This approach causes patterns in close proximity to be very easily merged. Being insensitive to the starting crack-network is one of the notable qualities of the L&M approach, where grouping is not affected by the starting crack-network. In a way, this approach is more strict in merging adjacent *crack patterns*.

The second stage attempts to group crack patterns based on characteristic similarities, which stem from features such as the *centroid*, *axis of minimum inertia*, *node density* and *dimension ratio*. These selected features are used as characteristics for each crack-network. An agglomerative hierarchical clustering using centroid-linkage is applied to group these features and to estimate an optimum number of clusters. A technique based on the second order differential of minimum distances was proposed.

In terms of evaluation, the effectiveness of the algorithm as a whole is not clearly visible

due to two main factors, the first being the absence of ground truth, where the level of subjectivity in accessing the successfulness of a crack-network grouping is relatively high, and secondly, the challenge of the problem itself. The level of randomness and uncertainty within the scope of craquelure analysis proves to be a huge obstacle.

However, the contribution of this chapter towards the thesis is seen as crucial, in the sense that it introduced the notion of content-based analysis using sub-image analysis, which was made possible through pattern approximations and pattern merging.

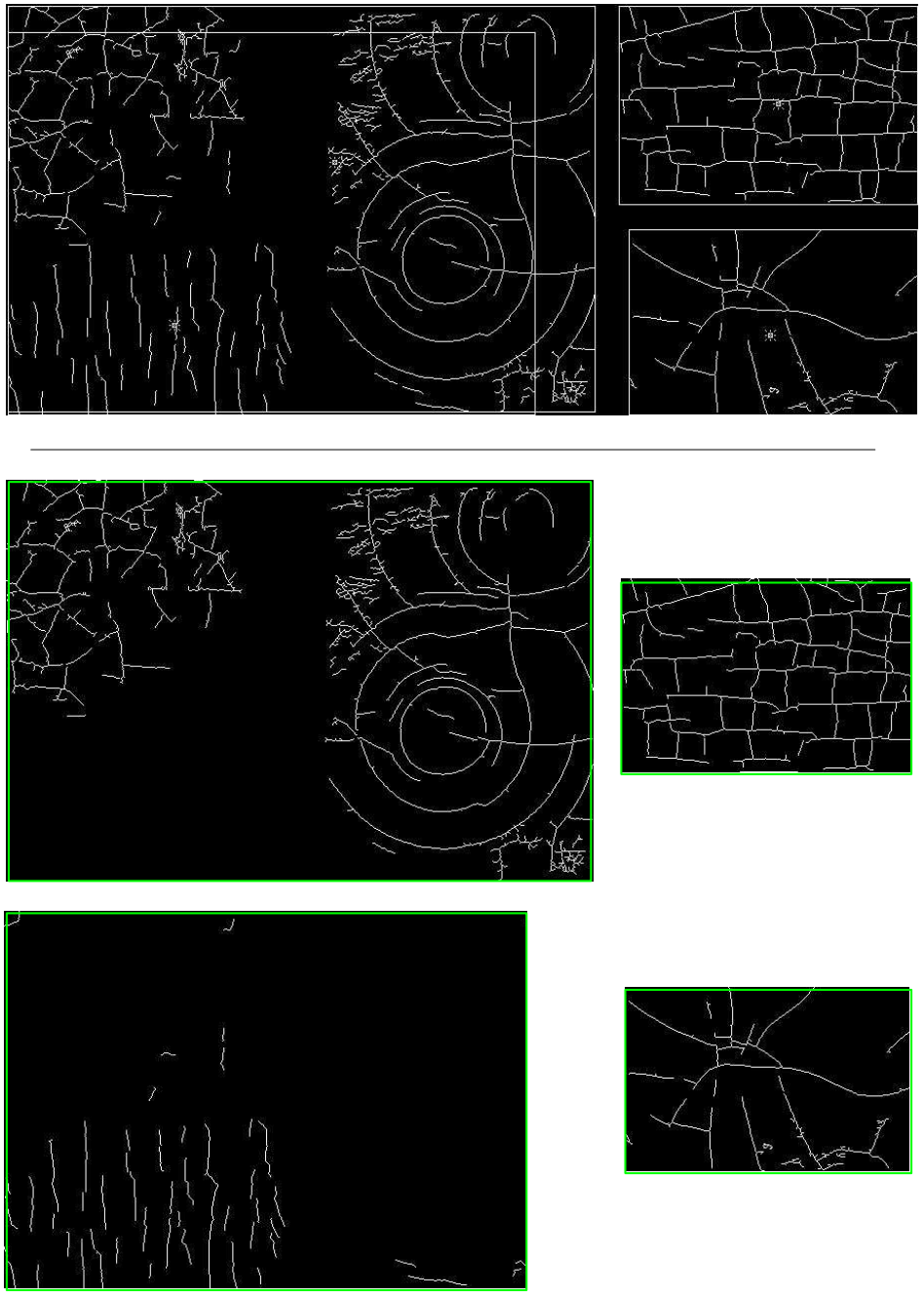


Figure 5.18: Crack pattern grouping results using L&M with the *strict area-based rule*.

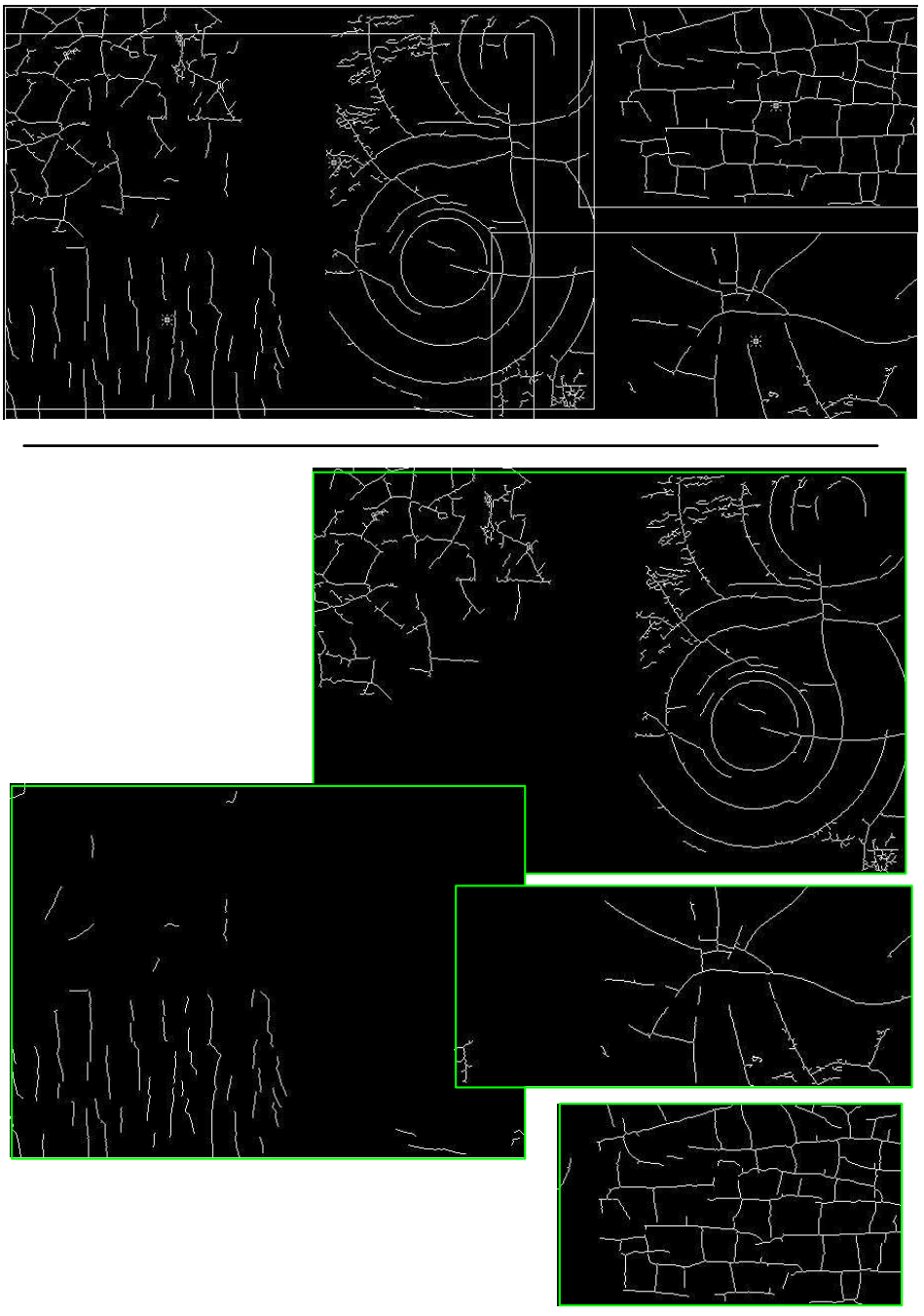


Figure 5.19: Crack pattern grouping results using L&M with the *strict logic-based rule*.

Chapter 6

Feature Extraction and Classification

6.1 Introduction

Descriptions and features cannot be considered pure knowledge representations. Nevertheless, they can be used for representing knowledge as part of a more complex representational structure. Descriptions usually represent some scalar properties of objects, and are called features [49]. Typically, a single description is insufficient for object representation; therefore, multiple descriptions are combined into what is called a *feature vector*.

The patterns of crack were described using structural descriptions, such as in [44], [46] and [149]. Bucklow [44] defined a descriptive framework of cracks based upon the following features:

1. Predominant direction and orientation of cracks
    - NO DIRECTION or DIRECTION; isotropy or anisotropy?
    - if anisotropic, then PARALLEL or PERPENDICULAR to grain?
2. Changes in direction of cracks
    - locally - SMOOTH or JAGGED
    - globally - STRAIGHT or CURVED
3. Relationship between crack directions.
    - paint islands - SQUARE or NOT SQUARE; is there an orthogonal relationship?

4. Distance between cracks.
    - spatial frequency - are the paint islands SMALL or LARGE?
5. Thickness of cracks.
    - are all cracks of UNIFORM thickness or are SECONDARY cracks present?
6. Junctions or termination of crack.
    - is crack CONNECTED or BROKEN?
7. Organization of cracks.
    - is crack network ORDERED or RANDOM?

The elaboration of the descriptions can be found in [44], where Bucklow also stressed that fewer than half of them proved to be really necessary for a very high level of discrimination between the categories used in the demonstrations. The remaining terms however enable finer discrimination within the categories. The discriminatory power of features varies and is dependent upon the characteristics of the classes sought. The characteristics of a pattern are determined not so much by individual features but by their relationships. The combination of features that best describes a crack pattern is to be ascertain.

A hierarchical approach in representing features of a crack pattern is described, that originates from a localised representation to a representation that can be acknowledged as a meaningful object-of-interest. Features used to describe crack patterns are also described in the later stages of this chapter. The resultant features are then used for classification.

6.2 Hierarchically Structured Representation of Features

Hierarchical representation of features is not a new approach in CBIR. Various researchers used the concept of representing objects in hierarchical form. Xu et al. [150] propose a system that employs a hierarchical content tree data structure for every image in a database. Similarity of content is computed by searching this representation in a top-down fashion, by first matching composite nodes (objects) if they are already formed, to reduce search time, and thereafter, matching combinations of elementary nodes if match has not been established at a higher level.

Features are hierarchically divided into four layers, namely *local layer*, *crack-network layer*, *global layer* and *image layer*. This is graphically explained in Figure 6.1. Conceptually, the

approach taken can be described as a structural scaling approach, which views objects-of-interest at hierarchically-separated yet related structural levels. The layers are organised in a data structure, so as to allow direct access to their components and to prepare a basis for further manipulation of their embedded data.

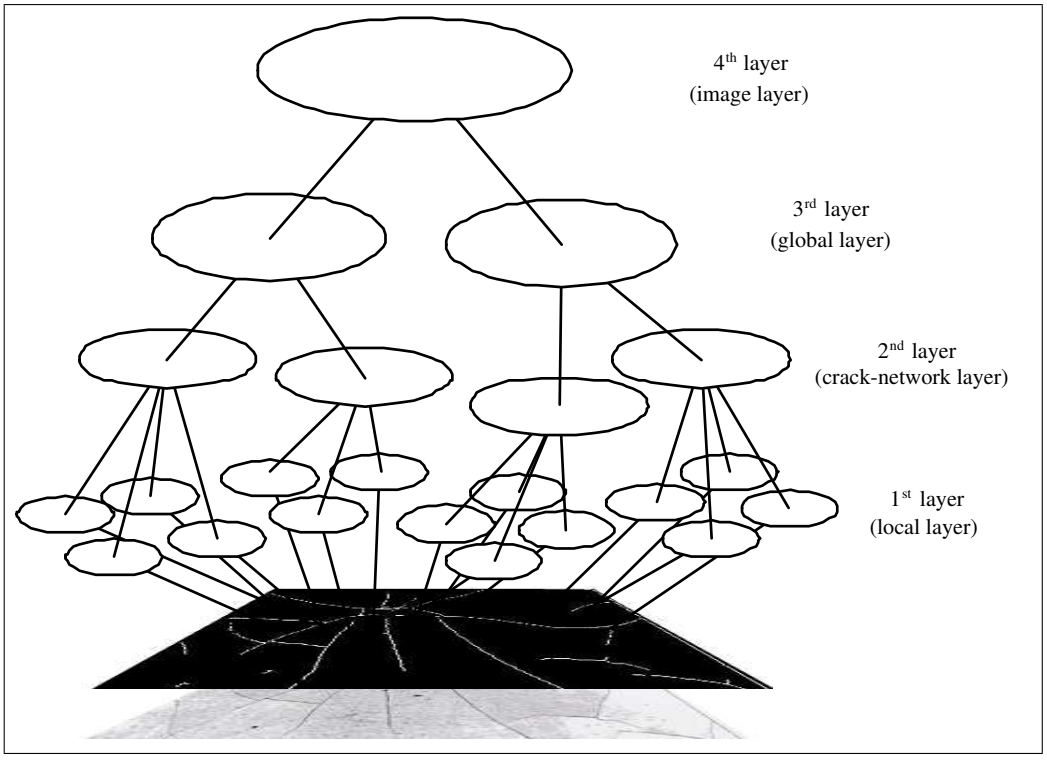


Figure 6.1: Hierarchy of features.

The first layer (*local layer*) is more like a "hidden layer", since it is not used explicitly as an object descriptor. However, its importance is quite significant, due to the fact that it functions as a "root" to enable the higher level features to "grow". This first layer of interest concentrates on fine entities, which involve line segments. The primary assumption is that a crack-network consists of multiple line segments. However, this assumption can be relaxed with crack-networks that do not contain more than a single line. In this case, there are three layers in the representation, i.e. the *local/crack-network layer*, the *global layer* and the *image layer*.

The second layer (*crack-network layer*) is formed from combinations of the first layer. Every node in the first layer has a corresponding significance value regarding the formation of a higher level. The *significance measure* is explained in Section 6.3.1. The object created at this layer is the entity of a crack pattern which is denoted as a crack-network.

Using the same procedure as for the lower layer, the *global layer* is formed from a weighted

combination of entities from the *crack-network layer*. The process of combining these entities has already been explained thoroughly in Chapter 5. It is important to note that not all features are computed using weightings or significance measures. Some features are calculated by straightforward assessment of the structural characteristics of a particular layer in most cases, the *global layer*.

The upper-most layer in the hierarchy is the *image layer*, which comprises every single entity within an analysed image.

6.2.1 The Basic Features

From a global point of view, a single crack-network holds information regarding the number of local entities detected by the *crack following* routine, where these include the number of nodes and line segments. These entities are accumulated as the crack contour is "followed". Basic features are also computed for use in higher level feature extraction processes.

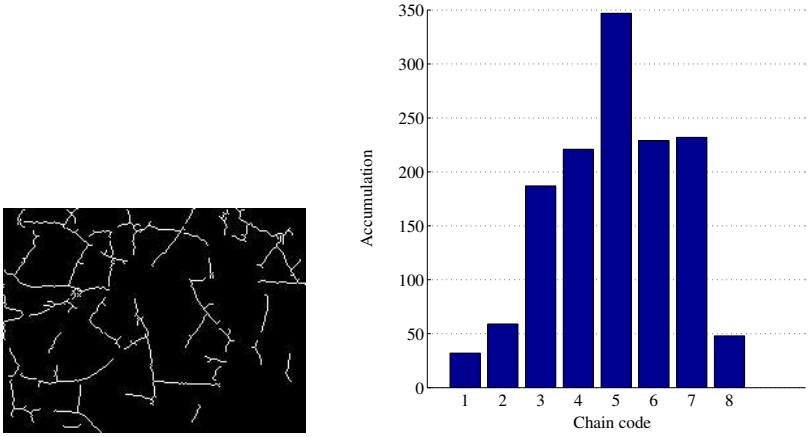
The perimeter is computed as the length of a chain [50, 151]. The formula for the perimeter is

$$P = n_e + \sqrt{2}n_o \tag{6.1}$$

where n_e is the number of even chain elements and n_o the number of odd chain elements (based on chain-code connectivity as in Figure 1.2(b)) . The total length of a crack-network length is calculated using Equation 6.1 and its significance will be explained in Section 6.3.

One of the most important items of information that needs to be captured by the *crack following* routine is line orientation. For this purpose, chain-codes are accumulated in a histogram with 8 bins, where each bin represents a chain-code. This histogram is called the orientation histogram. It roughly indicates the orientation spread of a particular crack-network. Thus, globally, it can be determined whether there are any dominant directions or whether the directions are equally spread. Figure 6.2 shows an example of orientation histograms for line segments.

Each crack-network consists of line segments and these line segments are connected by nodes. The two components (line segments and nodes) are considered as the local entities of a crack-network. The important feature of a node is its location in a crack-network. As a whole, this generally informs how concentrated or sparsely distributed nodes are in a crack-network.



(a) (b)

Figure 6.2: A crack contour (a) with its orientation histogram (b).

Line segments obtain almost the same type of information compared to the *crack-network layer*, except that they capture statistical data on a line-to-line basis. This means that each line segment has its own features. For every line segment, the edge points are marked and the length is recorded using Equation 6.1. Orientation histograms are also constructed for each line segment.

A complete crack-network data structure allows straightforward manipulation of crack pattern entities. This simple framework is useful for the later stages of extracting high-level features.

6.3 Extracting High-level Features

The next approach involves generating meaningful features to numerically describe crack patterns. The fact that cracks are represented by line segments is taken into account. The arrangements of line segments are the main factor in separating cracks into different pattern classes. Since in most cases crack patterns are formed through a combination of line segments, the approaches undertaken consider generating meaningful local features as an important starting step. These local features are then exploited to form global descriptors. Some features are highly dependent on relations between layers, while some are directly extracted at the top layer.

For the following sections, the *local*, *crack-network* and *global* layer parameters are symbolised with subscripts a, b and c respectively. The following are common terminologies and notations needed for the ensuing discussions:

(a) *Local layer* :

- n_a : total number of pixels in a line segment
- $C_a = \{c_1 c_2 ... c_{n_a - 1}\}$: a sequence of line segment chain-codes
- ℓ_a : length of a line segment
- s_a : significance measure of a line segment

(b) *Crack-network layer* :

- n_b : total number of line segments in a single crack-network
- $L_b = \{\ell_{a_{(1)}}, ..., \ell_{a_{(n_b)}}\}$: a set of line segment lengths
- ℓ_b : total length of a crack-network
- $S_b = \{s_{a_{(1)}}, ..., s_{a_{(n_b)}}\}$: a set of line segment significance measures
- s_b : significance measure of a crack-network

(c) *Global layer* :

- n_c : total number of crack-networks in a *global* layer
- m_c : total number of line segments in a *global* layer
- $L_c = \{\ell_{b_{(1)}}, ..., \ell_{b_{(nc)}}\}$: a set of crack-network lengths
- ℓ_c : total length of cracks in an image
- $S_c = \{s_{b_{(1)}}, ..., s_{b_{(nc)}}\}$: a set of crack-network significance measures

6.3.1 The Significance Measure

For every line segment, a significance value is computed that roughly tells numerically how significant a line segment is in global terms. In other words, it indicates how much a line segment contributes to the computation of a crack-network feature. In doing this, the line segment lengths, are used as an indication of their influence on the whole crack-network. The *significance measure*, s_a is computed as

$$s_{a_{(i)}} = \frac{\ell_{a_{(i)}}}{\ell_b} \quad \forall \quad i = 1, 2, ...n_b \tag{6.2}$$

where $S_b = \{s_{a_{(1)}}, ..., s_{a_{(n_b)}}\}$ is a set of significance values representing each line segment in a crack-network. For the *global layer*, the computation of S_c is

$$s_{b_{(i)}} = \frac{\ell_{b_{(i)}}}{\ell_c} \quad \forall \quad i = 1...n_c \tag{6.3}$$

where $S_c = \{s_{b_{(1)}}, ..., s_{b_{(n_c)}}\}$ is a set of significance values representing each crack-network in an image.

6.3.2 Line Segment Length as a Feature

The length of the line segments can be used as a feature, since the distribution of lengths is observed to vary over different crack types. The temporarily stored lengths of the line segments are utilised by first converting them into a measure of percentage with respect to the total length of the corresponding layer-of-interest (ℓ_b or ℓ_c), which are denoted as a set of values, $\overline{S_c} = \{\overline{s_a}1, \overline{s_a}2..., \overline{s_a}_{(m)}\}$ where m is the total number of lines in the whole *layer*. The mean and standard deviation of the percentage length distribution is then computed for a single *layer* denoted as μ_b, σ_b for the *crack-network layer* and μ_c, σ_c for the *global layer*.

The value of σ varies between 0 for a single line crack-network up to a variable maximum.

6.3.3 Straight Line to Actual Length Ratio

Another useful feature is the line length ratio (LR). Prior to the process, a straight line model of the crack patterns is constructed as shown in Figure 6.8. Straight line length $\hat{\delta}$ is defined as the Euclidean distance between two points, which can be as the following:

- node to node
- node to edge point
- edge point to edge point.

Actual length is the distance as given by Equation 6.1. LR corresponding to the *local layer* r_a is computed by taking the ratio between $\hat{\delta}$ and actual line segment length ℓ_a for all existing line segments.

Each crack-network is then assigned an LR value r_b based on the set of values obtained in the *local layer*. The same goes for the *global layer*, where a set of r_b is used to determine r_c. The computations are as shown in Equations 6.4 and 6.5.

$$r_b = \sum_{i=1}^{n_b} s_{b_{(i)}} r_a \tag{6.4}$$

$$r_c = \sum_{i=1}^{n_c} s_{c_{(i)}} r_b \tag{6.5}$$

The ratio between the direct distance and the actual distance between two points of a line segment gives a rough measure of "straightness". Locally it tells how straight the lines are. A low value generally means that a line is either circular or jagged.

6.4 Histogram-based Features

A histogram provides a frequency description of an event. For instance, the brightness/intensity histogram $h(z)$ of an image provides the frequency of the brightness value z in the image - and the histogram of an image with L grey levels is represented by a one-dimensional array with L elements. For example, an image with eight bit intensity resolution per pixel yields 2^8=256 elements in the array. The histogram is often visualised as a bar graph and it captures the global characteristics of an image. Histograms are also used for other image characteristics, such as pixel colour or object size.

Quite a significant amount of work has been done on analysing an image or a set of values based on histograms. Iivarinen and Visa [152] describe how histograms of chain-codes calculated from the chain-code of an object contour can be used to match objects. Brunelli and Mich [153] analyse the use of histograms of low level image features such as colour and luminance, as descriptors for image retrieval purposes. The issues of discrimination ability, histogram size and comparison are also touched on in their paper. Carson et al. [14] use colour histograms in the L*a*b space as one of the features for object matching. Apart from the ones mentioned, there are many fine examples of histograms used as features.

In the case of structured crack patterns, the most useful form of histogram is the orientation histogram, which stores a global description of its orientation frequency. Sub-sections 6.4.1 and 6.4.2.1 explain in detail the approach employed to extract useful properties of a crack pattern from an orientation histogram.

6.4.1 Directionality Measure

In rough observations, circular-shaped cracks and rectangular-shaped cracks differ significantly in terms of their orientation spread, as is apparent in their orientation histograms [33]. The approach taken considers this factor and extracts local orientation features which are then combined to produce more global features. The *directionality* roughly represents the straightness measure of a line segment. Firstly, local directionality values are computed for each line segment in a crack-network, using the following steps:

1. Generate orientation histograms for each line segment in a single crack-network and normalise the histograms, such that the values are within the interval [0,1] as shown by the equations below:

$$H = \{h_{a_{(0)}}, h_{a_{(1)}}..., h_{a_{(7)}}\}, \tag{6.6}$$

$$\hat{h_{a(j)}} = \frac{h_{a_{(j)}}}{\sum_{i=0}^{7} h_{a_{(i)}}} \quad \forall \quad j = 0, 1, ..., 7, \tag{6.7}$$

where H is the orientation histogram and $\hat{H}$ is the normalized orientation histogram defined as a set of accumulation values $\{\hat{h_{a(1)}}, \hat{h_{a(2)}}..., \hat{h_{a(7)}}\}$.

2. Four simple directionality histogram models are defined, where each bin represents a chain-code index of 0 to 7 respectively. The four histograms are defined by the functions

$$m_0[n] = 0.125 \sum_{j=0}^{7} \delta[n+j], \tag{6.8}$$

$$m_1[n] = 0.25 \sum_{j=0}^{3} \delta[n+j], \tag{6.9}$$

$$m_2[n] = 0.5 \sum_{j=0}^{1} \delta[n+j], \tag{6.10}$$

$$m_3[n] = \delta[n], \tag{6.11}$$

where $0 \leq n \leq 7$ and $\delta[n]$ represents a unit impulse. $m_0[n]$, $m_1[n]$, $m_2[n]$ and $m_3[n]$ represent ideal histograms as far as orientation spread is concerned, roughly indicating circular, semicircular, bidirectional and unidirectional spread, respectively.

3. Measure dissimilarity between each normalised histogram and all 4 ideal histograms. Dissimilarity corresponding to m_0 is as shown below:

$$e_0 = \sum_{j=0}^{7} \mid \overline{h_{a(j)}} - m_1[j] \mid \tag{6.12}$$

where e_0 represents error for a single histogram dissimilarity measure. A slightly different approach is taken to compute dissimilarity for m_1, m_2 and m_3, where each is circularly shifted and dissimilarities e_1, e_2 and e_3 are calculated for each rotation. The lowest dissimilarity values are taken to represent e_1, e_2 and e_3 respectively.

4. Construct a directionality histogram, $W_a = \{(2-e_0)/2, (2-e_1)/2, (2-e_2)/2, (2-e_3)/2\}$. Let $W_a = \{f_0, f_1, f_2, f_3\}$, the histogram indicates the score obtained by a particular line corresponding to each class defined in step 2. The higher the score, the more similar the orientation histogram of the line segment will be to the corresponding histogram model.

5. Find the maximum among all the values in the similarity histogram. Let k be the index of maximum similarity and $W_a[k]{=}z$ be the corresponding similarity value. Local directionality d_a is computed by using the following equation

$$d_a = \begin{cases} \dfrac{2^k - W_a[k-1] - 1}{2^k}, & \text{if} \quad k = 3 \quad \text{and} \quad \max(W_a) \neq 1 \\ \dfrac{2^{(k+1)} - W_a[k] - 1}{2^{(k+1)}}, & \text{if} \quad \text{otherwise} \end{cases} \tag{6.13}$$

Each line in a particular network has now been assigned a directionality d_a which in the interval [0,0.875].

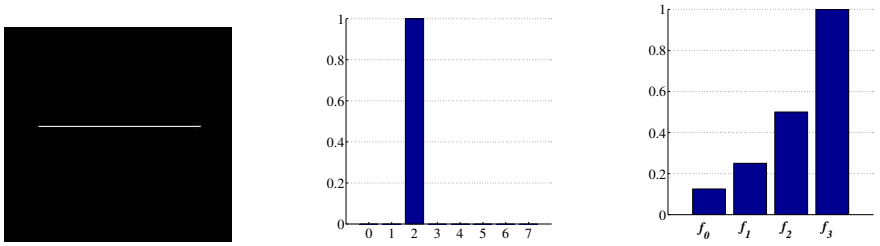
Simple examples are shown in Figure 6.3, where directionality histograms are generated for single line contours. The directionality values shown are normalised to the interval [0,1].

In the higher levels of the hierarchy, the computational options become wider. Generally, there are two approaches to calculating the directionality of a crack pattern in the crack-network and the global layer. The first approach utilises the orientation histogram of the appropriate crack pattern layer while the second technique uses significance measures to generate the directionality histogram.

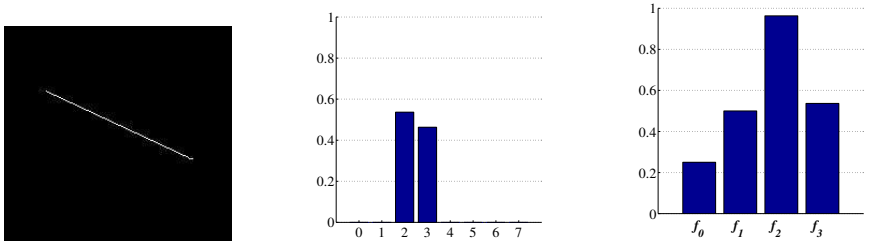
This first approach is similar to one described earlier for the computation of *local directionality*. However, in the analysis of the orientation histogram from a global view, the direction information is not relevant anymore. As an example, the chain-code '0' and '5' represent different directions, but they refer to the same orientation. Thus, in calculating global directionality, the source and destination information is ignored, unlike in the case of line segment directionality. A modification is first made to the orientation histogram by compressing it to 4 bins instead of 8, ($H = \sum_{i=0}^{3}(h_i + h_{i+4})$). Consequently, the resultant directionality histogram will be of 3 bins instead of 4.

A simplified and generalised explanation of the calculation of directionality from the orientation histogram in the crack-network and the global layers is shown in the following steps.

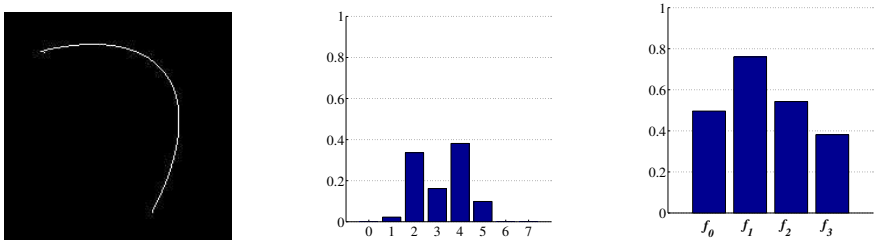
1. Normalise the orientation histogram such that the values are within the interval [0,1].



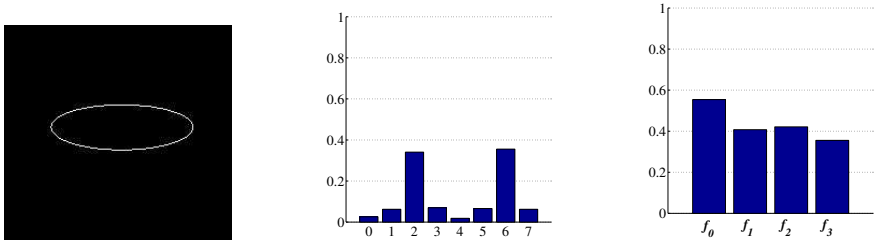
(a) $d_a = 1$.



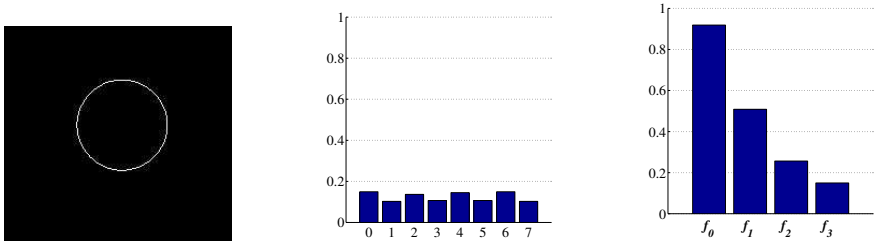
(b) $d_a = 0.8624$.



(c) $d_a = 0.64$.



(d) $d_a = 0.2548$.



(e) $d_a = 0.0464$.

Figure 6.3: Directionality in evaluating the straightness measure of five line structures is as shown here: from left to right, the line segment, the orientation histogram and the directionality histogram.

2. Define directionality histogram models to represent desired patterns.

3. Measure similarity between each normalised histogram and all model histograms.

4. Construct a directionality histogram W from the similarities.

5. Find the maximum, $\max(W)$ among all the values in the similarity histogram and locate the index k.

6. Compute score d_1 using Equation 6.13 or a generalised version in the Equation 6.14 below

$$d_1 = \begin{cases} \dfrac{2^k - W[k-1] - 1}{2^k}, & \text{if} \quad k = 3 \quad \text{and} \quad \max(W) \neq 1 \\ \dfrac{2^{(k+1)} - W[k] - 1}{2^{(k+1)}}, & \text{if} \quad \text{otherwise} \end{cases} \tag{6.14}$$

On the other hand, the second approach takes into account the significance of smaller patterns within a bigger layer. In the case of the crack-network layer, the length of each line segment plays the role of determining the measure of significance, as explained in Section 6.3.1. In general terms, the approach is explained in the following algorithm.

1. For all existing entities within the current layer, find the index k (where $k \in$[0,3]) of maximum similarity.

2. Multiply the maximum similarity by the corresponding significance measure of the particular craquelure entity.

3. Accumulate the resultant values in a new directionality histogram W of the appropriate bin, k.

4. Compute the directionality value d_2, using Equation 6.15 below:

$$d_2 = \begin{cases} \dfrac{k - W[k-1]}{3}, & \text{if} \quad k = 3 \quad \text{and} \quad \max(W) \neq 1 \\ \dfrac{k + 1 - W[k]}{3}, & \text{if} \quad \text{otherwise} \end{cases} \tag{6.15}$$

To simplify matters, the computation of directionality for a craquelure pattern is divided into two approaches. The first takes a straightforward step by only considering the orientation histogram as a means of providing knowledge of dominant chain-code direction spread, while the second approach takes the significance of a line segment within the whole

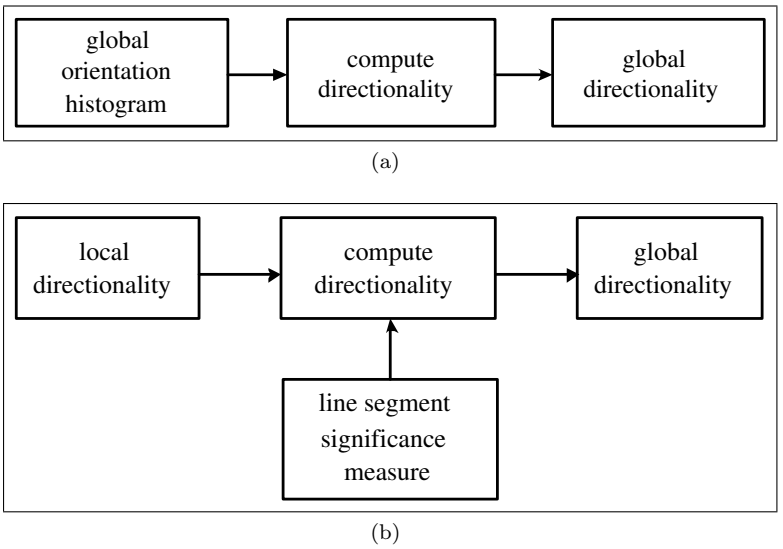


Figure 6.4: Two approaches to calculating global directionality: a) straightforward computation from the orientation histogram and b) computation using the significance measure of line segments as weightings for local directionality values.

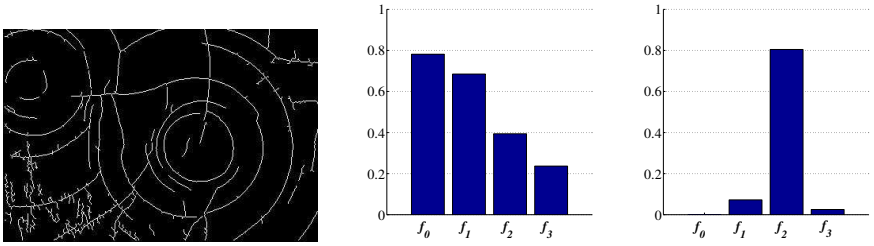
analysed craquelure pattern to determine how their directionality measures influence the directionality of the highest craquelure layer (see Figure 6.4 below for graphical explanation).

Figure 6.5 illustrates some examples of directionality measures using the first and second approaches on crack contours. For analytical purposes, for the remainder of this chapter, crack patterns in Figure 6.5(a), (b), (c), (d) and (e) are denoted as patterns A, B, C, D and E respectively. Figure 6.6 performs graphical comparison between d_1 and d_2 for patterns A, B, C, D and E. The directionality values shown are normalised to the interval [0,1].

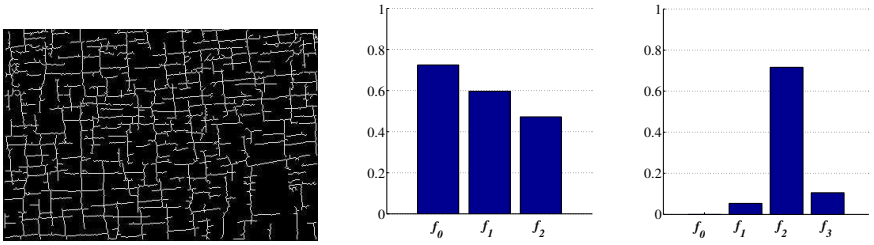
6.4.2 Straight Line Model

A regular task in image analysis is to determine the straight line that best fits a set of points. This line yields a concise representation of the set of points in terms of parameters such as end points, slope, or intercept. This methodology benefits higher level operations, such as template matching and object recognition. It is much simpler to compare objects by evaluating simplified parameters (such as slopes), rather than using raw image information to compare for similarity.

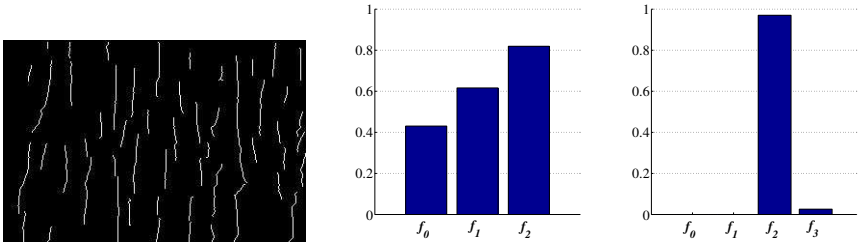
Connecting end points is the most direct strategy to fit a straight line. A more complicated strategy uses other points in the set of points, at the expense of computational cost. Least-squares line fitting is commonly used in statistical analysis, where the objective of the algorithm is to minimize the error of the fit to the line with respect to one of the line



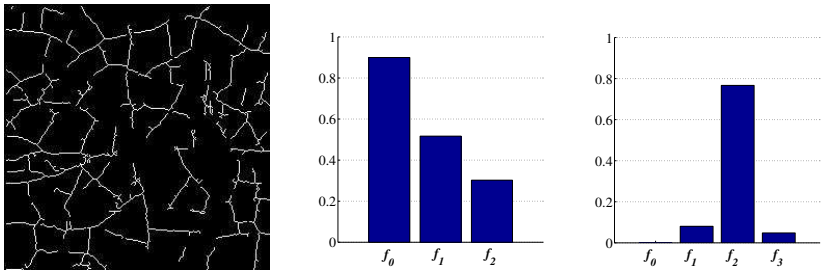
(a) $d_1 = 0.0899$, $d_2 = 0.7318$



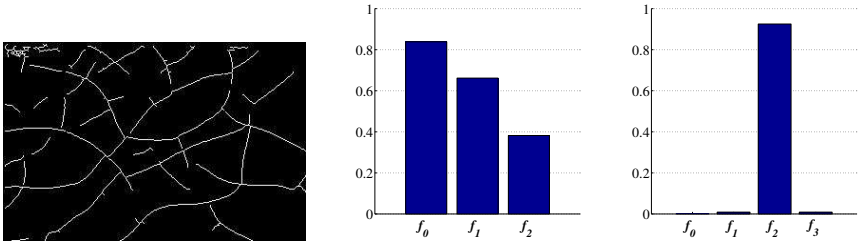
(b) $d_1 = 0.1835$, $d_2 = 0.7613$



(c) $d_1 = 0.7945$, $d_2 = 0.6766$



(d) $d_1 = 0.0668$, $d_2 = 0.7444$



(e) $d_1 = 0.1077$, d_2 =0.6916

Figure 6.5: Directionality measured on crack patterns with different patterns, namely unidirectional, random, circular, rectangular and spiderweb. Results are shown for method 1, which uses the orientation histograms directly and method 2, which utilises the significance measures as weighting for local directionality values. From left to right, crack patterns, the directionality histogram for method 1 and the directionality histogram for method 2.

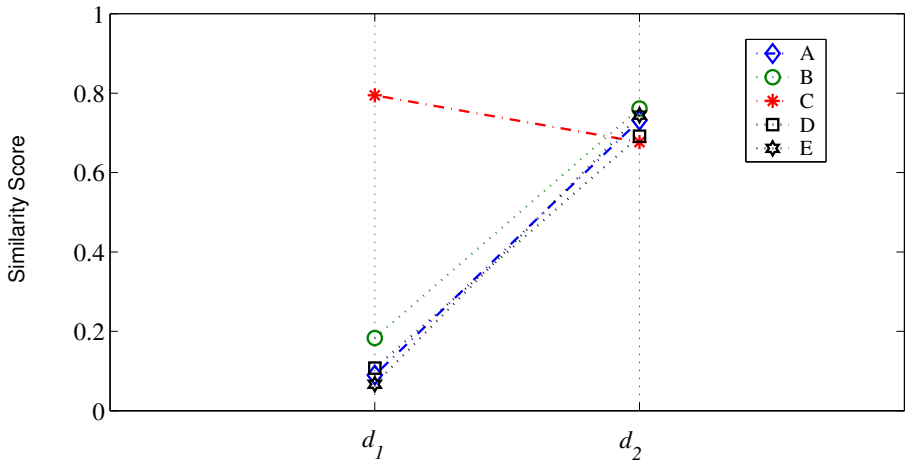


Figure 6.6: A graphical comparison between d_1 and d_2 among patterns A, B, C, D and E.

variables, y or x [57]. Another technique of fitting straight lines to a set of points is by the evaluation of eigenvectors. This technique uses eigenvector evaluation to minimise error in the direction perpendicular to the fitted line.

A straight line model of a crack pattern is implemented by constructing a direct path between two points (i.e. node to node, node to end point, end point to end point). In computing this model, the initial assumption is that for the majority of line segments, their line ratio (LR) is close to 1. For every straight line in a crack-network, a gradient is computed. Let two connected points be $p_1 = (y_1, x_1)$ and $p_2 = (y_2, x_2)$. The line gradient g between p_1 and p_2 is computed as

$$g \quad = \quad \frac{\Delta y}{\Delta x} \quad = \quad \frac{\mid y_1 - y_2 \mid}{\mid x_1 - x_2 \mid} \quad . \tag{6.16}$$

The line gradients are then quantised to an angular resolution of $\pi/16$ (see Figure 6.7). Each quantised gradient value is then accumulated in a 16 bin histogram between angles of -$\pi/2$ and $\pi/2$. In a sense, this histogram can also be acknowledged as an orientation histogram. However, to avoid confusion, this histogram is addressed as the *quantised gradient histogram* (QGH).

In some cases, especially for circular-shaped line segments, straight line models do not hold reliable information about the orientation of the line segments. The curvature details of a circular-shaped line segment are compressed to a high degree if represented by a straight line model. Thus, in this situation, the quantised gradient alone is not a good representation of the actual line segment.

A step used to counter this problem is weighted accumulation. In typical histogram con-

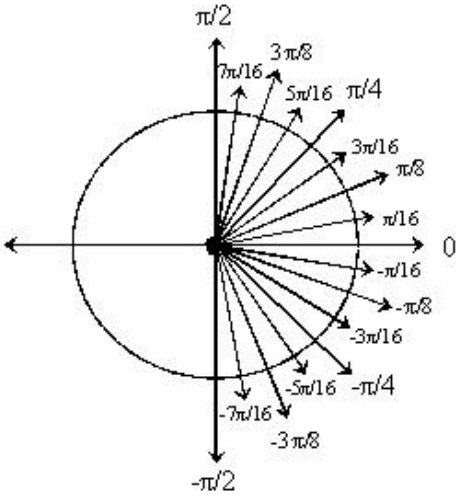


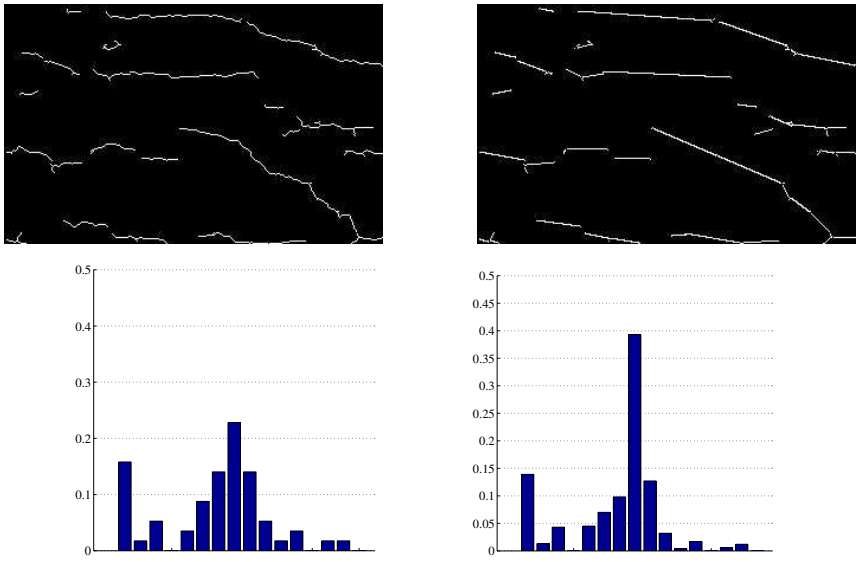
Figure 6.7: Angular resolution of $\pi/16$ is used to construct the *quantised gradient histogram* (QGH).

struction, every accumulation is an increment of one, while in a weighted histogram scheme, every singular value accumulation is multiplied by a weight. The weighting indicates the level of confidence to a particular entry of an accumulation. In this case, the weighting certifies the confidence of representing a particular line segment orientation by a quantised gradient. The significance measure of a line segment is used as a confidence value for each accumulation in the QGH. Figure 6.8 shows examples of the normal and weighted QGH.

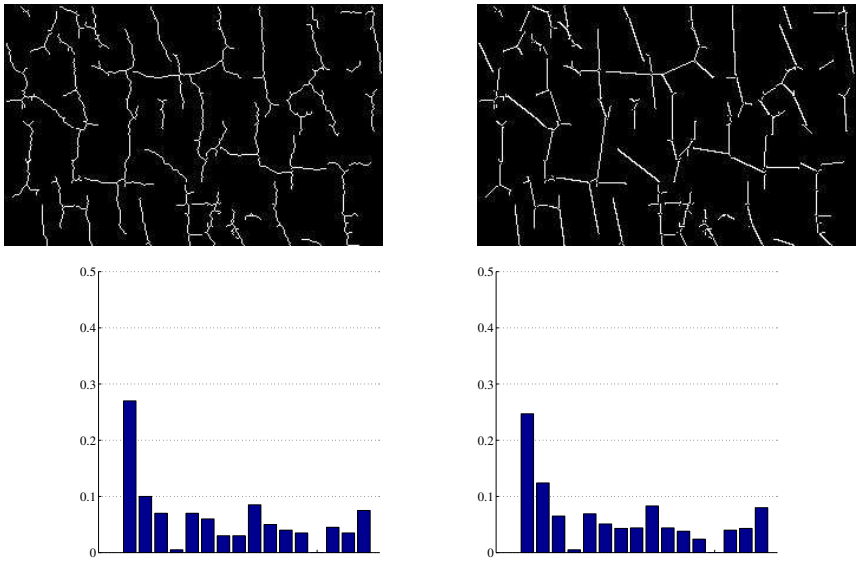
6.4.2.1 Unidirectionality and Rectangularity

Unidirectionality and rectangularity are two features based on QGH. Basically, the idea behind the two features is to extract information regarding the tendency of a crack pattern orientation histogram towards being unidirectional or rectangular in structure.

From a histogram of quantised orientation, the same procedure as explained in Section 6.4.1 is employed to calculate unidirectionality u and rectangularity r. The 16 bin QGH is first compared with "ideal" histogram models. The histograms for rectangularity are defined by Functions 6.17, 6.18, 6.19 and 6.20,



(a)



(b)

Figure 6.8: Modelling a crack pattern using straight line representation. The gradient of each line is quantised using an angular resolution of $\pi/16$. QGH is constructed using normal accumulation and weighted accumulation. Figure showing two sets of examples. From left to right and top to bottom, the original crack pattern, straight line representation, normal QGH and weighted QGH respectively.

$$m_0[n] = 0.0625\left(\sum_{j=0}^{15}\delta[n+j]\right), \tag{6.17}$$

$$m_1[n] = 0.125\left(\sum_{j=0}^{3}\delta[n+j]+\sum_{j=8}^{11}\delta[n+j]\right), \tag{6.18}$$

$$m_2[n] = 0.25\left(\sum_{j=0}^{1}\delta[n+j]+\sum_{j=8}^{9}\delta[n+j]\right), \tag{6.19}$$

$$m_4[n] = 0.5\left(\delta[n]+\delta[n+8]\right), \tag{6.20}$$

and the following functions for unidirectionality,

$$m_0[n] = 0.0625\sum_{j=0}^{15}\delta[n+j] \tag{6.21}$$

$$m_1[n] = 0.125\sum_{j=0}^{7}\delta[n+j] \tag{6.22}$$

$$m_2[n] = 0.25\sum_{j=0}^{3}\delta[n+j] \tag{6.23}$$

$$m_3[n] = 0.5\sum_{j=0}^{1}\delta[n+j] \tag{6.24}$$

$$m_4[n] = \delta[n], \tag{6.25}$$

where $0 \leq n \leq 7$.

Next, dissimilarities (four dissimilarities for rectangularity and five dissimilarities for unidirectionality) are computed between the QGH and the ideal histograms (four histograms for rectangularity and five histograms for unidirectionality). e_0 is computed directly by taking the summation of differences of each bin. e_1, e_2 and e_3 (an additional e_4 for unidirectionality) are computed by taking their minimum dissimilarities after summing up the differences between circularly shifted ideal histograms and the input QGH.

Similarity histograms are constructed for rectangularity $R = \{(2-e_0)/2, (2-e_1)/2, (2-e_2)/2, (2-e_3)/2\} = \{f_0, f_1, f_2, f_3\}$, and unidirectionality $U = \{(2-e_0)/2, (2-e_1)/2, (2-e_2)/2, (2-e_3)/2, (2-e_4)/2\} = \{f_0, f_1, f_2, f_3, f_4\}$. Let k be the index of maximum similarity and z be the maximum value of R and U. Rectangularity r and unidirectionality u are computed using the scoring function as denoted by Equation 6.13. Similar to the directionality measure, there are two available options in accumulating the quantised gradient, namely by

using normal accumulation and weighted accumulation. For analysis, unidirectionality and rectangularity, calculated using normal accumulation, are denoted by u_1 and r_1. On the other hand, u_2 and r_2 signify unidirectionality and rectangularity calculated using weighted accumulation.

Table 6.1 shows u_1, u_2, r_1, r_2 measures for the patterns A, B, C, D and E. Figure 6.9 presents the comparisons in a graph.

	u_1	u_2	r_2	r_2
A	0.1662	0.1556	0.178	0.1678
B	0.2846	0.9688	0.9151	0.9154
C	0.9473	0.9451	0.9217	0.9250
D	0.1412	0.1311	0.1513	0.1405
E	0.1710	0.1359	0.1832	0.1456

Table 6.1: Table showing comparisons of u_1, u_2, r_1, r_2 among the patterns A, B, C, D and E.

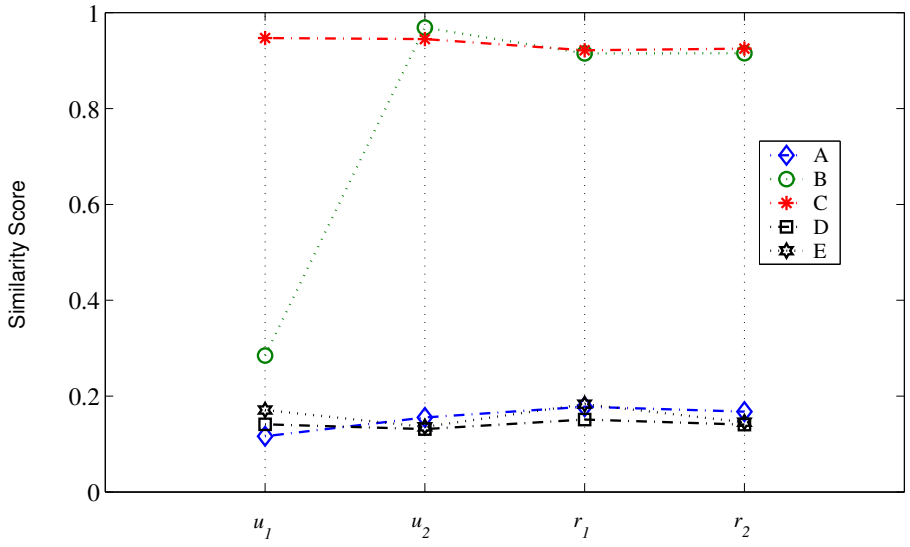


Figure 6.9: A graphical comparison of u_1, u_2, r_1, r_2 among patterns A, B, C, D and E.

6.4.3 The Histogram Shape Filter Set

A much simpler way to extract relevant information from a histogram is by using a "filter set", which comprises ideal histograms tuned towards very specific histogram shapes. The response of an input histogram towards the ideal filters generates a vector of values indicating the matching level between each filter with the input. The end product is a vector of matching scores which can be used as a feature vector as a whole or even separately.

Implementation-wise, the filter set is based on the histograms generated from the orientation histogram (see Section 6.2.1) and also the straight line representation (see Section 6.3.3). Three histograms are constructed to model the orientation histogram,

$$m_0 = \{1, 1, 1, 1\},$$
$$m_1 = \{1, 0, 1, 0\}, \text{ and}$$
$$m_2 = \{1, 0, 0, 0\}$$

while another three are used as a model for the QGH

$$m_3 = \{1, 1, 1, 1, 1, 1, 1, 1\},$$
$$m_4 = \{1, 0, 0, 0, 1, 0, 0, 0\}, \text{ and}$$
$$m_5 = \{1, 0, 0, 0, 0, 0, 0, 0\}.$$

The QGH is first compressed to 8 bins by adding up bins in pairs.

The 4 bin orientation histogram is then compared with m_0, m_1 and m_2 to establish the similarity measures f_0,f_1 and f_2 respectively. Similarly, the 8 bin QGH is then compared with m_3, m_4 and m_5 to produce similarity measures f_3,f_4 and f_5 respectively. The similarity values are then normalised between the values 0 and 1 by using Equation 6.26 below.

$$F = \{f_0, f_1, f_2, f_3, f_4, f_5\} =$$
$$\left\{ \frac{(f_0 - 0.25)}{0.75}, \frac{(f_1 - 0.5)}{0.5}, \frac{(f_2 - 0.25)}{0.75}, \frac{(f_3 - 0.125)}{0.875}, \frac{(f_4 - 0.25)}{0.75}, \frac{(f_5 - 0.125)}{0.875} \right\} \quad (6.26)$$

Figure 6.10 illustrates a comparison of F among the patterns A, B, C, D and E, while Table 6.2 displays the numerical values.

	f_0	f_1	f_2	f_3	f_4	f_5
A	0.8202	0.2032	0.1798	0.7477	0.235	0.1079
B	0.6329	0.5506	0.2957	0.423	0.6723	0.3926
C	0.2490	0.0094	0.7591	0.1894	0.3333	0.5427
D	0.8664	0.2004	0.0699	0.822	0.1973	0.1325
E	0.7845	0.0787	0.176	0.7563	0.0745	0.1003

Table 6.2: Table showing comparisons of F among the patterns A, B, C, D and E.

6.5 Features From Structural Statistics

Global statistical measures such as point density and distribution can be useful to model the behaviour of points in a given space. Quite generally, it is often useful to distinguish

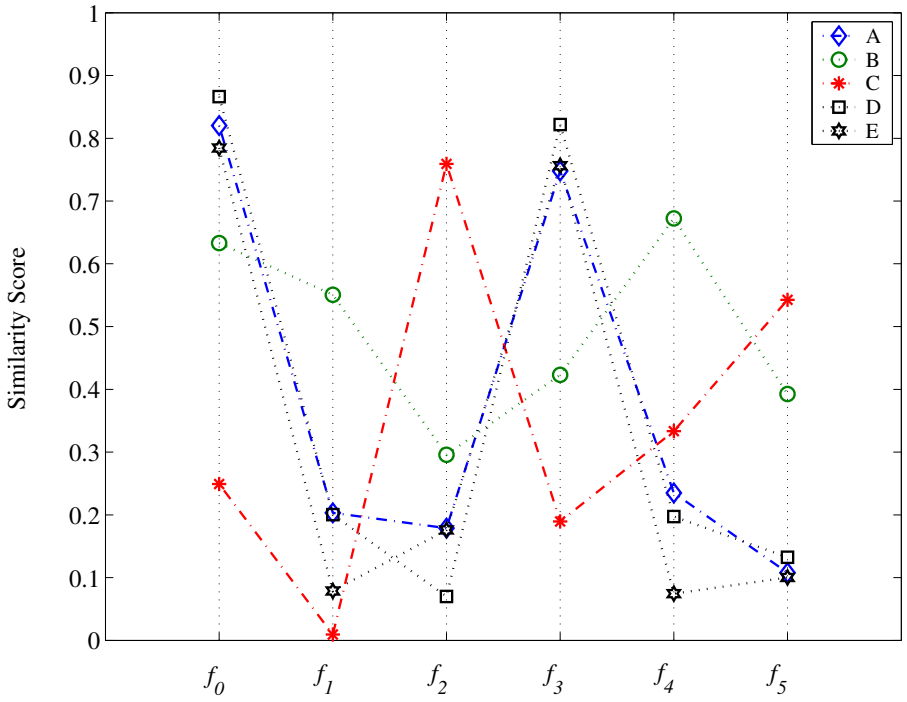


Figure 6.10: A graphical comparison of F among patterns A, B, C, D and E.

random and non-random patterns [57].

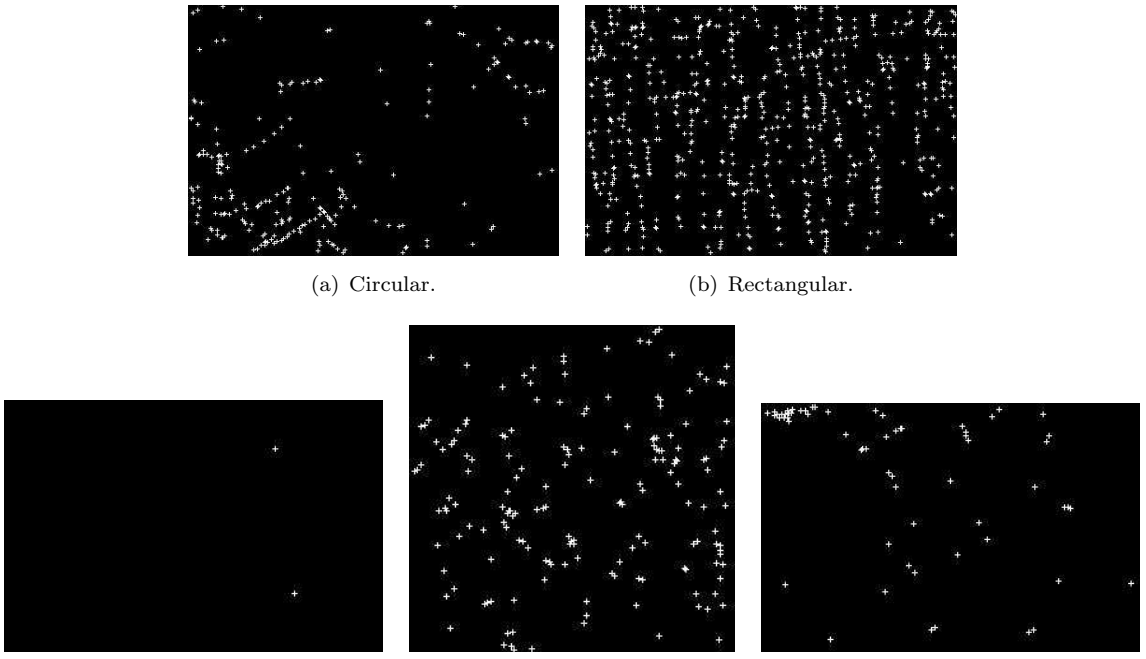
The statistics for the spatial distribution of structural features (nodes and line segments) may reveal information about the pattern they belong to.

One of the statistics of pattern nodes (see Figure 6.11) that may be interesting to observe is their population within a specified spatial boundary. Population can be assessed in terms of density with respect to several elements. Firstly, it can be based on population per pattern, which can be represented by the total number of crack pixels or total crack length. Secondly, it can also be based on the area of an approximation, which is either an MBR or an RMBR, depending on suitability. This measure indicates denseness of population for a pattern under observation.

The distribution of line segments within a crack pattern also has good potential as a feature. On rough observation, among the five classes, the unidirectional and circular crack patterns, for instance, will produce lower node and line segment population density compared to the rectangular and spider web crack patterns.

After intensive observations and trials, the following statistical measures are included into the shortlist of features related to structural statistics.

- The number of nodes per square root of crack pattern length, s_1



(a) Circular. (b) Rectangular.

(c) Unidirectional. (d) Spiderweb. (e) Random.

Figure 6.11: Distribution of nodes for patterns A, B, C, D and E.

- The number of line segments per square root of crack pattern length, s_2
- The number of line segments per number of nodes, s_3
- The standard deviation of the length of line segments, s_4

Figure 6.12 compares of s_1, s_2, s_3, s_4 among the patterns A, B, C, D and E, while Table 6.3 displays the numerical values.

	s_1	s_2	s_3	s_4
A	6.6114	2.9155	2.2677	23.1306
B	13.9217	6.377	2.1831	6.8120
C	1.3895	0.0651	21.3333	23.2689
D	7.2637	3.1509	2.3053	8.2248
E	3.4592	1.3334	2.5942	24.2853

Table 6.3: Table showing comparisons of s_1, s_2, s_3, s_4 among the patterns A, B, C, D and E.

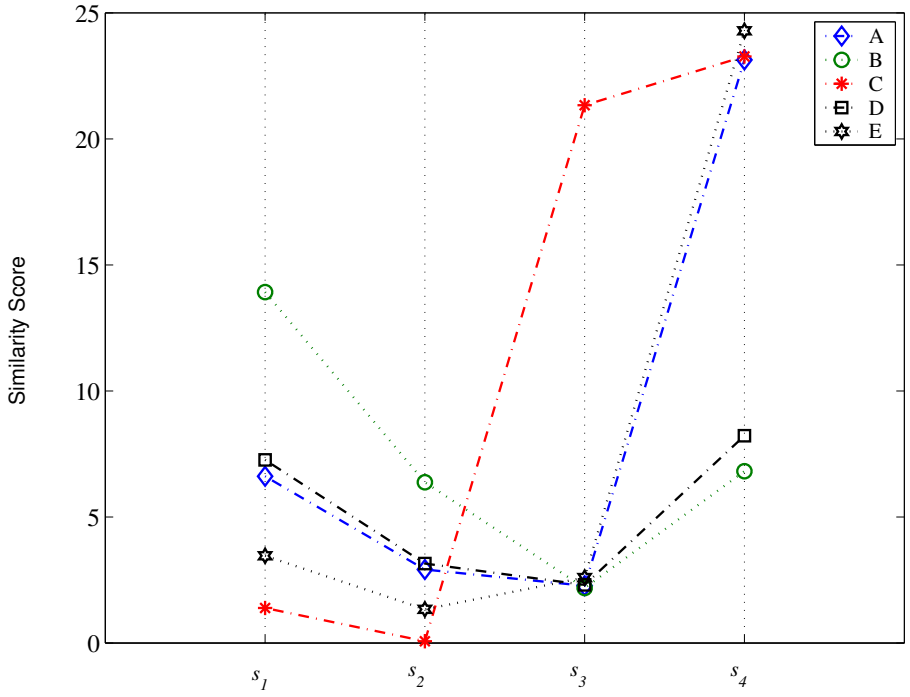


Figure 6.12: A graphical comparison of s_1, s_2, s_3, s_4 among patterns A, B, C, D and E.

6.6 Classification of Craquelure Patterns

The aim of the work in this thesis is to solve a content-based issue of classifying craquelure patterns with variable dimensions into 5 classes, namely circular, rectangular, unidirectional, spiderweb and random. The work presented so far concentrated on extracting information out of craquelure patterns and defining a solution for the interpretation of object-of-interest. Now, attention is turned to the problem of classification using the features that have been discussed earlier.

Related work was reported in [38], in which Varley classified a set of 40 images of the paint layer of old oil paintings showing the patterns of crack. The approach was motivated by the work of Bucklow [154], in which classification of crack images was based on subjective scores given by a panel of human subjects. Varley approached the problem by fitting a Bézier model to a crack image [36]. The data produced from this fitted model was huge (variable length) with each curve recording parameters of position, magnitude and gradient of the tangent vectors of the curve, together with the width of the curve. The number of parameters produced in the end was far too great for classification to be performed, even though the number of pixels was even greater. Varley later carried out feature selection using the Fisher Ratio [155] to extract the most relevant discriminatory features, with each feature given a score based on how well it discriminated crack images of different patterns. Varley then made classification using four top ranked features using a two-stage

approach, the linear discriminant classifier [139] followed by the *expectation maximisation* (EM) algorithm [156].

By definition, classification involves assigning an input pattern to one of the pattern classes (specified beforehand) under consideration, based on measured features [157]. In statistical classification, the decision making process can be summarised as follows: a given pattern is to be assigned to one of c categories $y_1,y_2,...,y_c$ based on a vector of d feature values $\boldsymbol{x} = (x_1,x_2,...,x_d)$.

The main problem in classification usually revolves around choosing the correct classifier, and, in practice, the choice of classifier is often based on which classifier(s) happen to be available, or best known to the user [157]. It can also be chosen according to the nature of information available to the user about the problem in hand. Jain et al. [157] presented various dichotomies that appear in statistical pattern recognition (see Figure 6.13). Descending the tree from top to bottom and traversing from left to right, less knowledge is presented about the data set to be classified, hence the classification problem increases.

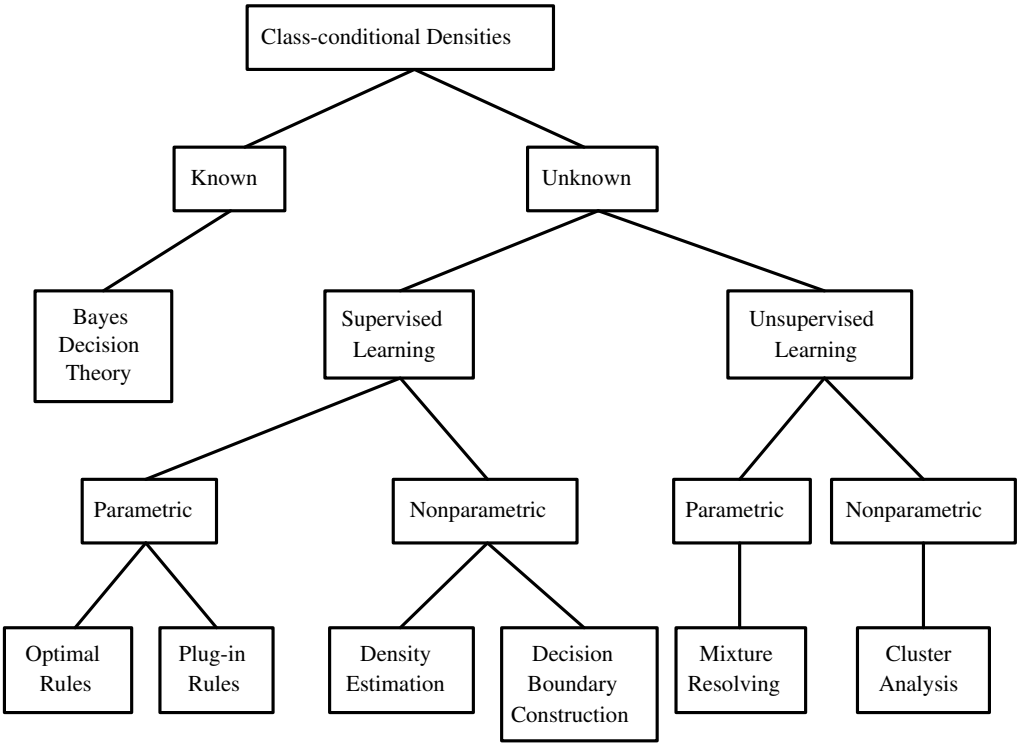


Figure 6.13: Various approaches in statistical pattern recognition.

The dichotomies as stressed by Jain et al. first of all depend on the availability of any class-conditional densities. The design of classifiers strives to integrate all available prob-

lem information, such as measurements, and also *a priori* known probabilities [139]. Class-conditional densities require the knowledge of $p(\boldsymbol{x}|y_i)$ and $P(y_i)$ for each class. In addition, information needs to be acquired about the class-specific means (μ_i), covariance matrices (Σ_i) and so on, for i = 1,2,...,c. A distribution *form* such as Gaussian and uniform distributions must also be specified. Frequently, these are not available.

If all of the class-conditional densities are available, then the optimal Bayes decision rule [139] can be used. However, in most situations, the class-conditional densities are not known and must be learned from the available training patterns. Here, two options are available, supervised (labelled training samples) or unsupervised (unlabelled training samples) learning, where the labelling determines to which class a particular sample belongs.

Within supervised learning, another dichotomy in statistical pattern recognition is based on parametric or nonparametric approaches. In both cases, a set of training samples for each underlying class is required, with each sample labelled as to its correct class. The task in hand is to learn from these training samples. In the parametric approach, a specific form (e.g Gaussian, uniform) for the distributions is required as well as estimates of the parameter values (i.e. $p(\boldsymbol{x}|y_i)$, $P(y_i)$, μ_i, Σ_i etc.) from the training set by using estimation algorithms such as the *maximum likelihood* (ML) estimation and the *Bayesian estimation* [139]. The inability to determine a specific form for the distributions and the incompatibility between the chosen form and the estimated parameters are the reasons that prohibit the usage of parametric approaches. The most likely solution to this is to resort to nonparametric learning techniques. The two most well-known nonparametric learning techniques are the k-nearest neighbour (k-NN) classifier and the Parzen classifier [139]. In practice, both of these approaches require specification of only one parameter, the number of neighbours k for k-NN and the smoothing parameter of the Parzen kernel.

The unsupervised learning approach on the other hand is used in a situation where samples are available, but unlabelled. The need to continuously learn and adapt to changes in the characteristics of the class-specific pattern generating systems is a reasonable reason for unsupervised learning. In this category, a mixture resolving technique [157] and cluster analysis [138, 157] are the two widely used approaches. The main problem with cluster analysis is the inability to determine the number of clusters beforehand, thus making clustering more tuned in with finding reasonable categorisation of data, if available. There are quite a number of algorithms for finding the optimum number of clusters, such as those reported in [158], [159] and [160], but these algorithms are still highly dependent on good choices of parameter values.

A summary of classification techniques is shown in Table 6.4 [157]. The choice of classifiers

basically depends on the nature of the problem in hand and the amount of knowledge available regarding the data to be classified. No particular classification strategy, no matter how simple or straightforward can be ruled out, and sophistication does not guarantee success in a pattern recognition problem. An in-depth study of a large set of classifiers was carried put by Michie et al. [161] over many different problems, and the study demonstrated large variability over many performances. In general terms, this demonstrates that there is no such thing as an optimal classification rule.

Experiments are conducted with various classification approaches throughout this study, taking into consideration the resources (test images, prior knowledge about the data set). These approaches included the k-means clustering, fuzzy k-means clustering, hierarchical agglomerative clustering, artificial neural networks (ANN) (feed-forward multi-layer perceptron with back-propagation learning) and the k-nearest neighbour technique. It is important to stress that the clustering-based techniques (k-means, fuzzy k-means and hierarchical agglomerative clustering) are not fully functional as classifiers since their end result is not categorisation of patterns into classes but merely categorisation of patterns into distinct groups with no direct way of associating them to particular classes. In order to make classifications using these techniques, an extended methodology is needed to associate the clustered data with some previously determined class template or map.

For implementation, no prior knowledge is assumed available concerning the nature of the craquelure data set, including labelled patterns. In these circumstances, cluster analysis or mixture resolving are the most appropriate approaches. In this implementation, clustering-based processes are carried out on image-by-image basis, which means that each process will produce b number of clusters, with $b = 1,2,...,d$ where d is the number of sample points (objects-of-interest) in the image. Before the process, there is no prior knowledge as to how many clusters are expected, thus, a "guess" or automatical specification of the numbers is required. However, before proceeding further, the thought of not having any labelled patterns makes classification using clustering-based techniques impossible, since there are no benchmarks to mark each class. An initial classification effort using fuzzy k-means clustering is demonstrated in [34].

With the presence of training images, more flexibility is available in choosing a classification strategy. Representatives for each craquelure class are collated and labelled. In the presence of labelled data samples, classification strategy is based on the k-NN rule which seems to be a simple and straightforward approach.

Technique	Properties	Comments
Template Matching	Assigns patterns to the most similar template.	The template and the metric have to be supplied by the user; the procedure may include nonlinear normalisations; scale (metric) dependent.
Nearest Mean Classifier	Assigns patterns to the nearest class mean.	Almost no training needed; fast testing; scale (metric) dependent.
Subspace Method	Assigns patterns to the nearest class subspace.	Instead of normalising on invariants, the subspace of the invariants is used; scale (metric) dependent.
1-Nearest Neighbour Rule	Assigns patterns to the class of the nearest training pattern	No training needed; robust performance; slow testing; scale (metric) dependent.
k-Nearest Neighbour Rule	Assigns patterns to the majority class among k nearest neighbour using a performance optimised value for k.	Asymptotically optimal; scale (metric) dependent; slow testing.
Bayes Plug-in	Assigns patterns to the class which has the maximum estimated posterior probabilities.	Yields simpler classifiers (linear or quadratic) for Gaussian distributions; sensitive to density estimation errors.
Logistic Classifier	Maximum likelihood rule for logistic (sigmoidal) posterior probabilities.	Linear classifier; iterative procedure; optimal for a family of different distributions (Gaussian); suitable for mixed data types.
Parzen Classifier	Bayes plug-in rule for Parzen density estimates with performance optimal kernel.	Asymptotically optimal; scale (metric) dependent; slow testing.
Fisher Linear Discriminant	Linear classifier using mean square error (MSE) optimisation	Simple and fast; similar to Bayes plug-in for Gaussian distributions with identical covariance matrices.
Binary Decision Tree	Finds a set of thresholds for a pattern-dependent sequence of features.	Iterative training procedure; overtraining sensitive; needs pruning; fast testing.
Perceptron	Iterative optimisation of a linear classifier	Sensitive to training parameters; may produce confidence values.
Multi-layer Perceptron (Feed-forward Neural Network)	Iterative MSE optimisation of two or more layers of perceptrons (neurons) using sigmoid transfer functions.	Sensitive to training parameters; slow training; nonlinear classification functions; may produce confidence values; overtraining sensitive, needs regularisation.
Radial Basis Network	Iterative MSE optimisation of a feed-forward neural network with at least one layer of neurons using Gaussian-like transfer functions.	Sensitive to training parameters; nonlinear classification functions; may produce confidence values; overtraining sensitive, needs regularisation; may be robust to outliers.
Support Vector Classifier	Maximises the margin between the classes by selecting a minimum number of support vectors.	Scale (metric) dependent; iterative; slow training; nonlinear; overtraining insensitive; good generalisation performance.

Table 6.4: Summary of classification techniques.

6.7 The Nearest Neighbour Rule

The nearest neighbour (NN) rule is one of the extensions of a suboptimal nonparametric classification approach, which is widely used, mainly due to its simplicity and straightforward implementation, requiring a minimum number of parameter value specifications.

The 1-NN classifier introduced by Cover and Hart [162] assigns an input sample pattern $\boldsymbol{x}$ to the class of its nearest neighbour, where the term "nearest" lies upon the user's specification, based on the various distance metrics, as explained in Section 5.6.2.4 (Euclidean distance is commonly used). The 1-NN classifier can always be a benchmark for other classifiers, since it appears to provide reasonable classification performance in applications. Furthermore, the 1-NN classifier only needs one user-specified parameter, which is the distance metric. However, with this simple rule, it is prone to errors, but has been no more than twice the Bayes error rate [163].

As an extension to the 1-NN, k-Nearest Neighbour (k-NN) [162] is introduced, with the number of neighbours extended from 1 to k. In the k-NN rule, the input pattern represented by the majority of the k-nearest neighbours, is assigned. By considering more than one neighbour, with each neighbour giving the same contribution to the final decision, there is a possibility of a tie among maximum-voted classes. A way to resolve this problem is to decrease the value of k by one and perform another run of the k-NN until no tie occurs. Another simple solution is to search for classes with the minimum sum of distances to each k neighbour.

Dudani [164] extended the k-NN rule into what is called the weighted k-NN rule, where the distance information determines the weight factor w_j and j is the total number of samples. The idea behind the weighted k-NN is to give each sample a different amount of influence on the decision process, such that nearby samples have more influence. Letting d_j be the distance between input pattern $\boldsymbol{x}$ and a sample $\boldsymbol{x}_j$, w_j computed such that the nearest neighbour's weight equals one and the k-th nearest neighbour becomes zero, as explained by Equation 6.27. The weighted k-NN decreases the probability of getting a tie in the voting process.

$$w_j = \begin{cases} \dfrac{d_k - d_j}{d_k - d_1}, & \text{if} \quad d_k \neq d_1 \\ 1, & \text{if} \quad d_k = d_1 \end{cases} \tag{6.27}$$

One technique to completely solve the tie problem in the k-NN rule is to incorporate fuzzy set theory [165] into the implementation. Keller et al. [166] enhanced the k-NN rule by

introducing the fuzzy k-NN. Unlike the k-NN, which produces "hard" (one pattern belongs to one class) class assignment, the fuzzy k-NN implements a "soft" (one pattern belongs to multiple classes) class assignments by assigning class memberships to an input pattern $\boldsymbol{x}$. The advantage of fuzzy k-NN is that no arbitrary assignments are made, with pattern $\boldsymbol{x}$ having membership to a certain degree to all c classes. The membership values in a way indicate an assurance measure to the resultant classification. Consequently, the assurance level can be utilised further, for instance as a high-level pattern descriptor, as demonstrated later.

In terms of implementation, the initial assignment of the membership value of each sample pattern is very important, producing more informative samples. The classification is more precise with more accurate initial membership values. The first step in the algorithm is to find the K-nearest neighbour to each labelled sample $\boldsymbol{x}$ in every class. K here is the number of neighbours considered in this initial fuzzy membership assignment, which is not necessarily the same as k in the classifier. With c number of classes, let n_i be the number of samples that belong to class y_i, where $\sum_{i=1}^{c} n_i = K$. Membership in each class is assigned according to the following equation:

$$\hat{u}_i(\boldsymbol{x}) = \begin{cases} 0.51 + \left(\frac{n_i}{K}\right) * 0.49, & \text{if} \quad i = j \\ \left(\frac{n_i}{K}\right) * 0.49, & \text{if} \quad i \neq j \end{cases} \tag{6.28}$$

where n_i is the number of neighbours belonging to the i_{th} class and j is the class in which $\boldsymbol{x}_i$ is labelled, while i = 1,2,...,c. After assigning the initial membership grade for the input pattern, the final membership value for the input pattern is determined by using the distance difference from the k nearest sample patterns and their initial membership values according to the following equation:

$$u_i(\boldsymbol{x}) = \frac{\sum_{j=1}^{k} \hat{u}_i(x_j) \left(\frac{1}{||x - x_j||^{\frac{2}{m-1}}} \right)}{\sum_{j=1}^{k} \left(\frac{1}{||x - x_j||^{\frac{2}{m-1}}} \right)} \tag{6.29}$$

where m is a scaling factor between 1 and 2.

Several methodologies for integrating "fuzziness", mainly using distance information as a cue to determine relatedness between the input pattern and the sample. In principle, the techniques implemented revolve around the same concept as the famous weighted k-NN and the fuzzy k-NN techniques. In its simplest form, "fuzziness" to an output of a k-NN

classifier can be determined using Equation 6.30 which shows how a membership function to an input pattern $\boldsymbol{x}$ is calculated for i = 1,2,...,c.

$$u_i(\boldsymbol{x}) \quad = \quad \frac{n_i}{k}. \tag{6.30}$$

Thus, in its simplest form, the input pattern x is assigned "fuzzy" values for $u_i(\boldsymbol{x})$ for all classes c. Higher values of k increase the fuzziness of the output, since it is more likely to include the participation of various class representative points in the membership determination.

It is quite harsh to exclude distance from membership determination, since distance in a way indicates similarity and its exclusion means only the quantity of representatives within k-voted samples are taken as cue.

A technique named the *distance weighted* k-NN which uses distance as a cue for classification is also implemented, taking the summation of distances between the input point $\boldsymbol{x}$ and all of its k nearest neighbours, denoted by d_T.

Next, the total distances of similar-class points are calculated, where the number of distances depends on the number of classes out of c classes which exist within the k nearest neighbours. Thus, it ends up with m number of distances d_m, where m is the number of classes out of c ($m \leq c$) which exist within the k nearest neighbours and $j = 1, ..., m$. From here, it can be seen that $\sum_{j=1}^{m} d_j = d_T$. To make the explanation clearer, let $\boldsymbol{i}$ be a set of all available classes and $\boldsymbol{j}$ be the set of classes within the k-NN domain of $\boldsymbol{x}$. The membership of the input point $\boldsymbol{x}$ within $\boldsymbol{i}$ is calculated by using the following equation:

$$u_i(\boldsymbol{x}) = \begin{cases} \dfrac{d_T - d_i}{d_T(m-1)}, & \text{if} \quad m > 1 \\ 1, & \text{if} \quad d_T = d_i \\ 0, & \text{if} \quad i \notin \boldsymbol{j} \end{cases} \cdot \tag{6.31}$$

This method in a way uses only distance as a determining factor in constructing the membership vector of a pattern, where the dominance of a class in the k-NN domain is determined by the accumulation of distances. In an instance where an input is very close to numerous class A samples, while reasonably far to a single class B sample, the accumulation of distances will make the input more likely to be classified as B rather than A. From here, it is quite evident that quantity also plays an important part within the k-NN domain.

To integrate both quantity and distance into the classification approach, the average distance $\bar{d}_j$ is calculated between the input point $\boldsymbol{x}$ and all other similar-class points. This is

done for all $j = 1,...,m$ within the k-NN domain. Next, the class membership is computed using the same methodology as in Equation 6.31, apart from the average distances. The mathematical description of this approach is as shown below:

$$u_i(\boldsymbol{x}) = \begin{cases} \dfrac{\bar{d}_T - \bar{d}_i}{\bar{d}_T(m-1)}, & \text{if} \quad m > 1 \\ 1, & \text{if} \quad \bar{d}_T = \bar{d}_i \\ 0, & \text{if} \quad i \notin \boldsymbol{j} \end{cases} \tag{6.32}$$

with $\bar{d}_T$ being the sum of average distances calculated as $\bar{d}_T = \sum_{j=1}^{m} d_j$. This way, quantity plays a role in averaging down the total distance, so that classes with more and closer representatives with respect to x within the k-domain are given the most attention. This approach is named the *average distance* k-NN.

The main difference between this implementation and the original fuzzy k-NN [166] is in terms of initial memberships. In the present case, no initial membership is determined for the training data set. The class membership is solely determined during classification based on two parameters/elements; k and distance. Larger values of k are more likely to produce a fuzzy output.

6.7.1 The Training Set

The test set is built from painting images obtained from collections at the National Gallery of London, the Louvre Gallery in Paris and also from images adapted from Bucklow [46, 149, 167]. For crack patterns, there are no standard databases of image collections that can be used for analysis, unlike for other popular problems such as fingerprint recognition and optical character recognition, where standard data sets are available for purchase (i.e. The National Institute of Standards and Technology (NIST) image database [168]).

To construct a test set, large image of paintings are browsed through and regions containing reasonably clear crack patterns are cropped. It was important that the crack patterns could be easily detected by the crack detection algorithm, so that actual detected crack patterns conformed to a class label.

The dimension of test images varies since crack patterns exist in various sizes and the features are invariant to scale. Most of the test images were approximately 220 x 320, 256 x 256, 400 x 260 and 512 x 512 in dimension. Test images were not restricted to grey-scale images only, although grey-scale is preferable to coloured. The crack detection algorithm

automatically converts coloured images into 1-band images, as colour information is not needed for analysis.

In terms of image type, X-radiograph and visible images are chosen as test set as long as the cracks were clearly visible and not too corrupted by brush stroke patterns and noise. The images in the test set were then manually labelled mainly from observation. The number and proportion of test images according to respective class are shown in Table 6.5.

Class	Total	Proportion
Circular	23	17.4%
Rectangle	36	27.3%
Unidirectional	21	15.9%
Spiderweb	20	15.2%
Random	32	24.2%

Table 6.5: Number and proportion of the test set according to class.

6.8 Feature Selection

From the beginning of this chapter, the approaches used to extract meaningful features have been outlined, but the effectiveness of these features still needs to be evaluated. For classification, the features with high discriminatory power should be chosen. There is an empirical rule of thumb [38] which states that for any classification exercise to be meaningful, there should be at least three times as many samples per class as there are features. Every extra feature adds another degree of freedom to the classifier, so the rule of thumb attempts to balance between predicting the class of an input data and fitting the classifier to the training set. Thus, choosing the right number of features for training and classification is deemed important and essential. Considering the class with the smallest number (spiderweb, 20), the maximum number of features that should be used is six.

For visualisation purposes, a symbolic representation of each feature is summarised in Table 6.6.

In order to select features according to their discriminatory powers, a technique known as the multi-class Fisher Ratio [155] is used for relevant features. The technique performs statistical analysis on features of different classes and assigns a score to indicate how strong the feature is. It scores a feature in terms of the ratio of its between-class separability and within-class spread. The higher ratios indicate higher discriminatory power. The Fisher Ratio is written as

Symbol	Feature
d_1	Directionality based on orientation histogram
d_2	Directionality based on weighted histogram accumulation
u_1	Unidirectionality based on normal accumulation of QGH
u_2	Unidirectionality based on weighted accumulation of QGH
r_1	Rectangularity based on normal accumulation of QGH
r_2	Rectangularity based on weighted accumulation of QGH
f_0	"Circular"-oriented histogram shape filtering using orientation histogram
f_1	"Rectangular"-oriented histogram shape filtering using orientation histogram
f_2	"Unidirectional"-oriented histogram shape filtering using orientation histogram
f_3	"Circular"-oriented histogram shape filtering using normal accumulation of QGH
f_4	"Rectangular"-oriented histogram shape filtering using weighted accumulation of QGH
f_5	"Unidirectional"-oriented histogram shape filtering using weighted accumulation of QGH
s_1	Number of nodes per square root of crack pattern length
s_2	Number of line segments per square root of crack pattern length
s_3	Number of nodes per number of line segments
s_4	Standard deviation of the length of line segments

Table 6.6: A symbolic representation of selected features.

$$F_k = \frac{1}{2N_c(N_c-1)} \sum_{i=0}^{N_c-1} \sum_{j=0}^{N_c-1} \frac{(\mu_{ik}-\mu_{jk})^2}{\left(\sigma_{ik}^2+\sigma_{jk}^2\right)}, \quad i \neq j \tag{6.33}$$

where N_c is the total test data, and μ_{ik}, μ_{jk}, σ_{ik} and σ_{jk} signify the mean (μ) and standard deviation (σ) for every k_{th} feature. A feature that has similar values for all the samples within the same class and where the mean values of that feature for each class differ significantly from other classes will possess a high Fisher Ratio. Table 6.7 and Figure 6.14 show the results for the sixteen features under consideration, sorted in descending order according to their respective scores.

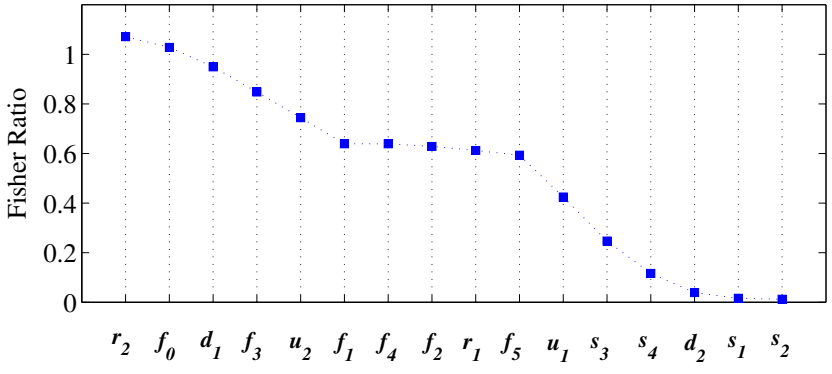


Figure 6.14: Plot of Fisher scores arranged in descending order.

Rank	Feature	Score	Rank	Feature	Score
1.	r_2	1.0709	9.	r_1	0.6121
2.	f_0	1.0281	10.	f_5	0.5935
3.	d_1	0.9494	11.	u_1	0.4248
4.	f_3	0.8480	12.	s_3	0.2469
5.	u_2	0.7446	13.	s_4	0.1169
6.	f_1	0.6395	14.	d_2	0.0391
7.	f_4	0.6391	15.	s_1	0.0162
8.	f_2	0.6284	16.	s_2	0.0125

Table 6.7: Fisher scores for selected features sorted in descending order. (max.=1.0709, min.=0.0125, mean=0.5381).

It is clear from Table 6.7 that most of the histogram-based features occupy the top half of the table with r_2, f_0, d_1, f_3, u_2, f_1, f_4, f_2, r_1 and f_5 obtaining above average scores. Among the histogram-based features, u_1 and d_2 demonstrate two of the less discriminating features, with values below the mean score. Another observation reveals that all the four structural-based features exhibit very low discriminatory power. This shows that features related to the spread of nodes or line segment length do not possess unique characterising power as far as this analysis is concerned.

6.9 Pattern Classification

In selecting the features for classification, the top six features are chosen to characterise each crack pattern. It is important to note that these features possess high discriminatory power as separate entities or values. This does not mean that the six features represent the most discriminating feature subset. Determining the best feature subset requires a different analysis. However for the pursuing experiment, these six features are used.

From earlier discussions, the suitability of the k-NN classifier within the scope of this analysis is clearly evident. Ideally, given the variety of k-NN classifier versions, further trials are required, first to determine the best classifier and secondly to conclude on the optimum value of k. By considering the nature of the problem, which clearly requires a notion of fuzziness in the analysis (see Section 2.6.4), a fuzzy-based technique is more desirable as a classifier. Although the images in the training set are hard classified, the output of a crack pattern classification should be in a fuzzy form. Thus, the next important decision to make is the optimum value of k.

A popular and simple strategy in estimating classification error is the leave-one-out strategy [138] which requires test images from among the training images themselves. From all the

training set, one image is taken out and used as a test image for classification. In the next iteration, the next image is taken out, until all the images in the training set have been independently classified. Assuming n training images, the leave-one-out strategy requires n number of iterations/classifications. The error estimation can be determined by the percentage of unsuccessful classifications.

The leave-one-out strategy is used to determine the value of k that produces minimum classification error where the error is calculated for k in the range of [1, 50]. Figure 6.15 shows the percentage of successful classifications among k for the k-NN, the distance weighted k-NN and the average distance k-NN using all the top 6 features. The fuzzy outputs are transformed into hard classifications by taking the maximum class assignments.

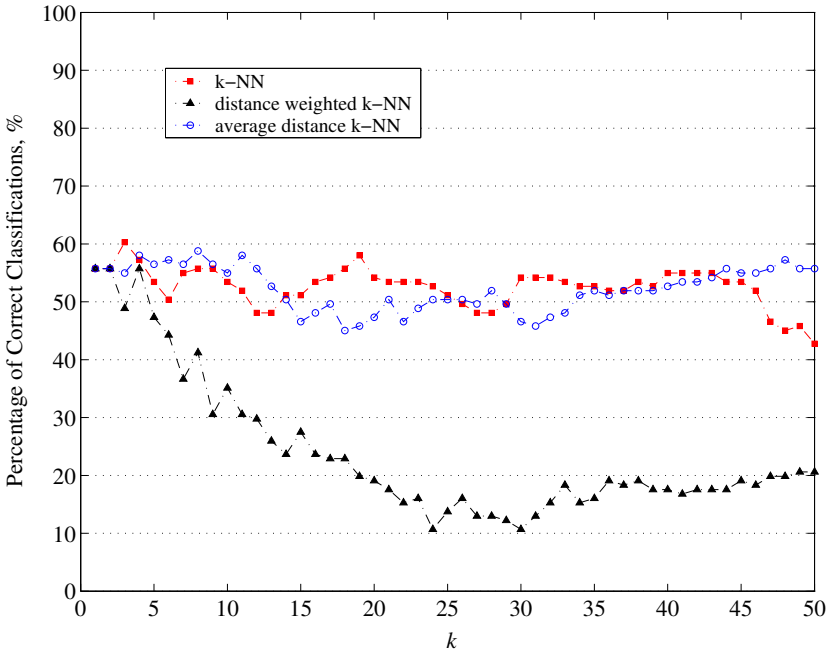


Figure 6.15: Percentage of successful classifications for k in the range [1, 50] using the top six features. The maximum successful classification percentage and the respective values of k are as follows: k-NN (60.3% at k=3), distance weighted k-NN (55.7% at k=1) and average distance k-NN (58.8% at k=8).

Optimal k values are obtained at k=3 for the k-NN, k=1 for the distance weighted k-NN and k=8 for the average distance k-NN. As can be seen, the distance weighted k-NN is very sensitive to an increase in k. The other two classifiers demonstrate more consistency for a wide range of k.

The very low correct classification percentage (60.3%, 55.7% and 58.8% respectively), were expected, since the six features used for classification are not highly effective in separating all the five desired patterns at a single process. Based on analysis performed on correct classification percentages of individual classes, it is observed that the results differ quite

significantly for different classes. This is shown graphically in Figure 6.16(a) for the k-NN classifier and in Figure 6.16(b) for the average distance k-NN classifier.

Also by taking the values of the correct classifications at the optimum values of k obtained from the earlier analysis (see Figure 6.15), it is quite evident that the six features used are tuned towards characterising the unidirectional and the rectangular patterns more than the other three. In other words, the unidirectional and rectangular patterns are the easiest to classify of all the classes. This is not very surprising considering the visually distinctive attributes they possess over the rest of the classes. Table 6.8 summarises the results at the respective values of k for the k-NN and average distance k-NN classifier.

Class	k-NN (k=3)	average distance k-NN (k=8)
Circular	17.4%	30.4%
Rectangular	77.8%	75.0%
Unidirectional	81.0%	81.0%
Spiderweb	60.0%	55.0%
Random	58.1%	48.4%

Table 6.8: Distribution of correct classifications over different classes.

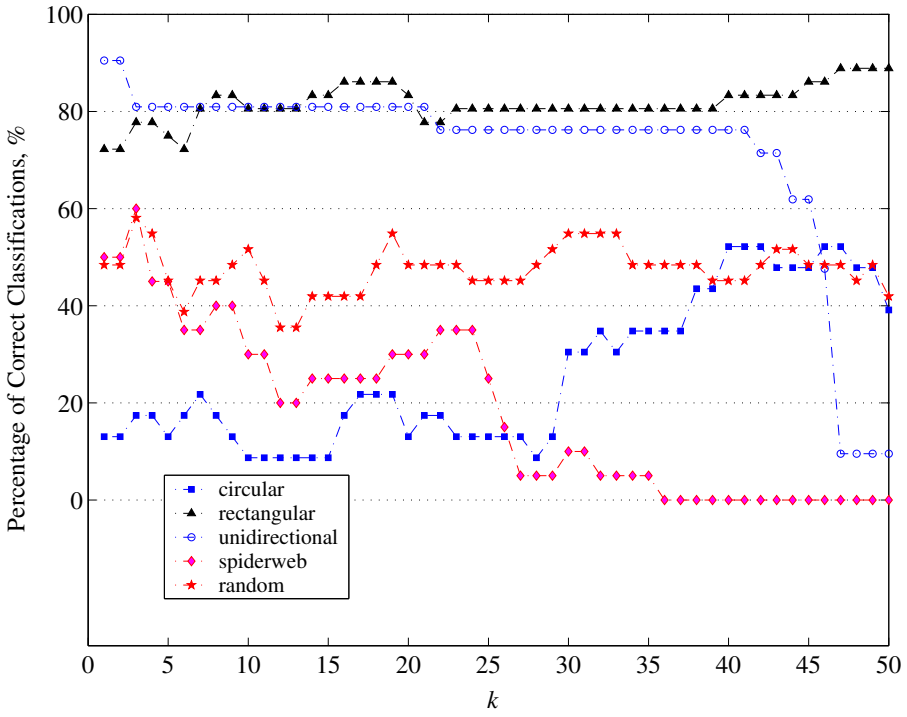
At this point, the need to obtain evenly spread correct classification percentages over all the classes is highly critical. If this is achieved, a pattern input into the classifier will be classified with an even chance or probability of success for the five classes.

6.9.1 Three-stage Classification

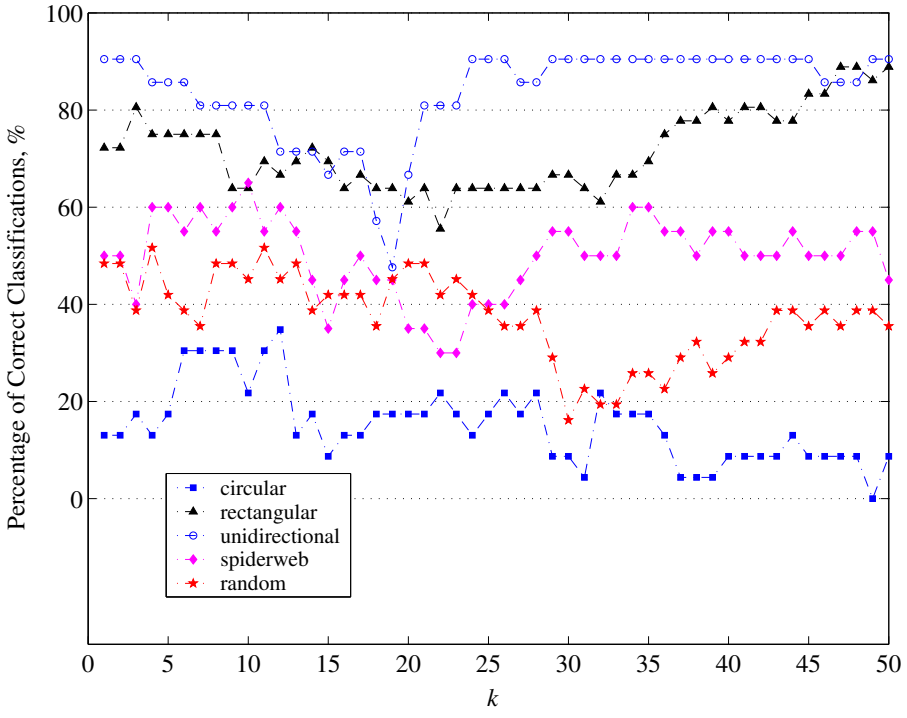
To solve the problem, a multi-stage classification strategy is performed instead of a single flow process. The key idea behind the multi-stage strategy is to classify patterns in a hierarchical fashion. This allows a single classifying stage to concentrate on separating patterns into fewer classes, compared to the traditional approach, where patterns are classified into all the c classes. The need for this originates from the small inter-class separability of the chosen features which makes it highly challenging to separate patterns into 5 classes in just a single flow.

6.9.1.1 Three-stage Classifier Using a 2-2-3 Strategy

A multi-stage classifier introduces more flexibility by allowing specialised (and thus fewer) features to be used at different stages of the classifier. This is made possible by the fact



(a)



(b)

Figure 6.16: Graphs showing significant differences in terms of correct classification percentage for individual classes for a) the k-NN classifier and b) the average distance k-NN classifier.

that inter-class separability of patterns can be increased at each level given that several pre-defined classes are merged into a larger class. This leads to the need for identifying *groups of classes* which in other terms refer to combinations of classes. As an example, the rectangular, circular, spiderweb and random crack patterns can all be grouped together by virtue that they generally have multi-orientation structures; let this group be named $G1$. The other class consists of the unidirectional pattern of uni-orientation structures. The division defines the two classes in the first stage of the classifier.

In the second stage, $G1$ can be further divided into two groups, the first being the rectangular pattern and the second, called $G2$, being a combination of the circular, spiderweb and random patterns. These three patterns are considered similar, since their patterns are multi-orientation in nature, while the rectangular pattern generally exhibits bi-directional structural orientations.

In the final stage, the separability between classes becomes narrower and a richer combination of features is needed to classify the remaining patterns, namely the circular, spiderweb and random patterns.

The flow diagram for the three-stage classifier is illustrated in Figure 6.17. The algorithm classifies an input pattern into two classes in the first stage, two classes in the second stage and three classes in the third stage. The notations u_1, u_2, u_3, u_4, u_5, u_{G1} and u_{G2} represent the fuzzy membership values assigned to the patterns unidirectional, rectangular, circular, spiderweb, random, $G1$ and $G2$ respectively. The actual fuzzy membership values for each class are defined by the following equations:

$$u_{unid} = u_1, \tag{6.34}$$

$$u_{rect} = u_2 * u_{G1}, \tag{6.35}$$

$$u_{circ} = u_3 * u_{G2} * u_{G1}, \tag{6.36}$$

$$u_{spid} = u_4 * u_{G2} * u_{G1}, \tag{6.37}$$

$$u_{rand} = u_5 * u_{G2} * u_{G1}. \tag{6.38}$$

Another input parameter is the value of k, represented by k_1, k_2 and k_3 for the first, second and third classifier stages respectively. Since the number of training samples used decreases as the classifier stages increase, the value of k must also be reassigned at different stages. An initial value k_1 is first assigned to the classifier and the inputs to the following classifiers

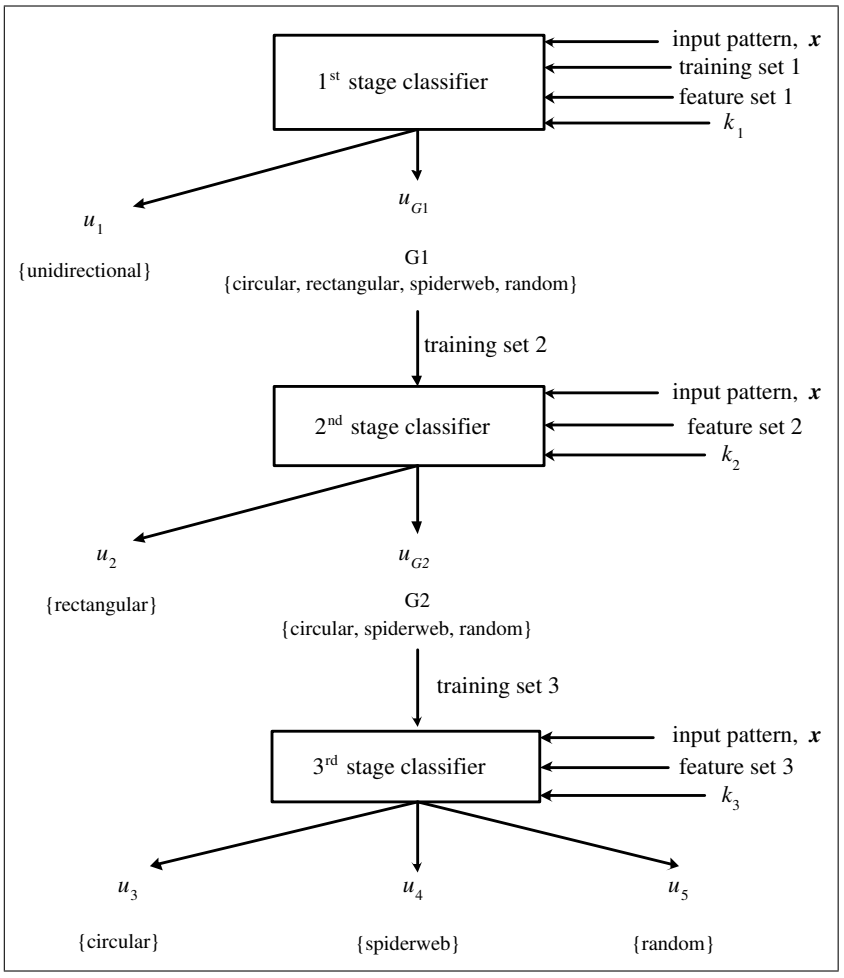


Figure 6.17: Figure showing the flowchart of the three-stage classifier where the input pattern is classified into two classes in the 1st stage, two classes in the second stage and three classes in the 3rd stage (2-2-3 strategy).

are k_2=ceil($4k_1/5$) and k_3=ceil($3k_1/5$). This schema only applies if the value of k_1 is more than five, otherwise k_3=k_2=k_1. The assignments take into consideration the portions of classes in the training set which are close to an even division of five.

Based on the evaluation of class separability by all the sixteen features using the Fisher Ratio, best sets of features can be chosen to represent classes at each classifier stage. Features are picked according to the score they obtained, with a maximum of six features per stage. Table 6.9 summarises the selected features with their respective scores.

Classification accuracy is again evaluated using the leave-one-out strategy. The k-NN and the average distance k-NN classifiers are used in each stage of the three-stage classifier, with k1, k2 and k3 as the number of nearest neighbours considered at each stage. Figure 6.19 shows comparisons of the two classifiers using the three-stage approach for k=1,...,50.

The three-stage classifier shows improvement for both the k-NN and the average distance

1^{st} stage		2^{nd} stage		3^{rd} stage	
Feature	Score	Feature	Score	Feature	Score
d_1	2.1229	f_4	1.2187	f_2	0.2030
u_2	1.9526	f_3	1.1194	f_0	0.1699
-	-	f_1	1.0767	f_5	0.1036
-	-	r_2	0.8018	s_4	0.0988

Table 6.9: Selected features used for the three-stage classifier using a 2-2-3 strategy based on Fisher Ratio scores.

k-NN (as much as 3.1% and 6.1% respectively). However, the improvements are still considered small for the amount of modification to the classifier; higher improvements are expected.

The intriguing aspect of the classification results is the total misclassification of spiderweb patterns (see Table 6.10). To make matters worse, the spiderweb pattern is misclassified for both classifiers for all values of k. This experiment triggered the idea of omitting random patterns from the training set. The vague definitive criteria of random patterns makes it unsuitable to be even considered as a class in the first place. The inclusion of a random pattern in the framework only increases the level of confusion for the classifier. Instead of being a class, random patterns should be more thought of as "a class which does not belong to any of the other four classes". This is a definition that will be further elaborated later in the chapter.

Class	k-NN (k=2)	average distance k-NN (k=2)
Circular	69.6%	69.6%
Rectangular	80.6%	86.1%
Unidirectional	90.5%	90.5%
Spiderweb	0.0%	0.0%
Random	61.3%	61.3%

Table 6.10: Distribution of correct classifications over different classes for the three-stage classifier. Spiderweb pattern is totally misclassified.

6.9.1.2 Three-stage Classifier Using a 2-2-2 Strategy

A huge step is taken next by excluding the training patterns representing the random pattern. Thus, the size of the training set decreases to some extent. A modification is made to the three-stage classifier, with a 2-2-2 approach used instead of a 2-2-3 (see Figure 6.18). The third stage of the classifier now only has to classify the input pattern into two classes, circular and spiderweb. Hopefully, with the omission of the random pattern, the

confusion level decreases. Table 6.11 revised Fisher Ratio scores for selected features of the three stages.

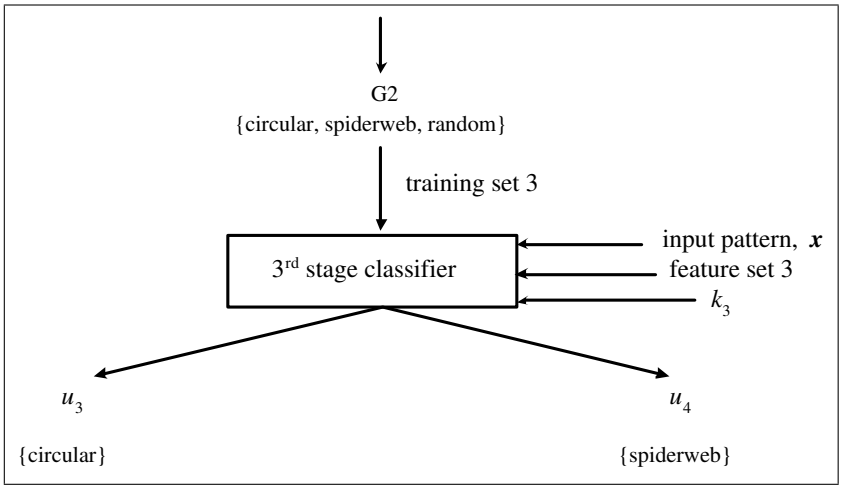


Figure 6.18: Figure showing the modification made to the flowchart of the three-stage classifier, with the third stage now needing to classify the input pattern into either one of two classes, circular or spiderweb.

1^{st} stage		2^{nd} stage		3^{rd} stage	
Feature	Score	Feature	Score	Feature	Score
r_2	3.0414	f_0	2.2073	s_4	0.1564
d_1	2.8886	f_3	1.9565	f_2	0.1229
u_2	2.8331	f_4	1.7016	f_5	0.0785
f_0	2.1186	f_2	1.4367	u_2	0.0154
-	-	r_2	1.2765	-	-

Table 6.11: Selected features used for the three-stage classifier using a 2-2-2 strategy based on Fisher Ratio scores.

The leave-one-out strategy is once again employed to estimate the accuracy of the classifier with four classes to classify into. The focus here is to reduce class-specific classification errors, especially for the spiderweb class, which proved to be the main drawback in the 2-2-3 strategy. The effect of omitting the random pattern on overall classification error remains the main interest. Based on the results shown in Figure 6.19, both the k-NN and average distance k-NN using a 2-2-2 strategy showed vast improvements over the 2-2-3 strategies in terms of correct classification percentage for specific values of k.

The k-NN classifier improves as much as 12.6% while the average distance k-NN classifier experienced a 17.1% improvement. Looking into the class-specific classification error, omission of the random pattern proves to be a crucial step according to the results shown in Table 6.12.

Although some classes experienced reduction in the percentage of correct classification, the

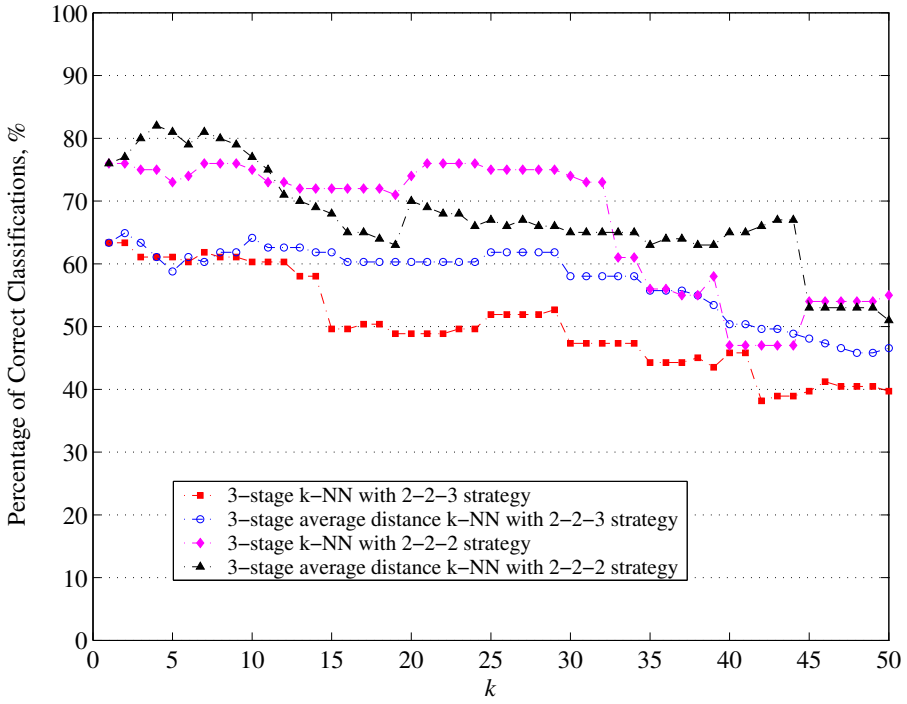


Figure 6.19: Percentage of successful classification for k in the range [1, 50] using the three-stage approach with the random pattern omitted leaving only four classes to classify into. The maximum successful classification percentage and the respective values of k are as the following; k-NN using a 2-2-3 strategy (63.4% at various values of k), average distance k-NN using a 2-2-3 strategy (64.9% at k=4), k-NN using a 2-2-2 strategy (76.0% at various values of k) and average distance k-NN using a 2-2-2 strategy (82.0% at k=4).

Class	k-NN (k=2)	average distance k-NN (k=4)
Circular	60.9%	65.2%
Rectangular	94.4%	100.0%
Unidirectional	76.2%	81.0%
Spiderweb	60.0%	70.0%

Table 6.12: Distribution of correct classifications over different classes for the three-stage classifier using the 2-2-2 strategy. Vast improvement is experienced over the 2-2-3 approach, especially for the spiderweb pattern.

classifier succeeded in classifying 60% and 70% of the spiderweb test samples. These are important results, considering that the 2-2-3 strategy failed miserably in the task, directly affecting the overall classification performance.

Another way of evaluating classification performance is by including fuzzy elements in the consideration of correct classifications. Previous experiments assumed hard classification to give total membership to a particular class. In evaluating fuzzy classification, soft memberships are made accountable in deciding whether a test pattern is correctly classified. As an example, let test pattern A be a circular class which is classified as 32% circular,

12% rectangular, 3% unidirectional and 53% spiderweb. In a hard classification rule, A is considered misclassified. However, in fuzzy terms, this is not really the case, since A is classified as 32% circular (second in rank) which represents quite a strong figure, considering the fact that there are four classes (25% per class in average). Furthermore, looking from human perception, it is very clear that subjectivity is quite inevitable when it comes to classifying crack patterns, or any problem which involves complicated combinations of line structures. Each crack class may contain an element of the others to a certain extent and this situation is the situation which the fuzzy strategy tries to model.

Thus, in the next analysis, correct classification is defined as any condition where the test pattern is classified into its actual class with more than 25% confidence, regardless of the rank. Figure 6.20 attempts to compare the performance of the k-NN and the average distance k-NN classifers with the fuzzy element brought onto the surface. As expected, the percentage increased dramatically when the new rule was applied. The interesting point from the analysis is that nearly all the test patterns (96% for both classifiers) received considerable amount of attention from the classifiers (above 25% confidence vote). Table 6.13 shows the class-specific classification success for both classifiers.

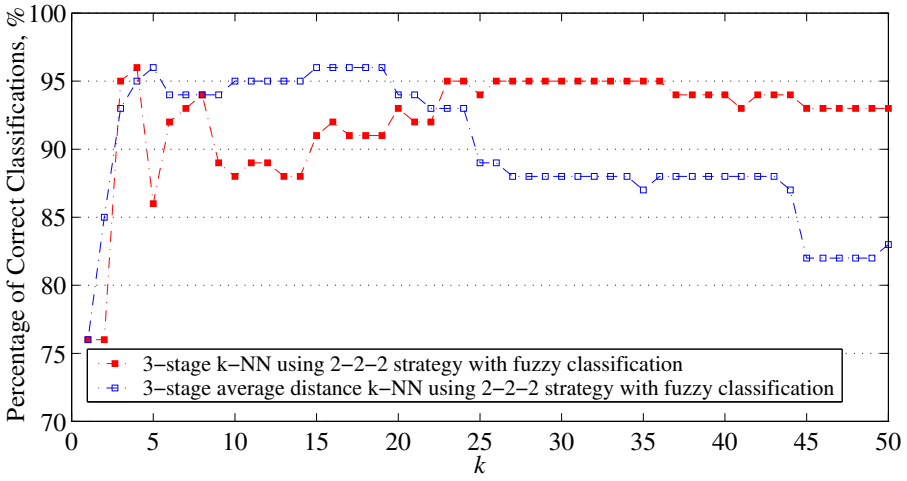


Figure 6.20: Percentage of successful classifications for k in the range [1, 50] when fuzziness is considered in defining correct classification. A pattern is considered correctly classified if more than 25% confidence is recorded for its actual class. The maximum successful classification percentage and the respective values of k are as follows; k-NN (96% at k=4) and average distance k-NN (96% at various value of k).

6.9.1.3 Selecting the Optimum Value of k

The next problem in hand involves selecting the optimum value or range for a number of neighbours k. An obvious way to decide is by choosing k, which leads to the least

Class	k-NN (k=4)	average distance k-NN (k=5)
Circular	100.0%	91.3%
Rectangular	100.0%	100.0%
Unidirectional	95.2%	95.2%
Spiderweb	85.0%	95.0%

Table 6.13: Distribution of correct classifications over different classes for the three-stage classifier using the 2-2-2 strategy with fuzzy elements taken into consideration (above 25% class confidence).

classification error. However, another point to ponder about this issue is whether, at this k, the variance of class-specific classification error is at minimum. It is important to make sure that the difference between classification errors for each class is kept at minimum while maintaining high overall classification performance. This can be assessed by the ratio of the mean correct classification percentage and its standard deviation over all 4 classes. The value of k which produces the maximum ratio can be considered the optimum value of k. In a case where there is a tie in terms of the value of k which produces the least error, the following methodology can be used to solve the problem. The ratio for a certain value of k is given as $R_k = \mu_k/\sigma_k$ where μ_k and σ_k are the mean and standard deviation of class-specific correct classification percentages for k nearest neighbours. By taking the maximum value of R_k over k=1,...,50, the optimum value of k can be determined as shown in Table 6.14 for the fuzzified leave-one-out classification performance evaluation.

Evaluation approach	k-NN			average distance k-NN		
	k	max(R_k)	% at k	k	max(R_k)	% at k
Hard	20	8.18	74.0%	4	5.12	82.0%
Fuzzy	16	15.34	92.0%	5	26.76	96.0%

Table 6.14: Comparisons between the k-NN and the average distance k-NN in terms of class-specific classification performance. The ratio between mean and standard deviation of correct classification percentage, R_k is used to assess classification performance at various values of k.

The results can be evaluated in several ways. In terms of computation complexity, which can be determined by the number of k taken to an optimum point, the average distance k-NN outperformed the k-NN in both hard and fuzzy approaches. The average distance k-NN only needs 4 or 5 nearest neighbours. This figure is much lower compared to the k-NN which needs 16 or 20. Although the average distance k-NN is slightly more complicated to compute compared to the k-NN, the gap between the optimum values of k is much greater. In terms of accuracy, again the k-NN falls behind the average distance k-NN, where the latter producing accuracy up to 82.0% and 96.0%, while the k-NN only succeeded in

correctly classifying 74.0% and 92.0% of the test samples. From the statistical figures, it is clear how the average distance k-NN outperformed the k-NN in many ways. Finally, based on the experiments, the value of k may either be 4 or 5 to achieve optimum classification performance.

The confusion matrices shown in Tables 6.15 and 6.16 try to analyse confusions within the classifier. It is noticeable how circular patterns are easily misclassified as spiderweb and vice versa. This shows that the two classes closely in resemble each other stronger features should be used to better distinguish between them.

	Assigned class			
Actual class	Circular	Rectangular	Unidirectional	Spiderweb
Circular	15	0	2	6
Rectangular	0	36	0	0
Unidirectional	0	4	17	0
Spiderweb	5	1	0	14

Table 6.15: Confusion matrix (k=4).

	Assigned class			
Actual class	Circular	Rectangular	Unidirectional	Spiderweb
Circular	13	1	2	7
Rectangular	0	36	0	0
Unidirectional	0	4	17	0
Spiderweb	4	1	0	15

Table 6.16: Confusion matrix (k=5).

Another issue that is worth mentioning here is the definition of a random type pattern which was left for discussion earlier in the chapter. The criteria that characterised the random pattern are quite vague, and, as mentioned earlier it can be defined as "a class which does not belong to any of the other four classes" or it can also be looked on as "a class which possesses elements of all four classes". Relating it to the behaviour of the classifier, it is quite suitable to co-assign the random pattern to any input patterns which have non-zero memberships in at least three classes with the maximum membership not larger than a certain percentage (maybe 50%). This is to make sure that the input pattern possesses elements of the majority of the classes, with none of the classes being highly dominant.

Figures 6.21 and 6.22 show results of individual classification of manually generated shapes and also real craquelure patterns (which are not part of the training set). Although the classifier is trained on actual craquelure patterns, tests are performed to observe the response of the classifier to manually generated patterns. Patterns which used to belong to the random class training set are also used.

6.10 Summary

This chapter dealt with various issues related to the extraction of pattern signature features and also the important steps required for classification of crack patterns.

Due to the various structural layers in a crack pattern (i.e. the local, crack-network and global layers), the way in which features are extracted can follow the same hierarchical concept. The key element here is the *significance measure* which determines the amount of contribution an object in a lower level has in the formation of another object in a higher level. Primarily, this measure is determined by the size of an object, which is characterised by its total length.

Various features have been generated from crack patterns, mostly based on normal or weighted orientation histograms and also statistical values originating from structural entities, which include nodes as well as line segments. Early tests revealed several interesting observations and features showing promising ability in discriminating crack patterns.

Experiments were conducted to demonstrate the effectiveness of the generated features. The k-NN is used as a basis for classification. Several varieties of k-NN based techniques (i.e. the distance weighted k-NN and the average distance k-NN) are introduced as classifiers alongside the traditional k-NN. These techniques are "fuzzified" by using votes and distance information. Classifier accuracy was estimated using the leave-one-out strategy. A three-stage classifier using a 2-2-3 strategy is tested and proved to be better than the conventional one-flow classifier with improvements as much as 3.1% and 6.1%. The definition of a random pattern class was revised and it was shown how the existence of the random pattern created severe confusion in classifier behaviour. The vagueness of its definition was identified as the main problem and the situation was improved by omitting the random pattern from the class set.

The three-stage classifier with a 2-2-2 strategy worked better, with up to 82% correct classification for the hard approach and 96% correct classification for the fuzzy approach, both using the average distance k-NN. From the experiments, it is concluded that the optimum values for k are 4 and 5. Finally, classification results are shown for manually drawn patterns and also for real craquelure patterns.

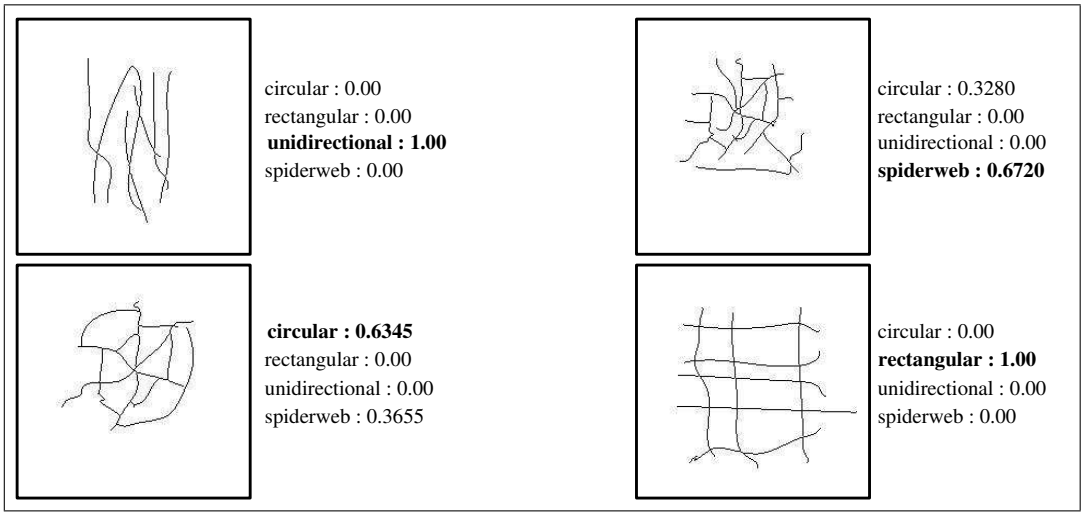


Figure 6.21: Results of classification on manually generated patterns.

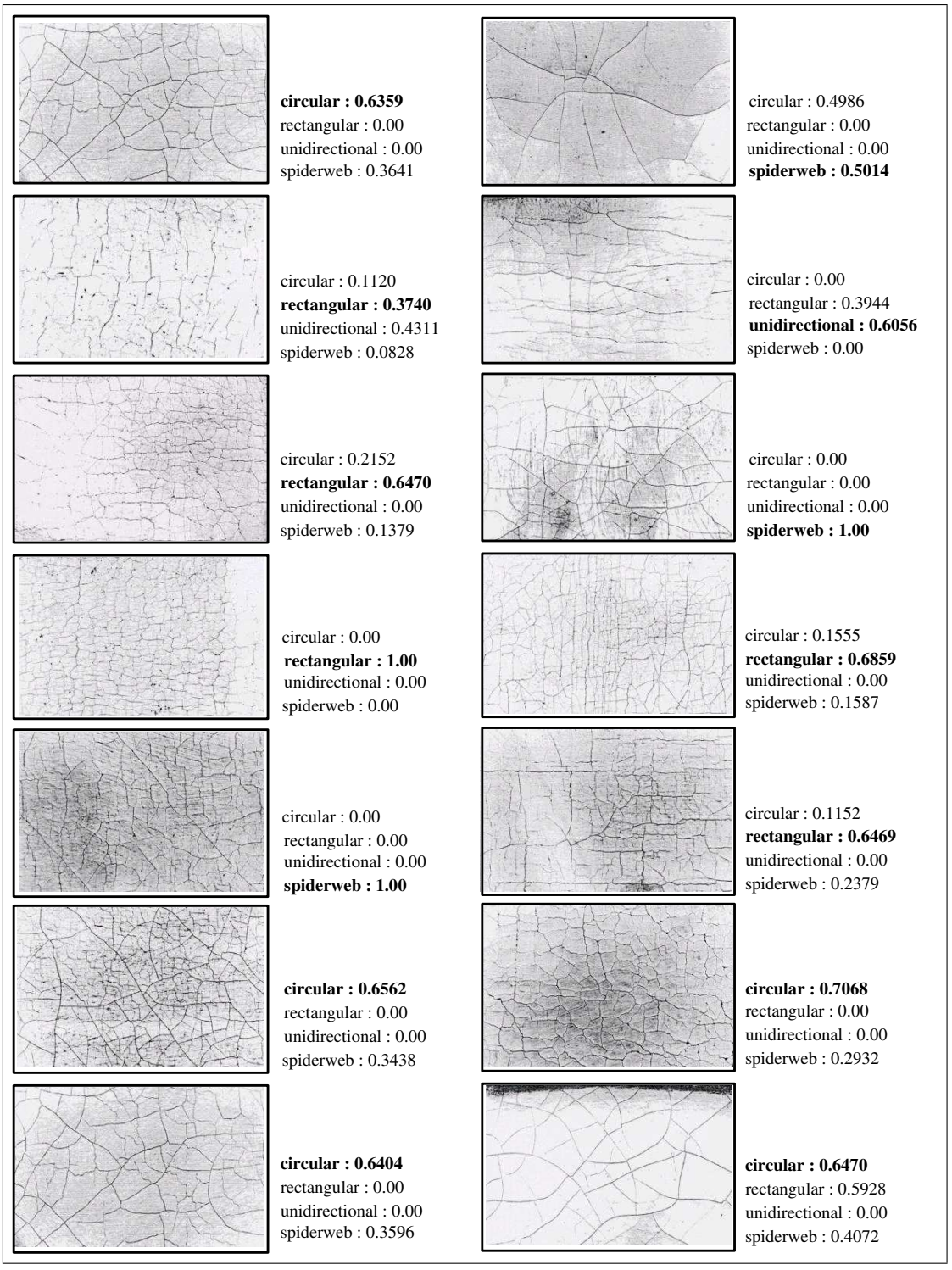


Figure 6.22: Results of classification on real craquelure patterns.

Chapter 7

System Implementation

7.1 Introduction

Real application of a content-based craquelure analysis system requires an integration of the sub-modules which were discussed earlier. In this chapter, all the sub-modules that were explained in the earlier chapters are integrated into an implemented system. The realisation of the proposed *Application Module* (AM) and *Processing Module* (PM) is demonstrated. Query formats, namely *query by image example* and *query by class* are explained with sample results. Craquelure pattern analysis and sub-image queries are also touched on later in the chapter.

7.2 System Implementation

The algorithms are fully written in C programming language, while the result viewer is a web-based interface powered by Hypertext Preprocessor (PHP). The implementation of the system begins from a command-line user interface which opens up a result viewer in a web browser once a query is entered.

Feature vectors are stored in data files, with each file associated with a single image. In each feature file there is information regarding the file source, image dimensions and objects-of-interest. For each object-of-interest, the upper-left coordinate point as well as the x and y coordinate lengths are recorded. This is followed by all the nine features. A similar strategy is used to store the classification information for each image. Classification files are generated for all images mainly to store the class memberships of all existing objects-of-interest. Similar to the feature file, information on the image source, image size and

the location of the objects-of-interest is the basic information stored. For each object-of-interest, classification details, i.e. the class memberships, are recorded in the same file. For a detailed explanation, see Appendix D.

At the time of writing, the system has only been implemented for experimental purposes and is not fully functional for actual usage as a commercial system. The command-line interface acts as a mediator between the user and the algorithm by getting queries and sending them to the query processor for further action. Most of the features of the system deal with query-retrieval scenarios, where a query (text-based or image-based) is entered and lists of results are displayed in the browser (texts or/and images). Another functionality of the system allows users to request information regarding a queried crack image which also includes information about the meaningful objects detected in the image. The result viewer displays the statistical values together with classification results.

The following sections explain the type of functionalities in the existing system.

7.2.1 Query by Image Example

Query by image example is a widely used technique for retrieving images in the database that contain similar attributes to the input image. A user simply enters an image as a query, the system searches the database based on the query, and, when matching images are found, they are presented to the user according to their respective matching scores. While the flow of the process is relatively straightforward, the attribute(s) used as the *matching factor* may vary.

From the system's point of view, there are three cues when searching for matches, using class, feature or fuzzy set similarities. Interestingly, these cues are related to each other in the sense that they originate sequentially from the other; fuzzy sets originate from classified features while classes originate from fuzzy sets.

7.2.1.1 Feature Matching

The features used for matching are partly the same features used in the three-stage classification, namely d_1, u_2, r_2, f_0, f_2, f_3, f_4 and f_5. s_4 is not used, since this statistical measure is not in the range of 0 to 1 unlike the rest of the features. Using a euclidean metric, the distance between features generated from the query image and pre-generated features of all images in the database are computed. Figures 7.1 and 7.2 show results for queries based

on feature similarity. For all results, scores are shown as dissimilarities (distances). Figure B.1 shows a screenshot of the result viewer.

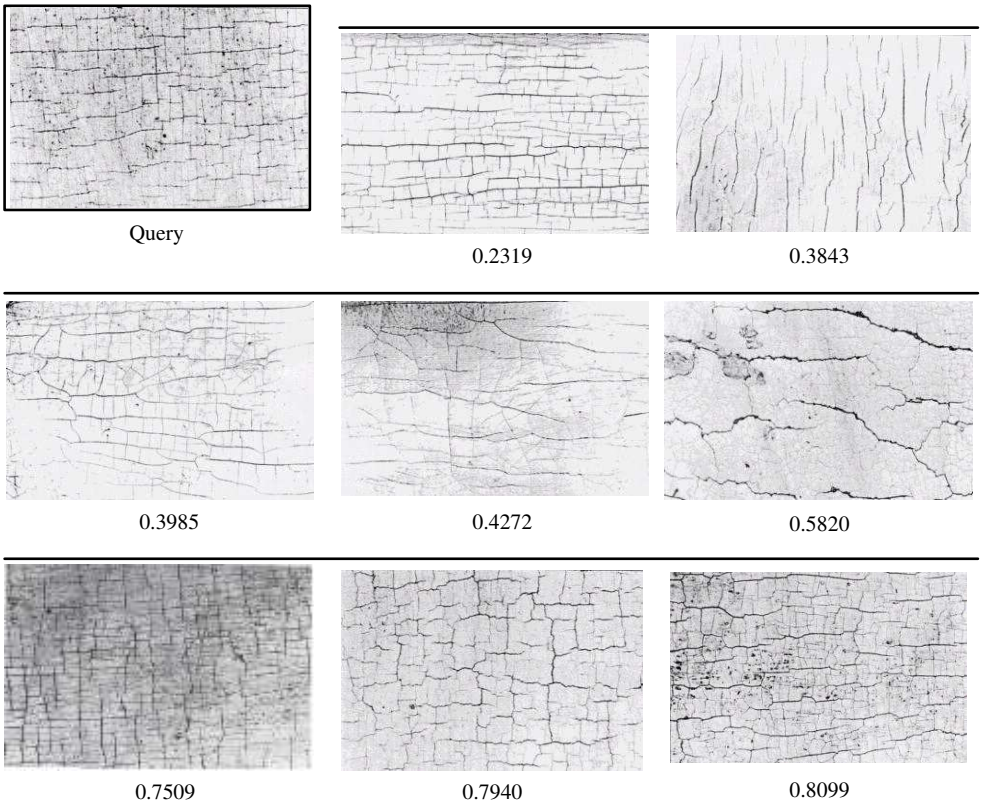


Figure 7.1: A query using feature vector as a cue for measuring dissimilarity.

7.2.1.2 Fuzzy Set Matching

Although rarely used, fuzzy sets (class membership for a crack pattern) can also be used as a means for computing similarity between two patterns. A fuzzy set of a pattern holds information regarding the class membership of that particular pattern in the form of a set of confidence measures with regard to all existing classes. Similar query images are used to demonstrate the use of fuzzy set as a cue for similarity searches and Figures 7.3 and 7.4 display the results. Figure B.2 shows a screenshot of the result viewer.

7.2.1.3 Class Matching

The simplest manner of *query by image example* looks for crack images in the database, which are categorised in the same class as the query image. This of course needs the system to first classify the query image to obtain its fuzzy set, and it is then categorised into the class with the highest value of fuzzy membership. Results are ranked in ascending

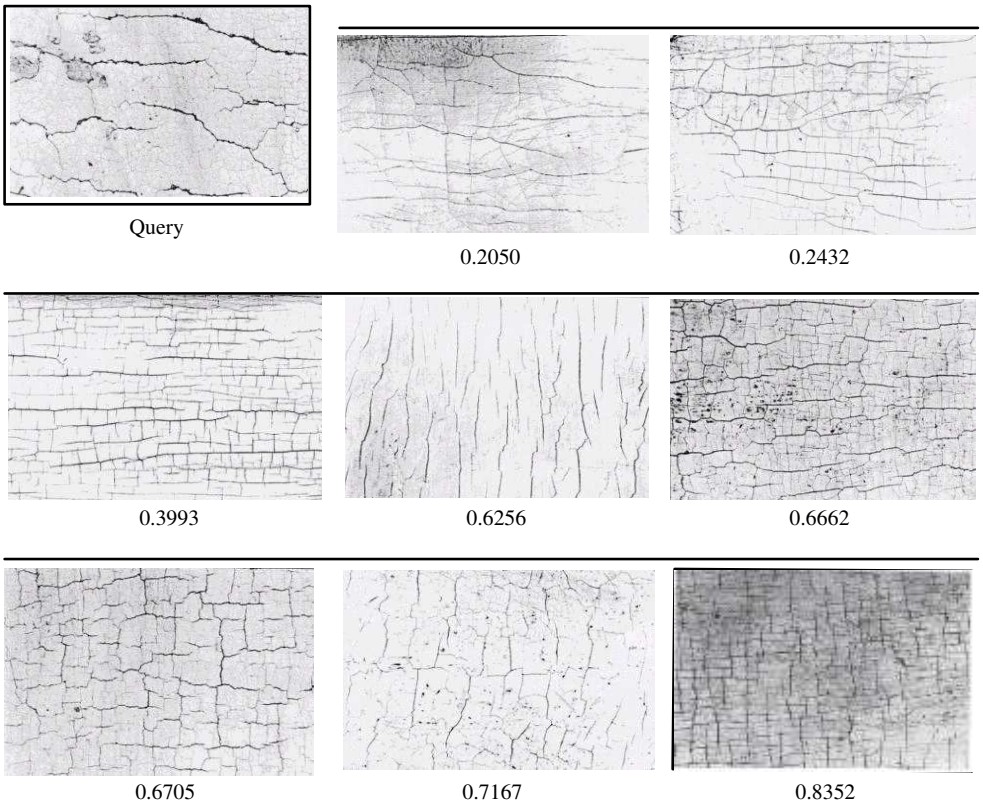


Figure 7.2: Another example of a query using feature vector as a cue for measuring dissimilarity.

order according to the corresponding fuzzy membership of each matching image. Using similar query images as previously, the query results for this particular method are obtained (Figures 7.5 and 7.6). Scores are determined by the membership of the corresponding pattern class. Figure B.3 shows a screenshot of the result viewer.

7.2.2 Query by Text

Text-based query is a simpler way to perform query, where description of the desired query is entered into the system in textual format. A straightforward scenario involves users requesting for all crack patterns with a particular attribute. A more complex example perhaps requires a mix of information from the user.

A user enters the class required (circular, rectangular, unidirectional or spiderweb) and the system retrieves all relevant images. A way to narrow the scope of search is by specifying threshold values for the search. As an example, "retrieve all rectangular patterns with fuzzy membership larger than 40%".

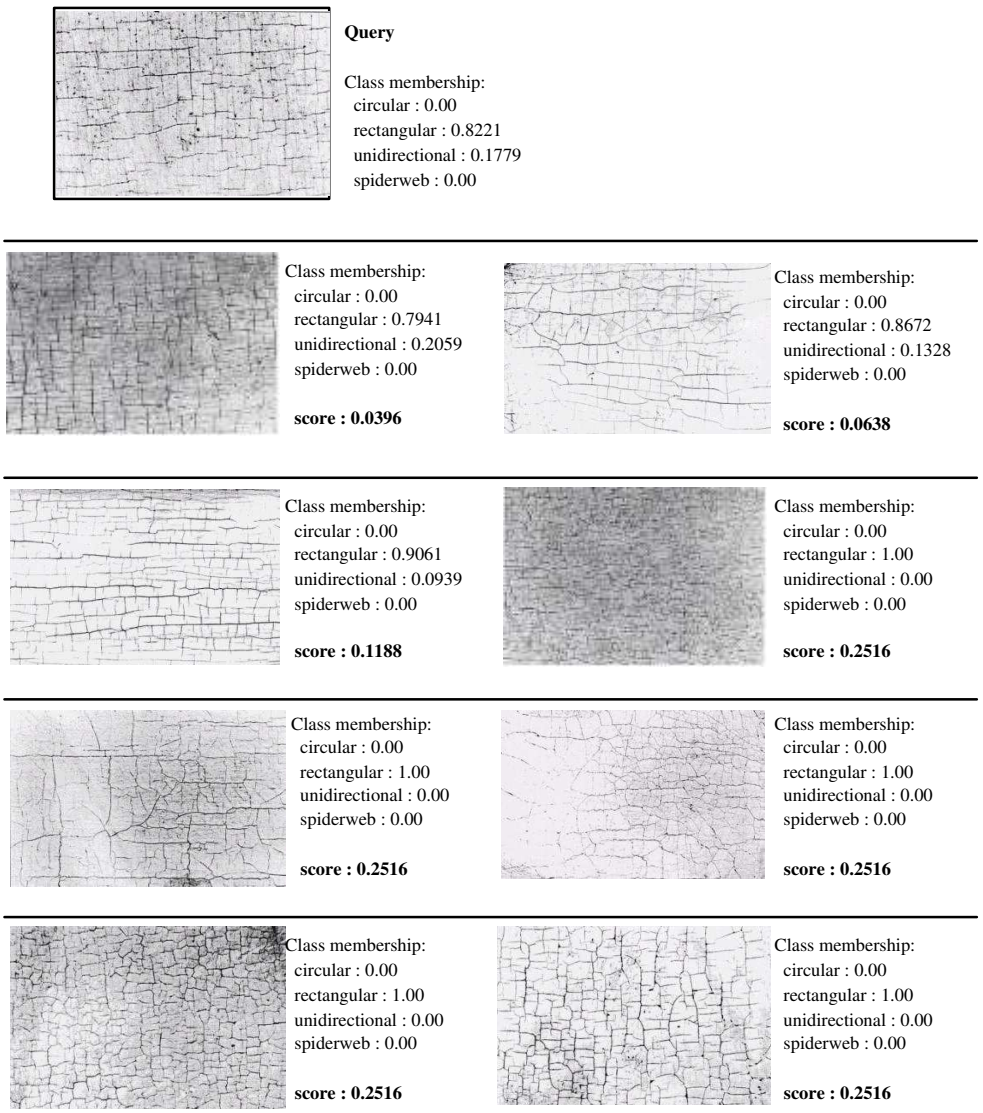


Figure 7.3: A query using fuzzy set as a cue for measuring dissimilarity.

7.2.3 Craquelure Pattern Analysis

Another important functionality that might be of interest to restorers or conservators is random analysis of craquelure patterns. Users, in this case enter an image into the system as a query. The system then extracts relevant information from the image, including partitioning it into objects-of-interest. Statistical information as well as class memberships for both the input image and the objects-of-interest are then computed and displayed to users. This functionality is exemplified by an actual screenshot of the result viewer as shown in Figure C.1, where the input image is partitioned into several automatically detected objects-of-interest. An overall statistical analysis, which covers the whole of the image, is displayed together with classification results, which includes class memberships.

Information associated with the objects-of-interest can be viewed by clicking on the objects'

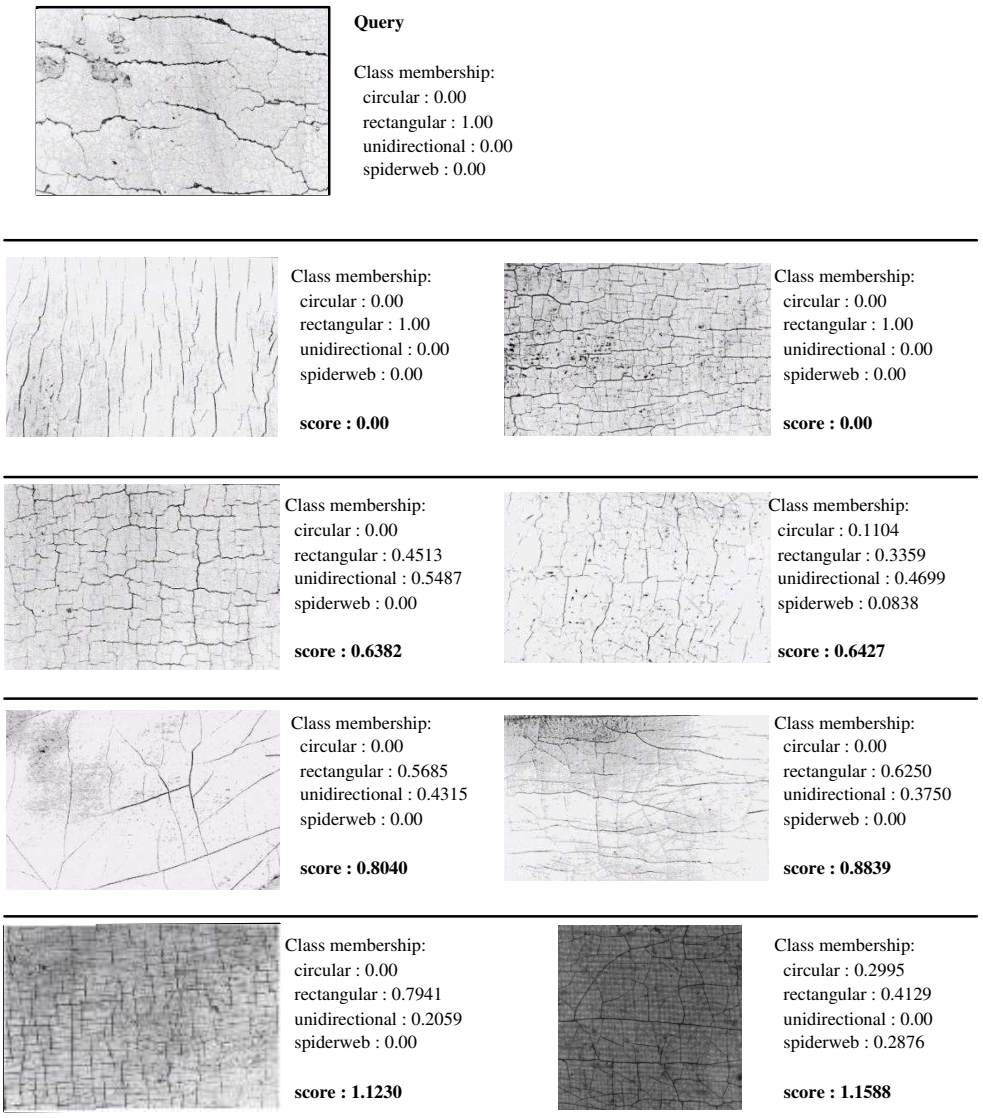


Figure 7.4: Another example of a query using fuzzy set as a cue for measuring dissimilarity.

centroid. A new window appears for each object-of-interest, displaying relevant information as shown in Figures C.2, C.3, C.4 and C.5.

7.2.3.1 Processing Speed

The time taken to execute pattern analysis of crack images is one of the concerns. Museum collections consist of very large images which require fast processing. In this sub-section, the performance of the algorithm is evaluated, in terms of the time taken to execute certain processes. Images of different sizes and complexity (amount of cracks) are used. The length, number of crack-networks, number of nodes and number of line segments are recorded for each process to model the measure of complexity of an image.

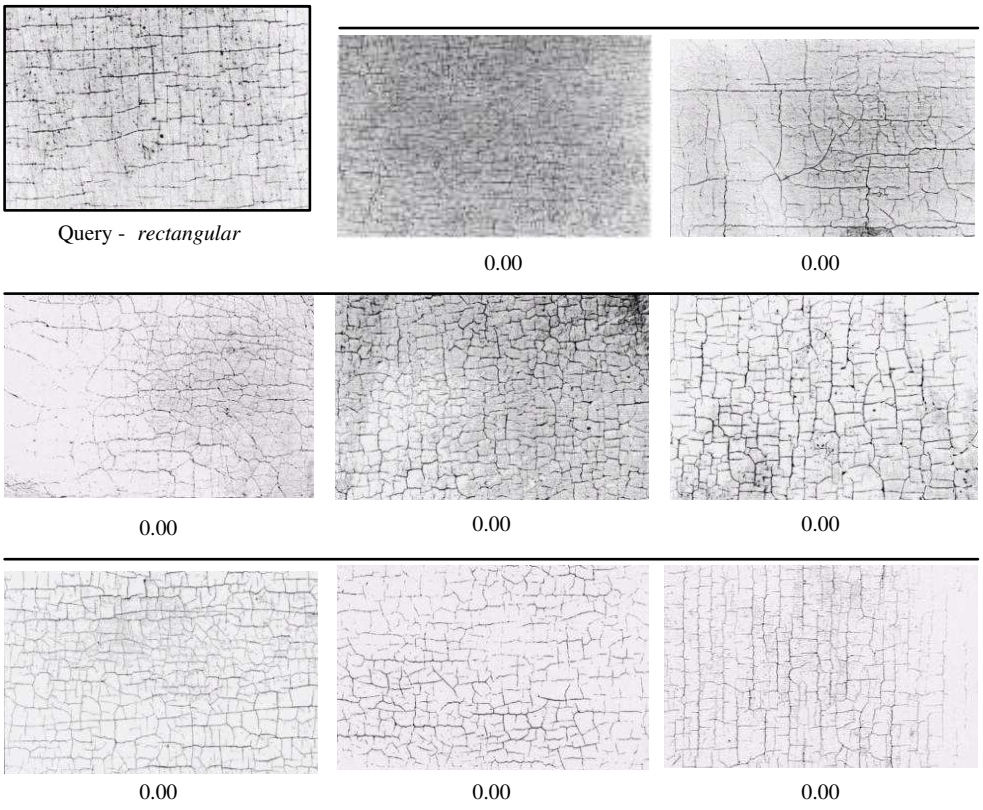


Figure 7.5: A query using class membership as a cue for measuring dissimilarity.

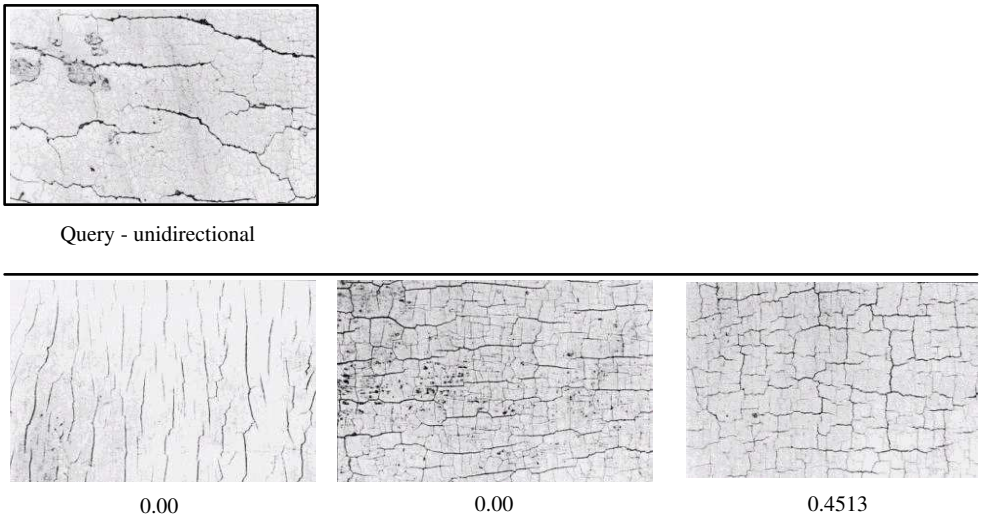


Figure 7.6: Another example of a query using class membership as a cue for measuring dissimilarity.

Processing speed is recorded for two separate processes. t_1 corresponds to the time taken in seconds to fully detect and prune crack patterns while t_2 represents the time taken to perform pattern structuring, merging, feature extraction and classification. These two are added to reveal the total processing time, t_{total}. The percentage of time taken for t_1 and t_2 are also calculated. Table 7.1 summarises the results.

The images are sorted in ascending order according to their size. As can be observed, the time taken to process the images is dependent on both the size and complexity. For the relatively small images (images A-O), a maximum of 5 seconds is recorded for t_1 (image N) and 22 seconds for t_2 (image O) while image O is the longest to process (26 seconds). Similarly, for the large images (images P-T), a maximum of 60 seconds is recorded for t_1 and 13610 seconds for t_2 (both for image T). Overall, image T took the most time to process (13670 seconds = 3 hours, 47 minutes and 50 seconds).

An important observation from the results is the effect of pattern complexity on the process time. Taking an example, image A took 3 seconds to process compared to only 2 seconds for image B although the former is smaller. The complexity of image A characterised by its structural statistics contributed to the extra time taken. Looking at the large images (images P-T), the total processing time is very much contributed by t_2 (with up to 99.6% for image T). This is due to the increase in both size and structural statistics of the images. However, the increments are not entirely linear. t_1 if carefully observed, exhibits a seemingly linear relationship with the image size. On the other hand, t_2 increases dramatically with an increase in complexity.

7.2.4 Sub-image Query and Retrieval

Chapter 5 was dedicated to the approaches used to extract sub-images in an image which are regarded as separate objects-of-interest. The ability to extract these regions is the first major obstacle within the scope of sub-image query and retrieval. As far as the research is concerned, more effort has to be put into solving the problem in order to properly separate objects-of-interest. In most cases, images are either over-segmented or under-segmented. However, as part of the research, a potential scenario has been structured and tested on some images.

The results are quite encouraging for some images, but worked poorly on others. Basically, the main problem lies in the fact that craquelure patterns are so random and unpredictable that it is highly challenging to segment these patterns according to a schema with no standard rule.

		Structural Statistics				Speed Analysis				
Image	Width x Height	Length	# of Crack-networks	# of Nodes	# of Lines	t_1 (sec.)	(t_1/t_{total})x100	t_2 (sec.)	(t_2/t_{total})x100	t_{total} (sec.)
A	314 x 205	2916.66	76	166	413	1	33%	2	66%	3
B	320 x 203	1332.71	24	22	70	1	50%	1	50%	2
C	256 x 256	3574.21	36	118	275	1	66%	0.5	33%	1.5
D	256 x 256	3234.52	36	89	217	1	50%	1	50%	2
E	315 x 209	2102.66	55	2	59	1	50%	1	50%	2
F	317 x 208	2513.61	32	68	170	1	50%	1	50%	2
G	316 x 209	2252.25	24	134	306	1	50%	1	50%	2
H	317 x 211	3570.69	17	93	203	1	66%	0.5	33%	1.5
I	323 x 209	3028.13	37	109	256	1	50%	1	50%	2
J	268 x 398	4998.00	49	225	519	2	40%	3	60%	5
K	268 x 398	10245.92	190	693	1619	2	20%	8	80%	10
L	268 x 403	3459.05	23	90	205	2	50%	2	50%	4
M	392 x 496	11199.63	110	948	2089	3	18.8%	13	81.2%	16
N	512 x 512	14903.52	117	551	1232	5	21.7%	18	78.3%	23
O	512 x 512	15079.65	102	562	1252	4	15.4%	22	84.6%	26
P	911 x 409	26648.55	134	1282	2761	7	12.1%	51	87.9%	58
Q	605 x 1188	42072.95	560	5682	12770	14	2.5%	551	97.5%	565
R	602 x 1356	64216.38	1338	6260	14457	16	1.4%	1175	98.6%	1191
S	1211 x 2377	177610.88	2549	22614	51134	51	0.5%	9611	99.5%	9662
T	1413 x 2207	197064.65	3194	23870	54278	60	0.4%	13610	99.6%	13670

Table 7.1: Analysis of processing speed for various images. The effects of the image size and the complexity of crack patterns are observed. t_1 corresponds to the time taken in seconds to fully detect and prune crack patterns while t_2 represents the time taken to perform pattern structuring, merging, feature extraction and classification. t_{total} is the total processing time.

7.3 Summary

This particular chapter has discussed the "end-product" of the whole research, which integrates all the small technical modules into an implementable application.

Two types of query style are presented which are the *query by image example* and *query by text*. In the implementation of *query by image example*, three different formats can be applied. The first, named *query using feature matching*, uses the feature vector of each crack pattern as a descriptor or cue for similarity/dissimilarity measure. The second technique utilises the fuzzy set or class memberships of craquelure patterns as cue, while the third takes a more straightforward matching approach by just retrieving all patterns which possess similar maximum class membership as the query pattern. The retrieval engine has the ability to retrieve images or sub-images depending on the way information is coded in the feature file.

Besides the query-retrieval functionality, the need to perform pattern analysis is also seen as a very useful functionality for the system. Therefore, it was shown how craquelure patterns can be analysed through a functionality which allows users to randomly query for statistical and classification information regarding a particular pattern of interest.

Algorithm speed has also been analysed and it is discovered that pattern structuring, merging, feature extraction and classification modules contributed the most to the overall processing time compared to the crack detection and pruning modules.

Chapter 8

Conclusions

The essence of this research is now recapitulated by a summing up of the contributions. Ideas and suggestions for future work are also discussed in the final part of this chapter and closing section of the research.

Computer vision has been attracting considerable attention from art-related communities and institutions. For various applications, computer vision now plays an increasingly important role especially in quality evaluation of art images, as a tool for art analysis, virtual enhancement as well as restoration and also image retrieval. Museums and art galleries are beginning to digitise their collections to make them available on the web for the public and also for internal use within the museums' or galleries' own environment.

Digitisation of art collections provides faster and more efficient storage, thus giving a new dimension into methods of information retrieval within the environment. Instead of storing information in a traditional manner, the ability to store it digitally has opened the path for further manipulation of the technology, where digital preservation and restoration can play their parts. Many collections of paintings and artefacts originated centuries ago and are in need of preservation and restoration to make sure that their physical appearance is maintained. Manual recording and detection of aging seem far from efficient, with the growth of collections; electronic-based approaches seem to be the best choice.

The motivation for this research originated from the fact that craquelure in paintings can be an important element in judging authenticity, use of materials, environmental and physical impact because these can lead to different craquelure patterns. Mass screening of craquelure patterns can provide a better alternative platform for conservators to identify causes of damage. The ability to screen the whole collection semi-automatically is believed to be a useful contribution to preservation. Crack formations are influenced by factors which also

relate to the wooden framework of the paintings. One of the image-based requirements from museums, is to automatically classify craquelure (cracks) in paintings for the purpose of aiding damage assessment using non-destructive monitoring and testing.

This research provided a content-based approach towards analysing craquelure patterns. The target was to provide a general but concise guide to the steps needed to provide analysis of craquelure patterns through operations such as segmentation, retrieval and classification, which have received less attention from the computer vision community, mainly because of its less significant importance.

8.1 Contributions and Summary

The idea of implementing a content-based retrieval system for the analysis of crack patterns is not a particularly new idea (see Chapter 1). However, in terms of research and implementation, there has been no thorough examination of the issue. This research has gone through the relevant steps needed to produce such a functionality. However, there are still many problems and obstacles. Knowing the problems is in a way also considered a good contribution to the research. Here, the contributions of this work are summed up.

The first crucial step in the research was to detect craquelure patterns, i.e. segmenting the elongated shapes of cracks from the image background. To achieve this, the top-hat operator was used as an enhancement mechanism and grid-based thresholding, using the Otsu technique was used to segment the patterns. Grid-based thresholding is necessary in order to cope with uneven illumination and lighting effects within the image. Segmented crack patterns were thinned to one-pixel wide.

The thinned contours of the crack patterns were then converted from pixel representations into an easily manipulated representation. A hierarchical Freeman chain-code based technique, which allows controlled pruning and statistical measurements of craquelure structural data, was developed. The beauty of this approach lies in the ability to extract statistical measurements, such as number of pixels, number of nodes and number of line segments. Conservative approximations of crack-networks through the use of the minimum bounding rectangle (MBR) and the rotated minimum bounding rectangle (RMBR) were introduced to allow a more general characterisation of crack pattern descriptors, since detailed information about a shape had been translated into a more simplified representation. All this was made possible through the hierarchically structured data built through the *crack following* routine. Another positive point of the technique is that it allows easy pruning of crack

patterns for the purpose of filtering noise and eliminating insignificant crack patterns. A down-point of the approach is the amount of memory used to structure the crack patterns. The more complicated a crack pattern is, the more memory is needed to produce the structured representation. This however can be overcome by powerful machine capabilities and optimised programming.

The most challenging part of the whole research was to segment crack patterns into objects-of-interest, which was attempted using a two-stage approach. The highly random nature of crack patterns made this task extremely difficult. The technique implemented benefited from the conservative approximation of a crack pattern using either the MBR or the RMBR. MBR has the advantage of being less time consuming to compute while the RMBR is a more accurate pattern approximation, since it holds orientation information. Two merging algorithms were introduced; the *merge and expand* (M&E) and the *label and merge* (L&M) to merge two crack patterns under consideration.

The first stage of the crack pattern grouping algorithm involves merging patterns which satisfy a proximity rule, which is determined by the intersection of pattern aproximations. Results showed that most unidirectional crack contours are not successfully merged in this stage, although they are close in proximity to each other and supposedly belong to the same object-of-interest. Since their bounding boxes do not intersect, they remain unmerged. To tackle this problem, a second stage was implemented, which tries to group patterns according to their characteristics or features. For that, agglomerative hierarchical clustering was utilised to cluster the feature points of all the merged crack patterns. The target was to group these features into distinct clusters based on the location of the pattern centroid, node density, axis of minimum inertia and dimension ratio. A technique to determine the optimum number of clusters was also introduced. The iterative technique relies on the second order differential of the minimum distance variance between cluster points. Results were quite promising on well-separated clusters. However, grouping combined line structures is very much an open issue, since it involves a high amount of subjectivity and human perception. The approach did not tackle this particular problem in too much depth because of the complexity.

The most distinct difference between the five crack patterns (i.e. circular, rectangular, unidirectional, spiderweb and random) is in terms of their orientation, which can be looked into in two ways, locally and globally. With both views originating from chain-code histograms, the local orientation concentrates on the line segment level, while global orientation refers to orientation at the crack-network layer or higher. From these concepts, unidirectionality measures are derived from the local and global orientation histograms. Significance

measures are also derived primarily from lengths of line segments as an indication of how influential or meaningful these line segments are globally. Another form of orientation histogram was deduced from a straight line representation of a crack pattern called the *quantised gradient histogram* (QGH) which was utilised to compute unidirectionality and rectangularity. Histogram shape filters were also used to create more features from orientation histograms and the QGH. Besides histogram-based features, structural related descriptors were also developed from node and line segment densities. Early experiments using five model patterns showed promising results in terms of these features' discriminating powers. However, none of the features show independency in separating all five classes.

A labelled test set was built to characterise crack patterns from the five classes for the purpose of classification. Measures of discriminant powers were evaluated using the Fisher ratio and performed on all sixteen selected features. In general, the results showed that histogram-based features had better discriminating powers compared to the structural features. Classification using the k-NN and the average distance k-NN revealed low correct classification rates of 60.31% and 58.78% respectively. Analysis unveiled low correct classification rates for the circular, spiderweb and random classes, which were the main causes of the poor performance.

As a solution to this, two major modifications were made to the classifier by first converting it into a three-stage classifier and secondly by excluding the random class from the test pattern, thus letting the classifier to decide on classifying input patterns into only four classes. The random pattern was believed to create confusion within the classifier for its vague definition and the absence of unique descriptors. Using the newly modified classifier, vast improvement occurred in the rate of correct classification, 76% recorded for the k-NN and 82% for the average distance k-NN. The class-specific correct classification rates also improved. Experiments revealed that the optimum values of k are 4 and 5 for the average distance k-NN rule.

From the analysis, it can be concluded that most of the features are best used to distinguish patterns which are based upon straight lines versus ones based on curvy-lines. There were insufficient features to differentiate circular patterns from spiderweb patterns based on the low Fisher scores in the third stage of the classifier. This was caused by the fact that the features used do not take pixel position properties into account since orientation alone is not enough to differentiate the two classes. The fuzzy class membership in a way helped in dealing with subjective perceptions, where patterns are not exclusively assigned to one class.

A prototype system which offers query-retrieval functionalities as well as craquelure pat-

tern analysis was explained. Within the query-retrieval framework, there are two types of queries, *query by image example* and *query by text*. *Query by image example* can be implemented in three ways, using different matching cues. First, *query using features* uses feature vectors as cues. Another way is by using fuzzy sets as cues while the final alternative takes maximum class membership as matching cues. Obviously, the three methods differ in terms of the type of information sought. In order to look for "patterns like this query pattern", feature cues are the most suitable, while if the question is "patterns possessing class elements like this query pattern", fuzzy sets are the most suitable form of matching cue. On the other hand, for a query such as "patterns with similar class to this query pattern", the maximum class membership should be used as a cue. Another means of querying for information is by straightaway typing in the class sought instead of entering an image as input.

One useful scenario in which this implementation can be useful is where a conservator tries to relate a crack pattern with its cause factor. With the list of matching results, a user can monitor the proportion of images with similar wooden structures at the back, thus giving a rough idea of how influential these wooden structures can be in producing unwanted cracks. A user is interested in monitoring all paintings with possible circular pattern cracks, which can be the direct effect of physical impact, circular crack patterns can be used as query. This will enable conservators to perform physical conservation to the detected areas or enforce careful handling procedures for the painting whenever it needs to be transported.

Another useful functionality offered by the prototype system is the ability to perform analysis in order to study or detect interesting craquelure elements in the images. The input image is segmented into objects-of-interest, which are then analysed, producing statistical measures and classification results. This functionality is seen as equally important as the query-retrieval functionality, since it allows images to be analysed separately. This way, peculiar areas within an image can be spotted and studied.

The contributions of the research have been summarised in a concise fashion. In general, the objectives of the research were met. As in much research, this work has confronted obstacles since the days of its infancy, and being able to identify problems is no less important. The next section discusses possible improvements as well as other applications that may benefit from the research.

8.2 Future Directions

Steps can be taken in the future to improve and allow other possible applications. Width is considered important, especially in determining the layer at which cracks form. The ability to preserve width information at the detection stage will enable more advanced analysis of crack patterns. This will open up more options for high-level feature extraction.

A more advanced crack detection approach can also be utilised to capture patterns with varying widths. Although multi-width and multi-orientation crack detection have been briefly touched in Chapter 3, they are not fully implemented. Detection of crack patterns in colour images should also be researched in more depth. In the current implementation, colour images are simply converted to monochrome in order to suppress colour information. However, this action does not suppress brush stroke patterns which proves to be a nuisance (see Figures A.4, A.6 and A.8). The ability to differentiate crack patterns from other unwanted patterns in colour images is highly desirable.

More effort has to be projected into the problem of segmenting crack patterns into objects-of-interest, based on difficulty and importance. Segmentation has always been a hot topic and the issue of segmenting a group of line segments into meaningful sub-groups has presented researchers with a very challenging task. In the context of craquelure analysis, being able to segment patterns "correctly" will produce a more reliable automatic system. As observed in Figures A.2 and A.6, on a large image the merging technique failed to perform due to the very complex and random nature of the crack patterns. A more intelligent strategy that relies on prior knowledge and case-specific rules might be the answer to this problem which involves a huge amount of subjectivity and human perception.

The current approach towards the extraction of features relies on two views, orientation and structural statistics. The use of more high-level features, such as those which deal with spatial information on crack contours can improve the correct classification rate. Texture analysis can also be studied for use in craquelure analysis.

More effort has to be put into optimising the algorithm for execution on large images since based on the experiment shown in Section 7.2.3.1, the performance of the algorithm in terms of execution time degrades quite dramatically as the image size and complexity increases. Museum collections which include X-ray and visible images are very large in size (some exceeding 10000x10000 in pixel resolution), thus requiring efficient algorithm executions. Parallel processing might be the answer to the problem.

In terms of the system itself, a more structured and systematic way of storing features and

the metadata of images will present developers and users with a more flexible interface to the system. This research has concentrated more on the core computer vision technicalities but lacks development from the *human-computer interaction* (HCI) point of view. Having a good user interface is highly desirable.

The main aim of the research was to study the possibilities of implementing content-based approaches into the analysis of craquelure patterns - which shows good potential for the future. However, one element this research did not cover is user evaluation which is seen as important to validate the results and as a means of getting useful feedback. Thus, collaboration among different domain experts is essential. Historians, conservators, art-experts and software engineers are among the professionals who can be included in the research circle.

Last but not least, this research has shown great potential in the area of craquelure analysis using content-based analysis. With the existence of such a system in the future, monitoring can be made in a more efficient and non-destructive way. From a wider research perspective, this research is potentially useful for other applications, especially areas of research related to computer vision: medical imaging and geographical information systems (GIS) are two good examples.

Appendix A

Analysis on Large Images

Sample results on very large images are as shown here. All images are processed on a Red Hat Linux machine with six 700Mhz Xeon Processors and 1.7GB 133MHz ECC SDRAM.

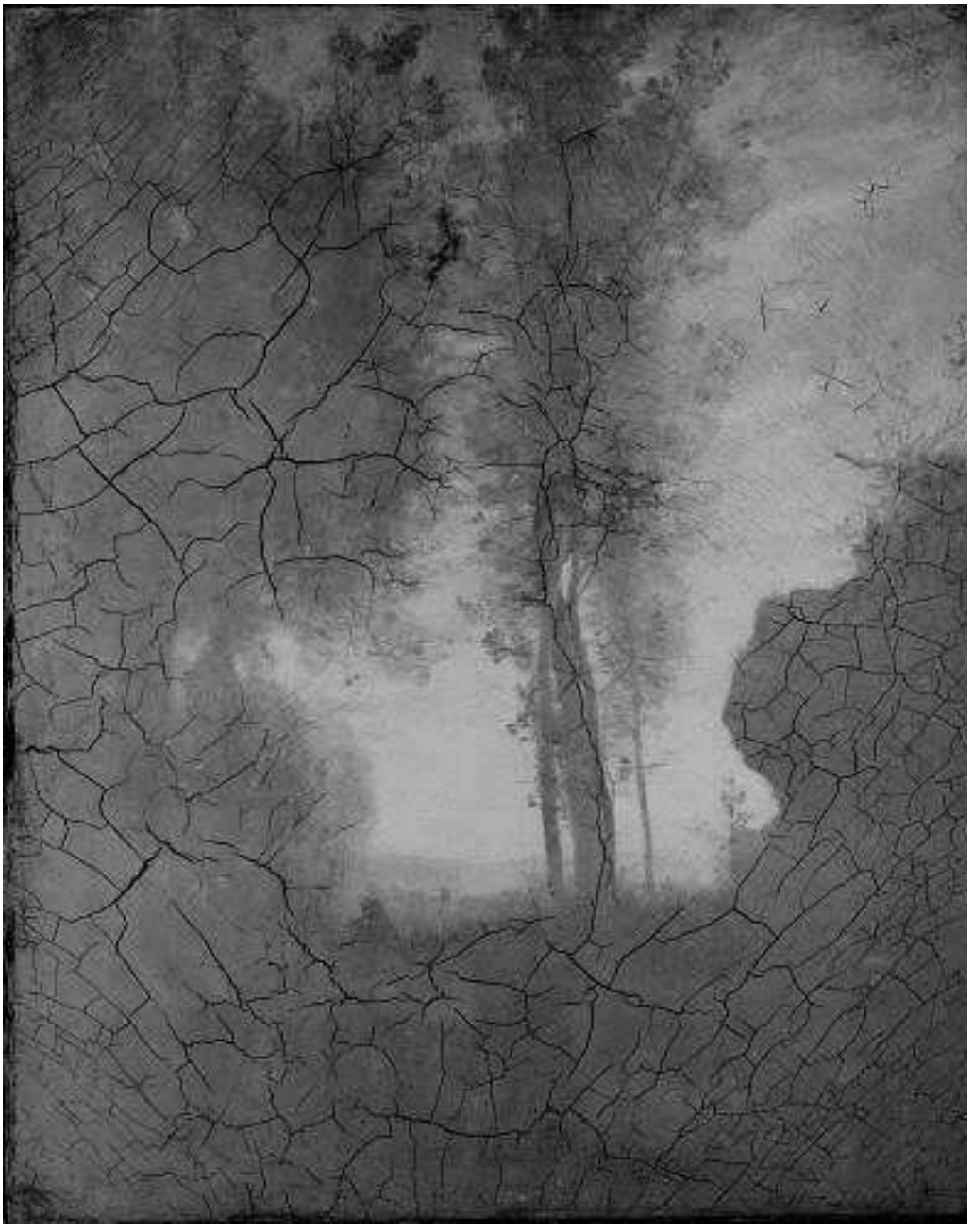


Figure A.1: An X-ray image, 392 x 496 in dimension. The image scaled for display.

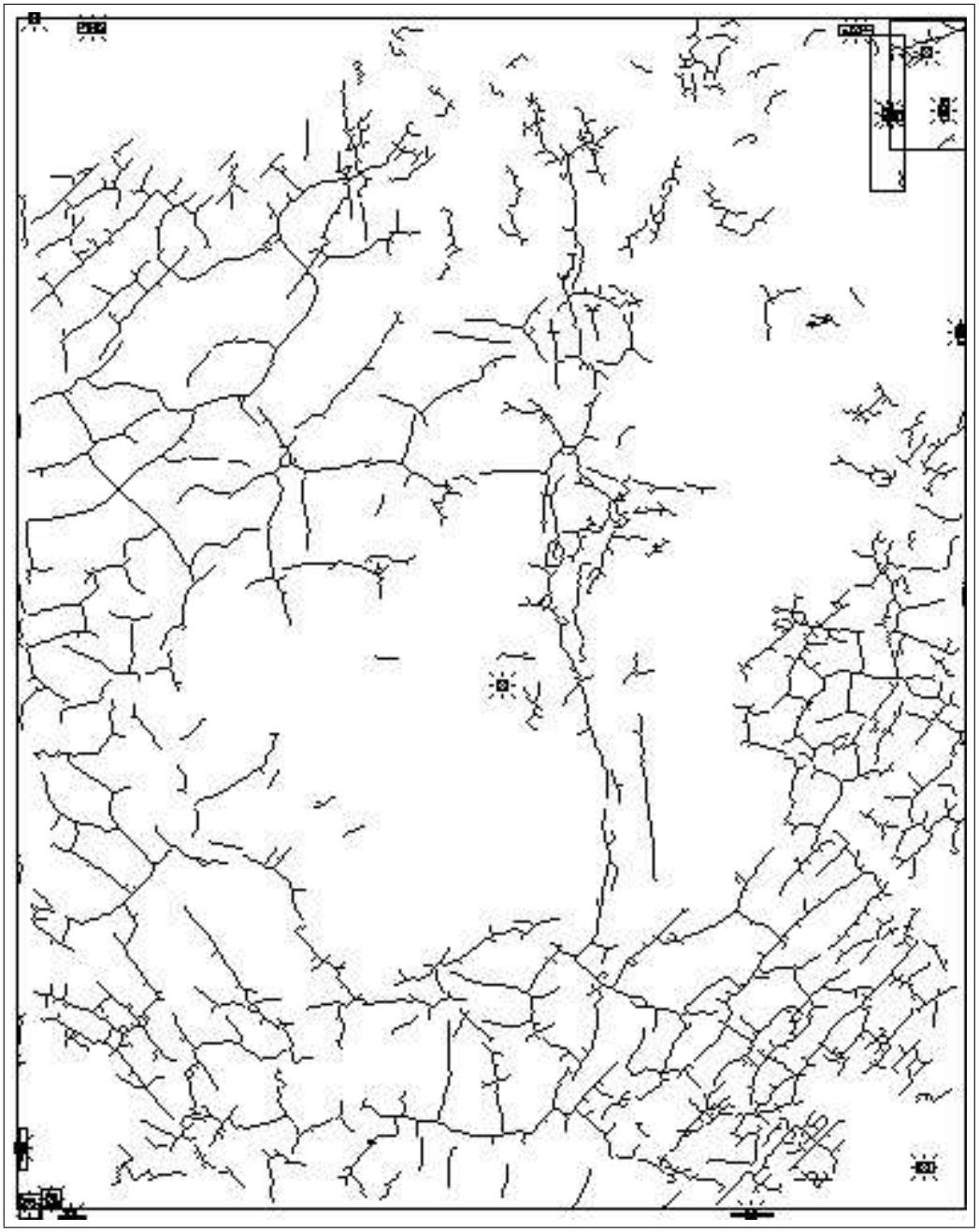


Figure A.2: Crack detected version of Figure A.1 with objects-of-interest displayed in MBRs. The whole process took 16 seconds to complete.

Figure A.3: "The Virgin and Child in an Interior" (Jacques Daret, National Gallery of London). 1413 x 2207 in dimension. The image scaled for display.

Figure A.4: Crack detected version of Figure A.3. Notice the edges and painting details such as brush stroke patterns are also detected due their crack-like characteristics. The process took 3 hours, 47 minutes and 50 seconds to complete.

Figure A.5: An extract from "Portrait of a Woman" (Salting Bequest, National Gallery of London), 602 x 1356 in dimension. The image scaled for display.

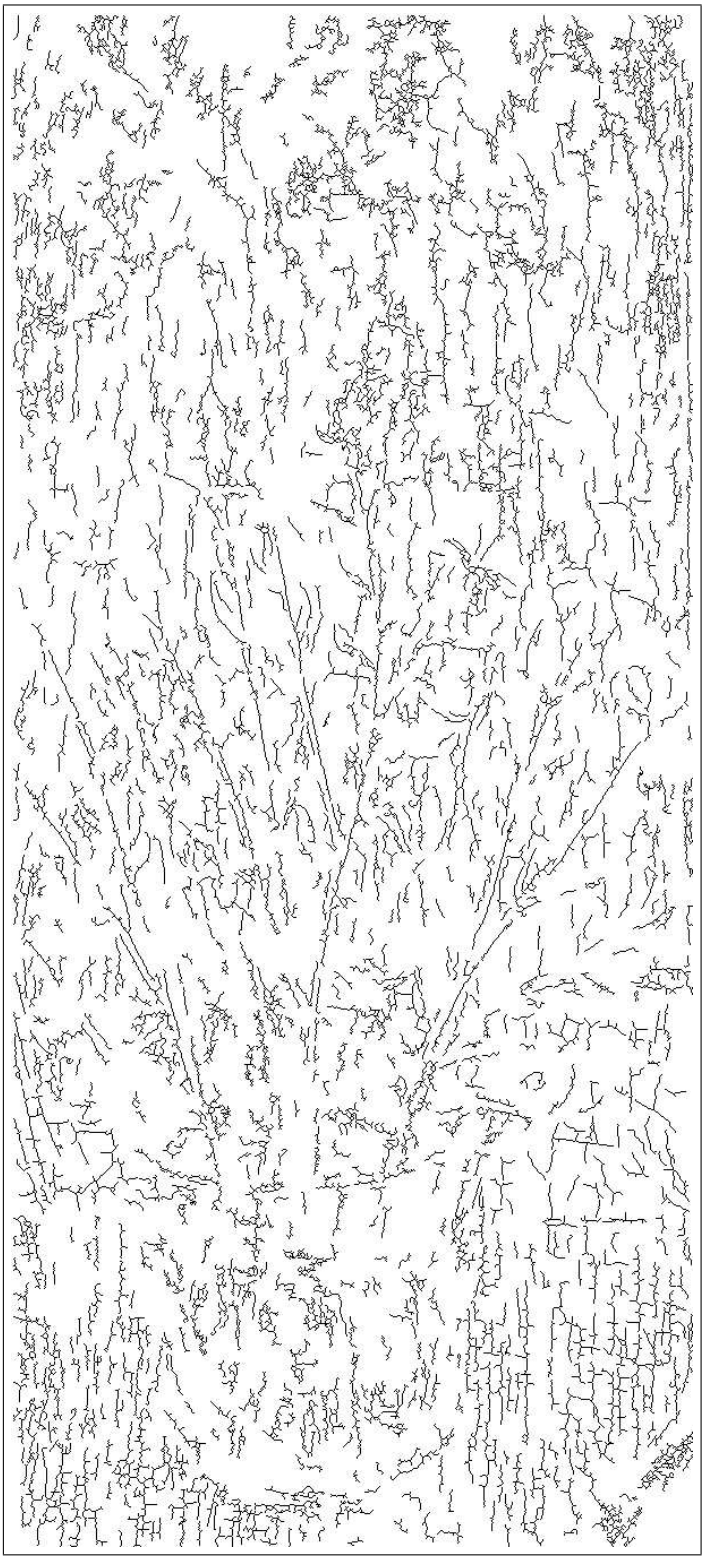


Figure A.6: Crack detected version of Figure A.5. Notice the edges and painting details such as brush stroke patterns are also detected due their crack-like characteristics. The process took 19 minutes and 51 seconds to complete.

Figure A.7: An extract from "Portrait of a Woman" (Salting Bequest, National Gallery of London), 1211 x 2377 in dimension. The image scaled for display.

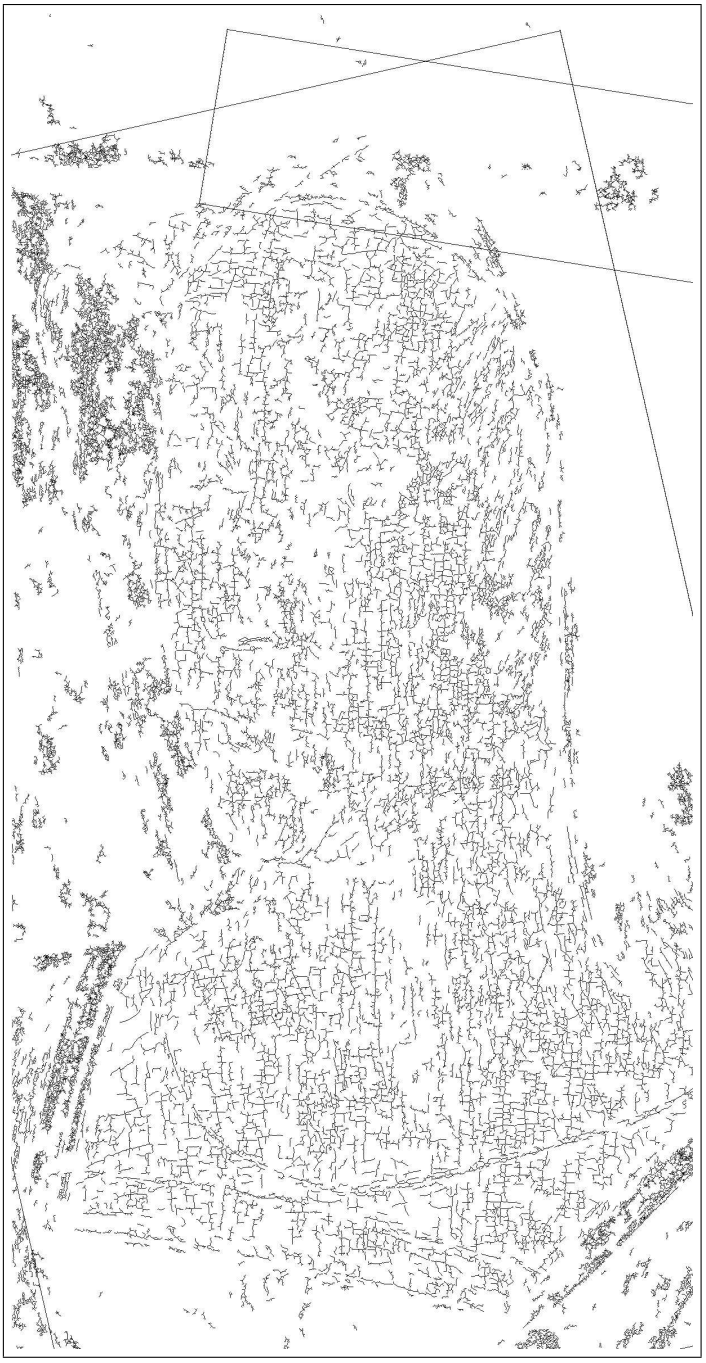


Figure A.8: Crack detected version of Figure A.7. Notice the edges and painting details such as brush stroke patterns are also detected due their crack-like characteristics. The process took 2 hours, 41 minutes and 2 seconds to complete.

Appendix B

Query for Similar Pattern

Example screenshots of the three versions of query by image example are shown.

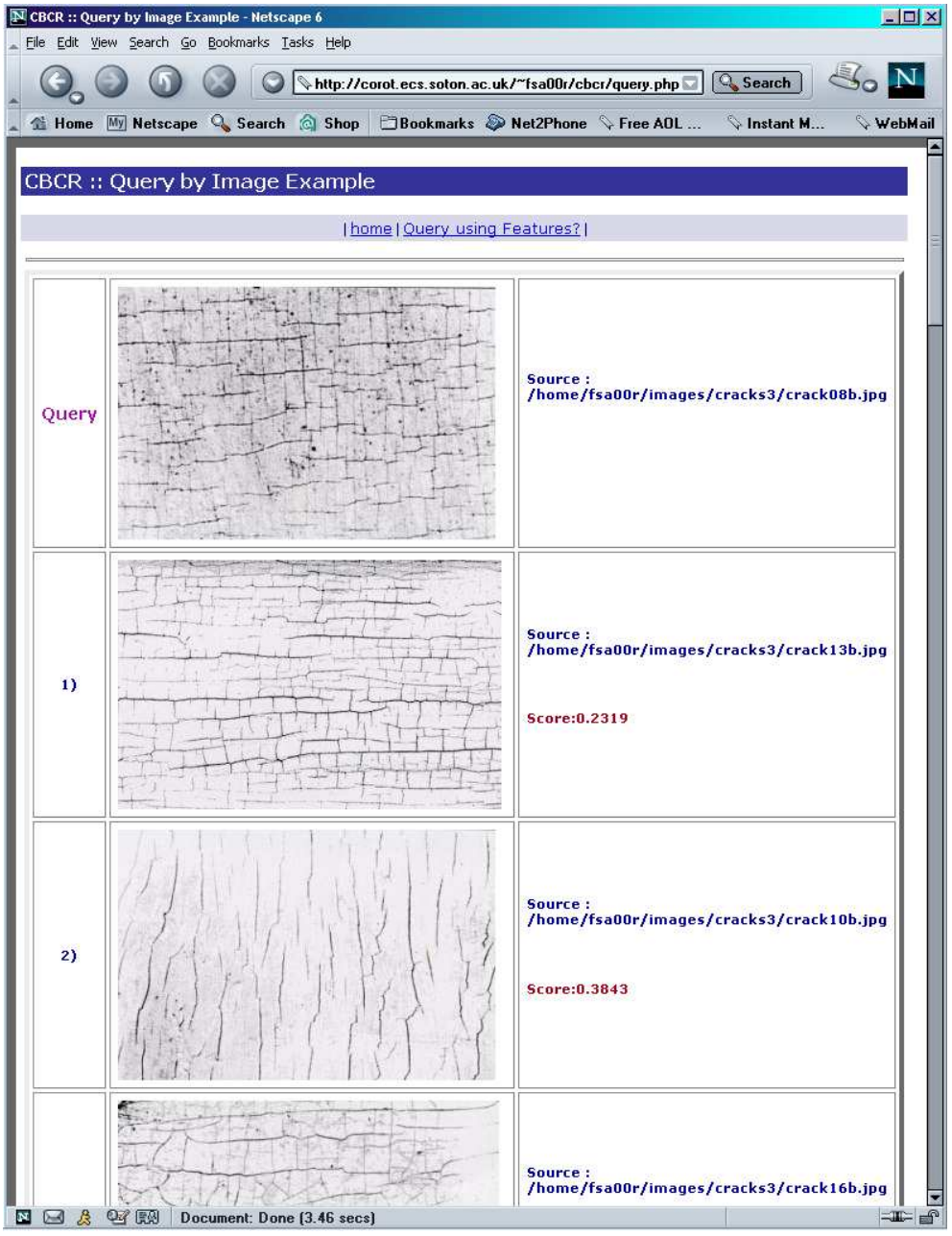


Figure B.1: A screenshot of the result viewer for *query using a feature vector* as cue.

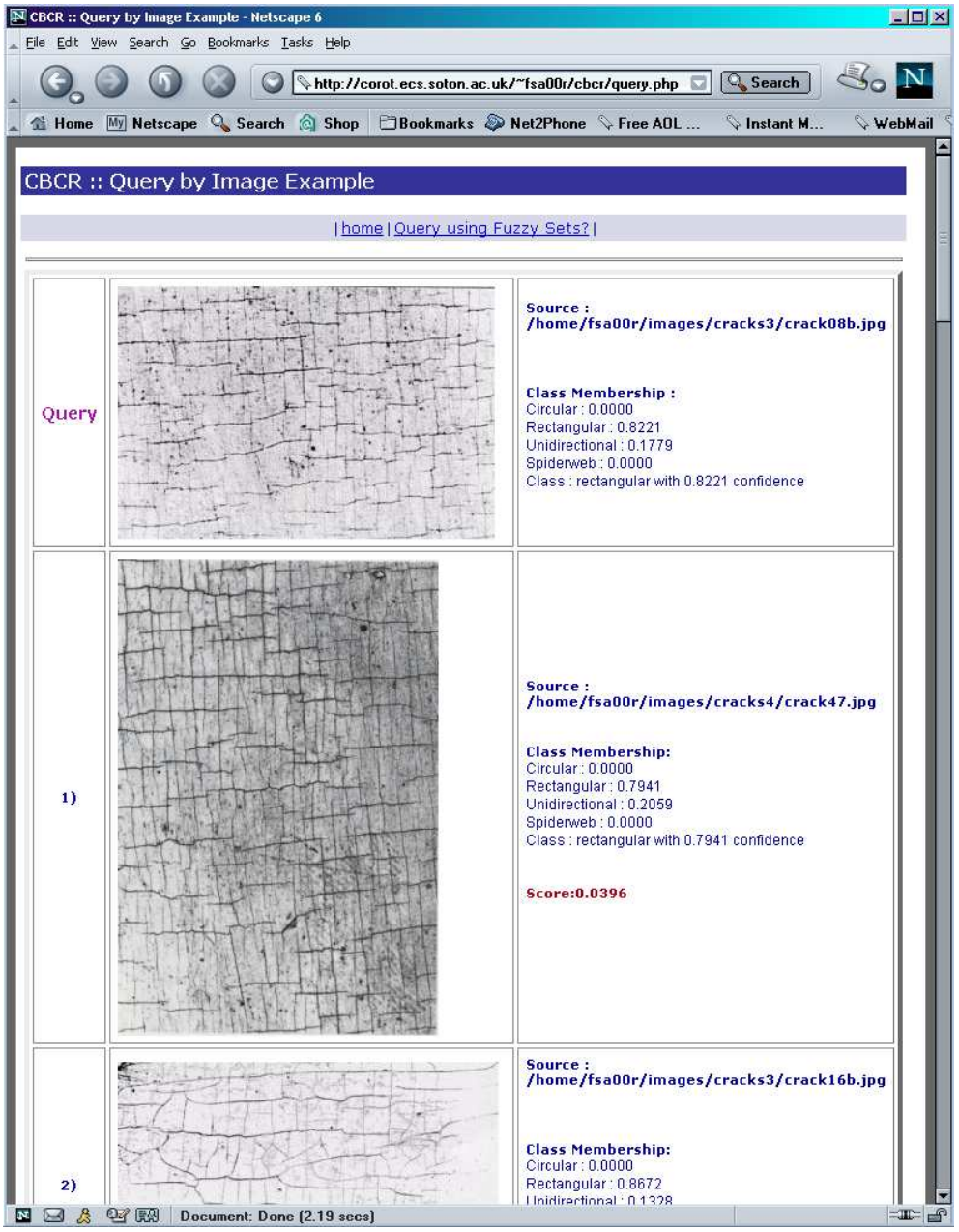


Figure B.2: A screenshot of the result viewer for *query using a fuzzy set* as cue.

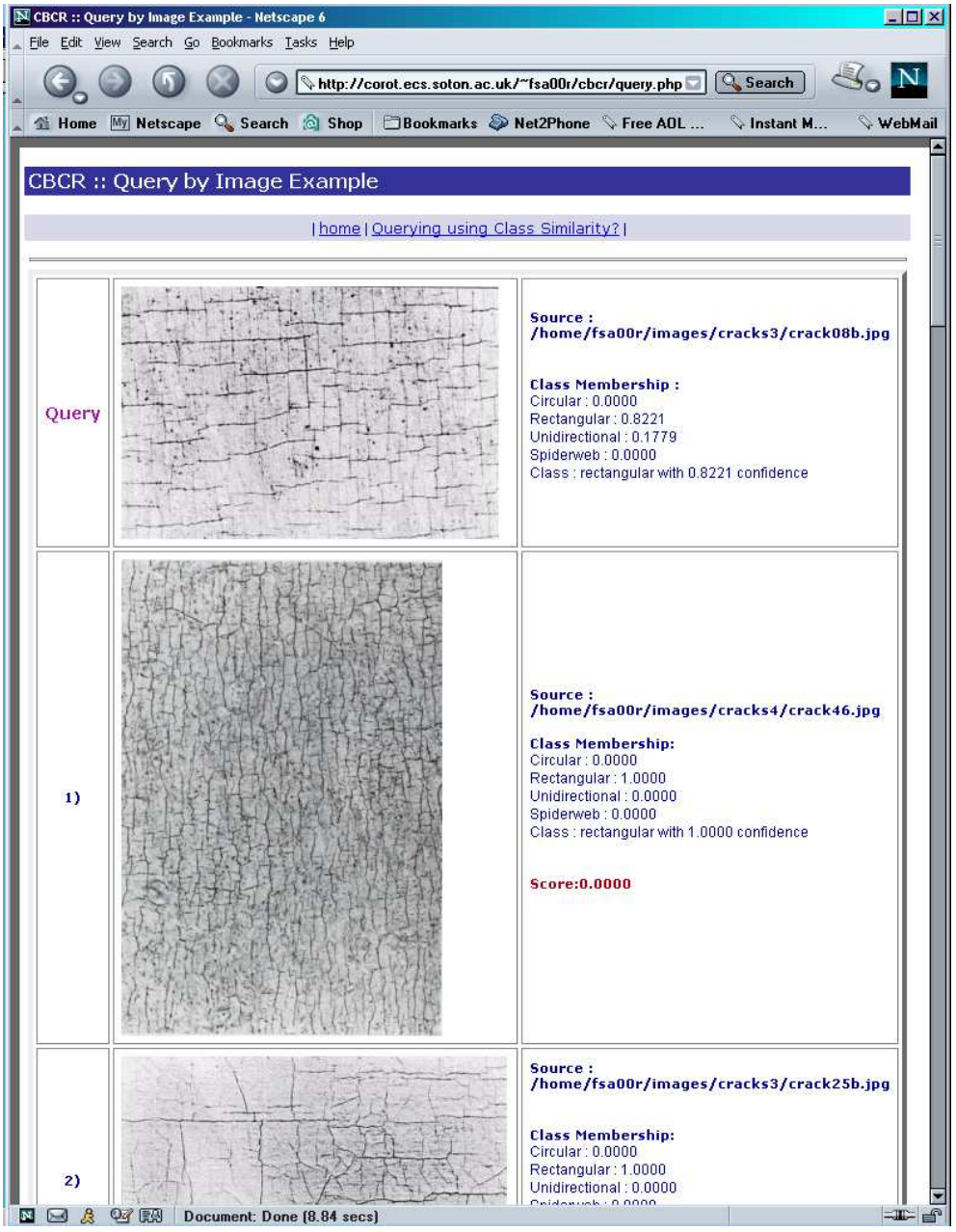


Figure B.3: A screenshot of the result viewer for *query using a class membership* as cue.

Appendix C

Craquelure Pattern Analysis

Example screenshots of the functionality to perform *craquelure pattern analysis* are shown. The main window displaying the image of interest with associate information is shown in Figure C.1. Figures C.2, C.3, C.4 and C.5 show windows which appear as the objects-of-interest (shown by the boxes) are clicked on.

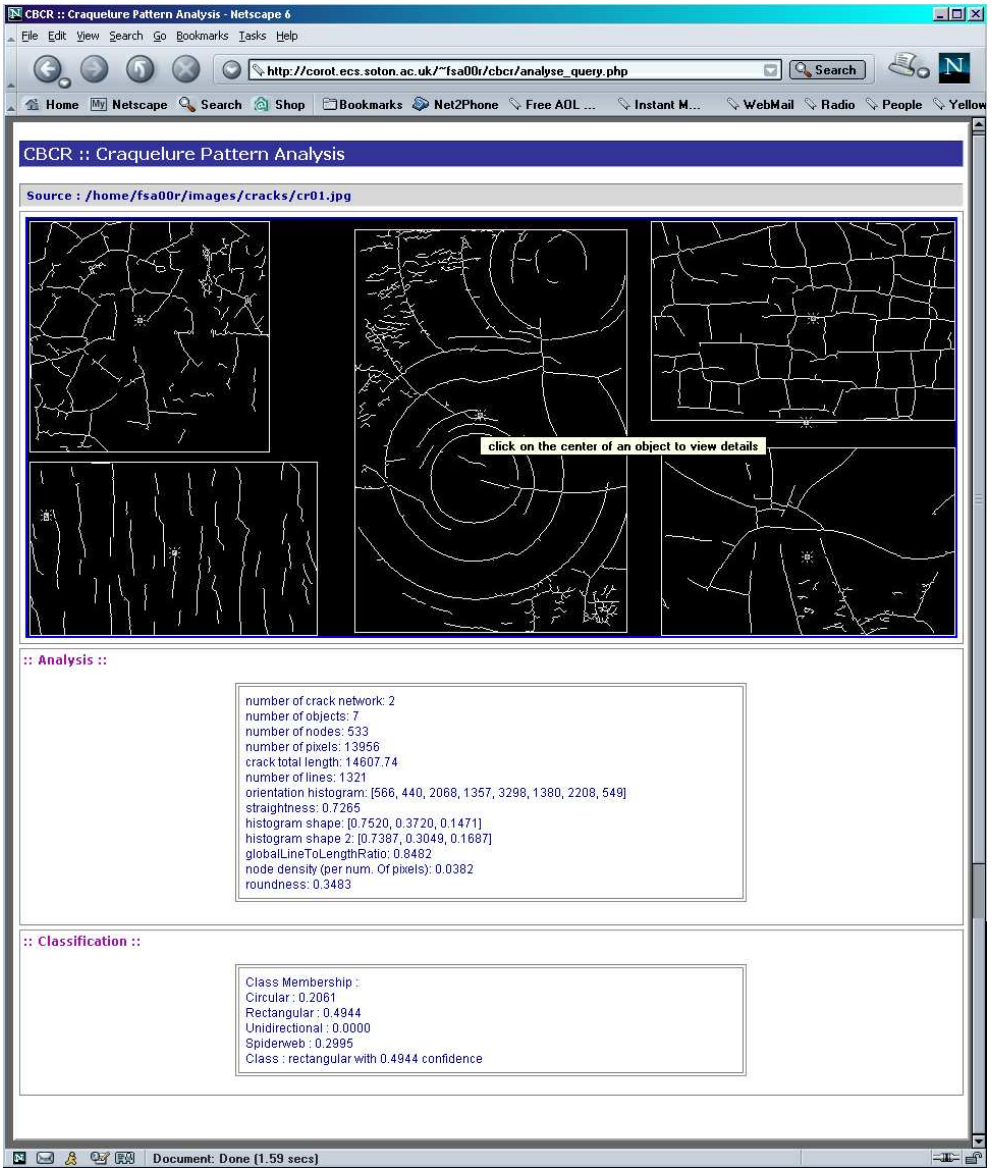


Figure C.1: A screenshot of the main result viewer page for *craquelure pattern analysis* functionality,which allows users to view statistical and classification information of an image of interest. Information associated with objects-of-interest can be viewed by clicking on the objects' centroid.

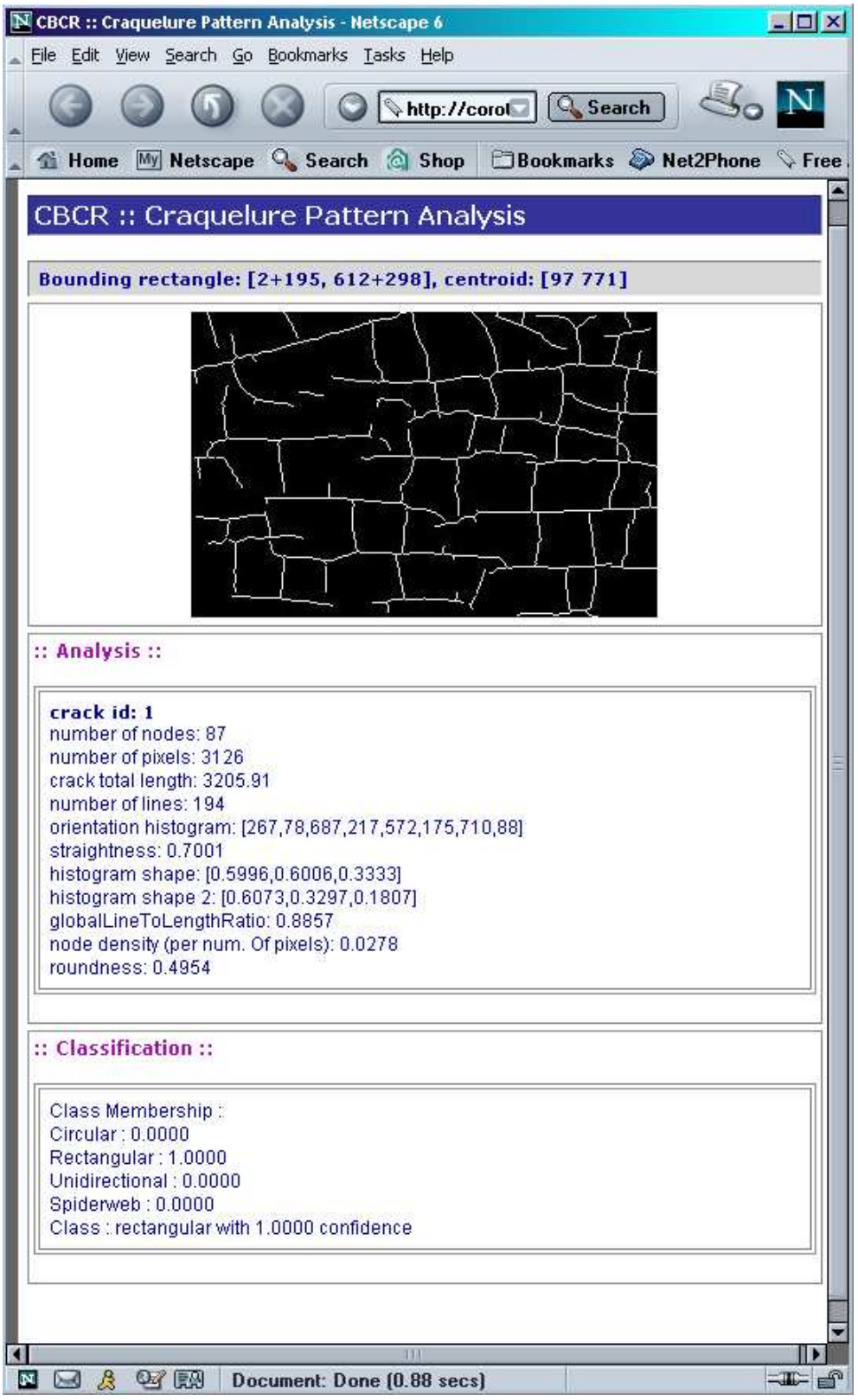


Figure C.2: Figure showing statistical and classification information associated with one of the objects-of-interest in Figure C.1.

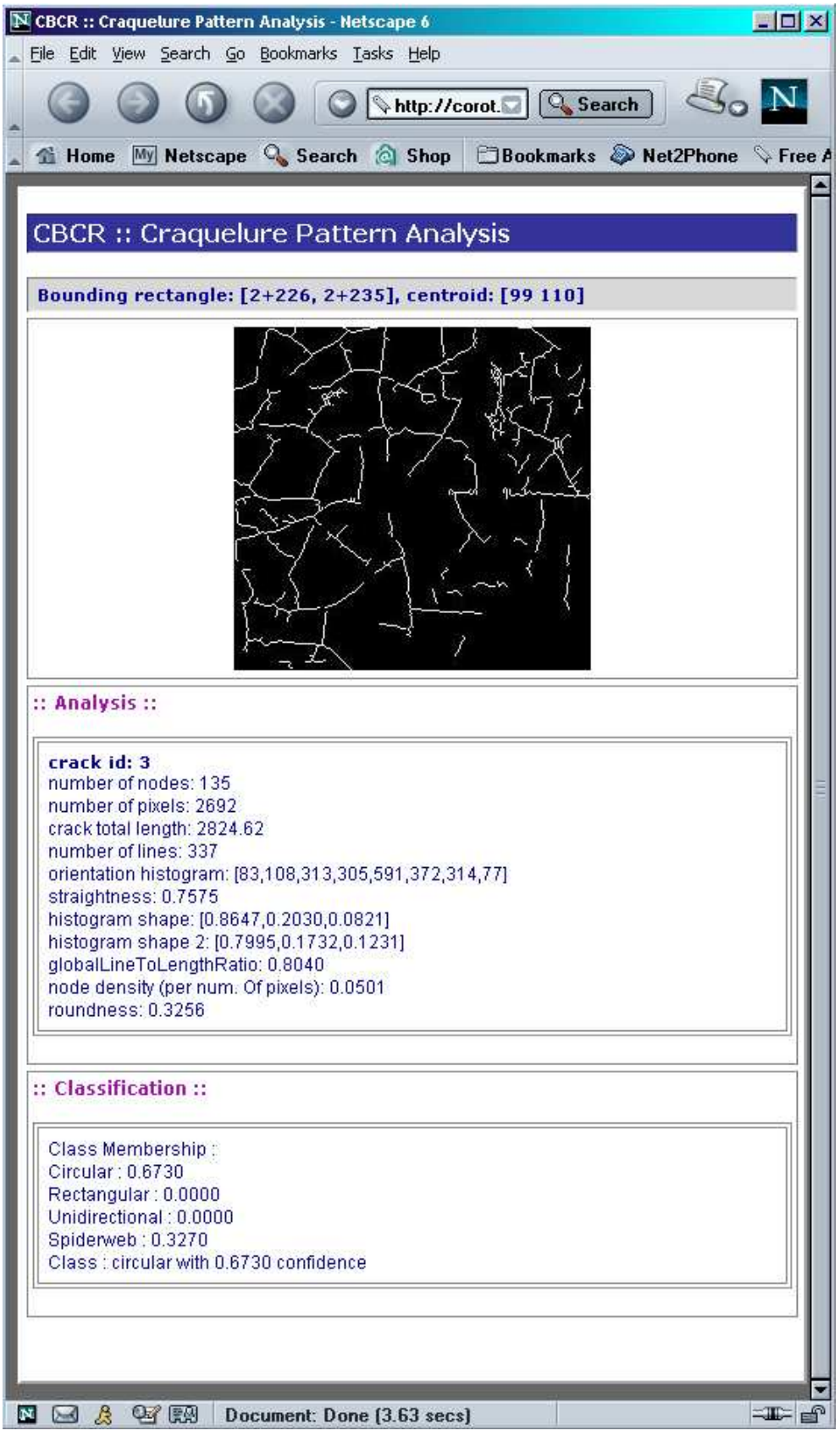


Figure C.3: Figure showing statistical and classification information associated with one of the objects-of-interest in Figure C.1.

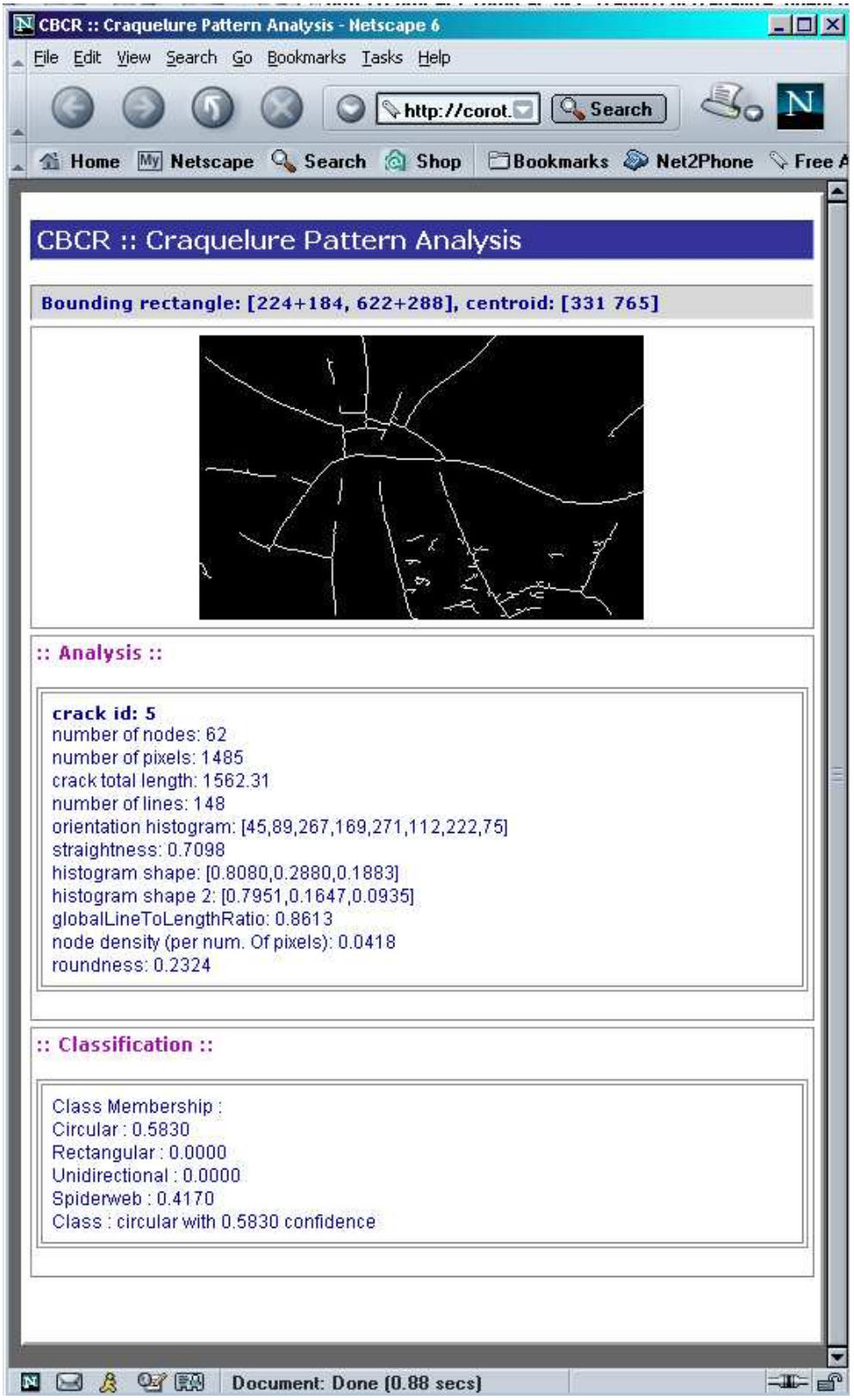


Figure C.4: Figure showing statistical and classification information associated with one of the objects-of-interest in Figure C.1.

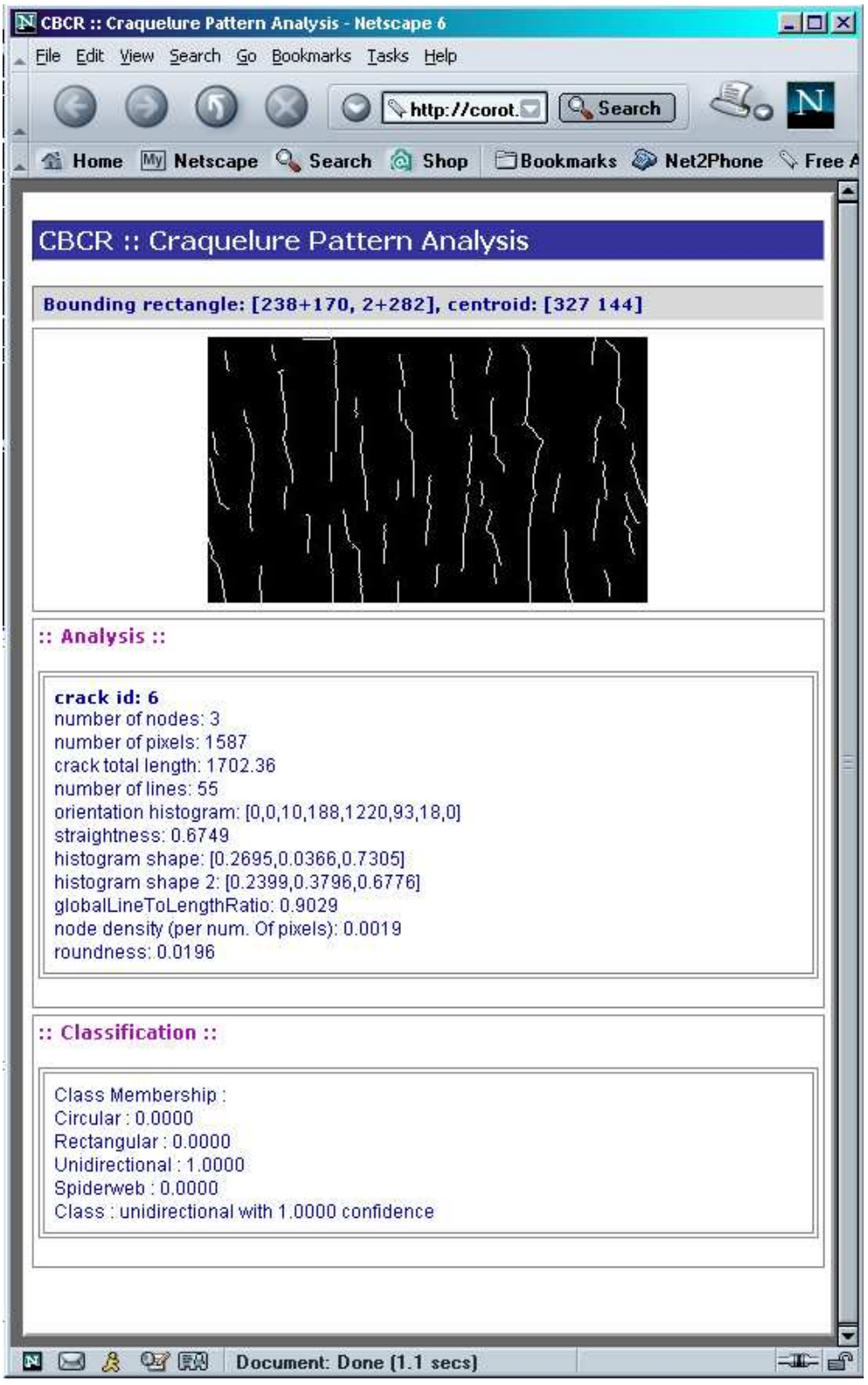


Figure C.5: Figure showing statistical and classification information associated with one of the objects-of-interest in Figure C.1.

Appendix D

System Interface

The system interface is written in C programming language and works under Linux operating system. It also requires VIPS image processing library to operate. The command-line user interface is divided into two main programs. The first, `gfmap` is a program to run modules related to classification. The second, `cbcr` controls modules for feature generation, pattern query and pattern analysis. The command to execute `gfmap` is as the following:

```
gfmap [options]
```

while the available `options` for `gfmap` are:

`-generateMap`	generate feature files and feature labels for classification
`-fisher`	compute Fisher Ratio for features
`-error`	compute correct classification rates

The command to execute `cbcr` is as shown below:

```
cbcr [options...] [uri/file]
```

and the available `options` for `cbcr` are:

`-generateFV -folder`	generate feature vector and class membership for all images in specified folder
`-generateFV -file`	generate feature vector and class membership for the specified image
`-queryImage -feature`	query by image example using feature vector as cue for matching
`-queryImage -fuzzy`	query by image example using fuzzy set as cue for matching
`-queryImage -class`	query by image example using maximum class membership as cue for matching
`-queryText [class]`	query by text
`-analyse`	craquelure pattern analysis

The following sections explain these two programs in more detail.

D.1 Classification and Data Generation

The classification and data generation modules are off-line processes which create data files for on-line processes.

D.1.1 Classification Module

Within the classification module, a file containing image sources and class assignments named `image_label.dat` is used by the *feature generator* to produce two other files containing feature vectors (`feature_map.dat`) and feature labels (`feature_label.dat`) associated to all the images listed in `image_label.dat`. The format of these three files are shown in Tables , D.2 and D.3.

```
[file source] [label]
      :
      :
```

Table D.1: Format of `image_label.dat`.

```
[number of sample] [n number of features] [c number of classes]
[label]
   :
   :
```

Table D.2: Format of `feature_label.dat`.

```
[number of sample] [n number of features] [c number of classes]
[feature 1] [feature 2] ... [feature n]
       :
       :
```

Table D.3: Format of `feature_map.dat`.

All these files are stored in the CL folder. Figure D.1 illustrates the process.

The command to run the program is as follows:

```
gfmap -generateMap
```

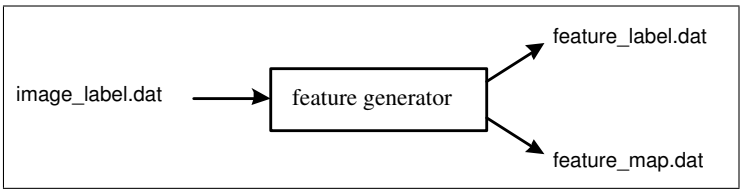


Figure D.1: The *feature generator* reads `image_label.dat` to generate `feature_map.dat` and `feature_label.dat`.

Once `feature_map.dat` and `feature_label.dat` have been generated, classification can be performed.

For experimental purposes, a functionality to calculate Fisher Ratio is also made available. The function reads `feature_map.dat` and `feature_label.dat`, calculate Fisher Ratio for all available features and prints the results. The command is as follows:

```
gfmap -fisher
```

Again, for experimentation, a function to compute correct classification rates based on the leave-one-out strategy is implemented. The function reads `feature_map.dat` and `feature_label.dat`, copies them in temporary arrays and performs classification for every training sample iteratively to estimate correct classification rate for specified values of k of the classifier. The command to run this program is as follows:

```
gfmap -error
```

D.1.2 Data Generation Module

The data generation module is tailored to generate feature vectors and class memberships for all processed images. Feature vector files are stored in the FV folder while class membership files are stored in FVC folders. A standardised naming schema is applied for both type of files where all feature vector files have names starting with `fv_` followed by the image filename (e.g. `fv_crack01.dat`). Class membership files are named with `fvc_` (e.g. `fvc_crack01.dat`). The format of the feature vector file and the class membership file are shown in Tables D.4 and D.5.

Input image(s) can be read individually as a file or collectively in a folder. The feature generator then constructs all the associate feature vector files and stores them in the FV folder. Information is then passed to the classifier which classifies the feature vectors by assigning class memberships and stores the class membership files in the FVC folder. Figure D.2 illustrates the flowchart of the data generation process.

```
[file source]
[height] [width]
[y-corner] [x-corner] [y-dimension] [x-dimension] [feature 1] ... [feature n]
    :
    :
```

Table D.4: Format of a feature file.

```
[file source]
[height] [width]
[y-corner] [x-corner] [y-dimension] [x-dimension] [class] [class 1] ... [class c]
    :
    :
```

Table D.5: Format of a class membership file.

The commands to perform data generation are as follows:

```
cbcr -generateFV -folder [uri]
```

for all images in `uri`, and

```
cbcr -generateFV -file [uri/file]
```

for a single image file.

The data generated can now be searched by the query engine and this is explained next.

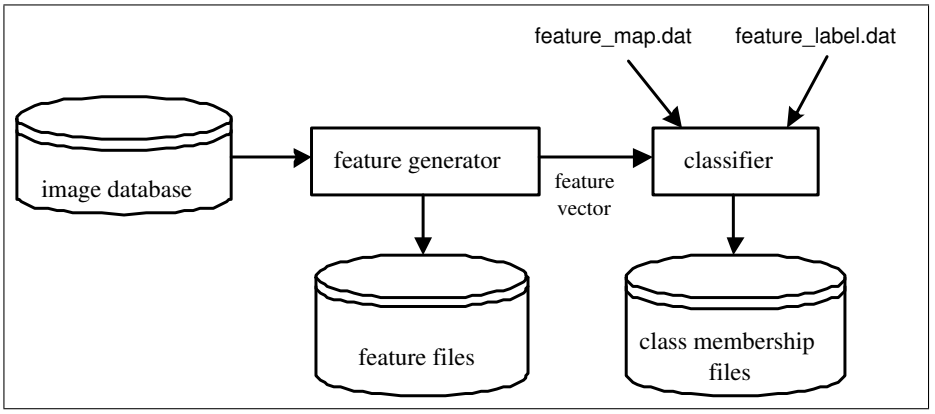


Figure D.2: Data generation process.

D.2 Query Engine

The query engine processes any queries issued by users. In general, there are three query options. The first allows users to enter an image into the system and query for "similar" images based on cues that will be explained later. The second type of query requests users to enter textual input as to the pattern class sought, while the third query requires an image as input for pattern analysis rather than for image matching. Any input goes through the query engine which determines the correct path to which the query should pass through. The forthcoming sections explain these query options in more detail.

D.2.1 Query by Image Example

The system assumes all the crack patterns detected in the input image as a single object-of-interest. The input image goes through a query processor that converts it into monochrome if it is a coloured image. This particular query option is further divided into three formats, which utilise different cues as matching factors. These formats are *query using feature vector*, *query using fuzzy set* and *query using class*. The feature generator first generates the feature vector of the input image. This needs to be done for all three cue types. The classifier will only get involved for *query using fuzzy set* and *query using class*, which needs the fuzzy set and maximum class membership as a matching factor. The details of these formats were explained in Chapter 7.

The commands to execute these programs are as follows:

```
cbcr -queryImage -feature [uri/file]
cbcr -queryImage -fuzzy [uri/file]
cbcr -queryImage -class [uri/file]
```

for *query using feature vector*, *query using fuzzy set* and *query using class* respectively.

D.2.2 Query by Text

This form of query takes textual input from users. Users can query for images with a certain maximum class membership. The command line to execute this program is as follows:

```
cbcr -queryText [class]
```

The options for `class` are:

`-c`	circular pattern class
`-r`	rectangular pattern class
`-u`	unidirectional pattern class
`-s`	spiderweb pattern class

The system then lists all the n matching images in the database.

D.2.3 Craquelure Pattern Analysis

The final form of query concentrates on the analysis of craquelure patterns rather than request for "similar" images. A user enters an input image which is then segmented into objects-of-interest. The main image as well as each object-of-interest are then analysed by the feature generator and the classifier. The command to run this program is shown below:

`cbcr -analyse` [`uri/file`]

The results are then displayed in the result viewer automatically on execution.

References

[1] A.K. Jain, C. Dorai, "Practising Vision: Integration, Evaluation and Applications," *Pattern Recognition*, vol. 30, no. 2, pp. 183–196, 1997.

[2] Y. Rui, T. Huang, S. Chang, "Image Retrieval: Current Techniques, Promising Directions and Open Issues," *Journal of Visual Communication and Image Representation*, vol. 10, no. 4, pp. 39–62, April 1999.

[3] S. Chang, A. Eleftheriadis, R. McClintock, "Next-Generation Content Representation, Creation and Searching for New-Media Applications in Education," *Proceedings of the IEEE*, vol. 86, no. 5, pp. 884–904, May 1997.

[4] M. Flickner, H. Sawhney, W. Niblack, J. Ashley, Q. Huang, B. Dom, M. Gorkani, J. Hafner, D. Lee, D. Petkovic, D. Steele, P. Yanker, "Query by Image and Video Content: The QBIC System," *IEEE Computer Magazine*, vol. 28, no. 9, pp. 23–32, September 1995.

[5] S. Mehrotra, Y. Rui, M. Ortega-Binderberger, T.S. Huang, "Supporting Content-based Queries over Images in MARS," in *IEEE International Conference on Multimedia Computing and Systems*, 3-6 June 1997, pp. 632–633.

[6] W.Y. Ma, B.S. Manjunath, "NeTra: A Toolbox for Navigating Large Image Databases," in *Proceedings of the International Conference on Image Processing*, 1997, pp. 568–571.

[7] J.R. Bach, C. Fuller, A. Gupta, A. Hampapur, B. Horowitz, R. Humphrey, R. Jain, C-F. Shu, "The Virage Image Search Engine: An Open Framework for Image Management," in *Proceedings of the SPIE Storage and Retrieval for Image and Video Database*, February 1996.

[8] J. Dowe, "Content-based Retrieval in Multimedia Imaging," in *Proceedings of the SPIE Storage and Retrieval for Image and Video Database*, 1993.

[9] A. Pentland, R.W. Picard, S. Sclaroff, "Photobook: Content-based Manipulation of Image Databases," *International Journal of Computer VIsion*, vol. 18, no. 3, pp. 233–254, 1996.

[10] J.R. Smith, S-F. Chang, "Visualseek: A Fully Automated Content-based Image Query System," in *Proceedings of the ACM Multimedia*, 1996.

[11] J.R. Smith, S-F. Chang, "Visually Searching the Web for Content," *IEEE Multimedia Magazine*, vol. 4, no. 3, 1997.

[12] K. Hirata, T. Kato, "Query by Visual Example," in *Proceedings of the 3rd International Conference on Extending Database Technology*, March 1992.

[13] C. Carson, S. Belongie, H. Greenspan, J. Malik, "Region-based Image Querying," in *Proceedings of the IEEE Workshop on Content-based Access of Image and Video Libraries in conjunction with IEEE CVPR '97*, 1997.

[14] C. Carson, M. Thomas, S. Belongie, J. M. Hellerstein, J. Malik, "Blobworld: A System for Region-Based Image Indexing and Retrieval," in *Third International Conference on Visual Information Systems*, Amsterdam, June 1999.

[15] M. Barni, F. Bartolini, A. De Rosa, "HVS Modelling for Quality Evaluation of Art Images," in *14th International Conference on Digital Signal Processing*, Santorini, Greece, July 2002.

[16] N. Eastaugh, "Examination of Paintings by Infra-red and Other Techniques," in *IEE Colloquium on NDT in Archaelogy and Art*, 25 May 1995.

[17] A.M. Bonacchi, V. Cappellini, M. Corsini, A. De Rosa, "Artshop: A Tool for Art Image Processing," in *14th International Conference on Digital Signal Processing*, Santorini, Greece, July 2002.

[18] M. Müller, *Mise en Œuvre des Techniques de Traitement des Images pour I'Interprétation des Peintures*, Telecom Paris, de I'Ecole Nationale Supérieure des Télécommunications, 1994.

[19] B. Smolka, M. Szezapanski, "New Technique for the Restoration of Noisy Color Images," in *14th International Conference on Digital Signal Processing*, Santorini, Greece, July 2002.

[20] A. De Polo, F. Alinari, "Digital Picture Restoration and Enhancement for Quality Archiving," in *14th International Conference on Digital Signal Processing*, Santorini, Greece, July 2002.

[21] M. Barni, F. Bartolini, V. Cappellini, "Image Processing for Virtual Restoration of Artworks," *IEEE Multimedia*, vol. 7, no. 2, pp. 34–37, April-June 2000.

[22] I. Giakoumis, I. Pitas, "Digital Restoration of Painting Cracks," in *ISCAS '98, Proceedings of the IEEE International Symposium on Circuits and Signals*, 31 May-3 June 1998, pp. 269–272.

[23] M. Pappas, I. Pitas, "Digital Color Restoration of Old Paintings," *IEEE Transactions on Image Processing*, vol. 9, no. 2, pp. 291–294, February 2000.

[24] N. Nikolaidis, I. Pitas, "Digital Image Processing in Painting Restoration and Archiving," in *Proceedings of the IEEE International Conference on Image Processing (ICIP)*, 2001.

[25] F. Heitz, H. Maî, C. de Couessin, "Application of Autoregressive Models to Fine Arts Painting Analyis," *Signal Processing*, vol. 13, no. 1, pp. 1–14, July 1987.

[26] P. Allen, M. Boniface, P. Lewis, K. Martinez, "Interoperability Between Multimedia Collections for Content and Metadata-Based Searching," in *Proceedings of the World Wide Web Conference*, Honolulu, Hawaii, 2002.

[27] J.M. Corridoni, A. Del Bimbo, S. De Magistris, E. Vicario, "A Visual Language for Color-Based Painting Retrieval," in *IEEE Symposium on Visual Languages*, 3-6 September 1996, pp. 68–75.

[28] Y. Isomoto, K. Yoshine, H. Yamasaki, N. Ishii, "Color, Shape and Impression Keyword as Attributes of Paintings for Information Retrieval," in *IEEE International Conference on Systems, Man, and Cybernetics*, October 1999, pp. 12–15.

[29] M. Westmacott, P. Lewis, K. Martinez, "Using Colour Pair Patches for Image Retrieval," in *First European Conference on Colour in Graphics, Imaging and Vision*, 2002, pp. 245–247.

[30] S. Chan, K. Martinez, P. Lewis, C. Lahanier, J. Stevenson, "Handling of Sub-Image Queries in Content-Based Retrieval of High Resolution Art Images," in *International Cultural Heritage Informatics Meeting 2*, 2002, pp. 245–247.

[31] A. Kushki, P. Andaroutsos, K.N. Plataniotis, A.N. Venetsanopoulos, "Fuzzy Aggregation of Image Features in Content-Based Image Retrieval," in *Proceedings of the IEEE International Conference on Image Processing (ICIP) vol. 1*, 2002, pp. 115–118.

[32] P.H. Lewis, K. Martinez, F.S. Abas, M.F. Ahmad Fauzi, M. Addis, C. Lahanier, S.C.Y. Chan, J.B. Mike, G. Paul, "An Integrated Content and Metadata based

Retrieval System for Art," *IEEE Transactions on Image Processing*, vol. 13, no. 3, pp. 302–313, 2004.

[33] F.S. Abas, K. Martinez, "Craquelure Analysis for Content-based Retrieval," in *14th International Conference on Digital Signal Processing*, Santorini, Greece, July 2002.

[34] F.S. Abas, K. Martinez, "Classification of Painting Cracks for Content-Based Analysis," in *Proceedings of the IS&T/SPIE 15th Annual Symposium Electronic Imaging: Science and Technology, Machine Vision Applications in Industrial Inspection XI*, Santa Clara, California, January 2003, pp. 149–160.

[35] F.S. Abas, K. Martinez, "Grouping of Crack Patterns using Proximity and Characteristic Rules," in *Proceedings of the 3rd IASTED International Conference on Visualization, Imaging and Image Processing*, Benalmadena, Spain, September 2003.

[36] A.J. Varley, P.J.W. Rayner, "Bézier Modelling of Cracks," in *International Conference in Image Analysis and Processing*, 1997, pp. 551–558.

[37] A.J. Varley, P.J.W. Rayner, "Analysis of Crack Patterns Using MCMC Sampling," in *International Conference on Mathematics in Signal Processing*, June 1996.

[38] A.J. Varley, *Statistical Image Analysis Methods for Line Detection*, Ph.D. Thesis, Cambridge University, 1999.

[39] F. Heitz, H. Maître, C. de Couessin, "Event Detection in Multisource Imaging: Application to Fine Arts Painting Analysis," *IEEE Transactions on Acoustics, Speech and Signal Processing*, vol. 38, no. 4, pp. 695–704, April 1990.

[40] S.R. Sternberg, "Grayscale Morphology," *Computer Vision, Graphics and Image Processing*, vol. 35, 1985.

[41] R.M. Haralick, S.R. Sternberg, X. Zhuang, "Grayscale Morphology," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 1986, pp. 523–550.

[42] R.M. Haralick, S.R. Sternberg, X. Zhuang, "Image Analysis Using Mathematical Morphology," *IEEE Transactions on Pattern Analysis and Machine Vision*, 1987.

[43] F. Meyer, "Iterative Image Transformations for an Automatic Screening of Cervical Smears," *J.Histoch.Cytochem*, vol. 27, 1979.

[44] S. Bucklow, "The Description of Craquelure Patterns," *Studies in Conservation*, vol. 42, 1997.

[45] P. Magliano, "Xeroradiography for Paintings on Canvas and Wood," *Studies in Conservation*, vol. 33, 1988.

[46] S. Bucklow, "A Stylometric Analysis of Craquelure," *Computers and Humanities*, vol. 31, 1998.

[47] S. Marshall, "Review of Shape Coding Techniques," *Image and Vision Computation*, vol. 7, 1989.

[48] J. Wang, W. Chang, R. Acharya, "Efficient and Effective Similar Shape Retrieval," in *Proceedings of the IEEE International Conference on Multimedia Computing and Systems vol. 1*, Florence, Italy, June 1999.

[49] M. Sonka, V. Hlavac, R. Boyle, *Image Processing, Analysis and Machine Vision*, Chapman and Hall Computing, London,UK, 1993.

[50] H. Freeman, "Boundary Encoding and Processing," *Picture Processing and Psychopictories, Academic Press*, 1970.

[51] H. Freeman, "Computer Processing of Line Drawing Images," *Computer Surveys*, vol. 6, no. 1, pp. 57–98, 1974.

[52] F. Arebola, P. Camacho, A. Bandera, F. Sandoval, "Corner Detection and Curve Representation by Circular Histograms of Contour Chain Code," *IEEE Electronics Letter*, vol. 35, no. 13, pp. 1065–1067, June 1999.

[53] F. Arrebola, A. Bandera, P. Camacho, F. Sandoval, "Corner Detection by Local Histograms of Contour Chain Code," *IEEE Electronics Letter*, vol. 33, no. 21, pp. 1769–1771, October 1997.

[54] H. Freeman, J. Saghri, "Comparative Analysis of Line Drawing Modelling Schemes," *Computer Graphics and Image Processing*, vol. 12, 1980.

[55] H. Freeman, "Shape Description Via the Use of Critical Points," *Pattern Recognition*, vol. 10, 1978.

[56] L. O'Gorman, "Primitives Chain Code," in *ICASSP-88, International Conference on Acoustics, Speech, and Signal Processing*, 11-14 April 1988, pp. 792–795.

[57] M. Seul, L. O'Gorman, M.J. Sammon, *Practical Algorithms for Image Analysis Description, Examples, and Code*, Cambridge University Press, Cambridge, UK, 2000.

[58] E. Persoon, K. Fu, "Shape Discrimination Using Fourier Descriptors," *IEEE Transactions on Systems, Man and Cybernetics*, vol. 7, 1977.

[59] A.S. Aguado, M.S. Nixon, M.E. Montiel, "Parameterising Arbitrary Shapes Via Fourier Descriptors for Evidence-Gathering Extraction," *CVGIP: Image Understanding*, vol. 69, no. 2, pp. 202–221, 1998.

[60] T. Pavlidis, "A Review of Algorithms for Shape Analysis," in *Proceedings of Computer Graphics and Image Processing vol. 7*, 1978, pp. 243–258.

[61] A.S. Aguado, M.E. Montiel, M.S. Nixon, "Extracting Arbitrary Geometric Primitives Represented by Fourier Descriptors," in *Proceedings of the IEEE International Conference on Pattern Recognition*, Vienna, Austria, 1996, pp. 547–551.

[62] C. Zahn, R. Roskies, "Fourier Descriptors for Plane Closed Curves," *Computer Graphics and Image Processing*, vol. 21, 1972.

[63] S. Loncaric, "A Survey of Shape Analysis Techniques," *Pattern Recognition*, vol. 31, no. 8, pp. 983–1001, 1998.

[64] R.J. Prokop, A.P. Reeves, "A Survey of Moment-based Techniques for Unoccluded Object Representation and Recognition," *Computer Vision, Graphics and Image Processing*, vol. 54, no. 5, pp. 438–460, 1992.

[65] A. Khotanzad, Y.H. Hong, "Invariant Image Recognition by Zernike Moments," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 12, no. 5, pp. 489–497, May 1990.

[66] M.K. Hu, "Visual Pattern Recognition by Moment Invariants," *IRE Transactions on Information Theory*, vol. 8, no. 2, pp. 179–187, 1962.

[67] S. Staniforth, "Lending Paintings - The Conservator's View," in *Proceedings of the International Conference on the Packing and Transportation of Paintings*, London, September 1991, p. 339.

[68] N. Stolow, B. Wennberg, "A Report on a Preliminary Study of Damage to Paintings During Transport and Temporary Exhibition by Means of Photographic Enlargement," in *ICOM - Committee for Conservation*, Amsterdam, 1972.

[69] S. Michalski, "Paintings - Their Response to Temperature, Relative Humidity, Shock, and Vibration," in *Proceedings of the International Conference on Packing and Transportation of Paintings*, London, September 1991, p. 241.

[70] A. Murray, R.E. Green, M.F. Mecklenburg, C.M. Fortunko, "Nondestructive Evaluation of Works of Art," in *Proceedings of the International Conference on the Packing and Transportation of Paintings*, London, September 1991, p. 249.

[71] K. Martinez, "High Resolution Digital Imaging of Paintings: The VASARI Project," in *NIT '91: 4th International Conference on New Information Technology*, Budapest, December 1991, p. 131.

[72] http://www.artisteweb.org/.

[73] "Resource Description Framework (RDF) Model and Syntax Specification W3C Recommendation", http://www.w3.org/TR/REC-rdf-syntax/, 22 February 1999.

[74] L. Vuurpijl, L. Schomaker, "Two-stage Character Classification: A Combined Approach of Clustering and Support Vector Classifiers," in *7th International Workshop on Frontiers in Handwriting Recognition*, Amsterdam, Netherlands, September 2000.

[75] X. Li, D.Y. Yeung, "On-Line Handwritten Alphanumeric Character Recognition Using Feature Sequences," *Pattern Recogntion*, vol. 30, no. 1, pp. 31–44, 1995.

[76] Ø.D. Trier, A.K. Jain, T. Taxt, "Feature Extraction Methods for Character Recognition - A Survey," *Pattern Recognition*, vol. 29, no. 4, pp. 641–662, 1996.

[77] V. Wu, R. Manmatha, E.M. Riseman, "Finding Text in Images," in *2nd International Conference on Digital Libraries*, July 1997, pp. 3–12.

[78] A.M. López, F. Lumbreras, "Evaluation of Methods for Ridge and Valley Detection," *IEEE Transactions on Pattern and Machine Intelligence*, vol. 21, no. 4, pp. 327–335, April 1999.

[79] D. Elberly, R. Gardner, B. Morse, S. Pizer, C. Scharlach, "Ridges for Image Analysis," *Journal of Mathematical Imaging and Vision*, vol. 4, no. 4, pp. 353–373, December 1994.

[80] J. Gaugh, S. Pizer, "Multiresolution Analysis of Ridges and Valleys in Grey-Scale Images," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 15, no. 6, pp. 635–646, June 1993.

[81] R. Haralick, "Ridges and Valleys on Digital Images," *Computer Vision, Graphics and Image Processing*, vol. 22, no. 10, pp. 28–38, April 1983.

[82] F. Zana, J.C. Klein, "Segmentation of Vessel-Like Patterns Using Mathematical Morphology and Curvature Evaluation," *IEEE Transactions on Image Processing*, vol. 10, no. 7, pp. 1010–1019, July 2001.

[83] F. Zana, J.C Klein, "Robust Segmentation of Vessels From Retinal Angiography," in *International Conference on Digital Signal Processing*, Santorini, Greece, July 1997.

[84] P. van del Elsen, J. Maintz, E-J. Pol, M. Viergever, "Automatic Registration of CT and MR Brain Images Using Correlation of Geometrical Features," *IEEE Transactions on Medical Imaging*, vol. 14, no. 2, pp. 384–396, June 1995.

[85] A.K. Jain, S. Prabhakar, L. Hong, "A Multichannel Approach to Fingerprint Classification," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 21, no. 4, pp. 348–359, April 1999.

[86] N. Merlet, J. Zerubia, "A Curvature-Dependent Energy Function for Detecting Lines in Satellite Images," in *SCIA*, Tromso, Norway, 1993.

[87] F. Tupin, H. Maitre, J. Mangin, J. Nicholas, E. Pechersky, "Detection of Linear Features in SAR Images: Application to Road Network Extraction," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 36, no. 2, pp. 434–453, 1998.

[88] G.J. Vanderbrug, "Semilinear Line Detectors," *Computer Graphics and Image Processing*, vol. 4, 1975.

[89] G.J Vanderbrug, "Line Detection in Satellite Imagery," *IEEE Transactions on Geoscience Electronics*, 1976.

[90] C. Kirbas, F. Quek, "A Review of Vessel Extraction Techniques and Algorithms," in *ACM Computing Surveys*, 2004.

[91] G. Matheron, "Eléments pour une Théorie des Milieux Poreux," in *Masson*, Paris, 1967.

[92] J. Serra, "Image Analysis and Mathematical Morphology," in *Academic Press*, London, 1982.

[93] R.M. Haralick, L.G. Shapiro, *Computer and Robot Vision*, Addison Wesley, Washington, 1992.

[94] G.H. Ritter, J.N. Wilson, *Handbook of Computer Vision Algorithms in Image Algebra*, CRC Press, Florida, 1996.

[95] N. Otsu, "A Threshold Selection Method From Gray-level Histogram," *IEEE Transactions on Systems, Man and Cybernetics*, vol. 8, 1978.

[96] S.Y. Chen, W.C Lin, C.T. Chen, "Split-and-Merge Image Segmentation Based on Localized Feature Analysis and Statistical Tests," *Graphical Models and Image Processing*, vol. 53, no. 5, pp. 457–475, September 1991.

[97] J. Kittler, J. Illingworth, J. Foglein, "Threshold Selection Based on a Simple Image Statistics," *Computer Vision, Graphics and Image Processing*, vol. 30, 1985.

[98] F. Martín, M. García, J.L. Alba, "New Methods for Automatic Reading of VLP's (Vehicle License Plate)," in *Proceedings of the SPPRA*, Crete, Greece, June 2002.

[99] A. Rosenfeld, A. Kak, *Digital Picture Processing*, Academic Press, New York, 1982.

[100] P.J. Green, "Reversible Jump Markov Chain Monte Carlo Computation and Bayesian Model Determination," *Biometrika*, vol. 82, no. 4, pp. 711–732, 1996.

[101] W.K. Pratt, *Digital Image Processing*, Wiley Interscience, 2nd edition, 1991.

[102] P.L. Rosin, "Measuring Rectangularity," *Machine Vision and Applications*, vol. 11, 1999.

[103] T. Brinkhoff, H-P. Kriegel, "Approximations for a Multi-step Processing of Spatial Joins," in *Geographic Information Systems, International Workshop on Advanced Information Systems*, Ascona, Switzerland, 1994, pp. 25–34.

[104] T. Brinkhoff, H-P. Kriegel, R. Schneider, "Comparison of Approximations of Complex Objects Used for Approximation-Based Query Processing in Spatial Database Systems," in *Proceedings of the 9th International Conference on Data Engineering*, Vienna, Australia, 1993, pp. 40–49.

[105] J.D. Foley, A. van Dam, S.K. Feiner, J.F. Hughes, *Computer Graphics: Principles and Practice (2nd Edition in C)*, Addison Wesley, 1996.

[106] L. McMillan, *An Image-Based Approach to Three-Dimensional Computer Graphics*, Ph.D. Thesis, University of North Carolina at Chapel Hill, 1997.

[107] C. Hoffman, *Geometric and Solid Modelling*, Morgan Kauffman, 1989.

[108] A. Garcia-Alonso, N. Serrano, J. Flaquer, "Solving the Collision Detection Problem," *IEEE Computer Graphics and Applications*, vol. 14, 1995.

[109] J.-C. Latombe, *Robot Motion Planning*, Kluwer Academic Publishers, 1991.

[110] J.-D. Liu, M.-T. Ko, R.-C. Chang, "Simple Self-Collision Avoidance for Cloth Animation," *Computers and Graphics*, vol. 22, 1998.

[111] J.D. Cohen, M.C. Lin, D. Manocha, M.K. Ponamgi, "I-COLLIDE: An Interactive and Exact Collision Detection System for Large-Scale Environments," in *Proceedings of the ACM Interactive 3D Graphics Conference*, 1995, pp. 189–196.

[112] M.P. Moore, J. Wilhelms, "Collision Detection and Response for Computer Animation," in *Proceedings of SIGGRAPH*, 1988.

[113] M. Wertheimer, "Laws of Organization in Perceptual Forms," *W.D Ellis, editor, A Source Book of Gestalt Psychology*, 1950.

[114] F. Arrebola, C. Urdiales, P. Camacho, F. Sandoval, "Vision System Based on Shifted Fovea Multiresolution Retinotopologies," in *Proceedings of the IEEE IECON vol. 3*, 1998, pp. 1357–1361.

[115] E.B. Meier, F. Ade, "Tracking Cars in Range Image Sequences," in *Proceedings of the IEEE ITSC*, 1997, pp. 105–110.

[116] F.R. Chen, D.S. Bloomberg, "Summarization of Imaged Documents Without OCR," *Computer Vision and Image Understanding*, vol. 7, no. 3, pp. 307–320, 1998.

[117] C.-W. Chang, S.-Y. Lee, "Video Content Representation, Indexing and Matching in Video Information Systems," *Visual Communications and Image Representation*, vol. 8, no. 2, pp. 107–120, January 1997.

[118] W.T. Fung, S.Y. Yuen, C.H. Tse, "Hierarchical Bounding Box Method for Searching the Speech Database for Speech Recognition," in *Proceedings of the IEEE ISCE*, 1997, pp. 43–46.

[119] E. Paquet, M. Rioux, A. Murching, T. Naveen, A. Tabatabai, "Description of Shape Information for 2-D and 3-D Information," *Signal Processing: Image Communication*, vol. 16, no. 1, 2000.

[120] M.I. Sezan, R.J. Qian, "MPEG-7 Standardization Activities," in *Proceedings of the IEEE International Conference on Image Processing vol. 3*, Chicago, Illinois, October 1998, pp. 517–520.

[121] H. Freeman, R. Shapira, "Determining The Minimum-area Encasing Rectangle for an Arbitrary Closed-curve," *Communications of the ACM*, vol. 18, no. 7, pp. 409–413, July 1975.

[122] A.K. Jain, *Fundamentals of Digital Image Processing*, Prentice-Hall International Inc., London, 1989.

[123] A. Robbles Kelly, E.R. Hancock, "Grouping Line-Segments Using Eigenclustering," in *Proceedings of the 11th British Machine Vision Conference*, Bristol, United Kingdom, 2000, pp. 586–595.

[124] G. Guy, G. Mendioni, "Inferring Global Perceptual Contours From Local Features," *International Journal of Computer Vision*, vol. 20, no. 1, 1996.

[125] P.V.C. Hough, *A Method and Means for Recognizing Complex Patterns*, U.S Patent No. 3,069,654, 1962.

[126] D.C.W. Pao, H.F. Li, R. Jayakumar, "Shapes Recognition Using the Straight Line Hough Transform: Theory and Generalization," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 14, no. 11, pp. 1076–1089, November 1992.

[127] D.G. Lowe, "Three-dimensional Object Recognition From Single Two-dimensional images," *Artificial Intelligence*, vol. 31, 1987.

[128] N. Ahuja, M. Tuceryan, "Extraction of Early Perceptual Structure in Dot Patterns: Integrating Region, Boundary and Component Gestalt," *CVGIP*, vol. 48, 1989.

[129] J. Dolan, R. Weiss, "Perceptual Grouping of Curved Lines," in *Proceedings of the IUW*, Palo Alto, CA, 1989, pp. 1135–1145.

[130] R. Mohan, R. Nevatia, "Using Perceptual Organization to Extract 3-D Structures," *IEEE Transaction on Pattern Analysis and Machine Intelligence*, vol. 11, no. 11, pp. 1121–1139, November 1989.

[131] A. Sha'ashua, S. Ullman, "Structure Saliency: The Detection of Globally Salient Structures Using a Locally Connected Network," in *Proceedings of the ICCV*, Tampa, Florida, December 1988, pp. 321–327.

[132] P. Parent, S.W. Zucker, "Trace Inference, Curvature Consistency, and Curve Detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 11, no. 8, pp. 823–839, August 1989.

[133] F. Heitger, R. von der Heydt, "A Computational Model of Neural Contour Processing: Figure-Ground Segregation and Illusory Contours," in *Proceedings of the ICCV*, 1993, pp. 32–40.

[134] A. Amir, *A Quantitative Approach to Perceptual Grouping in Computer Vision*, Ph.D. Thesis, Technion - Israel Institute of Technology, Haifa, Israel, 1997.

[135] D.G. Lowe, *Perceptual Organization and Visual Recognition*, Kluwer Academic Press Publication, 1985.

[136] S. Sarkar, K.L. Boyer, "Perceptual Organization in Computer Vision: A Review and Proposal for a Classifactory Structure," *IEEE Transactions on System, Man and Cybernetics*, vol. 23, no. 2, pp. 382–399, March/April 1993.

[137] L.E. Gordon, *Theories of Visual Perception*, John Wiley and Sons, First Edition, 1989.

[138] A.K. Jain, M.N. Murty, P.J. Flynn, "Data Clustering: A Review," *ACM Computing Surveys*, vol. 31, 1999.

[139] R.J. Schalkoff, *Pattern Recognition: Statistical, Structural and Neural Network*, John Wiley & Sons. Inc., New York, 1991.

[140] A.K. Jain, R.C. Dubes, *Algorithms for Clustering Data*, Prentice Hall, Englewood Cliffs, N.J, 1988.

[141] M.R. Anderberg, *Cluster Analysis for Applications*, Academic Press, Inc., New York, 1973.

[142] J.C. Bezdek, *Pattern Recognition With Fuzzy Objective Function Algorithms*, Plenum Press, New York, 1981.

[143] L. Kaufman, P. Rousseeuw, *Finding Groups in Data: An Introduction to Cluster Analysis*, John Wiley and Sons, New York, 1990.

[144] P. Berkhin, *Survey Of Clustering Data Mining Techniques*, Technical Report, Accrue Software, San Jose, CA, 2002.

[145] A.D. Gordon, "A Review of Hierarchical Classification," *Journal of the Royal Statistics Society*, vol. 150, 1987.

[146] C. Olson, "Parallel Algorithms for Hierarchical Clustering," *Parallel Computing*, vol. 21, 1995.

[147] R.A. Baeza-Yates, "Introduction to Data Structures and Algorithms Related to Information Retrieval," *Information Retrieval: Data Structures and Algorithms, W.B Frakes and R. Baeza-Yates, Eds. Prentice-Hall, Inc., Upper Sadle River, NJ*, 1992.

[148] G. Nagy, "State of the Art in Pattern Recognition," in *Proc. IEEE 56*, 1968, pp. 836–862.

[149] S. Bucklow, "Consensus in the Classification of Craquelure," *Hamilton Kerr Institute Bulletin, Fitzwilliam Museum, Cambridge*, vol. 3, 2001.

[150] Y. Xu, E. Saber, A.M. Tekalp, "Object Segmentation and Labeling by Learning From Examples," *IEEE Transactions on Image Processing*, vol. 12, no. 6, pp. 627–638, June 2003.

[151] A.M. Vossepoel, A.W.M. Smeulders, "Vector Code Probability and Metrication Error in the Representation of Straight Lines of Finite Length," *Computer Graphics and Image Processing*, vol. 20, 1982.

[152] J. Iivarinen, A. Visa, "Shape Recognition of Irregular Objects," in *Proceedings of the SPIE Intelligent Robots and Computer Vision: Algorithms, Techniques, Active Vision, and Materials Handling*, 1996, pp. 25–32.

[153] R. Brunelli, O. Mich, "On The Use of Histograms for Image Retrieval," in *Proceedings of the IEEE International Conference on Multimedia Computing and Systems*, Volume 2, 1999, pp. 143–147.

[154] S. Bucklow, *Formal Connoisseurship and the Characterisation of Craquelure*, Ph.D. Thesis, 1996.

[155] S. Krishnan, K. Samudravijaya, P.V.S. Rao, "Feature-selection for Pattern-classification with Gaussian Mixture-model - A New Objective," *Pattern Recognition Letters*, vol. 17, no. 8, pp. 803–809, 1996.

[156] A.P. Dempster, N.M. Laird, D.B. Rubin, "Maximum Likelihood From Incomplete Data via the EM Algorithm," *Journal of the Royal Statistical Society - B*, vol. 39, 1977.

[157] A.K. Jain, R.P.W. Duin, J. Mao, "Statistical Pattern Recognition: A Review," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, no. 1, pp. 4–37, January 2000.

[158] D. Comaniciu, P. Meer, "Mean Shift Analysis and Applications," in *Proceedings of the IEEE International Conference on Computer Vision*, 1999.

[159] Y. Cheng, "Mean Shift, Mode Seeking, and Clustering," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 17, no. 8, pp. 790–799, August 1995.

[160] H. Frigui, R. Krishnapuram, "A Robust Competitive Clustering Algorithm With Applications in Computer Vision," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 21, no. 5, pp. 1999, May 1999.

[161] D. Michie, D.J. Spiegelhalter, C.C. Taylor, *Machine Learning, Neural and Statistical Classification*, Ellis Horwood, New York, 1994.

[162] T.M. Cover, P.E. Hart, "Nearest Neighbour Pattern Classification," *IEEE Transactions on Information Theory*, vol. 13, no. 1, pp. 21–27, January 1967.

[163] R.O. Duda, P.E. Hart, *Pattern Classification and Scene Analysis*, Wiley-Interscience, New York, 1973.

[164] S. Dudani, "The Distance-weighted K-NN Rule," *IEEE Transactions on Systems, Man and Cybernetics*, vol. 6, no. 4, pp. 325–327, 1976.

[165] L. Zadeh, "Fuzzy Sets," *Information Control*, vol. 8, 1965.

[166] J.M. Keller, M.R. Gray, J.A. Givens Jr., "A Fuzzy K-Nearest Neighbor Algorithms," *IEEE Transactions on Systems, Man and Cybernetics*, vol. 15, no. 4, pp. 580–585, 1985.

[167] S. Bucklow, "The Description and Classification of Craquelure," *Studies in Conservation*, vol. 44, 1999.

[168] http://www.itl.nist.gov/iad/894.03/databases/defs/dbases.html.