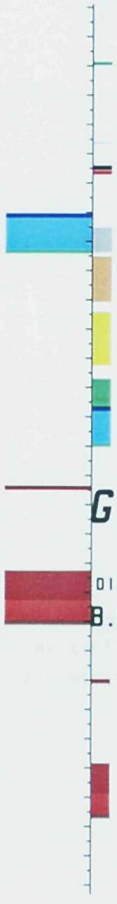

GIVING EYES TO ICT!

OR: HOW DOES A COMPUTER RECOGNIZE A COW?

DISSERTATION/ACADEMISCH PROEFSCHRIFT

B.A.M. Schouten

GIVING EYES TO ICT !

or

HOW DOES A COMPUTER RECOGNIZE A COW ?

ACADEMISCH PROEFSCHRIFT

ter verkrijging van de graad van doctor aan de Universiteit van Amsterdam op gezag van de Rector Magnificus Prof. Dr. J.J.M. Franse ten overstaan van een door het college voor promoties ingestelde commissie, in het openbaar te verdedigen in de Aula der Universiteit op

vrijdag 23 maart 2001, te 12.00 uur

door
Bernardus Antonius Maria Schouten
geboren te Amsterdam

Promotor

Prof. Dr. M.S. Keane	Universiteit van Amsterdam

Leden van de promotiecommissie

Prof. Dr. M.I Brin	University of Maryland
Dr. F. van der Ploeg	Staatssecretaris voor Cultuur, The Netherlands
Prof. Dr. D. Saupe	Universität Leipzig
Prof. Dr. A.M.W. Smeulders	Universiteit van Amsterdam
Prof. Dr. S.J. van Strien	Universiteit van Amsterdam

Faculteit Natuurwetenschappen, Wiskunde en Informatica.

Het onderzoek dat tot dit proefschrift heeft geleid is uitgevoerd op het Centrum voor Wiskunde en Informatica (CWI) in Amsterdam en werd mede mogelijk gemaakt door de Nederlandse Organisatie voor Wetenschappelijk Onderzoek (NWO).

Contents

I Overview 1

1 Introduction 3

1.1 How does a Computer Recognize a Cow ? 3

1.2 E-content 4

1.2.1 Frustration 5

1.3 Outline of the thesis 6

2 Information Retrieval Systems 11

2.1 Relational Database 11

2.2 First Generation Visual Information Retrieval Systems 12

2.3 Second Generation Visual Information Retrieval Systems 12

2.3.1 Visual Content 13

2.3.2 Examples 14

3 Features and Content Extraction 19

3.1 Examples of Features 19

3.1.1 Color 20

3.1.2 Texture 21

3.2 Questions, Questions, and more Questions. 22

4 Latest Developments and Research Proposals 29

4.1 Visual Intelligence 29

4.1.1 Reasoning and Interpretation 29
4.1.2 Pariss 30
4.2 (Research) Proposals 31
Bibliography 37

II Mathematical Background **41**

5 Fractal Geometry for Feature Extraction **43**
5.1 Ways to Create Fractals 44
5.1.1 Hausdorff Metric 45
5.1.2 Iterated Function Systems 45
5.2 Fractal Dimension 46
5.3 Partioned Iterated Function System (PIFS) 47
5.3.1 Collage Theorem 47
5.4 Fractal Image Compression (FIC) 48
5.5 Iterated.com: "Barnsley goes commercial" 49

6 FracFeat and other Feature Extractors **55**
6.0.1 Decompression for Feature Extraction 56
6.1 FracFeat 56
6.1.1 FracFeat Features 57
Bibliography 65

III Four Publications **67**

7 Feature Extraction using Fractal Codes **69**
7.1 Introduction 70
7.2 Background 70
7.2.1 Fractal Image Coding 70
7.2.2 Quadtrees and Multiresolution 72
7.3 Feature Extraction 73

7.3.1 Textural and Spatial Similarity 74
7.4 Feature Extraction using Fractal Codes 74
7.4.1 Texture 74
7.4.2 Spatial Similarity. 75
7.5 Results 76
7.6 Conclusions and Further Research 80
7.7 Acknowledgments 80
Bibliography 81

8 Fractal Transforms and Feature Invariances 83
8.1 Introduction 83
8.2 Fractal Feature Extraction 85
8.2.1 Fractal Image Coding 85
8.2.2 Features and Invariances 86
8.3 The Features 87
8.3.1 Introduction 87
8.3.2 Description of the Feature-bins 88
8.4 Experiments and Results 89
8.4.1 Introduction 89
8.4.2 Method 89
8.4.3 Test-case 90
8.4.4 Results 91
8.5 Discussion and Future Research 91
8.5.1 Future Research 93
Bibliography 97

9 Image Databases, Scale and Fractal Transforms 99
9.1 Introduction 99
9.2 Fractal Features 100
9.2.1 Features and Invariances 101
9.3 The Features 101

9.3.1 Introduction 101
9.3.2 Description of the Feature-bins 103
9.4 Experiments, Results and Discussion 103
9.4.1 Method 103
9.4.2 Results 105
9.4.3 Discussion 105
Bibliography 109

10 Pariss, an Interface that Learns by Example **111**
10.1 Introduction and Motivation 112
10.2 The Interface's Architecture 114
10.2.1 Introduction 114
10.2.2 Display Windows 114
10.2.3 Computation Engines 116
10.3 Using the Interface: An Interaction Scenario 121
10.4 Addressing the Problem of Partial Relevance 121
Bibliography 126

IV Links **127**

A Clicks **129**
A.1 E-content 129
A.2 Knowledge Map 130
A.3 Standards 130
A.4 Input for Content Based Image Retrieval (CBIR) Systems 130
A.5 Histogram Matching 131
A.6 Textures 131
A.6.1 Example of a QBIC Feature 132
A.7 Similarity Matching and Human Perception 132
A.8 Object Centered Neglect 132
A.9 Washburn 133

A.10 Fractal Dimension [3] 133
A.11 Pseudo-Code 133
A.11.1 Root Mean Square Error 134
A.12 Fractal Feature Extractors 134
A.13 Intelligence 135
A.14 FourEyes 135
A.15 Relevance Feedback: Rui 136
A.16 Browsing within Pariss 136
A.17 Relevance Feedback in Pariss 137
A.18 Other Methods of Content Extraction 137
A.19 Icons 137
A.20 Finding known Objects 138
A.21 Hierarchical Indexing and Searching 138

B Demos **153**
B.1 Zoom Invariance 153
B.2 VisTEX, Database of Homogeneous Textures. 153
B.3 Koe 153
B.4 Demo IFS 154
B.5 Demo FIC 154
B.6 Een Wereld aan ICT Toepassingen 154
B.7 Fractal Imager 154
B.8 Pariss 154
Bibliography 155

Samenvatting **157**

Dankwoord **159**

Curriculum Vitae **160**

Part I

Overview

Chapter 1

Introduction

1.1 How does a Computer Recognize a Cow ?

If computers can be seen as calculators, then the question arises whether intelligence and particularly visual intelligence can be produced by mere calculations. Unfortunately, this question will not be answered in this thesis. One thing is certain; we have definitely taken the road to visual intelligence in the future. We may wonder where this road will bring us and where we are now. *Image recognition* now becomes available in industrial products. In daily life we are, sometimes without being aware, already confronted with these techniques. If you are speeding for instance, the text of your license plate may be recognized.

Recognition is definitely a part of visual intelligence. It relates an emotion, experience or visual input to an earlier event. Image recognition is based on having seen something before. Our brain is very effective in this. One could say: "Once you have seen one cow, you have seen them all". Human beings are able to understand what makes a cow a cow and to work with *concepts*. Computers are far less intelligent. They cannot enjoy the variety of cows.

To a certain extent concepts can be expressed in language. Language makes it possible to describe these concepts and to explain what we see. Although it is a tedious task, someone escorting a blind person can describe the surroundings to him. Even harder it is for the blind person to imagine what has been described. The mapping from image space to language is not one-to one, after all: *"A picture is worth a thousand words"*. The same problem is encountered in image recognition by computers. In contemporary multimedia applications, an image is described by features, like the color or shape of an object. This is a many to one mapping and as a consequence, there is no one-to-one reverse mapping. A car can be red. But there are a lot of other objects with a red color.

I.T. can look but not see.

This thesis is mainly concerned with systems being able to retrieve visual information within the domain of *information technology* (IT). As we can learn from the human seeing-eye dog, one way of doing it is to describe the content in language, as done in the case of keywords. But then a more intelligent way of looking for similarities is required. *Visual information retrieval* (VIR) systems process visual content in a way human beings do.

Content based image retrieval (CBIR) is based on the fact that images can be retrieved because of their similarity to other images. A database is able to return images of cows as a result of the fact that the user has pointed out other cows or drawn an example of a cow.

One can distinguish three core components of these systems:

- **Content extraction.**
  Describe the content in such a way that it can be processed. *Feature extraction* is one way of doing this. An image or video is described according to several features like color or texture.
- **Similarity.**
  Once the content has been described, the system has to define how similar this content is to the content of other images. As a result, one has to define metrics approximating to what extent the different images, represented by their features are similar.
- **Interfaces.**
  For the user to communicate with the system, interfaces should be able to display and compose visual information. As the content of images is subjective, an intelligent system should be able to manage this subjectivity.

This thesis will focus on the first and the last items.

1.2 E-content

As we are living in the age of information technology, with the amount of *E-content* growing exponentially, the need to handle this information grows as fast. We are overwhelmed with information: Click A.1

`http://www.sims.berkeley.edu/how-much-info`

and the ability to retrieve this information becomes proportionally smaller to the growing amount of data. In the domain of *visual information systems*, information can be processed for several purposes:

- **Compression.**
  MPEG 4, 7, 21 add standards for describing content in multimedia databases, besides mere compression.
- **Retrieval.**
  To browse, query and download information from the web or other information databases.
- **Visualization.**
  In the field of fashion and design for instance, fashion depends on personal appreciation. Applications should give way to examine clothing as it appears (*intelligent clothing*) and search for products by visual means.
- **Security and authentication.**
  Document ownership, access and facilitation. Authorizing or blocking content (pornography). Filtering.
- **Delivery on demand.**
  Personal Content as in television on demand. Extracting personal content from a larger and more general content quantity. Indexing.
- **Manipulation.**
  Content being processed in a way that new content is created. Examples are editing music and video, archiving, billing, accounting, virtual shops, etc.
- **Quality control.**
  Visual inspection of products like textiles.

1.2.1 Frustration

Everybody experiences the frustration while exploring and searching the internet and hearing him or herself say:

- *You don't understand me, that is not what I mean!*
- *Don't ask me what I want, I'll know it when I see it!*
- *What I'm looking for, is not there!*
- *Can't you see, that's different?!*

The best cure for this frustration is to act like Picasso. "Don't look for things, just find them", is the best way to handle the growing amount of information, see Figure 1.1.

Content management systems including visual information retrieval is the fastest growing research field and finds its way into many E-commerce solutions. For a few years these

systems operate in the world wide web domain (search engines like QBIC, Altavista) as well as expert systems in, for instance, medical imaging. Please take a look at the video: "Een Wereld aan ICT Toepassingen" from the Dutch Ministry of Economical Affairs in which an application presented in this thesis for CBIR is exhibited. Other promising applications are: Click B.6

- **Art.**
  The user wants to find paintings or objects similar to the one he or she likes.

  `http://www.hermitagemuseum.org.`

- **Photographic databases.**
  A graphical designer may desire to use an application enabling him to search for similar alternatives, based on the image content. He then makes a choice according to his own personal taste.

- **Video library.**
  Full content search and retrieval of current and past TV broadcast. A fully automatic process enables daily content capture, information extraction and storage in on-line archives. As an example: the Informedia Digital Video Library project at Carnegie Melon University,

  `http://www.informedia.cs.cmu.edu.`

- **Criminal investigation.**
  The Los Angeles Police started in 1998 a pilot on the fast recognition of car thieves by a composite drawing. Going through a database of possible suspects can be made more efficient if an automated system could assist in recognizing a sketch made by the victim, see Figure 1.2.

- **Search engines.**
  Well-known examples are:

  `http://www.altavista.com, http://www.google.com.`

1.3 Outline of the thesis

Usually a Ph.D. thesis is written as a book. A subject like visual intelligence asks for a different approach. It has to be visually attractive and intelligent. The interfaces discussed in this thesis are reflected in the way this thesis is presented, a *compact disc* (CD). The CD lets you enjoy the content of this thesis in several modes; it enables the user to navigate through

the documents according to his own needs. It will be interactive and display demos that can be found on the internet and developed within the scope of this thesis.

The textual version consists of four parts. They can be read separately. The first part builds the stage on which the published papers (of the third part) perform; how to browse and query (visual information) and the interfaces communicating with the user. These Chapters are written for a larger audience.

After this introduction, Chapter 2 describes examples of *content based image retrieval* as they can be found in commercial systems and research prototypes. Chapter 3 elaborates on the actual content descriptors (called features) used in these systems. Chapter 4 finalizes part I and sketches research topics as a result of the discussion held in the previous Chapters. Part II describes the mathematical background needed to understand the use of fractal image compression for feature extraction as well as a description of FracFeat, the fractal feature extractor we develloped.

In Part III, four papers are presented. The first three explore the use of *fractal image compression* as a feature extractor on textural aspects and spatial similarity within images, for the purpose of image recognition in multimedia databases.

- B.A.M. Schouten and P.M. de Zeeuw. *Feature Extraction using Fractal Codes.* Proc. of VISUAL 99; Third International Conference on Visual Information Systems. Amsterdam, June 1999, Springer Verlag, pp. 483-492.
- B.A.M. Schouten and P.M. de Zeeuw. *Fractal Transforms and Feature Invariance.* ICPR 2000; Proceedings International Conference on Pattern Recognition, Barcelona 2000, Volume 3, pp. 992–997.
- B.A.M. Schouten and P.M. de Zeeuw. *Image Databases, Scale and Fractal Transforms.* ICIP 2000; Proceedings International Conference on Image Processing, Vancouver 2000.

These above papers resulted in a visual information retrieval system:

- B.A.M. Schouten and P.M. de Zeeuw, FracFeat:

  `http://www.desk.nl/fracfeat/index.html.`

An additional graphical user-interface (GUI) has been build in cooperation with Vadim Botchev (Rostov State University).

The last and fourth paper introduces an interface called PARISS for Content Based Image Retrieval supporting *relevance feedback* methods for improved image search. The interface enables the user to show by a "drag and drop" principle "what he means" by identifying similar images.

- G Caenen, G Frederix, A.A.M Kuijk, E.J Pauwels and B.A.M Schouten: *Show me what you mean! PARISS: A CBIR-interface that learns by example.* Proc. of VISUAL 2000: Fourth International Conference on Visual Information Systems. Lyon, November 2000, pp.257-268.

This paper is written together with G. Caenen ESAT-PSI, K.U. Leuven, Belgium, E. Pauwels and F. Kuijk, Center for Mathematics and Computer Sciences (CWI), G. Frederix, Dept of Mathematics, K.U. Leuven, Belgium. A first version of the interface described in this paper was introduced by Caenen, Pauwels and Frederix and baptized PARISS-XL. The novelty of PARISS-XL is the intuitively transparent manipulation of the information accrued during database exploration. In the contribution presented here we intend to take the interaction paradigm to the next level in attempt to further improve the efficiency of the interface. The promovendus wants to gratefully acknowledge the contribution of Frederix and Caenen, from who the idea for this interface originated, and who rephrased the problem of feature selection in terms of a logistic regression model.

Finally, part IV is an appendix for further reading. On the CD the items can be read interactive by clicking on the words, underlined in the text.

Coming from creative circles, I wanted to write a Ph.D. thesis that is enjoyable for everyone.

So please enjoy !

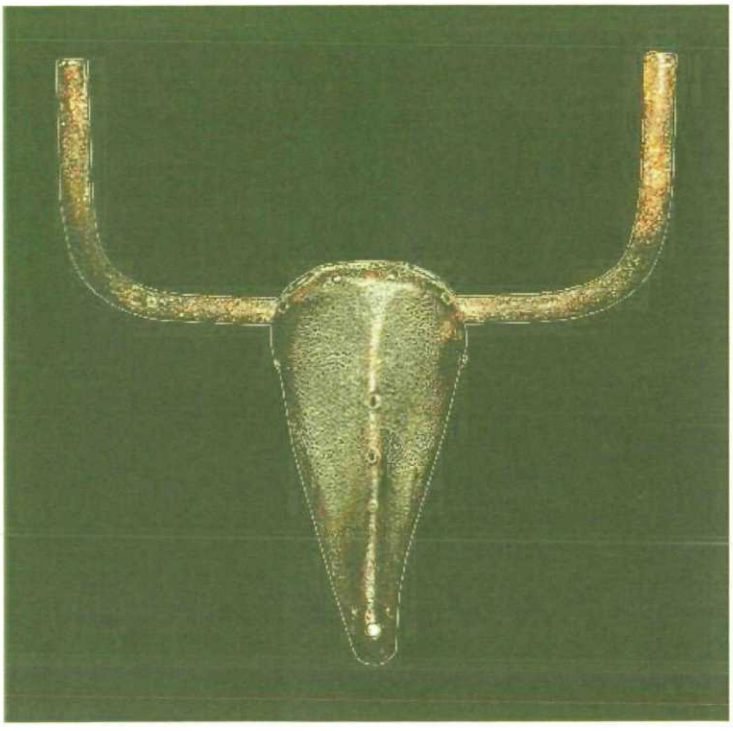

Figure 1.1: Pablo Ruiz Picasso, Tête de taureau. Paris. Spring/1942. Bicycle saddle and handlebars. 33.5 x 43.5 x 19 cm. Musée Picasso, Paris.

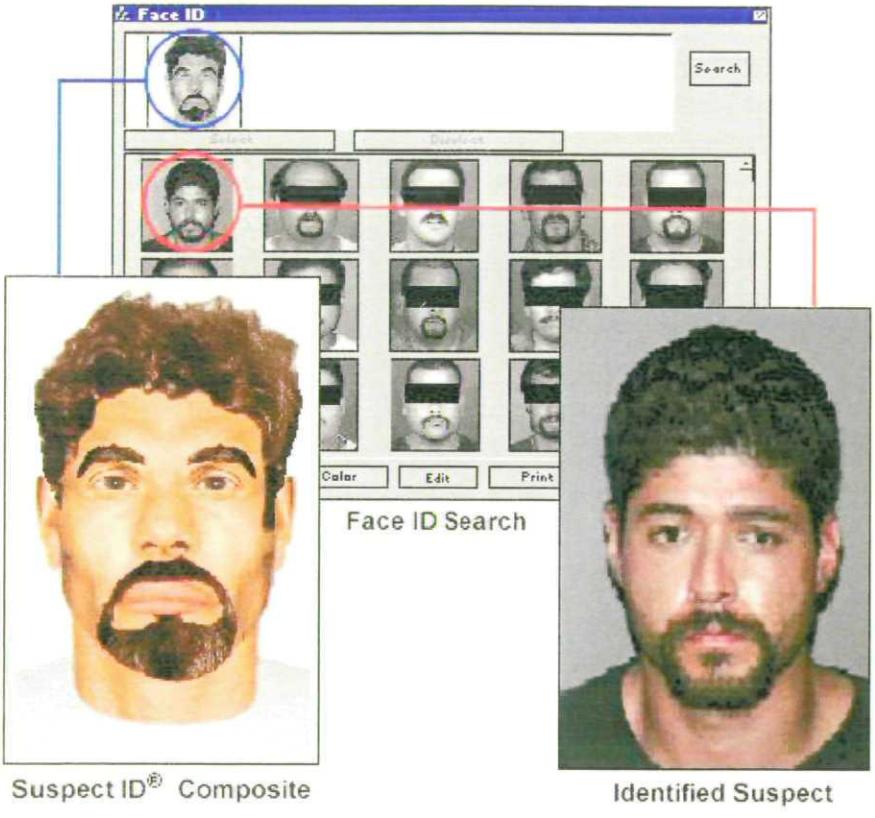

Figure 1.2: Los Ageles Sheriff Department identifies Suspect with Face ID^{TM}.

Chapter 2

Information Retrieval Systems

A text document can be well described by keywords. In the extreme case as many keywords as words in the document are needed to retrieve the document. Together the keywords constitute a description of the content. Visual content is more like poetry. It can have several layered meanings at the same time and is strongly dependent on the appreciation of the user. For this reason it is much more difficult to describe its content. Historically, document retrieval is done by keyword search, where no distinction is made to the nature of the document whether visual or textual.

In the last 10 years, with advancing information technology, image analysis and pattern recognition provided tools to access visual information according to human perception. Visual content descriptors used in contemporary multimedia applications describe the content in low level terms, like the color distribution of an image. These descriptors can be used to define a similarity between documents, such that they can be searched for in database systems.

2.1 Relational Database

In 1978 the ANSI/X3/SPARC Study Group [33] proposed to add an extra dimension to the way *database management systems* (DBMS) function. Until then DBMS envisioned a two level organization: the data as seen by the system (*internal level*) and the data as seen by the user (*external level*). The internal level takes care of the storage of the data. The external level consists of user views that enable the user to communicate with the database and to retrieve the data. A newly created third level, called the *conceptual level*, represents the entities, properties and relationships of the documents stored in the database. In this way relations are used to retrieve information from the database. The introduction of the third level was the starting point of the *relational database*.

2.2 First Generation Visual Information Retrieval Systems

In first generation visual information retrieval (VIR) systems, in order to query a visual or textual document one is invited to provide a keyword which (supposedly) labels the document. This keyword is "processed" in the database and a perfect match at keyword level is aimed for. An expert off-line has supplied the keywords, for a schematic overview see Figure 2.1. The views presented at the external level consist of search engines addressed by traditional query languages like *SQL*.

Standard contemporary systems are able to relate and associate between keywords, in this way these systems build upon the principles of the relational database, see Section 2.1. Semantically more complex queries can be made by Boolean expressions. If the relationship between the keywords is provided by natural language processing, semantically more intelligent queries can be made. Even more advanced systems provide the user with the possibility to browse the database. To search a large variety of multimedia data efficiently, browsing is a desired functionality; it enables the user to have an overview of the data. For browsing, visualization of the data space plays a crucial role [21, 14] (Figure 2.2). Click A.2

The limitations of the first generation systems are:

- The conceptual level is not created by the user but manually by an "expert": the descriptions of the content are made by an annotator. The user may find these choices arbitrary, even inappropriate.
- Content descriptors are created manually. Automatic or semi-automatic content description is desirable.
- Some content can hardly be described in any language e.g. textures or a painting by Picasso.
- The relations between the keywords are also described in the text domain. Concepts like similarity are hard to express in this domain.

As an illustration of these limitations, please have a look at the result of a search action using the word *koe* [1] within the popular search engine *Altavista*. Click B.3

2.3 Second Generation Visual Information Retrieval Systems

Multimedia demands a more sophisticated way of storing and accessing the objects within the database. At the internal level one sees a tendency to use more advanced databases able

[1] Meaning cow in Dutch.

to store complex data consisting of video, images, sound etc. The inclusion of multimedia in a database has a profound impact on its design, features and functions. If a database is only storing multimedia for delivery, like in first generation VIR systems, then a multimedia capable file server coupled with the ability to store pointers, filenames or object identifiers in the database is sufficient.

Utilizing deeper semantic knowledge about the media such as the ability to index, search and relate information, is truly a function which adds value to a database system. It needs to provide support for a range of existing and future data-types, and needs to include support for both temporal and spatial modeling such that data abstraction is maximized [15]. The Monet database at CWI is an example:

```
http://dbs.cwi.nl:8080/cwwwi/owa/cwwwi.print_projects?ID=41.
```

In visual information systems, feature extractors yield descriptions of the visual content. Data independence is important to easily update and change the information contained in a database. The *MPEG* standards provide an example. Click A.3

2.3.1 Visual Content

Metadata describing the visual content of images can be subdivided into [12]:

- **Content independent.**
  Data like author, date, size.
- **Content dependent.**
  - **Perceptual.** Perceptual facts within a video or image. Typically describing aspects like color, texture, shape etc within an image or video.
  - **Semantic.** Abstractions like objects in an image or a scene in a movie.
- **Psychological.**
  Data describing human values and emotions.

Improvements in the second generation retrieval systems are made at the perceptual level. Access to visual information is not merely by keywords but also by using objective measurements of the visual content. Firstly, we attach *features* to images. Usually a feature is some vector in $\mathbb{R}^n$ and the result of the analysis of pixel distributions and numerical discretization of perceptual properties. Secondly, these features are precompiled for all images in the database. In a broad sense features may include keyword based features as well as visual features. For a schematic overview see Figure 2.3.

The user has several methods to provide input for the system, e.g. by sketching or select- Click A.4

ing from image samples. The user selects which features are relevant to the image retrieval process and indicates which similarity measure should be used. The system compares the features of all the images with the query, according to the selected similarity measure and returns an index of images *similar* to the query image provided by the user.

Limitations of the system are:

- **Subjectivity of the features.**
  - Mathematically defined features may not have a clear perceptual meaning or interpretation.
  - Each feature describes just one aspect.
  - Features are driven by availability and not by necessity.
  - High level aspects of an image like objects can not sufficiently be described by (combinations of) low level features like color or texture. It is impossible to describe a beautiful painting of Picasso merely by the colors used.
- **Subjectivity of the user.**
  The content of an image is subjective and strongly dependent on culture, personal taste and different opinions the user can have at different times.
- **Subjectivity of the image.**
  An image may consist of many scenes, see Figure 2.4.

2.3.2 Examples

The goal is to recognize images by their content. Historically these images were manually described by keywords. In early examples of content based image retrieval like Photobook [23] and QBIC[22], one can select an image in the database and ask the system to retrieve images that look similar according to some feature like color, texture, graphical design etc. The user could indicate how strongly the feature should be taken into account when searching for similar images. A sketch interface could be used to indicate and locate different objects in an image.
Similarity is defined on the actual content of the image. Content is derived using image analysis tools. Photobook was one of the first systems for browsing and retrieval of still images by image content. Several subsystems according to the class of images emerged. One of the most well known subsystems is the one dedicated to face-similarity [23].

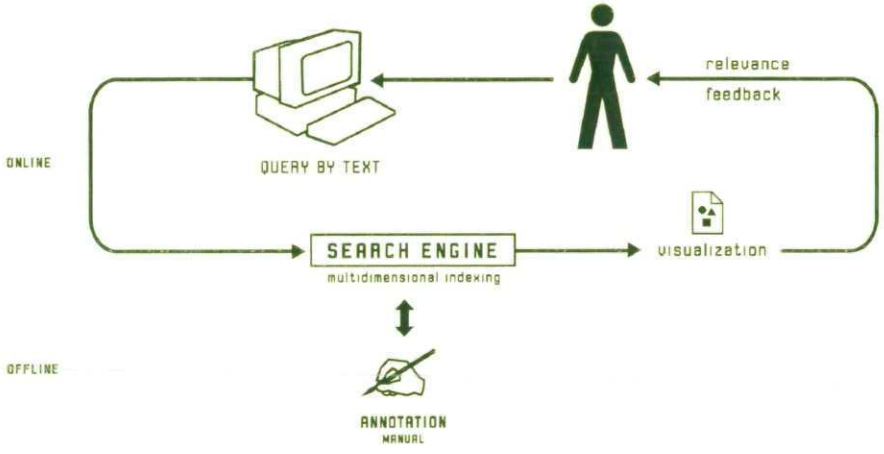

Figure 2.1: Schematic overview of first generation visual information retrieval system.

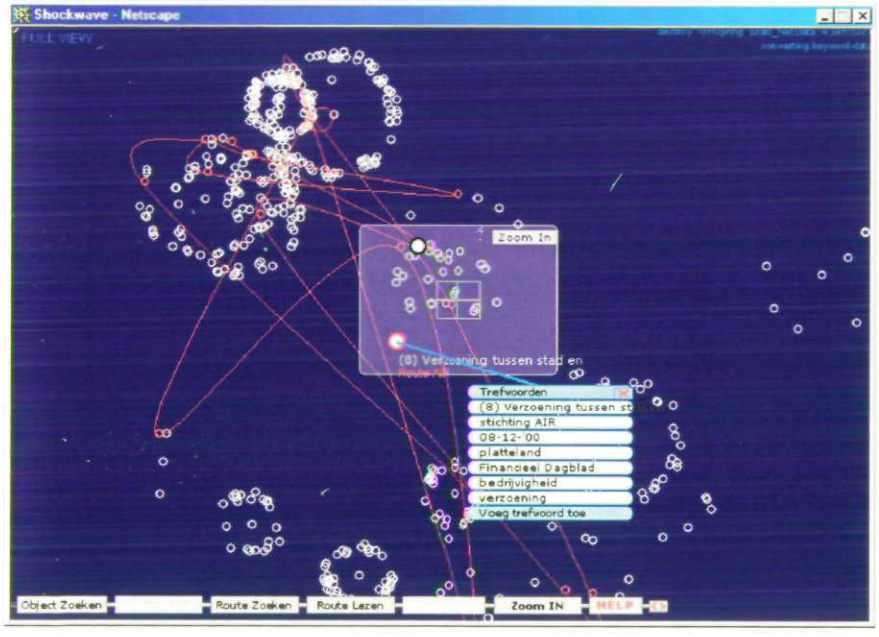

Figure 2.2: A datacloud showing an overview of the data in the database.

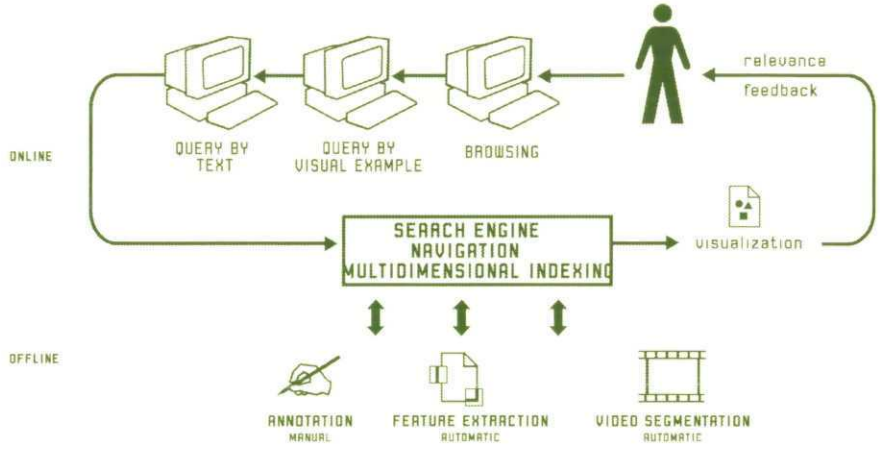

Figure 2.3: Schematic overview of second generation visual information retrieval system.

Figure 2.4: HRH Prince William of Orange milking a cow. The image shown can be appreciated for several reasons: The beautifully made *papier-mâché* cow. The infant. His Royal Highness Prince William of Orange.

Chapter 3

Features and Content Extraction

Interaction with visual content is essential to visual information retrieval. To describe the visual content in multimedia databases low level aspects of an image like color, together with high level concepts like objects are characterized by *features*. Low level features are directly related to perceptual aspects of image content. Several aspects of color, texture or object-shape can be modeled. In order to retrieve similar images within VIR systems these features are compared to the features of the other images present in the database.

A *feature space* consists of all features of all images within a database. If a feature space is endowed with a (similarity) metric, it will be called a *query space* [28]. In most systems features are represented through a numerical vector (n-tuple of numerical values). The query space is therefore modeled as a suitable n-dimensional feature space. Standard mathematical distances like the *Euclidean distance* or the *Minkovsky distance* are used to measure the distance between two points in this feature space. The *similarity measure* can be adapted by giving weights to the importance of certain features in the image (Figure 3.1).

3.1 Examples of Features

Content of multimedia-objects like video, images, sounds are described by several features, directly related to the content of the multimedia object. Features of images can be subdivided into:

- **Perceptual features.**
  Directly observable features like color distribution, textural properties and shape properties.

- **Semantic features.**
  Features related to the concept pictured in the image. Suppose a car has an appearance in an image, seeing it as an object is a quality of the observer related to the concept.
- **Psychological features.**
  Like emotions, appreciation (nice and ugly) and dislike.

A perceptual feature can be modeled more or less independently of the user but often loses its meaning without context. Semantic and psychological features depend heavily on the user. These features cannot be modeled without any knowledge of the context (domain knowledge) wherein the system is used. We confine ourselves to a short description of the most commonly used features like color and texture. For an extensive overview we refer the reader to Bimbo [3] or Huang et al. [11].

3.1.1 Color

Color is one of the most powerful features to describe an image with. The presence and distribution of colors induce sensations and convey meanings to the observer [3]. In the Bauhaus period artist and designers like Itten developed color schemes from a perceptual point of view (Figure 3.2). In his book *Visual Perception*, Tom Cornsweet [6] has made an extensive study.

Aspects of color can be modeled by several color attributes. Usually color stimuli are represented as points in three-dimensional color spaces (channels). There are several ways of expressing color channels, amongst them RGB Space (Figure 3.3). Each color in RGB is expressed as the combination of the primary colors: Red, Green and Blue. This model is used in most hardware (Televisions, Computers and Screens).

To describe low-level color properties of an image, color histograms are used. A color histogram denotes the joint probability of the intensities of the three color channels. A similarity measure is achieved by histogram matching and/or color moments. An aspect important to all features is the robustness to a change in lighting conditions or variations of the image like rotating the canvas or rescaling. Gevers [9] studied the invariances of color spaces to camera viewpoint, orientation and position of the object as well as changes in the color and intensity of the illumination.

Click A.5

As an example we refer the reader to the Blobworld system of the University of California, Berkeley:

`http://elib.cs.berkeley.edu/photos/blobworld.`

In this example, color queries can be processed by region. It will be clear that combining color information to the spatial relationships within an image (regions, objects) will enhance the results.

3.1.2 Texture

Texture is almost as effective in describing an image as color. [2, 5, 13, 16, 19, 20, 25, 22, 11]. Texture can be analyzed from both a mathematical and a psychological point of view. The advantage of a profound mathematical theory and *human perception* join together in using fractal geometry for feature extraction.

In most texture one finds a structural element which is repeated in the image by a placement rule. From a mathematical point of view one can distinguish *statistical* and *structural* texture.
A primitive (building block) is arranged according to a certain placement rule [26, 35]

$$f = R(e) \tag{3.1}$$

R represents the relation or *placement rule* and e denotes an element. Structural texture is characterized by a precise definition of R and e. Statistical texture by a more macroscopic view, with R and e exhibiting variance.

Tamura amongst others [32] was one of the first to recognize the importance of modeling in accordance with human perception. In psychological experiments by testers, several aspects of both structural and statistical texture were distinguished. She proposed:

Click A.6

contrast , *coarseness* , *directionality* , *line-likeness* , *regularity* and *roughness*

as the main aspects to distinguish the several aspects of texture. These principles have been used in e.g. QBIC [22] and FracFeat, a fractal feature extractor we builded which will be discussed in Section 6.1.

Fractal image compression is concerned with finding similarities between parts of an image and records the spatial relationships between them, see Figure 3.4. These spatial relationships relate to the placement rules of Tamura in a very simple way. Fractal features are well equipped to classify images into natural scenes and human environment (e.g. cities) [29]. Fractal features can be made invariant to a wide range of image variations, like contrast scaling, rotation of the image canvas and even folds. This makes fractal feature extraction applicable within the domain of textiles and fashion [30, 31]. The above aspects are subject of three papers from Chapter 7, 8 and 9. In Chapter 5 we provide a mathematical background on the use of *fractal geometry* for feature extraction.

In the early 1990's when wavelets were introduced, many researchers started to study the wavelet transform for texture representations. Among the different transforms, the Gabor [16] transform shows very good results in modeling both mathematical and perceptual aspects.

3.2 Questions, Questions, and more Questions.

Since there are no *all encompassing truths* in the perception of visual aspects, extracting meaningful features is a challenging research topic.

- **Never enough !**
  A person maybe recognized best by a scar. Faloutsos describes in his 'Gemini' approach a way of looking at the problem of finding the most useful feature:

  Question: If we are allowed to use only one numerical feature to describe each data object, what should this feature be ?[8]

  From this question immediately new questions arise like, which feature to choose, why, and how ?

- **Absolutely arbitrary !**
  Because of subjectivity in perception there does not exist a single best representation for a given feature; there are multiple representations which characterize the given feature from different aspects.

  The amount of features to be extracted from the image is endless. The choice of the features in the system depends heavily on the domain. Moreover, the same image can be retrieved for different reasons over time and the system should be able to come up with the right features and similarity measures every time. Because features are present in the system and provided by the "owner" of the system it is almost impossible to provide the right features. Features are only *partially relevant.*[1]. Another aspect is the invariance of a feature to perturbations of the image. An object can be seen from different angles, images can be rotated, lighting may vary.

- **Never right !**
  Similarity is provided by mathematically defined measures and features. The meaning of the measure to the user are ambiguous. Similarity is subjective and strongly dependent on culture, personal taste and again on different meanings the user can have at different times. In Figure 3.5 we show an example of the images returned by the Virage system [1], as input the hand of a human being was presented.

Some improvements are widely recognized: features should always be an integral part of the database in such a way that the user is able to relate to the system in an intuitive and transparent way. The user should be in the loop, adding knowledge about the similarity and the features wanted, and communicating with the system using *relevance feedback* and learning. Simple similarity metrics are not enough [20].

Click A.7

[1] In content based image retrieval images are returned to the user according to a search process. We distinguish the *partial relevance* of the feature extracted from an image and the *partial relevance* of the returned image to the user: *partial relevance of images*. The latter is subject of Chapter 4, [4]

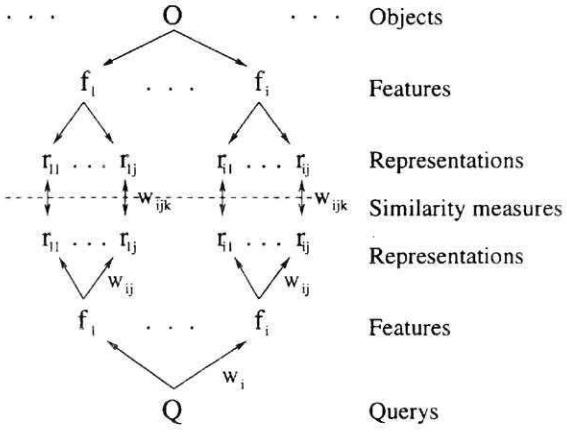

Figure 3.1: Schematic overview [27] of a query process by feature extraction and similarity search. A query image Q is represented by features $f_1, ..., f_i$. Each feature f_i is "composed" of several aspects $r_{i,1}, ..., r_{i,j}$. Weights $w_{i,j}$ are used to represent the importance of the different aspects. A *similarity measure* (dotted line) "compares" the aspects of the query image Q with the aspects of the other images O in the database. Weights $w_{i,j,k}$ can be given to "tune" the similarity measure.

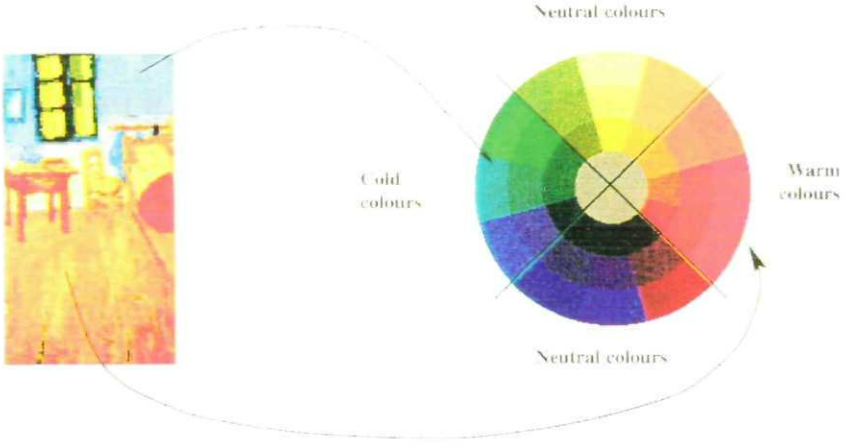

Figure 3.2: Ittens color circle, representing warm and cold colors located in opposite positions.

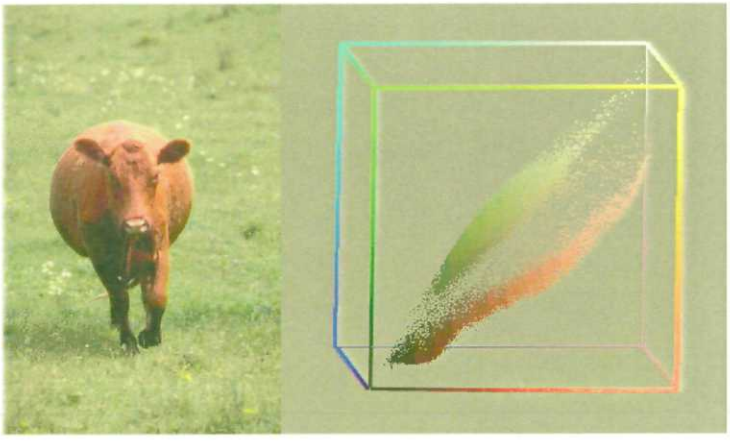

Figure 3.3: RGB space is used to plot color values for each pixel in the image (left). In this plot, the object is clearly seperated from the background (right).

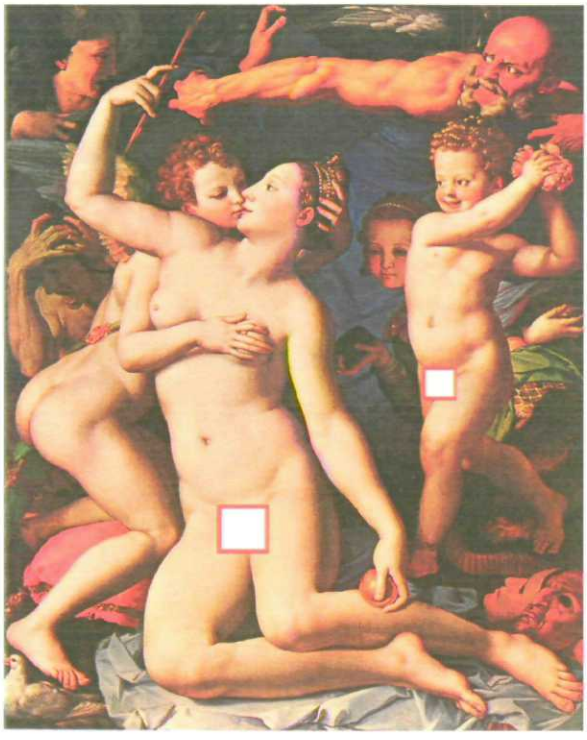

Figure 3.4: A fractal encoder searching for "similar" image blocks. The coder keeps track of the spatial relationships between the similar blocks, used to characterize the image with.

Figure 3.5: *Virage* system [1] at work. *top:* User is presented random images (query by example). Weights can be given to the importance of the features. *bottom:* Retrieved images, including the image of a pig.

Chapter 4

Latest Developments and Research Proposals

Where are we in fifty years ? Click A.13

4.1 Visual Intelligence

Relationships between high level concepts and low level features can be brought into visual information systems by knowledge available within the system or by interaction with the user (*relevance feedback*). A system where all solutions are equally likely is not considered intelligent. It should be biased to guide the system to an answer.

In general there are two ways of achieving some *visual intelligence* for VIR systems [24]: *off line reasoning* and *online learning* or in short the system can have knowledge and/or learn. Expert systems like medical databases are generally equipped with features tailored to describe the content in high level terms. If domain knowledge is not available like in the World Wide Web or not sufficient in the case of subjectivity, knowledge can be gathered by *relevance feedback* of the user.

4.1.1 Reasoning and Interpretation

Low-level features can be translated into higher level semantics by considering relationships between them [7]. Examples are color warmth for color or cheerfulness for sound. In this way more meaning related to human perception can be given to the features (Figure 4.1).

Even *visual signs* describing events, objects, actions can be achieved automatically through recognition and interpretation. Recognition is achieved by a set of low-level features, interpretation is achieved by comparing the recognized image or object to similar ones present in the system, which are already interpreted as concepts.

To reason the system needs an understanding about the several meanings visual signs can have. As an example we would like to mention the *FourEyes* system [17, 18] from M.I.T. Click A.14

4.1.2 Pariss

On the World Wide Web systems become available which allow image retrieval based on the actual content of the image. In this domain little is known about the user and for this reason limited meaning can be given to the features extracted from analysis of the pixel distributions within the image. As a result, ways need to be explored to express this subjectivity to the system. In earlier systems like QBIC the user is invited to give weights to the relevance of the different features. This presupposes knowledge about the features and the influence the features have on the retrieval process. These systems cannot effectively model high level concepts and user's perception subjectivity.

Rui [27] et al. propose the weights to be fixed by examples given by the user. These weights can then be updated by a process of *relevance feedback*. The user indicates the relevance of an image by marking the image from highly relevant to highly irrelevant. A standard deviation based weight updating process is then proposed. A particular feature showing little variation for the images marked "highly relevant", is updated and given more importance. Click A.15

A more sophisticated approach is taken in the *Pariss* interface [4] presented in Chapter 9. The architecture of the interface is designed to offer the user graphical tools to *show* which images he considers similar or an image being relevant for the search process. *Pariss* is short for *Panoramic, Adaptive and Reconfigurable Interface for Similarity Search*. This acronym refers to the following interface-characteristics:

- **Panoramic**:
  In a display window images are represented by points. By clicking on them the actual images are shown. Images are positioned according to user defined similarity criteria.
- **Adaptive**:
  Relevance feedback is used to refine a probability measure that represents the accumulation of information during the search process. A demo illustrates these principles. Click B.8
- **Reconfigurable:**
  Similarities between images can be (re)defined by rearranging images in the display window. A demo illustrates these principles. Click B.8

Interaction with the system is at two levels:

- The database can be *browsed* by the user in the display window. By rearranging the selected thumbnails the interface is requested to define new combinations of features that are able to describe the user-defined notion of similarity. Click A.16
- In addition collection boxes are available to have a positive or negative judgement about the relevance of the image with respect to the target image wanted. A sampling procedure spots trends in the feature values of the images within the collection box and produces new results likely to be more favorable to the user. Click A.17

4.2 (Research) Proposals

The methods within CBIR are based on describing the actual content of the image by low level aspects of the image. Despite high expectations these systems perform inadequately, especially in domains like the World Wide Web. This poor performance is due to the fact that it is too simple to hypothesize modeling human perception in simple features and a similarity metric which are both always relevant and are able to describe the variety of the visual world

As Santini [28] says, the meaning of an image is an ill posed problem, since it depends on the "situatedness" of the observer. The images retrieved are only an example of what the user is looking for, which can be semantically "less or more". As a red car is used as an example to retrieve similar images, "similar" can have different meanings; it might be that the user is interested in a car as an object, on the other hand he might be interested in only simple aspects like being red or having the same shape. The images the database is queried with are only examples of the situatedness of the user. Visualization of content at different levels of abstraction is wanted for.

As image content is not well defined, the descriptors will always fail.

We would like to propose some topics for further research related to content of this thesis.

1. **Other methods of relevance feedback.**
   In most systems, by using *relevance feedback* the partial relevance of the images is taken into consideration. Methods of relevance feedback need not to be restricted to the selection of output images. In fact they might be applicable to other procedures.

   When the user is kept in the loop "for ever" and features are not able to express the content present in image and appreciated by the user, there is a need for adding new content descriptors into the system. As features are precomputed in the system and it cannot be foreseen which feature relates to the perception of the user, a new way of creating content descriptors "on the fly" is wanted for. Click A.18

2. **Other ways of content visualization.**
   Query by image example has proven to be inadequate; in most cases this leads to non relevant results. The image being an example for the user, contains other image content that might be irrelevant or even unwanted. Visualizations with the ability to contain information at different levels of abstractions are required. More abstract visualizations of content might be achieved by pictograms or icons [34]. Also clear aspects of an image, like textural direction could be expressed in this way.

   Within the field of *Graphic Design* these methods have been studied systematically and the composition of these icons can be done by experts. However, a more desirable option would be to generate icons from the image itself or even from the features representing them. These icons can then be added to the query space amongst the other images and *guide* the search.

   Click A.19

3. **Fusion.**
   Then in the same way as keywords provide the possibility of more semantically enhanced queries through Boolean evaluation or natural language processing, we would propose the use of visual keywords with the ability to visually *compose* several aspects of an image (Figure 4.2).

4. **Contextual information retrieval.**
   Classification of a multimedia object can be obtained by combining features from text and visuals [10]. Classification of images might be achieved by thematic labeling of image segments. A "feature bank" consisting of a set of feature extractors (e.g. skin) is be used to extract the appropriate features (Figure 4.3).

5. **Search without interface. Agents.**
   In the domain of criminal investigations. Agents, which actively go out on the Internet and search for a specific document. These agents should be able to locate visual content within different applications, penetrate the document and report the existence of an image or object.

   Click A.20

6. **Intelligent video editing.** Locating scenes (in time) and image components within video streams (television) in order to make the content of these scenes and components adaptive to local, national, individual or cultural groups. Adaptive advertising.

7. **Collaborative information retrieval. Profiling.**
   Reasoning and interpretation are essential for visual information retrieval systems. Without knowledge of the domain wherein the system is used this meaning is hard to extract. Methods need to be explored of building *semantic profiles* extracted from interaction from the user with previous sessions or other systems. The interaction of the user with the system could be stored over the session.

8. **Hierarchical search.**
   Retrieval systems for very large databases impose strong demands on the size of the

feature vectors, this limits the effectiveness of the indexing techniques, and the efficiency of the searching algorithm. One way to approach this problem is to develop hierarchical indexing and searching strategies: in subsequent steps one performs an increasingly detailed search on a smaller and smaller subset of the database. By their multiresolution character, wavelets, and more generally, pyramid transforms (fractals) are tailor made to be exploited in the construction of hierarchical image indexing and searching schemes.

Click A.21

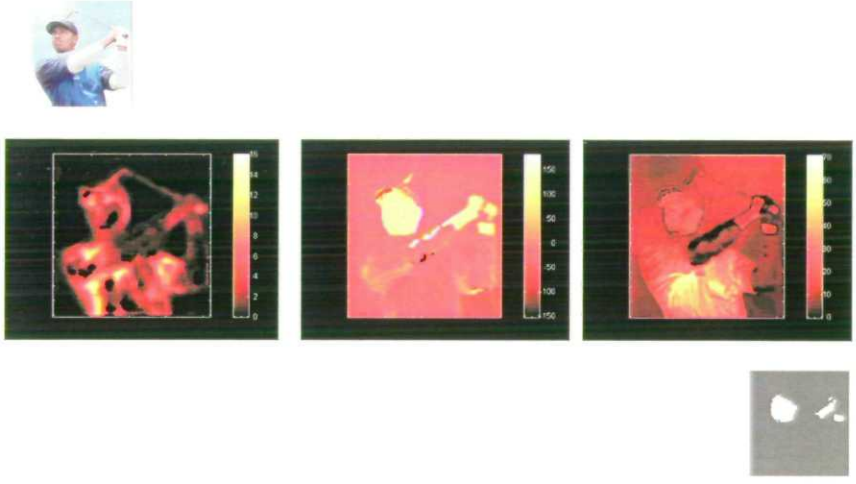

Figure 4.1: A combination of textural and color intensity maps is used to detect human skin.

door + mouth = ask

Figure 4.2: The concept *ask* as fusion of the concepts *door* and *mouth*.

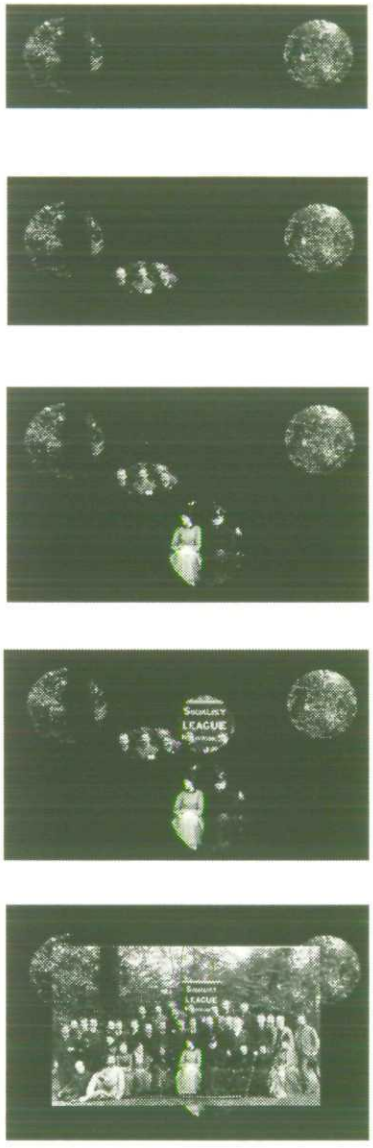

Figure 4.3: *Demo, only on CD.* In successive steps an image is analysed according to regions of interest. To label the image, labels at "lower levels" are related.

Bibliography

[1] J. Bach, C. Fuller, A. Gupta, A. Hampapur, B. Horowitz, R. Humphrey, F. Jain, and C. Shu. The virage image search engine: An open frame work for image management, 1996.

[2] L. Balmelli and A. Mojsilovic. Wavelet domain features for texture description, classification and replicability analysis. In *ICIP; Proceedings IEEE International Conference on Image Processing*, volume CDROM, Session H28AP5B4, 1999.

[3] A. Del Bimbo. *Visual Information Retrieval.* Morgan Kaufmann Publishers Inc., San Francisco, California, 1999.

[4] G. Caenen, G. Frederix, A.A.M. Kuijk, E.J. Pauwels, and B.A.M. Schouten. Show me me what you mean. pariss: a cbir interface that learns by example. In *Visual 2000: Fourth International Conference on Visual Information Systems, Lyon France*, 2000.

[5] B.B. Chaudhuri and N. Sarkar. Texture segmentation using fractal dimension. *IEEE transactions on Pattern Analysis and Machine Intelligence*, 17(1):72–77, January 1995.

[6] T.N. Cornsweet. *Visual Perception.* Academic Press, Orlando, 1970.

[7] A. del Bimbo. Semantics-based retrieval by content. In *ICIP; Proceedings IEEE International Conference on Image Processing*, page CDROM, 2000.

[8] C. Faloutsos. *Searching Multimedia By Content.* Kluwer Academic Publishers, Boston/London/Dordrecht, 1996.

[9] T. Gevers. *Color Image Invariant Segmentation and Retrieval.* PhD thesis, University of Amsterdam, May 1996.

[10] Th. Gevers, F. Aldershoff, and A. W. M. Smeulders. Classification of images on internet by visual and textual information. In *Internet Imaging, SPIE, San Jose,January,2000*, 2000.

[11] T. Huang and Y. Rui. Image retrieval: Past, present, and future. In *Proc. of Int. Symposium on Multimedia Information Processing, Dec 1997.*, 1997.

[12] V. Kashyap and A. Sheth K. Shah. *Multimedia Database Systems: Issues and Research Directions 1996*, pages 297–319. Springer, 1996.

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

[14] R. Kengeri, C.D. Seals, H.D. Harley, H.P. Reddy, and E.A. Fox. Interface and evaluation, usability study of digital libraries: Acm, ieee-cs, ncstrl, ndltd. *International Journal on Digital Libraries*, (1999)(2):157–169, 1999.

[15] S. Manegold, P. A. Boncz, and M. L. Kersten. Optimizing Database Architecture for the New Bottleneck: Memory Access. *The VLDB Journal*, 9(3), 2000.

[16] B.S. Manjunath and M.Y. Ma. Texture features for browsing and retrieval of image data. *IEEE transactions on Pattern Analysis and Machine Intelligence*, 18(8):837–842, August 1996.

[17] T. Minka. An image database browser that learns from user interaction. Technical Report 365, MIT Media Lab Perceptual Computing Section, 1995.

[18] T. Minka. Picard: Interactive learning using a society of models, 1997.

[19] A. Mojsilovic, J. Hu, and R.J. Safranek. Perceptually based color texture features and metrics for image retrieval. In *ICIP; Proceedings IEEE International Conference on Image Processing*, volume CDROM, Session W27PO25, 1999.

[20] A. Mojsilovic, J. Hu, R.J. Safranek, and S.K. Ganapathy. Matching and retrieval based on the vocabulary and grammar of color patterns. *IEEE Transactions on Image Processing*, 9(1):38–53, January 2000.

[21] T. Munzner. H3: Laying out large directed graphs in 3d hyperbolic space. In *Proceedings 1997 IEEE Symposium on Information Visualization*, 1997.

[22] W. Niblack, R. Barber, W. Equitz, M. Glasman, D. Petkovic, P. Yanker, C. Faloutsos, and G. Taubin. The QBIC project: querying images by content using color, texture and shape. *Storage and Retrieval for Image and Video Databases*, 1908:173–187, 1993.

[23] A. Pentland, R.W. Picard, and S. Scaroff. Photobook: Tools for content-based manipulation of image databases. In Wash SPIE, Bellingham, editor, *Storage and Retrieval for Image and Video Databases*, volume II(185), pages 34–47, 1994.

[24] R.W. Picard. Digital libraries: Meeting place for high-level and low-level vision. In *Asian Conference on Computer Vision, Signapore*, page ??, 1995.

[25] R.W. Picard and T.P. Minka. Vision texture for annotation. *Multimedia Systems*, 3:3–14, 1995.

[26] A. Rosenfeld. Visual texture analysis: An overview. Technical report, Computer Science Center, University of Maryland, Tech Report TR-406, August 1975.

[27] Y Rui, T. Huang, and S. Mehrotra. Relevance feedback techniques in interactive content-based image retrieval. In *Proc. of IST and SPIE: Storage and Retrieval of Image and Video Databases VI*, pages 25–36, 1998.

[28] S. Santini and R. Jain. Beyond query by example. In *Proceedings ACM 98, The 6th ACM International Multimedia Conference*, pages 345–350, 1998.

[29] B.A.M. Schouten and P.M. de Zeeuw. Feature extraction using fractal codes. In *Visual 99; Visual Information and Information Systems, Third International Conference*, pages 483–492. Springer, 1999.

[30] B.A.M. Schouten and P.M. de Zeeuw. Fractal transforms and feature invariance. In *Procs. 15th International Conference on Pattern Recognition (ICPR 2000)*, pages 992–997. IEEE Computer Society, 2000.

[31] B.A.M. Schouten and P.M. de Zeeuw. Image databases, scale and fractal transforms. In *ICIP 2000; International Conference on Image Processing*, 2000.

[32] H. Tamura, S. Mori, and T. Yamawaki. Texture features corresponding to visual perception. In *IEEE Transactions on Systems, Man and Cybernetics*, volume 8(6), 78.

[33] D. Tsichritzis and A. Klug. The ansi/x3/sparc dbms framework report of the study group on database management systems. Technical report, University of Toronto, Toronto, Canada, 1978.

[34] T. van Walsum, F.H. Post, D. Silver, and F.J. Post. Feature extraction and iconic visualization. In *IEEE Transactions on Visualization and Computer Graphics*, volume 2, pages 111–119, 1996.

[35] S.W. Zucker. Towards a model of texture. *Computer Graphics and Image Processing*, 5:190–202, June 1976.

Part II

Mathematical Background

Chapter 5

Fractal Geometry for Feature Extraction

Geronimo [6] in his lecture notes describes the importance of fractal geometry as follows: *Texture, color, brightness are among our first observations. It is widely believed that some local wavelet functions, in particular Gabor Functions, provide appropriate descriptions of visual cortical receptive fields. The brain plays a far more important role.*
When we look at picture, the first thing we do is to identify objects in the picture. Identification of objects is the process of segmenting the image and comparing these segments with similar objects in our memory. We can consider this process as finding similarities in the time dimension. We expand our view by identifying things and places spatially: for example, five people, two dogs and three trees. After a while we even see subtle details that we didn't notice in the first glimpse (multi resolution). All these processes can be classified as the tempo-spatial process of searching for similarities. This process is exactly the fractal mechanism in human vision. People always communicate new experiences using commonly shared past experiences

As a result of psychological studies we start to understand the functioning of the brain; similarity is a salient perceptual feature which can be used as a leading principle for researchers in their search how to compress and recognize visual information within computer vision. Mathematicians like Felix Klein (1849-1925),

Click A.8

`http://www.treasure-troves.com/bios/KleinFelix.html`,

described geometry as the study of properties of figures which remain invariant under a group. Archaeologists like Dorothy Washburn use the same principles to describe the ceramics of ancient cultures.

Click A.9

Self-similar fractals are invariant under certain transformations and their properties can best be described by these transformations. In *fractal image compression* only eight different (isometric) symmetry operations which leave the square unchanged are used, the so called square symmetries. These operations (modulo translations) form a subgroup (under multiplication) of the orthogonal group $O(\mathbb{R}^2)$ and can be characterized by linear algebra:

$$W(x) = Ax + b \tag{5.1}$$

for which $det(A) = 1$ or -1.

The extraction of fractal features of (textural) images, is based on the concept of fractal image compression. After Mandelbrot [10] in the 60's had formalized the concept of a fractal, Barnsley [2] in his book *Fractals Everywhere* introduced the theory of *iterated function systems* (IFS). A small set of affine transformations is able to generate types of self-similar fractals. Based on IFS theory, Barnsley has a patent for an algorithm that compresses images to FIF, the fractal image format.

Click B.4

To understand *fractal image compression* and how it can be used for the retrieval and classification of images, we first create an idea of fractal geometry. The word fractal was coined by Mandelbrot [10] and stems from the Latin *fractus*, meaning broken. Mandelbrot wanted to describe a class of objects, which are much more irregular in the sense of traditional geometric settings. Falconer [4] doesn't need a definition of a fractal. "Biologists have no definition of life, but still know what it is".

5.1 Ways to Create Fractals

Some fractals can be created by simple construction rules, an *iterated function system* (IFS) is one of the simplest ways of generating a certain type of fractal. To begin our survey we need to consider a *complete metric space* (X,d).

A space X is complete if every *Cauchy sequence* in X converges to a limit point in X and a mapping W on X is called *contractive* or a *contraction mapping* [16] if there exists a $s < 1$, such that for all $x,y \in X$ holds:

$$d(W(x),W(y)) \leq sd(x,y). \tag{5.2}$$

We will call s the *contractivity factor* of W. While iterating W an *attractor* is created as a consequence of the *Contraction Mapping Fixed-Point Theorem* [9].

Theorem 1 Contraction Mapping Fixed-Point Theorem [1] *Let X be a complete metric space and $W : X \rightarrow X$ be a contraction mapping. Then*

[1] For proofs on the theorems used in this Section we refer the reader to the book of Yuval Fisher, "Fractal Image Compression: Theory and Application" [5].

- *there exists a unique point* $x_W \in X$ *such that* $lim_{n\to\infty} W^{\circ n}(x) = x_W$ *for any* $x \in X$ *and*
- x_W *is invariant,* $W(x_W) = x_W$. *The point* x_W *is called the attractor of* W.

5.1.1 Hausdorff Metric

We now define a *space of fractals* $\mathcal{H}$. Let (X,d) be a complete metric space. $\mathcal{H}(X)$ represents the space whose elements are the compact subsets of X. We provide this space with the *Hausdorff metric*.

Definition 1 *The Hausdorff metric between two sets* $A,B \in \mathcal{H}(X)$ *is defined as:*

$$h(A,B) = max\{d(A,B),d(B,A)\}, \tag{5.3}$$

where

$$d(A,B) = max\{d(x,B) : x \in A\} \tag{5.4}$$

and

$$d(x,B) = min\{d(x,y) : y \in B\}. \tag{5.5}$$

If X is a complete metric space, $\mathcal{H}(X)$ is a complete metric space.

5.1.2 Iterated Function Systems

Although applicable to many spaces, we will set $\mathcal{H}(\mathbb{R}^2)$ as our space of fractals. Let $F \in \mathcal{H}(\mathbb{R}^2)$.

Definition 2 *An iterated function system (IFS) on* $\mathcal{H}(\mathbb{R}^2)$ *is a finite collection of contraction mappings:*

$$w_i : \mathcal{H}(\mathbb{R}^2) \to \mathcal{H}(\mathbb{R}^2), i = 1,..,N. \tag{5.6}$$

$$W(F) = \bigcup_{i=1}^{N} w_i(F). \tag{5.7}$$

Theorem 2 *If* $w_i : \mathbb{R}^n \to \mathbb{R}^n$ *is contractive with contractivity factor* s_i *for* $i = 1,\ldots,n$ *then*

$$W = \bigcup_{i=1}^{n} w_i : \mathcal{H}(\mathbb{R}^2) \to \mathcal{H}(\mathbb{R}^2), \tag{5.8}$$

is contractive in the Hausdorff metric with contractivity factor [2] $s = max(s_i),\ i = 1,\ldots,n.$

Since $\mathcal{H}(\mathbb{R}^2)$ is our space of fractals, we choose *affine transformations* as elements of an IFS and consider the *Sierpinsky triangle* (Figure 5.1) as an example. In this case $w_i, i = 1,\ldots,N$ can be written as:

$$w_i(\vec{x}) = A_i\vec{x} + \vec{o}, \quad A_i \equiv \begin{pmatrix} a_i & b_i \\ c_i & d_i \end{pmatrix}, \tag{5.9}$$

$$\vec{x} \equiv \begin{pmatrix} x \\ y \end{pmatrix}, \qquad \vec{o} \equiv \begin{pmatrix} e_i \\ f_i \end{pmatrix}. \tag{5.10}$$

for $\vec{x} \in F \subset \mathcal{H}(\mathbb{R}^2)$.

As a consequence the Sierpinsky triangle can be fully described by the parameters

$$\{a_i, b_i, c_i, d_i, e_i, f_i \mid i = 1,\ldots,3\}, \tag{5.11}$$

specified in Table 5.1.

Table 5.1: IFS description for Sierpinsky Triangle.

i	a_i	b_i	c_i	d_i	e_i	f_i
1	0.5	0	0	0.5	0	0
2	0.5	0	0	0.5	100	0
3	0.5	0	0	0.5	50	50

A demo is available which creates several fractals using IFS. Click B.4

5.2 Fractal Dimension

The topological dimension is not well suited to measure the irregularity and structure of a fractal. We introduce the *fractal dimension* and define[3]:

$$B_r^o(x) = \{y : \|y - x\| < r\} \tag{5.12}$$

[2] This condition is sufficient but not necessary, a weaker condition is possible, for which not all w_i are contractive [5].

[3] We restrict ourselves to *n-dimensional Euclidian space*, $\mathbb{R}^n$

as the *open ball* of center x and radius r. Then a set A is called a *neighborhood* of a point x if there is some open ball $B_r^o(x)$ centered at x and contained in A. The set A is totally disconnected if the connected components of each point consists of just that point. This will certainly be so if for any pair of points x and y in A we can find disjoint open balls $B^o(x)$, $B^o(y)$ and $A \subset B^o(x) \cap B^o(y)$. In this way the topological dimension can be defined as a recursive relationship.

Definition 3 *The topological dimension of a totally disconnected set is zero. The topological dimension of a set F is n if every neighborhood of every point within F, has a boundary with topological dimension* $n-1$.

Several definitions of the fractal dimension can be found in the literature [2, 3, 4, 5, 10] amongst them the *box counting dimension.* [4] , Click A.10

Definition 4 *Let* $F \subset \mathbb{R}^n$. *For each* $\varepsilon > 0$ *let* $N_\varepsilon(F)$ *be the smallest number of balls of radius no larger then* ε *necessary to cover F. The box dimension of F is:*

$$\lim_{\varepsilon \to 0} \frac{logN_\varepsilon(F)}{-log\varepsilon}$$

if this limit exists.

5.3 Partioned Iterated Function System (PIFS)

5.3.1 Collage Theorem

If we want to find an IFS W which can generate a given fractal F, we are faced with a complex problem. There is not an unique and simple solution for this problem.

The *collage theorem* however tells us that for F and F_W to be close, it is sufficient for F and $W(F)$ to be close. The collage theorem provides us with an upper bound for the distance between the attractor and the original set used as input for the system.

Theorem 3 Collage Theorem. *Let X be a complete metric space and* $W : X \to X$ *be a contraction mapping with contractivity factor s and let* F_W *be the attractor of this mapping. Then,*

$$h(F, F_W) \leq \frac{1}{1-s} h(F, W(F)). \tag{5.13}$$

The fundamental principle of fractal coding consists of the representation of an image by a contractive transform whose fixed point is close to that image. In general it is not easy to find

such a transform composed of affine transformations on the ENTIRE image. Fractal image compression (FIC) became reality with the introduction of the *partioned iterated function system*, which is an extended version of IFS. Instead of using the entire image for the individual maps, each of the individual maps operates on a subset, called *range block*, of the support of the image. Together these range blocks cover the image, constituting a *partition*.

Several image partitions can be used in FIC. A first choice is to divide the image into range blocks of fixed size. A better choice is to make use of a quadtree partition, the well-known image processing technique based on recursive splitting of selected image quadrants, enabling the resulting partition to be presented by a tree structure in which each internal node has four descendants [18] (Figure 5.2).

As we are interested in compressing gray scale images f, we would like to add two new parameters to the affine transformations of equations 5.9 and 5.10:

$$w_i(\vec{x}) = A_i\vec{x} + \vec{o},\ A_i \equiv \begin{pmatrix} a_i & b_i & 0 \\ c_i & d_i & 0 \\ 0 & 0 & s_i \end{pmatrix}, \tag{5.14}$$

$$\vec{x} \equiv \begin{pmatrix} x \\ y \\ f(x,y) \end{pmatrix}, \vec{o} \equiv \begin{pmatrix} e_i \\ f_i \\ o_i \end{pmatrix}. \tag{5.15}$$

The new parameters s_i and o_i can be seen as a contrast scaling and a luminance adjustment of the gray values within the image.

5.4 Fractal Image Compression (FIC)

A compression system using PIFS involves splitting the support E of an image into a set $\mathcal{R}$ of N non-overlapping *range blocks*, $R_i, i = 1,\ldots,N$. Then for each $R_i \in \mathcal{R}$, we would like to find an affine transformation w_i and another sub block of the image called *domain block*. The procedure is illustrated in a demo. Each transformation w_i maps a domain block D_i onto a rangeblock R_i, acting upon the restriction of the image f to the domain block (Figure 5.3).

Click B.5

$$w_i : G^{D_i} \to G^{R_i},\ G = \{0,\ldots,255\}, \tag{5.16}$$

$$w_i(f)(x,y) := \gamma_i\left(f\left(\rho_i^{-1}(x,y)\right)\right),\ (x,y) \in R_i. \tag{5.17}$$

γ_i is called the *geometric transform*; ρ_i the *massic transform* of w_i.

$W = \bigcup_{i=1}^{n} w_i$ acts upon the entire image and is defined as:

$$W(f)(x,y) = w_i(f)(x,y) \tag{5.18}$$

for each $(x,y) \in R_i$.

5.5 Iterated.com: "Barnsley goes commercial"

A frequently used coding scheme originates from Jacquin [7], a former Ph.D. student from Barnsley. The transformations w_i, $i = 1,...,N$ can be written down as:

$$w_i \begin{pmatrix} x \\ y \\ f(x,y) \end{pmatrix} = \begin{pmatrix} a_i & b_i & 0 \\ c_i & d_i & 0 \\ 0 & 0 & s_i \end{pmatrix} \begin{pmatrix} x \\ y \\ f(x,y) \end{pmatrix} + \begin{pmatrix} e_i \\ f_i \\ o_i \end{pmatrix}. \tag{5.19}$$

In the case by Jacquin the geometric transform γ_i modulo *translation*, $\begin{pmatrix} a_i & b_i \\ c_i & d_i \end{pmatrix}$ is one of a range of eight different possibilities:

$$\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}, \begin{pmatrix} -1 & 0 \\ 0 & 1 \end{pmatrix}, \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}, \begin{pmatrix} -1 & 0 \\ 0 & -1 \end{pmatrix}, \tag{5.20}$$

$$\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}, \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix}, \begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix}, \begin{pmatrix} 0 & -1 \\ -1 & 0 \end{pmatrix}. \tag{5.21}$$

The *massic transform* ρ_i, acting on the gray values by contrast scaling and luminance offset is constant for each domain block D_i:

$$\rho_i(f(x,y)) \equiv s_i * f(x,y) + o_i,\ i = 1,\ldots,N.\ s_i \leq 1. \tag{5.22}$$

Compression is obtained by storing the parameters

$$\{a_i, b_i, c_i, d_i, e_i, f_i \mid i = 1,\ldots N\} \tag{5.23}$$

as well as spatial information (size, location) regarding the range and domain blocks, in the fractal transform. In Figure 5.4 the decoding process is illustrated. Detail is created with every iterating step.

With courtesy of Michael Barnsley a commercial coder "*Fractal Imager*" is ready to install on this CDROM; a *pseudo-code* [5] can be found in Appendix A.

Click B.7

Click A.11

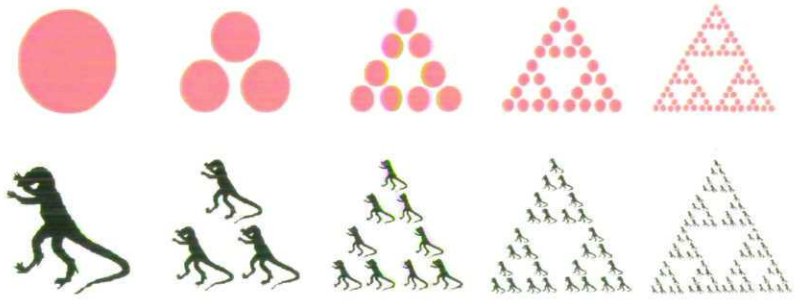

Figure 5.1: The Sierpinsky triangle created by an iterated function system consisting of three transformations. The figure can be created starting with any initial input.

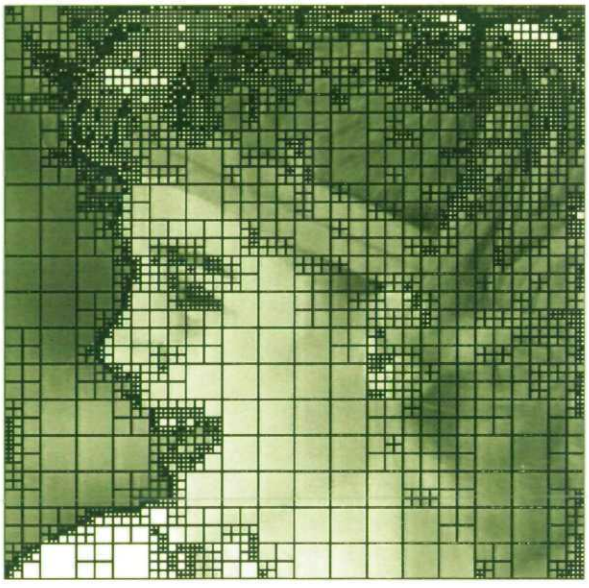

Figure 5.2: A quadtree partition is a representation of an image as a tree in which each node potentially has four subnodes. Each node of the tree corresponds to a square which is a quadrant of it's parent square.

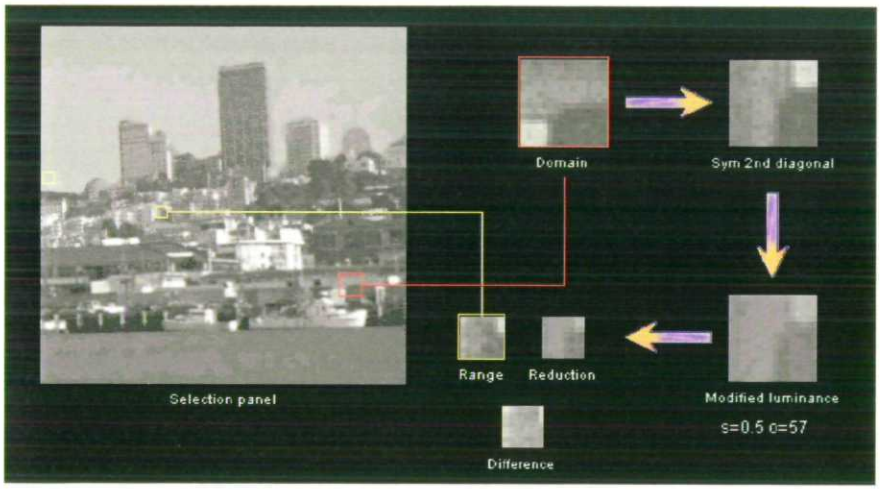

Figure 5.3: First a domain block is transformed by a *geometric transform* (sym. 2nd diagonal). Secondly it is the subject of a *massic transform*.

Figure 5.4: A painting by the Florentine Mannerist Angolo Bronzino (1503-1572), decoded by a fractal transform. Detail is created with every decoding step.

Chapter 6

FracFeat and other Feature Extractors

Within *content based image retrieval* (CBIR), researchers use the fractal transform for *fractal feature extraction*, usually in the domain of textures. We distinguish:

- Algorithms making use of different parameters drawn from the fractal code.
- Algorithms wherin the fractal coding is altered for specific recognition purposes.

Most methods for fractal feature extraction [1, 8, 11, 12, 15, 17, 13] are based upon the parameters from the fractal decomposition of images (see Section 5.4 and 5.5):

- γ_i: *geometric transform.* This transform is used to translate, rotate and/or flip the domain block onto the range block.
- x_i, y_i: position of the domain block.
- s_i, o_i : *massic transform.* Contrast scaling and luminance offset of the gray values.

Marie-Julie [11] proposes a method where images can be retrieved containing a predefined *pattern.* For both the pattern and the images in the database all maps w_i are stored as parameter vectors W_i:

$$W_i = (\gamma_i, x_i, y_i, s_i, o_i). \tag{6.1}$$

Each feature vector originating from the query pattern is compared with the feature vectors from all images in the database. A similarity measure is then calculated based on a l_2 distance

between the parameter vectors. Robustness of the algorithm to variations like rotations, shifts of the image canvas is achieved by compressing the images in different positions (*multicompression*). A drawback is the high complexity of this method.

6.0.1 Decompression for Feature Extraction

A nice way of extracting features from images employs the *decomposition scheme* of the fractal encoding which has been used by Neil [12] and Tan [15]. An image f is coded by finding a transform W for which $W(f)$ approximates f:

$$W(f) \approx f. \tag{6.2}$$

According to the Contractive Mapping Fixed-Point Theorem (Section 5.1), the image f is restored by iterating W starting with any initial image. Suppose a database contains a number of images $f^i, i = 1, ..., n$ and each image is represented by its fractal transform W^i. If an image f^j is presented as a query image to the database with the purpose to retrieve similar images, the distance:

$$d(W^j(f^i), f^i), j = 1, ...n$$

is minimized (Figure 6.1), where

$$d(f^i, f^j) = \sqrt{\sum_{k=0}^{I_h} \sum_{l=0}^{I_w} (f^i_{k,l} - f^j_{k,l})^2}, \tag{6.3}$$

and $f^i_{k,l}$ denotes the pixel value of image f^i at position (k,l). I_h and I_w are the height and width of the image. An index is build for j. This index orders the images starting with the most similar images to the query image f^j. Other methods are discussed in Appendix A. Click A.12

6.1 FracFeat

In Chapters 7, 8 and 9 features based on fractal image decomposition are introduced and their invariance under certain image transformations discussed. The above led to the construction of a computer code: *FracFeat*. It includes tools for adding images and the integrated computation of features together with facilities for querying. FracFeat is aimed at the domain of textiles and fashion. Databases used in this domain usually contain multiple textiles differing only in orientation, scale and the area cutout. Sometimes textiles are depicted in different levels of zoom. A demo of fracfeat is available at: Click B.2

`http://www.desk.nl/FracFeat/FracFeat.htm`.

Two main principles led to the use of FIC as a feature extractor:

- There is a content-dependent relationship between the choice of the coder for a specific range-domainblock combination. Range blocks from a certain region are approximated by domain blocks from the same region or a region with similar image content. This observation can be used to relate parameters of the fractal transform to perceptual properties like direction and scale of a texture. Click B.5
- Although fractal geometry can be for feature extraction of texture, generally there is a major drawback in using fractal transforms. A small perturbation of the region to be encoded, usually results in a major change of the transform (Figure 6.2). However the features defined in FracFeat do not suffer from this ill condition. By using features based on statistics kept during the actual decomposition we derive features from fractal transforms which are invariant to perturbations like rotation, translation, folding or contrast scaling and even a zoom-in at a homogeneous textural image.

6.1.1 FracFeat Features

Tamura [14] textural aspects were used to model our features: Click A.6

contrast , *coarseness* , *directionality* , *line-likeness* , *regularity* and *roughness*

- **FracFeat: Directionality**

  In fractal image coding, similarity search is applied between blocks within an image. In order to compare the gray values of the *range block* with the gray values of the *domain block*, the domain block is the subject of eight possible *geometric transforms* γ_i, $i = 1,...8$ (Section 5.4, 5.5). To detect the number of textural directions, we evaluated the used geometric transform γ_i, modulo translation.

  Figure 6.3 shows two images, a natural scene and a building structure. In the histograms, horizontally we depict the eight possible values for γ_i, $i = 1,...,8$. At the vertical axis we depict the fraction of the total number of range blocks in the image. It is clearly seen that for the image showing a natural scene, the geometric transforms are equally divided for $i = 1,...,8$. Regarding the building image, the distribution is biased towards $i = 1$ and $i = 3$, relating to the identity transformation and a rotation of π. These facts are in correspondence with the nature of the images; buildings having a more vertical orientation, while natural scenes show different textural directions.

- **FracFeat: Coarseness**

  Similarity search is applied to parts of an image, according to a quadtree partition of the image (Section 5.3). An image block is potentially subdivided into four sub blocks.

The set of ranges (*range pool*) can therefore be seen as a union of sets of range blocks of fixed size:

$$\mathcal{R} = R^0 \cup, ..., R^m, \tag{6.4}$$

where m is called the *quad tree depth*; we define:

$$s_m = \frac{\#(R^m)}{N}. \tag{6.5}$$

N is the total number of range blocks within the image; s_m is called the success-rate of the fractal encoding at quad tree depth m. In Figure 6.4 two examples are shown, *Clouds.0000* and *Metal.0001*. A histogram depicts the *accumulated* success-rate s_m versus the quad tree depth m for $m = 1, ..., 8$. The "Clouds" image illustrates a highly cumulative success rate, while the "Metal" image consists mainly of *statistical texture*, resulting in a histogram showing few range blocks matched.

- **FracFeat: Spatial dimension**

  Inspired by the fractal dimension we calculated:

$$d_i = {}^2log \frac{\#(R^m)}{\#(R^{m+1})}. \tag{6.6}$$

  This expression can be seen as a measure for the homogeneity of the range blocks at a certain level of the quadtree. This feature is modified in FracFeat into a local feature, evaluating d_i over a limited blocksize within the image. We show one example of this so called *spatial dimension feature*, which can be used to detect the edges, in Figure 6.5.

  The value of d_i exploits the fractal dimension of parts of the image at a certain quadtree depth. Typically lines and borders yield $d_i \approx 1$ and areas with lots of inner structure yield $d_i \approx 2$. We observe how it is roughly similar to the geometry and topology of the original images and indeed borders and inner areas can be identified by different values of d_i.

Table 6.1: Invariances for different fractal feature extractors.

	Rotation	Translation	Scale	Fold	Zoom
Tan	yes	multicompression	yes	no	no
Marie-Julie	multicompression	multicompression	yes	no	no
FracFeat	yes	yes	yes	yes	yes

At the end of this Chapter, we compare different fractal feature extractors as discussed in this Chapter, including FracFeat. We evaluated (Table 6.1) the robustness of the features in relation to a rotation, translation, rescaling and folding of the image canvas, as well as a a change in lighting conditions. This subject is elaborated in Chapter 8.

Because fractals are self-similar, in a way they can be considered as zoom-invariant. Although fractal compressed images are not real fractals, the features we derived can be made invariant to zooming in or out in homogenous image areas. In Chapter 9 we show how.

A simple way of making features invariant to a perturbation A of an image f is *multicompression*. Within this method both features of f and $A(f)$ are extracted and stored in the database.

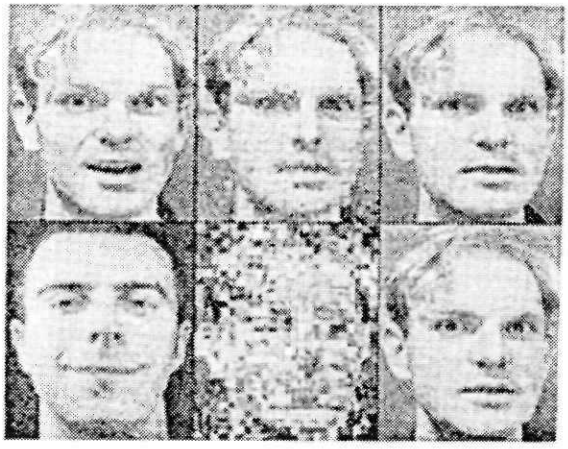

Figure 6.1: *left column:* Two database images f and g. *middle:* Distortion as a result of transforming the image by the fractal transform of a query image. *right:* The query image h.

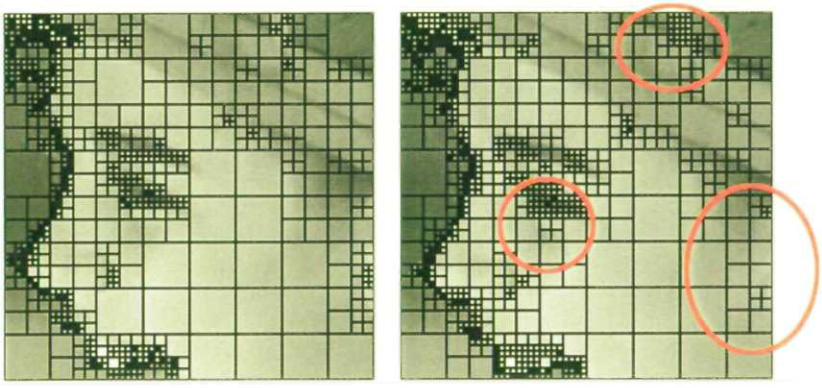

Figure 6.2: A small shift of the region to be encoded with respect to the image canvas, results in a modified quad tree structure, making it hard to "compare" shifted images.

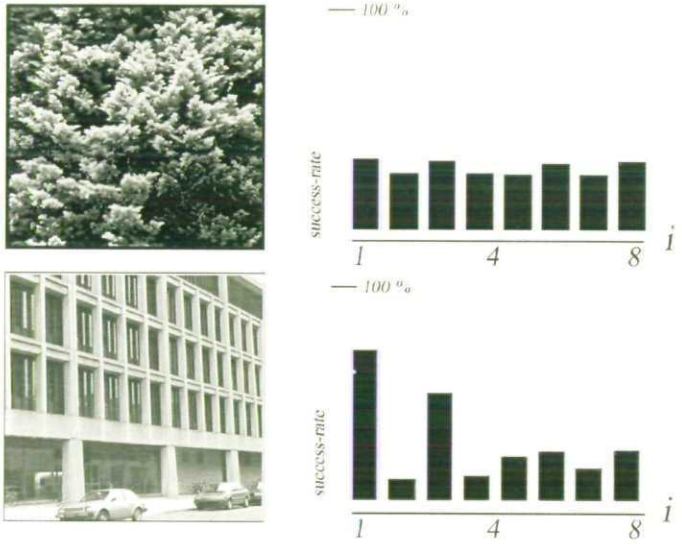

Figure 6.3: Two images *"Leaves.0006"* and *"Buildings.0009"* showing different *directionality* histograms.

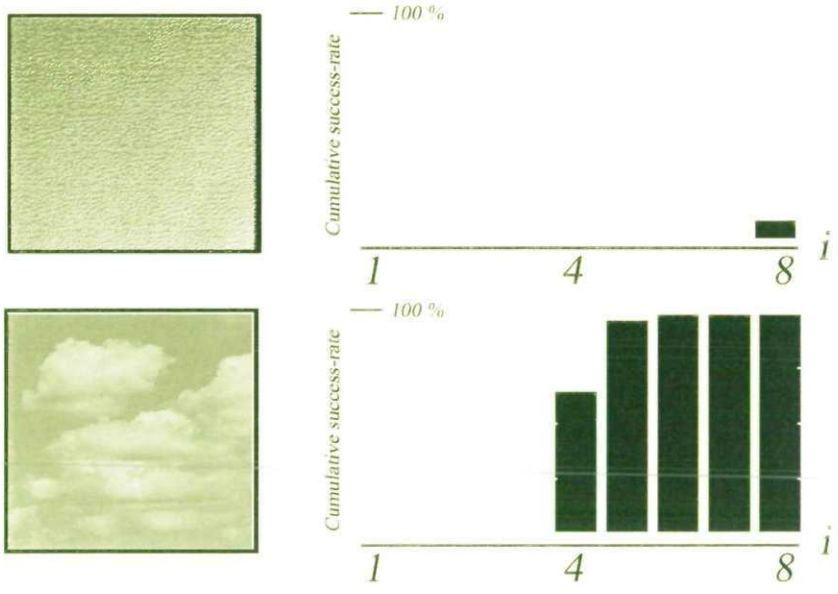

Figure 6.4: Two images *"Metal.0001"* and *"Clouds.0000"* showing different *coarseness* histograms.

00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
20 20 10 10 10 16 10 10 20 10 00 00 00 10 00 00
16 10 20 16 10 16 10 16 16 10 10 20 20 16 20 10
20 16 20 16 20 20 20 16 20 10 10 16 16 00 00 16
00 00 00 10 00 16 10 20 20 20 20 20 00 16 00 00
20 16 16 20 16 00 16 20 20 20 20 20 00 00 00 00
20 16 16 20 16 20 00 20 20 10 16 10 10 00 20 00
16 20 16 20 16 20 20 20 16 10 00 00 10 10 16 16
00 00 16 10 00 16 00 00 16 16 20 10 00 10 16 16
00 00 00 00 00 16 10 10 00 00 00 00 16 10 10 16
00 00 00 16 10 00 00 10 00 00 10 10 16 00 00 10
00 00 00 00 00 00 00 10 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 10 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 10 10 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

Figure 6.5: Spatial dimension (x10); "DogCageCity.0002" image.

Bibliography

[1] M. Baldoni, C. Baroglio, D. Cavagnino, and L.Egidi. *Learning to Classify Images by Means of Iterated Function Systems*, pages 173–182. World Scientific Publishing Co. Pte. Ltd, Singapore, 1998.

[2] M. Barnsley. *Fractals Everywhere.* Academic Press, Inc., San Diego, USA, 1988.

[3] B.B. Chaudhuri and N. Sarkar. Texture segmentation using fractal dimension. *IEEE transactions on Pattern Analysis and Machine Intelligence*, 17(1):72–77, January 1995.

[4] K. Falconer. *Fractal Geometry-Mathematical Foundations and Applications.* John Wiley and Sons,Chichester, 1990.

[5] Y. Fisher. *Fractal Image Compression: Theory and Application*. Springer-Verlag, 1995.

[6] J. Geronimo and N. Lu. Lectures on fractal techniques in image compression. Technical report, Georgia Institute of Technology, 1996.

[7] A. Jacquin. *A Fractal Theory of Iterated Markov Operators with Applications to Digital Image Coding*. PhD thesis, Georgia Institute of Technology, August 1989.

[8] A.Z. Kouzani, F. He, and K. Samut. Fractal face representation and recognition. In *IEEE International Conference on Systems, Man and Cybernetics*, pages 1609–1613, 1997.

[9] E. Kreyszig. *Introductory Functional Analysis with Applications*. Wiley, New York, 1978.

[10] B.B. Mandelbrot. *The Fractal Geometry of Nature.* W.H. Freeman and Company, New-York, 1983.

[11] J.M. Marie-Julie and H. Essafi. Image database indexing and retrieval using the fractal transform. In *ECMAST '97; Multimedia Applications, Services and Techniques*, pages 483–492. Springer, 1997.

[12] G. Neil and K.M. Curtis. Scale and rotationally invariant object recognition using fractal transformations. In *ICASSP; Proceedings IEEE International Conference on Acoustics, Speech, and Signal Processing*, volume 6, pages 3458–3461, 1996.

[13] B.A.M. Schouten and P.M. de Zeeuw. Feature extraction using fractal codes. In *Visual 99; Visual Information and Information Systems, Third International Conference*, pages 483–492. Springer, 1999.

[14] H. Tamura, S. Mori, and T. Yamawaki. Texture features corresponding to visual perception. In *IEEE Transactions on Systems, Man and Cybernetics*, volume 8(6), 78.

[15] T. Tan and H. Yan. Face recognition by fractal transformations. In *ICASSP; Proceedings IEEE International Conference on Acoustics, Speech, and Signal Processing*, volume 6, pages 3537–3540, 1999.

[16] M.J. Turner, J.M. Blackledge, and P.R. Andrews. *Fractal Geometry in Digital Imaging*. Academic Press, London, 1998.

[17] M. Vissac, J. Dugelay, and K. Rose. A fractals inspired approach to content-based image indexing. In *ICIP; Proceedings IEEE International Conference on Image Processing*, page CDROM, 1999.

[18] Brendt Wohlberg and Gerhard de Jager. A review of the fractal image coding literature. *IEEE Transactions on Image Processing*, 8(12):1716–1729, December 1999 1999.

Part III

Four Publications

The four Chapters in this part were published as:

- B.A.M. Schouten and P.M. de Zeeuw. *Feature Extraction using Fractal Codes.* Proc. of VISUAL 99; Third International Conference on Visual Information Systems. Amsterdam, Springer Verlag, pp. 483-492, 1999.
- B.A.M. Schouten and P.M. de Zeeuw. *Fractal Transforms and Feature Invariance.* ICPR 2000; Proceedings International Conference on Pattern Recognition. Barcelona, Volume 3, pp. 992–997, 2000.
- B.A.M. Schouten and P.M. de Zeeuw. *Image Databases, Scale and Fractal Transforms.* ICIP 2000; Proceedings International Conference on Image Processing. Vancouver, 2000.
- G Caenen, G Frederix, A.A.M Kuijk, E.J Pauwels and B.A.M Schouten: *Show me what you mean !. PARISS: A CBIR-interface that learns by example.* Proc. of VISUAL 2000; Fourth International Conference on Visual Information Systems. Lyon, pp.257-268, 2000.

We refer the reader to Chapter 5 for a mathematical background on the use of fractal geometry for feature extraction.

Chapter 7

Feature Extraction using Fractal Codes

Ben A.M. Schouten and Paul M. de Zeeuw [1]
Centre for Mathematics and Computer Sciences (CWI)
P.O. Box 94079, 1090 GB Amsterdam, The Netherlands
{B.A.M.Schouten, Paul.de.Zeeuw}@cwi.nl

Fast and successful searching for an object in a multimedia database is a highly desirable functionality. Several approaches to content based retrieval for multimedia databases can be found in the literature [7, 8, 10, 12, 14]. The approach we consider is feature extraction. A feature can be seen as a way to present simple information like the texture, color and spatial information of an image, or the pitch, frequency of a sound etc.

In this paper we present a method for feature extraction on texture and spatial similarity, using fractal coding techniques. Our method is based upon the observation that the coefficients describing the fractal code of an image, contain very useful information about the structural content of the image. We apply simple statistics on information produced by fractal image coding. The statistics reveal features and require a small amount of storage. Several invariances are a consequence of the used methods: size, global contrast, orientation.

[1] B.S. gratefully acknowledges support by NWO, The Netherlands.

7.1 Introduction

Automatic indexing and retrieval of images based on content is a challenging research area. What is called content is usually subjective and often depends on the context, domain, etc. This is the reason why content based access is still largely unsolved. High-level content based retrieval requires the use of domain knowledge and is therefore limited to a specific domain. Low-level retrieval techniques are more generic but they can characterize only low-level information such as color, texture, shape, motion etc.

We are interested in low-level retrieval techniques for grey scale images, based on texture and spatial (dis)similarity. We wish to locate a set of images related to a given image ("Query by example"). Fractal coding is effective for images having a degree of self-similarity. Here similar means that a given region in an image may be fitted to another region using some affine transformation. This notion of similarity is particularly useful for textured regions. We will make a distinction between three main aspects of texture: symmetry, contrast and coarseness [13]. The spatial similarity features will be based on spatial relationships, like the distance and the angle between the similar regions.

The power of the method lies in the multiresolution nature: retrieval of high-resolution database images with low-resolution original inexact queries is possible. Retrieval systems for very large databases impose strong demands on the size of the feature vectors, the effectiveness of the indexing techniques, and the efficiency of the searching algorithm. Therefore the features should be simple to compute and be discriminating. It is necessary to develop hierarchical indexing and searching strategies, that is, in subsequent steps one performs an increasingly detailed search on a smaller and smaller subset of the database. By their multiresolution character fractal coding techniques are apt to the construction of hierarchical image indexing and searching schemes.

This paper is a first survey on how effective feature extraction based on fractal codes can be. The need for features with discriminating potential and the possibilities offered by hierarchical schemes in this respect gave reason to write this paper.

7.2 Background

7.2.1 Fractal Image Coding

Fractal coding is a relatively new technique which emerged from fractal geometry. It has been studied thoroughly by several authors, see e.g. [3, 11]. Fractal coding is based on the self-similarity in a picture. This means that small pieces of the picture can be approximated by transformed versions of some other (larger) pieces of the picture. This phenomenon is exploited to extract features that relate to this self-similarity.

We give a brief introduction to fractal image coding, cf. [1, 5, 2]. Without loss of generality, we suppose that the image I measures $2^N \times 2^N$ pixels. We will denote this image area with E. We consider grey scale images and define $G = \{0, ..., 255\}$. So:

$$I : E \to G, \quad I \in G^E, \tag{7.1}$$

$I|_R$ is the restriction of the image to the region R and $I_R := I|_R$.

In fractal coding the image I is partitioned into non-overlapping sub blocks of fixed size, called range blocks. See Figure 7.1 (courtesy of Dugelay et al. [2]). The fractal encoder searches for every range block R, another block in the image (domain block D) that looks similar under an affine transformation. The range blocks are identified by the coordinates of the lower left corner of the block

$$\mathcal{R} = \left\{ (2^d m, 2^d n) \mid 0 \leq m, n \leq 2^{N-d} - 1. m, n \in \mathbb{Z} \right\}.$$

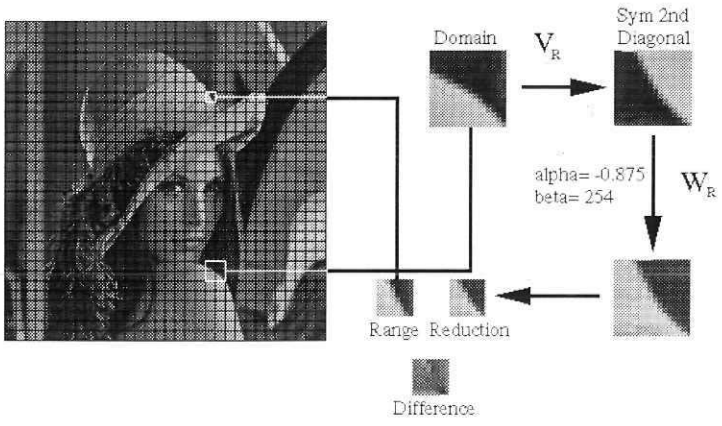

Figure 7.1: Fractal coding in steps

The goal of the expression scheme is to approximate, within a certain tolerance ε, the range block R by a certain domain block D of double size: $2^{d+1} \times 2^{d+1}$. The chosen domain block is extracted from a domain pool $\mathcal{D}$. There are several kinds of domain pools; for our survey we use the half overlapping domain pool

$$\mathcal{D} = \left\{ (2^{d+1}\frac{m}{2}, 2^{d+1}\frac{n}{2}) \mid 0 \leq m, n \leq 2^{N-d} - 1. m, n \in \mathbb{Z} \right\}.$$

The approximation of the range block by the domain block is done in several steps:

a. The domain block is brought into position by an symmetry operator V_R.

b. The grey values of the domain block are tuned by an operator W_R.
W_R consists of a contrast scaling α and a luminance offset β.

c. The size of the domain block is reduced with 75 % (averaging, down sampling).

The essential operator in the scheme is W_R. α within W_R is chosen in such a way that W_R is a contraction mapping. The other operators are used to make more fits possible. W_R is an affine mapping of grey values.

$$W_R : G \mapsto \mathbb{R} \tag{7.2}$$

Given a range block R the coder searches for a domain block $D_R \subset E$ and an affine mapping W_R such that according to the l_1 metric on G:

$$d(W_R(I_D), I_R) \leq \varepsilon \tag{7.3}$$

In the scheme the above procedure is repeated for all range blocks $R \in \mathcal{R}$. Then, the original image I is by approximation a fixed point for the map W:

$$W = \bigcup_{R \in \mathcal{R}} W_R \tag{7.4}$$

By the Fixed Point Theorem the image can be restored by iterating W in the decoding phase, starting with any picture. This implies that storage of the parameters of the map W is sufficient for the (near) reconstruction of the image.

7.2.2 Quadtrees and Multiresolution

Most fractal coding schemes use a quad-tree as a further subdivision of the image. In the first stage of the coding, the image is partitioned into range blocks of fixed size. According to the tolerance ε (7.3) there will or will not be a match between a range block and a domain block. This means, there are:

1. *successes* i.e. range blocks for which an approximation by a domain block has been found and,
2. *failures* i.e. range blocks for which no approximation could be found.

The procedure in fractal coding is to subdivide the failures into four sub blocks of 1/4 size. The search for successes will then start again; now only with range blocks of 1/4 size.

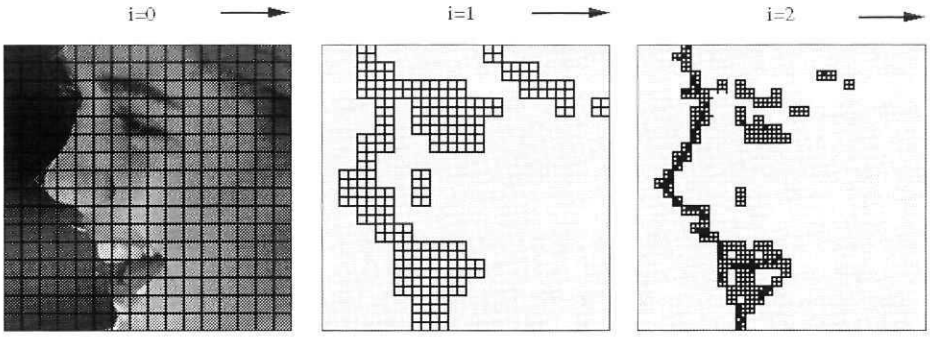

Figure 7.2: Subdivision by a quad-tree; failures at level i are illustrated at level $i+1$.

This "multiresolution" scheme is illustrated in Figure 7.2. The first level of the quad-tree, $i = 0$, contains range blocks of a fixed size, partitioning the image. The failures at a certain level i are divided into four sub blocks and illustrated at the next level $i+1$. The number of failures per level i is an important feature, we denote it by f_i. For convenience we also define s_i as the number of successes at level i.

7.3 Feature Extraction

Today there exists several implementations in multimedia database systems such as the QBIC system by IBM [8], Photo-book developed by the MIT Media Lab [10], and the Virage system developed by Virage Inc. In these, as well as many other approaches, one defines feature vectors of image properties. It is essential that such feature vectors are much smaller in size than the original images, but represent the image content as accurately as possible. Images are considered to be similar if the distance between their corresponding feature vectors, which are supposed to be elements of a given metric space, is small. For this reason, the discriminating power of the features has to be strong.
Features often used are color and texture [8, 10]. Furthermore, several authors have suggested to use shape properties [10], or relative position of objects within an image [6, 4], called spatial similarity.

7.3.1 Textural and Spatial Similarity

Texture in images has been recognized as an important aspect of human vision. Fractal image coding has good results in coding textures with the exception of statistical texture which is far from ideal [9]. However, we like to stress an important issue: compressing an image and featuring an image are quite different goals. There is no a priori reason why a transform used for indexing multimedia databases has to satisfy the same properties as one for compressing images. It can be argued that the disability of fractal coding to handle statistical texture is an advantage.

Here we are dealing with low-level feature extraction, without any segmentation. We like to show that fractal coding can model spatial info without segmentation and create several features for spatial similarity. In these features we like to express whether similarity between regions is bounded to a certain part of the image. Or whether there is a dominating direction between the blocks that are similar. The extracted information is modeled in a way that is independent of the size of the image; a very desirable item. Smaller thumb-nails can then be used to retrieve bigger images.

7.4 Feature Extraction using Fractal Codes

In our experiments the coding scheme was programmed to use five quad-tree levels. At the first step every image is divided into 16 range blocks, regardless the size of the image. At every level of the quad-tree several features will be extracted and with this more information about the image is added at every level of the quad-tree.

7.4.1 Texture

We like to distinguish three features for texture: symmetry, contrast and coarseness.

The **symmetry** feature is modeled by the operator V_{R_i}, see Figure 7.1. V_{R_i} relates a range block to one of the 8 symmetry operators that are used to bring a domain block into position to match this range block at a certain level i. In our experiments we will make histograms of several features. In the symmetry histogram, horizontally the 8 symmetry operators are denoted. The vertical axis shows the fraction by which the various symmetry operations occur at that level of the quad-tree, see Figures 7.4and 7.5.

Homogeneity of textural **contrast** is modeled by the mean and variance of the grey value scaling α. If a domain block, at a certain level i is matched with a range block, α is the scaling used on the grey values of the domain-block. Again all features are related to i.

The **coarseness** of texture is modeled by the number of successes s_i at a certain level of the quad-tree. So we count the number of ranges for which a proper domain block has been found at a certain level of the quad-tree. The depth appears to be very important in this respect. If a lot of large domain blocks can be mapped onto range blocks, the scale of the similarity will likely be coarse.

7.4.2 Spatial Similarity.

For the spatial (dis)similarity present in an image, three features are derived from the fractal code depending on the quad-tree level: uniformity, direction and dimension.

Uniformity and Direction.

The first two spatial features are modeled by one vector c_i, depending on the level i of the quad-tree. c_i is expressed in terms of its magnitude l_i and angle ϕ_i:

$$c_i = (l_i, \phi_i)$$

l_i measures the distance between a matched range and domain block. The perception is, that l_i is bounded if an image consists of different textures dividing the image into several regions. In our histograms this feature will be divided into 8 classes; length will be calculated as fraction of the distance from lower left corner to upper right corner of the image. In this way the feature is made size invariant.

The spatial **direction** feature (see Figure 7.3) measures the angle between the horizontal direction and the direction from upper left corner of the domain block to upper left corner of the range block at a certain level i. We choose to represent these features numerically by vectors $\in \mathbb{R}^8$, and graphically by histograms with eight bars, see Figures 7.4 and 7.5.

The spatial **dimension** feature relates to the *Box Counting Dimension* [3]. The image I is divided into 16^2 sub blocks. For each sub block we define:

$$d_i =^2 \log \frac{f_{i+1}}{f_i}$$

where f_i is the number of failures at level i. Figure 7.6 serves as an example. This feature distinguishes between images that have edges scattered all over the image, and images with a few clear-cut lines.

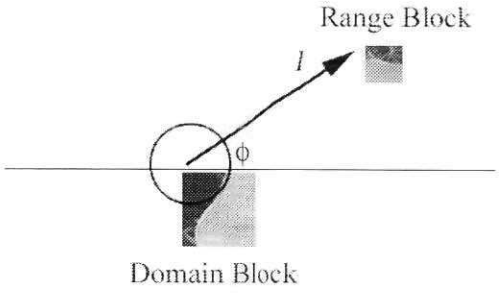

Figure 7.3: Spatial uniformity and direction.

7.5 Results

In this section we investigate the discriminating power of some features. Here we selected three features: textural symmetry, textural coarseness and spatial uniformity. The example images stem from the Vis-TeX Database of MIT. Figures 7.4 and 7.5 show 15 pictures and the corresponding histograms with respect to the selected features.

The first column shows the picture itself; the second column shows histograms related to textural symmetry; the eight values at the horizontal axis correspond to the eight symmetry operators that can be distinguished for mapping domains onto ranges. The first four values denote rotation of a square part over 0 (identity), 90, 180 and 270 degrees respectively. The second quartet denotes the same, but with an additional flip of the plane in which the image lies. The vertical axis shows the fraction by which the various symmetry operations occur at that level of the quad-tree. Although all features can be extracted at all levels, we present only the histogram which relates to the level which numbers the most successes, see Section 2.2.

We observe clearly a preference for the 0 and 180 degrees classes in the "Building" images. Other images have a much more even spread over the symmetry operations. Apparently there is a dominating direction in the picture, as could be expected. The feature appears to distinguish between images of man-made and natural environment.

The third column shows histograms with respect to textural coarseness. The horizontal axis of this histogram corresponds to the depth of refinement in the quad-tree of the encoding procedure. The vertical axis shows the accumulated success rate of the encoding. It is the

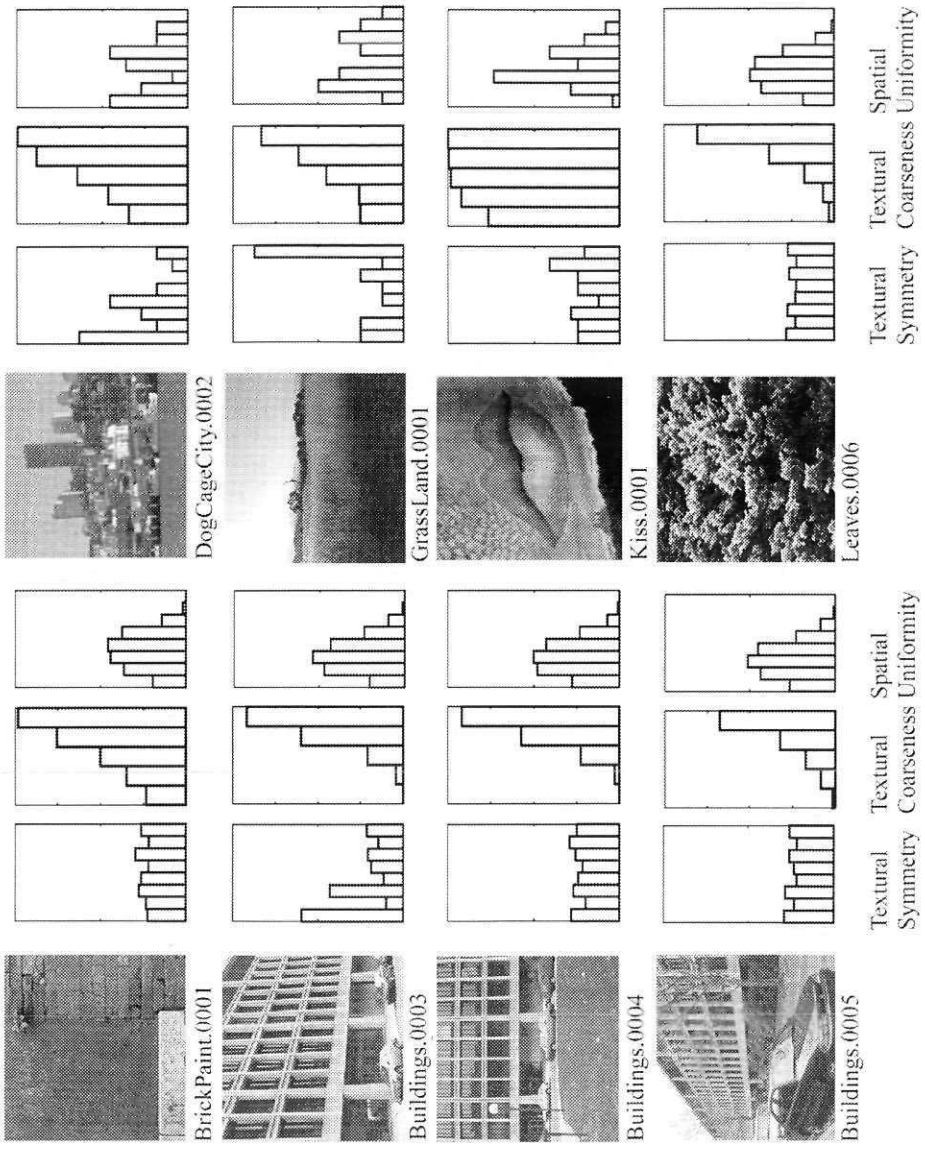

Figure 7.4: Pictures from M.I.T. Database and some features.

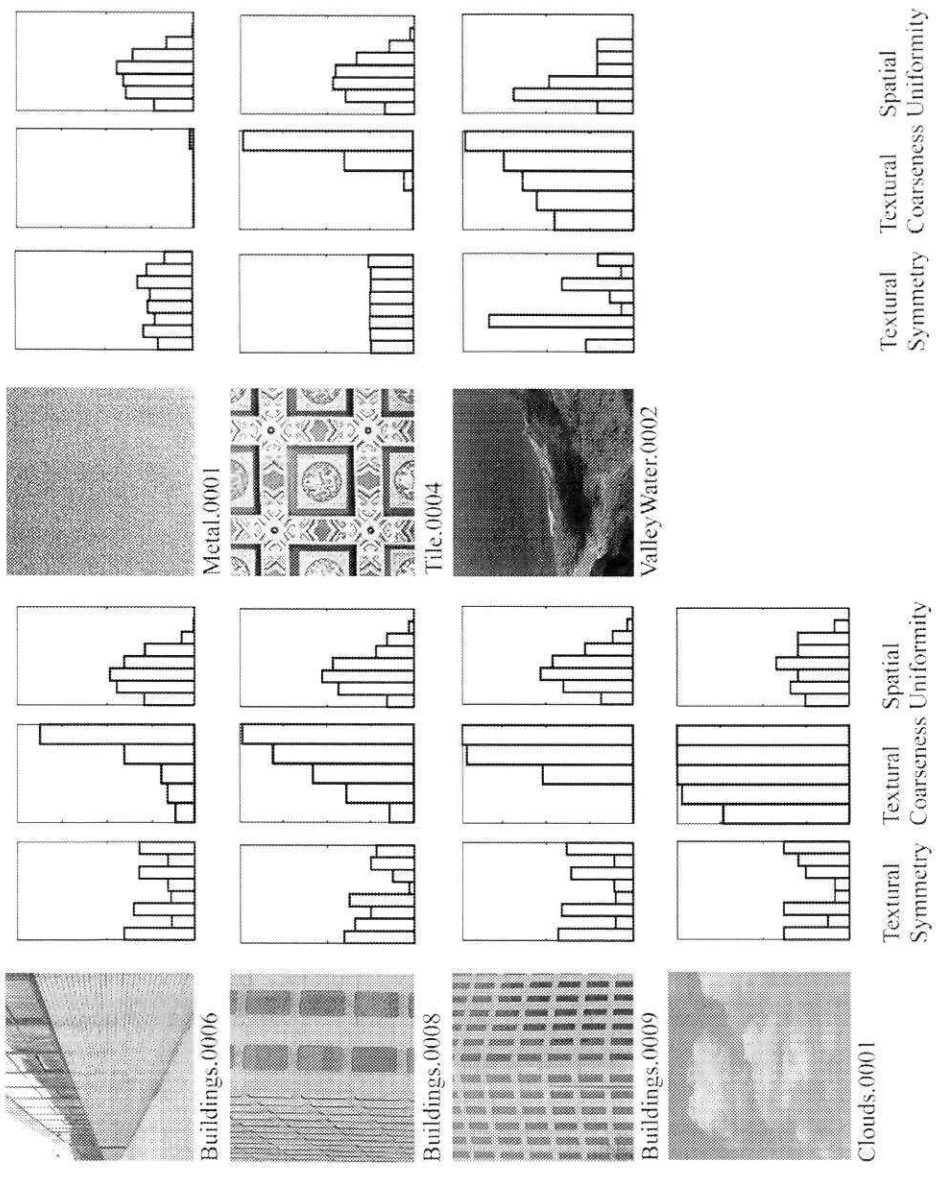

Figure 7.5: Pictures from M.I.T. Database and some features.

fraction of all pixels in the original image that are successfully mapped from domain blocks onto range blocks.

We observe how the "Metal" image is poorly matched even at large depths, due to the statistical texture present in the image. The "Clouds" image mainly seems to consist of similarity at large scale. "Kiss" which has clouds as a background almost shows the same histogram. "Buildings.0008" and "Buildings.0009" consist of building structure at a larger scale which is reflected by this feature.

The fourth column shows the histogram with respect to spatial uniformity. The horizontal axes shows 8 classes for the distance between matching domains and ranges.

"Kiss" is a nice example of an image that has texture centered in the image, which is reflected in a biased distribution with a preference for short distances. For "DogCageCity", "GrassLand" and "ValleyWater" the histogram shows two superposed distributions, corresponding to the two main textures.

Finally, we show two examples of the spatial dimension feature. Figure 7.6, exploits the fractal dimension, times 10, of parts of an image at a certain depth.

```
20 20 20 16 16 10 10 16 20 20 20 16 16 10 10 10
20 20 20 20 20 20 16 16 20 20 20 16 20 20 10 16
10 10 10 10 16 10 10 20 10 20 16 10 20 10 16
00 00 00 00 00 00 00 00 00 00 16 10 16 16 20 20
00 00 00 00 00 00 00 00 00 00 20 20 20 20 16 16
00 00 00 00 00 00 00 00 00 00 20 20 20 20 20 20
00 00 00 00 00 00 00 00 00 00 20 10 16 20 10 10
00 00 00 00 00 00 00 00 00 00 16 16 20 20 20 16
00 00 00 10 00 00 00 00 00 00 16 10 16 16 10 20
16 00 16 10 00 00 00 00 00 00 10 16 16 16 00 10
16 10 10 00 10 00 10 00 00 00 10 00 16 16 00 00
00 10 10 16 20 10 16 20 00 00 00 16 16 10 16 20
00 00 00 16 20 20 00 00 00 00 10 10 10 20 20 20
00 00 16 16 20 10 16 20 10 00 10 16 20 20 20 20
10 00 20 16 20 20 16 16 00 16 20 20 20 20 20 20
10 16 10 10 16 10 00 00 20 10 16 20 16 20 20 16
```

```
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
20 20 10 10 10 16 10 10 20 10 00 00 00 10 00 00
16 10 20 16 10 16 10 16 16 10 10 20 20 16 20 10
20 16 20 16 20 20 20 16 20 10 10 16 16 00 00 16
00 00 00 10 00 16 10 20 20 20 20 20 00 16 00 00
20 16 16 20 16 00 16 20 20 20 20 20 00 00 00 00
20 16 16 20 16 20 00 20 20 10 16 10 10 00 20 00
16 20 16 20 16 20 20 20 16 10 00 00 10 10 16 16
00 00 16 10 00 16 00 00 16 16 20 10 00 10 16 16
00 00 00 00 00 00 16 10 10 00 00 00 00 16 10 10 16
00 00 00 16 10 00 00 10 00 00 10 10 16 00 00 10
00 00 00 00 00 00 00 00 10 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 10 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 10 10 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

Figure 7.6: Spatial dimension (x10); "BrickPaint.0001" and "DogCageCity.0002" images.

Typically lines and borders yield $d_i \approx 1$ and areas with lots of inner structure yield $d_i \approx 2$. We observe how it is roughly similar to the geometry and topology of the original images and indeed borders and inner areas can be identified by different values of d_i.

7.6 Conclusions and Further Research

The features we studied appear to have discriminating power that relates to human vision. Next question is, whether this discriminating power can be used to successfully retrieve an image from a large database. We plan to address this question. The method can be used for a hierarchical search and it combines several desirable options. The first to mention is: feature extraction and compression. The second is that this method has proved to be invariant to size, orientation, contrast scalings and luminance offsets. Therefore our method may improve on previous approaches [7, 12, 14].

7.7 Acknowledgments

We like to thank Prof. M.S. Keane and Dr. H.J.A.M. Heijmans for their advice.

Bibliography

[1] G.M. Davis. A wavelet-based analysis of fractal image compression. *IEEE Transactions on Image Processing*, 7(2), 98.

[2] J.L. Dugelay, B. Fasel, V. Paoletti, and N. Vallet `www.eurecom.fr/~image/projet_etudiant_1997/english/codeur.html`.

[3] Y. Fisher. *Fractal Image Compression: Theory and Application*. Springer-Verlag, 1995.

[4] V.N. Gudivada and V.V. Raghavan. Design and evaluation of algorithms for image retrieval by spatial similarity. *ACM transactions on Information Systems*, 13(2):115–144, April 1995.

[5] A. Jacquin. *A Fractal Theory of Iterated Markov Operators with Applications to Digital Image Coding*. PhD thesis, Georgia Institute of Technology, August 1989.

[6] T. Kato. Database architecture for content-based image retrieval. In *SPIE Conf. on Image Storage and Retrieval Systems, San Jose*, volume 1662, pages 112–123, 1993.

[7] J.M. Marie-Julie and H. Essafi. Image database indexing and retrieval using the fractal transform. In Springer Verlag 1997, editor, *Proc. of Multimedia Applications, Services and Techniques*, pages 169–182, 1997.

[8] W. Niblack, R. Barber, W. Equitz, M. Glasman, D. Petkovic, P. Yanker, C. Faloutsos, and G. Taubin. The QBIC project: querying images by content using color, texture and shape. *Storage and Retrieval for Image and Video Databases*, 1908:173–187, 1993.

[9] G.E. Oien, R. Hamzaoui, and D. Saupe. On the limitations of fractal image texture coding. In *IEEE Nordic Signal Processing Symposium*, September 1996.

[10] A. Pentland, R.W. Picard, and S. Scaroff. Photobook: Tools for content-based manipulation of image databases. In Wash SPIE, Bellingham, editor, *Storage and Retrieval for Image and Video Databases*, volume II(185), pages 34–47, 1994.

[11] D. Saupe and R. Hamzaoui `ftp://ftp.informatik.uni-freiburg.de/papers/fractal/readme.html`.

[12] A.D. Sloan. Retrieving database contents by image recognition. *New Fractal Power, Advanced Imaging*, 9(5):26–30, 1994.

[13] H. Tamura, S. Mori, and T. Yamawaki. Texture features corresponding to visual perception. In *IEEE Transactions on Systems, Man and Cybernetics*, volume 8(6), 78.

[14] A. Zhang, B. Cheng, R. Achary, and R. Menon. Comparison of wavelet transforms and fractal coding in texture-based image retrieval. In *SPIE Conference on Visual Data Exploration and Analysis III, San Jose*, 1996.

Chapter 8

Fractal Transforms and Feature Invariances

Ben A.M. Schouten and Paul M. de Zeeuw [1]
Centre for Mathematics and Computer Sciences (CWI)
P.O. Box 94079, 1090 GB Amsterdam, The Netherlands
{B.A.M.Schouten, Paul.de.Zeeuw}@cwi.nl

In this paper, fractal transforms are employed with the aim of image recognition. It is known that such transforms are highly sensitive to distortions like a small shift of an image. However, by using features based on statistics kept during the actual decomposition we can derive features from fractal transforms which are invariant to perturbations like rotation, translation, folding or contrast scaling. Further, we introduce a feature invariance measure which reveals the degree of invariance of a feature with respect to a database. The features and the way their invariance is measured, appear well-suited for the application to images of textures.

8.1 Introduction

Fractals can be generated by Iterated Function Systems (IFS) [2]. In most cases, the function system, to generate the fractal, consists of a limited number of functions $\mathbb{R}^n \rightarrow \mathbb{R}^n$. The

[1] B.S. gratefully acknowledges support by NWO, The Netherlands.

domain for the functions in the system is some fixed part of $\mathbb{R}^n$. Simple variations like rotations of the fractal, lead to simple variations in the parameters of the function system. Chang [3] pays attention to the relationship of the fractal parameters (of the IFS) and some more complicated variations, like resize and relocation of the fractal. This is only done for binary deterministic fractals. The fractal transform of a *natural image* consists of a *Partial Iterated*

Table 8.1: Difference between PIFS and IFS

	Domain	Range	# Functions
IFS	Whole Image	Part of Image	Limited
PIFS	Part of Image	Part of Image	Numerous

Function System (PIFS). At the encoding, the image is subdivided into ranges, which form an non-overlapping cover of the image, see Figures 8.1, 8.2 and 8.3. For each range-block the encoder searches a different domain and a different function (affine transformation) to create the PIFS. The essential difference between IFS and PIFS, is the number of transformations in the system and the choice of the domain-blocks, see Table 8.1. Each function out of the system acts, by contrast scaling and luminance offset on the gray values of a local area of the image.

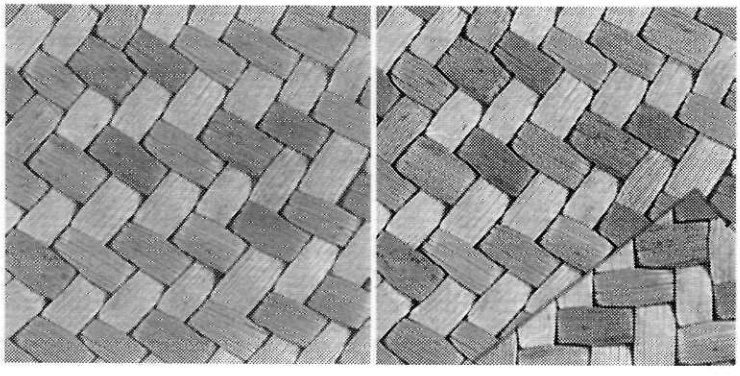

Figure 8.1: Fabric and fabric with a fold.

This paper is concerned with the use of fractal transformations as feature extractors [1, 6, 7, 8, 11, 12]. One of the reasons we want to investigate this problem is that many databases suffer from duplications. Often, similar images can be found in the database, mostly under some slightly different variations, like rotations, zooms, small translations etc. In the field of textile, for example, cloth may be presented in different folds but one still wants to recognize the texture.

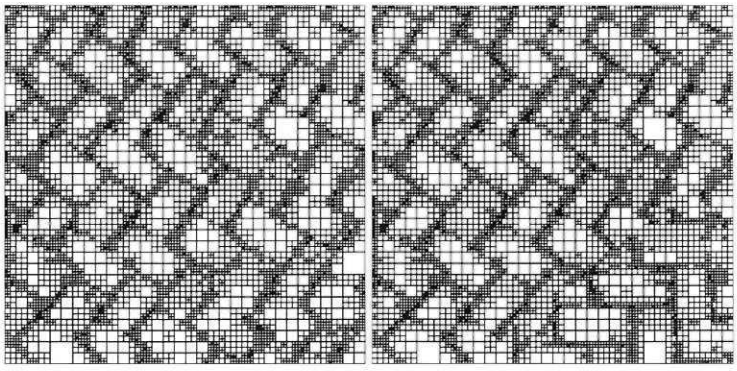

Figure 8.2: Quad-tree structure.

The assumption in this paper is that though perturbations like rotation and folding may produce quite different fractal transforms, the impact on well-chosen statistics of the transform remains limited.

We present a (computable) measure for the invariance of a feature with respect to a perturbation. Because two images can generally be expected to be neither identical nor completely dissimilar, invariance is often up to a degree.

The organization of the paper is as follows: in Section 8.2 the basics of fractal coding schemes and fractal feature extraction are presented, followed by a description of four statistical features in Section 8.3. In Section 8.4 we subject images from a database to different types of perturbations and we demonstrate our new method which identifies the perturbed images with their originals in the database. We introduce feature invariance measures. In Section 8.5 conclusions are summarized.

8.2 Fractal Feature Extraction

8.2.1 Fractal Image Coding

For completeness we give a brief description of fractal image compression (FIC) [4, 5]. Most of the feature extraction methods are based on the parameters used in FIC.
A given image is partitioned into non-overlapping range blocks, see Figures 8.2 and 8.3. The fractal encoder searches for parts called domain-blocks (which can be larger and overlapping) in the same image that look similar under some fixed number of affine transformations. Such

an affine transformation can be written as:

$$t_i(\vec{x}) = A_i\vec{x} + \vec{o}, \quad A_i \equiv \begin{pmatrix} a_{11} & b_{12} & 0 \\ c_{21} & d_{22} & 0 \\ 0 & 0 & u_i \end{pmatrix}, \tag{8.1a}$$

$$\vec{x} \equiv \begin{pmatrix} x \\ y \\ f(x,y) \end{pmatrix}, \qquad \vec{o} \equiv \begin{pmatrix} e_x \\ f_x \\ o_i \end{pmatrix}. \tag{8.1b}$$

Index i indicates the range-blocks within the image, $f(x,y)$ denotes the gray value at position (x,y). u_i is the contrast scaling and o_i is the luminance offset. The u_i and o_i are used to match the gray values of the domain with the gray values of the range-block, within the limits of an imposed accuracy ε. In practice 8 different degrees of freedom are used for A, composed of 4 rotations over $0, \frac{\pi}{2}, \pi, \frac{3\pi}{2}$, in combination with a possible flip of the domain-block along the second diagonal. In the code only the parameters of the transformations are stored. Usually, a domain-block has twice the size of a range-block.

The contractive nature of the transformations t_i makes the fractal encoder work. The transformation $T = \bigcup_{i=1}^{N} t_i$ (where N is the total number of range blocks in the image) has a fixed point which approximates the original image. It can be restored by iterating T in the decoding phase starting with an arbitrary given image. The code is producing detail at every iteration step.

8.2.2 Features and Invariances

Most of the fractal feature extractors use the parameters, discussed in the previous section, to describe the image or object [1, 7, 9]. Kouzani et al. [6] uses only the e_x, f_x parameters; Baldoni et al. [1] and Vissac et al. [12] use simplified or altered fractal schemes, only inspired by the original compression scheme. Other researchers use the behavior under decompression as a feature [8, 11].

There is a major drawback in using fractal transformations for feature extraction. The same image (attractor) can be the result of two totally different fractal transformations, making it hard to compare two images. This occurs for instance when an image is slightly translated. Invariance to small translations can be achieved by input image shifting [11]. The features can also be made invariant to scale and rotation [8, 11]. Marie-Julie [7] uses multi-resolution or multi-compression schemes in which several domain partitions are used for one image. However, all the above methods are computationally expensive. We proposed statistical analysis of the fractal parameters [9], assuming that well-chosen statistics of the different fractal transforms remain invariant. We strive for invariance with respect to folds and gloss as well (in the context of textile). In the literature no such invariances are found.

8.3 The Features

8.3.1 Introduction

As stated in the introduction we are interested in how several statistical aspects of the matching process within the fractal coding, alter by certain perturbations of the image. Here we give an outline of the features we employ, see also [9].

Most of the existing fractal coding schemes use a quad-tree structure as a subdivision of the image, see Figure 8.2. For a given accuracy ε (see Section 8.2.1), the algorithm finds a

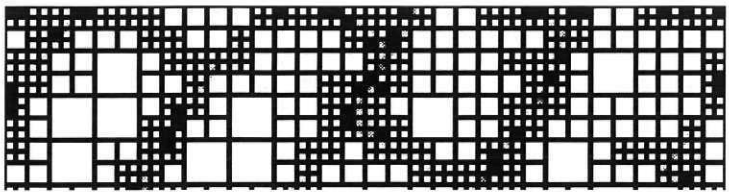

Figure 8.3: Detail of Figure 8.2, four depths i of the quad-tree are shown.

matching domain-block for the range-block in question. This is called a *success*. If there is no satisfactory match, the range-block splits into four equal parts. In this way several *depths* i of the quad-tree are created, containing range-blocks of the same size, see Figure 8.3.

We now introduce several feature histograms. Let L be the integer signifying the maximum depth imposed in the (fractal) decomposition with quad-tree refinement, likewise l signifies the minimum depth. A domain $\Omega_{l,L;k}$ is defined as:

$$\Omega_{l,L;k} \equiv \left\{(i,j) \in \mathbb{N}^2 \mid l \leq i \leq L, 1 \leq j \leq k\right\} \tag{8.2}$$

where i is associated with the depth in the quad-tree structure and k is the chosen number of *feature-bins*, see Section 8.3.2. A histogram h on $\Omega_{l,L;k}$ is defined as a function

$$h: \quad \Omega_{l,L;k} \to \mathbb{R}, \quad \text{with} \quad h \geq 0. \tag{8.3}$$

If $(i,j) \in \Omega_{l,L;k}$ then $h_{ij} = h(i,j)$ is called the *value* of h at (i,j). A histogram h on $\Omega_{l,L;k}$ is called a *(weighted) quad-tree feature histogram* if it satisfies the following additional requirements:

$$h_{ij} = w_i v_{ij}, \tag{8.4}$$

where

$$\sum_{i=l}^{L} w_i \leq 1; \quad w_i \geq 0 \tag{8.5}$$

and

$$v_{ij} \geq 0; \quad \sum_{j=1}^{k} v_{ij} = 1, \quad \forall i \in \mathbb{N} \quad \text{with} \quad l \leq i \leq L. \tag{8.6}$$

It follows at once that a (weighted) quad-tree feature histogram h satisfies:

$$\sum_{i=l}^{L} \sum_{j=1}^{k} h_{ij} \leq 1. \tag{8.7}$$

Requirement (8.6) can be interpreted as that at each depth i we have k bins v_{ij} of which the contents add up to 1. Requirement (8.5) can be interpreted as weighing the contents of the bins depending on depth i. For an interpretation of the bins see Section 8.3.2 (next).

8.3.2 Description of the Feature-bins

We use four different fractal image features to recognize a texture. The first feature is determined by the success rate (see Section 8.3.1) of the fractal decomposition at each depth in the quad-tree. The three different fractal features that follow relate to typical perceptual aspects of texture. Their definition involves the first feature.

1. *Coarseness Feature.*

   At each level i in the quad-tree we record the fraction w_i of the images area that is matched by the fractal decomposition (success). These fractions are the weights in (8.4). In case that an image has been fully resolved by fractal decomposition then the $\leq$ in both (8.5) and (8.7) turn into equal-signs. The w_i together ($l \leq i \leq L$) constitute a quad-tree feature histogram with $k = 1$ bins.

2. *Uniformity Feature.*

   When a range-block is approximated by a domain-block the spatial distance between the upper left corners can be calculated and divided by the maximum distance in the image. By splitting the interval $[0,1]$ into 8 equal intervals the above fractions are distributed over $k = 8$ different bins. Intuitively, the feature is able to detect whether one or more different textures are present in the image.

3. *Symmetry Feature.*

   The eight degrees of freedom in the affine transformations, see Section 8.2.1, yield $k = 8$ different feature-bins, constituted of rotations over $0, \frac{\pi}{2}, \pi, \frac{3\pi}{2}$ with a possible flip along the second diagonal. Intuitively, this feature is able to determine whether a texture has one or more dominating directions.

4. *Contrast Feature.*

   To match the gray values of the range-blocks by the gray values of the domain-blocks, a scaling factor u_i is used, see Section 8.2.1:

   $$t_i \circ f(x,y) = u_i \cdot f(x,y) + o_i, \quad 0 < ||u_i|| < 1.$$

   The range of this scaling factor is divided into 8 intervals, which leads to $k = 8$ feature-bins for this feature. Intuitively, the feature relates to the homogeneity of the gray values within the image.

Figure 8.4 gives examples of typical quad-tree feature histograms.

8.4 Experiments and Results

8.4.1 Introduction

Fractal feature extractors have been shown before to be effective for indexing multimedia database consisting of texture images [7, 9].

Here we demonstrate that the features as described above keep images distinguishable after they have been altered by either rotation, translation, brightness/contrast adjustment or folding. Below, we describe the details of the numerical experiment: the method of comparison, the test-case and the presentation of results.

8.4.2 Method

Each perturbed image (total of $4N$) is presented to the database for a match. We introduce an invariance measure for features w.r.t. a database D. Let a database D count N images q:

$$q_i \in D, i = 1, \ldots, N.$$

Let $p^{(n)}$ be a perturbation: an operator that perturbs an image q into an image $p^{(n)}(q)$. The collection of all perturbations of the images, is denoted by P:

$$p^{(n)} \in P, n = 1, \ldots, M.$$

Before we match images according to their features, we have to define a metric on our feature space. A quad-tree feature histogram can be interpreted as a point in $\mathbb{R}^n$ with $n \equiv k(L-l+1)$. The distance d between two quad-tree feature histograms is defined as the 2-norm of their distance in $\mathbb{R}^n$. For ease of notation we identify the image with its histogram. So $d(q_i, q_j)$ denotes the distance between the features (histograms) of image q_i and q_j.

We now define the *feature invariance measure* (FIM) with respect to the database D and perturbation $p \in P$:

$$\mu(D,p) = \sum_{i=1}^{N} \frac{d(p(q_i),q_i)}{\sum_{j=1}^{N} d(p(q_i),q_j)} \in \mathbb{R}. \tag{8.8}$$

Obviously $\mu \geq 0$ by definition of d. Typically $\mu \approx 0$ if a feature is invariant to a perturbation p. If it turns out that $\mu \approx 1$, then apparently no invariance is observed (this is the value that can be expected "at random"). The larger μ, the less invariant is the feature for the specified perturbation p. An important advantage of the FIM is its (expected) insensibility with respect to the *dimension* of the database D. This is expected because (8.8) is equivalent to:

$$\mu_i = \frac{d(p(q_i),q_i)}{\overline{d(p(q_i),q_j)}^{j}}, i = 1,\ldots,N, \tag{8.9a}$$

$$\mu(D,p) = \overline{\mu_i}^{i}, \tag{8.9b}$$

where $\overline{\mu_i}^{i}$ is the average of μ_i over $i = 1,\ldots,N$, etc.

8.4.3 Test-case

In our experiments we had the following configuration:

1. $M = 4$, that is 4 perturbations, namely: rotation, translation, brightness/contrast adjustment (b.c.a.), folds.
2. $N = 52$, that is we have 52 textures in the database.

The computation of μ requires the evaluation of N^2 distances d for each perturbation. The actual images we used are extracted from the VisTex database (MIT), supplemented with images from the Brodatz collection. It contains gray-scale images of fabrics (originals). For an illustration, see Figures 8.1, 8.5 and 8.6. The images consist of 512×512 pixels and 256 gray levels. Four quad-tree levels were taking into account, varying in size from 2×2 to 64×64. For our experiments we used the original fractal coding algorithm of Fisher [4].

8.4.4 Results

In Table 8.2 we present the results for the feature invariance measure (8.8) with respect to the example database of Section 8.4.3. We use the four features explained (and labeled

Table 8.2: Feature invariance measure

Feat.	Rotat.	Transl.	B.c.a.	Fold
1	2.4_E-3	5.3_E-2	2.4_E-1	2.3_E-2
2	6.2_E-2	1.4_E-1	3.0_E-1	9.6_E-2
3	9.7_E-2	1.5_E-1	2.8_E-1	1.4_E-1
4	2.8_E-2	1.2_E-1	2.8_E-1	7.3_E-2

1–4) in Section 8.3.2. The set P of perturbations consists of rotation (90°), translation, brightness/contrast adjustment and folding. We observe an excellent invariance if images are perturbed by either rotation, folding or translation and a moderate invariance if images are perturbed by brightness/contrast adjustment. Table 8.3 shows how well a particular feature matches a perturbed image with its original in the database. We observe how some of

Table 8.3: Percentage of successful queries

Feat.	Rotat.	Transl.	B.c.a.	Fold
1	100.0	84.6	48.1	98.1
2	100.0	92.3	51.9	100.0
3	96.2	94.2	63.5	94.2
4	100.0	98.1	67.3	100.0

the features score an optimal performance of 100% for some of the perturbations. However, the results of Table 8.3 are expected to depend on the dimension of the database in contrast with the results of Table 8.2.

8.5 Discussion and Future Research

We introduced features based on statistics stemming from fractal decompositions of images. Further we introduced a feature invariance measure which reveals the degree of invariance of a feature with respect to a database and to a perturbation. For an example database the results for this measure show that features are highly invariant to perturbations by rotation,

folding and translation and moderately invariant to brightness/contrast adjustment. Feature 4, the 'Contrast Feature' appears to be the most promising (see Table 8.3). As was conjectured before, the statistics of the fractal transform hardly change under a perturbation of the image. This fact is noteworthy. Remember that although the same image can result in two totally different fractal transform; the statistics of both transforms is shown to remain about the same.

If the features point images numerically out as close, then the images are visually close. As an example see Figure 8.5, all features of the left image point out that the right image is within a close range of similarity, which to the opinion of the authors is in accordance with visual perception.

The features didn't show any contradictory results; when an unperturbed (original) image was presented as a query for the database, all four features recognized the image without fault. As expected, the different features also express different aspects of the texture present in the image. Take a look at Figure 8.6 for instance. For the eye of the beholder the left and middle image are similar when the direction of the texture is taken into account. The left and the right image are similar when a scale aspect of the image is taken into account. The features seem to have this power of discernment.

Perturbing an image by a small shift does have a huge impact on the fractal transform of the image in question. For an example let us consider Figure 8.5. The content of these images change significantly with a small shift. In many fractal feature-applications images are scanned several times over with small shifts to remedy this. The results of our method show that irrespective of the shift of the image, still the correct image is recognized.

The influence of folding the image to the statistics of the fractal transform appears small. Moreover, if a feature relates a folded image to the wrong original, then these two image are confusingly similar, as can be observed from the results. Hitherto, folding of the fabric has been digitally simulated, we plan to perform tests on real folded texture.

We also plan to perform tests on images perturbed by realistic gloss. In our experiments we merely used brightness/contrast adjustment of the gray-values instead. Let

$$f : \mathbb{N}^2 \to [0,...,255] \subset \mathbb{N}$$

represent the gray-values of the image. A contrast scaling on the gray values results in gray-values $f^{\text{scaled}} = cf(x,y)$, where c is constant over the entire image. As this scaling is global, one would not expect the search of range-blocks and matching domain-blocks to be disturbed. However, saturation may occur at either the lower or the higher end of the scale of gray values. This effect may disturb the similarity search, depending on the image. As an example we plot the histogram of gray values of an image and the histogram of the same image, but perturbed as a consequence of a contrast scaling, see Figure 8.7. It may well be that realistic gloss proves less troublesome, a topic for further investigation.

8.5.1 Future Research

Presently we investigate whether, if an image has been altered by zooming in or out, the feature still recognizes the resulting image as similar, provided that we allow the histograms to translate along the axis of the quad-tree depth before comparison [10].

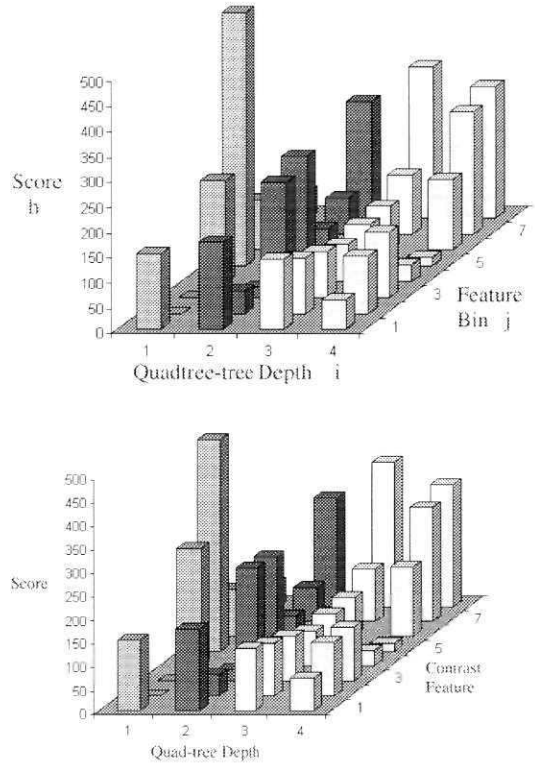

Figure 8.4: Two-dimensional quad-tree feature histograms (contrast feature) of a texture image from the database (top) and of an image of the same texture but folded (below).

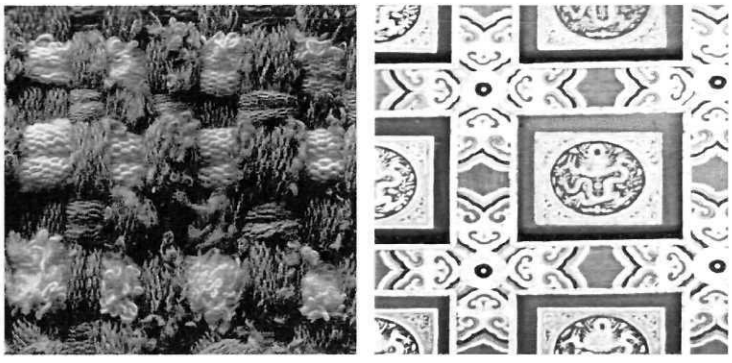

Figure 8.5: Confusingly similar images.

Figure 8.6: Different features express different aspects of similarity.

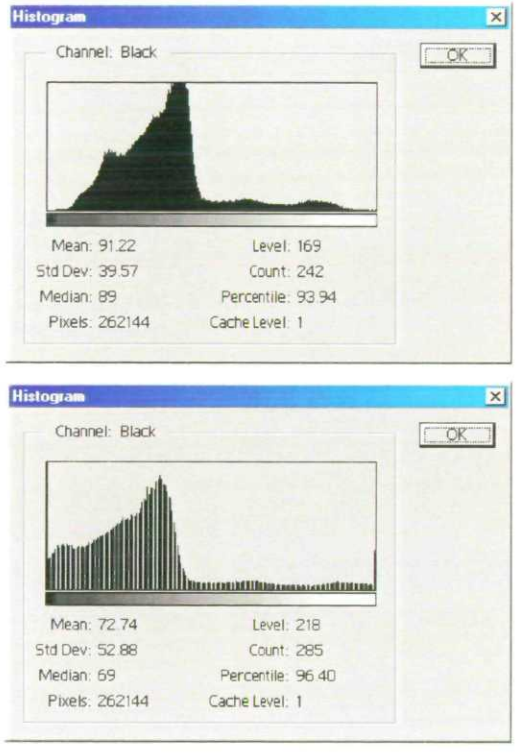

Figure 8.7: Gray-value histograms of a texture image from the database (top) and of the same image but with scaled contrast (bottom).

Bibliography

[1] M. Baldoni, C. Baroglio, D. Cavagnino, and L.Egidi. *Learning to Classify Images by Means of Iterated Function Systems*, pages 173–182. World Scientific Publishing Co. Pte. Ltd, Singapore, 1998.

[2] M. Barnsley. *Fractals Everywhere*. Academic Press, Inc., San Diego, USA, 1988.

[3] H.T. Chang, T.C. Chang, and K.L. Wen. Hierarchical fixed-point searchings method for resizing binary fractal image with ifs code modification. In *ICIP; Proceedings IEEE International Conference on Image Processing*, page CDROM, 1999.

[4] Y. Fisher. *Fractal Image Compression: Theory and Application*. Springer-Verlag, 1995.

[5] A. Jacquin. *A Fractal Theory of Iterated Markov Operators with Applications to Digital Image Coding*. PhD thesis, Georgia Institute of Technology, August 1989.

[6] A.Z. Kouzani, F. He, and K. Samut. Fractal face representation and recognition. In *IEEE International Conference on Systems, Man and Cybernetics*, pages 1609–1613, 1997.

[7] J.M. Marie-Julie and H. Essafi. Image database indexing and retrieval using the fractal transform. In *ECMAST '97; Multimedia Applications, Services and Techniques*, pages 483–492. Springer, 1997.

[8] G. Neil and K.M. Curtis. Scale and rotationally invariant object recognition using fractal transformations. In *ICASSP; Proceedings IEEE International Conference on Acoustics, Speech, and Signal Processing*, volume 6, pages 3458–3461, 1996.

[9] B.A.M. Schouten and P.M. de Zeeuw. Feature extraction using fractal codes. In *Visual 99; Visual Information and Information Systems, Third International Conference*, pages 483–492. Springer, 1999.

[10] B.A.M. Schouten and P.M. de Zeeuw. Image databases, scale and fractal transforms. In *ICIP 2000; International Conference on Image Processing*, 2000.

[11] T. Tan and H. Yan. Face recognition by fractal transformations. In *ICASSP; Proceedings IEEE International Conference on Acoustics, Speech, and Signal Processing*, volume 6, pages 3537–3540, 1999.

[12] M. Vissac, J. Dugelay, and K. Rose. A fractals inspired approach to content-based image indexing. In *ICIP; Proceedings IEEE International Conference on Image Processing*, page CDROM, 1999.

Chapter 9

Image Databases, Scale and Fractal Transforms

Ben A.M. Schouten and Paul M. de Zeeuw [1]
Centre for Mathematics and Computer Sciences (CWI)
P.O. Box 94079, 1090 GB Amsterdam, The Netherlands
{B.A.M.Schouten, Paul.de.Zeeuw}@cwi.nl

In contemporary image databases one finds many images with the same image content but perturbed by zooming, scaling, rotation etc. For the purpose of image recognition in such databases we employ features based on statistics stemming from fractal transforms of gray-scale images. We show how the features derived from these statistical aspects can be made invariant to zooming or rescaling. A feature invariance measure is defined and described. The method is especially suitable for images of textures. We produce numerical results which validate the approach.

9.1 Introduction

For the purpose of image recognition we are after feature invariance when images are either zoomed in or zoomed out. The operation of zooming-in can be seen as cropping followed by up-scaling (see e.g. Figure 9.1). The operation of zooming-out involves the addition of new

[1] B.S. gratefully acknowledges support by NWO, The Netherlands.

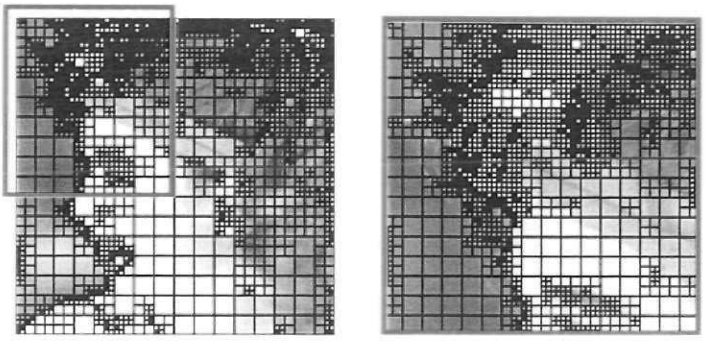

Figure 9.1: Zooming in at Teeny Image.

information, followed by down-scaling. Therefore, feature invariance appears not feasible at all for zooming-out. However, in the special case of texture-images the additional information is similar to the information already present (see e.g. Figure 9.4). In this paper we consider feature invariance for zoomed textile images.

The fractal transform of an image consists of a Partial Iterated Function System (PIFS). In a PIFS, the domain for every function in the system varies and is a part of the image itself. The number of functions in the system is large, typically hundreds. In this paper we examine the relationship of the statistical properties of the fractal functions in the system before and after zooming. Such a relationship can be used to create fractal features, invariant under scaling. The paper is concerned with the use of fractal transformations as feature extractors [1, 4, 5, 6, 10, 11].

After this introduction, Section 9.2 briefly presents the basics of fractal feature extraction, including fractal coding schemes. The choice of our features is explained in Section 9.3. In Section 9.4 we introduce a method to make the features less invariant to zooming and present results accordingly.

9.2 Fractal Features

For completeness we give a brief description of fractal image compression (FIC) [3]. Most of the feature extraction methods are based on the parameters used in FIC.

A given image is partitioned into non-overlapping range blocks, see Figure 9.1. The fractal encoder searches for parts called domain-blocks (which can be larger and overlapping) in the same image that look similar under some fixed number of affine transformations. Such an

affine transformation can be written as:

$$t_i(\vec{x}) = A_i\vec{x} + \vec{o}, \quad A_i \equiv \begin{pmatrix} a_{11} & b_{12} & 0 \\ c_{21} & d_{22} & 0 \\ 0 & 0 & u_i \end{pmatrix}, \tag{9.1a}$$

$$\vec{x} \equiv \begin{pmatrix} x \\ y \\ f(x,y) \end{pmatrix}, \qquad \vec{o} \equiv \begin{pmatrix} e_x \\ f_x \\ o_i \end{pmatrix}. \tag{9.1b}$$

Index i indicates the range-blocks within the image, $f(x,y)$ denotes the gray-value at position (x,y). u_i is the contrast scaling and o_i is the luminance offset. The u_i and o_i are used to match the gray-values of the domain with the gray-values of the range-block, within the limits of an imposed accuracy ε. Usually, a domain-block has twice the size of a range-block. The contractive nature of the transformations t_i makes the fractal encoder work. The transformation $T = \bigcup_{i=1}^{N} t_i$ (where N is the total number of range blocks in the image) has a fixed point which approximates the original image. It can be restored by iterating T in the decoding phase starting with an arbitrary given image.

9.2.1 Features and Invariances

Most of the known fractal feature extractors use the parameters, discussed in the previous section, to describe the image or object [1, 4, 5, 6, 10, 11, 7].

There is a major drawback in using fractal transformations for feature extraction. The same image (attractor) can be the result of two totally different fractal transformations, making it hard to compare two images. We proposed statistical analysis of the fractal parameters [7], assuming that well-chosen statistics of the different fractal transforms remain invariant. We strive for invariance with respect to a range of perturbations that occur in contemporary multimedia databases, like rotations, shifts, brightness adjustments [8]. In this paper, in the context of textile, zooming is considered. In the literature no such invariance is found.

9.3 The Features

9.3.1 Introduction

Here we give an outline of the features we employ, see also [7, 8]. Most of the existing fractal coding schemes use a quad-tree structure as a subdivision of the image, see Figure 9.2. For a given accuracy ε (see Section 9.2), the algorithm finds a matching domain-block for the range-block in question. This is called a *success*. If there is no satisfactory match, the range-block splits into four equal parts. In this way several *depths* i of the quad-tree are created,

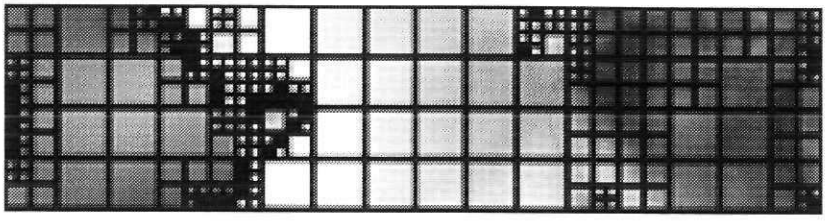

Figure 9.2: Detail of Figure 9.1, four depths i of the quad-tree are shown.

containing range-blocks of the same size, see Figure 9.2. We now introduce several feature histograms. Let L be the integer signifying the maximum depth imposed in the (fractal) decomposition with quad-tree refinement, likewise l signifies the minimum depth. A domain $\Omega_{l,L;k}$ is defined as:

$$\Omega_{l,L;k} \equiv \left\{(i,j) \in \mathbb{N}^2 \mid l \leq i \leq L, 1 \leq j \leq k\right\} \tag{9.2}$$

where i is associated with the depth in the quad-tree structure and k is the chosen number of *feature-bins*, see Section 9.3.2. A histogram h on $\Omega_{l,L;k}$ is defined as a function

$$h: \quad \Omega_{l,L;k} \to \mathbb{R}, \quad \text{with} \quad h \geq 0. \tag{9.3}$$

If $(i,j) \in \Omega_{l,L;k}$ then $h_{ij} = h(i,j)$ is called the *value* of h at (i,j). A histogram h on $\Omega_{l,L;k}$ is called a *(weighted) quad-tree feature histogram* if it satisfies the following additional requirements:

$$h_{ij} = w_i v_{ij}, \tag{9.4}$$

where

$$\sum_{i=l}^{L} w_i \leq 1; \quad w_i \geq 0 \tag{9.5}$$

and

$$v_{ij} \geq 0; \quad \sum_{j=1}^{k} v_{ij} = 1, \quad \forall i \in \mathbb{N} \quad \text{with} \quad l \leq i \leq L. \tag{9.6}$$

Requirement (9.6) can be interpreted in the way that at each depth i we have k bins v_{ij} of which the contents add up to 1. Requirement (9.4)–(9.5) can be interpreted as weighing the contents of the bins depending on depth i. For an interpretation of the bins see Section 9.3.2 (next).

9.3.2 Description of the Feature-bins

We describe two different fractal image features to recognize a texture: coarseness and contrast (we can think of more, see [8]). The definition of the second involves the first.

1. *Coarseness Feature.* At each level i in the quad-tree we record the fraction w_i of the images area that is matched by the fractal decomposition (success). These fractions are the weights in (9.4). In case that an image has been fully resolved by fractal decomposition then the "$\leq$" in (9.5) turns into an "$=$" sign. The w_i together ($l \leq i \leq L$) constitute a quad-tree feature histogram with $k = 1$ bins.

2. *Contrast Feature.* To match the gray-values of the range-blocks by the gray-values of the domain-blocks, a scaling factor u_i is used, see (9.1):

$$(t_i(\vec{x}))_3 = u_i \cdot f(x,y) + o_i, \quad 0 < ||u_i|| < 1.$$

   The range of this scaling factor is divided into 8 intervals, which leads to $k = 8$ feature-bins for this feature. Intuitively, the feature relates to the homogeneity of the gray-values within the image.

Figure 9.3 gives an example of a typical quad-tree feature histogram.

9.4 Experiments, Results and Discussion

Fractal feature extractors have been shown before to be effective for indexing multimedia database consisting of texture images [5, 7, 8, 9]. In [8] we demonstrated the invariance of the features, including those from Section 9.3.2 for rotation, translation, folding and brightness adjustments. Here we show that if an image is altered by zooming, the feature still distinguishes between images, especially if we allow the histogram to shift along the axis of the quad-tree depth before comparison.

9.4.1 Method

Each perturbed image is compared to all members of the database. Below we employ an invariance measure for features with respect to a database D. Let a database D count N

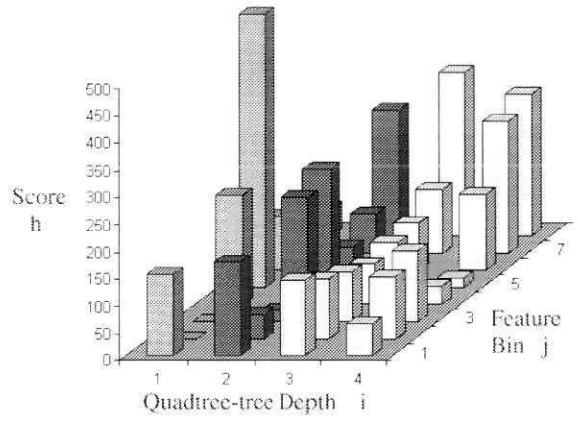

Figure 9.3: Two-dimensional quad-tree feature histogram (contrast feature).

images q:

$$q_i \in D, i = 1, \ldots, N.$$

Let $p^{(n)}$ be a perturbation: an operator that perturbs an image q into an image $p^{(n)}(q)$. The collection of all perturbations of the images, is denoted by P:

$$p^{(n)} \in P, n = 1, \ldots, M.$$

A quad-tree feature histogram can be interpreted as a point in $\mathbb{R}^n$ with $n \equiv k(L-l+1)$. The distance d between two quad-tree feature histograms is defined as the 2-norm of their distance in $\mathbb{R}^n$. So $d(h(q_i), h(q_j))$ denotes the distance between the feature histograms of images q_i and q_j. We denote

$$d_{ij} = d(h(p(q_i)), h(q_j)) \in \mathbb{R}.$$

We introduce a measure for feature invariance as follows. Firstly, for an entry q_j from the database D we compute d_{ij}, $i = 1, \cdots, N$ for a given perturbation $p \in P$. Secondly, we list d_{ij} in order of decreasing size. Thirdly, let $r_j = r(q_j, D)$ be the ranking number of d_{jj} in the list. That is, $r_j = 0$ is the best possible result and $r_j = N-1$ the worst possible. We now define the *absolute feature invariance measure* (AFIM) with respect to the database D and perturbation $p \in P$:

$$\nu(D, p) = \left(1 - \frac{\sum_{j=1}^{N} r_j}{(N-1)N}\right) \cdot 100. \tag{9.7}$$

If $\nu = 100$ this implies that all queries (perturbed images) are recognized without fault by the feature.

9.4.2 Results

As for perturbation of an image we confine ourselves to zooming procedures. In Table 9.1 we present the results for the feature invariance measure with respect to the database ($N = 52$). Columns 2 & 4 (No Shift) correspond to the straightforward computation according to (9.7).

Table 9.1: Absolute feature invariance measure.

p	coarseness feature		contrast feature	
zoom	No Shift	Shift $\leq \frac{1}{2}$	No Shift	Shift $\leq \frac{1}{2}$
out	89.0	93.7	97.4	97.6
in	82.3	93.0	93.6	98.1

Columns 3 & 5 (Shift $\leq \frac{1}{2}$) correspond to the case that the distance between two histograms $d(h(q'),h(q))$ is not computed as is but as

$$\min_{T \in \mathcal{T}} d(h(q'),T(h(q)))$$

instead, where T represents an operator that shifts a histogram along the axis of the quad-tree depth. We allow a histogram to shift over a maximum distance of $1/2$ in both the positive and negative direction (this defines $\mathcal{T}$). This corresponds to a zooming factor $2^{1/2}$ (in and out) of an image. We observe from Table 9.1 that the feature invariance benefits from taking the above shifts into account. The contrast feature appears to be very robust, even so without the shifting technique.

9.4.3 Discussion

In this paper, fractal transforms are employed with the aim of image recognition in multimedia-databases. We use features based on statistics stemming from fractal decomposition of images. We demonstrate that feature invariance with respect to zooming benefits from shifting the feature histogram. The thought behind our method is that the scales of texture in an image match with quad-tree depths in the fractal decomposition, and determine the outcome of the decomposition at each depth accordingly. That's why statistical aspects from an image move from one (quad-tree) depth to another when an image is zoomed, see Figure 9.5. This effect is compensated for by shifting the histogram into the opposite direction.

We finish up with an example, see Figure 9.6. The image Cloth139 (left) is presented to the database. Without compensating for zooming effects, the image Brick.000 (middle) is retrieved. The scale of both textures is confusingly similar. However, our shifting technique compensates for such effects and the original image (right) is retrieved. The features as described relate well to human perception [7, 8] and will be used for visual intelligence retrieval systems [2].

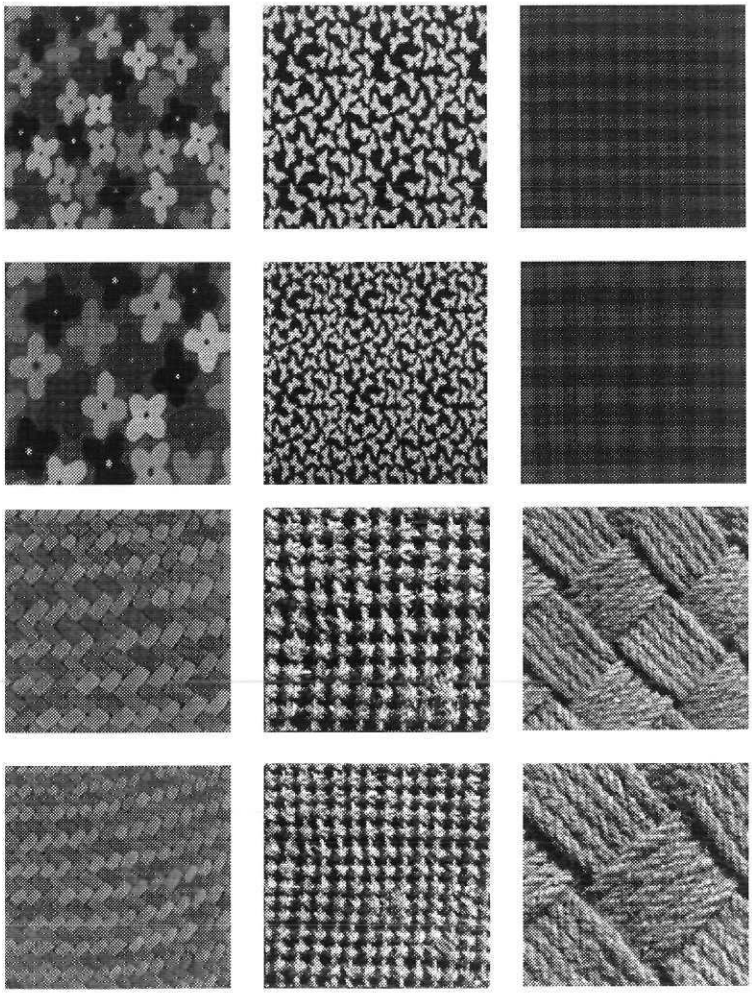

Figure 9.4: Examples from the database (excerpts from VISTEX and Brodatz Collection) and the zoomed versions.

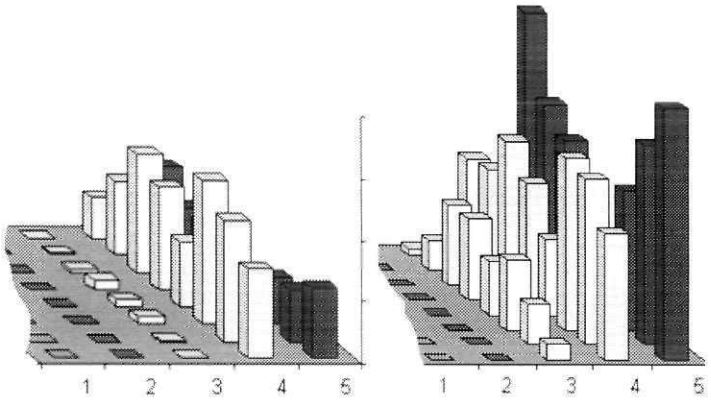

Figure 9.5: Quad-tree histograms of original image (left) and zoomed-in image (right), $l = 1$, $L = 5$.

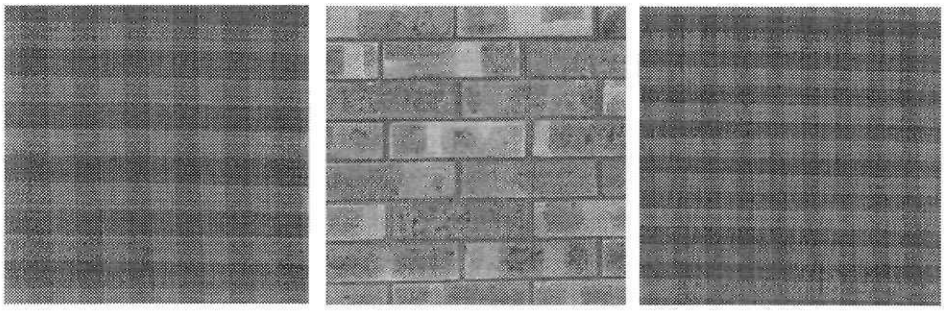

Figure 9.6: An image (left) is presented to the database. Without compensating for zooming a confusingly similar image (middle) is retrieved. After compensating the correct image (right) is retrieved.

Bibliography

[1] M. Baldoni, C. Baroglio, D. Cavagnino, and L.Egidi. *Learning to Classify Images by Means of Iterated Function Systems*, pages 173–182. World Scientific Publishing Co. Pte. Ltd, Singapore, 1998.

[2] G. Caenen, G. Frederix, A.A.M. Kuijk, E.J. Pauwels, and B.A.M. Schouten. Show me me what you mean. pariss: a cbir interface that learns by example. In *Visual 2000: Fourth International Conference on Visual Information Systems, Lyon France*, 2000.

[3] Y. Fisher. *Fractal Image Compression: Theory and Application*. Springer-Verlag, 1995.

[4] A.Z. Kouzani, F. He, and K. Samut. Fractal face representation and recognition. In *IEEE International Conference on Systems, Man and Cybernetics*, pages 1609–1613, 1997.

[5] J.M. Marie-Julie and H. Essafi. Image database indexing and retrieval using the fractal transform. In *ECMAST '97; Multimedia Applications, Services and Techniques*, pages 483–492. Springer, 1997.

[6] G. Neil and K.M. Curtis. Scale and rotationally invariant object recognition using fractal transformations. In *ICASSP; Proceedings IEEE International Conference on Acoustics, Speech, and Signal Processing*, volume 6, pages 3458–3461, 1996.

[7] B.A.M. Schouten and P.M. de Zeeuw. Feature extraction using fractal codes. In *Visual 99; Visual Information and Information Systems, Third International Conference*, pages 483–492. Springer, 1999.

[8] B.A.M. Schouten and P.M. de Zeeuw. Fractal transforms and feature invariance. In *Procs. 15th International Conference on Pattern Recognition (ICPR 2000)*, pages 992–997. IEEE Computer Society, 2000.

[9] B.A.M. Schouten and P.M. de Zeeuw. Image databases, scale and fractal transforms. In *ICIP 2000; International Conference on Image Processing*, 2000.

[10] T. Tan and H. Yan. Face recognition by fractal transformations. In *ICASSP; Proceedings IEEE International Conference on Acoustics, Speech, and Signal Processing*, volume 6, pages 3537–3540, 1999.

[11] M. Vissac, J. Dugelay, and K. Rose. A fractals inspired approach to content-based image indexing. In *ICIP; Proceedings IEEE International Conference on Image Processing*, page CDROM, 1999.

Chapter 10

Pariss, an Interface that Learns by Example

G. Caenen[2], G. Frederix[2,3], A.A.M Kuijk[1], E.J. Pauwels[1,2] and B.A.M. Schouten[1]

1. *Centre for Mathematics and Computer Science (CWI), Kruislaan 413, 1098 SJ Amsterdam, The Netherlands*
   {Eric.Pauwels, B.A.M.Schouten, Fons.Kuijk}@cwi.nl
2. *ESAT-PSI, K.U.Leuven, K. Mercierlaan 94, B-3001 Heverlee, Belgium*
   Geert.Caenen@esat.kuleuven.ac.be
   Eric.Pauwels@esat.kuleuven.ac.be
3. *Dept. of Mathematics, K.U.Leuven, Celestijnenlaan 200 B, B-3001 Heverlee, Belgium*
   Greet.Frederix@esat.kuleuven.ac.be

Keywords: Visual query browser, relevance feedback, incremental learning, assisted feature generation.

We outline the architecture of a CBIR-interface that allows the user to interactively classify images by dragging and dropping them into different piles and instructing the interface to come up with features that can mimic this classification. Logistic regression and Sammon projection are used to support this search mode.

10.1 Introduction and Motivation

The explosive growth of digital multi-media repositories has created challenging new problems regarding indexing, access and retrieval of information. These challenges are particularly acute for image-databases as there is no canonical format for encoding the information encapsulated in an image. It is the explicit goal of *Content-Based Image Retrieval* (CBIR) to design algorithms and interfaces that will assist the user in this task [2, 5].

The **aim of this paper** is to outline the architecture of an interface that allows the user to interactively guide the search by manually rearranging or classifying images. A first version of this interface was introduced in [3], and baptized PARISS, short for *Panoramic, Adaptive and Reconfigurable Interface for Similarity Search.* This lengthy acronym refers to the following interface-characteristics:

- *Panoramic:* The relative location of images with respect to the rest of the database can be displayed;
- *Adaptive:* Relevance feedback (in the form of examples and counter-examples) is used to incrementally refine the probability measure that represents the accumulation of information during the search process;
- *Reconfigurable:* Similarities can be defined interactively through direct manipulation of images.

The idea of using a manipulation tool to define similarities interactively was first introduced by Santini [6]. However, his approach is based on modifying the Euclidean metric into a general Riemannian one to absorb the discrepancies between the user-defined similarities and the distance between the actual feature-vectors. The methodology proposed in this paper differs in that it concentrates on transforming the features themselves rather than the metric.

Formal statement of the problem We assume that every one of the N images is represented by a K-dimensional feature-vector. In that sense, the database is represented by a $N \times K$ matrix where each row represents the numerical features of the corresponding image.

In abstract terms the CBIR-problem can be looked upon as an optimization problem for a function Φ that maps each image (or its feature-vector $\mathbf{x}$) to its user-defined numerical *relevance*. This relevance reflects the extent to which an image corresponds the user's goal, and for ease of argument we will assume that it can be assigned a numerical score ranging from *highly relevant* (1) to *not relevant at all* (-1):

$$\Phi : \mathbf{x} \in \mathbb{R}^K \longmapsto \text{relevance} \in \mathbb{R}$$

When conducting a *similarity* search Φ will be inversely proportional to the distance between the query image and the rest of the database; when *browsing* a catalogue looking for something interesting or beautiful, Φ will simply reflect the image's appeal. Clearly, this function is user-dependent, and each function-evaluation involves visual inspection of the image by the user.

Somewhat confusingly perhaps, we will refer to images as *relevant* for short, if they are either *highly relevant* ($\Phi \approx 1$, i.e. excellent examples of what we mean) or *highly irrelevant* ($\Phi \approx -1$, i.e. good counter-examples). For instance, if we are looking for images that are brightly red, then images that are predominantly red are highly relevant examples. However, images that are strikingly blue are also relevant in that they furnish the user with excellent counter-examples. Finally, an image exhibiting red patches could be said to be *partially relevant* as some of its aspects are informative to our query. The problem of extracting information from partially relevant images will be taken up in section 10.4 .

Different Search Strategies The role of the interface is to use the available information (i.e. function evaluations based on visual inspection) and suggest to the user potentially interesting images for further evaluation. In mathematical parlance this amounts to a stepwise optimisation of Φ — a problem of considerable difficulty as our knowledge of the function is restricted to the few points at which it is evaluated (by visual inspection). To address such a problem, there are basically two strategies, both implemented in the current version of the interface (for more details we refer the reader to sections 10.2.2 and 10.2.3).

1. **Probabilistic search:** In this approach, the search space is sampled at different locations and the function evaluations are used to bias the next sample in an attempt to increase the probability of a high Φ-yield. More specifically, the next batch of points that are selected for evaluation will be made to cluster roughly about the point that yielded the best Φ-value.

   To enable this sort of probabilistic search in PARISS, we have implemented a **collection box** — for want of a better word — in which the user can collect relevant images. This collection of relevant images is continuously analysed by a module called the **inference engine** (see below) in an attempt to spot trends in the features-values that can direct the sampling procedure to more promising regions of the search space.

2. **Gradient ascent:** In gradient methods one determines how the function is changing near the current location and this gradient is then used to move to higher Φ-values. To assist the user in finding perceptually meaningful gradients we have designed a **manipulation window** that allows him to manually rearrange images according to his own appreciation of their relative similarity. Once this is done, the **projection engine** searches for a projection of the dataset that best reproduces this requested configuration. This form of manipulation allows the user to impose certain gradients by enforcing how the visual qualities change when moving from image to image.

In the next section we will elaborate in more detail how the interface has been designed to support a seamless combination of these two search modi.

10.2 The Interface's Architecture

10.2.1 Introduction

Even medium-size image databases contain at least several thousand images. The complexity of the retrieval problem is further compounded by the fact that, in order to capture the visual content, one needs to extract quite a large number (K) of features. Values for K ranging between 50 and 200 are typical, rather than exceptional. In the terminology introduced earlier this means that we are faced with the challenge of navigating through a large cloud of data-points in the high-dimensional space $\mathbb{R}^K$.

These astronomical sizes contrast starkly with the small number of images (20 to 50) that can simultaneously be displayed on screen. The fraction of the database that one is exploring at any given time is therefore tiny, and this myopic view entails that it is very easy to lose one's bearings while exploring the database. The proposed interface was designed to alleviate these problems and in what follows, we will therefore outline its architecture which, for the sake of clarity, is divided in **displays** and **computation engines**. The former are used to display images for different forms of relevance feedback, while the latter are invoked to translate the user's input into new search directions.

10.2.2 Display Windows

At all times, the interface shows three display-windows, between which the user can move seamlessly (for a schematic overview we refer the reader to Fig. 10.1).

1. **Sample display**

   This screen displays the by now standard matrix of images that are (initially randomly) sampled from the database and presented to the user for inspection. The user can select images that are deemed relevant whereupon they are transferred to the *collection box* (see below). Each time a "refresh button" is pressed, the sampling algorithm is activated and a new sample is generated for inspection. The sampling algorithm that is used to generate the new sample can be biased by the *inference engine* (cfr. section 10.2.3) to accommodate the preferences of the user.

2. **Collection box for relevant images**

The second screen is used as a simple *collection box* in which there are two bins: one for the *examples* (i.e. similar or partially similar images), and one for the *counter-examples* (dissimilar images). Whenever the user comes across an image he considers relevant in that respect, it is transferred to the collection box. This box can be inspected at all times, and images that no longer seem relevant can be removed.

The collection box should be thought of as reflecting the user's cumulative (qualitative) knowledge about the database. This information will be turned into more quantitative measures by the *inference engine* (see section 10.2.3).

3. **Manipulation window**

This screen is used to manually redefine similarities between selected images. It therefore shows an xy-plot in which selected images (e.g. the ones that have been transferred to the *collection box*) are presented as thumbnails located at appropriate 2-dimensional xy-coordinates. The precise choice of these coordinates is altogether not very important as they will be changed during the search process, but to fix ideas we suggest to use the first two *principal component* coordinates (PC-coord) [1] as an initial choice.

The really interesting feature of this window is that it can be **manipulated**. More specifically, when inspecting the displayed projection, the relative positioning of the images might strike the user as unsatisfactory. For instance, it might be the case that although image A is located near image B and quite far from image C, it's the user's understanding that this should be the other way round. He can then drag the thumbnail B to a location near C.

After rearranging the images, he can then instruct the interface to find a **transformation** that will project the original feature-space onto the screen in such a way that the resulting configuration better resembles the one that was manually defined (the computational details are explained in Section 10.2.3). The underlying rationale is that the manually defined arrangement of the images will reflect the preferences and tendencies that implicitly exists in the user's mind. Hence, constructing (new) features that are able to reproduce this configuration will probably reorganise the database along the same lines, and suggest directions (gradients if you will) in which to search next. For instance, if images with a fine-grained texture are dragged to the left part of the screen, while coarse-grained images are collected in the right part, one expects to find medium-grained images when exploring the middle part of the screen.

To enable this type of exploration, the manipulation window is also **clickable**: you can click at *any position* in the display (i.e. not just on an already displayed image)

[1] Once all the K features have been determined for the N images in the database, one can compute the $K \times K$ covariance matrix and its eigenvectors. Using these eigenvectors as new coordinate system one can recompute the coordinates of all the points with respect to this new coordinate system. We suggest to use the coordinates that correspond to the eigenvectors associated with the two largest eigenvalues.

and the interface will look for the images the 2D-coordinates of which are closest to the selected point. If you consider any of these to be relevant, then you can add them to the *collection box*.

10.2.3 Computation Engines

Let us now take a peek under the bonnet and explain in some more detail the computational strategies that enable this type of interactivity.

1. **Projection Engine: Creating New Feature Combinations**

   The projection engine operates on the *manipulation window* and is activated to generate a transformation (A say) that yields features better conforming to the user's appreciation of the similarities. To keep things as simple as possible, we will assume that the transformation is *linear*, mapping the full feature space $\mathbb{R}^K$ onto a 2-dimensional display ($\mathbb{R}^2$):

   $$A : \mathbb{R}^K \longrightarrow \mathbb{R}^2$$

   A gradient descent method is used to determine the actual form of the linear transformation A and we will presently explain in more detail how we go about this. The reader who is not interested in the mathematics can proceed directly to the description of the *inference engine*.

   (a) Start by selecting a number (n say) of relevant or representative images in the database (e.g. images that have been amassed as examples or counter-examples in the collection box during an earlier exploration stage). These images are represented by their n feature-vectors $\mathbf{x}_i \in \mathbb{R}^K$.

   (b) Project the corresponding data-points onto a 2-dimensional subspace (selected manually or automatically) and show the result to the user (see also Fig.10.1). As explained above, this resulting 2-dimensional projection is displayed in the *manipulation window*, in which the images are shown as thumbnails, each positioned at a location that corresponds to their projected (2D) coordinates $\mathbf{q}_i$ $(i = 1, \ldots, n)$.

   (c) Next, allow the user to rearrange the thumbnails so that their new ("target") configuration $\mathbf{t}_i$ reflects more accurately the perceptual organization as perceived by the user. Once this is done, *the projection engine* is activated which attempts to find a linear mapping

   $$A : \mathbb{R}^K \longrightarrow \mathbb{R}^2$$

   (called *projection* for short) of the full feature space $\mathbb{R}^K$ onto a 2-dimensional space that, when applied to the selected points $\mathbf{x}_i$, best matches the user-defined configuration of the points $\mathbf{t}_i$.

The optimality is expressed with respect to *Sammon's metric stress* which is defined as:

$$S_M(A) = \sum_{i,j} \frac{(d(\mathbf{t}_i,\mathbf{t}_j) - d(A\mathbf{x}_i,A\mathbf{x}_j))^2}{d(\mathbf{t}_i,\mathbf{t}_j)}, \tag{10.1}$$

where d is the standard Euclidean metric on $\mathbb{R}^2$.

This Sammon-criterion can be minimized using *gradient descent.* Indeed, straightforward but tedious algebra shows that

$$\begin{aligned}
\frac{\partial S_M}{\partial a_{kl}} &= 2\sum_{i,j}\left(1 - \frac{d(A\mathbf{x}_i,A\mathbf{x}_j)}{d(\mathbf{t}_i,\mathbf{t}_j)}\right)\cdot\frac{\partial d(A\mathbf{x}_i,A\mathbf{x}_j)}{\partial a_{kl}} \\
&= 2\sum_{i,j}\left(1 - \frac{d(A\mathbf{x}_i,A\mathbf{x}_j)}{d(\mathbf{t}_i,\mathbf{t}_j)}\right)\cdot\frac{[A(\mathbf{x}_i-\mathbf{x}_j)]_k[\mathbf{x}_i-\mathbf{x}_j]_l}{d(A\mathbf{x}_i,A\mathbf{x}_j)} \\
&= 2\sum_{i,j}\left(\frac{d(\mathbf{t}_i,\mathbf{t}_j) - d(A\mathbf{x}_i,A\mathbf{x}_j)}{d(\mathbf{t}_i,\mathbf{t}_j)\,d(A\mathbf{x}_i,A\mathbf{x}_j)}\right)\cdot\left[A(\mathbf{x}_i-\mathbf{x}_j)(\mathbf{x}_i-\mathbf{x}_j)^T\right]_{kl}
\end{aligned}$$

Hence, the gradient can be expressed explicitly as

$$\nabla_A S_M = 2A\sum_{i,j}\left(\frac{d(\mathbf{t}_i,\mathbf{t}_j) - d(A\mathbf{x}_i,A\mathbf{x}_j)}{d(\mathbf{t}_i,\mathbf{t}_j)d(A\mathbf{x}_i,A\mathbf{x}_j)}\right)(\mathbf{x}_i-\mathbf{x}_j)(\mathbf{x}_i-\mathbf{x}_j)^T \tag{10.2}$$

with corresponding gradient descent dynamics: $\partial A/\partial t = -\varepsilon\nabla_A S_M$.

(d) Once the optimal transformation A has been found it is applied to all the elements in the database, i.e. not just to the original sample of size n on which the computation is based. This allows the user to have a fresh look at the database which now better reflects the user-defined (dis)similarity structure.

Extensions to quadratic (or higher order) transformations The use of a linear transformation immediately suggests the extension to quadratic and higher order transformations. However, such extensions are less straightforward as they may appear at first sight, as the number of parameter increases dramatically (exponentially) when the order of the transformation is augmented.

We therefore propose to restrict the use of the higher order transformations to instances where further fine-tuning of the 2-dimensional display is called for. Hence, the mapping $A : \mathbb{R}^K \longrightarrow \mathbb{R}^2$ maps the full feature space onto a 2-D display space, whereupon a quadratic transformation $Q : \mathbb{R}^2 \longrightarrow \mathbb{R}^2$ is invoked to model the non-linear aspects in the approximation. This quadratic transformation is of the form $Q(\mathbf{y}) = \eta \in \mathbb{R}^2$ where $\eta = (\eta_1,\eta_2)$ satisfies:

$$\begin{cases} \eta_1 &= \mathbf{y}^T\mathbf{Q}_1\mathbf{y} + \mathbf{p}^T\mathbf{y} + \alpha \\ \eta_2 &= \mathbf{y}^T\mathbf{Q}_2\mathbf{y} + \mathbf{q}^T\mathbf{y} + \beta \end{cases}$$

It's important to notice that these equations are *linear* in the unknown parameters and optimisation we used for the original projection engine can therefore easily be adapted for this more complicated case.

2. Inference Engine: Biasing the Sample

As explained above, the *collection box* is used to collect both examples and counter-examples of the sort of images the user deems relevant. The reason for collecting them is to be found in the fact that this information can be used to bias the next sample favorably so that the fraction of interesting retrieved images increases over time. To this end we have implemented an *inference engine* that uses these data to generate an estimate of where interesting images can be found.

To cast this in a more formal setting, we denote by p the probability that an image is relevant. Hence, $p \approx 1$ would mean that it is highly similar, whereas $p \approx 0$ indicates that it is highly dissimilar. Strictly speaking, the probability measure p depends on the full feature vector $\mathbf{x} = (x_1, x_2, \ldots, x_K)$ and the images gathered in the collection box yield information on locations where the probability is markedly high or low. However, it is clear that reliably modelling the full probability density $p(\mathbf{x})$ on such scant information will in general prove to be an intractable problem, so we do the next best thing and model p as a function of each feature x_i separately.

Predicting similarity using logistic regression In mathematical terms the problem boils down to this: the collection box contains a number of examples and counter-examples and for each single feature x_i (denoted x for short) we want to model the dependency of p on x through a function $p(x)$. The simplest case would be the one in which one can find a threshold value ($x^{(0)}$ say) that separates examples from counter-examples. Such information could then be fed back to the sampling procedure. However, in most cases the situation will be more complicated and trends will be less clear-cut. The best one can hope for is that one can correlate the probability p with the x-values so that trends become visible and can be harnessed to improve the efficiency of the search.

The standard way to handle such a situation is to invoke a *logistic regression model* (see [4])

$$\log \frac{p(x)}{1-p(x)} = f(x) \tag{10.3}$$

where the logit-ratio of p in the left-hand side[2] is expressed as an appropriate function of the feature-value x. For the application we have in mind, we have opted for a *quadratic* logistic regression model:

$$\log \frac{p(x)}{1-p(x)} = \alpha x^2 + \beta x + \gamma; \tag{10.4}$$

The parameters α, β and γ are determined by using *maximum likelihood estimation*, i.e. they optimize the probability of the actual configuration occurring. More explicitely, if we look up the x-value for each of the images in the collection box and then use eq.(10.4) to compute the probability p that they are in fact an example ($p > 1/2$) or counter-example ($p < 1/2$), then the parameters (α, β, γ) are chosen to optimize this prediction accuracy.

The reason for the choice of a *quadratic* function might need some clarification. If relevance is (directly or inversely) proportional to the feature value, then a linear model will suffice (i.e. we can put $\alpha = 0$); however there obviously are situations where for instance, only medium feature-values are acceptable, while extreme values (both larger and smaller) are unacceptable (have a look at Fig. 10.3). The quadratic model is the simplest model that can handle this sort of qualitative distinction.

Using regression diagnostics The use of regression has the additional advantage that we can invoke standard regression diagnostics to judge the fit and predictive power of the model. This helps us to gauge the success of the feature in predicting relevance and can therefore be used to narrow down the feature-set. More precisely, if for a particular feature x_i, the prediction of the fitted model (10.4) fails to square up with the relevance feedback from the user, this indicates that that particular feature x_i does not feature prominently in the perceptual appreciation of the user. Hence, a uniform sampling regime for that feature is advisable, as there is no reason to narrow its sampling-range.

Conversely, if logistic regression yields a well-fitting model for a different feature x_j, we can conclude that the feature plays an important role in the user's appreciation of the image, and we are well advised to bias the sampling-procedure as to favour feature-values x_j that have high p-value. That way, the fraction of relevant features in each new sample (as displayed on the *sample display*) will gradually increase.

[2]The logit-ratio $\log(p/(1-p))$ is introduced to transform the p-value, which is constrained to the interval $[0, 1]$, to a quantity that ranges over $\mathbb{R}$ and is therefore easier to link to a linear or polynomial regression model.

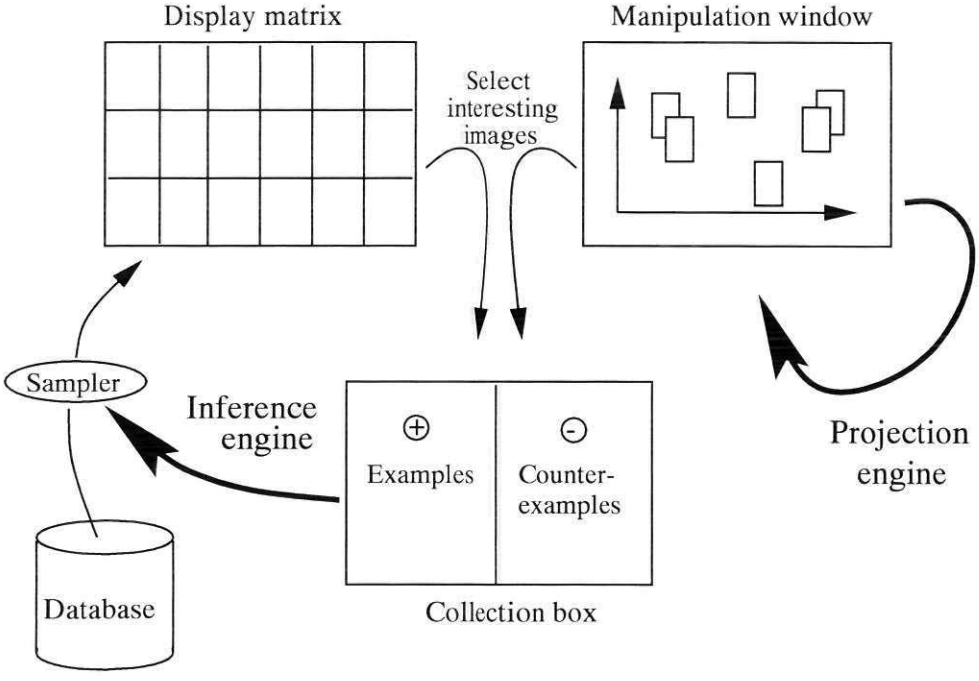

Figure 10.1: Images are sampled from the database and presented for inspection on the *sample display* (top left). The ones relevant for the search (examples or counter-examples) are transferred both to the *collection box* (bottom) and the *manipulation window* (top right). Here their locations reflect their similarity. Moreover, the user can rearrange them to better match his similarity-perception and instruct the *projection engine* to compute features that match this ordering. Statistics of the images in the collection box are used by the *inference engine* and used to bias the next sampling procedure.

10.3 Using the Interface: An Interaction Scenario

After the detailed description of the different components that constitute the interface in the preceding paragraphs, we will now wrap up this section by concisely outlining what an interactive browsing session might look like.

The query-scenario we have in mind is one in which the user has a (more or less crisply defined) mental picture of a *target image* by which he can judge the relevance of the images encountered during the database exploration. For ease of argument we will assume that in a first exploratory stage, the user has been shown random (or representative) samples drawn from the database and that he has selected a number of images that in some respect are considered relevant for his search.

The database search now proceeds by mixing and iterating the following basic operations (for an animated version of this slightly complicated process, we refer to our website):

1. *Browse* the database by either inspecting the sampling display or by clicking in promising regions of the manipulation window;
2. *Add* interesting images to the *collection box* and *remove* the ones that are no longer relevant;
3. *Generate* "home-made" features by rearranging selected thumbnails and requesting the interface to come up with *new combinations of features* that are able to reproduce the user-defined ordering.
4. *Bias* the sampling procedure by fitting regression models to the underlying probability distribution.

10.4 Addressing the Problem of Partial Relevance

The learning done by the interface is reflected in a gradual fine-tuning of the probability density $p(\mathbf{x})$ (cfr.(10.4)) that models the relevance of images. At the start of the exploration, samples are drawn *uniformly* from the whole database, since the uniform distribution is the appropriate probability density to model the original "un-informed" state of the user. However, by transferring relevant images to the collection box and using these collection box statistics to bias the sampling procedure (as explained in subsection 10.2.3) there is a gradual shift from the original "uniform" (i.e. non-informative) prior probability to an "informed" density peaking around the *target image*.

However, this fine-tuning is compounded by the fact that most of images selected for inclusion in the collection box will only be *partially* relevant. For instance, if one is looking for images of *red circles*, then all images showing *circles* are relevant, as are all images that are

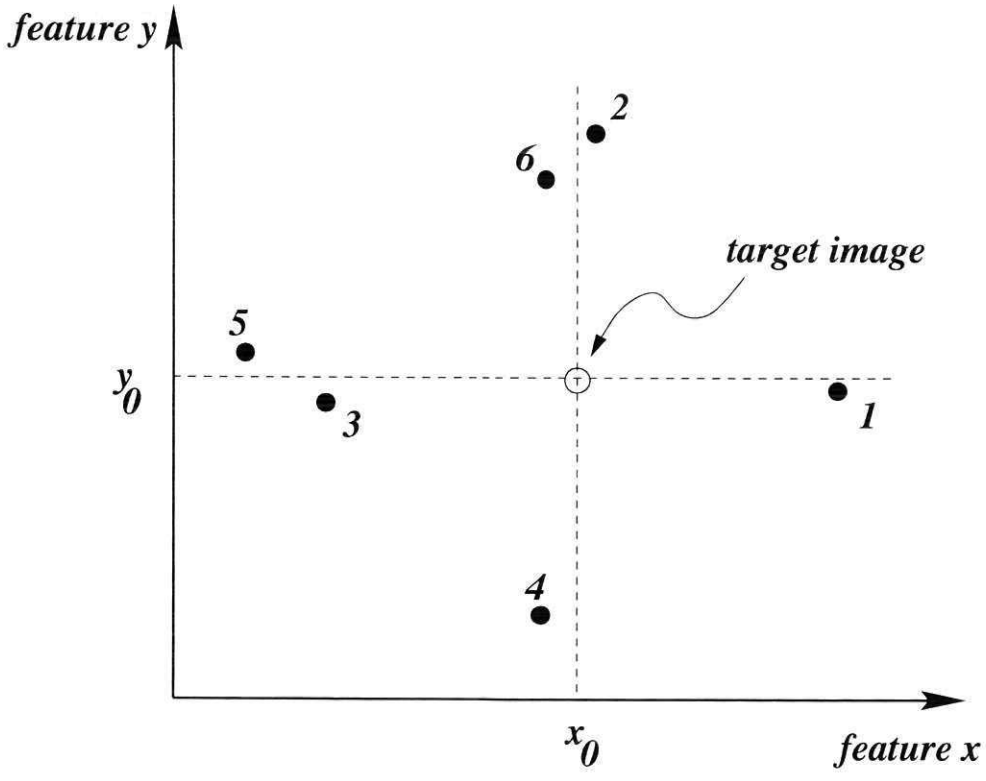

Figure 10.2: Collecting partially relevant images 1 through 6 allows one to estimate the location of the target image at (x_0, y_0). See main text for more details.

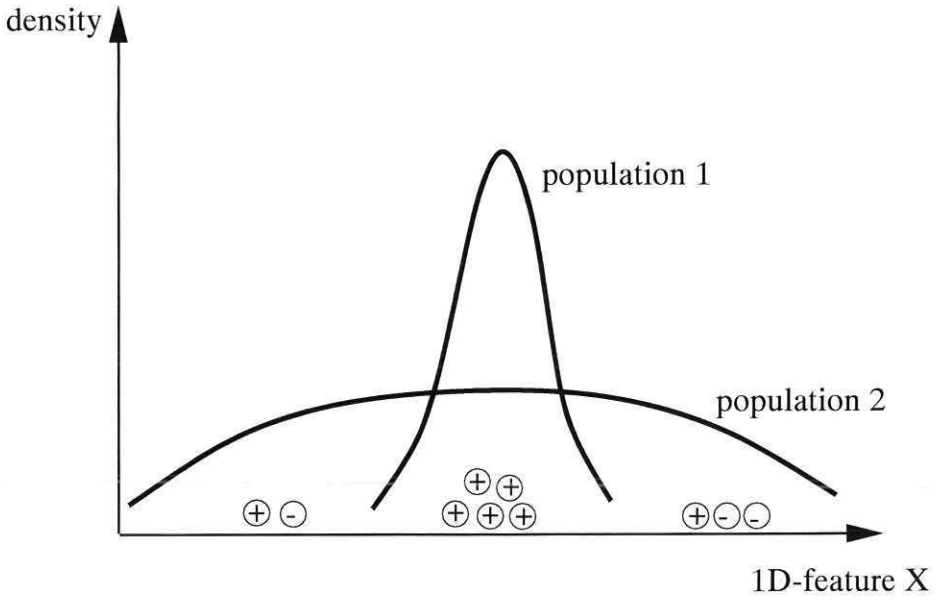

Figure 10.3: Decomposing the density generated by (partially relevant) examples (⊕) and counter-examples (⊖) generates a mixture distribution; the peaked distribution (population 1) collects all the examples that are relevant for the x-feature that is plotted along the axis. The diffuse background distribution (population 2) collects the counter-examples as well as the examples that are relevant with respect to another feature.

predominantly *red,* for the target image belongs to the intersection of these two classes. The problem we have to address is how to convey to the system that this partial information is meant to be complementary.

Interaction modes to deal with partial relevance As far as we can see, there are essentially two conceivable strategies to extract useful information from partially relevant images:

1. Either you can indicate for each partially relevant image, what *specific* feature is relevant for the search (e.g. the shape for image 1, and the colour for image 2). This is the approach championed by most interfaces. However, the requirement that features must be identified explicitly is a very restrictive one and in most cases not tenable. (After all, how would you inform the interface that you are looking for an image that is *just as colourful as this one, but as chaotic as that one*?)

2. The approach we propose is designed to circumvent this difficulty: there is no need to identify features, just point out additional images that are partially relevant. Over time, the statistics will automatically hone in on the relevant features. We will elaborate this proposal presently.

Integrating information from partially relevant images For clarity we will discuss the following caricature of the problem (see Fig 10.2): Assume that every image is characterized using two numerical features (x and y say); this means that every image can be represented as a single point $(x,y) \in \mathbb{R}^2$. Furthermore, assume that we are after the target image located at (x_0, y_0). While exploring the database we come across image 1 (with features x_1 and y_1, see Fig 10.2). Clearly this image is *partially relevant* to our search as it resembles the target image as far as the y-feature is concerned ($y_1 \approx y_0$). Because of this partial relevance it is added to the stack. Later on in the exploration, the user comes across image 2, which again is partially relevant, this time because of its x-feature ($x_2 \approx x_0$), and is therefore added to the stack. In the same vein images 3, 4, 5 and 6 are added, so that we end up with a stack containing 6 images, all of which are *partially*, but none of which is *completely* similar to the target image.

If we now project the six selected images on each of the two feature axes and construct the corresponding 1-dimensional densities, we would notice how in both cases the resulting density is in fact a mixture of two populations (see also Fig. 10.3). To fix ideas look at the projection on the x-feature: The first population comprises the images that are relevant with respect to x and therefore constitute a peaked density centered about the appropriate x-value.

The second population harbours the images that are relevant with respect to the y-feature. Their x-value therefore does not adhere to a narrow distribution and they are more or less uniformely distributed in the "background". Being able to tease apart these two distributions is key to efficiently locating the interesting region in the search space. Obviously, the quadratic

logistic regression model (as specified by eq.(10.4)) is one important tool in this process. Using counter-examples is another, as will be explained next.

The importance of being negative Since counter-examples — by their very definition — indicate what you are not looking for, none of their projections on the feature axes will occur near the target values. Put differently, the projections of counter-examples will show up in the "background" density, rather than in the peaked "foreground" density (see Fig. 10.3). As a consequence, the peaked foreground contains examples only, whereas the diffuse background is a mixture of both examples and counter-examples.

In order to model this mixture-population in the background, the 1-D logistic regression model will further reduce its probability over the background regions but concentrate it near the peak-region, thus creating an even more pronounced delineation of the region of interest around the target-points x_0 and y_0.

Conclusion In this paper we outlined the architecture of a CBIR-interface that offers the user a graphical tool to create new features by *showing* (as opposed to *telling*!) the interface what he means. It allows him to interactively classify images by dragging and dropping them into different piles and instructing the interface to come up with features that can mimic this classification. For more details and an early proto-type, we invite the reader to visit our website:
`http://www.cwi.nl/~pauwels/pariss.html`

Acknowledgments: E.P. and G.C. gratefully acknowledge partial support by the Belgian Fund for Scientific Research (F.W.O. Vlaanderen, Belgium), under grant G.0366.98. B.S. gratefully acknowledges support by NWO (The Netherlands).

Bibliography

[1] B.A.M. Schouten, P.M. de Zeeuw. Image Databases, Scale and Fractal Transforms. To appear in Proceedings ICIP 2000

[2] A.D. Del Bimbo: *Visual Information Retrieval*, Morgan Kaufmann Publishers, Inc., California, 1999.

[3] G. Frederix, G. Caenen and E.J. Pauwels: *PARISS: Panoramic, Adaptive and Reconfigurable Interface for Similarity Search.* To appear in Proceedings ICIP 2000.

[4] D W Hosmer and S Lemeshow: Applied Logistic Regression. Wiley 1989.

[5] T. Kakimoto and Y. Kambayashi: *Browsing functions in three-dimensional space for digital libraries*, International Journal on Digital Libraries, (2):68-78, 1999.

[6] S. Santini and R. Jain: *Beyond Query by Example.* Proceedings of ACM Multimedia'98, Bristol, UK.

Part IV

Links

Appendix A

Clicks

A.1 E-content

E-content is characterized by its digitised form. Once a document has been created it can be (re)distributed many times. E-content is typically available in large amounts everywhere and always. With the introduction of the World Wide Web and the boost of *Information and Communication Technology* (*ICT*), information is abundant. This fact has brought information closer to the user. From his home or office, and even on the road, specified information can be browsed, searched, retrieved, collected and printed more easily than ever.

As a consequence, E-content will gain its added value from enriching the information in such a way that it can be used better. As with other raw materials, added value is created by processing the raw material. A coffee bean is better enjoyed as a cup of coffee. In it's turn a cup of coffee is better enjoyed when it is served in a café. A maximum of added value is achieved as this cup of coffee is consumed in a café in Saint-Tropez, although some of us will not agree. Value is added to E-content by processing it in such a way that it can be managed and deployed. A chain of added value can be created according to the services added to the information, see Figure A.1.

E-marketing is an example of this. Product information can be delivered scalable and adaptive to the consumer with the introduction of electronic services and devices. For the user added value is created by personalizing the information to his needs (user profile). Another adaptation is by scaling the information to the capabilities of the device (mobile phone, fax, pc). For the distributor, value is created by analyzing and anticipating consumer demands, thereby optimizing his success in pre, sale and after sale contacts. This is done for instance by intelligent management systems that show buying behavior and other statistical relationships.

A.2 Knowledge Map

One of the Winners of the 2000 National Millennium Contest is "Datacloud" of ArchiNed. Datacloud is an internet environment combining a knowledge map, a fuzzy search engine and a discussion platform. Elements in the cloud are created (or edited) documents submitted by the users. The user-interface offers the possibility to gain insight into the documents with their complex structures and links. Documents can be browsed and queried. Subject of the documents is the repartitioning of the polder: "De Hoeksche Waard". The team of developers are from *V2 organization, Archined* and *Desk.*

`http://www.dwhw.nl.`

A.3 Standards

Organizations like W3C and ISO are creating standards for storing and processing multimedia information. In a world where information technology, consumer electronics and telecommunication products incorporate increasingly sophisticated technologies and the need for timely available standards is as strong as ever, MPEG provides a proven mechanism to feed research results into industry. MPEG established in 1988, produced MPEG-1, the standard on which such products as Video CD and MP3 are based, MPEG-2 the standard on which such products as Digital Television set top boxes and DVD are based and MPEG-4, the standard for multimedia on the web. The current thrust is MPEG-7 "Multimedia Content Description Interface". MPEG-7, formally named "Multimedia Content Description Interface", aims to create a standard for describing the multimedia content data that will support some degree of interpretation of the information's meaning, which can be passed onto, or accessed by, a device or a computer code.

`http://www.cselt.it/mpeg.`

A.4 Input for Content Based Image Retrieval (CBIR) Systems

In CBIR systems the user has several methods to provide input to the system [1]. Examples are sketching and query by example. Query by example consists of selecting an image from a random sample or previous results. Querying by sketch is commonly used to retrieve images containing objects with shapes similar to the sketch. Sketches represent a template of either the complete object silhouette or part of it. They can be manually authored or traced from an image, see Figure A.2.

A.5 Histogram Matching

Swain and Ballard propose *histogram matching* for color matching and indexing. In this thesis a similar measure is used for texture matching, see Chapters 7, 8, 9 and [12, 13, 14]. Histogram matching is defined as:

$$D_H(I_Q, I_D) = \frac{\sum_{j=1}^{n} min\,(H(I_Q, j), H(I_D, j))}{\sum_{j=1}^{n} H(I_D, j)}, \tag{A.1}$$

Where $H(I_D)$ denotes the histogram of the database images and $H(I_Q)$ denotes the histograms of the query image. j is the index of the bin in the histogram (Figure A.3).

A.6 Textures

Tamura et al. [16] experimented for the purpose of constructing psychometrics with prototypes of texture. In the psychological experiments of Tamura on basic textural properties, it was shown that there exists six specifications common to all visual textures, which relate to the perception of many human beings:

contrast , *coarseness* , *directionality* , *line-likeness* , *regularity* and *roughness*.

We show some examples:

- **Coarseness**
  When two patterns only differ in scale, the magnified one is coarser. For patterns, the bigger its element size and/or the less its elements are repeated, the *coarser* it is felt, see Figure A.4.
- **Contrast**
  Difference between pixel intensity variations within a texture define the notion of contrast. If pixel intensities vary highly, the texture is said to have high *contrast*, see Figure A.5.
- **Directionality**
  The *direction* of the texture. Here the orientation of the pattern doesn't matter, see Figure A.6.

Tamura derived corresponding computational measures derived from the psychological experiments. As an example a QBIC feature used in the QBIC system [8].

A.6.1 Example of a QBIC Feature

To compute Tamura directionality [8], a gradient is calculated for every pixel in the image. The gradient ∇ is expressed in terms of its magnitude $|\nabla|$ and angle $\theta(\nabla)$. As defined by Tamura, one takes the gradient for every pixel and ignores all pixels for which the gradient magnitude $|\nabla|$ is less than some threshold t. Then, for the remaining pixels, one creates a histogram of counts versus $\theta(\nabla)$.

A.7 Similarity Matching and Human Perception

The importance of aspects of texture depends heavily on the circumstances and the relationship with other features. Several aspects of texture become less or more important in combination with other features, e.g. color. Mojsilovic [7] et al. developed a technique for search and manipulation of color patterns. They focus on the integration of color and texture features. The leading principle is that human beings only have a functional notion of similarity within a particular domain. To perform similarity matching in a human manner one has to:

- Choose a specific application domain.
- Understand how users judge similarity within that domain.
- Build a system that will replicate human performance.

A.8 Object Centered Neglect

Neuro-physiological studies [4] showed that patients with a defect called *Object Centered Neglect* are only able to reproduce one half of a (symmetric) form presented to them. If these patients are presented with an image in the visual field, they tend to ignore its left side. Figure A.7 shows two pictures (left column) and two copies of those pictures (right column) produced by a patient. When the two flowers of model B are copied, the patient copied only the right half of each flower. When the same two flowers are joined by a common stem, the patient copied the plant, saw the plant as one object, and left out every "left" detail, including all of the leftmost flower. As objects are defined with respect to the object's (tilted) midline, it can be speculated that the brain uses symmetry to store and retrieve information efficiently.

A.9 Washburn

Dorothy Washburn, archaeologist and anthropologist at the Maryland Institute in Baltimore examined the figurative language of the Pueblo Indians in the South West of the United States of America. In her book "Symmetries in Culture Theory and Practice of Plane Pattern Analysis" [19] she proclaims it is the way structures are built, that is essential in classifying different textures, see Figure A.8.

A.10 Fractal Dimension [3]

The fractal dimension can be thought of as a scaling relationship. As an example we show in Figure A.9 sets and the scaling relationship for each (i.e. the way the number of boxes it takes to cover the set, scales with the size of the box). We describe the scaling relationships:

- A curve of length l can be covered by l/ε boxes of size ε and 2^1 l/ε boxes of size $\varepsilon/2$.
- A region of area A can be covered by A/ε^2 boxes of size ε and 2^2 A/ε boxes of size $\varepsilon/2$.
- A set with volume V can be covered by V/ε^3 boxes of size ε and 2^3 V/ε boxes of size $\varepsilon/2$.
- If the Sierpinsky triangle is covered with N boxes of size ε, then it takes $3N$ boxes of six $\varepsilon/2$ to cover it. This is shown for ε equal to half the width of the set in the figure.

For the first three examples, halving the box size corresponds to increasing the number of boxes required to cover the set by a factor of $2^1, 2^2$ and 2^3, at least in the limit when ε is very small. For the Sierpinsky triangle, however, the number of boxes increases by 2^d, where $d = log3/log2$, and so we call this the fractal dimension of the set. It is intuitively right since the set has "no area" but is more "meaty" than a curve.

A.11 Pseudo-Code

- Divide the image into a set of non overlapping ranges R_i. Mark all ranges uncovered.
- While there are uncovered ranges R_i do {
  - Choose domain D_i and w_i minimizing [1] $\varepsilon = d_{rms}(w_i(f), f)$.

[1] The *fractal transform* is chosen to minimize the collage error. This error provides an upperbound between the image f and the encoded image f_W, in terms of the distance between the image itself and the transform of the image. In practice we minimize $d_{rms}(w_i(f), f), i = 1, \ldots, N$ using the *root mean square error*, see Subsection A.11.1.

- If ε is smaller [2] then T or $size(R_i) \leq r_{min}$ then
  - mark R_i as covered and write out the transformation w_i;
- else
  - Partition R_i into smaller ranges [3] that are marked as uncovered. Remove R_i from the list of uncovered ranges.

}

A.11.1 Root Mean Square Error

The difference between two images f and g can be expressed in several ways. A common set of error-measures is based on the L_p norm. L_1 is an absolute error; L_2 is called the *root mean square error*.

$$L_p = \left(\frac{1}{MN} \sum_{j=0}^{N-1} \sum_{i=0}^{M-1} |f_{i,j} - \overline{f}_{i,j}|^p \right)^{\frac{1}{p}} \tag{A.2}$$

A.12 Fractal Feature Extractors

We describe briefly some methods from the literature.

The first method from Sloan et al. [15] can be used for indexing large image databases. Images are categorized in clusters on the basis of their similarity to an iconic image. For each icon, a joint fractal coding procedure approximates ranges of the icon by domains from the icon itself as well as from every database image. A rate R is calculated expressing how frequently a particular block in the database image is chosen as domain block for the iconic image. This rate R is then used as a measure for the similarity between the iconic and the database images.

Different from the approach where feature vectors are derived and similarity is calculated according to a distance function between these feature vectors, this method is directly related to the "amount" of similar image content (Figure A.10).

A nice example comes from Vissac et al. [18] They use a method only inspired by fractal image compression to query a database of Trademarks. No quadtree structure is used and no massic transform (Section 5.22). By doing so, real similarity between blocks is searched

[2] T gives an indication of the quality required and can be choosen arbitrary.

[3] For instance by using a quadtree procedure as discussed in Section 5.3.

for and not similarity between structures and smaller details. Another simplifying aspect is the reduction of the dimension of the domain pool. The similarity search in fractal encoding can be quite "chaotic". To maintain a global coherence it is a good idea to relate neighboring domain blocks with neighboring range blocks. In this way a spatial continuity is created in the similarity calculations. The use of a dynamic programming technique (Viterbi algorithm) ensures continuity and coherence of the localized block matching results. At the same time the domain pool is dramatically reduced.

A.13 Intelligence

Intelligence and in this case visual intelligence can be defined as a *Turing* machine, being able to give relevant answers to relevant questions. Possible questions could be:

- Sound system play me Beethoven's fifth symphony, by whistling the first five bars.
- Visual system show me the images depicting a sun set.

In order to do so, there is need for more levels of interaction to computer systems:

- Interfaces that can speak, listen, understand and show.
- Processing units able to understand and reason.
- Memory algorithms able to recognize, remember, store and compare.

A.14 FourEyes

FourEyes [5, 6] is an interactive, power-assisted tool for segmenting and annotating images. Click on some regions, give them a label, and FourEyes extrapolates the label to other regions on the image and in the database. This could be implemented by choosing a single image feature like color or texture and forming, say, a Gaussian model of the label in that feature space. However, that naively assumes that one model will be suitable for all kinds of labels in all kinds of imagery. Instead, FourEyes has a "society of models" among which it selects and combines models to produce a labeling. This selection process is guided solely by the examples given by the user.

`http://www-white.media.mit.edu/vismod/demos/photobook/foureyes/facialdatabase`

A.15 Relevance Feedback: Rui

Rui [10] et al propose weights to be fixed by examples given by the user. These weights can then be updated by a process of *relevance feedback*. The user indicates the relevance of an image by marking the image from highly relevant to highly irrelevant. A standard deviation based weight updating process is then proposed.

Let M be the number of objects marked with highly relevant or relevant classification. A $M \times K$ matrix is created where K is the number of features to characterize the images with. If all relevant objects have similar values for the k th feature component r_{ik}, k fixed and $0 \leq i \leq M$, it means that this feature is a good indicator for the relevance of the image.
The weight for this feature is then updated according to the standard deviation of the sequence r_{ik}, $0 \leq i \leq M$.

A.16 Browsing within Pariss

In our interface Pariss, browsing is enabled by presenting a two dimensional view *display window* of the n-dimensional data space representing the objects. An adaptive way of updating the display windows is created. The user is invited to position the image on an initial display widow according to his own subjective appreciation. The *projection engine* performs a transformation A on the features space in accordance to the user defined positioning t_i of the images (Figure A.11). We project the high dimensional feature space using a transformation A, when applied to the selected points $\mathbf{x}_i$, best matches the user-defined configuration of the points $\mathbf{t}_i$.

$$A : \mathbb{R}^K \rightarrow \mathbb{R}^2. \tag{A.3}$$

The optimality is expressed with respect to *Sammon's metric stress* which is defined as:

$$S_M(A) = \sum_{i,j} \frac{(d(\mathbf{t}_i, \mathbf{t}_j) - d(A\mathbf{x}_i, A\mathbf{x}_j))^2}{d(\mathbf{t}_i, \mathbf{t}_j)}, \tag{A.4}$$

where d is the standard Euclidean metric on $\mathbb{R}^2$. This Sammon-criterion can be minimized using *gradient descent.*

A disadvantage of the method is the co-dimension [9] of the display window $k-2$, In this way the windows do not represent the images in a way that is optimal to the user.

A.17 Relevance Feedback in Pariss

A probability p is modeled for the relevance of an image. $p \approx 1$ would mean that this image is highly similar, whereas $p \approx 0$ indicates that it is highly dissimilar. Strictly speaking, the probability measure p depends on the full feature vector $\mathbf{x} = (x_1, x_2, \ldots, x_K)$, however in our case we model p as a function of each feature independently.

A *logistic regression model* is used to model the relevance p. We opted for a *quadratic* logistic regression model:

$$\log \frac{p}{1-p} = \alpha x^2 + \beta x + \gamma; \tag{A.5}$$

The parameters α, β and γ are determined by using *maximum likelihood estimation*, i.e. they optimize the probability of the actual configuration occurring. More explicitly, if we look up the x-value for each of the images in the collection box and then use eq.(A.5) to compute the probability p that they are in fact an example ($p > 1/2$) or counter-example ($p < 1/2$), then the parameters (α, β, γ) are chosen to optimize this prediction accuracy.

A.18 Other Methods of Content Extraction

If available features are not able to express the similarity with an example image e, a joint fractal coding procedure (see Section A.12) could act as an "eigenfilter" on the database. Images are categorized in clusters on the basis of their similarity to the example image. Then the joint fractal coding procedure approximates ranges of the e by domains from e itself as well as from every database image. A rate R_e is calculated expressing how frequently a particular block in the database image is chosen as domain block for the example image e. This rate R_e is then used as a measure for the similarity between the example image e and the database images.

Different from the approach where feature vectors are derived and similarity is calculated according to a distance function between these feature vectors, this method is directly related to the "amount" of similar image content.

A.19 Icons

For retrieval of image data a hierarchical approach can be implemented based on (non-linear) multi resolution decomposition of images (pyramids, wavelets, fractals). Wavelet or fractal coefficients of an image can be used to calculate a feature vector for indexing and retrieval. An interesting property of this approach is that even feature vectors containing a very small

fraction of all coefficients can be used to compute icons (or thumbnails) of the images that are represented by these coefficients.

In general it is not possible to derive an aspect of an image by selecting wavelet coefficients. But, based on local analysis techniques the full resolution image can be analyzed to locate characteristic and homogeneous areas. Then an intelligent reduction of the image – that is, iconification – can be performed.

In Figure A.12 we illustrate the visualization of an icon by a fractal feature as discussed in Chapter 7. Ultimately one wants these icons to be interactive such that position, orientation, perspective tec. can be changed. These icons are then used within the query process and in relevance feedback procedures.

A.20 Finding known Objects

A simple approach to locate images containing known objects (Figure A.13) is based on color histograms. Local color histograms [2, 17, 11] are used to locate a known object in the image. Histogram intersections are evaluated between the histogram of the object and all image blocks of a fixed size in any position of the image. Drawback is the complexity of the methods used. Other methods are based on shape information. A method for using texture as an aspect to retrieve known objects or images is required.

A.21 Hierarchical Indexing and Searching

The feature vector x in a hierarchical indexing scheme consists of finitely many, say N, subvectors: $x = (x_1, x_2, \ldots, x_N) \in E_1 \times E_2 \times \cdots \times E_N$, where $x_i \in E_i$ contains less data than x_{i-1}. Furthermore, there should exists a metric d_i on the space E_i which yields the similarity between images at level i. Searching a hierarchically featured database is done in N steps: in the first step we discard all target images T whose N'th feature vector $x_n(T)$ satisfies

step 1: $\quad d_N(x_N(Q), x_N(T)) > \varepsilon_N$

Here Q is the query image and ε_n a given threshold. In the second step we discard all target images (remaining after step 1) for which

step 2: $\quad d_{N-1}(x_{N-1}(Q), x_{N-1}(T)) > \varepsilon_{N-1}$

We repeat this procedure until we have reached level 1. The fact that x_i contains (much) less data than x_{i-1} implies that step i requires (much) less computation time per image than step $i-1$. For huge databases, such a search strategy is likely to be effective.

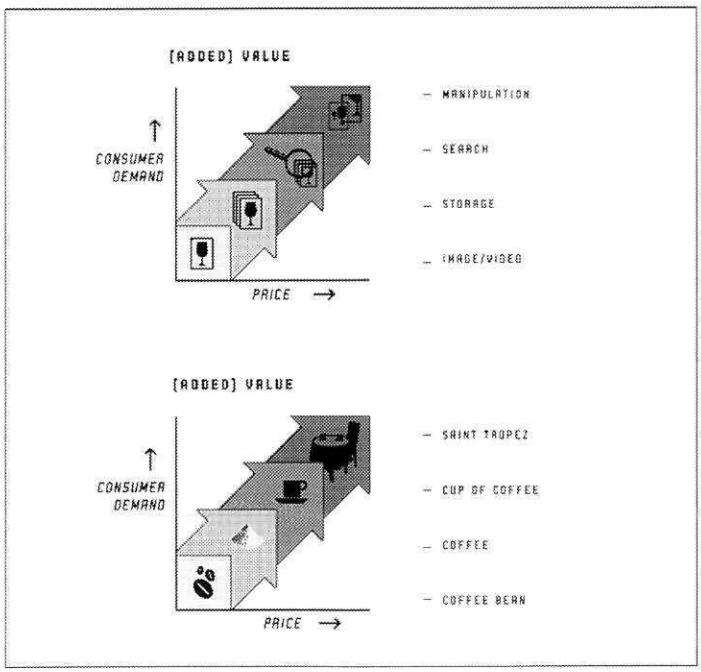

Figure A.1: An example of value added to *E-content*.

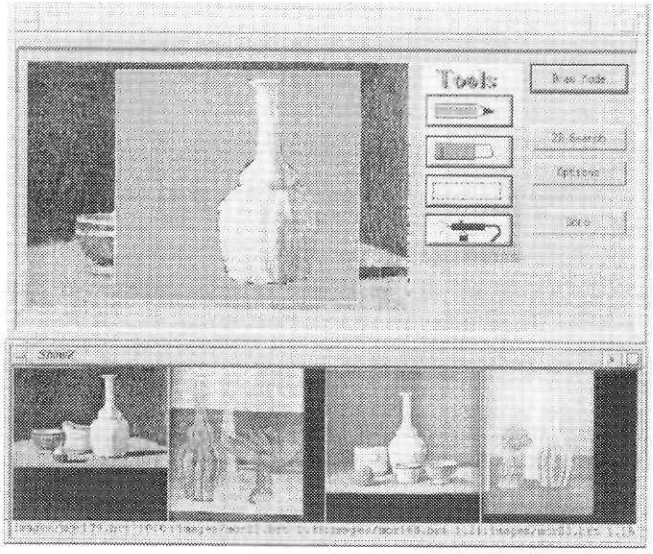

Figure A.2: Example of a query by sketch.

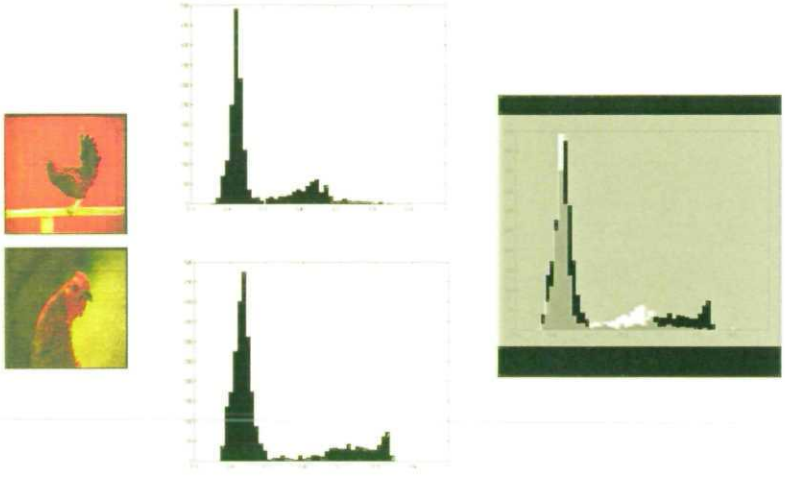

Figure A.3: Histogram intersection between two histograms A and B.

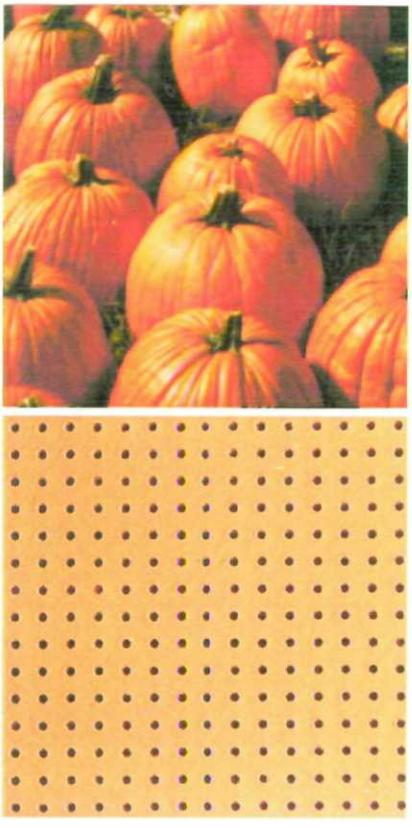

Figure A.4: Two textures, one is *coarser.*

Figure A.5: Two textures, one having high *contrast.*

Figure A.6: Two textures, one having a clear *direction*.

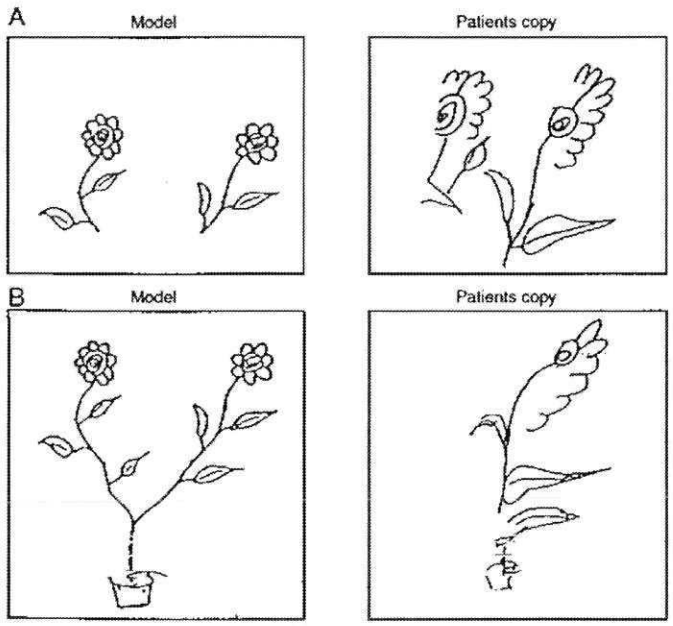

Figure A.7: A test for object-centered neglect. When asked to copy the two drawings on the left, a patient made the two copies on the right. Detail is omitted from the left half of the drawing as a whole. From Marshall and Halligan [4].

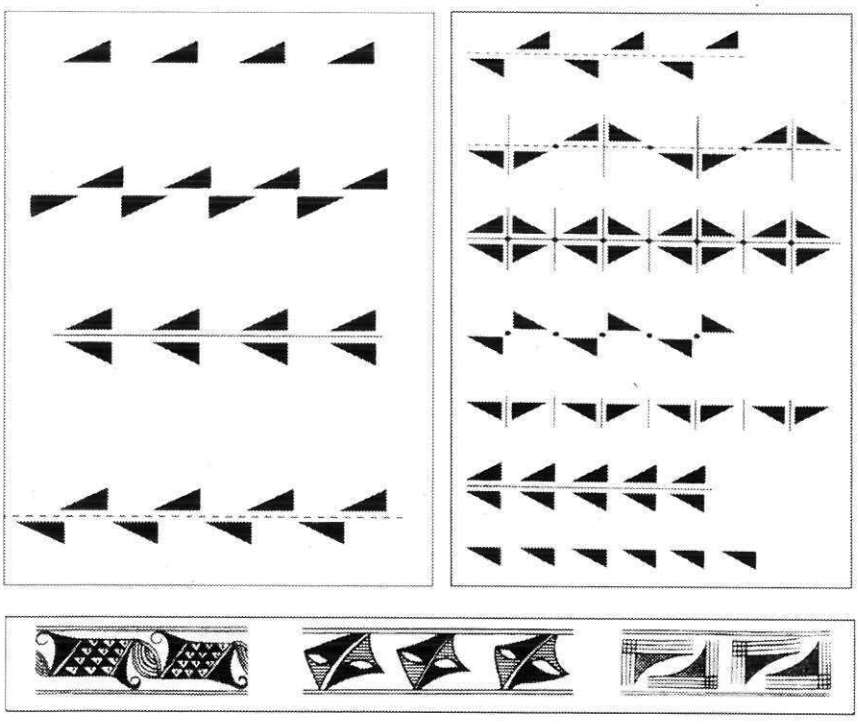

Figure A.8: *top:* A (2-dimensional) pattern is symmetric if there exists at least one of four basic plane symmetries: rotation, translation, reflection, glide reflection. *middle:* Seven possibilities to create a one-dimensional regular symmetric pattern. Number 2 is used by the Navajo Indians; number 4 used by the Hopi Indians. *bottom:* Examples of one-dimensional patterns of the Pueblo Indians, with rotational symmetry.

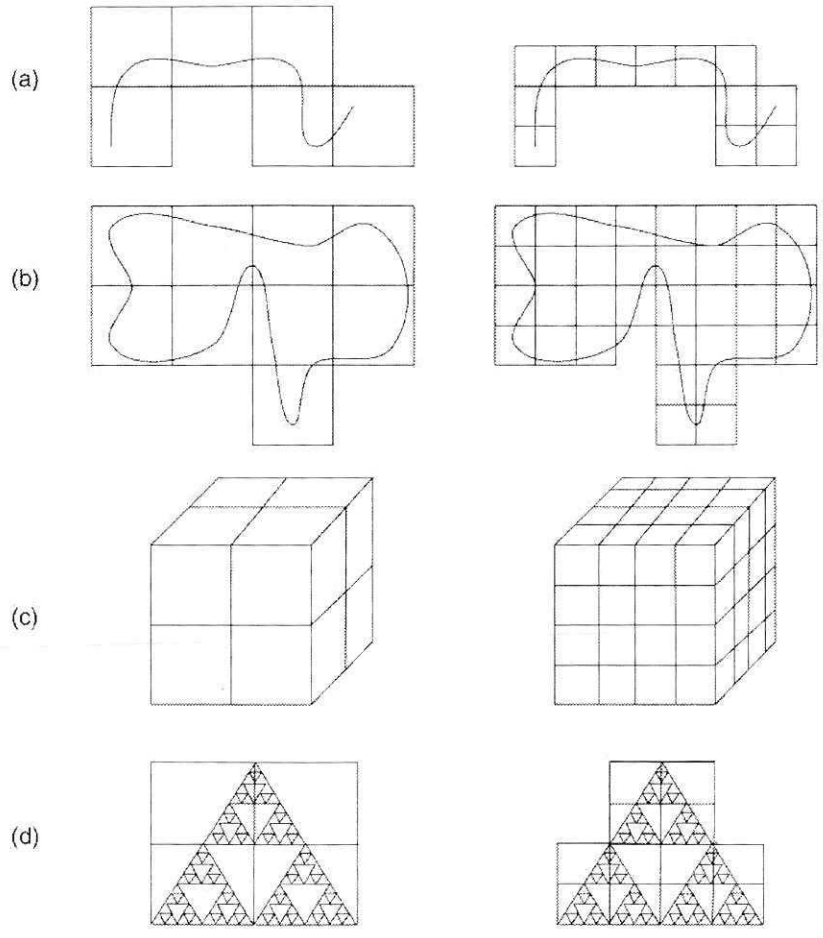

Figure A.9: Four examples of sets and the number of ε boxes required to cover them. Courtesy of Yuval Fisher [3].

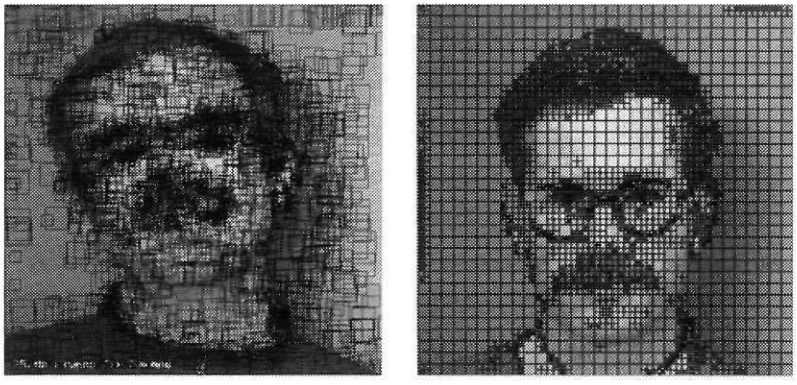

Figure A.10: A joint fractal coding technique is searching for blocks in one image to be approximated by blocks in another image.

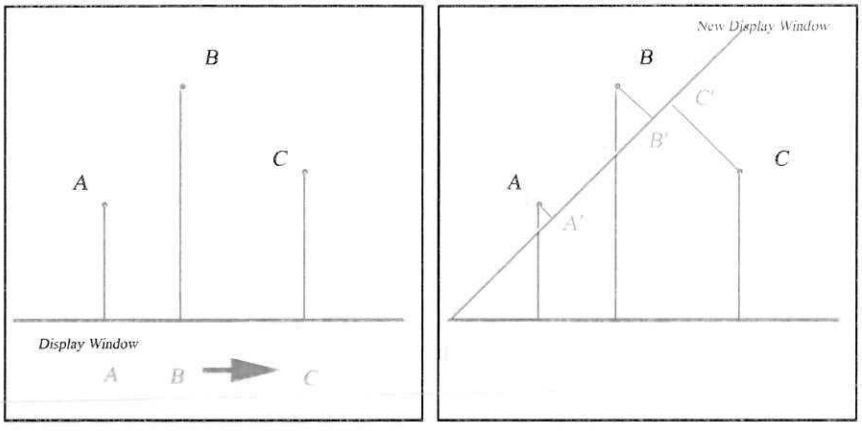

Figure A.11: *left:* Original configuration. The user "moves" image *B* towards *C*. *right:* New projection in accordance with the action of the user.

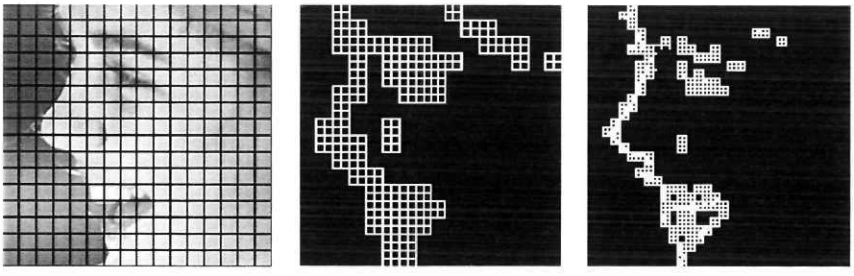

Figure A.12: Visualization of the *success rate* feature in an icon (fracFeat). A fractal coder divides an image into blocks called range blocks. Within a quadtree subdivision procedure, these blocks are approximated by domain blocks. In this case, three quadtree depths were used. *left:* Original image and initial partition. *middle:* Subdivided range blocks at the next guadtree depth are depicted. *right:* More detail is created.

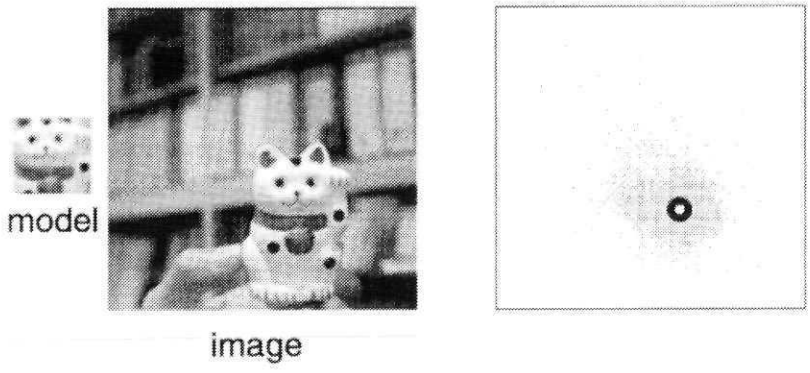

Figure A.13: Finding "Waldo" [17]-courtesy V.V. Vinod and H. Murase.

Appendix B

Demos

These demos will only available on the CDrom

B.1 Zoom Invariance

Fractal encoded images are not real fractals. Nevertheless fractal features derived from fractal encoded images, can be made invariant to zooming in at a homogenous textural images, see Chapter 9. Demo that illustrates the method.

B.2 VisTEX, Database of Homogeneous Textures.

B.3 Koe

Demo that shows the documents retrieved if the user uses a keyword in this case *koe* [1] to search the internet by the well-known search engine: *www.altavista.com.*

[1] Meaning cow in Dutch.

B.4 Demo IFS

Demo generating fractals by *iterated function system.* Courtesy Raj Iyer, University of Pennsylvania.

B.5 Demo FIC

Interactive Java demo illustrating the principles of fractal image compression. Courtesy Jean-Luc Dugelay, Institut Eurécom France.

B.6 Een Wereld aan ICT Toepassingen

Video produced by Ministery of Economical Affairs (EZ) of The Netherlands. Several future I.T. applications are shown.

B.7 Fractal Imager

Softeware to fractal encode color images. Courtesy of Michael Barnsley, Iterated.Com.

B.8 Pariss

Demo of the Pariss interface.

Bibliography

[1] A. Del Bimbo. *Visual Information Retrieval.* Morgan Kaufmann Publishers Inc., San Francisco, California, 1999.

[2] F. Ennesser and G.Medioni. Finding waldo, or focus of attention using local colour information. *IEEE trans. on Pattern Analysis and Machine Intelligence*, 17(8), 1995.

[3] Y. Fisher. *Fractal Image Compression: Theory and Application*. Springer-Verlag, 1995.

[4] J.C. Marshall and P.W. Halligan. Visuo-spatial neglect: A new copying text to asses perceptual parsing. *Journal of Neurology*, 240:37–40, 93.

[5] T. Minka. An image database browser that learns from user interaction. Technical Report 365, MIT Media Lab Perceptual Computing Section, 1995.

[6] T. Minka. Picard: Interactive learning using a society of models, 1997.

[7] A. Mojsilovic, J. Hu, R.J. Safranek, and S.K. Ganapathy. Matching and retrieval based on the vocabulary and grammar of color patterns. *IEEE Transactions on Image Processing*, 9(1):38–53, January 2000.

[8] W. Niblack, R. Barber, W. Equitz, M. Glasman, D. Petkovic, P. Yanker, C. Faloutsos, and G. Taubin. The QBIC project: querying images by content using color, texture and shape. *Storage and Retrieval for Image and Video Databases*, 1908:173–187, 1993.

[9] Y Rubner. *Perceptula Metrics for Image Database Navigation*. PhD thesis, Stanford University, May 1999.

[10] Y Rui, T. Huang, and S. Mehrotra. Relevance feedback techniques in interactive content-based image retrieval. In *Proc. of IST and SPIE: Storage and Retrieval of Image and Video Databases VI*, pages 25–36, 1998.

[11] R. Schettini. Multicoloured object recognition and location. *Pattern Recognition Letters*, 15:1089–1097, 1994.

[12] B.A.M. Schouten and P.M. de Zeeuw. Feature extraction using fractal codes. In *Visual 99; Visual Information and Information Systems, Third International Conference*, pages 483–492. Springer, 1999.

[13] B.A.M. Schouten and P.M. de Zeeuw. Fractal transforms and feature invariance. In *Procs. 15th International Conference on Pattern Recognition (ICPR 2000)*, pages 992–997. IEEE Computer Society, 2000.

[14] B.A.M. Schouten and P.M. de Zeeuw. Image databases, scale and fractal transforms. In *ICIP 2000; International Conference on Image Processing*, 2000.

[15] A.D. Sloan. Retrieving database contents by image recognition. *New Fractal Power, Advanced Imaging*, 9(5):26–30, 1994.

[16] H. Tamura, S. Mori, and T. Yamawaki. Texture features corresponding to visual perception. In *IEEE Transactions on Systems, Man and Cybernetics*, volume 8(6), 78.

[17] V.V. Vinod, H. Murase, and C. Hashizume. Focused colour intersection with efficient searching for object detection and image retrieval. In *IEEE Int. Conference on Multimedia Computing and Systems*, pages 229–233, 1996.

[18] M. Vissac, J. Dugelay, and K. Rose. A fractals inspired approach to content-based image indexing. In *ICIP; Proceedings IEEE International Conference on Image Processing*, page CDROM, 1999.

[19] Washburn, Dorothy K., and Donald W. Crowe. *Symmetries in Culture Theory and Practice of Plane Pattern Analysis*. The University of Washington Press, Hongkong, 1998.

Samenvatting/Dutch Summary

In ons digitale tijdperk groeit de hoeveelheid beschikbare informatie exponentieel. Dit schept de behoefte deze informatie beter te beheersen. Een oplossing voor dit probleem is het beschrijven van de inhoud met keywords. Bekende zoekmachines op het World Wide Web zoals *google.com* en *altavista.com* werken op deze manier. Echter, eenieder kent de frustratie tijdens het zoeken naar informatie op het internet en hoort zichzelf verzuchten: "Dit is niet wat ik bedoel, begrijp je niet dat dit essentieel anders is dan wat ik zoek".

Voor (visuele) informatie is het wenselijk op een meer inhoudelijke manier te kunnen zoeken. *Content based image retrieval* (CBIR) is erop gebaseerd afbeeldingen in een *database* te kunnen zoeken op basis van de gelijkenis met andere afbeeldingen. Een *interface* voor een visuele zoekmachine moet in staat zijn plaatjes van koeien te tonen op basis van gelijkenis met verschillende voorbeelden die de gebruiker heeft geselecteerd of getekend. Dit proefschrift beschrijft methoden van visuele informatie verwerking.

Voorbeelden van toepassingsgebieden zijn:

1. Digitale televisie. Het zoeken op inhoud van een televisie uitzending.
2. *E commerce. Product databases* zoals textiel, waarbij de gebruiker op zijn eigen smaak en voorkeur, producten kan uitzoeken.

De inhoud van een video of foto wordt aan de hand van een aantal karakteristieke kenmerken (*features*) zoals de kleur, vorm en textuur van een object of afbeelding beschreven. Een voorbeeld is de nieuwe video standaard MPEG 7, hiermede is het bijvoorbeeld mogelijk de fragmenten met doelpunten uit een voetbalwedstrijd te selecteren. Ook aspecten als de herkenning van een scene-overgang in een video of de aanwezigheid van een object binnen een afbeelding (bijvoorbeeld een gezicht) kunnen geanalyseerd worden.

Textiel is een voorbeeld van een product dat nauwelijks beschreven kan worden door *keywords* maar visueel goed geanalyseerd kan worden op basis van de patronen die zich herhalen. Probeert u de stof van uw kleding eens te beschrijven !

Dit proefschrift bestaat uit delen. Het eerste deel schetst het podium waarop vier gepubliceerde wetenschappelijke artikelen figureren. Deze hoofdstukken zijn geschreven voor een algemeen publiek. Hoofdstuk 2 beschrijft zoekmachines voor video en beeldmateriaal, zoals deze ontwikkeld worden in commerciële systemen en wetenschappelijke prototypes. Een ander aspect dat aan de orde komt, zijn de *interfaces* die nodig zijn om visuele informatie te tonen, te zoeken en door te bladeren. Zij verschillen essentieel van de traditionele interfaces.

Hoofdstuk 3 beschrijft hoe de visuele inhoud beschreven wordt aan de hand van verschillende eigenschappen zoals kleur en textuur (*features*). De aanleiding voor dit deel van het onderzoek is het succes van fractale beeldcompressie. Fractale beeldcompressie is erop gebaseerd de redundante (overtollige) informatie, o.a. veroorzaakt door de herhalingen binnen het patroon, te elimineren. Deze methode kan als zodanig gebruikt worden om de textuur eigenschappen van een afbeelding te analyseren.

In het tweede deel wordt de wiskundige achtergrond beschreven die ten grondslag ligt aan de gepubliceerde artikelen. Hier komt ook FracFeat ter sprake, een computer programma dat op deze principes is gebaseerd.
Het derde deel wordt gevormd door vier gepubliceerde wetenschappelijke artikelen. De eerste drie artikelen onderzoeken het gebruik van fractale meetkunde om patronen te kunnen herkennen en beschrijven. De robuustheid van de gebruikte methode ten opzichte van allerlei varianties van de afbeelding zoals verschuivingen, rotaties, verandering van belichtingsomstandigheden komen hier aan de orde. Het laatste en vierde artikel beschouwt Pariss, een CBIR *interface*. Uitgangspunt is een gebruikersvriendelijke *interface* die in staat is interactief het zoekproces te verfijnen door terugkoppeling te geven over de relevantie van het resultaat. De kenmerken (*features*) waarop de gelijkenis van de beelden gebaseerd zijn, kunnen door de gebruiker in een *window* aangepast worden. We beschrijven een architectuur, die de gebruiker de mogelijkheid geeft (*drag and drop*) interactief afbeeldingen te classificeren. De *interface* zal een combinatie van kenmerken gebruiken om de gehele *database* te ordenen aan de hand van deze classificaties.

Gezien mijn voorgeschiedenis als beeldend kunstenaar wilde ik een proefschrift schrijven dat intressant is voor meerdere discplines.

Ik hoop dat u er van kunt genieten !

Dankwoord/Acknowledgements

Diverse mensen ben ik dank verschuldigd voor hun inspiratie, hulp, samenwerking, geduld en vooral vriendschap waarvan ik de afgelopen jaren heb mogen genieten. Een aantal personen wil ik hierbij met namen noemen.

Allereerst mijn promotor Prof Mike Keane. Het was Mike die vond dat kunst een uitgangspunt is om wetenschap te beoefenen, ongeacht de leeftijd van de beoefenaar. De vrijheid die ik van hem voor mijn onderzoek heb gekregen, heeft in belangrijke mate bij gedragen aan de tot stand koming van de onderzoeksresultaten. Een inspiratiebron die ik ook na mijn promotie zal koesteren. Henk Heijmans heeft veel van de feitelijke begeleiding op zich genomen, waarvoor ik hem hartelijk wil bedanken

Als medeauteur van drie publicaties maar ook als collega ben ik Paul de Zeeuw veel dank verschuldigd. Paul zorgde altijd dat het beste boven kwam. Bij Eric Pauwels denk ik aan zijn inspirerende denkwijzen. De betrokkenheid van de ondersteunende afdelingen was hartverwarmend; zonder de steun van Jan Schipper en Francien Goudsbloem was het (nachtelijk) overwerk onmogelijk geweest. Ook het promoveren in de vorm van een interactieve CD werd door iedereen van harte ondersteund, maar vooral mogelijk gemaakt door: Monique Mulder, Matthijs Tammes en Ruben Pater van Mattmo Concept & Design en Thomas Wermer van de Hogeschool voor de Kunsten Utrecht. De medewerkers van Desk Creative Technology die enthousiast waren en bleven toen ik de wens uitsprak om te promoveren en hun verantwoordelijkheid namen.

Tot slot wil ik Marga van Berkel bedanken, mijn vriendin. De belofte om zoiets (polynonsense !) nooit meer te doen leg ik hierbij schriftelijk vast. Vele vrienden hebben mijn afwezigheid "getolereerd", voor hen ga ik een feest geven. Mijn vader die hier zo graag deel van was geweest en mijn moeder die mij op jonge leeftijd overhoorde om te kijken of ik me les wel geleerd had. Mam het is gelukt !

Allen hebben ze bijgedragen aan de tot stand koming van deze promotie.

Het leven bestaat uit uitdagingen !

Curriculum Vitae

Ben Schouten was born in 1953 and graduated from the Rietveld Art Academy in 1983. He found himself interested in patterns and iconography, and after traveling in the Magreb and studying numbering systems he rediscovered his fascination for mathematics. During this period he worked as both a professional artist and student, receiving his masters degree in mathematics, specializing in chaos theory, in August 1995. It was prof M.S. Keane who, recognizing the qualities of both artist and mathematician, invited him to CWI in 1996. Here he joined the research project into Image Processing, an up and coming field in Mathematics and Computer Science. In the same year Ben Schouten founded Desk.nl., an application service provider, providing innovative internet related solutions to a wide range of customers.

Index

affine transformations, 46
Altavista, 12
attractor, 44, 45

CBIR, 130
collage theorem, 47
complete metric space, 44
conceptual level, 11
Content Based Image Retrieval, 4, 7, 55, 130
Content Dependent Metadata, 13
Content Independent Metadata, 13
contraction mapping, 44
Contraction Mapping Fixed-Point Theorem, 44
contractive, 44
contractivity factor, 44

database management systems, 11
display window, 136
domain block, 48, 57

E-content, 129
Euclidean distance, 19
external level, 11

Feature extraction, 4
feature space, 19
features, 13, 19
FracFeat, 56
fractal dimension, 46, 47
fractal feature extraction, 55
fractal geometry, 21
fractal image compression, 7, 44, 48
Fractal Imager, 49
fractal transform, 133
Fractals Everywhere, 44

geometric transform, 48, 55, 57

Hausdorff metric, 45
histogram matching, 131
human perception, 21

ICT, 129
Image recognition, 3
Information and Communication Technology, 129
Information Technology, 4, 129
intelligent clothing, 5
internal level, 11
IT, 4
iterated function system, 44, 154

logistic regression model, 137

massic transform, 48, 49, 55
Metadata, 13
Minkovsky distance, 19
multicompression, 56, 59

n-dimensional Euclidian space, 46
neighborhood, 47

Object Centered Neglect, 132
off line reasoning, 29
online learning, 29

open ball, 47

Pariss, 30
partial relevance of a feature, 22
partial relevance of images, 22
partioned iterated function system, 48
partition, 48
Perceptual Metadata, 13
placement rule, 21
projection engine, 136
pseudo-code, 49
Psychological Metadata, 13

quad tree depth, 58
query space, 19

range block, 48, 57
range pool, 58
relational database, 11
relevance feedback, 7, 22, 29–31, 136
root mean square error, 133, 134

second generation visual information retrieval systems, 13
Semantic Metadata, 13
Sierpinsky triangle, 46
similar, 14
similarity measure, 19, 23
space of fractals, 45
spatial dimension feature, 58
SQL, 12
statistical texture, 58
Subjectivity of the features, 14
Subjectivity of the image, 14
Subjectivity of the user, 14

Textural, 21, 57, 131
    Coarseness, 21, 57, 131
    Contrast, 21, 57, 131
    Directionality, 21, 57, 131
    Line-likeness, 21, 57, 131
    Regularity, 21, 57, 131
    Roughness, 21, 57, 131
Texture, 21
    statistical, 21
    structural, 21
Turing, 135

Virage, 27
Visual information retrieval, 4
visual information systems, 4
visual intelligence, 29
visual signs, 30