TEXTURE DESCRIPTORS FOR CONTENT-BASED IMAGE RETRIEVAL

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
THE MIDDLE EAST TECHNICAL UNIVERSITY

BY

ABDURRAHMAN ÇARKACIOĞLU

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF

DOCTOR OF PHILOSOPHY

IN

THE DEPARTMENT OF COMPUTER ENGINEERING

AUGUST 2003

Approval of the Graduate School of Natural and Applied Sciences.

Prof. Dr. Canan Özgen
Director

I certify that this thesis satisfies all the requirements as a thesis for the degree of Doctor of Philosophy.

Prof. Dr. Ayşe Kiper
Head of Department

This is to certify that we have read this thesis and that in our opinion it is fully adequate, in scope and quality, as a thesis for the degree of Doctor of Philosophy.

Prof. Dr. Fatoş Yarman-Vural
Supervisor

Examining Committee Members

Prof. Dr. Fatoş Yarman-Vural _____

Prof. Dr. Uğur Halıcı _____

Prof. Dr. Enis Çetin _____

Assoc. Prof. Dr. Volkan Atalay _____

Assoc. Prof. Dr. Gözde Bozdağı Akar _____

ABSTRACT

TEXTURE DESCRIPTORS FOR CONTENT-BASED IMAGE RETRIEVAL

Çarkacıoğlu, Abdurrahman

Ph.D., Department of Computer Engineering

Supervisor: Prof. Dr. Fatoş Yarman-Vural

August 2003, 153 pages

Content Based Image Retrieval (CBIR) systems represent images in the database by color, texture, and shape information. In this thesis, we concentrate on texture features and introduce a new generic texture descriptor, namely, Statistical Analysis of Structural Information (SASI). Moreover, in order to increase the retrieval rates of a CBIR system, we propose a new method that can also adapt an image retrieval system into a configurable one without changing the underlying feature extraction mechanism and the similarity function.

SASI is based on statistics of clique autocorrelation coefficients, calculated over structuring windows. SASI defines a set of clique windows to extract and measure various structural properties of texture by using a spatial multiresolution method. Experimental results, performed on various image databases, indicate that SASI is more successful then the Gabor Filter descriptors in capturing small granularities and discontinuities such as sharp corners and abrupt changes. Due to the flexibility in designing the clique windows, SASI reaches higher average retrieval rates compared to Gabor Filter descriptors. However, the price of this performance is increased computational complexity.

Since, retrieving of similar images of a given query image is a subjective task, it is desirable that retrieval mechanism should be configurable by the user. In the proposed method, basically, original feature space of a content-based retrieval system is nonlinearly transformed into a new space, where the distance between the feature vectors is adjusted by learning. The transformation is realized by Artificial Neural Network architecture. A cost function is defined for learning and optimized by simulated annealing method. Experiments are done on the texture image retrieval system, which use SASI and Gabor Filter features. The results indicate that configured image retrieval system is significantly better than the original system.

Keywords: Texture, Similarity, Feature, Descriptor, Clique, Autocorrelation, Content-based retrieval.

ÖZ

İÇERİK TABANLI GÖRÜNTÜ ERİŞİMİ İÇİN DOKU TANIMLAYICILARI

Çarkacıoğlu, Abdurrahman
Doktora, Bilgisayar Mühendisliği Bölümü
Tez Yöneticisi: Prof. Dr. Fatoş Yarman-Vural

Ağustos 2003, 153 sayfa

İçerik-Tabanlı Görüntü Erişim (İTGE) sistemleri, veri tabanındaki görüntüleri renk, doku ve şekil bilgileri ile temsil ederler. Bu tezde, biz doku öznitelikleri üzerinde yoğunlaştık ve ismi Yapısal Bilginin İstatistiksel Analizi (YBİA) olan yeni bir genel doku tanımlayıcısı duyurduk. Ayrıca, bir İTGE sisteminin erişim oranlarının artırılması amacıyla, sistemim mevcut öznitelik çıkartma mekanizmasını ve benzerlik fonsiyonunu değiştirmeksizin, görüntü erişim sistemini ayarlanabilir duruma uyarlayan yeni bir yöntem önerdik.

YBİA, yapısal pencereler üzerinde hesaplanan otokorelasyon katsayılarının istatistiklerine dayalıdır. YBİA dokudaki çeşitli yapısal bilgilerin, çoklu çözünürlük yöntemiyle çıkartılması ve ölçülmesi amacıyla bir küme çubuk penceresi tanımlar. Farklı görüntü veri tabanları üzerinde yapılan deneysel sonuçlar, sert köşeler ve ani değişiklikler gibi küçük toplulaşmaları ve kesiklilikleri tespit etmekte YBİA'nın Gabor filtrelerinden daha başarılı olduğunu göstermiştir. Çubuk pencerelerinin tasarımındaki esneklik, YBİA'nın Gabor filtrelerine göre daha yüksek ortalama erişim oranlarına ulaşmasını sağlamaktadır. Fakat, böyle bir performans artışının bedeli artan hesap karmaşıklığıdır.

Verilen sorgu resmine benzer resimlere erişilmesi, subjektif bir iş olması sebebiyle, erişim mekanizmasının kullanıcı tarafından ayarlanılabilmesi istenir. Önerdiğimiz yeni yöntemde, basit olarak, bir içerik-tabanlı görüntü sisteminin öznitelik uzayı, öznitelik vektörlerinin birbirlerine uzaklıkları öğrenilerek ayarlanılabildiği, yeni bir uzaya doğrusal olmayan bir biçimde dönüştürülmektedir. Bu dönüşüm Yapay Sinir Ağları yapısı ile gerçekleştirilmektedir. Öğrenme için bir paha fonksiyonu tanımlanmakta ve simule edilmiş tavlama yöntemiyle eniyilenmektedir. Deneyler, YBİA ve Gabor filtreleri kullanan doku görüntüsü erişim sistemi üzerinde yapılmıştır. Sonuçlar ayarlanmış görüntü sisteminin orijinal sistemden oldukça iyi olduğunu göstermiştir.

Anahtar Kelimeler: Doku, Benzerlik, Öznitelik, Tanımlayıcı, Çubuk, Otokorelasyon, İçerik-tabanlı erişim.

To my family

ACKNOWLEDGMENTS

I would like to express my deepest appreciation and sincere gratitude to my supervisor, Professor Fatoş Yarman-Vural for her continuous feedback and support throughout my research. Without the help, supervision, and opportunity that she provided, this thesis would not be complete.

I also want to thank Prof. Uğur Halıcı and Prof. Volkan Atalay for their useful advices and suggestions on my research.

I wish to express sincere gratefulness and indebtedness to my parents and my brother who provide me with endless love and support. I would value them forever.

Finally, special thanks go to my wife, who always supports me and encourages me with her great patience whenever and wherever.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

This chapter gives a brief description on the background of this dissertation. Also, organization of the thesis is given shortly.

1.1 Motivation

Although, it was possible to read every book written in the 18th century, nowadays such a trial may take 15,000 years [1]. As the human knowledge gets expanding, no one individual can capture more than a small piece of the knowledge. Due to the exponential rate of growth of knowledge, people have to specialize in narrower fields. According to the Hawking [1], this is likely to be a major limitation in the future. Moreover, he said that we have to become more intelligent and better natured via changing and improving our DNA, which is out of the scope of this dissertation.

In order to overcome the bottleneck of the rate of the growth of the information *vs.* human capacity, computer science and technology offers a wide range of opportunities. The advances in computation power, storage, scanning and networking devices enable us to make available and to reach a great variety of information sources.

Digitized information includes text, image, video, and/or audio datum. Now-

adays, Internet is the most common way to share them. Tera bytes of information within over 800 million web pages available to the Internet users [2, 3]. Information search and retrieval are not only essential parts of public information sharing, but also fundamental transaction for professional information need such as military, government, architecture, advertising, fashion, publishing, and medical applications.

During the last two decades, search by using textual information has been successfully analyzed by the researchers. Currently, over 1200 text search engines are commercially available. On the other hand, image, video and audio information retrieval are still being studied.

When searching visual or aural contents, the attachment of text labels to the sources is inadequate, since not only because it is impractical, but also subjective to human [4]. As a result there is a great interest in content-based retrieval systems.

In recent years, very large image and video databases have grown rapidly. For example, satellites send 1.5 Tera bytes of images to NASA every day [5]. Thousands of medical images, fingerprint images, military related images, and even the personal images are stored in digital format throughout the world every day. Although a few number of content-based image retrieval (CBIR) systems exist, some of them are domain specific, while the others do not fully meet the expectations of the users.

We mainly concentrate on CBIR systems, which query the image database by example. In CBIR systems images are indexed and searched by their visual contents such as color, texture, or shape. This dissertation is primarily focused on texture descriptors for CBIR systems.

Most of the real life images contain various types of textures, which are very complex due to the changes in scale, orientation, shape, contrast, *etc.* Due to such complexities, a clear-cut, widely accepted definition of texture does not exist. However, texture is one of the most important visual properties of an image.

In recent years, textural information has been widely used as a visual primitive in many CBIR systems [6, 7, 8, 9]. The potential areas include industrial and biomedical surface inspection, ground classification and segmentation of satellite or aerial imagery, document analysis, scene analysis, texture synthesis for computer graphics and animation, biometric person authentication, content-based image retrieval and image coding [10, 11, 12].

Current CBIR systems represent textures by low-level image features. Well-known and widely used texture descriptors are based on Gabor filters. However, due to digitalization of analog Gabor filters, it has some problems with the textures that consist of small texels or sharp corners. Thus, texture feature extraction is still an active research area.

1.2 Major Contributions of the Thesis

This thesis is devoted to texture. The primary focus is on content-based image retrieval systems by texture. Our aim is to improve the retrieval rates of a CBIR system using texture. There are two main contributions of the thesis listed below:

- We introduce "Statistical Analysis of Structural Information" (SASI) as a new texture descriptor. SASI is more generic then the available texture descriptors in the literature, covering both statistical and structural textures. It has a great power in the design of the descriptor, which allows the user to capture small variations in the texture. SASI is based on the second-order statistics of autocorrelation coefficients calculated over moving clique windows. In order to represent textures, varying size and orientation clique windows are employed. Clique windows are the structuring elements, which can be defined by a neighborhood system. The order of the neighborhood system determines the structure of the clique windows.

  Due to flexibility in the definition of clique windows, SASI successfully represents wide variety of textures. We experiment the retrieval rate of

SASI on various texture database in the literature. We compare it to Gabor descriptor and find that SASI always gives better retrieval rates.

- Retrieving the similar images is a subjective task. As a result, it is not possible to create a CBIR system whose retrieval results are found to be satisfiable by different person. Thus, we suggest a method to adapt an image retrieval system into a configurable one. Basically, using Artificial Neural Networks, original feature space of a content-based retrieval system is nonlinearly transformed into a new space, where the distance between the feature vectors is adjusted by learning. After defining a proper cost function, learning is accomplished by the use of simulated annealing (SA). Broadly speaking, SA searches the weights and the bias terms of the ANN that maximize/minimize the cost function.

  We test the method on two texture image retrieval systems, which use SASI and Gabor Filter features, respectively. The results indicate that configured image retrieval systems are significantly better than their original counterparts.

1.3 Organization of the Thesis

The main contents of this thesis are listed as follows.

- A brief overview on content-based image retrieval is given in Chapter 2. Also, the concept of similarity and similarity functions are presented.

- In Chapter 3, the definition of texture, existing texture analysis methods and their brief comparisons are given.

- A new generic texture descriptor for image retrieval, namely SASI is introduced along with the background definitions in Chapter 4.

- In Chapter 5, first, SASI is analyzed in detail and compared to Gabor filter descriptors. Then, using various image databases, SASI and Gabor

filter descriptor are tested on image retrieval problem. SASI is compared to Gabor filter and it is shown that SASI gives the best retrieval rates for each image database.

- In Chapter 6, we suggest a method to adapt an image retrieval system into a configurable one.
- Chapter 7 concludes this thesis and presents suggestions for future work.

CHAPTER 2

BRIEF OVERVIEW ON IMAGE RETRIEVAL

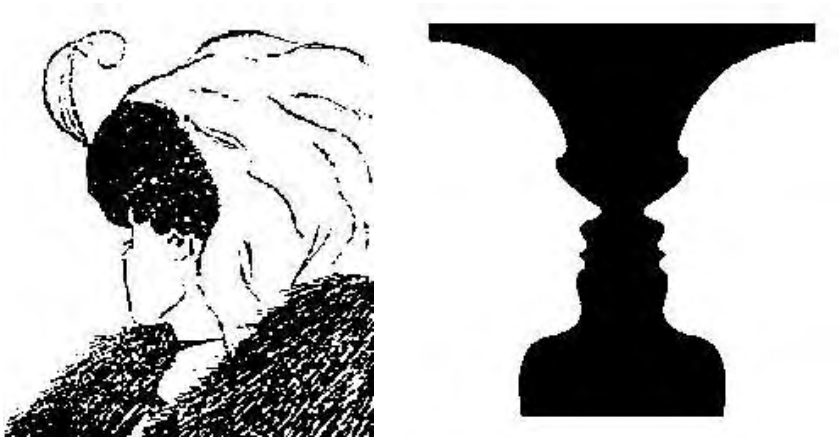
Since 1970, both Database Management and Computer Vision community has been working on image retrieval from different perspectives [9]. While the former is interested in text-based image retrieval, the latter explores the content-based image retrieval in which the image is represented by its salient features. In this chapter, we summarize these two approaches by emphasizing the second one.

In text-based image retrieval (TBIR) systems, an operator, more specifically a human indexer, manually annotates each image in the collection by text. In other words, images are described as a set of keywords or free text [13]. Then, such annotations are stored in a traditional Database Management System (DBMS) [14, 15].

Generally, images are retrieved from DBMS by using conventional queries that are executed on exact or probabilistic match of the query text. Query text can be either a single keyword or a description of an object depicted in the image. The DBMS cannot retrieve the desired images, unless the images in the database are correctly and sufficiently described. Retrieval performance is directly related to congruence between the vocabularies of the operator(s) and the user(s) of a TBIR system.

TBIR has two main problems. The first problem emerges due to the human subjectivity on complex images. Since "a picture is worth a thousand words", it

is not always possible to define or describe wide variety of images just by using some textual information. Also, different people or the same person in different situations describe or judge the same image differently, due to human perception subjectivity. Figure 2.1 shows such subjectivity by example. Some people see a young lady in Figure 2.1.a while the others see a hag. Again, some people see a vase in Figure 2.1.b, while the others see two people looking at each other.



(a) (b)

Figure 2.1: Human perception subjectivity. (a) Can you see a hag or a young lady? (b) Can you see a vase or two people looking at each other?

The second problem lies in the difficulty of manual indexing. As the number of images gets larger, the total amount of time spent in manual image annotation is also increased.

Although a number of surveys about TBIR exist in the literature [14, 15, 16, 17, 18], the most interesting remark was done by Berrut et al [16]. Their survey found that despite the above difficulties and disadvantages, users of TBIR systems seemed generally satisfied with their system due to high expressive power of the keyword indexing [19].

2.1 Content-Based Image Retrieval (CBIR)

The earliest use of the term content-based image retrieval in the literature seems to have been by Kato [20]. In CBIR, images are indexed by their own visual

contents such as color, texture, and shape. Visual contents are extracted from the images as automatically as possible [18]. Thus, CBIR systems have two main advantages over TBIR systems. First, they minimize the human effort. Second, due to reduced people intervention, subjectivity is also reduced. This feature makes CBIR systems more useful in many areas, such as search and browse large image collections.

2.1.1 Applications of CBIR

Detailed applications for CBIR technology can be found in [21]. Some of them are listed below:

- **Web searching**: A large number of digital images are accessed by the Internet users. CBIR systems can help the users to effectively find what they are looking for.

- **Medical diagnosis**: A large number of medical images have been stored by hospitals. Thus, CBIR systems can be used to aid diagnosis by identifying similar past cases.

- **Journalism and advertising**: Articles, photographs, videos of the newspapers, journals or televisions are queried by using CBIR systems.

- **Military**: Databases of all images in military applications; such as remotely sensed data, weapons, aircrafts, automatic target recognition, *etc.*

- **Intellectual property**: Most of the companies have their own trademark image. Whenever a new trademark image is to be registered, it must be compared with existing marks to eliminate duplications.

- **Crime prevention**: After a serious crime, law enforcement agencies search their archives for visual evidence. Such archives include photographs, fingerprints, tyre treads, shoeprints, and *etc.* of the past occasions. Thus, a CBIR system may help those agencies in finding related evidence.

2.1.2 Characteristics of Image Queries

CBIR systems can be evaluated according to the queries they handle. The queries are classified into three levels [22, 19]. Queries of the level 1 consist of primitive features such as color, texture, shape, or location of certain image elements. Queries of the level 2 and level 3 are composed of logical and abstract attributes, respectively. Logical features require some degree of logical inference about the identity of the objects depicted in the image, whereas abstract attributes involve a significant amount of high-level reasoning about the meaning and purpose of the objects depicted. Example queries for each level are listed below.

- Level 1
  - "Retrieve images that look like (or similar) to 'this' image" (This type of queries are also called query by example).
  - "Retrieve images with blue rectangle at the top of the image"
  - "Retrieve images that contain yellow squares"
- Level 2
  - "Retrieve images of a woman"
  - "Retrieve images of the Eiffel tower"
- Level 3
  - "Retrieve images depicting suffering"
  - "Retrieve images of Turkish folk dancing"

When interpreting and executing the queries of Level 1, CBIR systems uses features, which are both objective and directly derivable from the images themselves. Unlike Level 2 and 3, there is no need to refer any external knowledge base. Some researchers prefer to use the terms *lower-level approaches* for Level 1

and *higher-level approaches* for Level 2 and 3 [23], while the others call Level 2 and 3 together as *semantic image retrieval* [21].

Most of the higher level queries require automatic object recognition and classification, which are still among the unsolved problems in computer vision and image understanding literature [24]. Moreover, the queries of Level 2 and 3 cannot be interpreted and executed, unless underlying primitive (low-level) features are sufficient, effective, and accurate.

The major problem in CBIR systems is that the lack of a direct link between the high-level human concepts of images and the low-level features used by the CBIR systems. This fact is called the *semantic gap* problem.

The available CBIR systems, whether commercial or experimental, operate at Level 1 [19]. More specifically, most of the CBIR researches, including this dissertation, have been focused on "query by example". In query by example, the user does not have any particular target in mind, but selects an image or draws a sketch and asks to retrieve similar images. Thus, the basic operation is ordering a portion of image database with respect to a similarity metric [25].

The performance of a CBIR system is measured by *precision*, which is the number of relevant images retrieved relative to the total number of retrieved images and *recall*, which is the number of relevant images retrieved, relative to the total number of relevant images in the database.

2.1.3 Image Segmentation

Image segmentation is one of the most important preliminary steps in CBIR systems [26, 27, 28]. Performance of both the shape and layout features depends on "good" segmentation [9].

Segmentation can be defined as the partitioning of an image into a number of regions, which are homogeneous according to some criterion. The ultimate goal of the segmentation is to identify the semantically meaningful components in the image. Such components can be used for higher-order representation of the image.

Widely used image segmentation methods include *thresholded methods* [29], *feature space clustering* [30], *fuzzy clustering* [31], *edge-based methods* [32], *region growing methods* [33], *morphological methods* [34], *graph theoretic methods* [35], and *MRF based methods* [36]. Unfortunately, there is not a generic method that works well for all type of images [26].

2.1.4 A Typical CBIR System Operates on Level 1 (low-level)

A typical Level 1 CBIR system has two main tasks [37]. The first task is visual content extraction or feature extraction. The second one is similarity measurement.

Digitized images are stored in databases as 2-dimensional pixel intensities without inherit meaning. Thus, extracting meaningful, useful, and accurate information from those raw data is the main issue. This process is called feature extraction. The extracted features, also called *image signatures* [37], can be represented by [13] *symbolic values*, *numeric values*, *linguistic values*, *attribute relational graphs* (ARG), and *spatial relations*. In this dissertation we focused on numeric vector features, which represent the content of each image in the database as a vector, called *feature vector*.

In contrast to TBIR systems, CBIR systems are focused on similarity matching, not exactness. Although exactness is a precise concept, similarity matching is an approximation based on the similarity function applied on a pair of signatures of images [38].

2.1.4.1 Feature Extraction Methods

Visual features of images are extracted by using image processing, pattern recognition, and computer vision methodologies [39, 40, 41]. The most common features are based on color, texture, and shape properties of images. They are also known as low-level image contents or low-level visual features [42, 43]. Veltkamp and Tanese [44] analyze 56 CBIR system and found that 46 of them use any kind of color features, 38 of them use texture features and 29 of them use shape

features.

Color is an important attribute for human perception [45]. Also, it is the most widely used visual features in image retrieval [9]. Colors are represented as a point in 3-D color space. Widely used color spaces in CBIR are: *RGB, Munsell, CIE L*a*b*, CIE L*u*v*, HSV, and CMYK*. The uniformity is the most desirable property of a color space [46]. Uniformity means that two color pairs that are equal in similarity distance in a color space are perceived as equal by human viewers.

In a CBIR system, before analyzing an image, color resolution of the image is reduced in order to decrease the computational complexity. This process is called *color quantization*. Popular color quantization schemas are *uniform quantization, vector quantization, tree-structured vector quantization and product quantization* [47, 48].

Color descriptors include *color moments* [49], *color histogram* [50, 51], *color coherence vector* [52], and *color correlogram* [53].

Color histogram is the most extensively used descriptor [19]. It describes the distribution of each color in the color space of the image. Features extracted from color histogram are invariant to image rotation, translation and viewing axis [50].

Query by shape is the most important component of the CBIR systems. In general shape descriptors for image retrieval can be divided into two categories, boundary-based and region based. Boundary-based descriptors extract features from the contour or border of the object shape and internal details are ignored. On the other hand, the region-based methods consider internal details as well as the boundary details [54]. Well known shape descriptors are *aspect ratio*, *circularity*, *moment invariants* [55], *active contour models or snakes* [56], *Fourier descriptors* [57], and *perceptual shape descriptor* [58].

In recent years, **textural information** has been widely used as a visual primitive in many CBIR systems [6, 7, 8, 9]. This dissertation primarily focuses on texture descriptors for CBIR systems. Thus, the concept of texture is

analyzed in detail in the following chapters.

The above features can be global or local. Global features extract visual information from the whole image, whereas local ones concentrate on the regions or objects of an image.

2.1.4.2 Similarity Measures

Finding a good similarity measure that match with human perception between images based on the feature set is a challenging task. Also, how human judge the similarity between images is an active research area [59, 4].

The retrieval result is a list of images ranked by their similarity with the query image. Therefore, a typical CBIR system should calculate visual similarities between a query image and images in the database. Given two images, I_x and I_y, with their, $\bar{x}$ and $\bar{y}$, there is a function that gives the distance between their signatures, *i.e.* feature vectors. The similarity can be defined in terms of distance function as follows:
An image x is more similar to y than another image z in the image database I, when

$$D\big(\bar{x}, \bar{y}\big) < D\big(\bar{x}, \bar{z}\big), \qquad \forall I_z \in I \ , \tag{2.1}$$

where $D(\boldsymbol{.}\ , \boldsymbol{.})$ is a distance function, $\bar{x}$, $\bar{y}$, and $\bar{z}$ are the feature vectors of image I_x, I_y, and I_z respectively. However, the form of the distance function $D(\boldsymbol{.}\ , \boldsymbol{.})$ depends on the form of the feature representation, in general it is assumed that features are represented as an M dimensional vector, where x_i represents the ith feature value of the feature vector $\bar{x}$.

Since similarity can be defined using distance function, the term distance function and similarity function are used interchangeably throughout this dissertation.

A distance function between two points can be classified as metric if it satisfies the following axioms. Otherwise, the function is considered as non-metric [60].

Metric Axioms

A set X with elements, called points, is called a metric space if for any two points a and b in X, there is a number $D(a,b) \in \mathbf{R}$ called the distance from a to b such that

1. $D(a,b) \geq 0$ (Non-negativity) ,
2. $D(a,a) = 0$ (Identity) ,
3. $D(a,b) = D(b,a)$ (Symmetry) ,
4. $D(a,b) \leq D(a,c) + D(c,b) \quad \forall c \in X$ (Triangle inequality) .

A similarity function does not need to satisfy the properties of distance metrics. However, the metric distances have two main advantages [61]; computational efficiency, and well-studied mathematical theory. Nevertheless, psychophysical experiments show that human perception does not agree with the metric axioms [25].

Some of the popular distance functions can be summarized as follows [62, 63, 64]:

Minskowski Metrics or $\mathbf{L_r}$ – norms: It is a general class of distance metrics and defined as follows:

$$D_r\big(\bar{x},\bar{y}\big) = \sqrt[r]{\sum_{i=1}^{M} \big|x_i - y_i\big|^r} \ . \tag{2.2}$$

When L_1 norm is used, it is called *Manhattan/city block/taxi/absolute value* distance, as shown below.

$$D_1\big(\bar{x},\bar{y}\big) = \sum_{i=1}^{M} \big|x_i - y_i\big| \tag{2.3}$$

When L_2 norm is used, it is called *Euclidean* distance.

$$D_2\big(\bar{x},\bar{y}\big) = \sqrt{\sum_{i=1}^{M} \big|x_i - y_i\big|^2} \tag{2.4}$$

For $r = \infty$, it is called *Chebyshev* distance.

$$D_\infty\big(\bar{x},\bar{y}\big) = max\ _{\forall i}\big|x_i - y_i\big| \tag{2.5}$$

Dot Product:

$$\bar{x} \odot \bar{y} = \bar{x}^T \bar{y} = ||\bar{x}|| \;\; ||\bar{y}|| \;\; \cos\left(\bar{x}, \bar{y}\right) \tag{2.6}$$

Similarity Rule:

$$S\left(\bar{x}, \bar{y}\right) = \frac{\bar{x} \odot \bar{y}}{\bar{x} \odot \bar{x} + \bar{y} \odot \bar{y} - \bar{x} \odot \bar{y}} \tag{2.7}$$

Normalized Correlation:

$$\rho\left(\bar{x}, \bar{y}\right) = \frac{\bar{x} \odot \bar{y}}{\sqrt{\left(\bar{x} \odot \bar{x}\right) \left(\bar{y} \odot \bar{y}\right)}} \tag{2.8}$$

Camberra Distance:

$$D\left(\bar{x}, \bar{y}\right) = \sum_{i=1}^{M} \frac{|x_i - y_i|}{|x_i + y_i|} \tag{2.9}$$

Quadratic Form Distance:

$$D\left(\bar{x}, \bar{y}\right) = \sqrt{\left(\bar{x} - \bar{y}\right)^T \; A \; \left(\bar{x} - \bar{y}\right)} \; , \tag{2.10}$$

where $A = [a_{ij}]$ is a problem specific positive definite similarity matrix, and a_{ij} denotes the similarity between feature i and j. Unlike the Minskowski distance, Quadratic form considers similar feature pairs defined in the matrix A. It is generally used for color histogram-based image retrieval.

Mahalanobis Distance: It may help a CBIR system, when features are correlated to each other. It is defined as,

$$D\left(\bar{x}, \bar{y}\right) = \sqrt{\left(\bar{x} - \bar{y}\right)^T \; C^{-1} \; \left(\bar{x} - \bar{y}\right)} \; , \tag{2.11}$$

where C is the covariance matrix of the feature vectors. If the features are independent to each other, then Mahalanobis distance can be simplified as follows:

$$D\left(\bar{x}, \bar{y}\right) = \sqrt{\sum_{i=1}^{M} \frac{\left(x_i - y_i\right)^2}{\sigma_i}} \; , \tag{2.12}$$

where σ_i is the variance of the ith feature.

Bhattacharyya Distance: It can be useful, when underlying distributions differ only by their second order statistics [65, 66].

$$D\Big(\bar{x},\bar{y}\Big) = \frac{1}{4}(\bar{x}-\bar{y})^T\ (C_x+C_y)^{-1}\ (\bar{x}-\bar{y}) + \frac{1}{2}\ \ln\frac{|\frac{C_x+C_y}{2}|}{\sqrt{|C_x||C_y|}}\ , \tag{2.13}$$

where C_x and C_y are the covariance matrices of $\bar{x}$ and $\bar{y}$, respectively.

Kullback Leibler (KL) Divergence: KL is also known as *relative entropy*. Since entropy is a measure of the uncertainty over the true content of a message, KL measures how compact one feature distribution can be coded using the other one as the codebook. It can be defined as:

$$D\Big(\bar{x},\bar{y}\Big) = \sum_{i=1}^{M} x_i \log\frac{x_i}{y_i}\ . \tag{2.14}$$

Jeffrey Divergence: Unlike KL-divergence, Jeffrey Divergence is symmetric and numerically more stable.

$$D\Big(\bar{x},\bar{y}\Big) = \sum_{i=1}^{M} x_i \log\frac{x_i}{\hat{x}_i} + y_i \log\frac{y_i}{\hat{x}_i}\ , \tag{2.15}$$

where

$$\hat{x}_i = \frac{x_i+y_i}{2}\ .$$

Histogram Intersection: It is defined as

$$D\Big(\bar{x},\bar{y}\Big) = \frac{\sum_{i=1}^{M}\min\Big(x_i,y_i\Big)}{\sum_{i=1}^{M} y_i}\ . \tag{2.16}$$

Many other distance functions exist in the literature. However, the success of a particular distance depends on the data. Generally, given a feature space, researchers try to find the best distance function by trial and error.

Some distance functions intuitively work on distributions of feature vectors, such as *Mahalanobis*, *Bhattacharyya*, and *Quadratic form* distances, while the others require feature vectors, *e.g.*, *Euclid*, *Normalized Correlation*, *Similarity Rule* etc.

In most of the distance functions, if one of the feature elements has a relatively large range comparing to the others, it can subdue the other feature elements. In order to prevent such problems, *normalization* and/or *weighting* is employed. In the normalization phase, mostly, feature values are divided by their corresponding standard deviations. On the other hand, in the weighting phase, effects of the feature attributes can be increased or decreased by changing their corresponding weights.

2.1.5 Popular CBIR Systems

Several content-based image retrieval systems have been proposed and some of them are available as commercial packages. In the following, some of the popular ones are summarized:

QBIC: It is commercially available and developed by IBM. Color, shape and texture features are used to represent the images. Multidimensional indexes are created by the use of R trees. For more detailed information see `http://wwwqbic.almaden.ibm.com`

Virage: It is another well-known commercial CBIR system developed by Virage Inc. Virage is a module-based system that system developers can add their own modules. For demonstration see `http://www.virage.com`

Photobook: It is an experimental system developed by a research group at Massachusetts Institute of Technology. The initial version was designed to assist the user in annotation. Search and retrievals are implemented by the use of appearance, 2d shape and texture features. Textual annotations can, also, be employed. Further information is available at the web site
`http://vismod.www.media.mit.edu/~tminka/photobook`

VisualSEEk: VisualSEEk was developed at Columbia University. System retrieves images according to the spatial relationship of the regions. Further information is available at `http://www.ctr.columbia.edu/VisualSEEk`

Netra: This experimental system is developed by Ma and Manjunath in the UC Santa Barbara. It provides region-based searching, based on local image

properties. These are color, texture, shape and spatial location information. A Web demonstration is available at `http://maya.ece.ucsb.edu/Netra`.

A more detailed survey of available CBIR system can be found in [19] and [44]. Roughly speaking today's CBIR systems suffer from two main problems [67]. Firstly, user interfaces are too complicated for an average user. Second problem is the unsatisfactory results and the long response time. Thus, the techniques used in CBIR are still subject of substantial development.

2.2 Summary

In this chapter, we briefly survey the image retrieval techniques for large image databases. In the classical text-based retrieval, each image in the database is indexed with a set of relevant text phrases by a human indexer. Due to human perception subjectivity and laborious indexing task, TBIR systems cannot answer the needs and expectations, especially, for large scale image collections. In order to improve the quality of search and browse, intensive research efforts have been carried out since 1990's.

In CBIR, the images are automatically indexed by their own visual properties. CBIR systems can be categorized according to the query type that they can handle. Thus, query types are analyzed and classified into three levels; level 1 (*i.e. low-level*), level 2 (*i.e. logical*), and level 3 (*i.e. abstract*). Low-level type queries can be interpreted by using color, texture, and shape features of the image. On the other hand, logical and abstract type queries require some degree of reasoning and domain specific knowledge. Although the users' higher-level of queries remain unsolved, currently most of the CBIR systems are focused on low-level ones.

In a typical content-based image retrieval system, the query pattern is query-by-example, which searches the top N-closest images to a query image. Similarity is expressed by mathematical distance function. Although various distance functions are defined, none of them give satisfactory results for all type of feature.

CHAPTER 3

TEXTURE

In this chapter, first the definitions of texture are discussed. Then, texture analysis and various texture descriptors found in the literature are presented.

3.1 What is Texture ?

The word texture comes from the Latin word *textura*, which means textile fabric [68]. Tree barks, water, bricks, clouds, leather, and soil are some real-life examples as seen in Figure 3.1 . The concept of texture is intuitively obvious for us, but it is hard to define. Although there is no formal definition, we describe texture as fine, coarse, grained, smooth, regular/irregular, directional, *etc.* Nevertheless, these descriptions are imprecise and non-quantitative.

Roughly speaking, textures can be classified as *regular* or *irregular*. Regular textures are composed of structurally repeated similar patterns (like brick wall) whereas irregular ones like cloud or grass, cannot be constructed by regularly arranged patterns.

Texture carries an important role in many areas of image processing, especially in classification, image segmentation, image retrieval, realism in computer graphics and image encoding. It also provides depth and orientation of an object.

Although researchers have been studying on the subject since 1970, no one

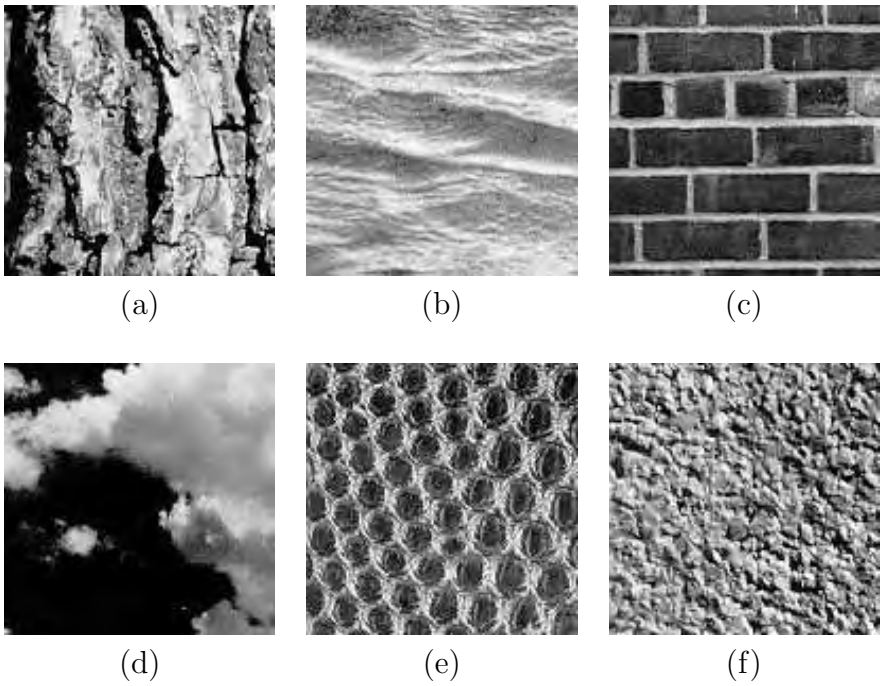


Figure 3.1: Sample textures: (a) tree bark, (b) water, (c) brick wall, (d) cloud, (e) leather, and (f) soil.

gives a widely accepted mathematical definition of texture yet.

In the literature, many of the studies give a try for defining texture. Below some partial definitions of texture are given.

- "The term texture generally refers to repetition of basic texture elements called texels. The texel contains several pixels, whose placement could be periodic, quasi-periodic or random. Natural textures are generally random, whereas artificial textures are often deterministic or periodic. Texture may be coarse, fine, smooth, granulated, rippled, regular, irregular or linear." [39]

- "Texture is characterized not only by the gray value at a given pixel, but also by the gray value 'pattern' in a neighborhood surrounding the pixel." [69]

- "We consider a texture to be stochastic, possibly periodic two dimensional image field" [70]

- "A region in an image has a constant texture if a set of local statistics or other local properties of the picture function are constant, slowly varying or approximately periodic" [71]

- "We may regard texture as what constitutes a macroscopic region. Its structure is simply attributed to the repetitive patterns in which elements or primitives are arranged according to a placement rule." [72]

- "Texture is an apparently paradoxical notion. On the one hand, it is commonly used in the early processing of visual information, especially for practical classification purposes. On the other hand, no one has succeeded in producing a commonly accepted definition of texture. The resolution of this paradox, we feel, will depend on a richer, more developed model for early visual information processing, a central aspect of which will be representational systems at many different levels of abstraction. These levels will most probably include actual intensities at the bottom and will progress through edge and orientation descriptors to surface, and perhaps volumetric descriptors. Given these multi–level structures, it seems clear that they should be included in the definition of, and in the computation of, texture descriptors." [73]

- "Texture has been extremely refractory to precise definition." [74]

Although there is no universally agreed definition, almost all researchers agree on;

- while color is a point property, texture is a local-neighborhood property [75, 76]

- a texture is a region that can be perceived as being spatially homogeneous in some sense [68]

Scale is a crucial concept that must be considered, when dealing with textures, because, the same texture at various scales may be perceived as different [77].

Thus, there may be several levels of completely different textures in the same image, but at different scales.

3.2 Texture Analysis

In general, three different types of texture analysis are considered. These are *classification*, *segmentation* and *synthesis*.

Texture classification deals with the recognition of objects and/or image regions using texture properties. In other words, given a texture image, finding its class is a texture classification problem. The classes may be specified a priori by an analyst or automatically detected during the classification process. While the former is called *supervised classification*, the latter is called *unsupervised classification*.

There is a tight relation between classification and similarity search. While the former gives exact results, the latter provides a list of results ranked by the similarity measure.

The objective of texture segmentation is to separate an image into regions of distinct textural behavior to obtain a boundary map. Texture segmentation is very useful for image understanding such as retrieving images with similar texture from a database. Figure 3.2 shows a sample textured image and its segmented version.

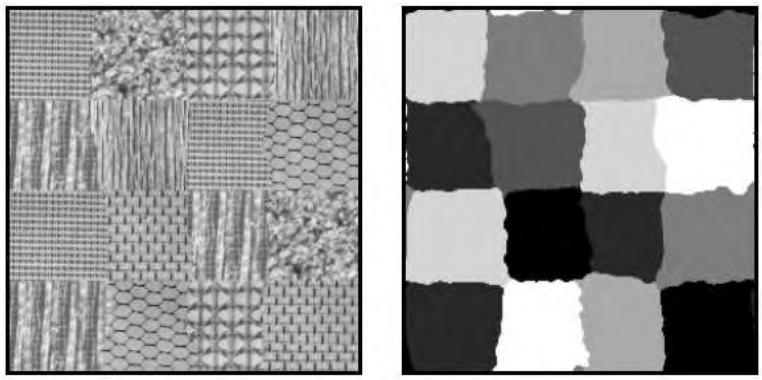


Figure 3.2: A textured image and its segmented version.

Classification and segmentation share closely related objectives. Classification can lead to segmentation and vice-versa. Broadly speaking, texture segmentation can be viewed as a pixelwise texture classification with no a priori knowledge of the number of texture components or the properties of each component [68].

Texture synthesis directly related to realistic scene generation. The goal of the texture synthesis can be summarized as follows: Given a texture sample, synthesize a new texture that, when perceived by human, appears to be generated by the same underlying process [78]. Figure 3.3 shows a texture sample and its synthesized version.

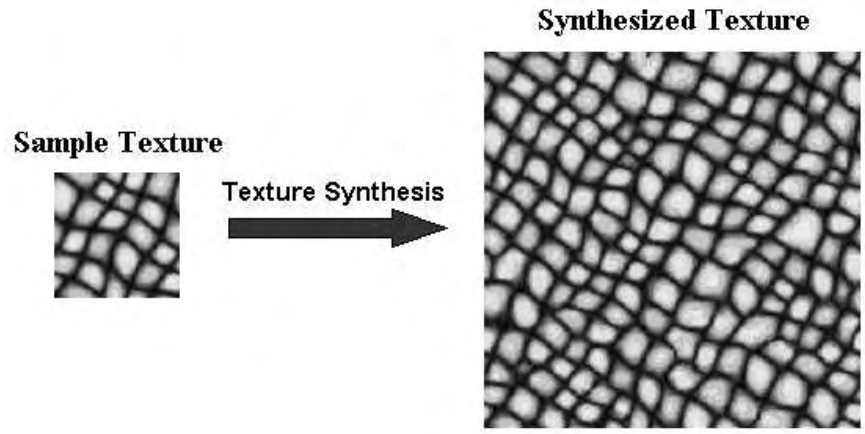


Figure 3.3: A texture sample and its synthesized version.

3.3 Texture Analysis in Grayscale or Color ?

It is shown that the retrieval performance resulting from the combination of structure, color, and texture is superior than using them alone [79]. Thus, recently, integrative color texture models based on Wavelets [80], Gabor filter [81], co-occurrence [82], and auto covariance [83] have been proposed.

Although most of the real-life textures are colored as exemplified in Figure 3.4, large number of researches, including this dissertation, has been focused on grayscale texture analysis [84], because of the fact that even for grayscale

textures, existing descriptors are still not powerful to represent textural properties of an image. Generally, in order to analyze color texture, color signal is divided into luminance and chrominance components. Textural features are then extracted from the luminance part.

Figure 3.4: Four color texture samples.

3.4 The Importance of Texture in Content-Based Image Retrieval

Due to the complexity of texture in real life, it is very hard to categorize them using keywords alone. Also, texture images generally contain unique visual patterns or spatial arrangements of pixels, so that describing textures, gray-level or color alone, may not yield to classify similar ones [85]. For example, without employing textural properties sky and sea cannot be distinguished from each other, since they have similar colors.

Texture features such as contrast, uniformity, coarseness, roughness, regularity, frequency, density and directionality provide significant information for image interpretation and classification [86]. Such features can be used for, not only separating regions in an image, but also identifying the content of an image.

Texture features have been successfully used to provide a meaningful tool for searching image databases. Given a query image as a key, a CBIR system should search the image database and retrieve the most similar N-images. Such searching mechanism is called *query-by-texture* [85].

Assuming that an image is a mosaic of textures, some example queries can be the followings:

- "Identify all parking lots in the aerial image shown in Figure 3.5"

  Hint: A parking lot with cars parked at regular intervals can be interpreted as a texture when viewed from a distance.

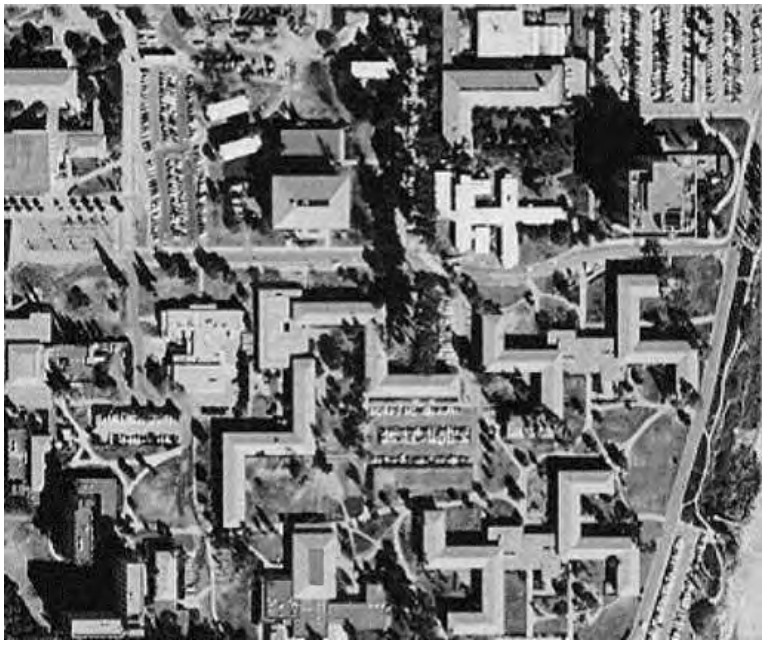


Figure 3.5: A sample aerial image.

- "Forecast the tomorrow's weather by analyzing the satellite image of the Marmara area shown in Figure 3.6"

  Hint: Clouds are textures.

Figure 3.6: Satellite image of the Marmara area.

3.5 Feature Extraction Methods for Texture Analysis

Texture analysis methods extract textural information from an image. Like the definition of texture, it is not easy to categorize texture analysis methods in the literature. Since most of them use hybrid methods or slightly different versions of the existing methods. Various taxonomy of the methods for texture analysis is proposed [86, 39]. For the purpose of this dissertation, texture descriptors are classified into three main categories, namely *statistical*, *structural*, and *spectral*.

3.5.1 Statistical Methods

Textures that are random in nature are well suited for statistical characterization. The main disadvantage of the statistical methods is the ignorance of geometrical and structural information. Moreover, the features, directly extracted from gray values of an image, are sensitive to noise and monotonic shifts in the grayscale [12].

3.5.1.1 First-Order Statistical Methods

The first-order gray level statistics can be derived from the information provided by the intensity histograms. It is also called *Gray Level Distribution Moments*. Assume that, G is the number of gray levels and h_i is the number of pixel in an image with gray level i, then the normalized histogram H_i is defined as $H_i = \frac{h_i}{N}$. Statistics computed from H_i include:

1. The *mean* gray level
$$\mu = \sum_{i=0}^{G-1} iH_i \ , \tag{3.1}$$
where μ measures the average intensity in the image.

2. The gray level *standard deviation*
$$\sigma = \sqrt{\sum_{i=0}^{G-1} (i-\mu)^2 H_i} \ , \tag{3.2}$$
where σ measures the global contrast in the image.

3. The *coefficient of variation*
$$cv = \frac{\sigma}{\mu} \ , \tag{3.3}$$
where cv is a measure of relative dispersion.

4. The *skewness*
$$\gamma_1 = \frac{1}{\sigma^3} \sum_{i=0}^{G-1} (i-\mu)^3 H_i \ , \tag{3.4}$$
where γ_1 measures the symmetry of the histogram. A histogram is called symmetric if it looks the same to the left and right of the mean values.

5. The *kurtosis*
$$\gamma_2 = \frac{1}{\sigma^4} \sum_{i=0}^{G-1} (i-\mu)^4 H_i - 3 \ , \tag{3.5}$$
where γ_2 is a measure of whether the histogram is peaked or flat relative to a normal distribution.

6. The *energy*
$$e = \sum_{i=0}^{G-1} H_i^2 \ , \tag{3.6}$$
where $G^{-1} \leq e \leq 1$. It measures the nonuniformity of the histogram.

7. The *entropy*

$$s = -\sum_{i=0}^{G-1} H_i \log H_i \ . \tag{3.7}$$

Entropy measures the uniformity of the histogram.

The above measures, either alone or combination of them, can be used for representing a texture. However, perceptually different textures may have the same first-order statistics. Thus, first-order statistics on gray values of an image is seldomly successful tool in CBIR systems.

3.5.1.2 Second-Order Statistical Methods

One of the most popular traditional statistical method is the *gray level cooccurrence matrices (GLCM)* introduced by Haralick [74]. The use of GLCM in texture analysis is also called, *the spatial gray level dependence method* (SGLDM).

The method relies on the idea that each pixel is correlated with its neighbors. A GLCM, denoted as c, is defined with respect to given (row,column) displacement h, and element c_{ij}, is the number of points having gray level j occurs in position h, relative to a point having gray level i. Let N_h be the total number of pairs, then $C_{ij} \approx c_{ij}/N_h$ denotes the elements of the normalized GLCM, C [68]. GLCM can be viewed as a directional histogram, which considers relative orientation of gray levels in an image.

Haralick defined 14 different texture features, each of which is extracted from the GLCM. Some of them are given below [68, 87]:

1. *Energy or Angular Second Moment* is defined as

$$\varepsilon = \sum_{i=0}^{G-1} \sum_{j=0}^{G-1} C_{ij}^2 \ , \tag{3.8}$$

where $G^{-2} \leq \varepsilon \leq 1$. Energy is a measure of homogeneity in the image. Homogeneous images do not have dominant gray-value transitions.

2. *Entropy* measures the randomness of an image from

$$\mathcal{S} = -\sum_{i=0}^{G-1} \sum_{j=0}^{G-1} C_{ij} \log C_{ij} \ , \tag{3.9}$$

where $0 \leq \mathcal{S} \leq \log G^2$. High entropy means that the elements of the cooccurrence matrix are almost equal. This is possible for random textures.

3. *Contrast* measures the local variations of gray levels in an image from

$$\mathcal{C} = \sum_{i=0}^{G-1} \sum_{j=0}^{G-1} (i-j)^2 C_{ij} \ . \tag{3.10}$$

In other words, it is a measure of coarseness of a texture.

4. *Homogeneity* is a measure of monotonicity and is calculated as

$$\mathcal{H} = -\sum_{i=0}^{G-1} \sum_{j=0}^{G-1} \frac{C_{ij}}{1+|i-j|} \ . \tag{3.11}$$

5. *Autocorrelation* measures the gray level linear dependencies in the image and is defined as

$$\rho = \sum_{i=0}^{G-1} \sum_{j=0}^{G-1} \frac{(i-\mu_x)(j-\mu_y)C_{ij}}{\sigma_x \sigma_y} \ , \tag{3.12}$$

where $-1 \leq \rho \leq 1$.

6. *Diagonal Moment* measures the difference in correlation for both high gray levels and low gray levels and is calculated as

$$\mathcal{D} = \sum_{i=0}^{G-1} \sum_{j=0}^{G-1} |i-j|(i+j-\mu_x-\mu_y)C_{ij} \ . \tag{3.13}$$

There are two main disadvantages of using GLCM [86]. First, establishing GLCM is computationally expensive. Secondly, it is difficult to select most relevant features among a large number of feature sets computed from the GLCM for a given displacement h.

3.5.1.3 Higher-Order Statistical Methods

There are two main approaches in higher-order statistical methods; *gray level run length matrix* and *neighboring gray level dependence matrix*.

Gray Level Run-Length Matrix (GLRLM)

A set of consecutive pixels with the same gray level is called a gray level run. The number of pixels in a run is the run-length. In order to extract texture features *gray level run length matrix (GLRLM)* are computed. Each element, r_{ij}, of the GLRLM represents the number of runs of gray level i having length j. R_{ij} is the normalized r_{ij}. GLRLM can be computed for any direction. The features derived from the GLRLM is tabulated in Table 3.1.

Table 3.1: The features derived from the GLRLM.

Short Runs Emphasis	$RF1 = \sum_{i=0}^{G-1} \sum_{j=1}^{L} \frac{R_{ij}}{j^2}$
Long Runs Emphasis	$RF2 = \sum_{i=0}^{G-1} \sum_{j=1}^{L} j^2 R_{ij}$
Gray Level Nonuniformity	$RF3 = \sum_{i=0}^{G-1} [\sum_{j=1}^{L} R_{ij}]^2$
Run Length Nonuniformity	$RF4 = \sum_{j=1}^{L} [\sum_{i=1}^{G-1} R_{ij}]^2$
Run Percentage	$RF5 = \frac{N_r}{N}$

The features in Table 3.1 are the statistics of runs in an image and can be used as a texture descriptor. However, they suffer from the sensitivity to noise [88]. They are relatively better in representing binary textures [68].

Neighboring Gray Level Dependence Matrix (NGLDM)

Unlike gray level run length matrix method, neighboring gray level dependence matrix approach considers all neighbors of a pixel. Given a distance d, each element, q_{ks}, of a NGLDM represents the number of pixels with gray level k having s neighbors with the same gray levels. Let, Q_{ks} is the normalized version

of q_{ks}. Table 3.2 shows the features derived from the NGLDM.

Table 3.2: The features derived from the NGLDM.

Small Number Emphasis	$N1 = \sum_{k=0}^{G-1} \sum_{s=0}^{S} \frac{Q_{ks}}{1+s^2}$
Large Number Emphasis	$N2 = \sum_{k=0}^{G-1} \sum_{s=0}^{S} s^2 Q_{ks}$
Number Nonuniformity	$N3 = \sum_{s=0}^{S} [\sum_{k=0}^{G-1} Q_{ks}]^2$
Second Moment	$N4 = \sum_{k=0}^{G-1} \sum_{s=0}^{S} Q_{ks}^2$
Entropy	$N5 = -\sum_{k=0}^{G-1} \sum_{s=0}^{S} Q_{ks} \log Q_{ks}$

The features in Table 3.2 are, also, sensitive to noise. Therefore, like GLRLM features, they are better in representing binary textures.

3.5.1.4 Tamura Features

Tamura defines six features including coarseness, contrast, directionality, line-likeness, regularity, and roughness in 1978 [72]. These features are defined as follows:

Coarseness:

It is a measure of the granularity of the texture. A moving window with size $2^k \times 2^k$ $(k = 0, 1, ..., 5)$ is defined for each pixel (x, y). Then, moving averages $A_k(x, y)$ can be computed as follows:

$$A_k(x, y) = \sum_{i=x-2^{k-1}}^{x+2^{k-1}-1} \sum_{j=y-2^{k-1}}^{y+2^{k-1}-1} f(i, j)/2^{2k} \ , \tag{3.14}$$

where $f(i, j)$ is the intensity at pixel (i, j).

First, the differences between pairs of non-overlapping moving averages in the horizontal and vertical directions for each pixel are computed as follows:

$$E_{k,h}(x, y) = |A_k(x + 2^{k-1}, y) - A_k(x - 2^{k-1}, y)| \ , \tag{3.15}$$

and

$$E_{k,v}(x,y) = |A_k(x, y + 2^{k-1}) - A_k(x, y - 2^{k-1})|. \tag{3.16}$$

Then, the value of k that maximizes E in either direction is used to set the best size for each pixel,*i.e.*

$$S_{best}(x,y) = 2^k \ . \tag{3.17}$$

The coarseness is the average value of the S_{best} over the entire image and is defined as follows:

$$C = \frac{1}{m \times n} \sum_{i=1}^{M} \sum_{j=1}^{N} S_{best}(i,j) \ . \tag{3.18}$$

Contrast:

It measures the variations of gray levels in the image and can be defined as

$$C_t = \frac{\sigma}{\alpha_4^{1/4}} \ , \tag{3.19}$$

where α_4 is the kurtosis and

$$\alpha_4 = \frac{\mu_4}{\sigma^4} \ , \tag{3.20}$$

where μ_4 is the fourth moment about the mean.

Directionality:

In order to compute directionality, image is convoluted with 3×3 vertical and horizontal edge masks. The angle of gradient vector at each pixel is defined as:

$$\theta = \tan^{-1}(\Delta_V / \Delta_H) + \pi/2 \ , \tag{3.21}$$

where Δ_V and Δ_H are measured using the following 3×3 moving window operators:

$$\begin{pmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{pmatrix} \quad , \quad \begin{pmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{pmatrix} . \tag{3.22}$$

After quantizing θs, histogram of θ (H_D) is constructed. Strong peaks in the histogram indicate that the image is highly directional. Then, directionality measure can be defined as:

$$\mathcal{D} = \sum_{p}^{n_p} \sum_{\forall \theta \in w_p} (\theta - \theta_p)^2 H_D(\theta) \ , \tag{3.23}$$

where $\mathcal{D}$ is a measure of directionality, p is a peak, w_p is the set of bins, θ_p is the bin that takes the peak value.

3.5.1.5 Markov Random Fields

Markov Random Field models have been applied to various image processing applications such as texture synthesis [70], classification [89], image segmentation, restoration and compression. MRF model successfully represent the textures, which consist of small primitives.

MRF is a probabilistic process in which all interactions are local; the probability that a cell is in a given state is determined by the probabilities for states of neighboring cells.

The image is usually represented by an $M \times N$ lattice denoted by

$$\mathcal{L} = \{(i,j)|1 \leq i \leq M, 1 \leq i \leq N\} \ . \tag{3.24}$$

$f(i,j)$ is a random variable, which represents the gray level at location (i,j) on lattice $\mathcal{L}$. The Markovianity can be defined as:

$$P(f(i,j)|\mathcal{L}) = P(f(i,j)|\eta_{i,j}) \ , \tag{3.25}$$

where $\eta_{i,j}$ is the neighboring set of pixel (i,j). Different forms of probability distributions yield different MRF models. Widely used models include the Gaussian MRF (GMRF), and the simultaneous autoregressive (SAR) model [90].

The GMRF is a stationary, noncausal 2-D autoregressive process and described by following equation:

$$f(i,j) = \sum_{(k,l)\in\eta_{i,j}} \beta_{k,l} f(k,l) + \varepsilon_{i,j} \ , \tag{3.26}$$

where $\{\varepsilon_{i,j}\}$ is a stationary Gaussian noise sequence with zero mean and $\beta_{k,l}$ are the weights associated with each of the neighboring pixels.

If ε is an independent, identically distributed, zero mean and unit variance noise, *i.e.* white noise, then the model is called simultaneous autoregressive model (SAR).

Compared to other MRF models, SAR model uses fewer parameters. However, it is not rotation invariant. By changing the neighborhood set definition, *rotation-invariant SAR model* can be defined [91].

In order to describe texture in different granularities and to enable multi-scale texture analysis, the *multi-resolution simultaneous autoregressive model* (RISAR) is proposed in [92]. An image is represented by a multi-resolution Gaussian pyramid with lowpass filtering and sub-sampling, applied at several successive levels. At each level, either SAR or RISAR model can be used. Although it has a better performance than many other texture features, such as Wold decomposition and wavelet transformation [93], the MRSAR cannot distinguish images, when the structured pattern is involved.

3.5.1.6 Fractals

A fractal texture is characterized by self-similarity. Given a bounded set A in the Euclidean space, the set A is said to be self-similar, when A is the union of N non-overlapped copies of itself, each of which has been scaled down by a ratio of r. This self-similarity is quantified by its fractal dimension and defined as

$$D = \frac{\log\ N}{\log(1/r)}\ . \tag{3.27}$$

The fractal dimension characterized the roughness of a texture image [94, 95]. The main problems for the method are both the difficulty in finding the fractal dimension and the lack of self-similarity in most of the real-life textures. Another problem is that visually different textures may have equal fractal dimensions [96].

Recently, Kaplan [97, 95] has defined extended fractal features for database image retrieval, where the images are consisting of homogeneous textures. The features are based on Brownian motion model characterized by Hurst parameter. They conclude that multi-scale Hurst parameters allow better texture discrimination than the traditional methods.

3.5.2 Structural Methods

Structural methods are based on the theory of formal languages [39, 98]. A textured image is regarded as generation of repeated texture primitives, also called texture elements, using a set of placement rules [99]. For example; in a brick wall image, the primitive is a brick and the arrangements of bricks are determined by the placement rule [68].

Structural analysis is used, when texture elements can be clearly identified [100]. Although some randomness can be employed in the placement rules, structural methods and their variants work well on deterministic textures. However, real life textures are in general non-deterministic.

3.5.3 Spectral Methods

The most popular spectral feature extraction methods are Fourier, wavelet, and Gabor filtering and Wold decomposition [101, 93, 102]. Spectral methods extract features from the energy distribution in the frequency domain.

3.5.3.1 Fourier Transformation

Fourier transform based methods usually work on textures showing strong periodicity. Various number of features can be extracted from the Fourier power spectrum. Liu and Jernigan introduced 28 texture features derived from normalized Fourier transform coefficients [103] such as *rings*, *wedges*, *inertia*, *entropy*, *anisotropy*, and *etc.*

A power spectrum of a texture image can be used for measuring the periodicity and directionality information of the texture. For example, a fine texture has high frequency components, while coarse one has low frequency components. Figure 3.7 shows some texture images and their corresponding Fourier spectrums. Methods based on Fourier transformation perform poorly in practice, due to its lack of spatial information.

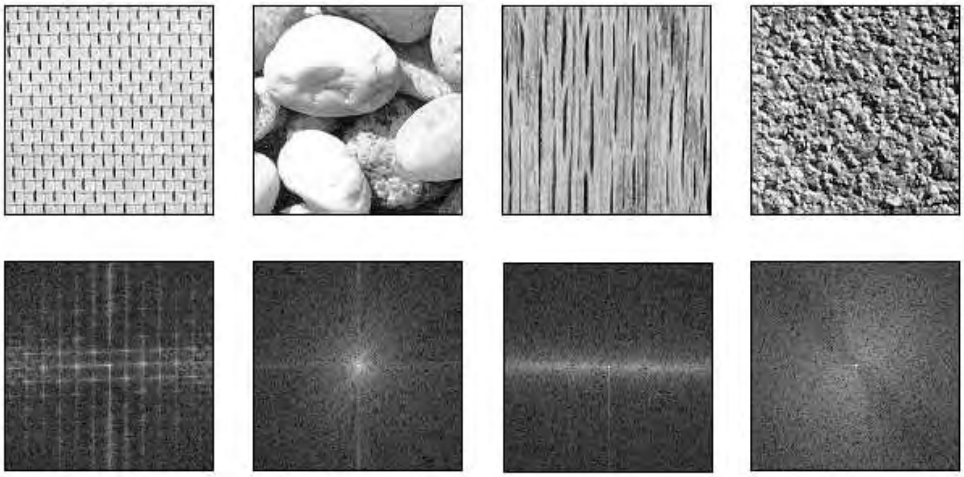


Figure 3.7: Texture images and their Fourier spectrums.

3.5.3.2 Wavelet Transform

Psycho-visual researches [104, 105] indicate that the human visual system process images in a multi-scale manner. This approach motivates the use of multi-scale or multi-resolution approaches for texture analysis. Wavelet transform provides a formal approach to texture analysis [106, 107, 37]. Figure 3.8 shows a sample texture and its corresponding wavelet transformation as a 2-D intensity diagram.

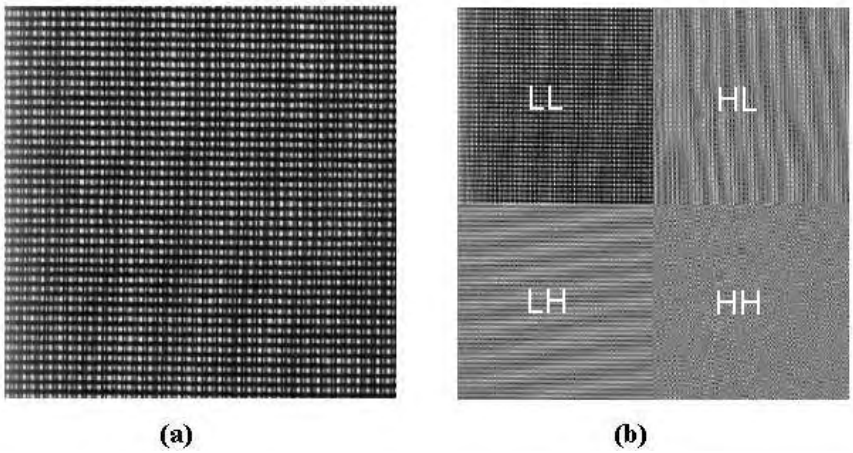


Figure 3.8: A sample texture (a) and its one level wavelet decomposition (b).

The standard *pyramidal wavelet decomposition* (PWT), as shown in Fig-

ure 3.9, is the process that repeats filtering iteratively for the low pass subimages. PWT is suitable for signals consisting of components with information concentrated in lower frequency channels.

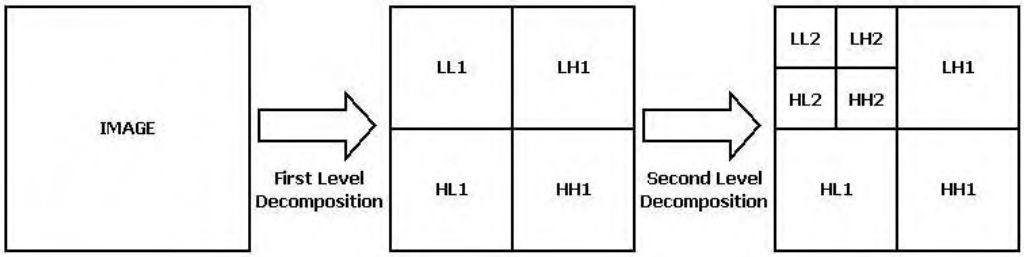


Figure 3.9: Pyramid-structured wavelet transform decomposition of an image.

When high pass subimages are further decomposed, as shown in Figure 3.10, the process is called *uniform tree-structured* (TWT) or *wavelet packet decomposition*. TWT provides a rich range of possibilities for analysis. In texture analysis, TWT features are more successful than that of PWT. This is basically because of the fact that dominant frequencies of textured images do not only lie in the lower resolution band, but perhaps in the middle band as well [108]. However, the TWT is sensitive to image size.

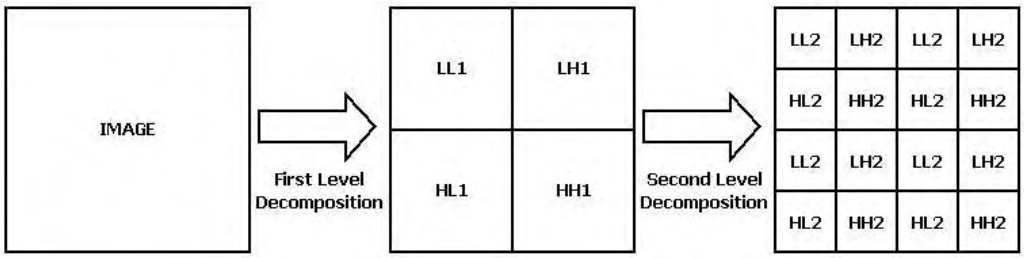


Figure 3.10: Uniform tree-structured wavelet transform decomposition of an image.

Textural information can be extracted from the transformed coefficients. Extracted information includes *mean*, *standard deviation*, *energy*, *moments*, *entropy*, *etc.*

3.5.3.3 Gabor Filters

The human visual cortex has separate cells that respond to different frequencies and orientations. The Gabor filter bank can localize the energy simultaneously both in spatial and frequency domains and Gabor functions model quite well the visual cortex [109]. Moreover, psycho-physiological experiments with Gabor filters for texture analysis show that Gabor filters perform remarkably similar to the human visual system [104, 105]. Because of these reasons, Gabor filter method is widely regarded as the state-of-the-art method in texture analysis [110].

Among many other Gabor filter feature extraction methods, the successful results are reported by Manjunath & Ma [93, 111]. For this reason, we follow their approach in the rest of this section.

A two-dimensional Gabor function is the Gaussians modulation of complex sinusoids. They can be defined as:

$$G(x,y) = \left(\frac{1}{2\pi\sigma_x\sigma_y}\right) \exp\left[-\frac{1}{2}\left(\frac{x^2}{\sigma_x^2} + \frac{y^2}{\sigma_y^2}\right)\right] \exp[2\pi jWx] \ , \tag{3.28}$$

where σ_x and σ_y are the standard deviations along X axis and Y axis, and W is the modulation frequency. The real and imaginary part of the function can be defined as follows:

$$G_{real}(x,y) = \left(\frac{1}{2\pi\sigma_x\sigma_y}\right) \exp\left[-\frac{1}{2}\left(\frac{x^2}{\sigma_x^2} + \frac{y^2}{\sigma_y^2}\right)\right] \cos[2\pi Wx] \ , \tag{3.29}$$

and

$$G_{imaginary}(x,y) = j\left(\frac{1}{2\pi\sigma_x\sigma_y}\right) \exp\left[-\frac{1}{2}\left(\frac{x^2}{\sigma_x^2} + \frac{y^2}{\sigma_y^2}\right)\right] \sin[2\pi Wx] \ . \tag{3.30}$$

Figure 3.11 shows 3-D profiles of the real and imaginary components of a Gabor function and Figure 3.12 depicts various Gabor filters as a 2-D intensity diagram.

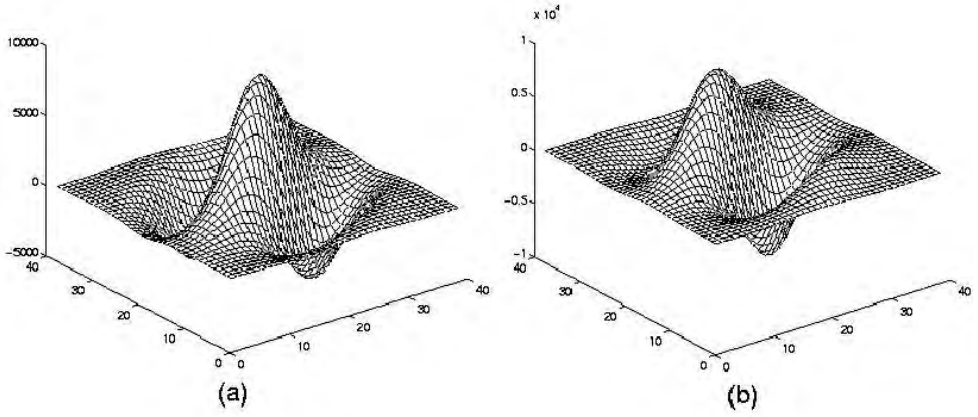


Figure 3.11: Gabor function in 3-D. (a) The real component. (b) The imaginary component.

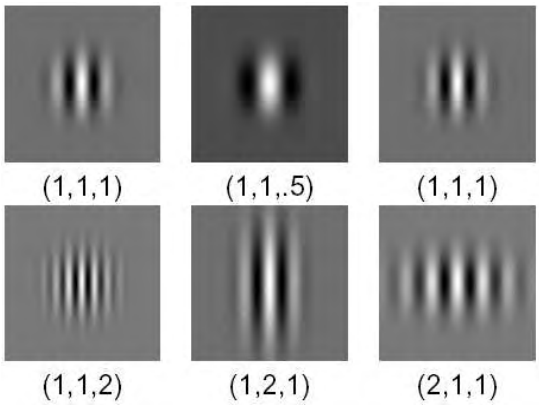


Figure 3.12: Different Gabor filters with varying (σ_x, σ_y, W).

A bank of Gabor filters can be generated by dilating and rotating the above Gabor function:

$$G_{mn} = a^{-m} G(x', y'), \quad a > 1,\ m, n = integer\ , \tag{3.31}$$

$$x' = a^{-m}(x\cos\theta + y\sin\theta)\ , \tag{3.32}$$

$$y' = a^{-m}(-x\sin\theta + y\cos\theta)\ , \tag{3.33}$$

where $\theta = n\pi/K$, $n = 0, 1, ..., K-1$ and $m = 0, 1, ..., S-1$; K is the total number of orientations and S is the total number of scales.

The variables in the above equations are defined as follows:

$$a = (U_h/U_l)^{\frac{1}{S-1}}\ , \tag{3.34}$$

$$W_{m,n} = a^m U_l \ , \tag{3.35}$$

$$\sigma_{x,m,n} = \frac{(a+1)\sqrt{2\ln 2}}{2\pi a^m (a-1) U_l} \ , \tag{3.36}$$

$$\sigma_{y,m,n} = \frac{1}{2\pi \tan(\frac{\pi}{2N})\sqrt{\frac{U_h^2}{2\ln 2} - (\frac{1}{2\pi\sigma_{x,m,n}})^2}} \ . \tag{3.37}$$

$U_l = 0.05$ and $U_h = 0.4$ are the commonly used constants in the literature.

The scale factor a^{-m} is meant to ensure the equal energy among different filters. Distinctive discontinuities between texture patterns are detectable only if Gabor filter parameters are suitably chosen [112]. In other words, in order to extract meaningful information, "appropriate" subset of filters must be employed. Figure 3.13 shows the filters in the spatial domain. The figure contains four scales ($S = 4$) and six orientations ($K = 6$).

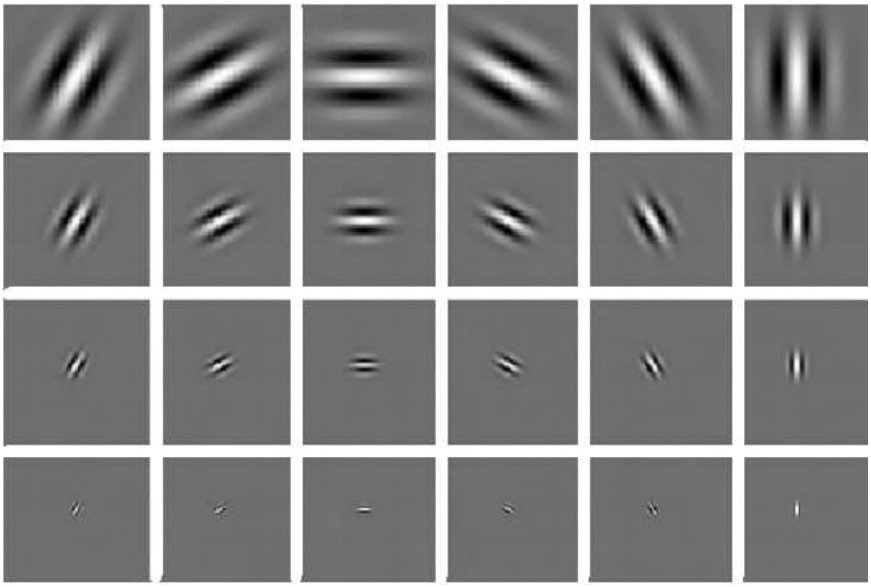


Figure 3.13: Intensity plot of Gabor Filters (real part) in the spatial domain.

Given image $I(x, y)$, its Gabor filtered output is defined as;

$$w_{mn}(x, y) = \int\int I(x, y) G^*_{mn}(x - x_1, y - y_1) dx_1 dy_1 \ . \tag{3.38}$$

Basically, each Gabor filter captures the energy at a specific frequency and a specific direction of an image. Figure 3.14 shows the Gabor filter responses of

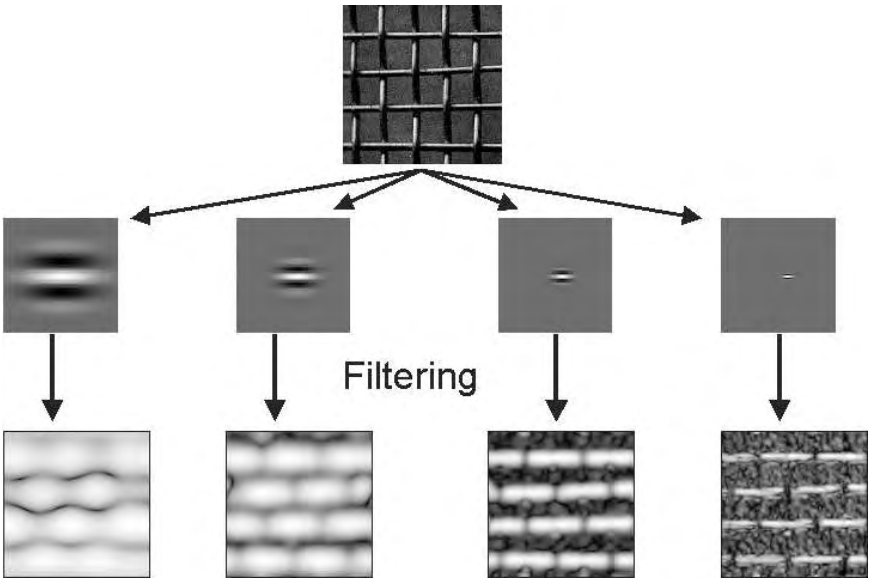


Figure 3.14: Horizontal Gabor filter responses (4 scale) of a given image.

an image. Second order statistics of the Gabor Filter responses of a given texture is used as a texture feature. Thus, an image is represented by a vector $\bar{f}$ [113].

$$\bar{f} = \left[\mu_{00}, \sigma_{00}, \mu_{01}, \sigma_{01}, ..., \mu_{(S-1),(K-1)}, \sigma_{(S-1),(K-1)}\right] \ , \tag{3.39}$$

where

$$\mu_{mn} = \int\int |w_{mn}(x,y)| \ dxdy \ , \tag{3.40}$$

$$\sigma_{mn} = \sqrt{\int\int (|w_{mn}(x,y)| - \mu_{mn})^2 \ dxdy} \ . \tag{3.41}$$

3.5.3.4 Wold Decomposition

According to the Wold theory [114, 115], a regular, homogeneous, random, 2D field $y(m,n)$, $(m,n) \in Z^2$ can be uniquely decomposed into three mutually orthogonal components as follows:

$$y(m,n) = w(m,n) + d(m,n) \ , \tag{3.42}$$

where field $\{d(m,n)\}$ is deterministic and field $\{w(m,n)\}$ is purely indeterministic. The field $\{d(m,n)\}$ can be further decomposed into

$$d(m,n) = v(m,n) + h(m,n) \ , \tag{3.43}$$

where $\{h(m,n)\}$ is harmonic field and $\{v(m,n)\}$ is a generalized evanescent field. The above defined three wold components, *harmonic*, *evanescent*, and *indeterministic* correspond to *periodicity*, *directionality*, and *randomness* of texture respectively [116]. These three types of texture feature is regarded as natural texture discrimination feature [116]. Figure 3.15 shows an example of periodic, directional and random texture.

(a) (b) (c)

Figure 3.15: Examples of (a) periodic, (b) directional, and (c) random real-life textures.

In general, harmonic component $h(m,n)$ can be detected by the use of Fourier transformation, whereas evanescent component $v(m,n)$ can be detected by the use of a Hough transformation [115]. Perceptually structured textures usually have dominant harmonic components, which appear as structured peaks in the frequency domain. On the other hand, strong evanescent components correspond to eminent directionality in patterns such that local inhomogeneities have only a minor effect on these components.

3.6 Texture Representations in Current CBIR Systems

The statistical and spectral methods are commonly used representations in existing CBIR systems. In particular, Tamura features are used in QBIC [55] and Photobook [8]. The Wold model is used in PhotoBook. Gabor features are used in Netra. VisualSEEk use texture features that is based on Wavelet transform.

Recently, MPEG group have announced a set of descriptor explained in the next section.

3.6.1 Texture Descriptors in MPEG-7

MPEG-7 is an ISO/IEC standard developed by *Moving Picture Experts Group* (MPEG), which aims to create a standard for describing content of multimedia data [117].

MPEG-7 has introduced three texture descriptors: *homogeneous texture*, *texture browsing*, and *edge histogram* [118], which are summarized below.

3.6.1.1 Homogeneous Texture Descriptor

The homogeneous texture descriptor (HTD) provides a precise quantitative description of a texture that can be used for search and retrieval [118, 117]. In order to obtain an image signature, first the image is filtered with a bank of Gabor filters with five scale and six orientation. The first and second moments of energy in the frequency bands are, then, used as the components of the descriptor. The HTD feature vector is defined as follows:

$$HTD = [f_{dc}, f_{sd}, e_1, e_2, \ldots, e_{30}, d_1, d_2, \ldots, d_{30}] \ , \tag{3.44}$$

where f_{dc} and f_{sd} are the mean intensity and the standard deviation of the image, e_i and d_i, $(1 \leq i \leq 30)$ are the first and second order moments of the energy in the frequency band i, respectively.

The values of the HTD are nonlinearly scaled and quantized into 8-bits. Then, feature values are normalized with the standard deviation for a given database. Texture similarity is measured with the L_1 norm.

3.6.1.2 Texture Browsing

The texture browsing descriptor categorizes texture in terms of regularity (*highly regular, regular, slightly regular, irregular*), coarseness (*fine, medium, coarse, very coarse*), and directionality ($0°$, $30°$, $60°$, $90°$, $120°$, $150°$) similar to a human characterization. It can be used for browsing type of applications, but it may not be appropriate for similarity ranking. One possible scheme for similarity retrieval is the use of texture browsing descriptor to find a set of candidates and

then use homogenous texture descriptor to get a similarity ranking among the candidate images [118, 117].

Feature extraction of this descriptor proceeds similarly as the homogenous texture descriptor. Images are filtered with a bank of 24 Gabor filters (6 orientations, 4 scales). A 12-bit descriptor is computed from the filtered image. The first 2-bit characterize texture's regularity, next 6-bit (3 $bits \times 2$) directionality and the last 4-bit (2 $bits \times 2$) coarseness.

3.6.1.3 Edge Histogram

The edge histogram descriptor represents the spatial distribution of five type of edges, namely, four type of directional edge (*i.e.*, horizontal, 45° diagonal, and 135° diagonal) and a non-directional edge (*i.e.*, isotropic) are considered [118, 117, 119]. It is useful for image-to-image matching, where underlying texture is not homogeneous.

In order to compute the edge histogram, first a given image is subdivided into equal size 16 subimages. Then, each subimage is filtered with 2×2 edge filters (as shown in Figure 3.16) and a histogram with 5-bins are constructed. Bins are nonuniformly quantized using 3 bits/bin. Finally, edge histogram descriptor with size 240 ($16 \times 5 \times 3 = 240$) is constructed. Histograms are matched with the L_1 norm.

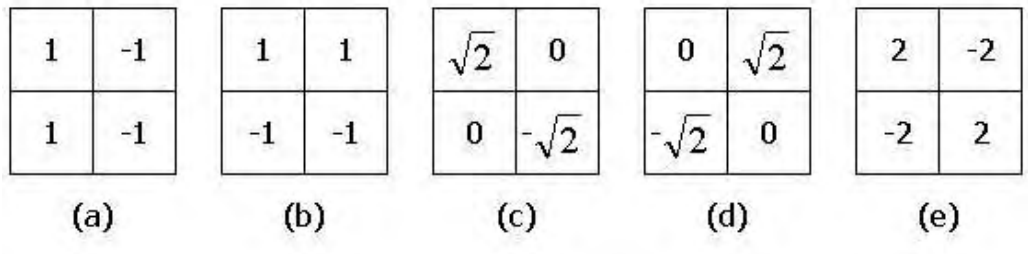


Figure 3.16: (a) Horizontal, (b) vertical, (c) 45° diagonal, (d) 135° diagonal, and (e) Isotropic edge filters.

3.7 Comparisons of Texture Representation

Although various studies about the comparisons of texture representations exist in the literature, none of them analyze the representations in a systematic manner. We cannot find a study, which compare all of the existing texture representations using a wide range of images. Most of the studies concentrate on a few texture representations and also, in general, their experimental image databases contains fewer images. Below, some well-known and widely referred studies are given.

- Weszka et. al. in [88] compare the texture classification performance of first-order statistical features, second-order statistical features and Fourier power spectrum features. They reported that Fourier features have the worst performance, whereas the others are comparable.

- Ohanian and Dubes in [120] compare MRF model, multichannel filtering, fractal-based and co-occurrence features. They concluded that co-occurrence matrix representation performed best in their test set. Although numerous textural features are extracted from co-occurrence matrix, these features are expensive to compute, and they are not efficient for image classification and retrieval [121].

- Castelli in [121] states that experiments with some natural texture databases had shown that the Wold model provides "better quality" retrieval than MR-SAR or the Tamura's features. Tamura features has the same accuracy, but slower than SAR.

- Reed and Buf [122] provides a review of the texture segmentation and feature extraction techniques. The study includes Laws masks, co-occurrence matrices, gray-level dependency matrix, fractal models, stochastic models, Gabor power spectrum and global power spectrum. The authors concluded that all the techniques have distinct application areas.

- Four filtering methods of texture discrimination are compared by Chen and Chen [123]. These methods include Fourier transform, spatial filter, Gabor filter and wavelet transform. They find that wavelet and Gabor features perform equally well. These two perform better than the other two features.

- Ma and Manjunath in [124] compare various wavelet transform features, namely, *orthogonal wavelet transform* (OWT), *bi-orthogonal wavelet transform* (BWT), *tree-structured decomposition using orthogonal filter bank* (TOF), *tree-structure decomposition using bi-orthogonal filter bank* (TBF), and *Gabor wavelet transform* (GWT). They found that Gabor wavelet transform based features was the best among the others. They, also, compared *Gabor filter features*, *pyramid structured wavelet transform features* (PWT), *tree structured wavelet transform features* (TWT), and *multi-resolution simultaneous autoregressive model features* (MR-SAR) in [111] and concluded that Gabor features give the best overall performance and this is closely followed by MR-SAR features.

3.8 Discussions

Practically speaking, texture descriptors represent distinctive characteristics of a texture, which are specific to the problem domain. Unfortunately, none of the existing descriptors has been shown to give satisfactory results over a wide range of textures.

The success of a texture descriptor heavily depends on the data type and the application area. A major problem in representing texture is that the textures in the real world are often quite complex, due to changes in orientation, scale or other visual appearance such as brightness and contrast. Additionally, it is difficult to include extremely large number of attributes of texture under a single mathematical representation.

Statistical descriptors exploit the local correlation of image pixels, whereas spectral descriptors capture global information about the energy on different

scales. While statistical descriptors successfully analyze textures with weak edges or random nature, spectral and structural descriptors are best suited for periodic or almost periodic textures. In a given problem domain, various types of textures may be mixed.

Although at present the most promising texture descriptor is obtained from Gabor filter banks with varying size and orientation, selection of the parameters for Gabor Filter descriptor depends on the characteristics of the textures in the image database. Since the Gabor functions are not orthogonal, there is a trade-off between redundancy and completeness in the design of the Gabor Filter Banks. Otherwise, the implementation of a complete Gabor expansion would entail a generally impractical number of filters. Also, in a digital world, it is not always possible to cope with all sizes of analog Gabor Filters, which may cause problems, especially, with the textures that consist of small texels or sharp corners. Another limitation of the Gabor descriptor is the restriction of the filtering area, which must fit in a rectangle, unless some pre-processing is done.

CHAPTER 4

A GENERIC TEXTURE DESCRIPTOR for IMAGE RETRIEVAL: Statistical Analysis of Structural Information (SASI)

In this chapter, a generic texture descriptor for image retrieval is introduced. The Statistical Analysis of Structural Information (SASI) descriptor is based on second order statistics of clique autocorrelation coefficients, which are the autocorrelation coefficients over a set of moving windows. The clique windows of various size and shape, which are defined by a neighborhood system, are used as a tool for describing the characteristics of textures in different granularity. The order of the neighborhood system controls the structure of the clique windows. Because of the flexibility in the definition of clique windows, SASI can cope with a broad class of textures, which may consist of discontinuities or small primitives.

4.1 Definitions

SASI is based on the concept of clique [125] and autocorrelation coefficient. In the following, SASI descriptor is introduced along with the background defini-

tions [126].

Definition 4.1.1 (Neighboring set of a pixel) . *For a regular lattice $\mathcal{L}$, the neighboring set of a pixel ij with coordinate (i, j) is defined by the following recurrence relation:*

$$\forall kl \in \mathcal{L},\ ij \neq kl\ ,$$

$$\eta_{ij}^{d} = \eta_{ij}^{d-1} \cup \arg \min_{kl \notin \eta_{ij}^{d-1}} D(ij, kl)\ ,$$

and

$$\eta_{ij}^{1} = \arg\min D(ij, kl)\ ,$$

where

$\mathcal{L} = \{(i, j) | i, j \in \mathcal{N}, 1 \leq i \leq Width_{\mathcal{L}}\ and\ 1 \leq i \leq Height_{\mathcal{L}}\}$,

$D(ij, kl)$ denotes the distance function between pixel ij and kl,

d is the order of neighborhood system and $d \in \mathcal{N}$.

The neighboring relationship has the following properties:

1. a pixel is not neighboring to itself: $ij \notin \eta_{ij}^{d}$,
2. the neighboring relationship is commutative: $ij \in \eta_{kl}^{d} \Leftrightarrow kl \in \eta_{ij}^{d}$.

Widely used first and second order neighborhood systems can be seen in Figure 4.1. A more general scheme is shown in Figure 4.2, where the labels from 1 to 5 indicate the order of the neighborhood system with respect to the Euclidean distance.

Pixels near the edge of the lattice have fewer neighbors than the interior pixels. This fact is compensated by assuming that the lattice has a periodic or torus structure (as shown in Figure 4.3), which means that the left edge is connected to the right edge and the upper edge is connected to the lower edge [70, 126, 127, 128, 129].

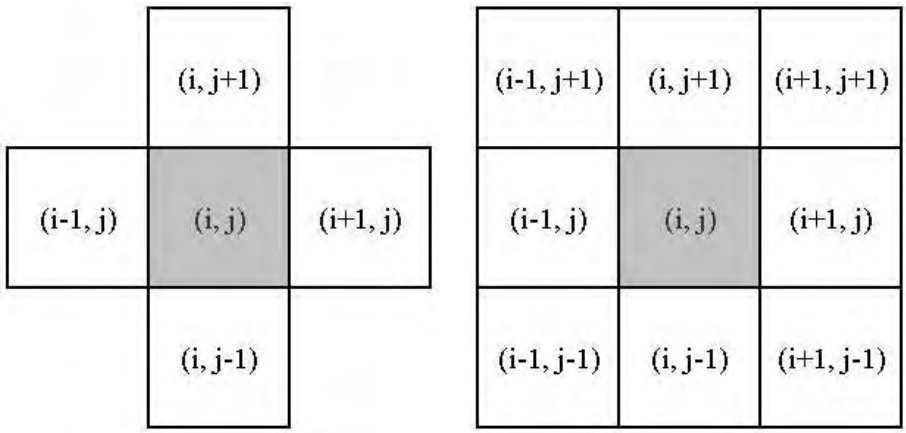


Figure 4.1: Neighbors of pixel ij in the (a) first-order and (b) second-order neighborhood system.

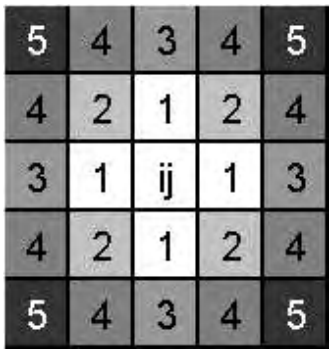


Figure 4.2: Neighbors of pixel ij. The labels $d = 1, \ldots, 5$ indicate the order of neighborhood system.

Figure 4.3: A torus structure.

Definition 4.1.2 (Base clique type) *Given the neighborhood system η^d, base clique type set P is defined as,*

$$P = \left\{ \bar{p} = (k,l) - (i,j) | \forall kl \in \eta_{ij}^d \right\}.$$

Note that $\bar{p}$ is a position vector between two locations (i,j) and (k,l) of a lattice and called base clique type. Figure 4.4 indicates the base clique types for the second order neighborhood system η^2 , where $|P| = 8$.

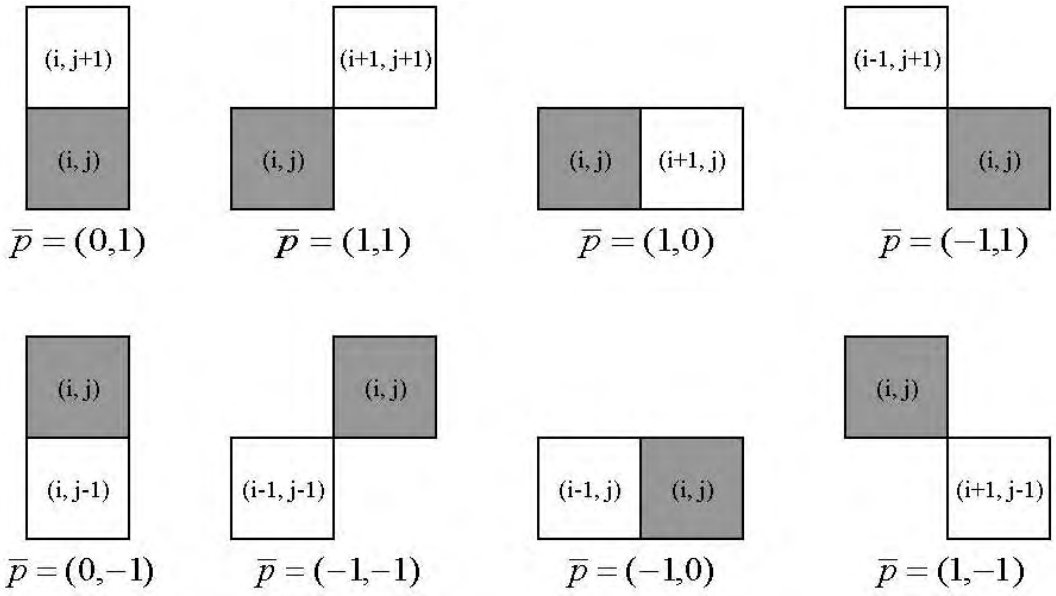


Figure 4.4: Base clique types $\bar{p}$, in η^2 neighborhood. Shaded pixel is taken as a seed pixel.

Definition 4.1.3 (Base clique test predicate) *$B_{\bar{p}}(ij,kl)$ is a Boolean function, which tests the neighboring and relative orientation of pixel ij and kl with respect to each other, given by:*

$$B_{\bar{p}}(ij,kl) = \begin{cases} True & \text{if } (k,l) - (i,j) = \bar{p} \text{ and } kl \in \eta_{ij}^d \\ False & \text{otherwise} \end{cases},$$

where $\bar{p} \in P$.

Unlike neighboring relation, base clique test predicate is not commutative: $B_{\bar{p}}(ij,kl) \neq B_{\bar{p}}(kl,ij)$.

Definition 4.1.4 (Clique chain) *Given pixel ij as a seed, the clique chain $C_L^{\bar{p}}(ij)$ with length L is a set defined by,*

$$\forall \bar{p} \in P, \qquad C_L^{\bar{p}}(ij) = \{ij, kl, mn, \ldots, qr, st, uv \mid$$
$$B_{\bar{p}}(ij, kl) \wedge B_{\bar{p}}(kl, mn) \wedge \ldots \wedge B_{\bar{p}}(qr, st) \wedge B_{\bar{p}}(st, uv)\},$$

where

total number of pixels in $C_L^{\bar{p}}(ij)$ is L,

(ij, kl) and (st, uv) is the first and the last neighboring pair of pixels, with the base clique type $\bar{p}$, respectively.

While in η^2, clique chains are lines of pixels with various directions, for higher order neighborhood systems, they become dash lines of pixels. Since $\eta^2 = 8$, only 8 direction clique chains can be obtained, as shown in Figure 4.5. Note that each $C_L^{\bar{p}}(ij)$ is symmetric to $C_L^{-\bar{p}}(ij)$.

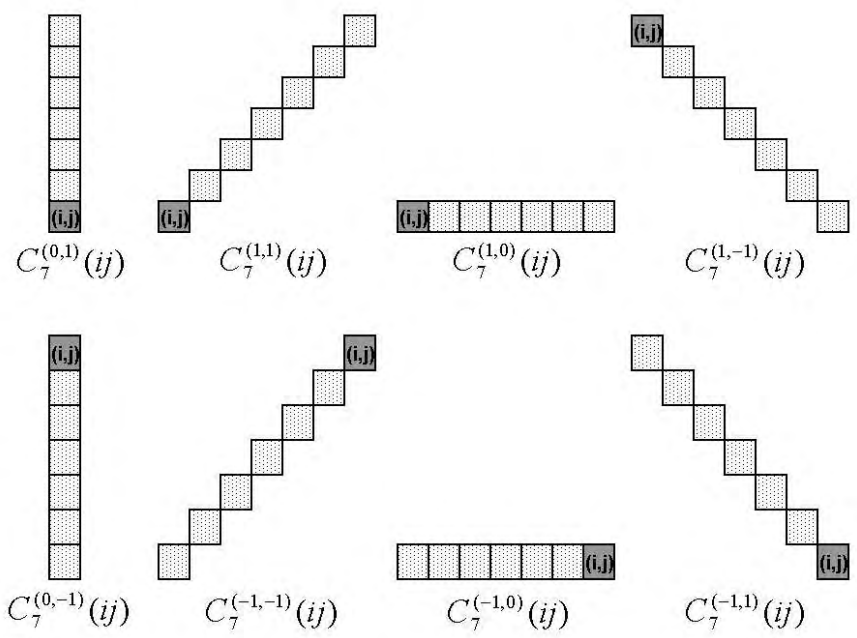


Figure 4.5: Eight orientations of clique chain with length 7.

By definition, a clique chain defined in η^{d_1} can also be defined in η^{d_2} if $d_2 > d_1$. Figure 4.6 shows additional clique chains that can be defined, as the order of the neighborhood system is increased from η^2 to η^3.

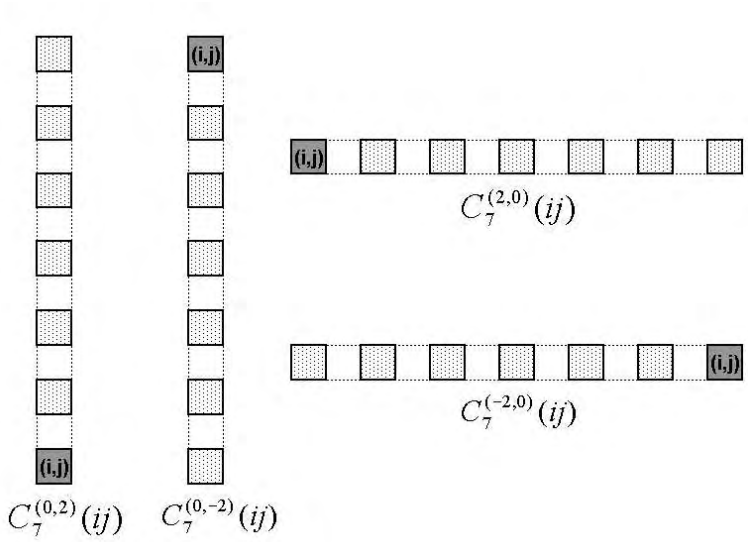


Figure 4.6: Clique chains defined in η^3.

Definition 4.1.5 (Clique Window) *Clique Window* $W_{S,L}^{\bar{p},\langle\bar{c}\rangle}(ab)$ *with seed* ab *is an* $S \times L$ *structuring element, which consists of* S *clique chains, defined as*

$$W_{S,L}^{\bar{p},\langle\bar{c}\rangle}(ab) = \Big\{ C_L^{\bar{p}}(ab) \cup C_L^{\bar{p}}(cd) \cup C_L^{\bar{p}}(ef) \cup \ldots \cup C_L^{\bar{p}}(wx) \cup C_L^{\bar{p}}(yz) \mid B_{\bar{c}_1}(ab, cd) \wedge B_{\bar{c}_2}(cd, ef) \wedge \ldots \wedge B_{\bar{c}_{S-1}}(wx, yz) \Big\},$$

where $ab,\ cd,\ ef,\ \ldots\ ,\ wx,\ yz \in \mathcal{L}$, $\langle\bar{c}\rangle$ is an ordered $S-1$ tuple of base clique types such that $\langle\bar{c}\rangle = \rangle \bar{c}_1, \bar{c}_2, \ldots, \bar{c}_{S-1}\rangle \mid \bar{c}_i \in P$, for $i = 1 \ldots S-1$. Each element of $\langle\bar{c}\rangle$, denoted as $\bar{c}_i$, represents the base clique type of the ith and $(i+1)$th clique chain pair. It specifies how the clique chains, $C_L^{\bar{p}}$, are connected to each other. Parameters $\bar{p}$, $\langle\bar{c}\rangle$, S, and L determine the structure of the clique window.

The clique window is called *regular* if

- $\bar{c}_i = \bar{c}_j$ for all $i, j = 1 \ldots S-1$, and
- $\frac{\bar{c}_i}{|\bar{c}_i|} \neq \frac{\bar{p}_i}{|\bar{p}_i|}$ and $\frac{\bar{c}_i}{|\bar{c}_i|} \neq -\frac{\bar{p}_i}{|\bar{p}_i|}$.

Otherwise it is *irregular*.

Figure 4.7 illustrates some of the clique windows defined in η^2. In contrast to Figure 4.7(a)–(e), which are regular clique windows, Figure 4.7(f)–(i) are the

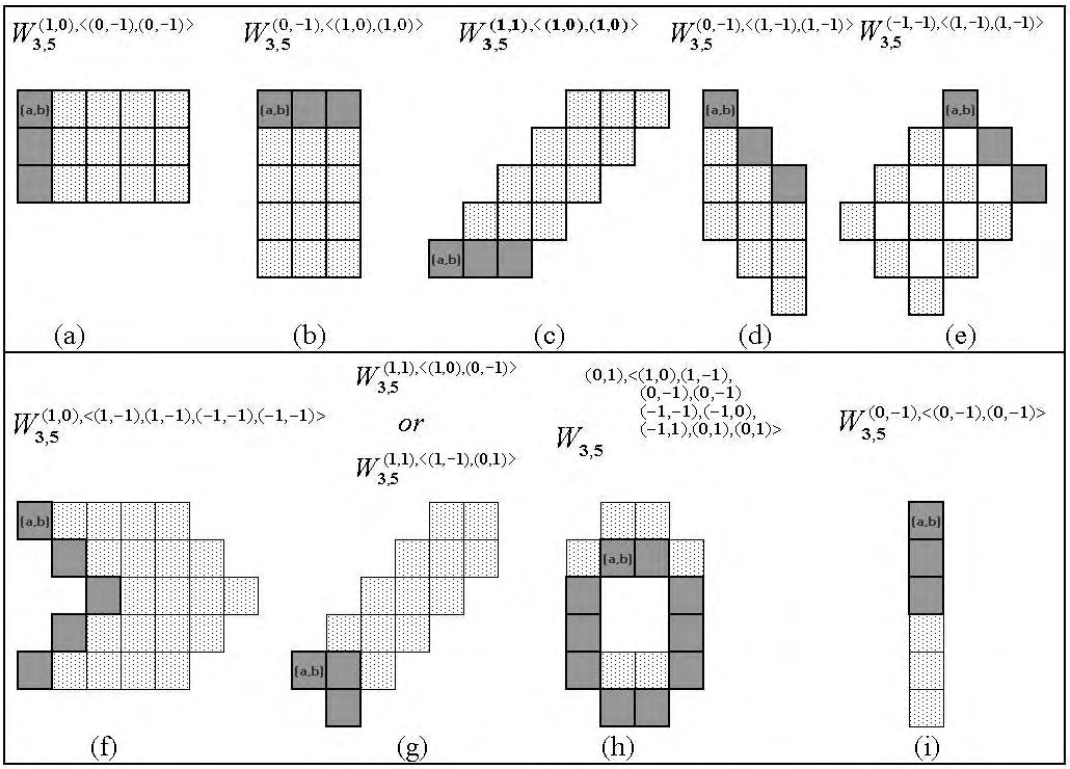


Figure 4.7: Some of the regular and irregular clique windows defined in η^2. (a)–(e) are regular, but (f)–(i) are irregular.

examples of irregular clique windows. For a regular clique window $W_{S,L}^{\bar{p},\langle\bar{c}\rangle}(ab)$, the parameters S, L and $\bar{p}$, $\langle\bar{c}\rangle$ determine the size and orientation of the clique window, respectively. On the other hand, most of the time, it is hard to talk about the size and orientation of the irregular clique windows. This fact is depicted in Figure 4.7(h). Thus, for the irregular clique windows, rather than the size and orientation, the structure becomes the main issue.

In this study, we mainly concentrate on the regular clique windows. For the sake of simplicity, $W_{S,L}^{\bar{p},\langle\bar{c}\rangle}(ab)$ is abbreviated as $W_{S,L}^{\bar{p},\bar{c}}(ab)$, since in regular clique windows, all $\bar{c}_i$s are equal to each other. Also due to the symmetric relations a regular clique window $W_{S,L}^{\bar{p},\bar{c}}(ab)$ has the same structure as $W_{S,L}^{-\bar{p},\bar{c}}(ab)$,$W_{S,L}^{\bar{p},-\bar{c}}(ab)$, and $W_{S,L}^{-\bar{p},-\bar{c}}(ab)$ as shown in Figure 4.8.

In η^2, 12 different clique windows (ignoring the symmetric ones) can be defined as shown in Figure 4.9. One can employ the higher order neighborhood

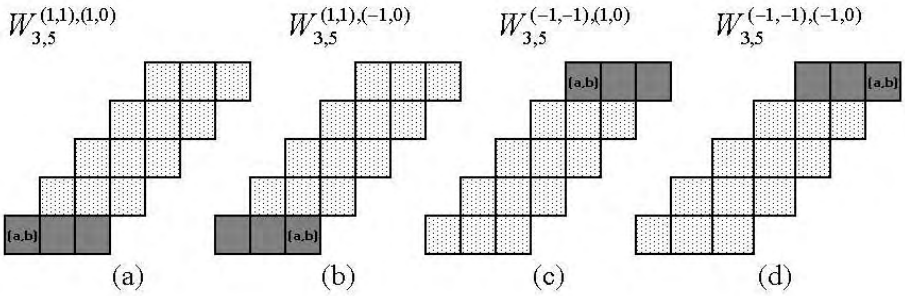


Figure 4.8: Representation of some regular clique windows.

systems, in order to incorporate the characteristics of the images in the database. Then, the clique windows can be defined based on the clique type set for a given neighborhood system. For example, 26, 86 and 124 regular clique windows can be defined in η^3, η^4, and η^5 respectively. Note that a clique window defined in η^{d_1} can also be defined in η^{d_2} if $d_2 \geq d_1$.

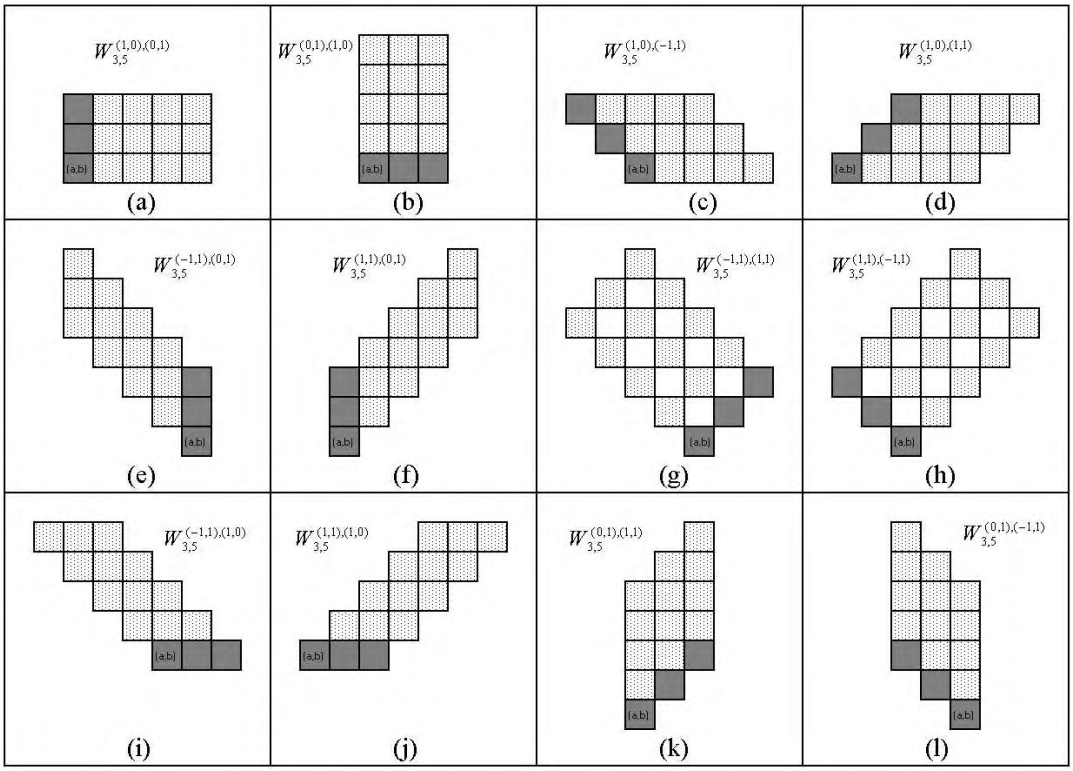


Figure 4.9: Regular clique windows defined in η^2.

In this dissertation, only the regular clique windows are used for measuring the texture similarity. For practical reasons, the definitions below are given for regular clique windows. The generalization to irregular cases requires some extra work.

Definition 4.1.6 (Clique autocorrelation coefficient) *Clique autocorrelation coefficient at lag vector* $\bar{\mathrm{v}} = (\mathrm{v}_x, \mathrm{v}_y)$ *of a given seed pixel* ab *for a regular clique window* $W_{S,L}^{\bar{p},\bar{c}}(ab)$ *is given by*

$$r(\bar{\mathrm{v}})^{W_{S,L}^{\bar{p},\bar{c}}(ab)} = \frac{\sum_{\forall(i,j)\wedge(i+\mathrm{v}_x,j+\mathrm{v}_y)\in W_{S,L}^{\bar{p},\bar{c}}(ab)} (x_{i,j}-\bar{x}_{i,j})(x_{i+\mathrm{v}_x,j+\mathrm{v}_y}-\bar{x}_{i+\mathrm{v}_x,j+\mathrm{v}_y})}{\sqrt{\sum_{\forall(i,j)\in W_{S,L}^{\bar{p},\bar{c}}(ab)} (x_{i,j}-\bar{x}_{i,j})^2 \sum_{\forall(i+\mathrm{v}_x,j+\mathrm{v}_y)\in W_{S,L}^{\bar{p},\bar{c}}(ab)} (x_{i+\mathrm{v}_x,j+\mathrm{v}_y}-\bar{x}_{i+\mathrm{v}_x,j+\mathrm{v}_y})^2}} , \quad (4.1)$$

where

$x_{i,j}$ is the gray value of the image at position (i,j),

$\bar{x}_{i,j} = \frac{1}{N_{W_{S,L}^{\bar{p},\bar{c}}(ab)}} \sum_{\forall(i,j)\in W_{S,L}^{\bar{p},\bar{c}}(ab)} x_{i,j}$ is the mean value of the gray levels,

$N_{W_{S,L}^{\bar{p},\bar{c}}(ab)}$ is the number of pixels in the clique window $W_{S,L}^{\bar{p},\bar{c}}(ab)$.

Lag vector $\bar{\mathrm{v}}$ is a vector between two locations of a clique window. Note that, autocorrelation coefficients of a clique window depend only on the length and direction of the lag vector.

Clique autocorrelation coefficients can be considered as a *short-term correlogram* over the clique window defined by the clique chain. They enable us to capture stationary information at various scale and orientation within an image.

Since the autocorrelation coefficients at all lags bear redundant information, as experimented later in Section 5.1, there is no need to use all of them for the representation of a texture in a multidimensional space. Therefore, it is reasonable to take the lag vector $\bar{\mathrm{v}}$ of a clique window $W_{S,L}^{\bar{p},\bar{c}}(ab)$ as,

$$\bar{\mathrm{v}} = n \times \bar{p} = (n\ p_x, n\ p_y) ,$$

where n is a lag multiplier and $n \in \mathcal{N}, 1 \leq n \leq L-1$. In other words, the lag vector $\bar{\mathrm{v}}$ is taken as the same as the base clique type of the clique chains, which make the clique window.

Definition 4.1.7 (Second order statistics of autocorrelation coefficients) *Mean value and standard deviation of clique autocorrelation coefficients with lag vector* $\bar{\mathrm{v}}$ *of all clique windows* $W_{S,L}^{\bar{p},\bar{c}}(ab)$ *is defined as*

$$\mu_{S,L}^{\bar{p},\bar{c}}(\bar{\mathrm{v}}) = \frac{1}{N_{\mathcal{L}}} \sum_{\forall (a,b) \in \mathcal{L}} r(\bar{\mathrm{v}})^{W_{S,L}^{\bar{p},\bar{c}}(ab)} \ , \tag{4.2}$$

and

$$\sigma_{S,L}^{\bar{p},\bar{c}}(\bar{\mathrm{v}}) = \sqrt{\frac{1}{N_{\mathcal{L}}} \sum_{\forall (a,b) \in \mathcal{L}} \left(r(\bar{\mathrm{v}})^{W_{S,L}^{\bar{p},\bar{c}}(ab)} - \mu_{S,L}^{\bar{p},\bar{c}}(\bar{\mathrm{v}}) \right)^2} \ , \tag{4.3}$$

respectively, where

$\bar{\mathrm{v}}$ is the lag vector,

$\bar{p}$ and $\bar{c}$ are the base clique types,

S is the number of clique chain,

L is the clique chain length,

$N_{\mathcal{L}}$ is the number of pixels in the lattice.

Definition 4.1.8 (SASI descriptor) *For a given texture* T, SASI descriptor *is defined as an* $N \times 1$ *vector with the entries* $\mu_{S,L}^{\bar{p},\bar{c}}(\bar{\mathrm{v}})$,$\sigma_{S,L}^{\bar{p},\bar{c}}(\bar{\mathrm{v}})$ *as*

$$D_T = \Big\{ \mu_{S_1,L_1}^{\bar{p}_1,\bar{c}_1}(\bar{\mathrm{v}}_1), \mu_{S_2,L_2}^{\bar{p}_2,\bar{c}_2}(\bar{\mathrm{v}}_2), \ldots, \mu_{S_Q,L_Q}^{\bar{p}_Q,\bar{c}_Q}(\bar{\mathrm{v}}_Q),$$

$$\sigma_{S_1,L_1}^{\bar{p}_1,\bar{c}_1}(\bar{\mathrm{v}}_1), \sigma_{S_2,L_2}^{\bar{p}_2,\bar{c}_2}(\bar{\mathrm{v}}_2), \ldots, \sigma_{S_Q,L_Q}^{\bar{p}_Q,\bar{c}_Q}(\bar{\mathrm{v}}_Q) \Big\} \ , \tag{4.4}$$

where $2 \times Q$ (Q mean values + Q standard deviations) is the size of the feature vector.

For each selected clique window $W_{S_i,L_i}^{\bar{p}_i,\bar{c}_i}(ab)$ (where $i = 1 \ldots$ total number of clique windows selected), total number of mean value and standard deviations calculable are $2 \times (L_i - 1)$, since lag vector $\bar{\mathrm{v}}$ is defined as $\bar{\mathrm{v}} = n \ \times \ \bar{p}_i$

where $n \in \mathcal{N}, 1 \leq n \leq L_i - 1$. Therefore, the maximum value of Q is $\sum_{i=1}^{total\_number\_of\_clique\_windows\_selected}(L_i - 1)$.

Definition 4.1.9 (Normalized SASI descriptor)

Given, $D_T = [f_1, \ldots, f_{2\times Q}]$*, then* normalized SASI descriptor*,* $D'_T = [f'_1, \ldots, f'_{2\times Q}]$*, is defined by normalizing the entries of* D_T *as follows:*

$$f'_i = \frac{f_i - \mu_{f_i}}{\sigma_{f_i}}, \qquad i = 1, 2, \ldots, Q \ , \tag{4.5}$$

where μ_{f_i} is the mean value and σ_{f_i} is the standard deviation of the features over the entire database.

D'_T measures the structural information by using the second order statistics of local autocorrelation coefficients for texture T. The size of the descriptor D'_T depends on the image database. We use the distance metric defined below in order to measure the mathematical similarity between the textures.

Definition 4.1.10 (SASI Distance) *The mathematical similarity between the textures* T_1 *and* T_2 *is measured by the following metric:*

$$S\big(D'_{T_1}, D'_{T_2}\big) = \frac{D'_{T_1} \odot D'_{T_2}}{D'_{T_1} \odot D'_{T_1} + D'_{T_2} \odot D'_{T_2} - D'_{T_1} \odot D'_{T_2}} \ , \tag{4.6}$$

where $\odot$ stands for dot product. This distance measure is also known as *similarity rule* as defined in Section 2.1.4.2. Although many other distance functions can be used, in our experiments done in Section 5.2.1.1, best results are obtained with the *similarity rule*.

4.2 Algorithm of SASI

The pseudo-code of SASI is shown in Algorithm 1. The most crucial part of the algorithm is the selection of the clique window sizes, S and L. A preliminary analysis on database, as discussed in the next section, may help us to determine them. Window sizes depend on the size of the texture primitives and resolution of the images in the database. Basically, clique windows should be small enough to capture small primitives and big enough to capture large patterns or primitives in the images of the database. One can employ all possible

Algorithm 1 Algorithm of SASI

Begin

Select neighborhood system η^d

Select the clique windows $W_{S,L}^{\bar{p},\bar{c}}$ as a subset of all clique windows

Select the lag vectors used for each clique window

$D_T = \phi$

For each clique window $W_{S,L}^{\bar{p},\bar{c}}$

For each lag vector $\bar{\mathrm{v}}$

For each pixel (ab)

Define clique window $W_{S,L}^{\bar{p},\bar{c}}(ab)$

Calculate $r(\bar{\mathrm{v}})^{W_{S,L}^{\bar{p},\bar{c}}(ab)}$ using Equation 4.1.

Next pixel (ab)

Calculate mean value $\mu_{S,L}^{\bar{p},\bar{c}}(\bar{\mathrm{v}})$ using Equation 4.2

Calculate standard deviation $\sigma_{S,L}^{\bar{p},\bar{c}}(\bar{\mathrm{v}})$ using Equation 4.3

$D_T = D_T \cup \{\mu_{S,L}^{\bar{p},\bar{c}}(\bar{\mathrm{v}}), \sigma_{S,L}^{\bar{p},\bar{c}}(\bar{\mathrm{v}})\}$

Next lag vector $\bar{\mathrm{v}}$

Next clique window $W_{S,L}^{\bar{p},\bar{c}}$

Construct normalized D_T' vector using Equation 4.5

end.

size clique windows and related autocorrelation coefficients, but this time, the computational power is wasted. Additionally, increasing the dimension of the feature vector may not improve the representation capability of the descriptor. This is a well-known phenomenon, called *curse of dimensionality*, in pattern recognition.

4.3 Summary

In this chapter, a generic texture descriptor, namely, Statistical Analysis of Structural Information (SASI) is introduced as a representation of texture. SASI is based on statistics of clique autocorrelation coefficients, calculated over structuring windows. SASI defines a set of clique windows to extract and measure various structural properties of texture by using a spatial multi-resolution method.

CHAPTER 5

PERFORMANCE EVALUATIONS of SASI

Two sets of experiments are done to show the power of SASI. First, SASI descriptor is analyzed in detail and various properties of SASI is compared to Gabor Filter descriptor. Latter, SASI and Gabor Filter descriptor are tested on the image retrieval problem by using four different image databases, namely Brodatz Album [130], CUReT [131], PhoTex [132], and VisTex [133]. The experiments are, also, performed on a database generated by joining all the images of these four databases.

Brodatz Album contains 112 pictures with size 512×512 and 256 gray values after digitizing, showing a variety of textures, collected for artistic purposes [130]. It is a de facto standard set of images for texture retrieval problem. Images can be seen in Appendix A, as thumbnails. Due to its popularity and comparable studies exist in the literature [134, 111, 93]; a comparative analysis is provided on the Brodatz Album in the following sections.

Columbia-Utrecht Reflectance and Texture Database (CUReT) are formed by the researchers at Colombia University and Utrecht University [131]. It contains 61 different pictures with various size and color. Thumbnails of the images can be seen at http://www.cs.columbia.edu/CAVE/curet/html/sample.htm. Before applying SASI and Gabor descriptors, each image is rescaled to 512×512 and converted to gray scale. Final version of the images are depicted

in Appendix B.

Jerry Wu from Heriot-Watt University at Edinburgh creates Photometric Texture Database (PhoTex) [132]. 30 different pictures with size 512×512 and 256 gray values exist in PhoTex. Images can be seen at Appendix C and further information is available at http://www.cee.hw.ac.uk/texturelab/database/jwdb-/thumbnails.htm.

Vision Texture Database (VisTex) is formed by the Vision and Modeling group at the MIT Media Lab [133]. It contains 167 colored reference textured images with size 512×512. Images are grouped according to their contents. Thumbnails of the images are shown in Appendix D. Further details can be found at http://www-white.media.mit.edu/vismod/imagery/VisionTexture-/vistex.html.

5.1 SASI in Detail

In this section, first, traditional correlogram method is examined to show the redundancy in autocorrelation coefficients in the analysis of texture. Next, the clique windows are employed in order to show the effect of the window sizes in constructing the SASI descriptor. Then, SASI descriptor is compared to Gabor Filter descriptor.

In order to analyze SASI in detail, five texture images are selected from Brodatz Album, namely D001, D035, D052, and D004 as shown in Figure 5.1. D001 contains sharp edges with large texels and high contrast. On the other hand D035, D052, and D004 have coarse to fine textures with relatively low contrasts.

5.1.1 Traditional Correlogram Analysis

Traditional correlogram is a special case of SASI, where the size and the shape of the clique windows are chosen as the size and the shape of the image itself and clique autocorrelation coefficient is calculated for all lag vectors. The resulting series is called the *autocorrelation series* or *correlogram*.

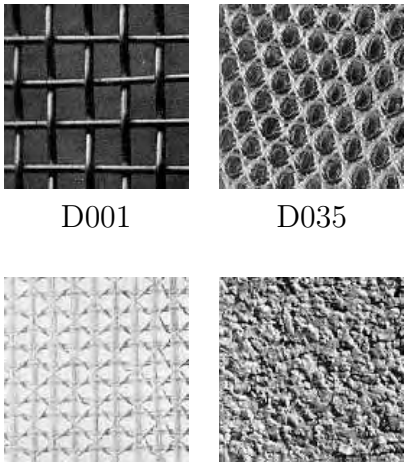


Figure 5.1: Sample texture images from Brodatz Album.

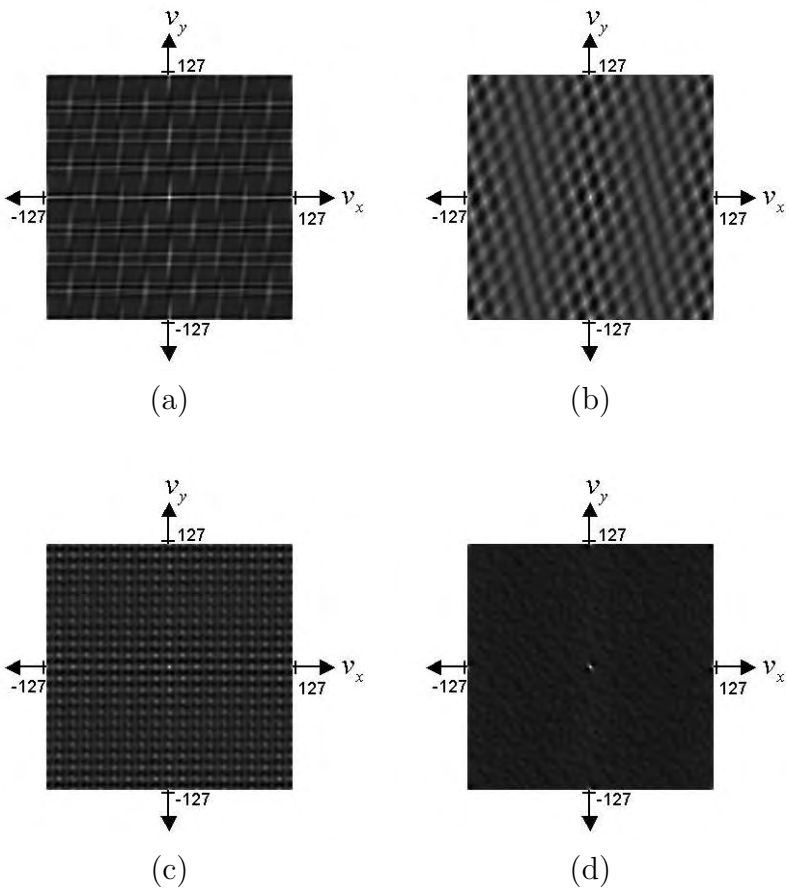


Figure 5.2: The correlograms of texture (a) D001, (b) D035, (c) D052, and (d) D004 as 2-d intensity diagrams.

In the literature, correlogram is used to measure the coarseness of a texture [39, 135]. Basically, correlogram evaluates the linear spatial relationship between the texture primitives that made up a texture.

Figure 5.2(a)–(d) shows the correlograms of texture D001, D035, D052, and D004 from Brodatz Album, assuming that images are in torus structure, as shown in Figure 4.3. In order to depict real valued correlograms, as shown in Figure 5.2, real values are linearly mapped, between 0 to 255 gray values.

If the texture primitives are large as in D035, the autocorrelation function decreases slowly with the increasing lag distance as depicted in Figure 5.2(b). On the other hand, if the primitives are small, the autocorrelation function decreases rapidly as shown in Figure 5.2(d). Moreover, if the primitives are periodically occurred in the image, then correlogram is also periodic. For example, the correlogram of D001 in Figure 5.2(a) shows that approximately at each 30 pixel in X and Y direction texture primitives are repeated.

5.1.1.1 Redundancy in Correlogram

It is well known that correlogram bears redundant information [39, 136, 137]. This redundancy is partially observed by using Principal Component Analysis. Principal Component Analysis is a mathematical procedure that transforms a number of correlated variables into a (smaller) number of uncorrelated variables called principal components, each of which is a particular linear combination of the original variables.

When the rows of the correlogram, denoted as $r(i, L)$, where i is the row number and $L = \{-127, \ldots, 127\}$, are chosen as variables as in Table 5.1, it can be seen that few new variables or principal components are sufficient to capture the information provided by the existing variables.

This fact is exemplified in the sample Brodatz textures, D001, D035, D052, and D004 in Table 5.2, 5.3, 5.4, and 5.5, respectively. The principal components are extracted in decreasing order of importance. The *eigenvalues* measure the amount of the variation explained by each PC and reach the largest value for the

Table 5.1: Principal component analysis of a correlogram.

Variable #	Definition
Variable 1:	$r(-127, L)$,
Variable 2:	$r(-126, L)$,
...	...
Variable 254:	$r(126, L)$,
Variable 255:	$r(127, L)$.
where $L = \{-127, \ldots, 127\}$.	

first PC and smaller for the subsequent ones. The other two measures, namely, *proportion* and *cumulative*, are computed from eigenvalues. Proportion is the normalized version of eigenvalues and is a measures of the importance of a PC.

Analyzing the Table 5.2, 5.3, and 5.4, we found that only 5 principal components for texture D001, D035, and D052 are sufficient to capture almost all the information provided by 255 variables. That means, cumulative importance at principal component 5 is higher than 95%. Event though, D004 is a random-like texture, 25 principal components can reach over 90% cumulative. In other words, 25 principal components capture most of the information provided by the whole variables.

Table 5.2: Principal component analysis of the correlogram of texture D001.

	Eigenvalue	Proportion	Cumulative
Principal component 1	197.574	0.775	0.775
Principal component 2	30.713	0.120	0.895
Principal component 3	12.028	0.047	0.942
Principal component 4	7.029	0.028	0.970
Principal component 5	5.663	0.022	0.992
Principal component 6	0.499	0.002	0.994
Principal component 7	0.329	0.001	0.995
Principal component 8	0.319	0.001	0.997
...	...	...	...
Principal component 255	0	1	1

Table 5.3: Principal component analysis of the correlogram of texture D035.

	Eigenvalue	Proportion	Cumulative
Principal component 1	89.536	0.351	0.351
Principal component 2	87.242	0.342	0.693
Principal component 3	35.998	0.141	0.834
Principal component 4	34.981	0.137	0.972
Principal component 5	0.861	0.003	0.975
Principal component 6	0.799	0.003	0.978
Principal component 7	0.783	0.003	0.981
Principal component 8	0.757	0.003	0.984
...	...	...	...
Principal component 255	0	1	1

Table 5.4: Principal component analysis of the correlogram of texture D052.

	Eigenvalue	Proportion	Cumulative
Principal component 1	229.865	0.901	0.901
Principal component 2	6.183	0.024	0.926
Principal component 3	5.703	0.022	0.948
Principal component 4	4.905	0.019	0.967
Principal component 5	1.438	0.006	0.973
Principal component 6	1.297	0.005	0.978
Principal component 7	1.071	0.004	0.982
Principal component 8	0.847	0.003	0.986
...	...	...	...
Principal component 255	0	1	1

A similar analysis indicates that there is no need to calculate all lags of the clique autocorrelation coefficient for determining the SASI descriptor. Thus, as mentioned in Definition 4.1.6, lag vector $\bar{\mathrm{v}}$ of a clique autocorrelation $r(\bar{\mathrm{v}})^{W_{S,L}^{\bar{p},\bar{c}}(ab)}$ is taken as, $\bar{\mathrm{v}} = n \times \bar{p}$, where n is a lag multiplier.

In the next section, the effect of window size parameters S and L on texture representation will be explored in detail.

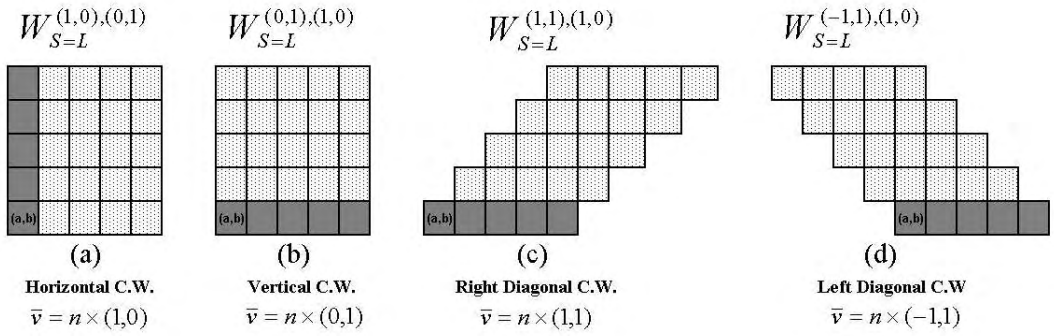
Table 5.5: Principal component analysis of the correlogram of texture D004.

	Eigenvalue	Proportion	Cumulative
Principal component 1	24.302	0.095	0.095
Principal component 2	21.274	0.083	0.179
Principal component 3	18.749	0.074	0.252
Principal component 4	18.073	0.071	0.323
Principal component 5	16.355	0.064	0.387
Principal component 6	13.538	0.053	0.440
Principal component 7	12.212	0.048	0.488
Principal component 8	10.784	0.042	0.531
Principal component 9	9.230	0.036	0.567
Principal component 10	9.191	0.036	0.603
Principal component 11	8.077	0.032	0.634
Principal component 12	7.697	0.030	0.665
Principal component 13	7.448	0.029	0.694
Principal component 14	6.741	0.026	0.720
Principal component 15	6.395	0.025	0.745
Principal component 16	5.320	0.021	0.766
Principal component 17	5.076	0.020	0.786
Principal component 18	4.894	0.019	0.805
Principal component 19	4.714	0.019	0.824
Principal component 20	4.333	0.017	0.841
Principal component 21	3.891	0.015	0.856
Principal component 22	3.410	0.013	0.869
Principal component 23	3.374	0.013	0.883
Principal component 24	2.657	0.010	0.893
Principal component 25	2.439	0.010	0.903
...	...	...	...
Principal component 255	0	1	1

5.1.2 Selection of Clique Window Size in SASI Descriptor

In order to analyze clique window size, four regular clique windows, namely $W_{S=L}^{(1,0),(0,1)}$,$W_{S=L}^{(0,1),(1,0)}$, $W_{S=L}^{(1,1),(1,0)}$, and $W_{S=L}^{(-1,1),(1,0)}$ are defined, as shown in Figure 5.3, which are horizontal, vertical, right and left diagonal clique windows, respectively. For each clique window, possible lag vectors for the clique autocorrelation coefficients are, also, shown.

For the notational simplicity, horizontal, vertical, right diagonal, left diagonal clique windows will be represented as W_S^H, W_S^V, W_S^{RD}, and W_S^{LD}. Working with these clique windows may help us to analyze the effects of the clique window



Lag vector $\bar{\mathrm{v}} = n \times \bar{p}$, where $n \in \mathcal{N}$ and $1 \leq n \leq L-1$

Figure 5.3: Horizontal, vertical, right, and left diagonal clique windows.

sizes and the clique autocorrelation lags on SASI.

Different properties or components of the texture are captured by the clique autocorrelation coefficients at different lag vectors applied on the clique windows. Figure 5.4(a)–(d) illustrates the relation between the clique window size versus mean values and standard deviations of autocorrelation coefficients of textures D001, D035, D052, and D004 in the Brodatz Album, respectively. Note that for texture D001 the mean values and standard deviations of the autocorrelation coefficients remain almost the same for larger values of clique window than the size 25×25. Therefore, for this particular example, it is shown that using clique window size larger than 25×25 does not bring any critical information. Similar analysis is done for the texture D035, D052 and D004 shown in Figure 5.4(b),(c), and (d). In this case, the largest window sizes might be 17×17, 15×15, 19×19.

In Figure 5.4(a)–(d), the clique autocorrelation coefficients are calculated for 4 orientations of clique windows with the lag multiplier $n = 1$. Whereas in Figure 5.5(a)–(d), directions of lag vectors are fixed, but varying lag multiplier n is employed for textures D001, D035, D052, and D004. Due to the dominant horizontal and vertical effects in texture D001 and dominant diagonal effects in texture D035 and D052, related clique windows are selected to examine the dominant features of both textures. It is hard to talk about dominant effect for texture D004. For this particular texture, a clique window would give result in

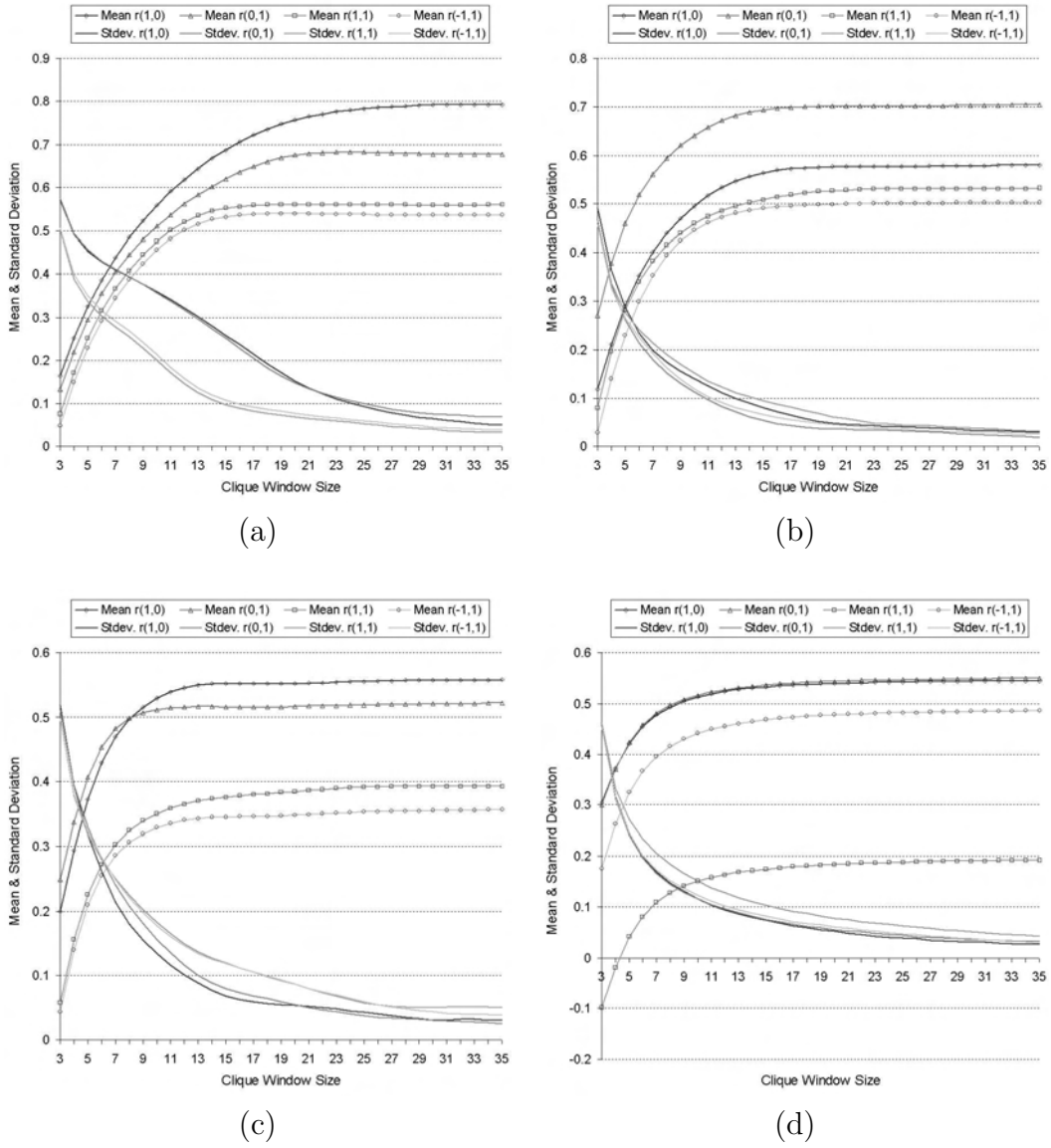


Figure 5.4: Window sizes vs. mean values and standard deviations for texture (a) D001, (b) D035, (c) D052, and (d) D004.

the effect of that particular direction. To illustrate, for example, the horizontal effect, W_S^H is applied on D004. Figure 5.5(a)–(d) indicate that using clique window size larger than 33×33 for texture D001, 25×25 for texture D035, 19×19 for texture D052, and 21×21 for texture D004 brings very small information.

Note that, the mean value of the clique autocorrelation coefficients approaches to the autocorrelation coefficient of the entire texture, as the size of the clique window gets larger.

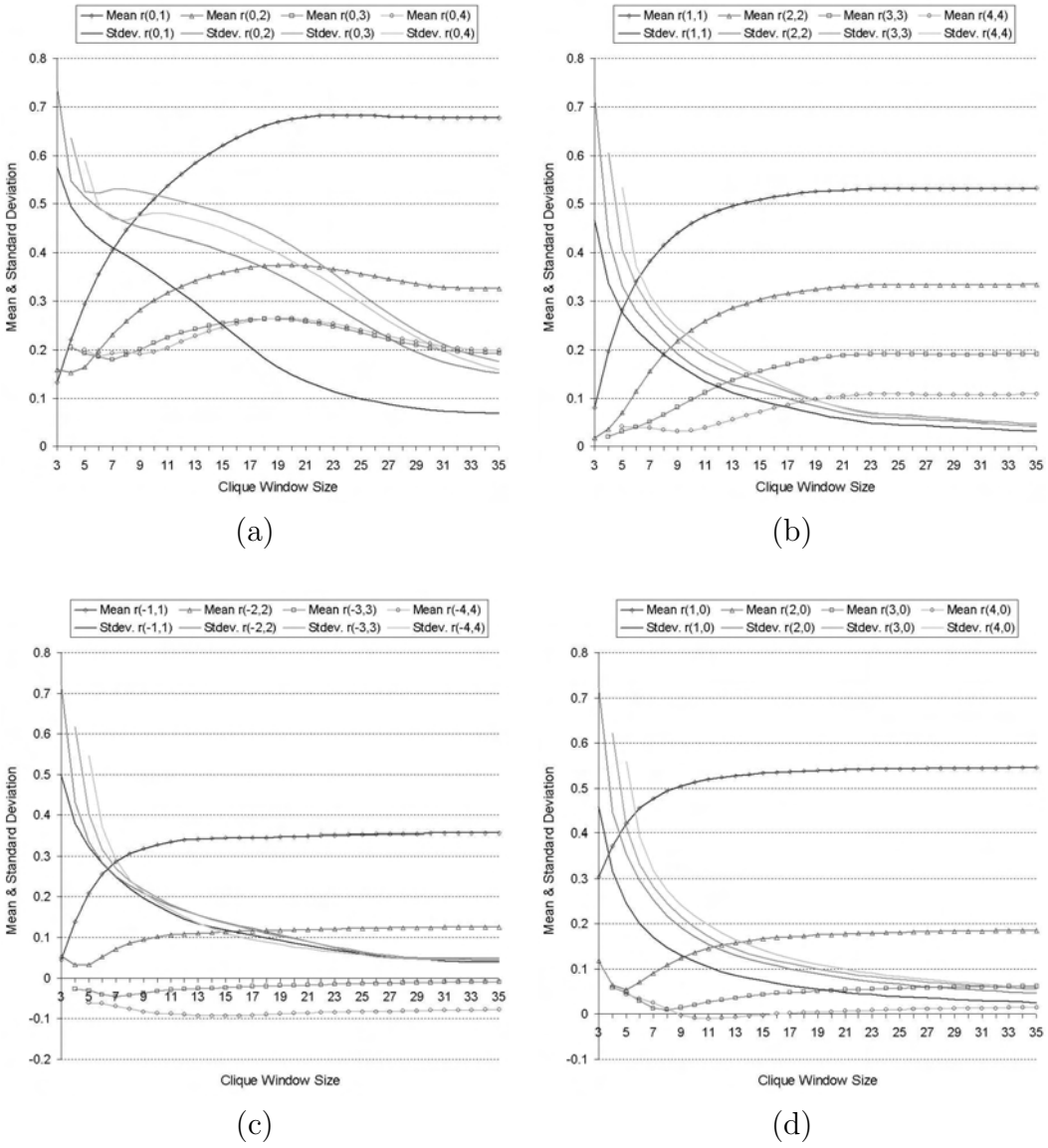


Figure 5.5: Window sizes vs. mean values and standard deviations for texture (a) D001, (b) D035, (c) D052, and (d) D004 using vertical, right diagonal, left diagonal, and horizontal clique windows, respectively.

The window sizes and the lag vectors of the autocorrelation coefficients for each clique window are the critical parameters of SASI. Therefore, a preliminary analysis on the images of the database is required to select these parameters before the calculation of SASI descriptor. In our experiments, we find them by trial and error.

5.1.3 Gabor versus SASI descriptor

Gabor Filter descriptor, reported by Manjunath and Ma [111, 93] use a dictionary, which contains four scales and six orientations. Each filter captures the relevant texture primitives of the image. Second order statistics of the Gabor Filter (4 scales $\times$ 6 orientation = 24 filter) responses of a given texture is used as a texture descriptor. Thus, an image is represented by a vector $\bar{f}$ of size 48 [93].

$$\bar{f} = \left[\mu_{00}, \sigma_{00}, \mu_{01}, \sigma_{01}, ..., \mu_{(S-1),(K-1)}, \sigma_{(S-1),(K-1)}\right] \ ,$$

where the subscript S represents the scale ($S = 0, \ldots, 3$) and K represents the orientation ($K = 0, \ldots, 5$). The distance between two images, namely T_1 and T_2, is defined as

$$d(T_1, T_2) = \sum_S \sum_K \left| \frac{\mu_{SK}^{T_1} - \mu_{SK}^{T_2}}{\alpha(\mu_{SK})} \right| + \left| \frac{\sigma_{SK}^{T_1} - \sigma_{SK}^{T_2}}{\alpha(\sigma_{SK})} \right| ,$$

where $\alpha(\mu_{SK})$ and $\alpha(\sigma_{SK})$ are the standard deviations of respective features over the entire database.

The algorithmic complexity of Gabor Filter descriptor is $O(N \times \log_2 N)$ when filtering is implemented in frequency domain, whereas the complexity of SASI is $O(S \times L \times N)$ where $S \times L$ is a clique window size and N is the image size and $N = Width \times Height$ of the image. If the size of the clique window is small compared to the image size ($S \times L < \log_2 N$) than SASI descriptor requires less computational power than that of Gabor Filter. However in our experiments, we use clique windows of size 3×3, 5×5, and 7×7 for 128×128 images. Therefore, for the databases used in this study, Gabor Filter is less expensive in terms of the computational complexity.

As in Gabor Filter, SASI captures the components of texture with different coarseness. As a result, coarse-to-fine components of the textures are represented in large-to-small size clique windows. This fact is depicted in Figures 5.6, 5.7, 5.8, and 5.9, where various sizes of clique windows decompose the image into various granularities.

D001

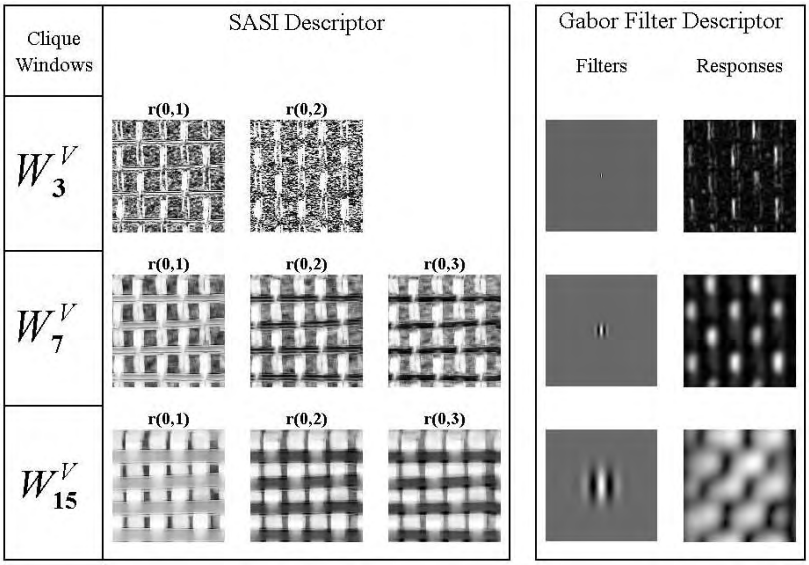
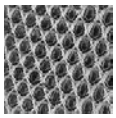


Figure 5.6: SASI and Gabor Filters vertical analysis of texture D001 from Brodatz Album.



D035

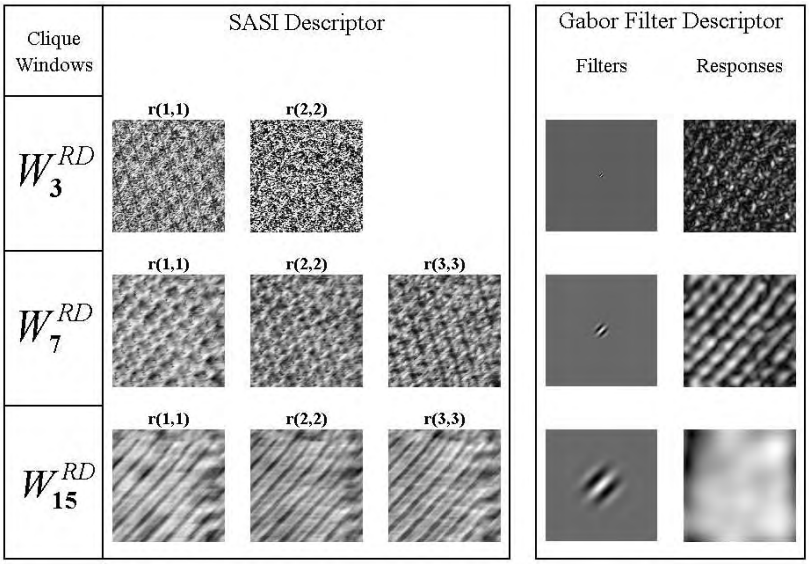
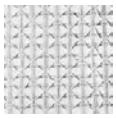


Figure 5.7: SASI and Gabor Filters right diagonal analysis of texture D035 from Brodatz Album.



D052

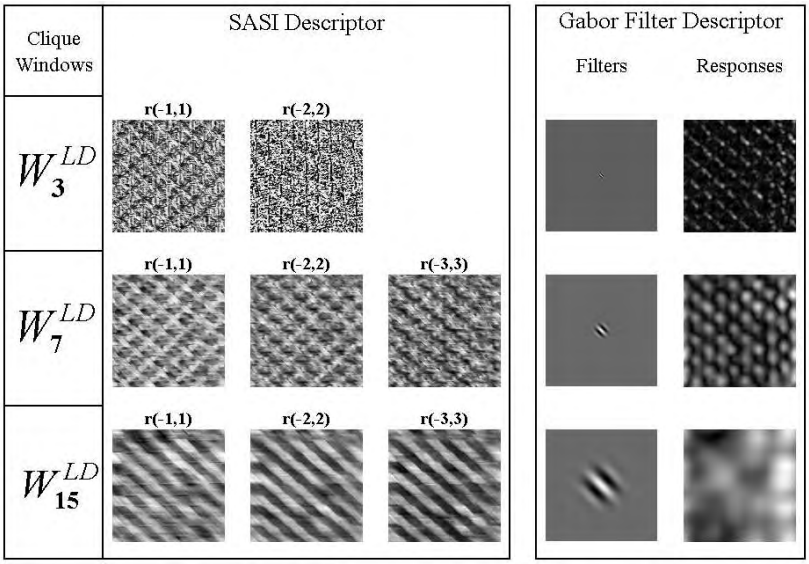


Figure 5.8: SASI and Gabor Filters left diagonal analysis of texture D052 from Brodatz Album.

D004

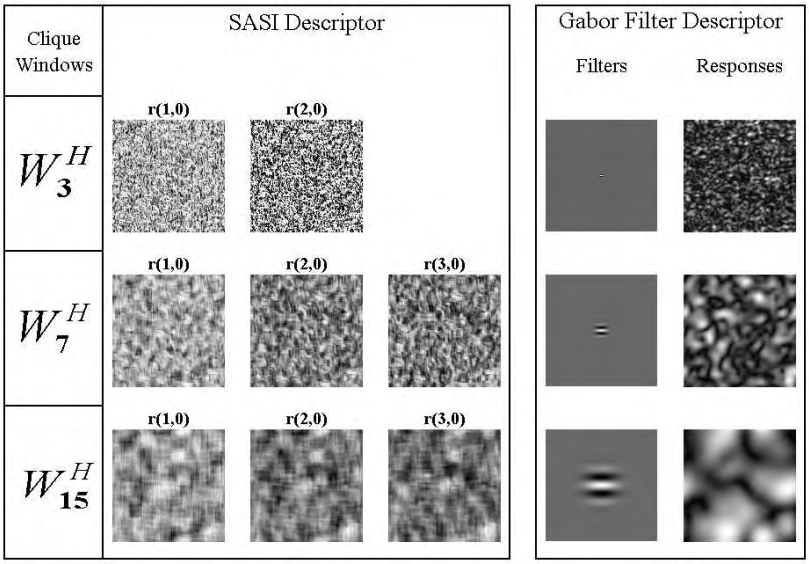


Figure 5.9: SASI and Gabor Filters horizontal analysis of texture D004 from Brodatz Album.

In order to depict the characteristics of both SASI and Gabor Filter descriptors, four selected texture, namely D001, D035, D052, and D004 are used. Like in the previous section, considering the structure of the textures, D001 is analyzed by vertically oriented clique windows and Gabor Filters, whereas D035 and D052 is analyzed by diagonal clique windows and Gabor Filters, since these effects are dominant in the selected textures. Also, the horizontal effects are analyzed in D004.

Table 5.6 indicates the parameters of Gabor and SASI descriptors used in the experiments. The outputs of the clique autocorrelation coefficients and the Gabor Filter responses shown in Figures 5.6, 5.7, 5.8, and 5.9 are scaled to 0 to 255, where 255 (white) and 0 (black) correspond to the high and low responses, respectively.

Table 5.6: Parameters of Gabor and SASI.

SASI clique window size	Gabor filter parameters
3×3	Lower frequency=0.05
7×7	Higher frequency=0.4
15×15	Scale=3, Orientation=4

The results of the filter responses and the clique autocorrelation coefficients depicted in Figure 5.6, 5.7, 5.8, and 5.9 are not directly comparable. However, by analyzing these figures, one can get an idea about how these two descriptors work. Although there is no one to one mathematical correspondence between SASI and Gabor descriptors. Window size of the SASI has some resemblance to the Gabor filter parameter. Therefore, for each image, small to big clique window versus narrow to wide Gabor filter is employed.

A comparison of SASI and Gabor Filter outputs in Figure 5.6 indicates that while SASI captures the sharp edges, Gabor has a tendency to smooth them. The Gaussian structure of the Gabor Filter naturally, bends the straight lines while SASI captures them without any deformation.

It can be seen from Figure 5.7 and 5.8 that, Gabor Filter fails to capture small texels because of the error in discrete approximation of Gabor function for

small windows. The output of fine parameters of Gabor is almost white noise (no pattern).

5.2 Image Retrieval

Textural information can be used in two main application domains: 'between–image search' and 'within–image search'. The first domain deals with searching an image database and finding the most similar image to a given query image. The latter deals with texture segmentation problem, searching a region within an image and finding the most similar region to a given object or a region. Although the proposed descriptor can be used in both domains, in this study, we are mainly concentrated on between-image search problem since in this domain, the performance of a descriptor can be easily evaluated in terms of the average retrieval rates [134, 138, 111, 7, 93]. On the other hand, the concept of similarity is quite subjective.

There are two popular methods for testing the performance of a texture descriptor:

- Each image in the database is divided into sub-images,
- The images in the database are grouped by the user.

The first method enables us to identify each subimage without human subjective support, unless images are similar to each other, whereas the latter method requires grouping criteria that may differ from user to user. Although human support adds subjectivity to the performance measuring process, without this support human visual system consistency of a descriptor cannot be fully measured. This is a dilemma of the performance measuring process.

5.2.1 Image Retrieval without Human Subjectivity

In our experiments, all of the images in Brodatz Album, CUReT, PhoTex and VisTex databases are partitioned into 16 nonoverlapping regions, as shown in

Figure 5.10. Hence, for Brodatz Album $112 \times 16 = 1792$ subimages, for CUReT $61 \times 16 = 976$ subimages, for PhoTex $30 \times 16 = 480$ subimages, and for VisTex $67 \times 16 = 2672$ subimages are obtained, as shown in Table 5.7. Table 5.10 shows the performance of the proposed descriptor for each image database measured in terms of the average retrieval rate, which is defined as the average percentage number of patterns belonging to the same image as the query pattern in top 15 matches (self matches are excluded) [111, 93, 134]. In another words, for each subimage, its most similar 15 subimages are searched within the entire database consists of subimages. In the ideal case, retrieved 15 closest and the query subimage should come from the same original image. This type of performance appraisal is widely used in between-image search applications.

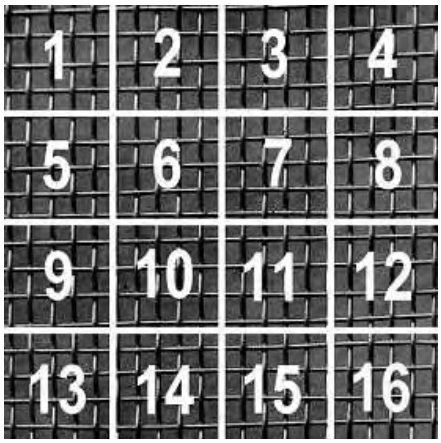


Figure 5.10: Subimages of size 128×128 in D001 with size 512×512.

Table 5.7: Properties of the Brodatz, CUReT, PhoTex, and Vistex image databases.

	Brodatz	CUReT	PhoTex	VisTex	Mixture database
Properties	512×512 gray valued	Various size, colored	512×512 gray valued	512×512 colored	Various size, mixed
Preprocessing	—	Gray scaled, rescaled	—	Gray scaled	Gray scaled, rescaled
# of image	112	61	30	167	370
# of subimage	1792	976	480	2672	5920

Throughout the image retrieval experiments 3×3, 5×5, and 7×7 clique

Table 5.8: The clique window sizes and autocorrelation coefficients for Brodatz Album, CUReT, PhoTex, and VisTex image databases.

Clique window size	Lag Multiplier	# of clique autocorrelation coefficient
3×3	$n = 1, \ldots, 2$	2
5×5	$n = 1, \ldots, 3$	3
7×7	$n = 1, \ldots, 5$	5
		Total=10

Table 5.9: Number of clique windows vs. feature size.

Neighborhood system	# of clique windows	Feature size
η^1	2	40
η^2	12	240
η^3	26	520
η^4	86	1720
η^5	124	2480

Table 5.10: Average retrieval rates of the Brodatz, CUReT, PhoTex, and Vistex image databases.

Descriptor		Brodatz	CUReT	PhoTex	VisTex	Mixture database
GABOR		74.07%	78.61%	80.27%	46.19%	60.56%
SASI	in η^1	70.40%	83.82%	86.84%	47.86%	62.71%
	in η^2	75.47%	85.38%	90.75%	51.80%	66.71%
	in η^3	75.93%	85.68%	92.70%	52.54%	67.20%
	in η^4	75.31%	83.75%	92.27%	51.51%	66.10%
	in η^5	74.84%	82.21%	92.11%	50.65%	65.29%

windows are employed. Table 5.8 shows the autocorrelation coefficients and the related window sizes, which are selected in the preliminary analysis of Brodatz, CUReT, PhoTex and VisTex image databases, as explained in the previous section. As it can be seen from the Table 5.8 for a given clique window type, 10 autocorrelation coefficient are calculated and the feature vector of size 20 (10 mean value + 10 standard deviation) is formed.

In order to make a systematic analysis on the performance of SASI descriptor all clique windows defined in η^1, η^2, η^3, η^4, and η^5 are employed. Table 5.9

shows the number of clique windows and corresponding feature vector size for each neighborhood system.

After calculating the SASI descriptor, the ranking process is accomplished by using Equation 4.6.

The average retrieval rates of Gabor and SASI descriptor are computed for images in Brodatz, CUReT, PhoTex and VisTex databases, respectively and the results are indicated in Table 5.10. Note that SASI descriptor achieves average retrieval rate between 47-92% whereas Gabor Filter remains in the range of 46-80%. We, also, formed a large image database, called *mixture database*, by combining all the subimages of the Brodatz, CUReT, PhoTex and VisTex. In this experiment, for each subimage, its most similar 15 subimages are searched within 5920 subimages. The average retrieval rate for SASI descriptor (in η^3) is 67.20% whereas that of Gabor is 60.56%.

In SASI increasing the order of the neighborhood system larger than 3, decreases the average retrieval rate, due to curse of dimensionality. Thus, the below experiments are done by using SASI with clique windows defined in η^3.

Although we did not perform a systematic set of experiments to check the consistency of SASI to the human visual system, during the experiments, we observed that SASI retrieves images, which are quite consistent to our intuition. In order to show these informal results, six examples, where 6 query texture and their 30 closest textures in the Brodatz Album, are given in Appendix E. Also, SASI is tested on satellite images as shown in Appendix F.

The retrieval rate for each image in the Brodatz album for Gabor and SASI descriptor is shown in Table 5.11. Figure 5.11 indicates the percentage of retrieving the correct subimages as a function of number of retrieved subimages. In Figure 5.11, horizontal axis represents the number of retrieved subimages and vertical axis represents the percentage of the correct retrieved subimages. The performance increases to 93% if the top 100 retrievals are considered instead of 15 retrieval considerations.

Table 5.11: Retrieval rates for the 112 texture images in Brodatz Album.

Texture	Gabor	SASI	Texture	Gabor	SASI	Texture	Gabor	SASI
D001	99.17	100.00	D039	52.08	67.92	D077	100.00	100.00
D002	74.58	61.25	D040	70.42	75.42	D078	97.08	97.08
D003	91.67	82.08	D041	71.67	73.33	D079	96.67	100.00
D004	100.00	90.00	D042	33.75	50.00	D080	86.25	74.58
D005	68.75	58.33	D043	10.00	12.08	D081	100.00	97.92
D006	100.00	100.00	D044	12.92	15.42	D082	100.00	100.00
D007	51.25	49.58	D045	11.67	22.08	D083	99.58	100.00
D008	90.83	96.25	D046	89.17	95.42	D084	100.00	100.00
D009	95.42	92.50	D047	100.00	100.00	D085	100.00	100.00
D010	80.00	79.58	D048	72.92	97.08	D086	68.75	82.50
D011	100.00	100.00	D049	100.00	100.00	D087	94.58	100.00
D012	89.17	90.42	D050	76.25	80.42	D088	22.08	64.58
D013	62.92	52.08	D051	79.58	85.83	D089	34.58	63.33
D014	100.00	100.00	D052	67.08	100.00	D090	37.50	37.92
D015	65.00	65.00	D053	100.00	100.00	D091	34.58	22.50
D016	100.00	100.00	D054	48.75	53.75	D092	96.25	96.25
D017	100.00	100.00	D055	100.00	100.00	D093	87.50	75.42
D018	86.67	95.83	D056	100.00	100.00	D094	99.58	88.75
D019	86.67	99.58	D057	100.00	100.00	D095	100.00	99.58
D020	100.00	100.00	D058	15.42	15.42	D096	77.08	82.92
D021	100.00	100.00	D059	27.92	22.08	D097	34.17	26.25
D022	71.25	87.08	D060	64.17	42.92	D098	42.50	57.08
D023	55.83	44.17	D061	44.17	38.75	D099	37.08	40.83
D024	93.75	93.75	D062	71.67	65.00	D100	41.67	45.00
D025	97.92	81.25	D063	30.83	23.75	D101	57.50	98.75
D026	80.83	80.83	D064	99.17	100.00	D102	74.17	99.17
D027	53.33	34.17	D065	100.00	100.00	D103	66.67	75.83
D028	82.92	81.67	D066	90.00	95.83	D104	68.33	90.42
D029	97.92	100.00	D067	62.92	63.75	D105	67.50	50.00
D030	40.83	50.00	D068	98.33	100.00	D106	58.33	46.67
D031	24.58	39.58	D069	48.75	33.75	D107	52.92	35.83
D032	99.17	97.92	D070	50.00	81.25	D108	45.83	36.25
D033	90.42	82.08	D071	61.67	72.50	D109	79.58	94.58
D034	95.42	96.25	D072	57.92	54.17	D110	98.75	93.75
D035	80.42	80.42	D073	42.92	60.00	D111	60.83	82.08
D036	66.67	72.50	D074	89.17	79.17	D112	71.67	55.42
D037	99.17	100.00	D075	97.92	100.00			
D038	85.83	81.67	D076	99.17	100.00	**Average**	**74.07**	**75.93**

5.2.1.1 The Effects of Various Distance Functions on Retrieval Rate

When ranking texture similarities, Gabor filter descriptor uses L_1 norm distance function and SASI descriptor utilize *"similarity rule"* as exemplified in the pre-

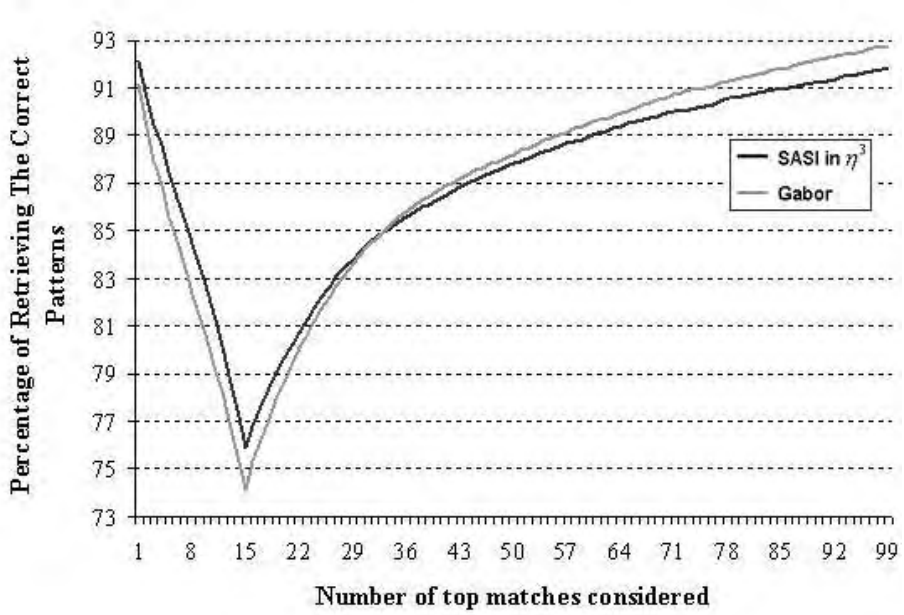


Figure 5.11: Average retrieval rates as a function of number of retrieved subimages for Brodatz Album.

vious section. In this section, we analyze the effects of the different distance functions on the average retrieval rates for both SASI and Gabor filter.

Recall that, there are many metric and non-metric distance function as mentioned in Section 2.1.4.2. However, some of the functions work on "distributions" rather than vectors. These functions include *Quadratic Form Distance*, *Mahalanobis Distance*, *Bhattacharyya Distance*, and *Histogram Distance*. Also, *Kullback Leibler Divergence* and *Jeffrey Divergence* can be applied on non-negative feature vectors.

Since, both SASI and Gabor filter descriptors are based on feature vectors, which may include nonnegative values, we analyze L_1, L_2, L_∞, *Similarity Rule*, *Normalized Correlation* and *Camberra Distance* functions.

Table 5.12 shows the average retrieval rates for each distance function. The experiments are done as in the previous section. It is seen that, the worst average retrieval is obtained on *Camberra Distance* and L_∞, whereas the others are almost compatible. However, the best results are shown on L_1 and *Similarity*

Table 5.12: Average retrieval rates of the Brodatz Album for different distance functions.

Distance Function	Gabor Filter Descriptor	SASI Descriptor				
		η^1	η^2	η^3	η^4	η^5
L_1	73.96%	68.98%	73.83%	74.33%	73.71%	73.48%
L_2	72.62%	69.70%	74.41%	74.79%	73.96%	73.50%
L_∞	62.33%	68.08%	70.80%	70.83%	67.97%	66.97%
Similarity R.	73.29%	70.40%	75.47%	75.93%	75.31%	74.84%
Normalized C.	71.09%	69.67%	75.44%	75.74%	74.93%	74.40%
Camberra Dst.	49.57%	50.45%	51.88%	53.50%	51.93%	50.29%

Rule for Gabor and SASI descriptor, respectively.

The *curse of dimensionality* effects are also shown in Table 5.12. For all distance function, using the neighborhood system larger than η^3 decrease the average retrieval rate.

5.2.2 Image Retrieval with Human Help: Clustering

Brodatz, CUReT, PhoTex and VisTex image databases were never intended to give a fully representative sample set of a broad class of textures for testing the full performance of texture descriptors. As described earlier, during the evaluation of the performance of a descriptor, the images in the database are partitioned into n subimages. Then, for each subimage, its most similar $n-1$ subimages are searched within the subimages in the database. In this case, images can be considered as distinct classes, whereas the subimages correspond to the entries of each class.

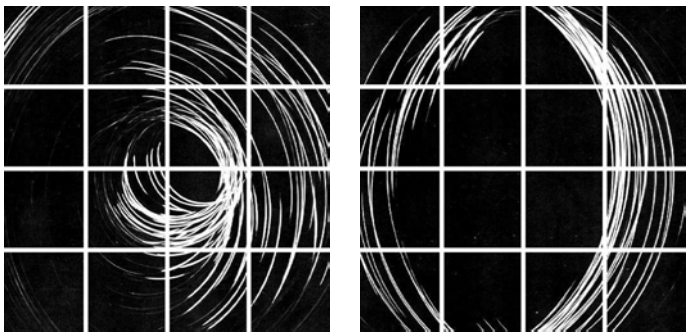
It is expected that the query and the retrieved most similar, i.e. closest, $n-1$ subimages are regions of the same image. This expectation is only valid for an image database, where images of the database are visually different from each other whereas the subimages are visually similar. However, the databases used in the experiments are far from satisfying this expectation.

There are two major problems for measuring the performance of a descriptor. Firstly, some images in the database are quite similar to each other, as shown

D030 D031

Figure 5.12: Similar texture samples in the Brodatz Album.

in Figure 5.12. Secondly, splitting an image into subimages may sometimes yield visually dissimilar textures, as depicted in Figure 5.13. These problems prevent us to measure the consistency of a descriptor with the human visual system. In order to avoid the above problems, subimages may be clustered



D043 D044

Figure 5.13: Sample textures with dissimilar subimages.

by the human support. However, in this case the measured performance of a descriptor is human specific. Also, as the number of clusters is increased, the human subjectivity is also increased.

Since it is hard to manually group the subimages, 112 textured images of Brodatz Album are visually grouped into 32 different clusters, each of which contains 1-8 similar texture [93, 134, 138]. In this study, we use the clustering schema defined in [134] as shown in Table 5.13. After the grouping, each image is partitioned into 16 subimages. Note that, this clustering process can eliminate

Table 5.13: Texture clusters identified by human for Brodatz Album.

Cluster 1	D001,D006,D014,D020,D049
Cluster 2	D008,D056,D064,D065
Cluster 3	D034,D052,D103,D104
Cluster 4	D018,D046,D047
Cluster 5	D011,D016,D017
Cluster 6	D021,D055,D084
Cluster 7	D053,D077,D078,D079
Cluster 8	D005,D032,D033
Cluster 9	D023,D027,D028,D030,D054,D098,D031,D099
Cluster 10	D007,D058,D060
Cluster 11	D059,D061,D063
Cluster 12	D062,D088,D089
Cluster 13	D024,D080,D081,D105,D106
Cluster 14	D050,D051,D068,D070,D076
Cluster 15	D025,D026,D096
Cluster 16	D094,D095
Cluster 17	D069,D071,D072,D093
Cluster 18	D004,D029,D057,D092
Cluster 19	D039,D040,D041,D042
Cluster 20	D003,D010,D022,D035,D036,D087
Cluster 21	D048,D090,D091,D100
Cluster 22	D043,D044,D045
Cluster 23	D019,D082,D083,D085
Cluster 24	D066,D067,D074,D075
Cluster 25	D101,D102
Cluster 26	D002,D073,D111,D112
Cluster 27	D086
Cluster 28	D037,D038
Cluster 29	D009,D109,D110
Cluster 30	D107,D108
Cluster 31	D012,D013
Cluster 32	D015,D097

the problems of Brodatz Album mentioned above, to a certain extent.

This time, the query and retrieved most similar subimages are tested for belonging to the same cluster. Since clusters contain different number of images rather than average retrieval rates, weighted average retrieval rates, where the weights are the number of images in each cluster, are considered. Figure 5.14 illustrates an evaluation based on 32 clusters. Weighted average retrieval rate of SASI descriptor is higher than that of Gabor Filter descriptor. It is seen from

Figure 5.14 that, when clustering is employed, the most similar 8 subimages of a given query subimage are in the same cluster at the rate of 90%.

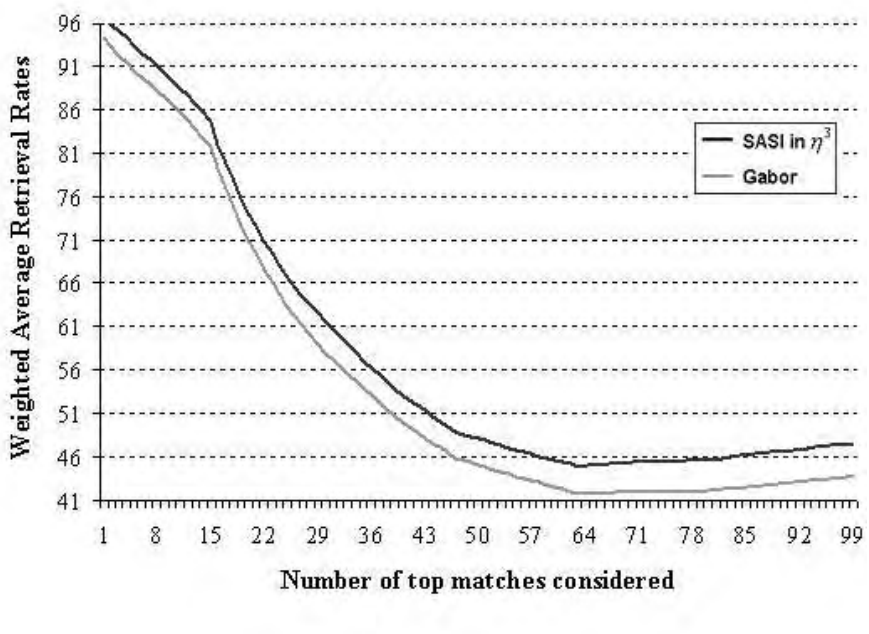


Figure 5.14: Retrieval performance after clustering for Brodatz Album.

Table 5.14: Image groups in the Vistex database.

No	Group Name	Number of Image	No	Group Name	Number of Image
1	Bark	13	11	Misc.	4
2	Brick	9	12	Paintings	13
3	Building	11	13	Sand	7
4	Cloud	2	14	Stone	6
5	Fabric	20	15	Terrain	11
6	Flower	8	16	Tile	11
7	Food	12	17	Water	8
8	Grass	3	18	WheresWaldo	3
9	Leaves	17	19	Wood	3
10	Metal	6			

As stated in earlier, images in the VisTex database were grouped according to their contents by the researchers at the MIT Media Lab. In Table 5.14, 19 groups of images in the VisTex are shown. Same analysis defined in the previous paragraph is applied on visually grouped images of VisTex. Figure 5.15 shows

an evaluation based on 19 clusters, where the most similar 8 subimages of a given query subimage are in the same cluster at the rate of 85%.

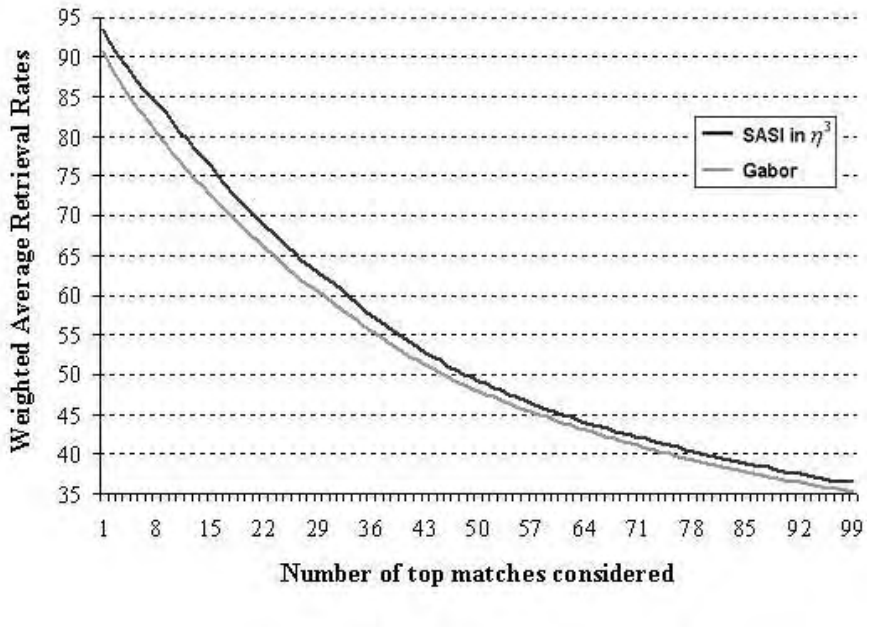


Figure 5.15: Retrieval performance after clustering for VisTex image database.

5.3 Summary

In this chapter, a structural texture descriptor, SASI, is introduced and compared with widely used Gabor Filters. SASI represents a texture in a multidimensional feature space based on the second order statistics of *autocorrelation coefficients* over a set of moving *clique windows*.

The clique windows of various size and shape, which are defined by a neighborhood system, are used as a tool for describing the characteristics of textures in different granularity. The order of the neighborhood system controls the structure of the clique windows. Because of the flexibility in the definition of clique windows, SASI can cope with a broad class of textures, which may consist of sharp corners or small primitives or texels.

The sizes of the clique windows and the lag vectors for the autocorrelation coefficients are the parameters of SASI. Selection of these parameters requires

domain dependent analysis.

During the experiments, it is observed that SASI descriptor captures the structural property of the texture better than the Gabor filters for discontinuities such as sharp corners and high contrast edges. This is basically because of the flexibility in designing the clique window and the representation capacity of the autocorrelation coefficient defined over the clique window.

The second order statistics of clique autocorrelation coefficients on a given texture provides considerable information about the appearance of texture. This fact is verified during the performance tests based on average retrieval rates on four different sets of databases, namely Brodatz Album, CUReT, PhoTex, VisTex, and mixture database obtained by adding all the images in these databases.

CHAPTER 6

LEARNING SIMILARITY SPACE

In this chapter, we propose a method to adapt a content-based image retrieval system into a configurable one. Basically, original feature space of a content-based retrieval system is nonlinearly transformed into a new space, where the distance between the feature vectors is adjusted by learning. The transformation is realized by Artificial Neural Network architecture. A cost function is defined for learning and optimized by simulated annealing method.

6.1 Similarity Matching

Most of the available image retrieval systems are designed by using a fixed set of features and a similarity metric that restricts the performance and human preferences in a specific task. On the other hand, it is well known that the design of a generic feature space, which is linearly separable, is almost impossible in many practical problems.

Roughly glanced though the literature published recently, most of the works, use signal and/or image processing feature extraction methods with or without preprocessing step in order to represent a given image. It is expected that extracted features are computationally feasible, reduce the problem data without discarding valuable information and represent the original image successfully.

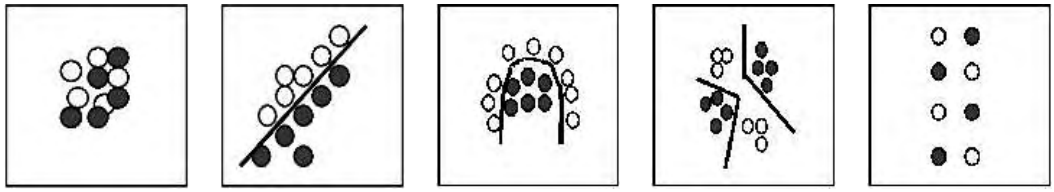
The effectiveness of the representation space is determined by how well patterns, in our case images, from different classes can be separated [139, 140]. Nevertheless, there does not exist a single best representation for a given image [9].

After having selected the *right* set of features, and having characterized an image as a point in a multidimensional vector space, researchers make some assumption about the metric of the space [139, 25]. Typically, feature space is assumed to be Euclidean [25]. After selecting the metric of the space, a distance function is defined, such as Euclidean, Mahalanobis or City Block, in order to measure the distance between the feature vectors. As stated earlier in Section 2.1.4.2, the smaller the distance, the more similar the images to the query. Mathematically, it is expected that feature vectors of similar images are close to each other and this closeness is measured by a distance function.

The similarity metric is critical for content-based image retrieval systems [25, 138]. Unfortunately, image similarity computed by existing mathematical metric is not always consistent with the human perception. For example Euclidean distance may not effectively preserve the perceptual similarity, due to subjectivity of perceived similarity with respect to the related task and database [59]. Moreover, similarity measure based on the nearest neighbor criterion in the feature space is unsuitable in many cases [9]. This is particularly true when the image features correspond to low-level image attributes such as texture, color or shape.

Figure 6.1, shows some problematic examples of features from two different classes [41]. In such cases, using Euclidean distance for the nearest neighbor search might retrieve patterns without any perceptual relevance to the original query [134].

In order to overcome the bottleneck of Euclidean metric discussed above, some efforts have spent in the context of image retrieval [134]. Manjunath and Ma [93] present a learning based approach to retrieve the similar image patterns by using self-organizing maps to get coarse labeling, followed by fine-



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

Figure 6.1: Possible problematic feature space in 2-d. Circles and squares are the feature values of two different classes. (a) Features are inadequate to distinguish the different classes. (b) Features are linearly separable. (c) Decision boundary is curved. (d) Distinct subclasses exist in the data. (e) Feature space is too complex.

tuning process using learning vector quantization. Santini and Jain [25] develop a similarity measure based on fuzzy logic. Minka and Picard report a system, which learns grouping of similar images from positive and negative examples provided by the users during query sessions [59]. Guo, Zhang and Li [138] define a new metric called distance-from-boundary by the use of Support Vector Machines to measure image similarities. The basic idea is that a non-linear boundary separates images from the dissimilar ones.

In this study, we mainly concentrate on texture based image retrieval systems, that query the image database by example, where the user does not have any particular target in mind, but selects an image or draws a sketch and asks to retrieve similar images. Thus, the basic operation is ordering a portion of image database with respect to a similarity metric [141].

Most of the image retrieval systems are non-configurable in which the retrieval process does not depend on the content of the database. We propose a new method, which enables us to make the system configurable without changing the underlying feature extraction mechanism. This task is achieved by the nonlinear transformation of the feature vectors. The transform domain is called "similarity space", where associated distance between feature vectors is trainable. As a transformation scheme Artificial Neural Network is employed. Simulated annealing is used to optimize a predefined cost function. Experiments on the Brodatz Album, indicate better performance in the similarity space com-

pared to the original feature space.

6.2 The Transformation from Feature Space to Similarity Space

We search a space called "similarity space", where the selected distance measure such as Euclidean distance, between patterns can be adjusted to distinguish patterns from different classes and to assign visually similar patterns into the same class. Therefore, in such a space between class variances should be large enough whereas within class variances should be as small as possible. As expected, mapping from the original feature space to similarity space is highly non-linear, subjective and task dependent.

The nonlinear transformation is accomplished by the use of neural networks as indicated in Figure 6.2.

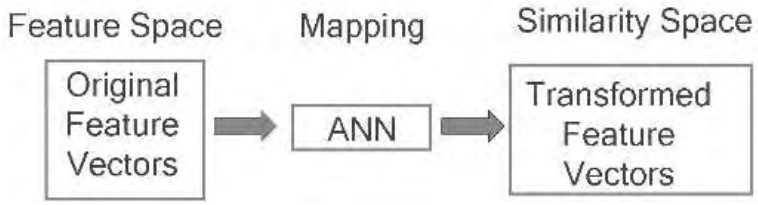


Figure 6.2: A block diagram of the transformation.

A typical ANN architecture, which can be used for such transformation, is shown in Figure 6.3. Since the outputs of ANN are bounded, *i.e.* between 0 to 1, similarity space is also bounded.

The number of input neuron is taken equal to the dimension of the original feature space. On the other hand, determining the number of hidden neuron and also output neuron (i.e. dimension of the similarity space) is not usually straightforward. The goal is to use as few neurons as possible for each layer. Note that reducing the number of output layer reveals to reducing the number of dimensions in the similarity space. Also, if the number of output neuron is chosen less than the number of input neuron, then the transformation works as

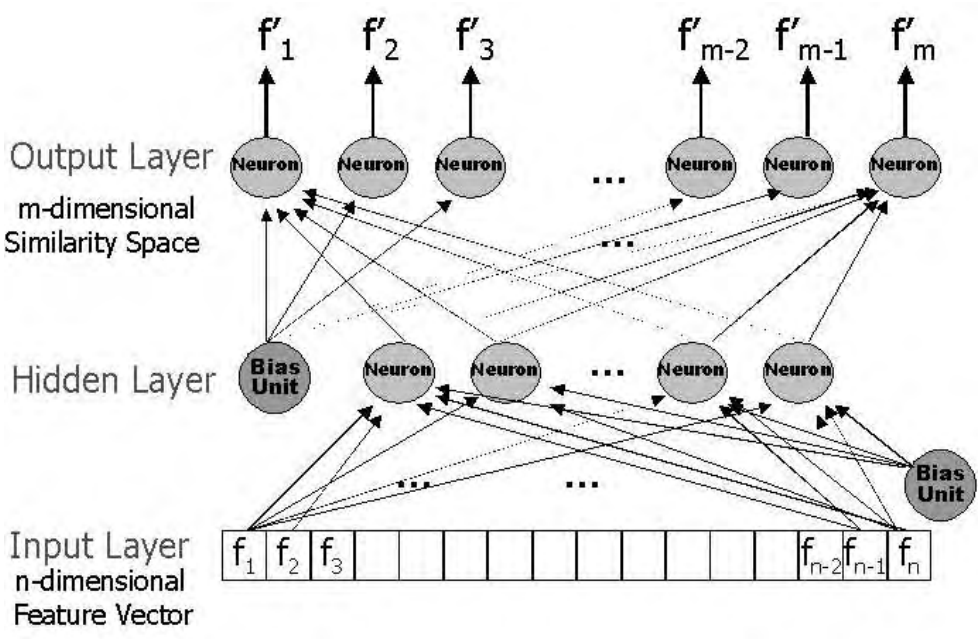


Figure 6.3: The ANN architecture used for transformation.

a nonlinear dimension reduction system [140].

After selecting the number of hidden and output neurons, one can determine the similarity space algorithmically. This time standard backpropagation algorithm can be used for the training since the input versus output is known. Note that, Backpropagation algorithm minimizes the mean square error (MSE) between the generalized and the actual outputs. However, the magnitude of the error does not clearly indicate how successful the ANN is separating the classes to be identified [142].

Due to aforementioned problems, training should be handled as an unsupervised way. First, a cost function must be defined in order to measure the goodness of the similarity space. Second, an algorithm that searches the optimal parameters of the cost function should be employed.

6.3 ANN Training as a Global Optimization Problem

Although various unsupervised learning algorithms such as hill climbing, genetic algorithms, and *etc.* are available to optimize the cost function, we cast the

training of the ANN as a problem of global optimization and use the simulated annealing method.

Mathematically speaking, given a set $\mathcal{S}$ of feasible solutions and real valued cost function $\mathbf{g} : \mathcal{S} \rightarrow R$, global optimization may be formulated as the search for $s \in \mathcal{S}$ such that $\mathbf{g}(s) \leq \mathbf{g}(s') \quad \forall s' \in \mathcal{S}$.

For neural networks, $\mathcal{S}$ is the space of connection weight vectors including bias terms and $\mathbf{g}$ is the cost function. Simulated annealing requires the notion of a neighborhood structure over $\mathcal{S}$, where the neighborhood $N(s_c)$ of the current solution $s_c \in \mathcal{S}$ is the set of new solutions that can be generated from s_c. Typically, $N(s_c)$ consists of slight perturbations of s_c, e.g. a weight vector can be perturbed by adding a random vector in $[-e, +e]^d$, where d is the number of connection weights.

Simulated annealing described in Algorithm 2 is an iterative algorithm that allows escape from local minima in the error surface by probabilistically accepting disimprovements, or "up-hill moves". Although downhill moves are allowed anytime, uphill moves are more likely during the beginning of the process, when the temperature is high, and they become less likely at the end as the temperature becomes lower.

6.4 Texture Image Retrieval

The nonlinear transformation schema proposed in the previous section can be used for any image retrieval and search problem. However, in this study, we suffice to apply the method to the texture image retrieval problem. Since the scope of this dissertation is restricted to texture image retrieval. At this point a cost function is needed to design an optimal transformation, which improves separability of the similarity space. There are many ways of defining the cost function depending on the nature of the problem and the characteristics of the images in the database. One may define the cost function, which considers the full ranking, *i.e.* for each query, the user can determine the order of retrieved subimages. In this study we only cluster the similar textures as close as possible

Algorithm 2 Pseudo code for simulated annealing algorithm.

Begin

$s_c \leftarrow$ random solution in $\mathcal{S}$

$T \leftarrow T_0$

Repeat I times

Repeat J times

Choose $s' \leftarrow$ a random element from $N(s_c)$

$\Delta = \mathbf{g}(s) - \mathbf{g}(s')$

if $\Delta \leq 0$ **then**

$s_c \leftarrow s'$

else

$s_c \leftarrow s'$ with probability $e^{-(\Delta/T)}$

endif

$T \leftarrow T \times k$, where $0 < k < 1$

End.

and separate distinct classes as much as possible.

6.5 Cost Function

Let, A_{xy} and B_{xy} represent the distances from a point x to point y, where the points x and y are in the same class and different class respectively. Mathematically,

$$A_{xy} = \begin{cases} D(x,y) & \text{if x and y are in the same class} \\ \text{Undefined} & \text{Otherwise} \end{cases} , \tag{6.1}$$

and

$$B_{xy} = \begin{cases} D(x,y) & \text{if x and y are } \mathbf{not} \text{ in the same class} \\ \text{Undefined} & \text{Otherwise} \end{cases} , \tag{6.2}$$

where $D(x,y)$ is a distance function between x and y.

Our aim is to minimize A, while maximizing B for each point. Thus, a cost function can be defined as follows:

$$\mathbf{g}(\bar{w}) = E(B - A) \approx \frac{1}{N} \sum_{\forall x,y} (B_{xy} - A_{xy}) , \tag{6.3}$$

where N represents the number of points, and $\bar{w}$ is a weight vector of the ANN.

In some cases, an unbalanced increase in $B-A$ for one point, can significantly increases the cost function with the price of disturbing the separability of the rest of the classes. This situation may maximize the cost function, yet being very poor in well separated clustering. In order overcome this problem, each A_{xy} and B_{xy} is scaled between 0 to 1 by the sigmoid function as shown in Figure 6.4.

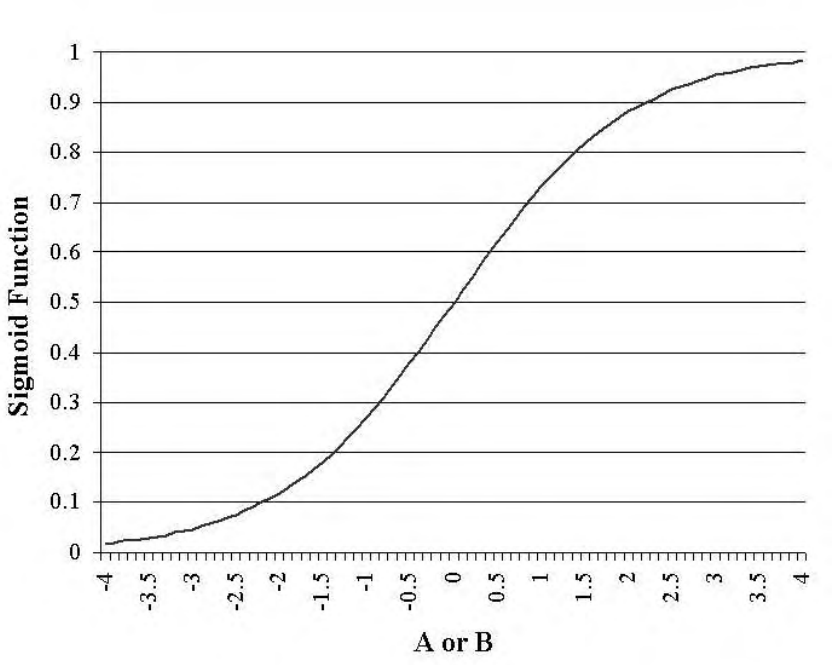


Figure 6.4: Sigmoid function, where c=1.

$$Sgm(x) = \frac{1}{1+e^{-cx}} \ , \tag{6.4}$$

where c is a constant, which depends on the images in the database and dimension of the similarity space. Cost function can be redefined as follows,

$$\mathbf{g}(\bar{w}) = E(Sgm(B) - Sgm(A)) \approx \frac{1}{N} \sum_{\forall x,y} (Sgm(B_{xy}) - Sgm(A_{xy})) \ . \tag{6.5}$$

Due to the computational complexity, there is no need to calculate all A_{xy} and B_{xy}. Thus, for each point, its farthest $n_{SameClass}$ point in the same class and nearest $n_{DifferentClass}$ point in the other classes can be calculated, instead

of A_{xy} and B_{xy} respectively. At the first glance, it can be argued that in order to find farthest and nearest points, an algorithm must calculate all of the A_{xy} and B_{xy}. However, our numerical experiments show that, during the training, there is no need to search $n_{SameClass}$ farthest and $n_{DifferentClass}$ nearest points each time. Since, the training is done using simulated annealing, where the weights are slightly perturbed, the outputs of the ANN are also slowly changed during the iteration. Thus, for each point, its farthest $n_{SameClass}$ and nearest $n_{DifferentClass}$ point can searched at some predefined interval.

It is well known that ANN has tendency to *memorization* rather than *generalization* when overtraining is occurred. A generalization performance of an ANN is proportional to both the number of neurons and the size of the weights. In other words, not only large number of neuron, but also large weights can decreases the generalization performance of an ANN [143, 144]. Moreover, Bartlet in [144] states that generalization performance of an ANN depends on the size of the weights rather than the number of the weights. Thus, another term, also called *weight decay*, is added to cost function as below:

$$\begin{aligned} \mathbf{g}(\bar{w}) = E\Big(Sgm(B) - Sgm(A)\Big) - \tau||\bar{w}|| \approx \\ \frac{1}{N}\sum_{\forall x,y}\Big(Sgm(B_{xy}) - Sgm(A_{xy})\Big) - \tau||\bar{w}|| \ , \end{aligned} \tag{6.6}$$

where $||\bar{w}|| = \sum_{\forall i} w_i \times w_i$, $w_i \in \bar{w}$ and τ is constant that penalize the cost function. Algorithm 3 shows the pseudo-code for cost function computation.

6.6 Experiments

We redefine the problem defined in Section 5.2.1. Basically, each image in the Brodatz Album are divided into 16 nonoverlapping subimage and totally 1792 subimages are obtained. In order to construct training and test image database, for each image, we select 4 subimages randomly. As a result, we have two image databases, namely *training* and *test*, which consist of $112 \times 4 = 448$ and $112 \times 12 = 1344$ subimages, respectively.

Algorithm 3 Pseudo code for cost function computation.

Begin

$Cost = 0$

For each point x

At each K iteration

Begin

Calculate the all distances from the points in the same class to x

Sort them

Store $n_{SameClass}$ farthest point in array FARTHEST[x]

Calculate the all distances from the points in different class to x

Sort them

Store $n_{DifferentClass}$ nearest point in array NEAREST[x]

End

For each point y in FARTHEST[x]

Calculate the distance D from x to y

$Cost = Cost + sgm(D)$

For each point y in NEAREST[x]

Calculate the distance D from x to y

$Cost = Cost - sgm(D)$

$Cost = Cost/(Number\ of\ point \times n_{SameClass} \times n_{DifferentClass}) - \tau||\bar{w}||$

End.

In the training phase, for each subimage in the training image database, its most similar 3 subimages are searched in the similarity space. It is expected that, query and the retrieved subimages are the parts of the same image. The effectiveness of the representation is measured by the average retrieval rate. Hence, maximizing the cost function corresponds to find the weights of the ANN that maximize the average retrieval rate. Also, due to the penalty term, *i.e.* weight decay, in the cost function, small weights are preferable to large ones, which prevents memorization.

Similarly, in the test phase, for each subimage in the test image database, its most similar 12 subimages are searched in the similarity space.

We test both SASI and Gabor filter descriptor performance in the similarity space. In other words, normalized SASI and Gabor filter feature vectors serve as inputs to the ANN.

For Gabor filter descriptor, as explained in Section 5.1.3, second order statistics of the Gabor Filter (4 scales * 6 orientation = 24 filter) responses of a given texture are used as a texture descriptor. Thus, an image is represented by 48 real numbers. We construct an ANN, which has 48 input, 12 hidden, and 8 output neurons, which corresponds to reducing the 48 dimensional space into a space of 8 dimension. The number of hidden and output neuron is chosen by trial and error. Including the bias-terms, the number of weights to be searched is $48 \times 12 + 12 + 12 \times 8 + 8 = 692$.

For SASI descriptor, as stated in Section 5.2.1, the second order statistics of the autocorrelation coefficients calculated on 3×3, 5×5, and 7×7 clique windows defined in η^2 are used. This time, an image is represented by 240 real numbers, as seen in Table 5.9. The same ANN structure, defined above, is used. The number of weights to be searched is $240 \times 12 + 12 + 12 \times 8 + 8 = 2996$.

In order to avoid wasting computation time, we select T_0 such that average accepting uphill moves would be 0.8 at the beginning. Also I=1000, J=10000, k=0.99, $\tau = 0.01$, and $n_{SameClass} = n_{DifferentClass} = 3$ is selected. Scaling parameter c is chosen as 16. The implementation of the neighborhood $N(s_c)$ of

the current solution is done as follows:

$$w_i = w_i + \Big(\text{a random number in } [-0.001, 0.001] \times |w_i|\Big), \quad \forall w_i \in \bar{w} \ . \tag{6.7}$$

The last term, $|w_i|$, enables to produce a new solution, even if the weights are large, thus, the addition of random values between -0.001 to 0.001 to all w_i does not create a new solution.

The experiment was repeated 10 times. Table 6.1 and 6.2 summarizes the training and test results, respectively. Considering the average retrieval rates given in Table 6.2, it is clear that similarity space is more effectively represents the images comparing the normalized original feature space and due to the dimension reduction, i.e. $R^{240} \rightarrow [0,1]^8$ and $R^{48} \rightarrow [0,1]^8$ for SASI and Gabor filter descriptors, respectively. Approximately, 5% and 3% increases on the average retrieval rates for Gabor filter and SASI descriptor are obtained, respectively. Retrieving cost (space + computational time) is, also, saved.

Table 6.1: Training results on Brodatz Album using SASI and Gabor Filter descriptor are shown. For each experiment, average retrieval rates are calculated.

Experiment #	Gabor		SASI in η^2	
	Original Space	Similarity Space	Original Space	Similarity Space
1	73.07%	85.19%	73.96%	88.84%
2	74.26%	83.11%	75.22%	88.62%
3	72.10%	85.49%	73.96%	88.47%
4	71.28%	86.83%	73.51%	89.81%
5	70.83%	82.89%	72.99%	88.10%
6	69.27%	81.70%	73.66%	87.87%
7	74.26%	85.57%	75.15%	89.43%
8	73.07%	85.19%	73.14%	87.20%
9	69.72%	85.27%	74.55%	88.39%
10	71.65%	86.90%	74.11%	87.95%
Average	71.95%	84.81%	74.03%	88.47%

6.7 Summary

This study attacks the separability problem of feature space, designed for content-based image search and retrieval systems. The proposed method concen-

Table 6.2: Test results on Brodatz Album using SASI and Gabor Filter descriptor are shown.

Experiment #	Gabor		SASI in η^2	
	Original Space	Similarity Space	Original Space	Similarity Space
1	73.92%	77.54%	74.47%	79.04%
2	72.50%	77.06%	73.89%	77.10%
3	72.64%	78.42%	74.36%	76.65%
4	72.48%	77.61%	74.30%	77.24%
5	72.59%	77.92%	74.42%	76.40%
6	72.44%	77.72%	74.41%	78.44%
7	72.37%	76.01%	73.74%	75.55%
8	72.36%	78.10%	74.59%	78.10%
9	72.51%	78.21%	73.90%	77.85%
10	72.55%	77.75%	73.71%	77.10%
Average	72.64%	77.63%	74.18%	77.35%

trates on the similarity metrics for texture descriptors. For this purpose, a nonlinear transformation scheme maps the original image into similarity space, where the patterns are better separated for distinct textures and closely clustered for similar textures. The nonlinear transformation is optimal with respect to a predefined cost function. Furthermore, it reduces the dimension of the features, in similarity space. Results indicate that the similarity space is more successful than the original space in retrieving the similar texture images.

CHAPTER 7

CONCLUSIONS

In this thesis, we first introduce a generic texture descriptor for content-based image retrieval, namely SASI, and compare it to one of the most popular and successful descriptor, Gabor Filters. Secondly, in order to overcome the Euclidean space limitations, we suggest a method to adapt an image retrieval system into a configurable one and test this method using SASI and Gabor filter descriptors on texture image retrieval problem.

SASI descriptor consists of second order statistics of autocorrelation coefficient at different lags over a set of clique windows. The concept of clique chain is employed for constructing these structural windows. Clique windows are defined by using a set of neighborhood systems. Changing the order of the neighborhood system, various regular or irregular clique windows are generated. The size of the clique windows and the lag vectors for the autocorrelation coefficients are the parameters of SASI. Selection of these parameters requires domain dependent analysis. The traditional correlogram is a special case of SASI, where only one clique window is used with the size of the image itself and clique autocorrelation coefficient is calculated for all lag vectors. Therefore, SASI can be considered as a generalized correlogram, with varying size and neighborhood system.

SASI descriptors have some superiorities compared to Gabor filters. First of all, the Gaussian structure of the Gabor filters, has the tendency to bend the

straight lines and smoothes the sharp edges. On the other hand, the flexibility in designing a large class of clique windows enables one to capture a great variety of textures without any distortion. Secondly, while SASI descriptor can successfully extract small texels, Gabor functions fail to detect them, due to the relatively large error of the discrete filter approximation. As a result, SASI descriptor captures the structural property of the texture better than the Gabor Filters. This fact is verified during the performance tests based on average retrieval rates applied on subimages and visually clustered images of Brodatz Album, CUReT, PhoTex and VisTex databases. Finally, during the experiments it is observed that SASI descriptor is more consistent to the Human Visual System compared to the Gabor filters, in retrieving the similar images. This is quite reasonable considering the fact that SASI does not restrict the textures to obey the laws of Gaussian nature. On the other hand, the main disadvantage of SASI descriptor is its high computational complexity, especially for large size clique windows required for capturing large texels.

After feature extraction, the selection of similarity distance may be the most crucial phase for content-based image retrieval systems. It is hard to find a distance function that is quite consistent with the human perception. When the retrieval rates are not sufficient, researchers generally try a new distance function or search a new feature extraction method. In order to overcome these difficulties we propose a method, which increases the retrieval rates without changing the underlying feature extraction method and distance function. Basically, original feature space of a content-based retrieval system is nonlinearly transformed into a new space, where the distance between the feature vectors is adjusted by learning. This issue is directly related to separability problem.

In particular, the proposed method concentrates on the similarity metrics for texture descriptors. For this purpose, a nonlinear transformation scheme maps the original image into similarity space, where the patterns are better separated for distinct textures and closely clustered for similar textures. The nonlinear transformation is optimal with respect to a predefined cost function.

Furthermore, it reduces the dimension of the features, in similarity space. Results indicate that the similarity space is more successful than the original space in retrieving the similar texture images.

The method is also independent from the nature of the problem and can be applied to wide range of pattern recognition problems. By changing the cost function, various types of problem can be handled. The dimension reduction in the similarity space needs to be explored further.

REFERENCES

[1] Stephan Hawking. Life in the universe. Public Lectures.

[2] W. Hu and Y. Chen. An overview of world wide web search technologies. In *Proceedings of the 5th World Multi-Conference on Systemic, Cybernetics and Informatics, Orlando, Florida, USA*, pages 356–361, November 1996.

[3] S. Lawrence and C. Giles. Accessibility of information on the web. *Nature*, 400:107–109, 1999.

[4] D. McG. Squire. Learning a similarity-based distance measure for image database organization from human partitionings of an image set. In *Proc. of 4th Workshop on applications of Computer Vision (WACV98),Princeton, NJ, USA*, pages 88–93, 1998.

[5] F. Banfi. Image databases: State of the art and future directions. Internal Working Paper 96-10, University of Fribourg, Switzerlenad, Semptember 1996.

[6] A. K. Jain, S. K. Bhattacharjee, and Y. Chen. On texture in document images. In *Proceedings of Computer Vision and Pattern Recognition*, pages 677–680, 1992.

[7] P. Wu, B. S. Manjunath, S. Newsam, and H.D. Shin. A texture descriptor for browsing and similarity retrieval. *Journal of Signal Processing: Image Communication*, 16(1-2):33–43, 2000.

[8] A. Pentland, R. W. Picard, and S.Sclaroff. Photobook: tools for content based manipulation of image databases. In *Proc. of SPIE, Storage and Retrieval for Image and Video Databases-II,No.2185*, pages 34–47, 1994.

[9] Y. Rui, T. Huang, and S. Chang. Image retrieval: Current techniques, promising directions and open issues. *Journal of Visual Communication and Image Representation*, 10(4):39–62, 1999.

[10] M. Pietikainen, T. Ojala, and Z. Xu. Rotation invariant texture classification using feature distributions. *Pattern Recognition*, 33:43–52, 2000.

[11] A. K. Jain and A. Vailaya. Image retrieval using color and shape. *Pattern Recognition*, 29(8):1233–1244, 1996.

[12] M .Pietikainen and T. Ojala. Rotation invariant texture classification using feature distributions. In *Scandinavian Conference on Image Analysis*, pages 103–110, 1997.

[13] Peter L. Stanchev. General image retrieval model. In *Proc. 27-th Conf. of the Union of Bulgarian Math.,Pleven, Bulgaria*, pages 63–71, 1998.

[14] Hideyuki Tamura and Naokazu Yokoya. Image database systems: A survey. *Pattern Recognition*, 17:29–43, 1984.

[15] Shi-Kuo Chang and Arding Hsu. Image information systems: Where do we go from here? *IEEE Trans. on Knowledge and Data Eng.*, 4(5):431–442, 1992.

[16] C. Berrut et al. Status review on non-text information retrieval. Technical Report ELPUB106, European Commission, Brussels, 1995.

[17] E. Rasmussen. Indexing images. *Annual Review of Information Science and Technology*, 32:169–196, 1997.

[18] Abby A. Goodrum. Image information retrieval: An overview of current research. *Informing Science, Special Issue on Information Science Retrieval*, 3(2):63–67, 2000.

[19] John Eakins and Margeret Graham. Content-based image retrieval: A report to the jisc technology applications programme. Technical report, Institute for Image Data Research, University of Northumbria, 1999.

[20] T. Kato. Database architecture for content-based image retrieval. In *SPIE 1662, Image Storage and Retrieval Systems*, pages 112–123, 1992.

[21] V. N. Gudivada and V. V. Raghavan. Content-based image retrieval systems. *IEEE Computer*, 28(9):18–22, 1995.

[22] J. P. Eakins. Automatic image content retrieval – are we getting anywhere? In *Proceedings of Third International Conference on Electronic Library and Visual Information Research (ELVIRA3), De Montfort University*, pages 123–135, 1996.

[23] Qasim Iqbal and J. K. Aggarwal. Lower-level and higher-level approaches to content-based image retrieval. In *Proceedings of the IEEE South West Symposium on Image Analysis and Interpretation, Austin, Texas, USA*, pages 197–201, 2000.

[24] Ying Li, X. Wan, and C.-C. Jay Kuo. *Search and Retrieval of Digital Imagery*. John Wiley and Sons, 2001.

[25] Simone Santini and Ramesh Jain. Similarity measures. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21(9):871–883, 1999.

[26] Alvin Harvey Kam. *A general Multiscale Scheme for Unsupervised Image Segmentation*. PhD thesis, University of Cambridge, 2000.

[27] W. Yu, J. Fritts, and F. Sun. A hierarchical image segmentation algorithm. In *International Conference on Multimedia and Expo ICME 2002*, pages 221–224, 2002.

[28] S. Sural, G. Qian, and S. Pramanik. Segmentation and histogram generation using the hsv color space for image retrieval. In *IEEE International Conference on Image Processing Rochester*, pages II:589–592, 2002.

[29] Y. Nakagawa and A. Rosenfold. Some experiments on variable thresholding. *Pattern Recognition*, 11:191–204, 1979.

[30] R. Porter and N. Canagarajah. A robust automatic clustering scheme for image segmentation using wavelets. *IEEE Trans. on Image Proc.*, 5(4):662–665, 1996.

[31] L.O. Hall, A.M. Bensaid, L.P. Clarke, R.P. Velthuizen, M. Silbiger, and J.C. Bezdek. A comparision of neural network and fuzzy clustering techniques in segmenting magnetic resonance images of the barin. *IEEE Trans. on Neural Networks*, 3(5):672–681, 1992.

[32] S. Lancer. Edge-based colour segmentation on the cie lab space. In *Proc. of the 15th DAGM-Symp. Mustererkennun, Lubeck, Germany*, pages 639–646, 1993.

[33] I.A. Ismaili and D.F. Gillies. Color image segmentation using regression analysis in rgb space. *Machine Graphics and Vision*, 3(1):373–384, 1994.

[34] L. Vincent and P. Soille. Watersheds in digital spaces: An efficient algorithm based on immersion simulations. *IEEE Trans. on Pat. Anal.and Mach. Intell.*, 13(6):583–598, 1991.

[35] Z. Wu and R. Leahy. An optimal graph theoretic approach to data clustering: Theory and its application to image segmentation. *IEEE Trans. on Pat. Anal. and Mach. Intell.*, 15(11):1101–1113, 1993.

[36] S. Krishnamachari and R. Chellappa. Multiresolution gauss-markov random field models for texture segmentation. *IEEE Trans. on Image Processing*, 6(2):251–267, 1997.

[37] M. N. Do and M. Vetterli. Wavelet-based texture retrieval using generalized gaussian density and kullback-leibler distance. *IEEE Trans. Image Processing*, 11(2):146–158, 2002.

[38] John M. Zachary. *An information theoretic approach to content based image retrieval*. PhD thesis, Department of Computer Science,Louisiana State University, 2000.

[39] Anil K. Jain. *Fundamentals of Digital Image Processing.* Prentice Hall, 1989.

[40] Rafael Gonzales and Richard Woods. *Digital Image Processing.* Addison Wesley, 1992.

[41] Richard Duda and Peter Hart. *Pattern Classification and Scene Analysis.* John Wiley and Sons, 1973.

[42] J. Puzicha, T. Hofmann, and J. M. Buhmann. Non-parametric similarity measures for unsupervised texture segmentation and image retrieval. In *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition*, pages 267–272, 1997.

[43] Y. Deng and B. S. Manjunath. An efficient low-dimensional color indexing scheme for region-based image retrieval. In *Proc. IEEE Intl. Conference on Acoustics, Speech and Signal Processing (ICASSP-99), Phoenix, Arizona*, pages 3017–3020, 1999.

[44] Remco C. Veltkamp and Mirela Tanase. Content-based image retrieval systems: A survey. Technical Report UU-CS-2000-34, Department of Computing Science, Utrecht University, 2002.

[45] Changliang Wang. Content-based image indexing and retrieval: Feature extraction and feature similarity metrics, 2000.

[46] E. Mathias. Comparing the influence of color spaces and metrics in content-based image retrieval. In *Proc. of Inter. Sym. on Computer Graphics, Image Processing an Vision*, pages 371–378, 1998.

[47] P. Heckbert. Color image quantization for frame buffer display. *Computer Graphics*, 16(3):299–304, 1982.

[48] X. Wan and C.-C. Jay Kuo. Color distribution analysis and quantization for image retrieval. In *Proc. SPIE Storage and Retrieval Still Image Video Databases IV 2670,*, pages 8–16, 1996.

[49] M. Stricker and M. Orengo. Similarity of color images. In *SPIE Storage and Retrieval for Image and Video Databases III, vol.2185*, pages 381–392, 1995.

[50] M. J. Swain and D. Ballard. Color indexing. *International Journal of Computer Vision*, 7(1):11–32, 1991.

[51] T. Kato, T. Kurita, and H. Shimogaki. Intelligent visual interaction with the database systems toward the multimedia personal interface. *Journal of Information Processing of Japan*, 2:134–143, 1991.

[52] G. Pass and R. Zabith. Histogram refinement for content-based image retrieval. In *IEEE Workshop on Applicaitons of Computer Vision*, pages 96–102, 1996.

[53] J. Huang and et. al. Image indexing using color correlogram. In *IEEE Int. Conf. on Computer Vision and Pat. Rec., Puerto Rico*, pages 762–768, 1997.

[54] Maytham Safar, Cyrus Shahabi, and Cheng-Hai Tan. Resiliency and robustness of alternative shape-based image retrieval. In *International Database Engineering and Application Symposium*, pages 337–348, 2000.

[55] M. Flickner, H. Sawhney, W. Niblack, J. Ashley, Q. Huang, B. Dom, M. Gorkani, J. Hafner, D. Lee, D. Petkovic, D. Steele, and P. Yanker. Query by image and video content: The qbic system. *IEEE Computer*, 28(9):23–32, 1995.

[56] S. R. Gunn and M. S. Nixon. A robust snake implementation; a dual active contour. *IEEE Trans. Pattern Anal. Mach. Intell.*, 19(1):63–68, 1997.

[57] Y. Rui, A. C. She, and T. S. Huang. Modified fourier descriptors for shape representation – a practical approach. In *Proc. of First International Workshop on Image Databases and Multimedia Search, The Netherlands*, 1996.

[58] Nafiz Arica and Fatos Yarman-Vural. A perceptual shape descriptor. In *Proc. International Conference on Pattern Recognition (ICPR)*, Quebec, Canada, 2002.

[59] D. McG. Squire and T. Pun. Assessing agreement between human and machine clusterings of image databases. *Pattern Recognition*, 31(12):1905–1919, 1998.

[60] Walter Rudin. *Principles of Mathematical Analysis*. International Series in Pure and Applied Mathematics, McGraw-Hill, 1976.

[61] Y. Rubner and C. Tomasi. Texture metrics. In *Proc. of the IEEE International Conference on Systems, Man and Cybernetics, San-Diego, CA*, pages 4601–4607, 1998.

[62] D. Randall Wilson and Tony R. Martinez. Improved heterogeneous distance functions. *Journal of Artificial Intelligence Research*, 6:1–34, 1997.

[63] S. Aksoy and R. M. Haralick. Probabilistic vs. geometric similarity measures for image retrieval. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition 2(1)*, pages 357–362, 2000.

[64] R. Jain, S.N.J. Murthy, and L. Tran. Similarity measures for image database. In *Proc. IEEE Conf. Fuzzy Logic 1(1)*, pages 1247–1254, 1995.

[65] K. Fukunaga. *Introduction to Statistical Pattern Recognition.* Second Edition, Academic Press, 1990.

[66] K. Xu, B. Georgescu, D. Comaniciu, and P. Meer. Performance analysis in content-based retrieval with textures. In *Proc. of the IEEE International Conference on Pattern Recognition (ICPR 2000)*, pages 4275–4278, 2000.

[67] C. Breiteneder and H. Eidenberger. Content-based image retrieval in digital libraries. In *Kyoto International Conference on Digital Libraries*, pages 67–74, 2000.

[68] Jens Micheal Carstensen. *Description and Simulation of Visual Texture.* PhD thesis, Technical University of Denmark, 1992.

[69] A.K. Jain and K. Karu. Learning texture discrimination masks. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 18:195–205, 1996.

[70] G.R. Cross and A.K. Jain. Markov random field texture models. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 5:25–29, 1983.

[71] J. Sklansky. Image segmentation and feature extraction. *IEEE Transactions on Systems, Man and Cybernetics*, 8:237–247, 1978.

[72] H. Tamura, S. Mori, and Y. Yamawaki. Textural features corresponding to visual perception. *IEEE Transactions on Systems, Man and Cybernetics*, 8:460–473, 1978.

[73] S. W. Zucker and K. Kant. Multi-level representations for texture discrimination. In *Proceedings of the IEEE Conference on Pattern Recognition and Image Processing*, pages 609–614, 1981.

[74] R.M. Haralick, K. Shanmugam, and I. Dinstein. Textural features for image classification. *IEEE Transactions on Systems, Man and Cybernetics*, 3(6):610–621, 1973.

[75] C. Carson, S. Belongie, H. Greenspan, and J. Malik. Region-based image querying. In *IEEE Workshop on Content-based Access of Image and Video Libraries*, pages 42–49, 1997.

[76] S. Belongie, C. Carson, H. Greenspan, and J. Malik. Color and texture-based image segmentation using em and its application to content-based image retrieval. In *Proc. of the Int. Conference on Computer Vision*, pages 675–682, 1998.

[77] D. Lee and T. Schenk. Image segmentation from texture measurements. *International Archives of Photogrammetry and Remote Sensing*, III:195–199, 1992.

[78] Li-Yi Wei. *Texture Synthesis by Fixed Neighborhood Searching.* PhD thesis, Stanford University, 2001.

[79] H. Muller, W. Muller, D.M. Squire, S. Marchand-Maillet, and T. Pun. Performance evaluation in content-based image retrieval: overview and proposals. *Pattern Recognition Letters*, 22(5):593–601, 2001.

[80] G. V. Wouver, P. Scheunders, and D. Dyck S. Livens. Wavelet correlation signatures for color texture characterization. *Pattern Recognition*, 32:443–451, 1999.

[81] C. Palm, D. Keysers, T. Lehmann, and K. Spitzer. Gabor filtering of complex hue/saturation images for color texture characterization. In *Proc. of 5th Joint Conference on Information Science JCIS2000*, pages 45–49, 2000.

[82] C. Palm, T. Lehmann, and K. Spitzer. Color texture analysis of moving vocal cords using approaches from statistics and signal theory. In *Proc. of 4th International Workshop: Advances in Quantitative Laryngoscopy, Voice and Speech Research*, pages 46–56, 2000.

[83] R. Lakmann and L. Priese. A reduced covariance color texture model for micro-textures. In *Proc. of 10th Scandinavian Conference on Image Analysis*, pages 947–953, 1997.

[84] H. Yu, M. Li, H. Zhang, and J. Feng. Color texture moments for content-based image retrieval. In *IEEE International Conference on Image Processing Rochester*, pages III:929–932, 2002.

[85] J. R. Smith and S. Chang. Quad-tree segmentation for texture-based image query. In *Proc. of 2nd Annual ACM Multimedia Conference, San Francisco*, pages 279–286, 1994.

[86] Mihran Tuceryan and Anil K. Jain. *The Handbook of Pattern Recognition and Computer Vision (2nd Edition), Chapter 2: Texture Analysis*. World Scientific Publishing Co., 1998.

[87] K. J. Khiani, S. M. Yamany, and A. A. Farag. Classification of the effects of f-actin under treatment of drugs in endothelial cells. In *Proceedings of the ANNIE-96, Artificial Neural Networks in Engineering, Rolla, Missouri*, pages 320–326, November 1996.

[88] J. Weszka, C. Dyer, and A. Rosenfold. A comparative study of texture measures for terrain classification. *IEEE Transaction on System, Man and Cybernetics*, 6(4):269–285, 1976.

[89] R. Chellappa and S. Chatterjee. Classification of textures using gaussian markov random fields. *IEEE Trans. Acoust., Speech, Signal Proc.*, 33:959–963, 1985.

[90] S. Geman and D. Geman. Stochastic relaxation, gibbs distributions and the bayesian restoration of images. *IEEE Trans. on Pattern Anal. and Machine Intell.*, 6(6):721–741, 1984.

[91] J. Wan, J. Mao, and C. D. Wang. Multiresolution rotation invariant simultaneous auto regressive model for texture analysis. In *Proc. the International Conference on Pattern Recognition (ICPR88)*, pages 845–847, 1988.

[92] J. Mao and A. K. Jain. Texture classification and segmentation using multiresolution simultaneous autoregressive models. *Pattern Recognition*, 25(2):173–188, 1992.

[93] B. S. Manjunath and W. Y. Ma. Texture features for browsing and retrieval of image data. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 18:837–842, 1996.

[94] A. P. Pentland. Fractal-based description of natural scenes. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 6:661–674, 1984.

[95] L. M. Kaplan, R. Murenzi, and K. R. Namuduri. Fast texture database retrieval using extended fractal features. In *SPIE Storage and Retrieval for Image and Video Databases, San Jose, CA, USA*, pages 162–175, 1998.

[96] J. M. Keller, S. Chen, and R. M. Crownover. Texture description and segmentation through fractal geometry. *Computer Vision, Graphics and Image Processing*, 45:150–166, 1989.

[97] L.M. Kaplan. Extended fractal analysis for texture classification and segmentation. *IEEE Transactions on Image Processing*, 8(11):1572–1585, 1999.

[98] S. Y. Lu and K. S. Fu. A syntactic approach to texture analysis. *Computer Graphics and Image Processing*, 7:303–330, 1978.

[99] Yan Qiu CHEN. *Novel Techniques For Image Texture Classification.* PhD thesis, University of Southampton, 1995.

[100] F. Tomita and Tsuji Saburo. *Computer Analysis of Visual Textures.* Kluwer Academic Publishers, 1990.

[101] A. Laine and J. Fan. Texture classification by wavelet packet signatures. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 15:1186–1191, 1993.

[102] T. Randen and J. H. Husoy. Filtering for texture classification: A comparative study. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 21:291–310, 1999.

[103] S. S. Liu and M. E. Jernigan. Texture analysis and discrimination in additive noise. *Computer Vision, Graphics and Image Processing*, 49:52–67, 1990.

[104] I. Fogel and D. Sagi. Gabor filters and texture discriminator. *Biological Cybernetics*, 61:103–113, 1989.

[105] A. Jain and F. Farrokhnia. Unsupervised texture segmentation using gabor filters. *Pattern Recognition*, 24:1167–1186, 1991.

[106] M. Vetterli. Wavelets and filter banks: theory and design. *IEEE Trans. on Signal Processing*, 40:2207–2231, 1992.

[107] M. H. Grob, R. Koch, L. Lippert, and A. Dreger. Multiscale image texture analysis in wavelet spaces. In *Proc. IEEE-ICIP'94*, pages III:412–416, 1994.

[108] T. Chang and C.C. Jay Kuo. Texture analysis abd classification with tree-structured wavelet transform. *IEEE Trans. on Image Processing*, 2:429–441, 1993.

[109] T.S. Lee. Image representation using 2d gabor wavelets. *IEEE Trans. On Pattern Analysis and Machine Intelligence*, 18(10):959–971, 1996.

[110] T. Ojala, K. Valkealahti, E. Oja, and M. Pietikainen. Texture discrimination with multidimensional distributions of signed gray level differences. In *Proceedings of the Fourth Asian Conference on Computer Vision*, pages 1082–1088, 2000.

[111] W. Y. Ma and B. S. Manjunath. Texture features and learning similarity. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 425–430, 1996.

[112] D. Dunn and W. Higgins. Optimal gabor filters for texture segmentation. *IEEE Trans. Image Processing*, 4:947–964, 1995.

[113] W. Y. Ma and B. S. Manjunath. A pattern thesaurus for browsing large aerial photographs. Technical Report 96-10, University of California, 1996.

[114] J. M. Frankos. Orthogonal decomposition of 2d random fields and their applications in 2d spectral estimation. In *Signal Processing and its Application*, pages 20–227, 1993.

[115] F. Liu and R. W. Picard. Periodicity, directionality and randomness: Wold features for image modelling and retrieval. *IEEE Trans. on Pattern Analysis and Machine Learning*, 18(7):722–733, 1996.

[116] A. R. Rao and G. L. Lohse. Towards a texture naming system: Identifying relevant dimensions of texture. In *Proc. IEEE Conf. Visualization*, pages 1:220–227, 1993.

[117] mpeg 7. http://mpeg.telecomitalialab.com/standards/mpeg–7/mpeg–7-.htm.

[118] B. S. Manjunath, J-R. Ohm, V. V. Vasudevan, and A. Yamada. Color and texture descriptors. *IEEE Trans. on Circuits and Systems for Video Technology*, 11(6):703–715, 2001.

[119] T. Ojala, T. Maenpaa, J. Viertola, J. Kylonen, and M. Pietikainen. Emprical evalutation o mpef-7 texture descriptors with a large scale experiment. In *The 2nd International Workshop on Texture Analysis*, pages 99–102, 2002.

[120] P. P. Ohanian and R. C. Dubes. Performance evaluations for four classes of texture features. *Pattern Recognition*, 25(8):819–833, 1992.

[121] V. Castelli and L. D. Bergman. *Image Databases: Search and Retrieval of Digital Imagery*. Wiley, NewYork, 2002.

[122] T.R. Reed and J.M.H. Buf. A review of recent texture segmentation and feature extraction techniques. *Computer Vision Graphics and Image Processing: Image Understanding*, 57(3):359–372, 1993.

[123] C.C. Chen and C.C. Chen. Filtering methods for texture discrimination. *Pattern Recognition Letters*, 20:783–790, 1999.

[124] W. Y. Ma and B. S. Manjunath. A comparision of wavelet transform features for texture image annotation. In *Proc. of IEEE International Conference on Image Processing*, pages II:256–259, 1995.

[125] S. Z. Li. *Markov Random Field Modeling in Computer Vision*. Springer-Verlag, 1995.

[126] A. Carkacioglu and F. T. Yarman-Vural. Sasi: A generic texture descriptor for image retrieval. *Pattern Recognition*, 36(11):2615–2633, 2003.

[127] A. Carkacioglu and F. T. Yarman-Vural. Set of texture similarity measures. In *Machine Vision Applicaitons in Industrial Inspection, SPIE Proceedings, Vol.3029*, pages 118–127, 1997.

[128] A. Carkacioglu and F. T. Yarman-Vural. A set of texture similarity measures for binary and gray level markov random field textures. In *IAPR-ICIAP Vol.2*, pages 127–133, 1997.

[129] A. Carkacioglu and F. T. Yarman-Vural. Sasi: a new texture descriptor for image retrieval. In *IEEE International Conference on Image Processing*, pages 137–140, 2001.

[130] P. Brodatz. *Textures, A photographic album for artists and designers*. Dover Publications, New York, 1966.

[131] K. J. Dana, B. van Ginneken, S. K. Nayar, and J. J. Koenderink. Reflectance and texture of real world surfaces. *ACM Transactions on Graphics*, 18(1):1–34, 1999.

[132] J. Wu. Investigation the use of photometric stereo for surface rotation invariant texture classification. Technical Report Technical report RM/99/01, Department of Computing and Electrical Engineering, Heriot-Watt University, Edinburgh, 1999.

[133] R. Picard, C. Graczyk, S. Mann, J. Wachman, L. Picard, and L. Campbell. Vistex. via ftp:whitechapel.media.mit.edu, 1995. Copyright 1995 Massachusetts Institute of Technology.

[134] G. Guo, S. Z. Li, and K. L. Chan. Learning similarity for texture image retrieval. In *Proceedings of the European Conference on Computer Vision, Dublin, Irelan*, pages 178–190, 2000.

[135] M. Singh and S. Singh. Spatial texture analysis: A comparative study. In *Proceedings of the 15.th International Conference on Pattern Recognition (ICPR'02)*, pages 676–679, 2002.

[136] C. Chatfield. *The analysis of Time Series*. Chapman and Hall, London, 1989.

[137] T. C. Bailey and A. C. Gatrell. *Interactive spatial data analysis*. Longman, Essex, 1995.

[138] G. Guo, H. Zhang, and S. Z. Li. Distance from boundary as a metric for texture image retrieval. In *Proceedings of the ICASSP*, 2001.

[139] A. K. Jain, R. P. W. Duin, and J. Mao. Statistical pattern recognition: A review. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(1):4–37, 2000.

[140] A. Carkacioglu and F. T. Yarman-Vural. Learning similarity space. In *IEEE International Conference on Image Processing*, pages 405–408, 2002.

[141] S. Santini and R. Jain. Similarity matching. In *Recent Developments in Computer Vision, Second Asian Conference on Computer Vision, ACCV '95, Singapore*, pages 571–580, December 5-8, 1995.

[142] R. Hochman, M. K. Taghi, B. A. Edward, and John P. H. John. Evolutionary neural networks: A robust approach to software reliability problems,albuquerque. In *Proc. the Eighth Int. Symposium on Software Reliability Engineering*, pages 13–26, Nov 2 - 5, 1997.

[143] Anders Krogh and John A. Hertz. A simple weight decay can improve generalization. In John E. Moody, Steve J. Hanson, and Richard P. Lippmann, editors, *Advances in Neural Information Processing Systems*, volume 4, pages 950–957. Morgan Kaufmann Publishers, Inc., 1992.

[144] P. Bartlet. The sample complexity of pattern classification with neural networks: The size of the weights is more important than the size of the network. *IEEE Transactions on Information Theory*, 44(2):525–536, 1998.

APPENDIX A

Brodatz Album

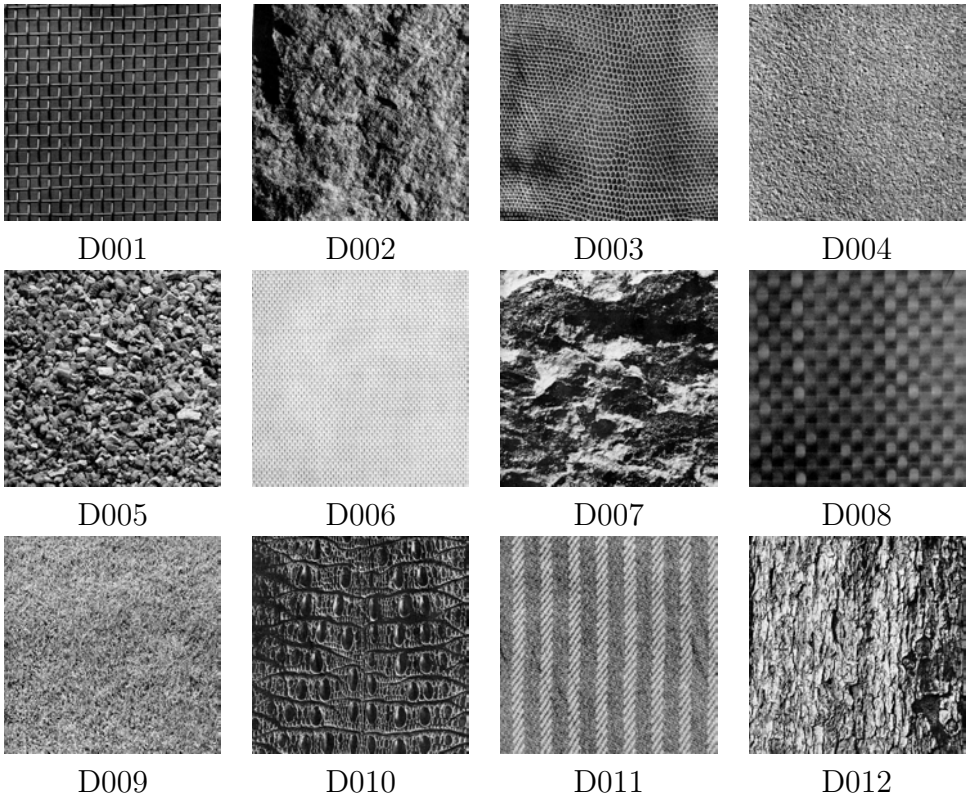


Figure A.1: Brodatz Album images shown as thumbnails, D001–D012.

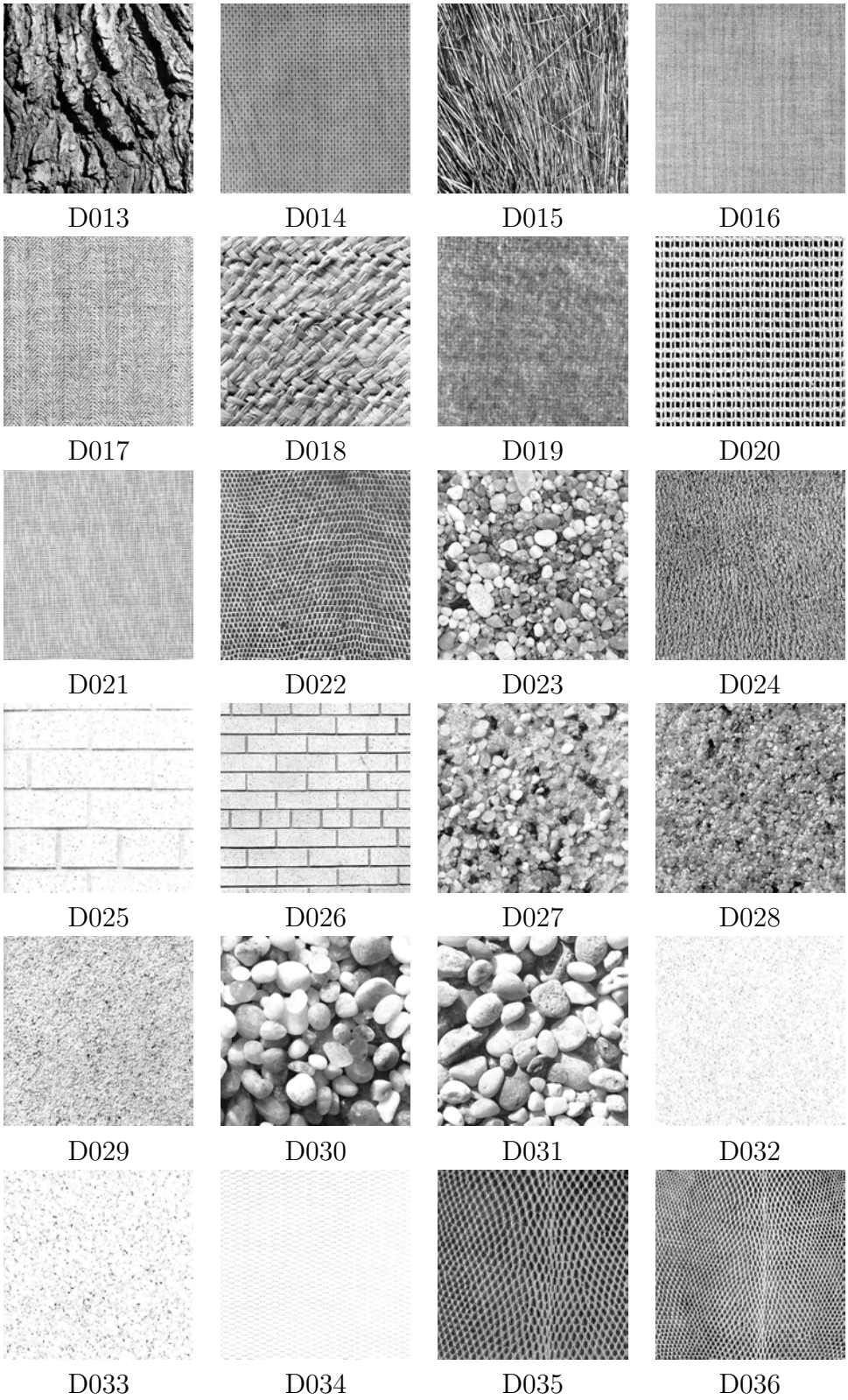


Figure A.2: Brodatz Album images shown as thumbnails, D013–D036.

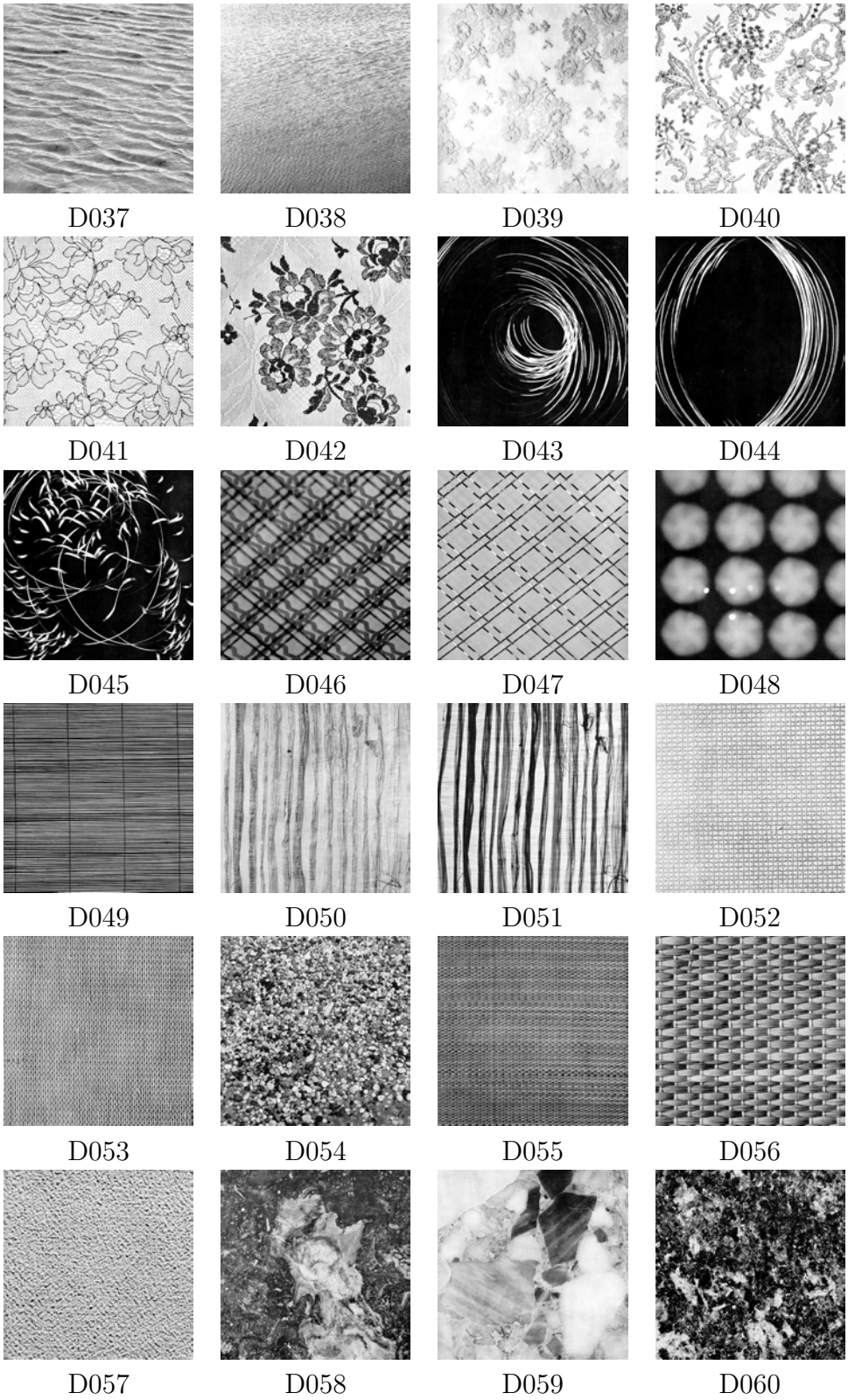


Figure A.3: Brodatz Album images shown as thumbnails, D037–D060.

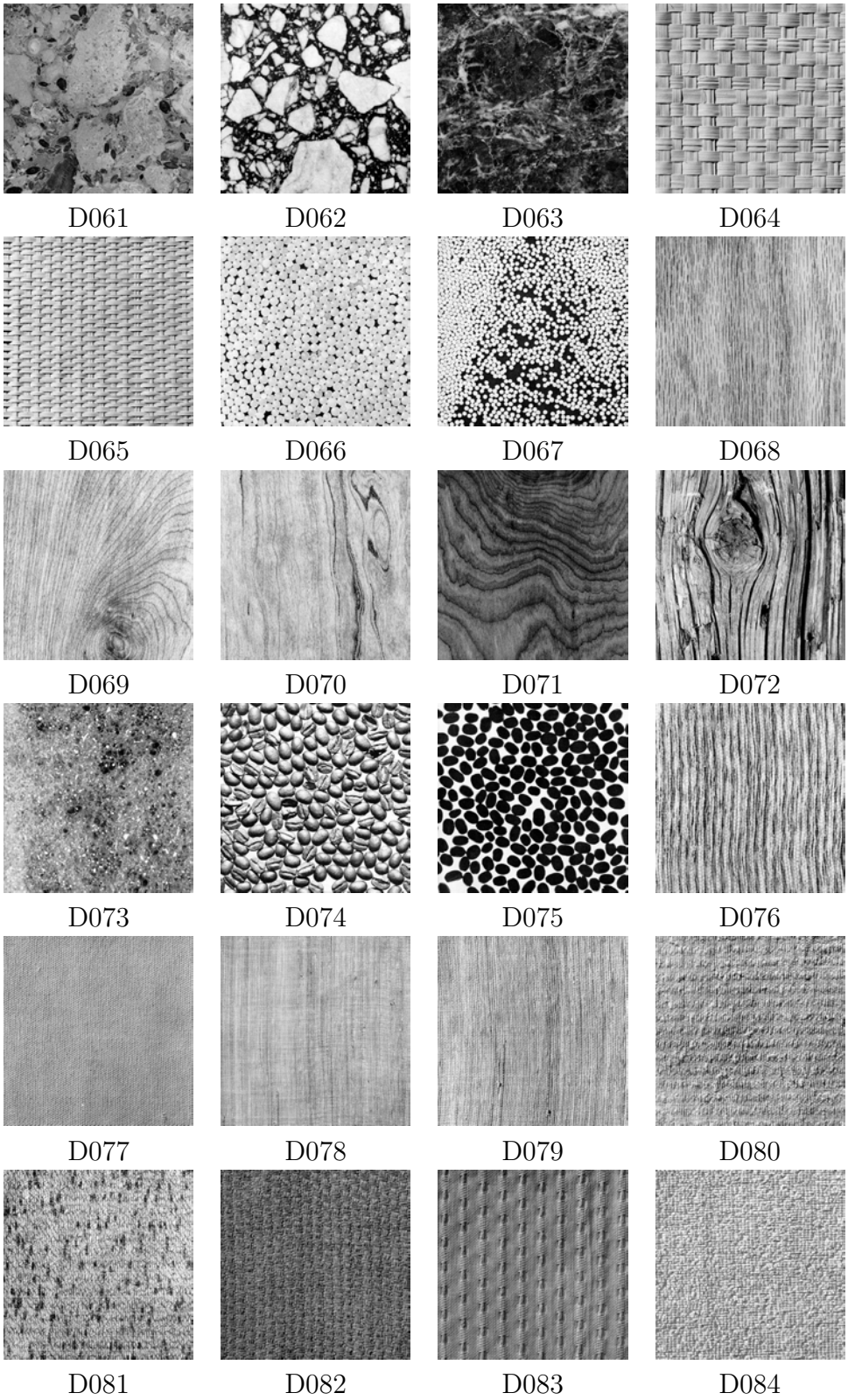


Figure A.4: Brodatz Album images shown as thumbnails, D061–D084.

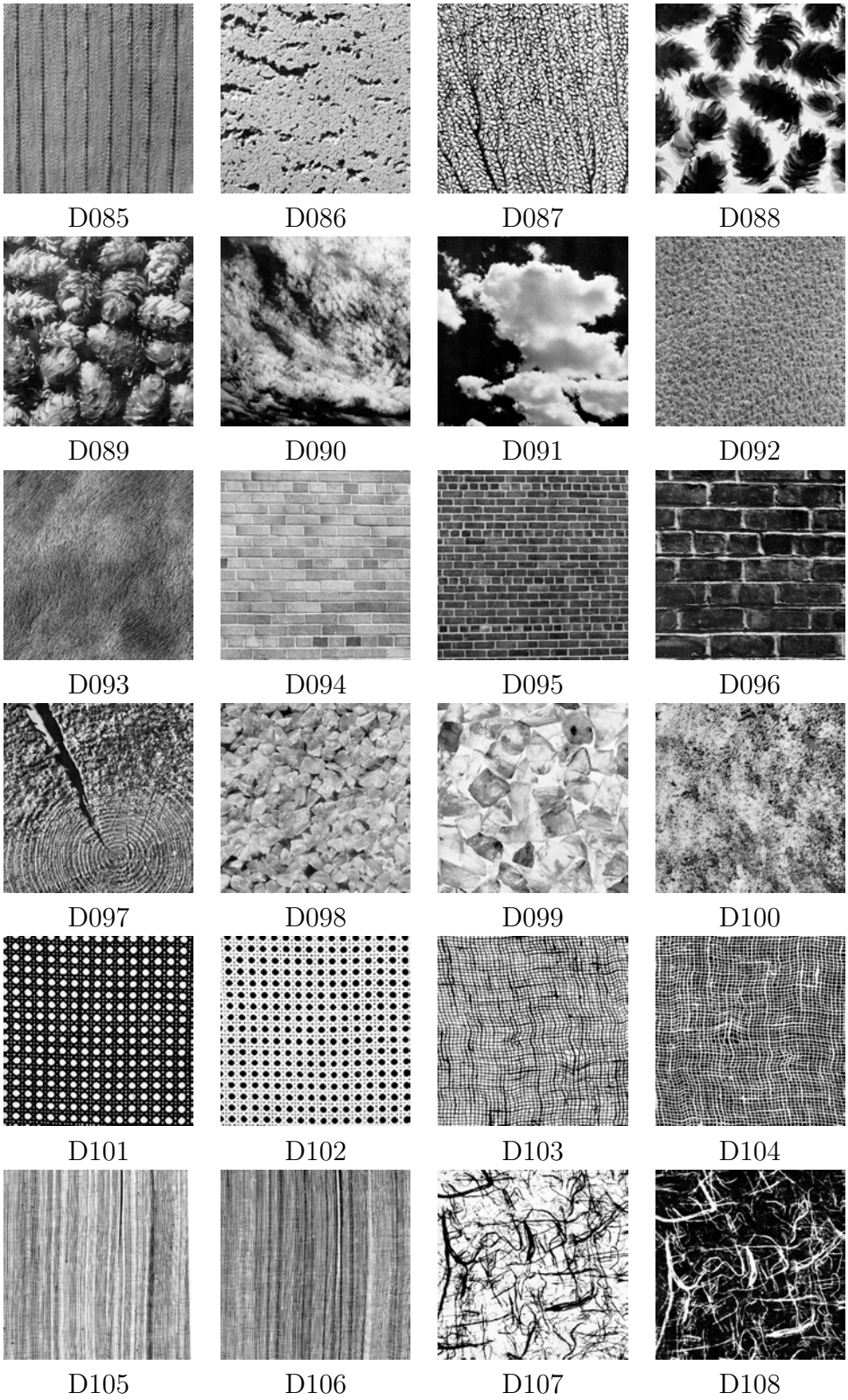
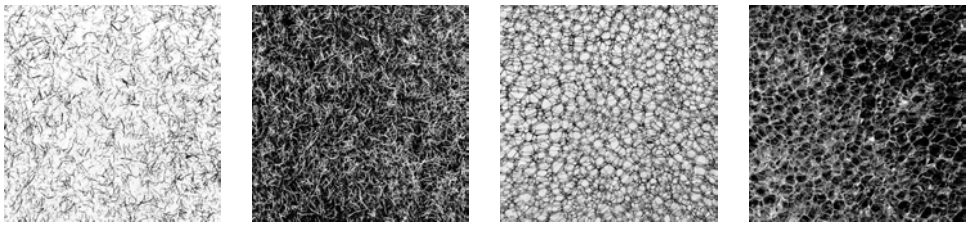


Figure A.5: Brodatz Album images shown as thumbnails, D085–D108.



D109 D110 D111 D112

Figure A.6: Brodatz Album images shown as thumbnails, D109–D112.

APPENDIX B

CUReT Image Database

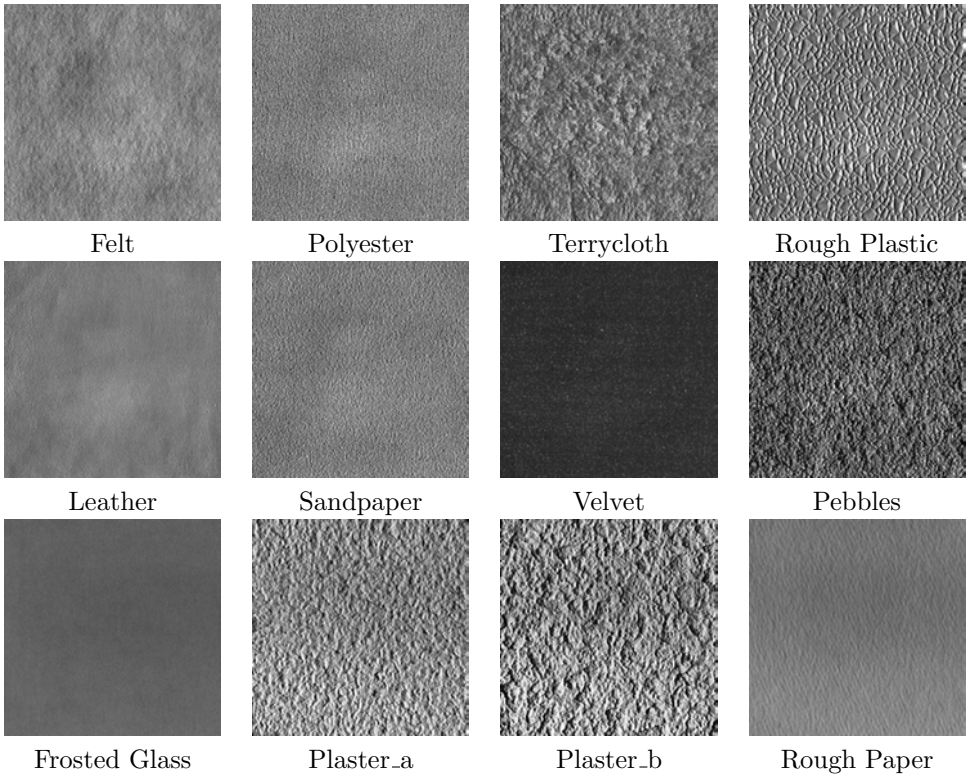


Figure B.1: CUReT image database thumbnails, 1–12.

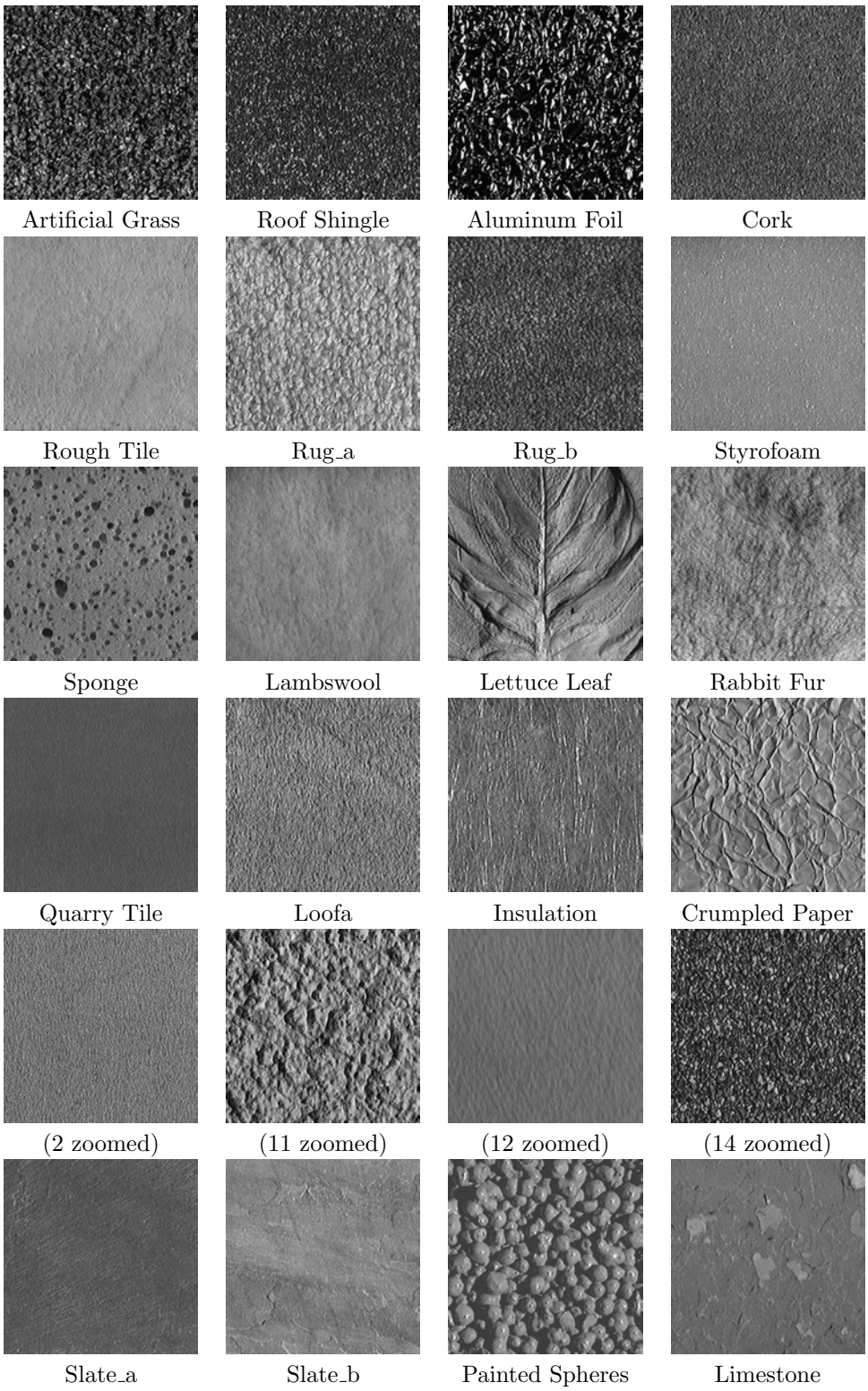


Figure B.2: CUReT image database thumbnails, 13–36.

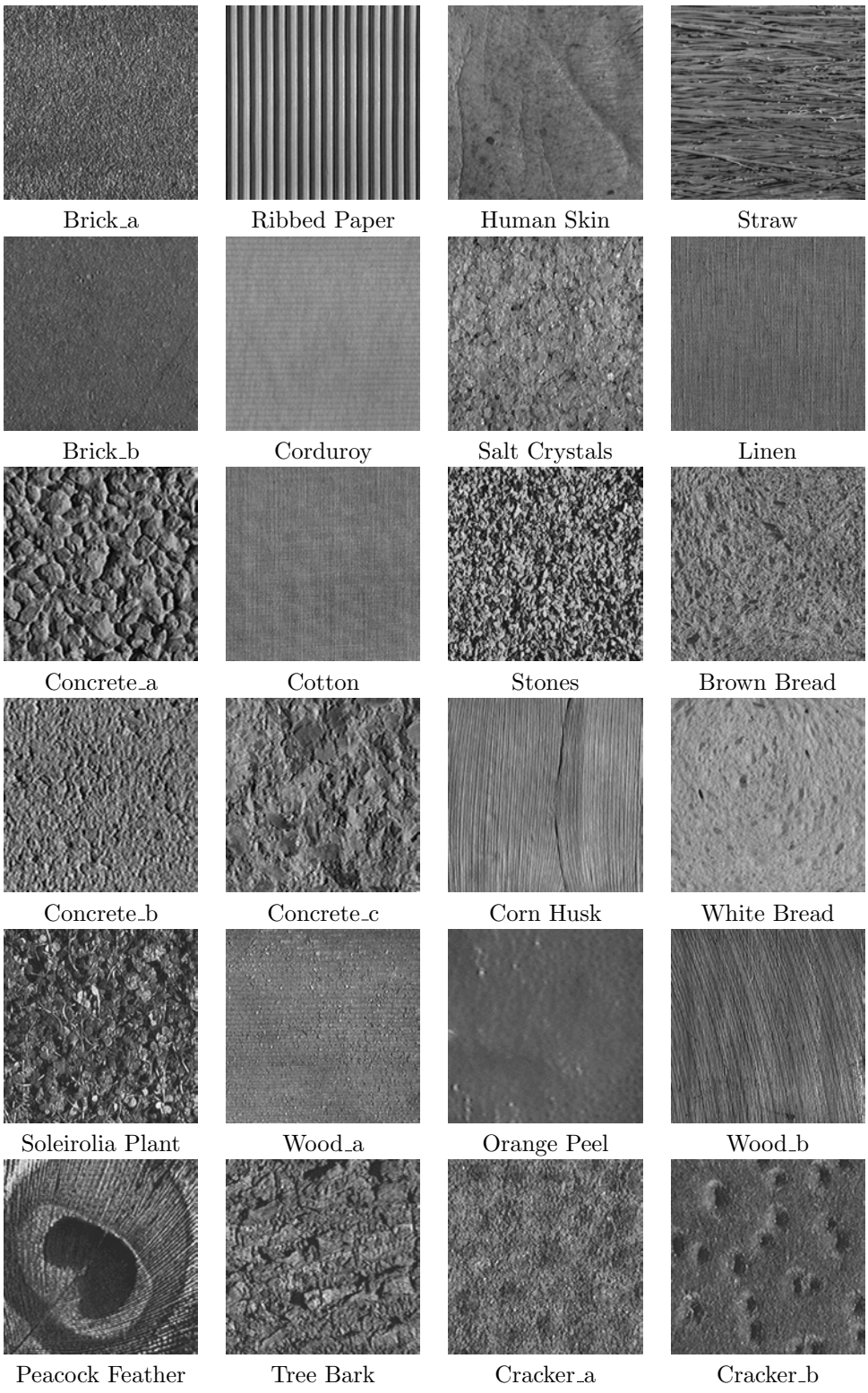
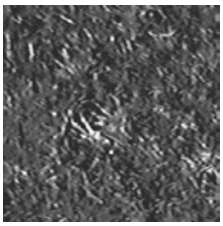


Figure B.3: CUReT image database thumbnails, 37–60.



Moss

Figure B.4: CUReT image database thumbnails, 61.

APPENDIX C

PhoTex Image Database

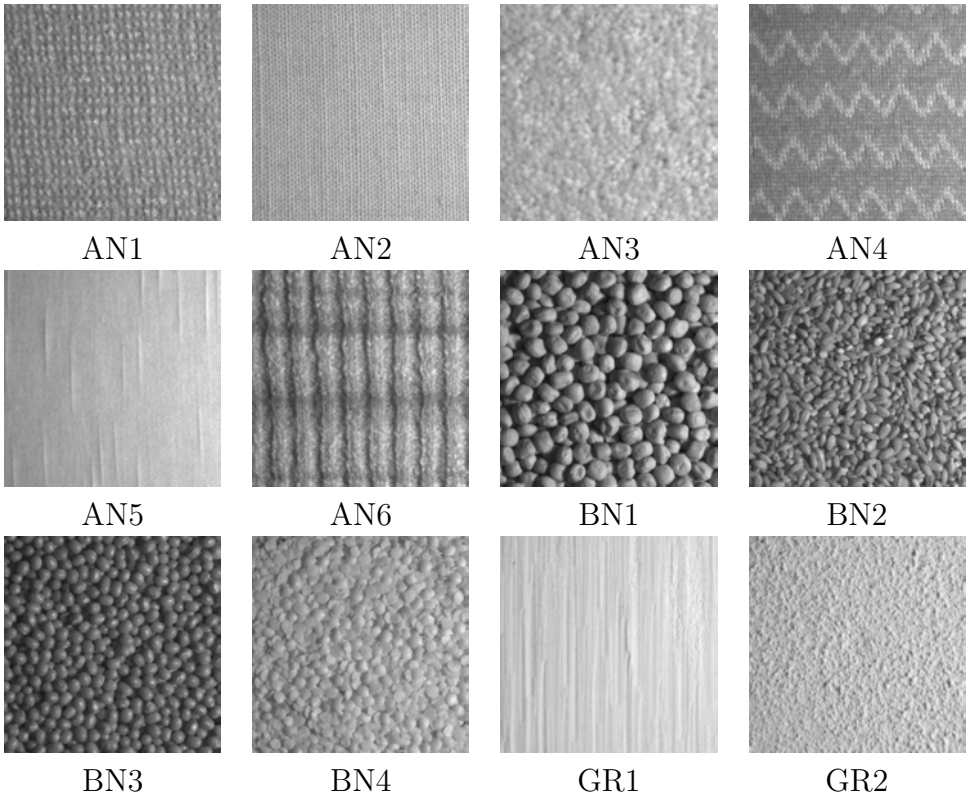


Figure C.1: PhoTex image database thumbnails, 1–12.

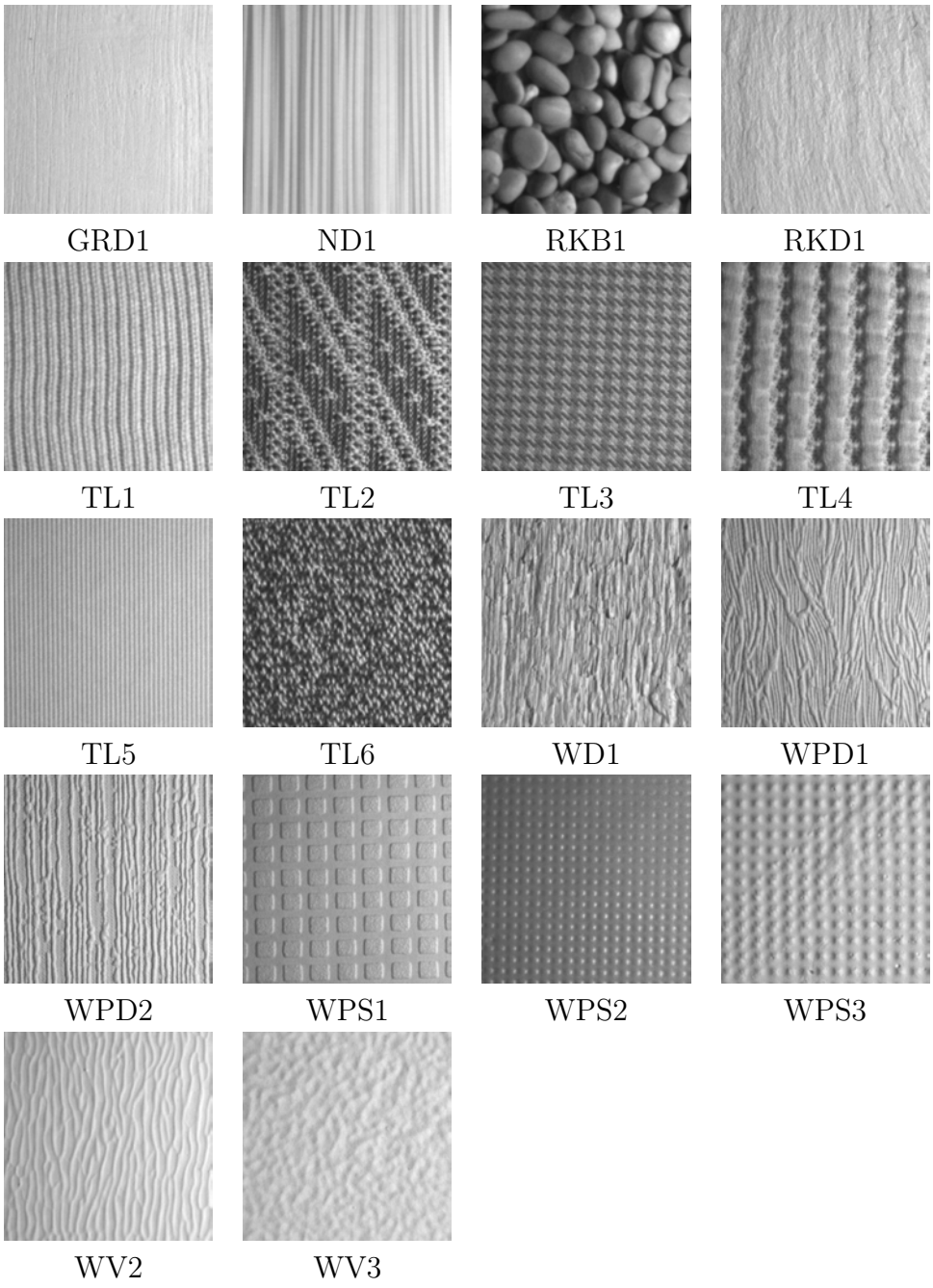


Figure C.2: PhoTex image database thumbnails, 13–30.

APPENDIX D

VisTex Image Database

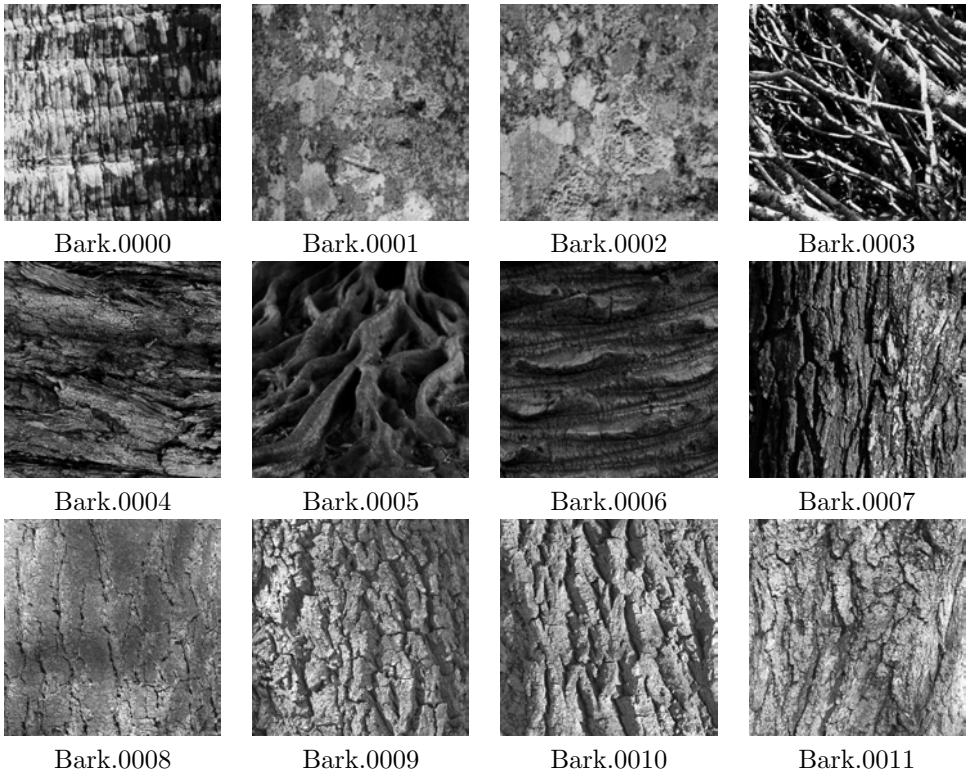


Figure D.1: VisTex image database thumbnails, 1–12.

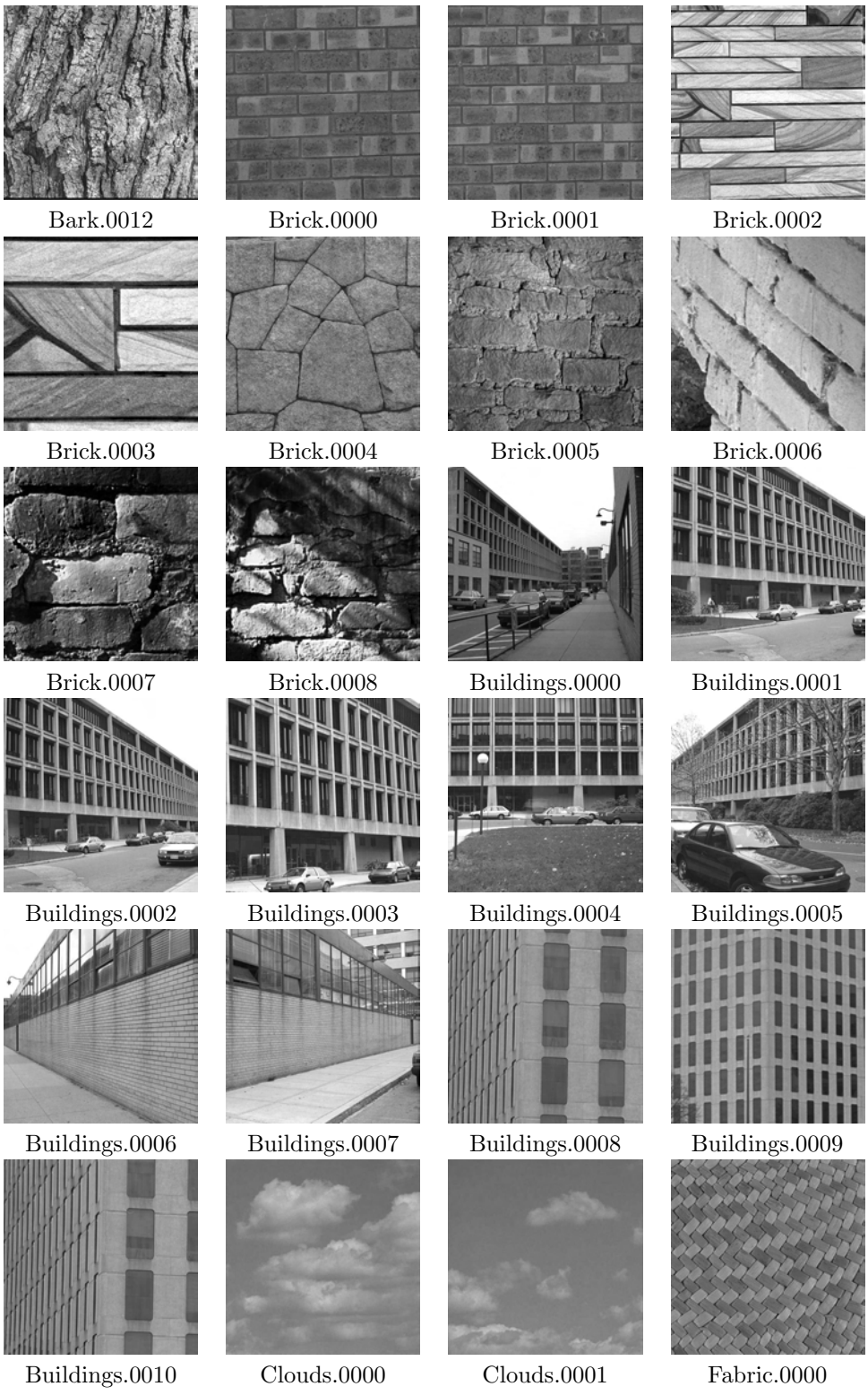


Figure D.2: VisTex image database thumbnails, 13–36.

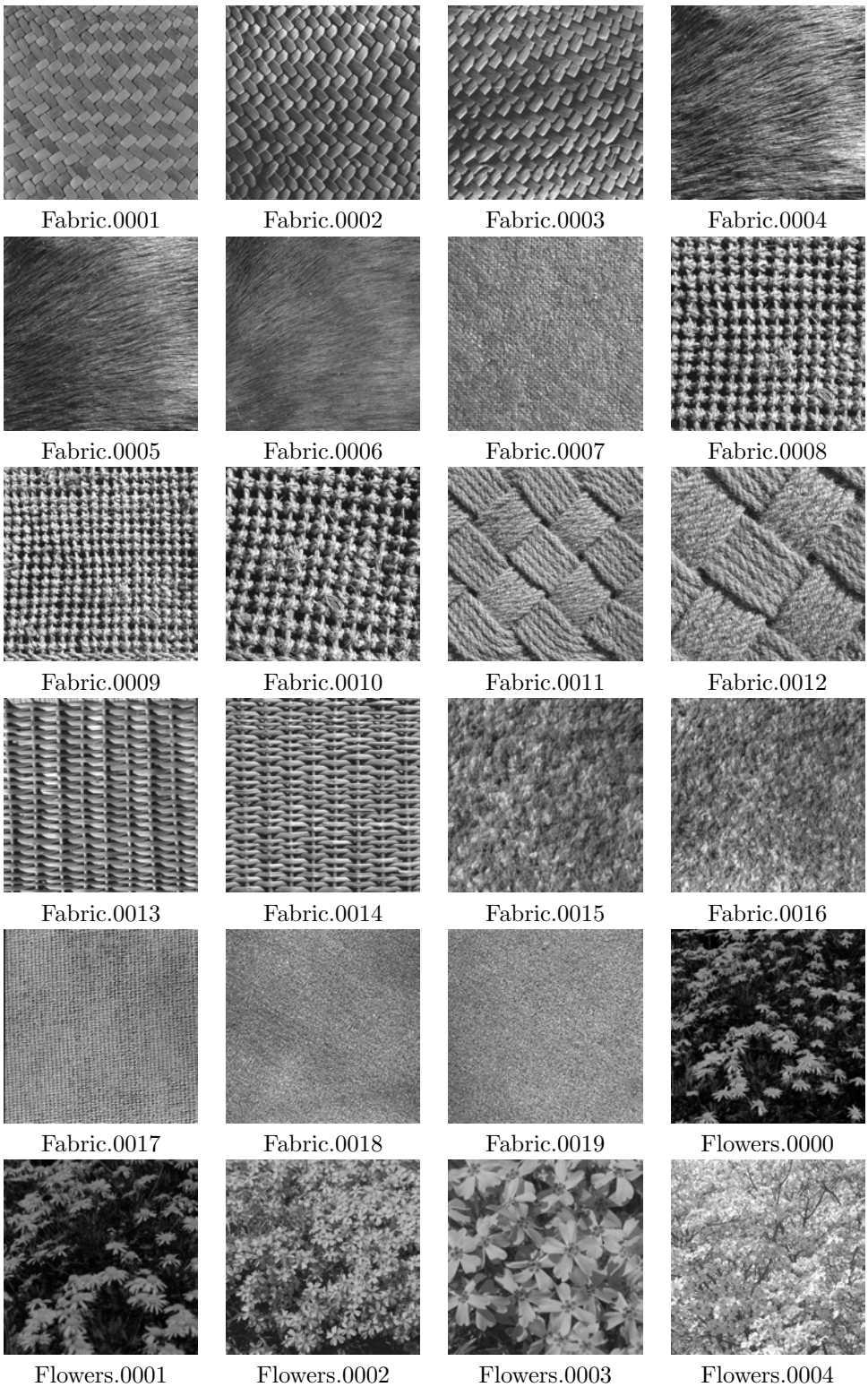


Figure D.3: VisTex image database thumbnails, 37–60.

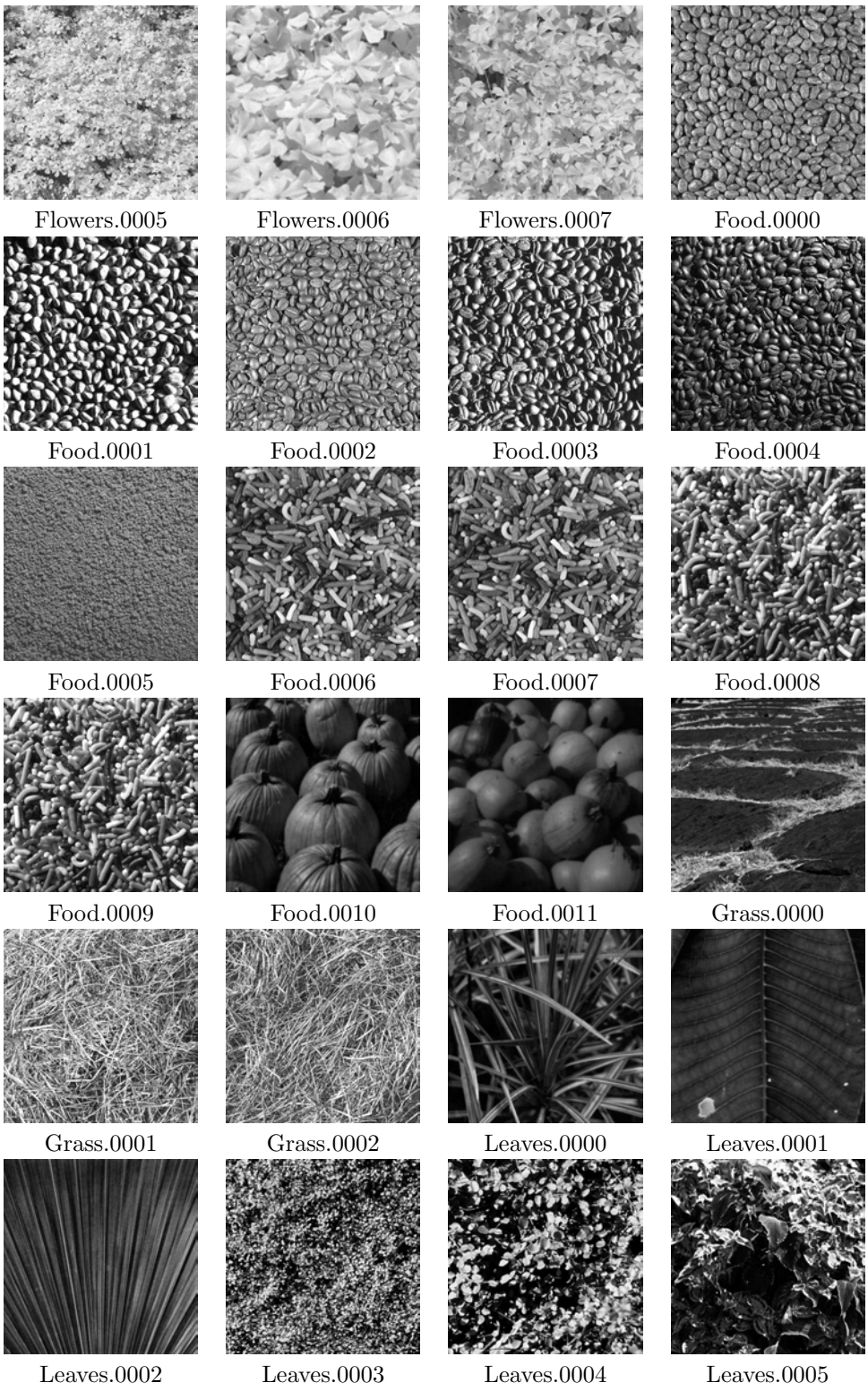


Figure D.4: VisTex image database thumbnails, 61–84.

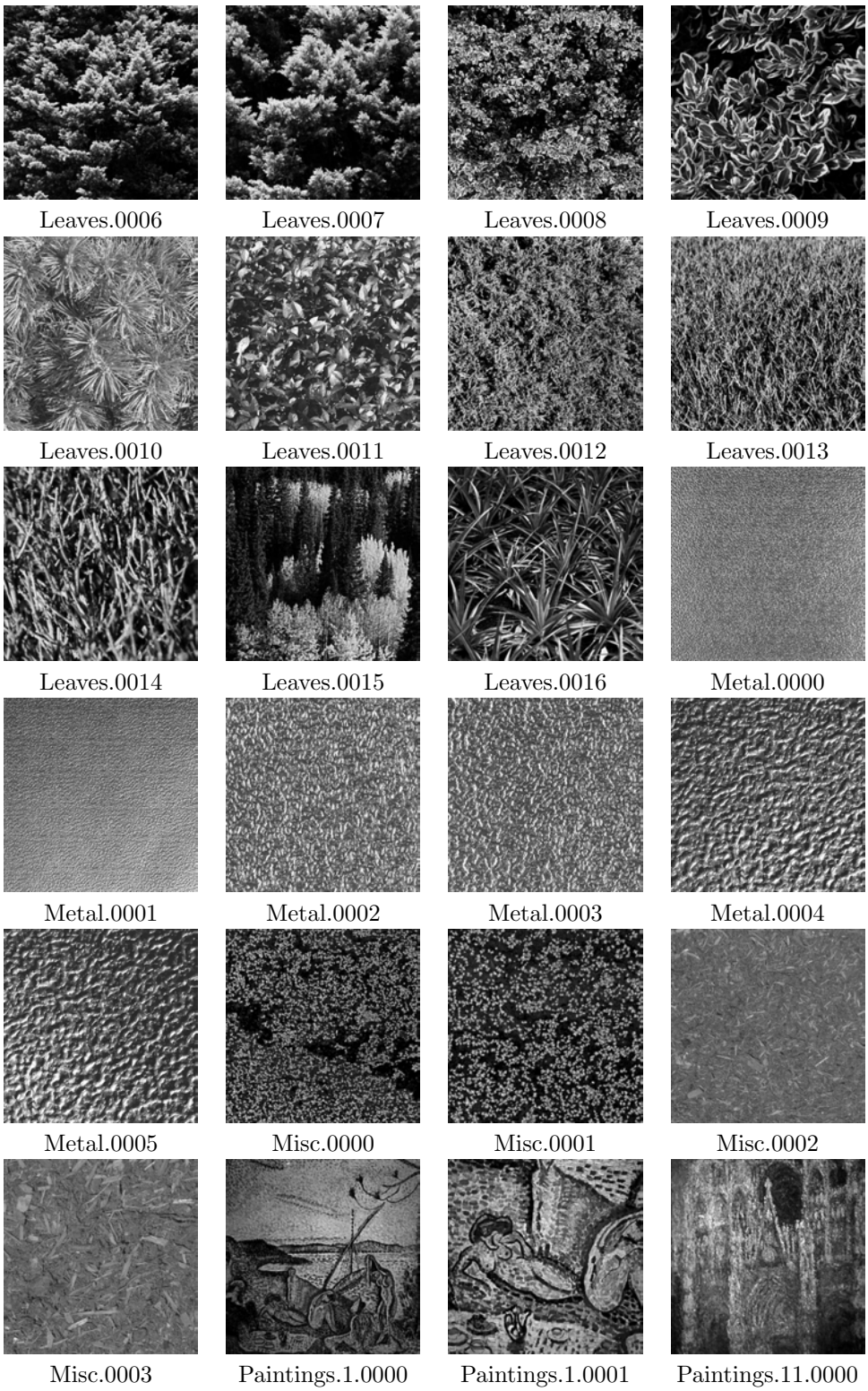


Figure D.5: VisTex image database thumbnails, 85–108.

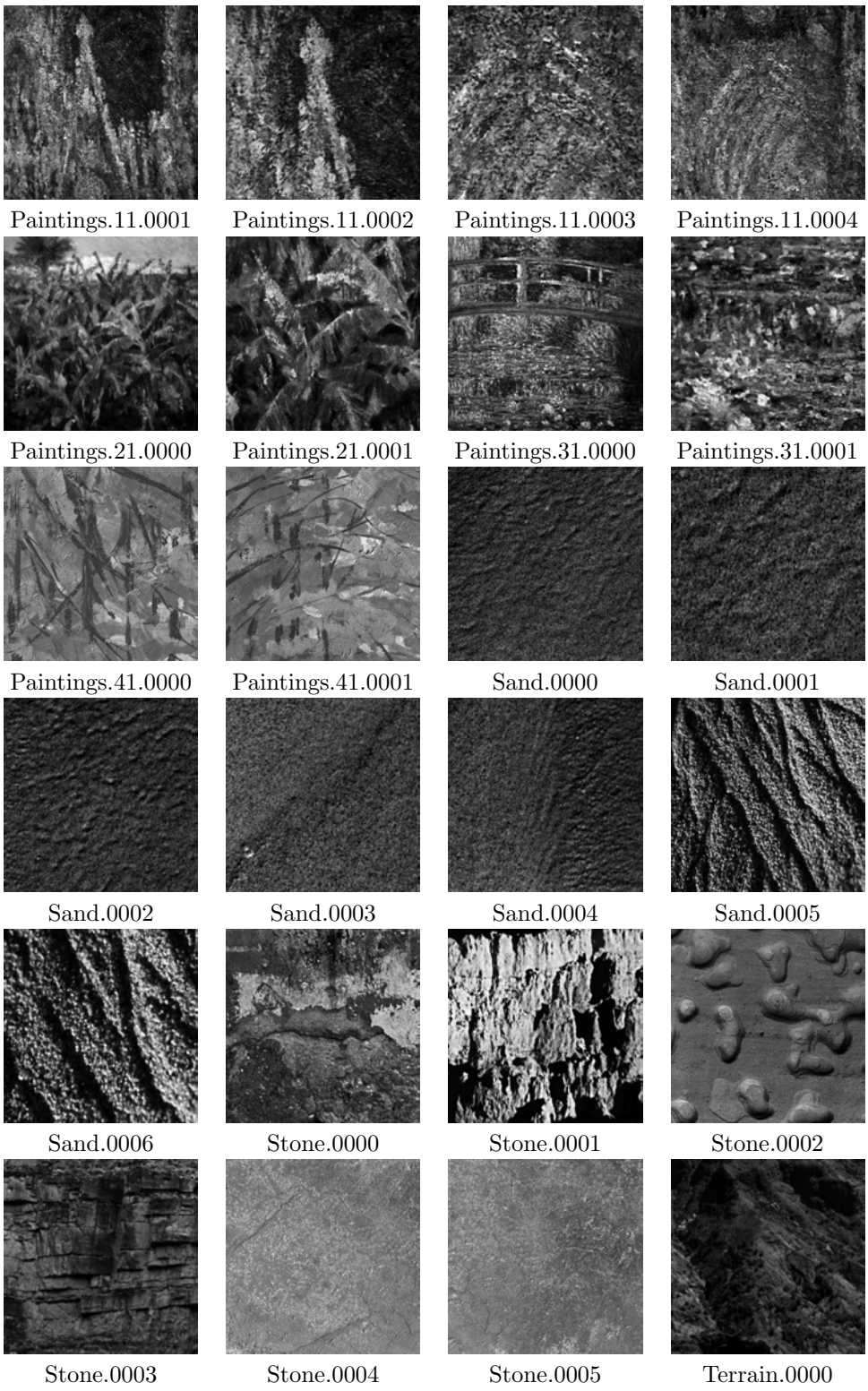


Figure D.6: VisTex image database thumbnails, 109–132.

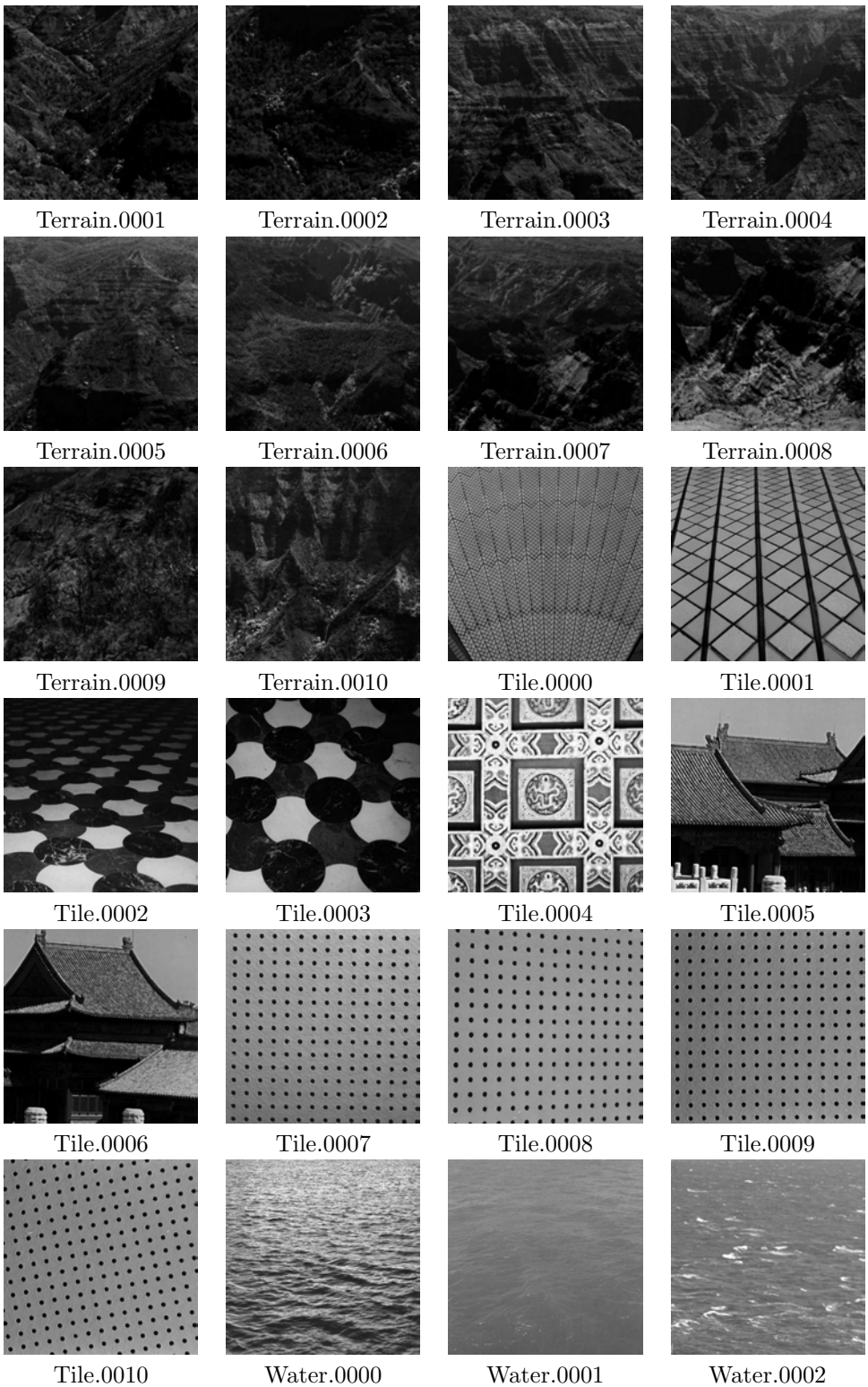


Figure D.7: VisTex image database thumbnails, 133–156.

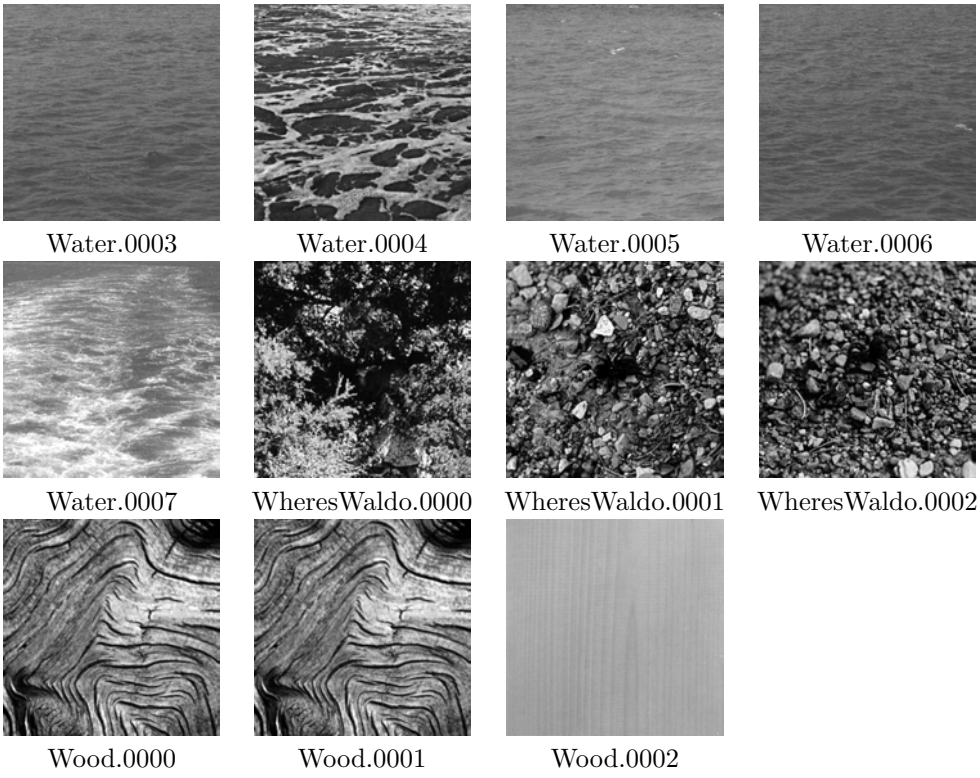


Figure D.8: VisTex image database thumbnails, 157–167.

APPENDIX E

Image Retrieval Examples on Brodatz Album

In this chapter, we select six texture subimages, namely, D001_1, D004_1, D018_1, D035_1, D052_1, and D101_1 from Brodatz Album and their most similar 30 subimages are retrieved by using both SASI and Gabor filter descriptor.

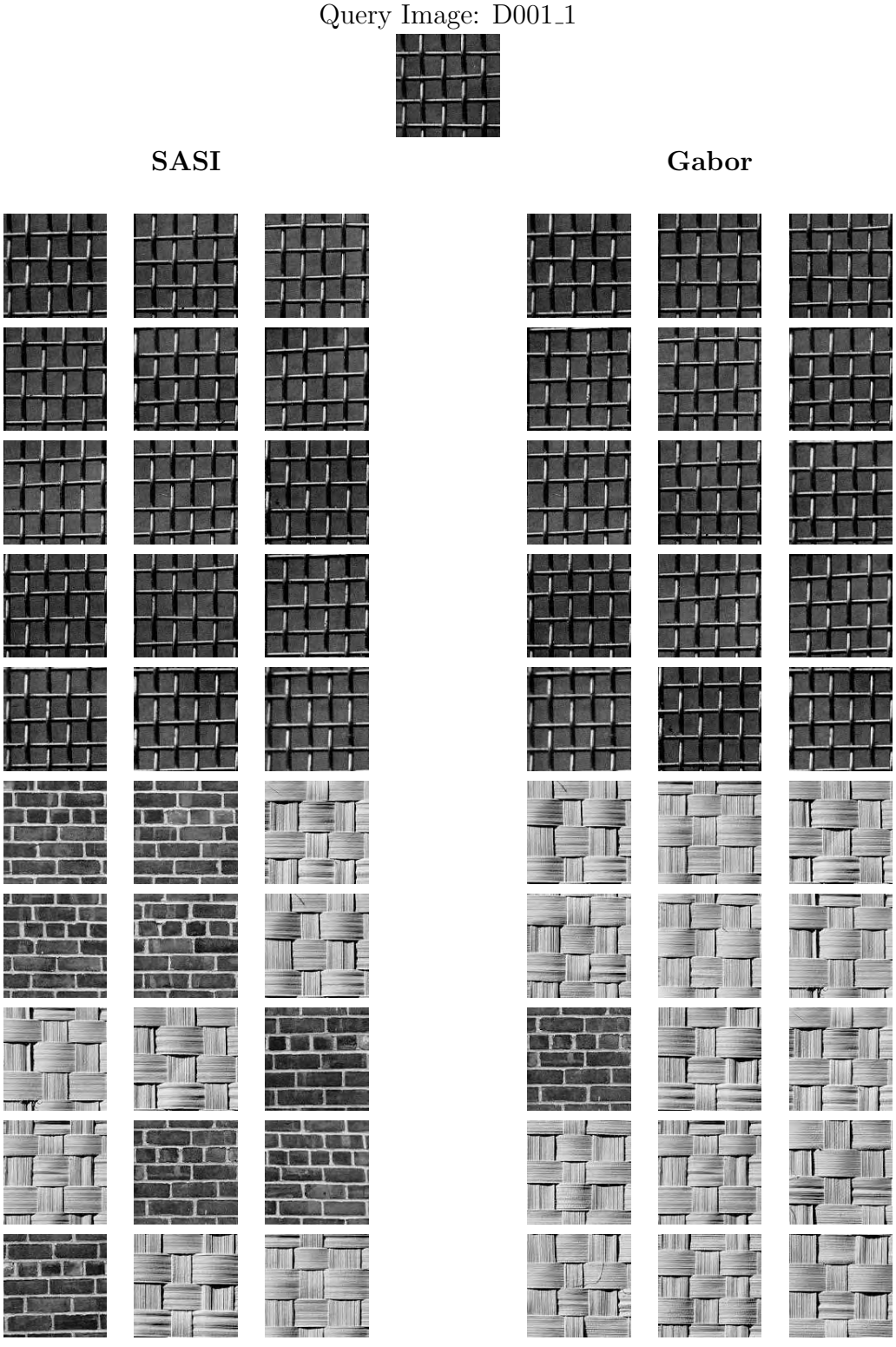


Figure E.1: The most similar 30 images of a query image D001_1 are depicted. Images are ordered by the distance from left to right, top to bottom (excluding self matches).

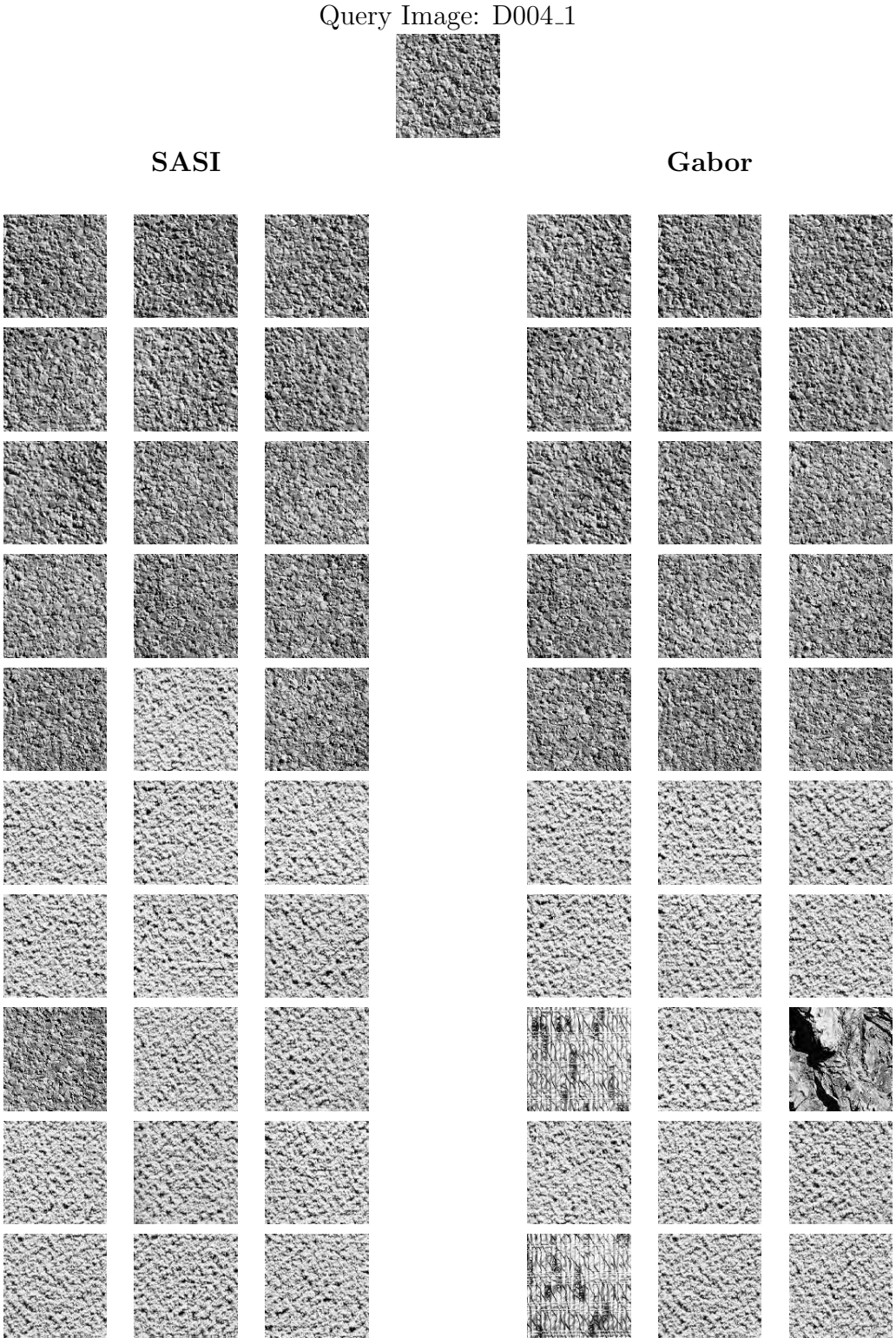


Figure E.2: The most similar 30 images of a given query D004_1.

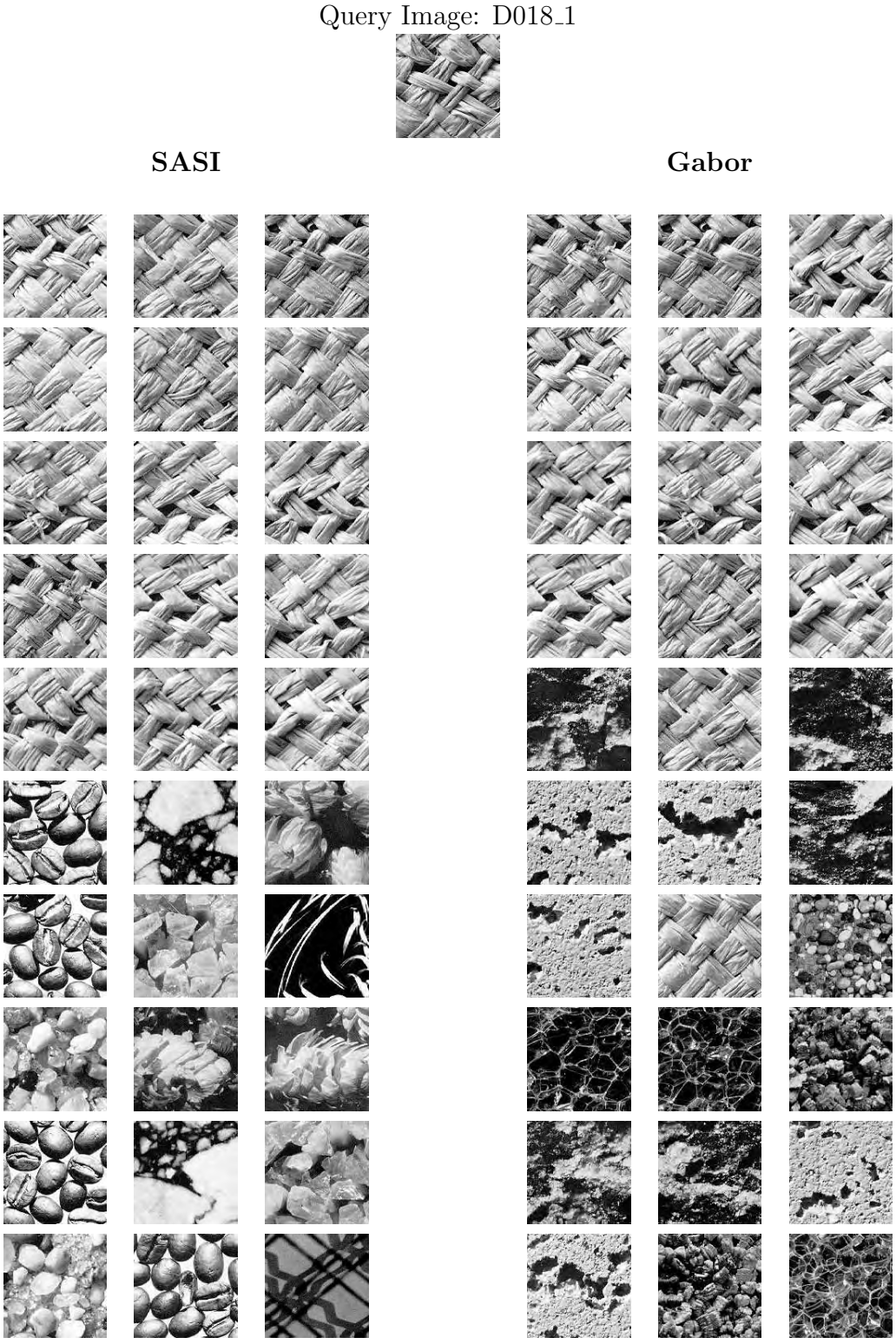


Figure E.3: The most similar 30 images of a given query D018_1.

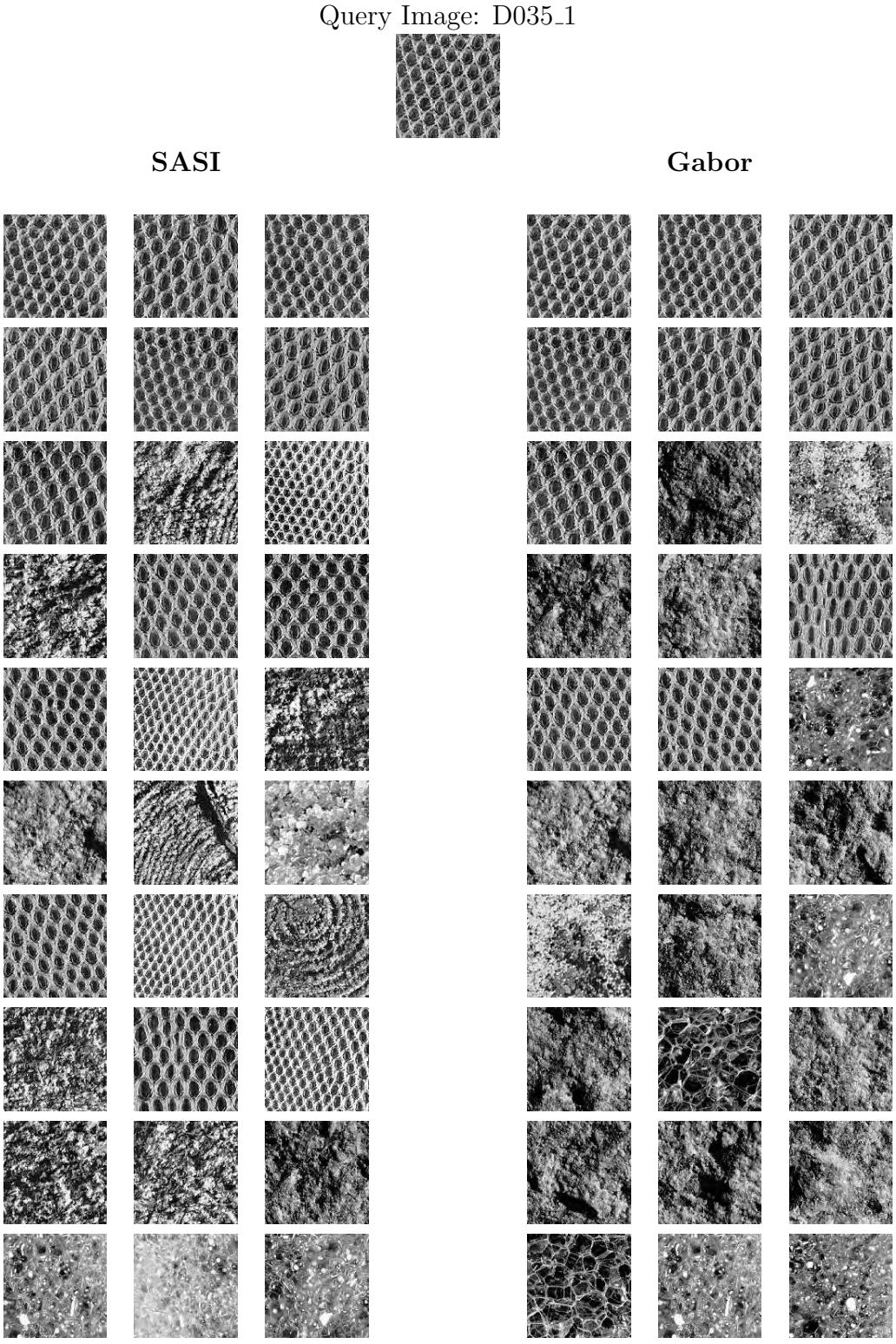


Figure E.4: The most similar 30 images of a given query D035_1.

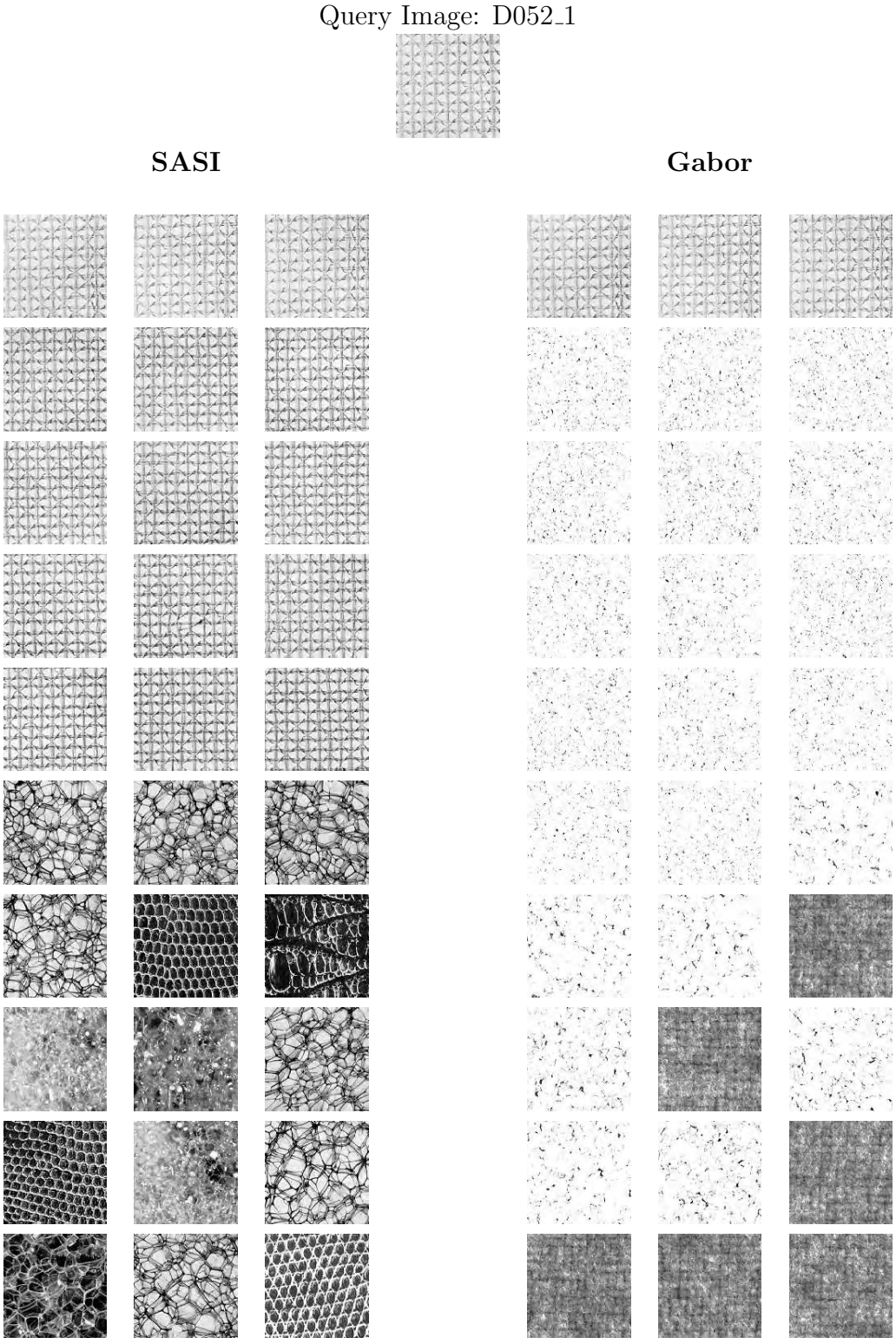


Figure E.5: The most similar 30 images of a given query D052_1.

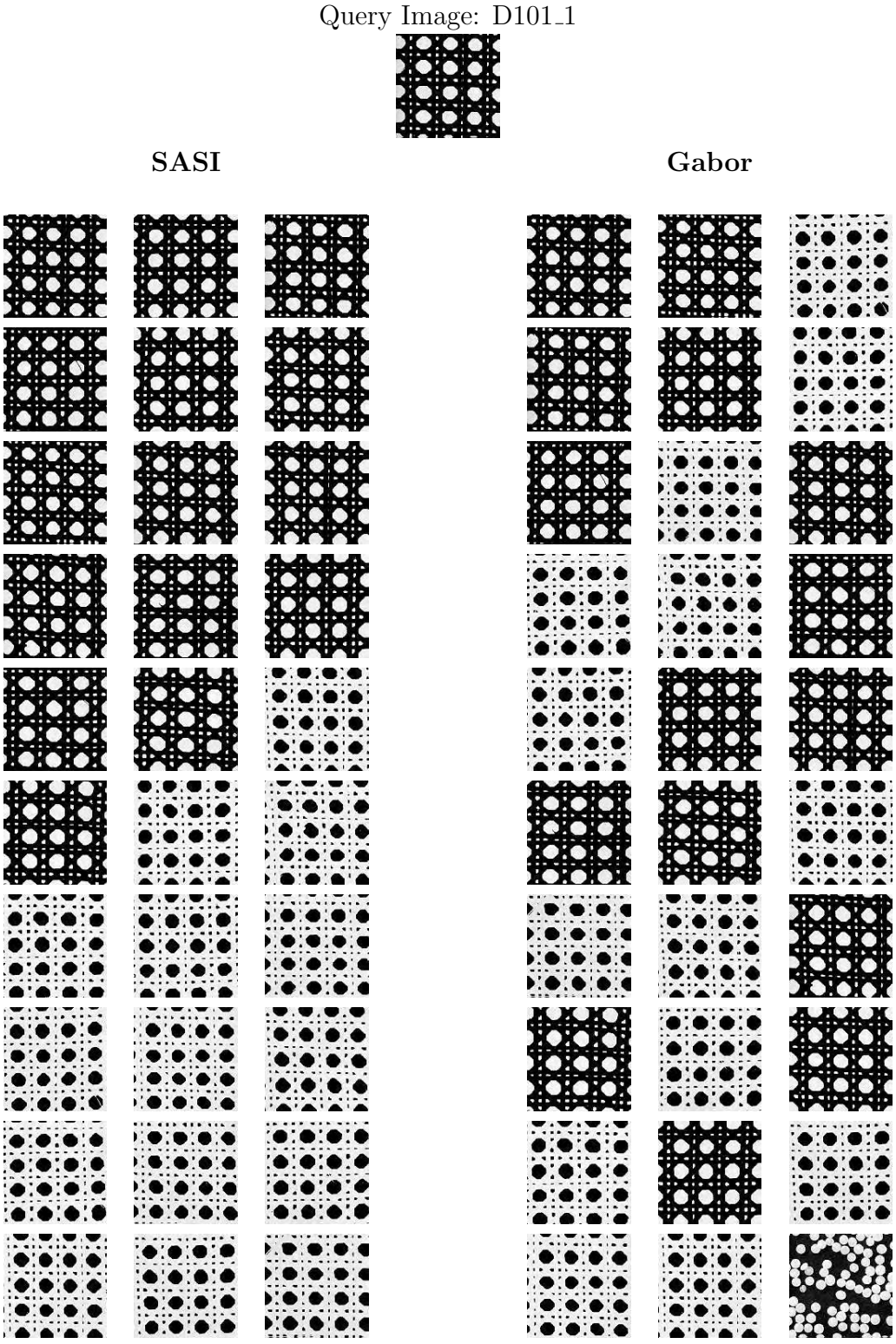


Figure E.6: The most similar 30 images of a given query D101_1.

APPENDIX F

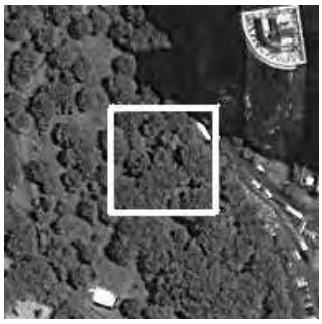
Image Retrieval Examples on Satellite Images

In this chapter, we test SASI descriptor on satellite images. Basically, a satellite image is partitioned into 64×64 blocks of pixels. Then, for each block, a SASI (in η^3) texture feature vector is computed as explained in Section 5.2.1. Finally, the most similar 9 images of a given query image are retrieved.

F.1 A Satellite Image of Manhattan, New York, USA

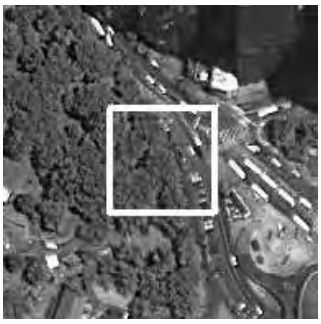
Figure F.1: A one-meter resolution satellite image of Manhattan with size 2688×2496. After partitioning the image into 64×64 blocks of pixels, 1638 blocks are obtained.

Query Pattern



#1

#2



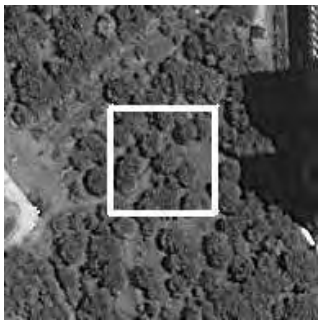
#3

#4

#5

#6

#7



#8

#9

Figure F.2: A sample query image and its most similar 9 images in the satellite image of Manhattan.

Query Pattern

#1

#2

#3

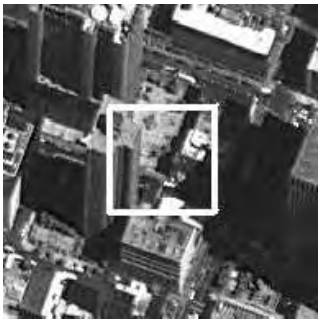
#4

#5

#6

#7

#8



#9

Figure F.3: A sample query image and its most similar 9 images in the satellite image of Manhattan.

F.2 A Satellite Image of Clevland, Ohio, USA

Figure F.4: A five-meter resolution satellite image of Clevland with size 1280×960. After partitioning the image into 64×64 blocks of pixels, 300 blocks are obtained.

Query Pattern

#1

#2

#3

#4

#5

#6

#7

#8

#9

Figure F.5: A sample query image and its most similar 9 images in the satellite image of Clevland.

Query Pattern

#1

#2

#3

#4

#5

#6

#7

#8

#9

Figure F.6: A sample query image and its most similar 9 images in the satellite image of Clevland.

F.3 A Satellite Image of Baghdad, Iraq

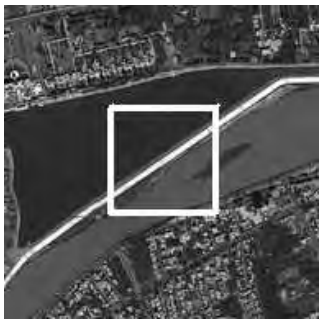
Figure F.7: A 0.6-meter resolution satellite image of Baghdad with size 2176×2176. After partitioning the image into 64×64 blocks of pixels, 1156 blocks are obtained.

Query Pattern

#1

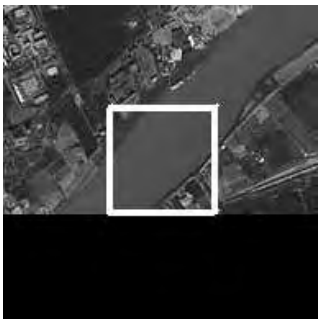
#2

#3



#4

#5



#6

#7

#8

#9

Figure F.8: A sample query image and its most similar 9 images in the satellite image of Baghdad.

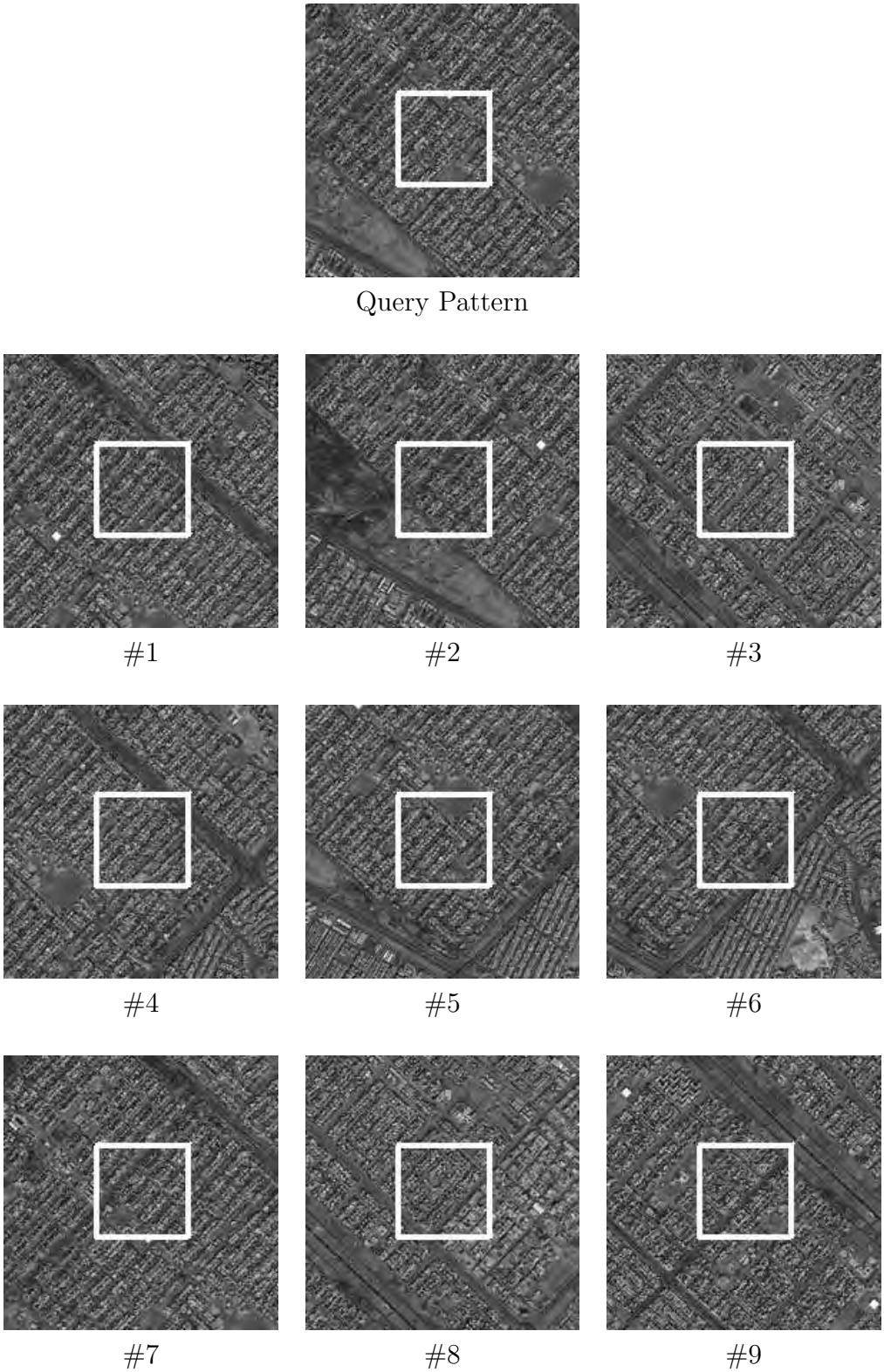


Figure F.9: A sample query image and its most similar 9 images in the satellite image of Baghdad.

VITA

ABDURRAHMAN ÇARKACIOĞLU received the BSc degree in Computer Science and Engineering from Hacettepe University in 1993, MSc degree in Computer Engineering from Middle East Technical University (METU) in 1997. He has been also working as a system analyst in Capital Markets Board of Turkey since 1995. His research interests include content based image retrieval, texture retrieval and representation, pattern recognition, machine learning and analysis of stock market movements.

Publications

- SASI: A Generic Texture Descriptor for Image Retrieval. Abdurrahman Çarkacıoğlu and Fatoş Yarman-Vural. *Pattern Recognition*, 36(11):2615–2633, 2003.

- Learning Similarity Space. Abdurrahman Çarkacıoğlu and Fatoş Yarman-Vural. *International Conference on Image Processing* (ICIP02), September 2002, Rochester, NY, USA.

- SASI: A New Texture Descriptor for Image Retrieval. Abdurrahman Çarkacıoğlu and Fatoş Yarman-Vural. *International Conference on Image Processing* (ICIP01), October 2001, Thesselenoki, Greece.

- Multi-Level Object Description: Color or Texture. Pınar Duygulu, Abdurrahman Çarkacıoğlu and Fatoş Yarman-Vural. *First IEEE Balkan Conference on Signal Processing, Communications, Circuits, and Systems*, June

1-3, 2000, Istanbul, Turkey.

- Ottoman Archives in Image Database. Abdurrahman Çarkacıoğlu and Fatoş Yarman-Vural. *SPIE, Electronic Imaging Working Group Newsletter*, January 1998 (Invited).

- A Set of Similarity Measures for Binary and Gray Level Markov Random Fields Textures. Abdurrahman Çarkacıoğlu and Fatoş Yarman-Vural. *IAPR - ICIAP Lecture Notes on Computer Science*, Vol. 1310, pp. 127–134, 1997, ISSN 0302-9743.

- Set of Texture Similarity Measures. Abdurrahman Çarkacıoğlu and Fatoş Yarman-Vural. *SPIE Electronic Imaging 97, Machine Vision Applications in Industrial Inspection*, Vol. 3029, pp. 118–127, 1997, San Jose, USA.

- Texture Similarity Measures for Markov Random Fields Model. Abdurrahman Çarkacıoğlu and Fatoş Yarman-Vural. *International Symposium on Computer and Information Systems IX* (ISCIS), vol.1, pp. 63–72, 1996, Antalya, Turkey.