THE LOCAL BINARY PATTERN APPROACH TO TEXTURE ANALYSIS – EXTENSIONS AND APPLICATIONS

TOPI MÄENPÄÄ

Infotech Oulu and
Department of Electrical and
Information Engineering,
University of Oulu

OULU 2003

TOPI MÄENPÄÄ

THE LOCAL BINARY PATTERN APPROACH TO TEXTURE ANALYSIS – EXTENSIONS AND APPLICATIONS

Academic Dissertation to be presented with the assent of the Faculty of Technology, University of Oulu, for public discussion in Kuusamonsali (Auditorium YB210), Linnanmaa, on August 8th, 2003, at 12 noon.

OULUN YLIOPISTO, OULU 2003

Supervised by
Professor Matti Pietikäinen

Reviewed by
Doctor Jukka Iivarinen
Professor Maria Petrou

ISBN 951-42-7076-2 (URL: http://herkules.oulu.fi/isbn9514270762/)

ALSO AVAILABLE IN PRINTED FORMAT
Acta Univ. Oul. C 187, 2003
ISBN 951-42-7075-4
ISSN 0355-3213 (URL: http://herkules.oulu.fi/issn03553213/)

OULU UNIVERSITY PRESS
OULU 2003

Mäenpää, Topi, The local binary pattern approach to texture analysis – extensions and applications
Infotech Oulu and Department of Electrical and Information Engineering, University of Oulu, P.O.Box 4500, FIN-90014 University of Oulu, Finland
Oulu, Finland
2003

Abstract

This thesis presents extensions to the local binary pattern (LBP) texture analysis operator. The operator is defined as a gray-scale invariant texture measure, derived from a general definition of texture in a local neighborhood. It is made invariant against the rotation of the image domain, and supplemented with a rotation invariant measure of local contrast. The LBP is proposed as a unifying texture model that describes the formation of a texture with micro-textons and their statistical placement rules.

The basic LBP is extended to facilitate the analysis of textures with multiple scales by combining neighborhoods with different sizes. The possible instability in sparse sampling is addressed with Gaussian low-pass filtering, which seems to be somewhat helpful.

Cellular automata are used as texture features, presumably for the first time ever. With a straightforward inversion algorithm, arbitrarily large binary neighborhoods are encoded with an eight-bit cellular automaton rule, resulting in a very compact multi-scale texture descriptor. The performance of the new operator is shown in an experiment involving textures with multiple spatial scales.

An opponent-color version of the LBP is introduced and applied to color textures. Good results are obtained in static illumination conditions. An empirical study with different color and texture measures however shows that color and texture should be treated separately.

A number of different applications of the LBP operator are presented, emphasizing real-time issues. A very fast software implementation of the operator is introduced, and different ways of speeding up classification are evaluated. The operator is successfully applied to industrial visual inspection applications and to image retrieval.

Keywords: cellular automata, classification, color, multi-scale, real-time

Acknowledgements

The work reported in this thesis was carried out in the Machine Vision Group of the Department of Electrical and Information Engineering at the University of Oulu, Finland. Many of the ideas and experiments presented are results of very close co-operation with many people, to whom I want to express my gratitude. Especially, I would like to acknowledge Professor Matti Pietikäinen for supervising the thesis and for the valuable support I have received from him, and Docent Timo Ojala for his expert advice and the work he has put in to our common publications. In addition, I would like to mention Matti Niskanen and Markus Turtinen for their contribution to slowing down the work with the countless coffee breaks they arranged.

I am grateful to Professor Maria Petrou and Doctor Jukka Iivarinen for reviewing the thesis and for the comments and constructive criticism I received. For the financial support obtained for this thesis, I would like to thank the Graduate School in Electronics, Telecommunications, and Automation, the Nokia Foundation, the Tauno Tönning Foundation, the Foundation for the Promotion of Technology (TES), and the Ulla Tuominen Foundation.

Finally, I would like to give thanks to my wife Anna for providing a support to lean on. Our three children, Siiri, Elmeri, and Oskari, also need to be acknowledged for being great counterbalances to work.

Oulu, May 2003 Topi Mäenpää

Original publications

I	Mäenpää T, Pietikäinen M & Ojala T (2000) Texture classification by multi-predicate local binary pattern operators. Proc. 15th International Conference on Pattern Recognition, Barcelona, Spain, 3: 951–954.
II	Mäenpää T, Ojala T, Pietikäinen M & Soriano M (2000) Robust texture classification by subsets of local binary patterns. Proc. 15th International Conference on Pattern Recognition, Barcelona, Spain, 3: 947–950.
III	Ojala T, Mäenpää T, Pietikäinen M, Viertola J, Kyllönen J & Huovinen S (2002) Outex — New framework for empirical evaluation of texture analysis algorithms. Proc. 16th International Conference on Pattern Recognition, Québec City, Canada, 1: 701–706.
IV	Ojala T, Pietikäinen M & Mäenpää T (2002) Multiresolution gray scale and rotation invariant texture analysis with local binary patterns. IEEE Transactions on Pattern Analysis and Machine Intelligence 24(7): 971–987.
V	Mäenpää T & Pietikäinen M (2003) Multi-scale binary patterns for texture analysis. Proc. 13th Scandinavian Conference on Image Analysis, Göteborg, Sweden, in press.
VI	Mäenpää T, Pietikäinen M & Viertola J (2002) Separating color and pattern information for color texture discrimination. Proc. 16th International Conference on Pattern Recognition, Québec City, Canada, 1: 668–671.
VII	Ojala T, Mäenpää T, Viertola J, Kyllönen J, Pietikäinen M (2002) Empirical evaluation of MPEG-7 texture descriptors with a large-scale experiment. Proc. The 2nd International Workshop on Texture Analysis and Synthesis, Copenhagen, Denmark, pp 99–102.
VIII	Mäenpää T, Viertola J & Pietikäinen M (2003) Optimizing color and texture features for real-time visual inspection. Pattern Analysis and Applications. vol. 6, in press.
IX	Mäenpää T, Turtinen M & Pietikäinen M (2003) Real-time surface inspection by texture. Real-Time Imaging, accepted.

The writing of papers I and II and the implementations of the experiments reported in them was mainly carried out by the author. Much of the ideas on which the material was based originated from the co-authors. With Paper III, the author was in charge of building the texture imaging system. The author also designed the system with Dr. Ojala and Prof. Pietikäinen, and assisted in writing the parts of the publication dealing with technical details. Papers I and II created the basis for Paper IV, although the paper itself was mostly written by Dr. Ojala and Prof. Pietikäinen. The role of the author was to implement analysis methods, and to perform experiments. Also in Paper VII, the author had quite a minor role in the writing process, but was responsible for building the experimental set-up. Paper VI and Paper VIII were written by the author, while Mr. Viertola performed most of the experiments. In Paper IX, the author performed most of the experiments, and mainly wrote the paper. All experiments in Paper V were performed and the paper completely written by the author. Prof. Pietikäinen served as a valuable source of comments and guidance, just as with the other papers.

Symbols and abbreviations

BTCS	Binary texture co-occurrence spectrum
CBIR	Content-based image retrieval
CCD	Charged coupled device, an image sensor in digital cameras
CCR	Coordinated clusters representation
CIE	Commission Internationale de l'Eclairage, International Commission on Illumination
CAM	Cellular automata machine
CPU	Central processing unit
CISC	Complex instruction set computer
DCT	Discrete cosine transform
GLC	Gray-level co-occurrence
GLCM	Gray-level co-occurrence matrix
GLD	Gray-level difference
GLTCS	Gray level texture co-occurrence spectrum
GMRF	Gaussian Markov random field
HSV	Hue, saturation, value, a color model
IEC	International Electrotechnical Commission
ISO	International Organization for Standardization
LBP	Local binary pattern
LBPCA	LBP coupled with cellular automaton rules
LBPF	Low-pass filtered LBP
LBP/C	LBP and contrast (joint distribution)
L*a*b*	Perceptually uniform color space, defined by CIE
LVQ	Learning vector quantization
MPEG	Moving Micture Experts Group, a working group of ISO/IEC
MRF	Markov random field
MSB	Most significant binary digit (bit)
OCLBP	Opponent color LBP

rg	Intensity-normalized chromaticity space
SFFS	Sequential forward floating search, a feature selection method
SOM	Self-organizing map
SPD	Spectral power distribution
TS	Texture spectrum
TU	Texture unit
TUTS	Texture unit texture spectrum
VAR	Local variance, a measure of image contrast
XM	eXperimentation Model, part of the MPEG-7 standard
XML	eXtensible Markup Language
ZCTCS	Zero crossings texture co-occurrence spectrum
x	Universal variable used with many functions
T	Local image texture
$t(x)$	Distribution of values
G_P	Circularly symmetric neighbor set
P	The number of samples in a neighbor set
R	The radius of a neighbor set
g_c	Gray value of the center of a neighborhood
g_p	Gray value of a pixel in a neighborhood
x_c	The x coordinate of the center of a neighborhood
y_c	The y coordinate of the center of a neighborhood
p	Index of a pixel in a neighborhood; probability
a_k	Binary weight
k	Bit index
$s(x)$	Sign function
$U(x)$	Uniformity measure
$\mathrm{ROR}(x,i)$	Circular shift to the right
$F_b(x,i)$	A function that extracts b bits from a binary number
B	The number of bins in a discrete distribution
b	Index of a bin in a distribution; the number of bits with $F_b(x,i)$; the outcome of a CA rule with R_b^h
S	Feature distribution of a texture sample; the number of scales used with the LBPCA operator.
M	Feature distribution of a texture model
$G(S,M)$	Log-likelihood ratio statistic (G statistic)
$L(S,M)$	Log-likelihood statistic
C	Index of the most likely class
i	Class index; number of shifts with the ROR function; bit index
n	Scale index with low-pass filtering and with cellular automata
t	An integration variable

r_n	The outer radius of the "effective area" of a low-pass filtered sample in an LBP code on scale n
μ	Mean value
σ	Standard deviation
σ_n	The standard deviation of a Gaussian low-pass filter on scale n
w_n	The spatial size of a Gaussian low-pass filter on scale n
$f_G(x, y)$	Bivariate Gaussian distribution in Cartesian coordinates
$f_G(r, \theta)$	Bivariate Gaussian distribution in polar coordinates
$\mathrm{erf}_{2\mathrm{D}}(r)$	Two-dimensional "error function"
R_b^h	A set containing the occurrences of a certain binary neighborhood with a certain outcome
h	A three-bit binary neighborhood
r	The most likely cellular automaton rule for a pixel's neighborhood

Contents

1 Introduction

1.1 Texture and its properties

Texture can be broadly defined as the visual or tactile surface characteristics and appearance of something. Textures can consist of very small elements like sand, or of huge elements like stars in the Milky Way. Texture can also be formed by a single surface via variations in shape, illumination, shadows, absorption and reflectance. Just about anything in the universe can appear as a texture if viewed from a proper distance. However, it is important to recognize that "texture regions give different interpretations at different distances and at different degrees of visual attention" (Chaudhuri *et al.* 1993). A single star, viewed from a distance, is not a texture, but its surface might be.

In digital images, the characteristics of a texture can be sensed via variations in the captured intensities or color. Although in general there is no information on the cause of the variations, differences in image pixels provide a practical means of analyzing the textural properties of objects. Unfortunately, no one has so far been able to define digital texture in mathematical terms. It is doubtful whether anyone ever will. Haralick *et al.* (1973) noted: "texture has been extremely refractory to precise definition". Ten years later, Cross & Jain (1983) put it simply: "We consider a texture to be a stochastic, possibly periodic, two-dimensional image field." But thirteen years later, it was not so clear any more: "Texture, despite eluding formal definition, has found many applications in computer vision." (Jain & Karu 1996).

The lack of a theory makes the problem of analyzing textures ill-posed from the point of mathematics, as analysis methods can be proved neither correct nor wrong. Thus, evaluation must be carried out in an empirical manner. However, texture measures have been successfully used in many machine vision tasks.

A number of people have found ways of categorizing textures according to their appearance. One, quite crude way, is to divide textures into two main categories, namely stochastic and deterministic. The most important features for the human visual system are, according to Tamura *et al.* (1978), line-likeness, regularity and roughness. Rao & Lohse (1993) in turn categorize textures with three orthogonal dimensions: repetitive vs. non-repetitive, non-directional with high contrast vs.

directional with low contrast, and simple granular textures vs. fine-grained complex textures. In a multichannel model based on human vision, coarseness, edge orientation, and contrast are seen as the most important attributes (Levine 1985, pp. 426–430).

1.2 Motivation

Texture analysis plays an important role in many image analysis applications. Even though color is an important cue in interpreting images, there are situations where color measurements just are not enough — nor even applicable. In industrial visual inspection, texture information can be used in enhancing the accuracy of color measurements. In some applications, for example in the quality control of paper web, there is no color at all. Texture measures can also cope better with varying illumination conditions, for instance in outdoor conditions. Therefore, they can be useful tools for high-level interpretation of natural scene image content. Texture methods can also be used in medical image analysis, biometric identification, remote sensing, content-based image retrieval, document analysis, environment modeling, texture synthesis and model-based image coding.

Since the sixties, texture analysis has been an area of intense research. Nonetheless, progress has been quite slow, introducing just a few noticeable improvements. The methods developed have only occasionally evolved into real-world applications. In short, analyzing real world textures has proved to be extremely hard. Maybe the most difficult problems are caused by the natural inhomogeneity of textures, varying illumination, and variability in the shapes of surfaces.

In most applications, image analysis must be performed with as few computational resources as possible. Especially in visual inspection, the speed of feature extraction may play an enormous role. The size of the calculated descriptions must also be kept as small as possible to facilitate classification.

Often, the Gabor filtering method of Manjunath & Ma (1996) is credited as being the current state-of-the-art in texture analysis. It has shown very good performance in a number of comparative studies. The strength of the method may be attributed to the incorporation of the analysis of both spatial frequencies and local edge information. Although theoretically elegant, it tends to be computationally very demanding, especially with large mask sizes. It is also affected by varying illumination conditions.

To meet the requirements of real-world applications, texture operators should be computationally cheap and robust against variations in the appearance of a texture. These variations may be caused by uneven illumination, different viewing positions, shadows etc. Depending on the application, texture operators should thus be invariant against illumination changes, rotation, scaling, viewpoint, or even affine transformations including perspective distortions. The invariance of an operator cannot however be increased to the exclusion of discrimination accuracy. It is easy to design an operator that is invariant against everything, but totally useless as a texture descriptor.

The local binary pattern (LBP) operator was developed as a gray-scale invariant pattern measure adding complementary information to the "amount" of texture in images. It was first mentioned by Harwood *et al.* (1993), and introduced to the public by Ojala *et al.* (1996). Later, it has shown excellent performance in many comparative studies, in terms of both speed and discrimination performance. In a way, the approach is bringing together the separate statistical and structural approaches to texture analysis, opening a door for the analysis of both stochastic microtextures and deterministic macrotextures simultaneously. It also seems to have some correspondence with new psychophysical findings in the human visual system. Furthermore, being independent of any monotonic transformation of gray scale, the operator is perfectly suited for complementing color measurements — or to be complemented by an orthogonal measure of image contrast. The LBP operator can be made invariant against rotation, and it also supports multi-scale analysis.

1.3 The contribution of the thesis

In its basic form, the LBP operator cannot properly detect large-scale textural structures, which can be considered its main shortcoming. The small local neighborhood also affects the rotation invariant version, as local artifacts tend to decrease its performance.

In this thesis, the basic LBP operator is extended in a number of ways. It is made to measure texture information on multiple scales through the use of arbitrary neighborhood sizes, and any number of neighborhood samples. Based on statistical analysis of pattern occurrences, the concept of uniform patterns is introduced and successfully used in enhancing the robustness and speed of rotation invariant texture analysis. This approach, in addition to a greedy search method, is also used to make the operator more robust against 3-D distortions. The multi-resolution version is also combined with Gaussian filtering to account for the possible problems caused by sparse sampling. Furthermore, a novel way of encoding arbitrarily large binarized neighborhoods with cellular automata is presented.

To be able to use the LBP as a supplement to color measurements in visual inspection, a method of optimizing independent color and texture measures is introduced. With an empirical evaluation of color indexing, texture and color-texture methods it is shown that color and texture patterns are separate phenomena that can — or even should — be used separately. Apart from color, this argument is shown to be applicable to other orthogonal measures, like contrast. Finally, the computational performance of the LBP operator is greatly enhanced.

The performance of the LBP is shown in large comparative studies. The discrimination performance is demonstrated to be as high as that of any current texture analysis method, or even better. At the same time, its computational burden is a fraction of that of the others.

1.4 Summary of original papers

This thesis consists of nine publications. Paper I introduces the multiresolution LBP and evaluates its performance in a supervised texture segmentation problem from Randen & Husoy (1999). In all but two segmentation problems, the multiresolution LBP is shown to perform better than any of the methods evaluated by Randen & Husoy.

The application-specific optimization of LBP distributions by pruning is first presented in Paper II. A set of textures in the CUReT (Dana *et al.* 1999) database is utilized. It is shown that the classification accuracy of tilted textures can be enhanced by a greedy search procedure discarding useless or even harmful LBP codes. In this paper, the concept of "uniform" patterns is also introduced, based on statistical analysis of pattern occurrences in a set of Brodatz textures.

Paper III describes the Outex image database that was created to facilitate the empirical evaluation of texture analysis algorithms. The database currently contains 319 different textures, each imaged under three different light sources, six different spatial resolutions, and nine rotation angles, resulting in a total number of 162 images per texture sample. The texture images are used first in Paper IV as a test bench for rotation and gray scale invariant texture classification.

The definition of LBP was extended to its current general form with arbitrary, circularly sampled neighborhoods in Paper IV. A multiresolution operator is successfully applied to a set of Brodatz textures used by Porter & Canagarajah (1997) in a comparative study of rotation invariant analysis methods. As a more challenging setup, a set of Outex textures is also used with two variations. First, only rotation invariance is tested. In the second form, rotation and gray scale invariance are required at the same time. Very high classification rates are obtained in all three problems with joint distributions of the multiresolution LBP and a local variance measure.

Two new ways of extending the LBP operator to multiple resolutions are presented in Paper V. The multi-resolution version presented in Paper IV is combined with Gaussian low-pass filtering to solve the possible problems of sparse sampling. Furthermore, a novel way of combining the multi-resolution LBP with cellular automata is presented. A feature vector containing the marginal distributions of LBP codes and cellular automata rules is shown to be a powerful texture measure when there is a need to cope with variations in image scale.

In Paper VI, different color, texture and joint color-texture methods are evaluated with a set of natural textures in static and varying illumination conditions. As a conclusion, the paper suggests that color and texture are phenomena that can — or even should — be treated separately.

In Paper VII, a large-scale experiment is performed to compare the performance of texture analysis methods in a retrieval problem. All textures in the Outex database are utilized, resulting in one of the largest — or maybe the largest — number of texture classes ever used in experiments. The texture descriptors in the MPEG-7 multimedia content description standard are evaluated, in addition to the LBP. The results show that the discriminative power of the LBP is superior to that of the others. In addition, the LBP operator is shown to be up to 6000 times faster than the others. Again, the accuracy of the LBP can be enhanced with the multiresolution version.

The optimization method presented in Paper II is extended to joint color-texture descriptions in Paper VIII. An alternative method, based on genetic algorithms, is also proposed. It is, however, shown that the deterministic, greedy search produces better results in almost every case, in terms of classification accuracy and feature vector length. The method is applied to the detection of defects in parquet slabs. Although this application seems to obtain no benefit from optimized texture descriptors, the color descriptors are successfully optimized and combined with texture for better performance.

The computational performance of the LBP is addressed in Paper IX, in which a framework for real-time surface inspection with the LBP is presented. The paper describes a fast implementation of the LBP operator. LBP distributions are successfully pruned with the SFFS feature selection algorithm (Pudil *et al.* 1994) and self-organizing maps (Kohonen 1997) to achieve real-time performance in a very demanding inspection application.

2 Texture analysis

2.1 Paradigms

Texture analysis methods have been traditionally divided into two categories. The first one, called the statistical or stochastic approach, treats textures as statistical phenomena. The formation of a texture is described with the statistical properties of the intensities and positions of pixels. Difference histograms and co-occurrence statistics, researched by Haralick *et al.* (1973), Weszka *et al.* (1976), and Unser (1986), can serve as simple examples of statistical texture measures. These kinds of texture models work best with stochastic microtextures. The second category, called the structural approach, introduces the concept of texture primitives, often called texels or textons. To describe a texture, a vocabulary of texels and a description of their relationships is needed. The goal is to describe complex structures with simpler primitives, for example via graphs. Structural texture models work well with macrotextures with clear constructions. Through the pioneering work of Julesz (1981) and Beck *et al.* (1983), primitive-based models have been widely used in explaining human perception of textures.

Another way of classifying texture methods has been proposed by Chellappa & Manjunath (2001). According to them, modern methods either try to understand the process of texture formation, or base themselves on the theory of human perception. Tuceryan & Jain (1999) have a more fine-grained categorization in which texture methods are divided into geometric, model based, statistical and signal processing based approaches.

The process of texture formation can be described for example by Markovian processes or auto-regressive models. In the near past, Gaussian Markov random fields were extensively used as a general texture model, studied for example by Chellappa *et al.* (1999). Since the beginning of the eighties, the signal processing model has gained considerable interest. Maybe the first work linking a human visual model to texture analysis was represented by Faugeras (1978). The proposal of Campbell & Robson (1968), later confirmed by DeValois *et al.* (1982), linking scale and orientation selectivity to the visual cortex has later inspired for example Malik & Perona (1990) in their study of preattentive texture discrimination. Nowadays, Gabor (1946) filtering, first used in image analysis by Granlund (1978) (although many people still cite Daugman (1988)), might be the most commonly used texture analysis method in the research community (Jain & Farrokhnia 1991, Manjunath & Ma 1996).

2.2 Classification

The goal of classification in general is to select the most appropriate category for an unknown object, given a set of known categories. Since perfect classification is often impossible, the classification may also be performed by determining the probability for each of the known categories. Another approach is to use an additional *reject* class for objects that cannot be classified with a sufficient degree of confidence (Shapiro & Stockman 2001). The objects are presented with feature vectors that describe their characteristics with numbers. (Duda *et al.* 2001)

Classifiers have been traditionally divided into two categories: parametric and non-parametric. The difference is in that parametric classifiers, like Bayesian and Mahalanobis classifiers, make certain assumptions about the distribution of features. Non-parametric classifiers, like the k-NN classifier, can be used with arbitrary feature distributions and with no assumptions about the forms of the underlying densities (Duda *et al.* 2001).

Both parametric and non-parametric classifiers need some knowledge of the data, be it either training samples or parameters of the assumed feature distributions. They are therefore called supervised techniques. With non-supervised techniques, classes are to be found with no prior knowledge. This process is often called clustering. Examples of such methods include vector quantization, utilized in texture classification, for example by McLean (1993), and self-organizing maps (Kohonen 1997). Different classification techniques are well covered by Theodoridis & Koutroumbas (1999).

A supervised classification process involves two phases. First, the classifier must be presented with known training samples or other knowledge of feature distributions. Only after that can the classifier be used in recognizing unknown samples. Prior to the training and the recognition, the samples must be processed with a texture analysis method to get a feature vector. Any method, for example those presented in Section 2.1, can be used for this purpose. Different feature extraction and classification principles have been evaluated for example by Weszka *et al.* (1976), Conners & Harlow (1980), and Randen & Husoy (1999). For a concise description of texture classification, see Ojala & Pietikäinen (1999a).

2.3 Segmentation

Segmentation is a process in which an image is partitioned into regions that present meaningful objects or areas in the image. Such can be, for example, humans, cars, buildings, and the sky. Segmentation has two objectives: to decompose the image into parts for further analysis, and to provide a more meaningful or a more efficient representation. The extracted regions should be uniform with respect to some measurable characteristics, like color, intensity, or texture. Obviously, regions adjacent to each other should be different with respect to these characteristics. Some other properties, like smooth region borders and a relatively large region size, can be used in further constraining the segmentation process. (Shapiro & Stockman 2001)

Image segmentation algorithms can be divided into two major categories: supervised and non-supervised. With non-supervised segmentation, the types of different objects, and even the number of objects or object types are unknown. With supervised segmentation, the algorithms can utilize prior knowledge of the number of objects or their types. Another categorization divides algorithms to region-based and boundary-based.

Most texture analysis operators provide a measure of the image texture for each pixel, or for small areas in an image. The process of region-based image segmentation involves grouping adjacent areas with similar textural characteristics together. With boundary-based segmentation, the goal is to find areas where textural properties change rapidly. For both tasks, a large number of different algorithms exist. Often, the algorithms are applicable not only to texture, but also to color images. Many of the early texture segmentation algorithms are influenced by the recursive region splitting algorithm of Ohlander *et al.* (1978), which is credited to be the first such work on natural color images.

The first non-supervised region-based texture segmentation algorithms used methods like split-and-merge (Chen & Pavlidis 1979) or pyramid node linking (Pietikäinen & Rosenfeld 1981) (Ojala 1997). A good overview of the early methods was presented by Haralick & Shapiro (1985), and should be consulted for comprehensive coverage. Later, the segmentation problem has been tackled for example with neural networks (Jain & Karu 1996, Çesmeli & Wang 2001), robust statistics (Boyer *et al.* 1994), fractal dimension (Chaudhuri & Sarkar 1995), and hidden Markov models (Chen & Kundu 1995). The approach of Zhu & Yuille (1996), called *region competition*, combines aspects of the region growing (Adams & Bischof 1994) and snake (Kass *et al.* 1988) methods.

Shi & Malik (2000) proposed a new approach to the segmentation problem. Instead of analyzing local features, they aim at finding the “global impression of an image”. In this approach, image segmentation is treated as a graph partitioning problem, and a criterion called a *normalized cut* is used in finding an optimal segmentation.

An example of an artificially generated segmentation problem is shown in Figure 2.1 (left). Synthetic problems are common in evaluating the performance of segmentation algorithms because the “correctness” of the result can be easily compared to a known *ground truth* (right). With natural images, the production of the ground truth is a challenging problem, and cannot typically be solved unambiguously.

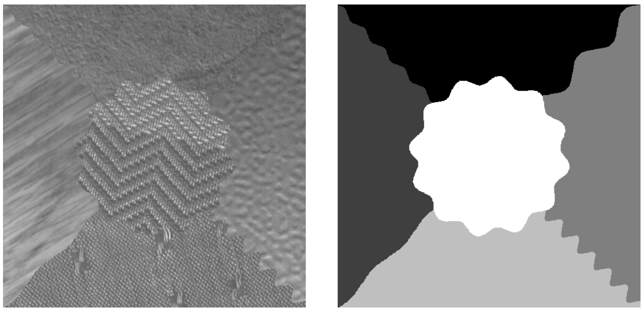

Figure 2.1. An artificially generated segmentation problem (left) and the corresponding ground truth.

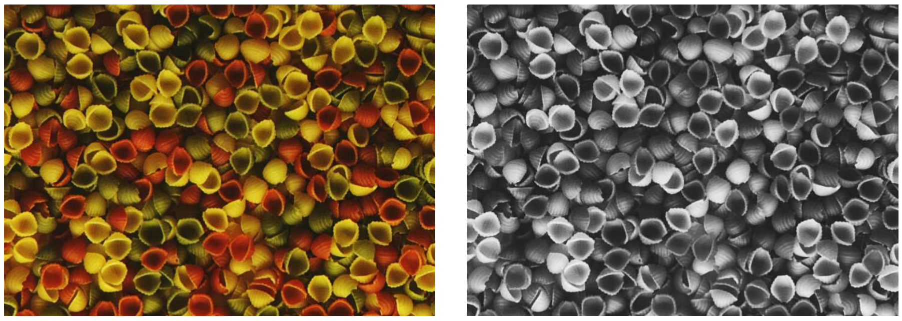

Figure 2.2. A color texture and its gray-scale version.

2.4 Color and texture

Texture analysis methods have been developed with gray-scale images, intuitively for good reasons. Humans can easily capture the textures on a surface, even with no color information. Figure 2.2 shows a photograph of tricolor pasta and its gray-scale version. The only thing that cannot be told, based on the gray-scale information, is the color of the pasta — the texture itself is the same. The human visual system is able to interpret practically achromatic scenes for example in low illumination levels. Color acts just as a cue for richer interpretations. Even when color information is distorted, for example due to color blindness, the visual system still works. Intuitively, this suggests that at least for our visual system, color and texture are separate phenomena. Nevertheless, the use of joint color-texture features has been a popular approach to color texture analysis.

One of the first methods allowing spatial interactions within and between spectral bands was proposed by Rosenfeld *et al.* (1982). Statistics derived from co-occurrence matrices and difference histograms were considered as texture descriptors. Panjwani & Healey (1995) introduced a Markov random field model for color images which captures spatial interaction both within and between color bands. Jain & Healey (1998) proposed a multiscale representation including unichrome features computed from each spectral band separately, as well as opponent color features that capture the spatial interaction between spectral bands. Recently, a number of other approaches allowing spatial interactions have been proposed.

In some approaches, only the spatial interactions within bands are considered. For example, Caelli & Reye (1993) proposed a method which extracts features from three spectral channels by using three multiscale isotropic filters. Paschos (2001) compared the effectiveness of different color spaces when Gabor features computed separately for each channel were used as color texture descriptors. Segmentation of color images by taking into account the interaction between color and the spatial frequency of patterns was recently proposed by Mirmehdi & Petrou (2000). Recently, Gevers & Smeulders (2001) published work on color constant ratio gradients.

In the human eye, color information is processed at a lower spatial frequency than intensity (Wandell 1995). This fact is utilized in image compression and also in imaging sensors. Because of the fact that photographs are usually intended for a human audience, there is no need to acquire colors in high resolution. For example, CCD chips in consumer electronics measure color on each pixel using just one sensor band instead of all three. There is only one color receptor (R, G, or B) for each pixel. Therefore, the full color information must be calculated from the neighbors using an interpolation routine of some type. All this suggests that unlike texture, color should be treated as a regional property whose interactions at pixel level are not very important.

The argument about separate color and texture information is not based on pure intuition. Research on the human visual system has provided much evidence that the image signal is indeed composed of a luminance and a chrominance component. Both of these are processed by separate pathways (Papathomas *et al.* 1997, DeYoe & van Essen 1996), although there are some secondary interactions between the pathways (DeValois & DeValois 1990). Also the psychophysical studies of Poirson & Wandell (1996) suggest that color and pattern information are processed separately.

In their recent paper on the vocabulary and grammar of color patterns, Mojsilovic *et al.* (2000) suggest that the overall perception of color patterns is formed through the interaction of a luminance component, a chrominance component and an achromatic pattern component. The luminance and chrominance components approximate signal representation in early visual cortical areas while the achromatic pattern component approximates the signal formed at higher processing levels. The luminance and chrominance components are used in extracting color-based information, while the achromatic pattern component is utilized as texture pattern information. Mojsilovic *et al.* conclude that human perception of pattern is unrelated to the color content of an image. The strongest dimensions are "overall color" (presence/absence of dominant color) and "color purity" (degree of colorfulness), indicating that, at the coarsest level of judgment, people primarily use color information to judge similarity. The pure texture-based dimensions

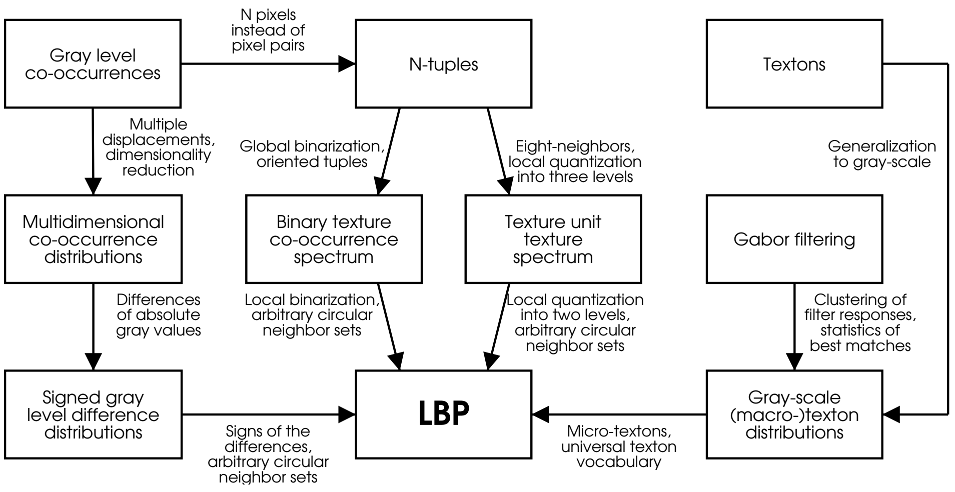
Gray level co-occurrences
N pixels instead of pixel pairs
N-tuples
Textons
Multiple displacements, dimensionality reduction
Global binarization, oriented tuples
Eight-neighbors, local quantization into three levels
Generalization to gray-scale
Multidimensional co-occurrence distributions
Binary texture co-occurrence spectrum
Texture unit texture spectrum
Gabor filtering
Differences of absolute gray values
Local binarization, arbitrary circular neighbor sets
Local quantization into two levels, arbitrary circular neighbor sets
Clustering of filter responses, statistics of best matches
Signed gray level difference distributions
LBP
Gray-scale (macro-)texton distributions
Signs of the differences, arbitrary circular neighbor sets
Micro-textons, universal texton vocabulary

Figure 2.3. **LBP in the field of texture operators.**

are ”directionality and orientation” and ”regularity and placement rules”. The optional fifth dimension, ”pattern complexity and heaviness” (a dimension of general impression), appears to contain both chrominance and luminance information perceived, for example, as ”light”, ”soft”, ”heavy”, ”busy” and ”sharp”.

A number of approaches utilizing separate color and texture information have been proposed. For example, Tan & Kittler (1993) extracted texture features based on the discrete cosine transform from a gray level image, while measures derived from color histograms were used for color description. A granite classification problem was used as a test bed for the method. Dubuisson-Jolly & Gupta (2000) proposed a method for aerial image segmentation, in which likelihoods are computed independently in color and texture spaces. Then, the final segmentation is obtained by evaluating the certainty with which each classifier (color or texture alone) would make a decision.

In some applications, like paper and textile inspection, texture must be used due to the fact that there is no color information (See e.g. Baykut *et al.* 2000, Kumar & Pang 2002). However, in many applications including some visual inspection tasks, color is the main information source. Color descriptors have been used in inspecting wood, ceramic tiles, food etc. (Boukouvalas *et al.* 1999, Kauppinen 1999). Sometimes, however, color features alone do not provide sufficiently accurate information. Uneven illumination conditions, similarly colored but differently textured surfaces, small defects and many other phenomena call for support from texture analysis methods.

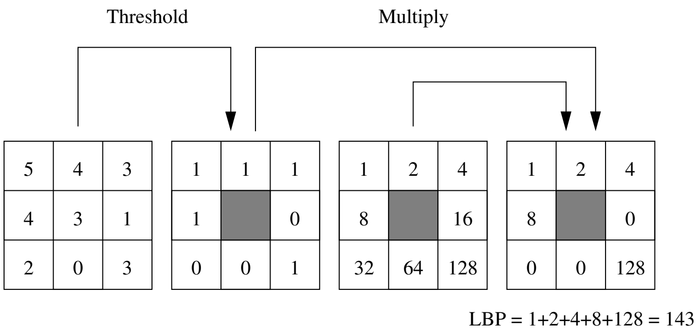

Figure 2.4. Calculating the original LBP code and a contrast measure.

2.5 Methods related to the LBP

In this section, LBP is positioned in the field of texture operators. The roots of the operator are described starting from co-occurrence statistics, N-tuple methods, and from textons. Although LBP was not directly derived from any of the methods presented, it is very closely related to some of them. In Figure 2.3, the related texture analysis methods are represented as a diagram. The arrows in the diagram represent the relations between different methods, and the texts beside the arrows summarize the main differences between them.

2.5.1 The original LBP

The LBP operator was first introduced as a complementary measure for local image contrast (Harwood *et al.* 1993, Ojala *et al.* 1996). The first incarnation of the operator worked with the eight-neighbors of a pixel, using the value of the center pixel as a threshold. An LBP code for a neighborhood was produced by multiplying the thresholded values with weights given to the corresponding pixels, and summing up the result (Figure 2.4).

Since the LBP was, by definition, invariant to monotonic changes in gray scale, it was supplemented by an orthogonal measure of local contrast. Figure 2.4 shows how the contrast measure (C) was derived. The average of the gray levels below the center pixel is subtracted from that of the gray levels above (or equal to) the center pixel. Two-dimensional distributions of the LBP and local contrast measures were used as features. The operator was called LBP/C, and very good discrimination rates were reported with textures selected from the photographic album of Brodatz (1966) (Ojala *et al.* 1996).

2.5.2 Co-occurrences and gray-level differences

The gray-level co-occurrence (GLC) statistics were first described by Haralick *et al.* (1973), and are still being actively developed. Typically, GLC features are extracted in two stages. First, a set of gray-level co-occurrence matrices (GLCM) is computed. This happens by selecting a few *displacement operators*, with which the image is scanned. Some typical sets of displacements are shown in Figure 2.5. When an image is processed, all the pairs of pixels are found that are positioned relative to each other as the displacement operator indicates. A statistic of the gray levels of these pixel pairs is collected in a two dimensional co-occurrence matrix. The number of rows and columns in this matrix equals the number of gray levels in the image. In other words, the co-occurrence matrix represents the joint probability of the pairs of gray levels that occur at pairs of points separated by the displacement operator (Tomita & Tsuji 1990). Since the co-occurrence matrix collects information about pixel pairs instead of single pixels, it is called a second-order statistic.

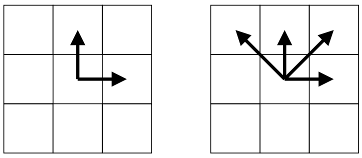

Figure 2.5. Displacement operators for the calculation of co-occurrence matrices.

In the second stage, the actual features are derived from an averaged co-occurrence matrix, or by averaging the features calculated from different matrices. The co-occurrence matrices give information about the homogeneity of an image, its contrast, linearity, etc. (Tomita & Tsuji 1990). Conners & Harlow (1980) chose five efficient measures extracted from the GLCMs: energy, entropy, correlation, homogeneity and inertia.

The gray-level difference (GLD) methods closely resemble the co-occurrence approach (Weszka *et al.* 1976). The difference is that instead of the absolute gray levels of the pair of pixels, their difference is utilized. In this way it is possible to achieve invariance against changes in the overall luminance of an image. Another advantage is that in natural textures, the differences tend to have a smaller variability to the absolute gray levels, resulting in more compact distributions. As features, Weszka *et al.* (1976) proposed the use of the mean difference, the entropy of differences, a contrast measure, and an angular second moment.

Valkealahti & Oja (1998) have presented a method in which multi-dimensional co-occurrence distributions are utilized. In their approach, the gray levels of a local neighborhood are collected into a high-dimensional distribution, whose dimensionality is reduced with vector quantization. For example, if the eight-neighbors are used, a nine-dimensional (eight neighbors and the pixel itself) distribution of gray values is created instead of eight two-dimensional co-occurrence matrices. An enhanced version of this method was later utilized by Ojala *et al.* (2001) with signed gray-level differences.

Looking at the derivation of LBP (Section 3.1) it is apparent that LBP can be regarded as a specialization of the multi-dimensional signed gray-level difference method. The reduction of dimensionality is directly achieved by considering only the signs of the differences, which makes the use of vector quantization unnecessary.

A comprehensive comparison of the original LBP, the LBP/C, GLCM, GLD, and many other methods has been carried out by Ojala *et al.* (1996) and Ojala (1997). As a conclusion, Ojala (1997, p. 58) writes: "Of the 22 local texture operators considered in the classification experiments LBP has generally been superior to other measures, providing excellent results especially for deterministic textures."

2.5.3 Texture spectrum and N-tuple methods

At the turn of the nineties, He & Wang introduced a new model of texture analysis based on the so-called texture unit (TU) (He & Wang 1990, Wang & He 1990). In their model, texture information was collected from a 3×3 neighborhood, which was divided into three levels according to the value of the center pixel. Each neighbor got the label 0, 1, or 2 depending on whether it was below, equal to or above the value of the center pixel, respectively. This resulted in a total number of $3^8 = 6561$ different texture units which were collected into a feature distribution called texture spectrum (TS). Consequently, the method is typically called "TUTS" in the literature.

The TUTS method closely resembles the original LBP. The only difference is that instead of the three levels proposed by He & Wang, thresholding to two levels is used. This makes the distribution more compact and reduces the effect of quantization artifacts. Furthermore, due to the two-level thresholding, the TUTS method is not strictly invariant to monotonic changes in gray values. Heikkinen (1993) reported a study in which LBP and TUTS were used in metal strip inspection. It was noted that there was no considerable difference with respect to classification accuracy. The difference in classification speed was however large, as the LBP feature vector is over 25 times shorter. The shorter feature vector has the additional advantage that with small textures, more reliable statistics can be obtained.

The N-tuple methods, first described by Aleksander & Stonham (1979), closely resemble the TUTS method. The main difference is in that whereas TUTS considers the eight-neighbors of a pixel, N-tuples consider N arbitrary samples. Patel & Stonham (1991) first used oriented N-tuple operators with globally thresholded binary images. The method was termed the binary texture co-occurrence spectrum (BTCS). The BTCS method was soon extended to gray-scale images by Patel & Stonham (1992), resulting in a gray level texture co-occurrence spectrum

(GLTCS). Rank coding was used to reduce the dimensionality of the features. Later, Hepplewhite & Stonham (1996) returned to the binary representation by using the BTCS with binarized edge images. This method was called the zero crossings texture co-occurrence spectrum (ZCTCS).

The concept of arbitrarily sampled "neighborhoods" in the N-tuple methods can be seen as a precursor of the circular sampling in LBP. Furthermore, the binary N-tuples in the BTSC method closely resemble LBP codes. The main difference is in the binarization, which is done locally in the LBP. Unfortunately, there seems to be no reported studies that compare the LBP to the N-tuple methods.

2.5.4 Texton statistics

Textons were originally used by Julesz (1981) as the basic units of human preattentive texture discrimination. The textures he considered consisted of separated binary texture primitives — orientation elements, crossings and terminators. Until recently, this model was not extended to gray-scale textures. Malik *et al.* (1999) reintroduced the term "texton" in a study of texture segmentation. In their model, multi-dimensional features are created by filtering with a Gabor filter bank, and representative vectors (i.e. textons) are found with k-means clustering. The texton vocabulary is created once for a set of textures, and used in describing all textures in it. Distributions of textons are used for texture description. Later, a number of people have used similar models for recognizing textures in the CUReT (Dana *et al.* 1999) database (Cula & Dana 2001a,b, Leung & Malik 1999, 2001, Varma & Zisserman 2002).

Varma & Zisserman (2002) consider the Gabor-based textons micro-primitives. However, even the smallest Gabor filters calculate the weighted mean of pixel values over a small neighborhood. LBP in turn considers each pixel in the neighborhood separately, thus providing even more fine-grained information. LBP can therefore be regarded as a micro-texton, as opposed to the Gabor-based macro-textons. Another difference is that in LBP there is no need to create an application-specific texton vocabulary. Instead, a generic vocabulary of micro-textons can be used for most purposes. The "Gabor texton" approach was recently compared to the LBP by Pietikäinen *et al.* (2003) with 3-D textures from the CUReT database. The LBP provided better performance even with a more challenging set of textures and with a significantly smaller computational overhead.

Recently, Sánchez-Yáñez *et al.* (2003) introduced a method called "coordinated clusters representation" (CCR) for texton-based texture discrimination. The CCR method extracts 3×3 neighborhoods from a globally binarized image, and uses the distribution of these local "textons" as a texture feature. The idea is very similar to the LBP, but with a major shortcoming: global binarization.

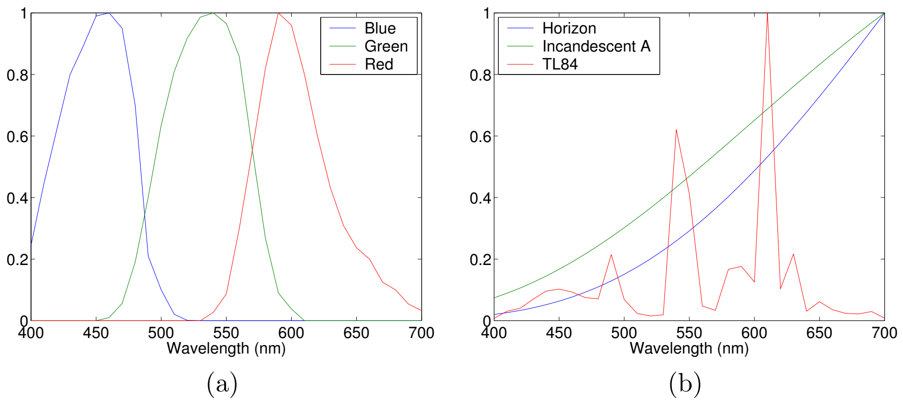

Figure 2.6. Spectral sensitivities of a Sony DXC-755P digital camera (a), and SPDs of light sources (b).

2.6 Empirical evaluation

One of the major problems in evaluating computer vision algorithms is the lack of a universally accepted framework for performance characterization. Several researchers have argued in favor of the systematic comparative evaluation of computer vision algorithms (Haralick 1992, Jain & Binford 1991, Pavlidis 1992, Phillips & Bowyer 1999, Price 1986). However, when it comes to texture analysis, “standardization” largely means using the same image sets for experiments.

For years, the photographic album of Brodatz (1966) has been the most widely exploited source of textures. The VisTex (MIT Media Lab 1995) and CUReT (Dana *et al.* 1999) databases are other well-known sources. The MeasTex database of Smith & Burns (1997) is an attempt to provide not only image data but also a framework for evaluating algorithms. However, it has not gained any considerable interest. Unfortunately, using images from the same databases does not guarantee comparable experimental results. The problem is most evident with Brodatz’ textures due to the fact that researchers have had to digitize the pictures themselves. Preprocessing, division into sub-images, and partitioning into training and testing sets all have a clear effect on the outcome of an experiment. Even when all these factors can be fixed, different criteria can be employed in quantifying the empirical evaluation.

In Paper III, a new framework for the empirical evaluation of texture analysis algorithms is presented. The framework, called Outex, is based on a large collection of natural textures. The database contains both surface textures and outdoor scenes, the former of which are imaged in a strictly controlled laboratory environment.

The database makes it possible to study a wide variety of aspects in texture analysis algorithms. Each surface texture sample is imaged under three different light sources, nine rotation angles, and six spatial resolutions. Thus, there are a

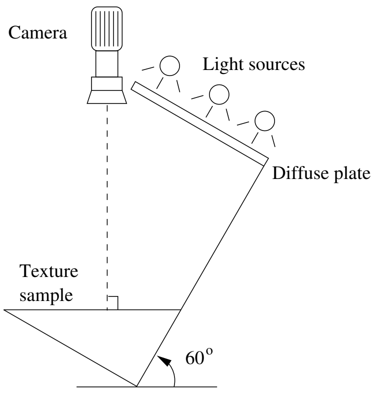

Figure 2.7. Relative positions of texture sample, light sources, and camera.

total number of 162 images per texture sample. A sketch of the imaging setup is shown in Figure 2.7. The properties of the light sources and the camera are well known (see Figure 2.6). With this data, it is easy to construct experiments for illumination, scale and rotation invariance.

To extend the evaluation framework beyond an image database, a number of texture classification, retrieval and segmentation problems have been constructed. The problems are precisely described in terms of test suites that define the input and output data. That is, preprocessing, division into sub-images, partitioning into training and testing, and the evaluation criterion are all fixed. Analysis methods are treated as “black boxes”; their internal properties, such as the number of features or discriminant functions, are of no interest as long as the desired output is obtained with the given input data.

The Outex database contains four basic types of test suites: texture classification (TC), texture retrieval (TR), supervised texture segmentation (SS), and unsupervised texture segmentation (US). A test suite may contain a large number of classification or segmentation problems with a common structure. Individual tests in a suite differ in input data. The large number of tests in a single suite is needed to facilitate thorough and rigorous evaluation of the properties of an algorithm. All parameters external to the algorithm are required to be constant in all tests in a suite. This way, the robustness of the algorithm with respect to variations in input data can be evaluated. Test suites are built to test different aspects of analysis algorithms. Robustness against changes in image scale, rotation, illumination, or any combination of these three can be measured.

In addition to the test suites built from Outex image data, a collection of contributed test suites is also provided. These test suites are built from data used in works published in well-known journals, providing a useful benchmark. All test suites are accompanied with a unique ID to make finding the correct data easy.

In the experiments presented in this thesis, the Outex database has been extensively used. For comparison purposes, some other sources of textures have also been utilized. As just noted, the results with these may not be fully comparable with those reported previously. When possible, these data have been compiled into an Outex test suite to allow easy replication of the experiments.

3 LBP and its extensions

In this chapter, the LBP operator is first derived from a general definition of texture in a local neighborhood. In Section 3.2, it is explained how the LBP operator can be seen as a unifying approach to structural and statistical texture analysis. The non-parametric classification principle used in conjunction with the operator is explained in Section 3.3.

A number of extensions to the LBP texture operator are also presented. In Section 3.4, a rotation invariant version is derived, and the number of rotation invariant codes is reduced by introducing the concept of uniformity. A discussion about the complementing roles of contrast and texture patterns is presented in Section 3.5. In Section 3.6, three ways of extending the LBP to multiple resolutions are presented. Section 3.8.4 combines texture and color measures, and provides experimental evidence on the complementary roles of color and texture. Finally, in Section 3.8, a number of empirical studies are presented to validate the proposed extensions.

3.1 Derivation

The derivation of the LBP follows that represented in Paper IV. Due to the lack of a universally accepted definition of texture, the derivation must start with a custom one. Let us therefore define texture T in a local neighborhood of a gray-scale image as the joint distribution of the gray levels of $P + 1$ ($P > 0$) image pixels:

$$T = t(g_c, g_0, \ldots, g_{P-1}), \tag{3.1}$$

where g_c corresponds to the gray value of the center pixel of a local neighborhood. $g_p (p = 0, \ldots, P - 1)$ correspond to the gray values of P equally spaced pixels on a circle of radius R ($R > 0$) that form a circularly symmetric set of neighbors. This set of $P + 1$ pixels is later denoted by G_P. In a digital image domain, the coordinates of the neighbors g_p are given by $(x_c + R\cos(2\pi p/P), y_c - R\sin(2\pi p/P))$,

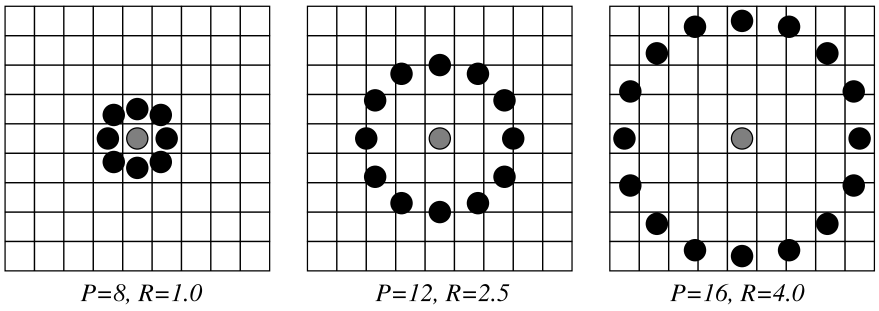

Figure 3.1. Circularly symmetric neighbor sets. Samples that do not exactly match the pixel grid are obtained via interpolation.

where (x_c, y_c) are the coordinates of the center pixel. Figure 3.1 illustrates three circularly symmetric neighbor sets for different values of P and R. The values of neighbors that do not fall exactly on pixels are estimated by bilinear interpolation. Since correlation between pixels decreases with distance, much of the textural information in an image can be obtained from local neighborhoods.

If the value of the center pixel is subtracted from the values of the neighbors, the local texture can be represented — without losing information — as a joint distribution of the value of the center pixel and the differences:

$$T = t(g_c, g_0 - g_c, \ldots, g_{P-1} - g_c). \tag{3.2}$$

Assuming that the differences are independent of g_c, the distribution can be factorized:

$$T \approx t(g_c)t(g_0 - g_c, \ldots, g_{P-1} - g_c). \tag{3.3}$$

In practice, the independence assumption may not always hold. Due to the limited nature of the values in digital images, very high or very low values of g_c will obviously narrow down the range of possible differences. However, accepting the possible small loss of information allows one to achieve invariance with respect to shifts in the gray scale.

Since $t(g_c)$ describes the overall luminance of an image, which is unrelated to local image texture, it does not provide useful information for texture analysis. Therefore, much of the information about the textural characteristics in the original joint distribution (Eq. 3.2) is preserved in the joint difference distribution (Ojala *et al.* 2001):

$$T \approx t(g_0 - g_c, \ldots, g_{P-1} - g_c). \tag{3.4}$$

The $P-$dimensional difference distribution records the occurrences of different texture patterns in the neighborhood of each pixel. For constant or slowly varying regions, the differences cluster near zero. On a spot, all differences are relatively large. On an edge, differences in some directions are larger than the others.

Although invariant against gray scale shifts, the differences are affected by scaling. To achieve invariance with respect to any monotonic transformation of the gray scale, only the signs of the differences are considered:

$$T \approx t(s(g_0 - g_c), \dots, s(g_{P-1} - g_c)), \tag{3.5}$$

where

$$s(x) = \begin{cases} 1 & x \geq 0 \\ 0 & x < 0 \,. \end{cases} \tag{3.6}$$

Now, a binomial weight 2^p is assigned to each sign $s(g_p - g_c)$, transforming the differences in a neighborhood into a unique LBP code. The code characterizes the local image texture around (x_c, y_c):

$$\mathrm{LBP}_{P,R}(x_c, y_c) = \sum_{p=0}^{P-1} s(g_p - g_c) 2^p. \tag{3.7}$$

In practice, Eq. 3.7 means that the signs of the differences in a neighborhood are interpreted as a P-bit binary number, resulting in 2^P distinct values for the LBP code. The local gray-scale distribution, i.e. texture, can thus be approximately described with a 2^P-bin discrete distribution of LBP codes:

$$T \approx t(\mathrm{LBP}_{P,R}(x_c, y_c)). \tag{3.8}$$

Let us assume we are given an $N \times M$ image sample ($x_c \in \{0, \dots, N-1\}, y_c \in \{0, \dots, M-1\}$). In calculating the $\mathrm{LBP}_{P,R}$ distribution (feature vector) for this image, the central part is only considered because a sufficiently large neighborhood cannot be used on the borders. The LBP code is calculated for each pixel in the cropped portion of the image, and the distribution of the codes is used as a feature vector, denoted by S:

$$\begin{aligned} S = \; & t(LBP_{P,R}(x, y)), \\ & x \in \{\lceil R \rceil, \dots, N-1-\lceil R \rceil\}, y \in \{\lceil R \rceil, \dots, M-1-\lceil R \rceil\}. \end{aligned} \tag{3.9}$$

The original LBP (Section 2.5.1) is very similar to $\mathrm{LBP}_{8,1}$, with two differences. First, the neighborhood in the general definition is indexed circularly, making it easier to derive rotation invariant texture descriptors. Second, the diagonal pixels in the 3×3 neighborhood are interpolated in $\mathrm{LBP}_{8,1}$.

3.2 Bringing stochastic and structural approaches together

The stochastic formulation of texture is based on a model in which a texture is viewed as a sample of a two-dimensional stochastic process describable by its statistical parameters (Faugeras & Pratt 1980). Usually, as in MRF models, the interactions between pixels are assumed to be mainly of a local type (Woods 1972).

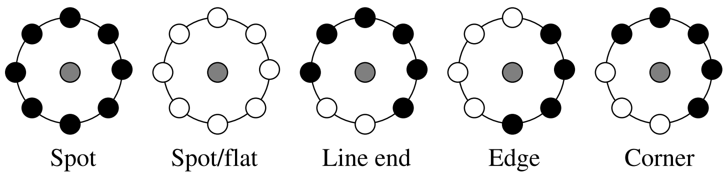

Figure 3.2. Different texture primitives detected by the LBP.

Cross & Jain (1983) write: "The brightness level at a point in an image is highly dependent on the brightness levels of neighboring points unless the image is simply random noise". Jain & Karu (1996) note that texture is characterized not only by the gray levels of pixels, but also by local gray value "patterns".

In the structural approach, the spatial structure of a texture is emphasized. According to Faugeras & Pratt (1980), the deterministic approach considers texture a basic local pattern that is periodically or quasi-periodically repeated over a region. Cross & Jain (1983) assume these primitives to be of some deterministic shape with varying sizes. In their definition of the structural approach, placement rules specify the positions of the primitives with respect to each other, and on the image.

The problem with two distinct approaches has been recognized, most recently by Sánchez-Yáñez *et al.* (2003): "Any texture contains both regular and statistical characteristics. In practice one can find textures between two extremes, completely periodic and completely random ones. That is why it is difficult to classify texture by one single method." That is also why unified models, in which the deterministic and non-deterministic parts of a texture field are separated, have been proposed, even though Tomita & Tsuji (1990, p. 100) doubt: "There is no unique way to analyze every texture." Francos *et al.* (1993) use a Wold-like decomposition to accomplish this task. This method does, however, have a number of shortcomings. First, Wold-like decomposition requires textures to be realizations of regular homogeneous random fields. Second, the decomposition requires the extraction of a harmonic component which is prominent only in highly regular textures, and badly affected by 3-D distortions (Mäenpää 1999). Third, the method does not actually unify the stochastic and structural approaches, as the concept of a texton is neglected. Rather, it provides a means of analyzing the stochastic and deterministic parts of a texture separately.

The LBP method can be regarded as a truly unifying approach. Instead of trying to explain texture formation on a pixel level, local patterns are formed. Each pixel is labeled with the code of the texture primitive that best matches the local neighborhood. Thus, each LBP code can be regarded as a micro-texton. Local primitives detected by the LBP include spots, flat areas, edges, edge ends, curves and so on. Some examples are shown in Figure 3.2 with the $\mathrm{LBP}_{8,R}$ operator. In the figure, ones are represented as white circles, and zeros are black.

The combination of the structural and stochastic approaches stems from the fact that the distribution of micro-textons can be seen as statistical placement rules. The LBP distribution therefore has both of the properties of a structural analysis method: texture primitives and placement rules. On the other hand, the distribution is just a statistic of a non-linearly filtered image, clearly making the method a statistical one. For these reasons, it is to be assumed that the LBP distribution can be successfully used in recognizing a wide variety of texture types, to which statistical and structural methods have conventionally been applied separately.

3.3 Non-parametric classification principle

In classification, the dissimilarity between a sample and a model LBP distribution is measured with a non-parametric statistical test. This approach has the advantage that no assumptions about the feature distributions need to be made. Originally, the statistical test chosen for this purpose was the cross-entropy principle introduced by Kullback (1968) (Ojala *et al.* 1996). Later, Sokal & Rohlf (1969) have called this measure the G statistic:

$$G(S,M) = 2\sum_{b=1}^{B} S_b \log \frac{S_b}{M_b} = 2\sum_{b=1}^{B} \left[S_b \log S_b - S_b \log M_b \right], \tag{3.10}$$

where S and M denote (discrete) sample and model distributions, respectively. S_b and M_b correspond to the probability of bin b in the sample and model distributions. B is the number of bins in the distributions.

For classification purposes, this measure can be simplified. First, the constant scaling factor 2 has no effect on the classification result. Furthermore, the term $\sum_{b=1}^{B} [S_b \log S_b]$ is constant for a given S, rendering it useless too. Thus, the G statistic can be used in classification in a modified form:

$$L(S,M) = -\sum_{b=1}^{B} S_b \log M_b. \tag{3.11}$$

The G statistic is certainly not the only possible alternative. For a thorough survey of different dissimilarity measures, the work of Puzicha *et al.* (1999) should be consulted. The results of their work may not, however, be directly applicable to the LBP due to the characteristics of the LBP distribution.

Model textures can be treated as random processes whose properties are captured by their LBP distributions. In a simple classification setting, each class is represented with a single model distribution M^i. Similarly, an unidentified sample texture can be described by the distribution S. L is a pseudo-metric that measures the likelihood that the sample S is from class i. The most likely class C of an unknown sample can thus be described by a simple nearest-neighbor rule:

$$C = \underset{i}{\text{arg min}}\ L(S, M^i) \tag{3.12}$$

Apart from a log-likelihood statistic, L can also be seen as a dissimilarity measure. Therefore, it can be used in conjunction with many classifiers, like the k-NN classifier or the self-organizing map (SOM).

3.4 Rotation invariance

Due to the circular sampling of neighborhoods, it is fairly straightforward to make LBP codes invariant with respect to rotation of the image domain. "Rotation invariance" here does not however account for textural differences caused by changes in the relative positions of a light source and the target object. It also does not consider artifacts that are caused by digitizing effects. Each pixel is considered a rotation center, which seems to be the convention in deriving rotation invariant operators.

With the assumptions stated above, the rotation invariant LBP can be derived as follows. When an image is rotated, the gray values g_p in a circular neighbor set move along the perimeter of a circle centered at g_c. Since the neighborhood is always indexed counter-clockwise, starting in the direction of the positive x axis, the rotation of the image naturally results in a different $\mathrm{LBP}_{P,R}$ value. This does not, however, apply to patterns comprising of only zeros or ones which remain constant at all rotation angles. To remove the effect of rotation, each LBP code must be rotated back to a reference position, effectively making all rotated versions of a binary code the same. This transformation can be defined as follows:

$$\mathrm{LBP}_{P,R}^{ri} = \min\{\mathrm{ROR}(\mathrm{LBP}_{P,R}, i) \mid i = 0, 1, \ldots, P-1\} \tag{3.13}$$

where the superscript ri stands for "rotation invariant". The function $\mathrm{ROR}(x, i)$ circularly shifts the P-bit binary number x i times to the right ($|i| < P$). That is, given a binary number x:

$$x = \sum_{k=0}^{P-1} 2^k a_k, \quad a_k \in \{0, 1\}, \tag{3.14}$$

the ROR operation is defined as:

$$\mathrm{ROR}(x, i) = \begin{cases} \sum_{k=i}^{P-1} 2^{k-i} a_k \ + \ \sum_{k=0}^{i-1} 2^{P-i+k} a_k & i > 0 \\ x & i = 0 \\ \mathrm{ROR}(x, P+i) & i < 0 \end{cases} \tag{3.15}$$

In short, the rotation invariant code is produced by circularly rotating the original code until its minimum value is attained. The LBPROT operator introduced by Pietikäinen *et al.* (2000) is equivalent to $\mathrm{LBP}_{8,1}^{ri}$. Figure 3.3 illustrates six rotation invariant codes in the top row. Below these, examples of rotated neighborhoods that result in the same rotation invariant code are shown. The first two neighborhoods are immutable by rotation, and there are consequently no other neighborhoods that could produce the same codes. The third one is special in that it has only two different rotated versions. The other three neighborhoods have

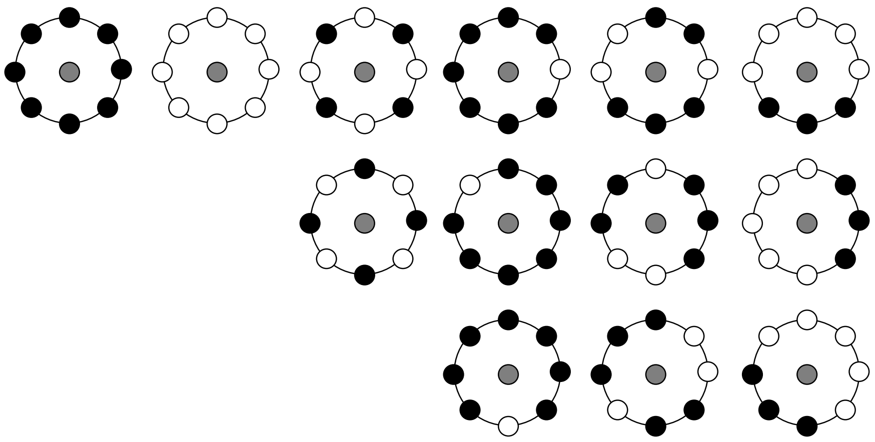

Figure 3.3. Neighborhoods rotated to their minimum value (top row) and neighborhoods that produce the same rotation invariant LBP codes.

a total of seven different rotated versions, two of which are shown as examples. In total, there are 36 different 8-bit rotation invariant codes. Therefore, $\mathrm{LBP}_{8,R}^{ri}$ produces 36-bin histograms.

Determining the number of rotation invariant codes given the number of bits is a non-trivial task. So far, an algorithm performing this operation with a complexity of $O(N^2)$ (N is the number of bits) has been developed.

In practice, images are seldom rotated in 45° steps. This makes the crude quantization of the angular space in the $\mathrm{LBP}_{8,R}$ operator quite inefficient. Furthermore, the occurrence frequencies of the 36 different patterns vary a lot. Some of them are encountered only rarely, making them statistically unstable.

The concept of "uniform" patterns was introduced in Paper II. It was observed that certain patterns seem to be fundamental properties of texture, providing the vast majority of patterns, sometimes over 90%. This proposion was further confirmed with a large amount of data in Paper IV. These patterns are called "uniform" because they have one thing in common: at most two one-to-zero or zero-to-one transitions in the circular binary code. The LBP codes shown in Figure 3.2 are all uniform. Examples of non-uniform codes can be seen in Figure 3.3, in the third and fifth columns.

To formally define the "uniformity" of a neighborhood G, a uniformity measure U is needed:

$$U(G_P) = |s(g_{P-1} - g_c) - s(g_0 - g_c)| + \sum_{p=1}^{P-1} |s(g_p - g_c) - s(g_{p-1} - g_c)|. \quad (3.16)$$

Patterns with a U value of less than or equal to two are designated as "uniform". For a P-bit binary number, the U value can be calculated efficiently as follows:

$$U(x) = \sum_{p=0}^{P-1} F_1(x \text{ xor } \mathrm{ROR}(x,1), p), \tag{3.17}$$

where b is a binary number. The function $F_b(x,i)$ extracts b circularly successive bits from a binary number x:

$$F_b(x,i) = \mathrm{ROR}(x,i) \text{ and } (2^b - 1), \tag{3.18}$$

where i is the index of the least significant bit of the bit sequence. The operators "and" and "xor" denote bitwise logical operations. Here, only F_1 is needed, but the general definition is later utilized in Section 3.6.3.1.

The total number of patterns with $U(G_P) \leq 2$ is $P(P-1)+2$. When "uniform" codes are rotated to their minimum values, the total number of patterns becomes $P+1$. The rotation invariant uniform ($riu2$) pattern code for any "uniform" pattern is calculated by simply counting ones in the binary number. All other patterns are labeled "miscellaneous" and collapsed into one value:

$$\mathrm{LBP}_{P,R}^{riu2} = \begin{cases} \sum_{p=0}^{P-1} s(g_p - g_c) & U(G_P) \leq 2 \\ P+1 & \text{otherwise .} \end{cases} \tag{3.19}$$

In practice, $\mathrm{LBP}_{P,R}^{riu2}$ is best implemented by creating a look-up table that converts the "basic" LBP codes into their LBP^{riu2} correspondents.

A straightforward fix to the problem of the crude 45° quantization of $\mathrm{LBP}_{8,R}$ is to employ a larger number of samples in the circular neighborhood. By increasing P a finer angular resolution with $360°/P$ quantization steps is achieved. The value of P cannot however be increased carelessly. First, the number of possible pattern codes that need to be mapped to their rotation invariant counterparts is 2^P. Obviously, the size of the look-up table will put a practical upper bound for the range of usable P values. It would be possible to calculate the mapping on-line, without a look-up table, but then the computational simplicity of the method would suffer significantly. Another issue that limits the range of usable values for P is the nature of digital images. For example, there are only nine pixels in a 3×3 neighborhood. If a P greater than eight is used in conjunction with $R = 1$, the samples contain a lot of redundant information.

3.5 Contrast and texture patterns

As discussed in Section 1.1, contrast is a property of texture usually regarded as a very important cue for our vision system, but the LBP operator by itself totally ignores the magnitude of gray level differences. In many applications, especially in industrial visual inspection, illumination can be accurately controlled. In such a situation, a purely gray-scale invariant texture operator may waste useful information, and adding gray-scale dependent information may enhance the accuracy of the method. Furthermore, in applications like image segmentation, gradual changes in illumination may not require the use of a gray-scale invariant method (Ojala & Pietikäinen 1999b).

In a more general view, texture is distinguished not only by texture patterns but also the strength of the patterns. Texture can thus be regarded as a two-dimensional phenomenon characterized by two orthogonal properties: spatial structure (patterns) and contrast (the strength of the patterns). Pattern information is independent of the gray scale, whereas contrast is not. On the other hand, contrast is not affected by rotation, but patterns are, by default. These two measures supplement each other in a very useful way. The LBP operator was originally designed just for this purpose: to complement a gray-scale dependent measure of the "amount" of texture. Ojala *et al.* (1996) used the joint distribution of LBP codes and a local contrast measure (LBP/C) as a texture descriptor.

Rotation invariant local contrast can be measured in a circularly symmetric neighbor set just like the LBP:

$$\mathrm{VAR}_{P,R} = \frac{1}{P}\sum_{p=0}^{P-1}(g_p - \mu)^2 \text{ , where } \quad \mu = \frac{1}{P}\sum_{p=0}^{P-1} g_p \ . \qquad (3.20)$$

$\mathrm{VAR}_{P,R}$ is, by definition, invariant against shifts in the gray scale. Since contrast is measured locally, the measure can resist even intra-image illumination variation as long as the absolute gray value differences are not badly affected. A rotation invariant description of texture in terms of texture patterns and their strength is obtained with the joint distribution of LBP and local variance, denoted as $\mathrm{LBP}^{riu2}_{P_1,R_1}/\mathrm{VAR}_{P_2,R_2}$. In the experiments presented in this thesis, $P_1 = P_2$ and $R_1 = R_2$, although nothing prevents one from using different values. The main reason for this choice is that the semantics for the general case are quite hard to understand. For example, $\mathrm{LBP}^{riu2}_{8,1}/\mathrm{VAR}_{16,2}$ measures the joint distribution of local 8-bit patterns with radius 1, and the local contrast of 16 neighbors with radius 2. It is difficult to give a meaningful interpretation to such a distribution.

3.6 Multi-resolution LBP

The most prominent limitation of the LBP operator is its small spatial support area. Features calculated in a local 3×3 neighborhood cannot capture large-scale structures that may be the dominant features of some textures. However, adjacent LBP codes are not totally independent of each other. Figure 3.4 displays three adjacent four-bit LBP codes. Assuming that the first bit in the leftmost code is zero, the third bit in the code to the right of it must be one. Similarly, the first in the center code and the third bit of the rightmost one must be either different or both equal to one. The right half of the figure shows an impossible combination of the codes. Each LBP code thus limits the set of possible codes adjacent to it, making the "effective area" of a single code actually slightly larger than 3×3 pixels. Nevertheless, the operator is not very robust against local changes in the texture, caused, for example, by varying viewpoints or illumination directions. An operator with a larger spatial support area is therefore needed.

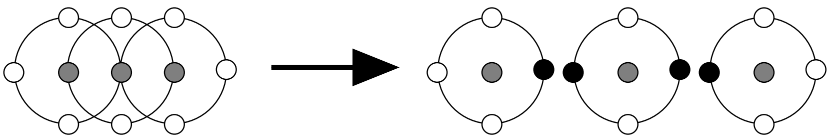

Figure 3.4. Three adjacent LBP$_{4,R}$ neighborhoods and an impossible combination of codes. A black disk means the gray level of a sample is lower than that of the center.

3.6.1 Combining operators

A straightforward way of enlarging the spatial support area is to combine the information provided by N LBP operators with varying P and R values. This way, each pixel in an image gets N different LBP codes. The most accurate information would be obtained by using the joint distribution of these codes. However, such a distribution would be overwhelmingly sparse with any reasonable image size. For example, the joint distribution of LBP$_{8,1}$, LBP$^{u2}_{16,3}$, and LBP$^{u2}_{24,5}$ would contain $256 \times 243 \times 555 \approx 3.5 \times 10^7$ bins. Therefore, only the marginal distributions of the different operators are considered even though the statistical independence of the outputs of the different LBP operators at a pixel cannot be warranted. The aggregate dissimilarity between a sample and a model can be calculated as a sum of the dissimilarities between the marginal distributions:

$$L_N = -\sum_{n=1}^{N} L(S^n, M^n), \tag{3.21}$$

where S^n and M^n correspond to the sample and model distributions extracted by the nth operator.

3.6.2 Combining with filtering

From a signal processing point of view, the sparse sampling exploited by LBP operators with large neighborhood radii may not result in an adequate representation of the two-dimensional image signal. Aliasing effects are an obvious problem. So might be noise sensitivity as sampling is made at single pixel positions, without low-pass filtering. One might argue that collecting information from a larger area would thus make the operator more robust. From the statistical point of view, even sparse sampling is however acceptable provided that the number of samples is large enough. The sparse sampling approach is also commonly used with GLC methods.

The LBP operator can be combined with multi-scale filtering in a straightforward way. Using Gaussian low-pass filters, each sample in the neighborhood can be made to collect intensity information from an area larger than the original single pixel. The filters and sampling positions are designed to cover the neighborhood as well as possible while minimizing the amount of redundant information. As a consequence, the radii of the LBP operators used in the multi-resolution version grow exponentially.

3.6.2.1 Gaussian filter bank design

Figure 3.5 shows a neighborhood quantized angularly to eight sectors. Without filtering, an LBP code would be constructed by sampling the neighborhood at the centers of the solid circles. With large radii, the distance between samples is large, presumably making the codes unreliable. With a low-pass filter, the intensity information for a sample is collected from a larger area, indicated by the solid circles. The outer radius of this "effective area" with respect to the center of the neighborhood is given by

$$r_n = r_{n-1}\left(\frac{2}{1-\sin(\pi/P_n)} - 1\right),\ n \in \{2,\ldots,N\} \tag{3.22}$$

where N is the number of scales, and P_n is the number of neighborhood samples at scale n. Since low-pass filtering is useful only with radii larger than one for $P = 8$, r_1 is set to 1.5, which is the shortest distance between the center and the border of a 3×3 neighborhood.

The radii of the LBP operators are chosen so that the effective areas touch each other. Consequently, the radius for an LBP operator at scale n ($n \geq 2$) is halfway between r_n and r_{n-1}:

$$R_n = \frac{r_n + r_{n-1}}{2}. \tag{3.23}$$

These radii are illustrated with dotted circles in the figure.

The effective areas are realized with Gaussian low-pass filters designed so that 95% of their mass lies within the solid circles. This way, each pixel in a filtered image represents a weighted sum of the neighborhood. Each bit in an LBP code thus carries information about the whole effective area, not just a single pixel. The effective areas could be designed so that they filled the sampling sector more precisely by utilizing filters that are not circularly symmetrical. A different filter should then be used for each orientation. Circularly symmetrical filters have the advantage that only one filter is needed for each scale. The spatial width and height of the Gaussian filter at scale n are given by

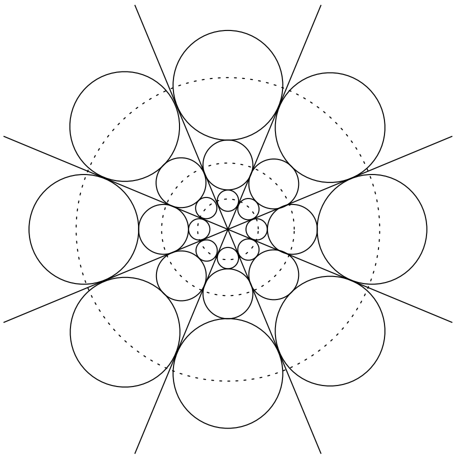
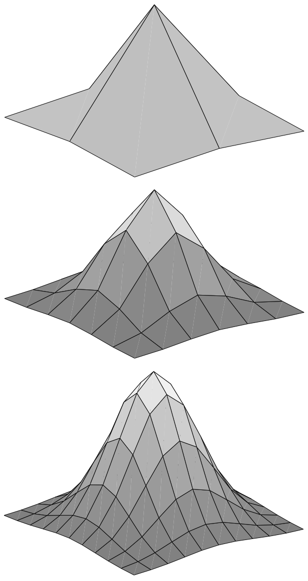

Figure 3.5. The effective areas of filtered pixel samples in an eight-bit multi-resolution LBP operator (left), and Gaussian low-pass filters for scales 2, 3, and 4 (right).

$$w_n = 2\left\lceil \frac{r_n - r_{n-1}}{2} \right\rceil + 1. \tag{3.24}$$

If a 2-D Gaussian, $f_G(x, y)$, is treated as a zero-mean distribution of two independent variables with equal standard deviations, its width can be conveniently described by the standard deviation σ ($\sigma > 0$):

$$f_G(x, y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}, \tag{3.25}$$

where x and y are the random variables, whose values are expressed in a Cartesian coordinate system. To integrate Eq. 3.25 within a circle, one needs to use polar coordinates:

$$f_G(r, \theta) = \frac{1}{2\pi\sigma^2} e^{-\frac{r^2}{2\sigma^2}}. \tag{3.26}$$

A two-dimensional analogy of the one-dimensional Gaussian integral (i.e. the "error function") can be constructed with a definite integral:

$$\begin{aligned} \mathrm{erf}_{\mathrm{2D}}(r) &= \frac{1}{2\pi\sigma^2} \int_0^{2\pi} \int_0^r e^{-\frac{t^2}{2\sigma^2}} t \; \mathrm{d}t \; \mathrm{d}\theta \\ &= 1 - e^{-\frac{r^2}{2\sigma^2}}, \quad r \geq 0. \end{aligned} \tag{3.27}$$

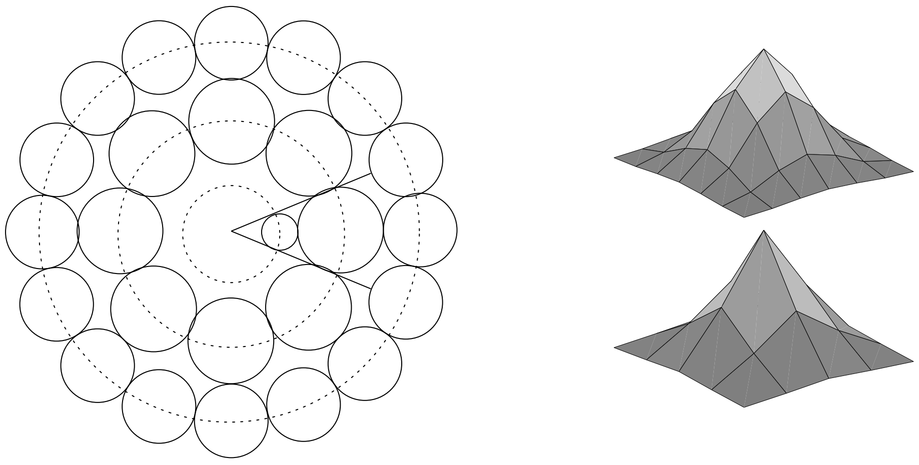
Figure 3.6. The effective areas of neighborhood samples in a multi-resolution operator with P values 8 and 16 (left), and the corresponding Gaussian low-pass filters for scales 3 and 4 (right).

where r is the integration radius, and t is a dummy integration variable. In terms of statistics, Eq. 3.27 gives the probability that the values of independent random variables x and y in Eq. 3.25 are within a circle of radius r, i.e. the mass of the distribution within this radius.

The inverse of Eq. 3.27 is given by

$$\mathrm{erf_{2D}}^{-1}(p) = \sigma\sqrt{-2\ln(1-p)}, \quad p \in [0,1], \tag{3.28}$$

where p is a probability. The error function for the "standard normal" distribution is obtained by setting σ to unity. Given these definitions, the standard deviation of the Gaussian filter at scale n can be calculated from

$$\sigma_n = \frac{r_n - r_{n-1}}{2 \times \mathrm{erf_{2D}}^{-1}(0.95)}. \tag{3.29}$$

Due to the clipping of the Gaussian function, quantization effects, and the rectangular shape of the filters, the sum of the filter coefficients varies. For this reason, the sum is normalized to unity. The result is, as shown in the rightmost part of Figure 3.5, a bank of unit-sum Gaussian filters with an exponentially growing size. The effective areas of the neighborhood samples are always positioned so that they touch each other. In Figure 3.6, the smaller neighborhood is used by an 8-bit LBP operator at exponential scale 3 ($R = 5.4$), and the larger one with a 16-bit operator at scale 4 ($R = 9.3$).

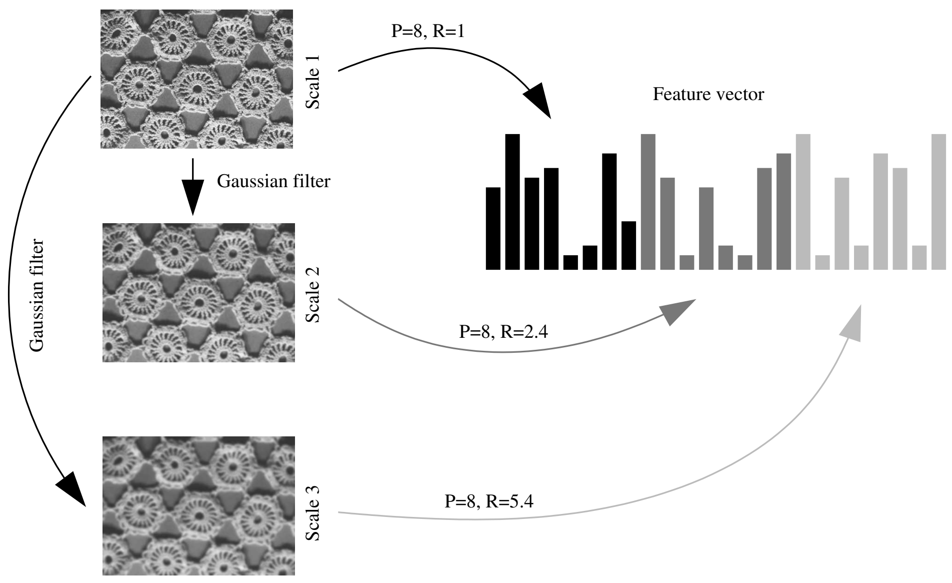

Figure 3.7. Building a multi-scale LBP with low-pass filtering.

3.6.2.2 Building a multi-resolution filtered LBP

The procedure used in building a multi-resolution filtered LBP is very similar to that described in Section 3.6.1. The only difference is in that the neighborhood samples with radii greater than one are obtained via low-pass filtering. In addition, neighborhood radii are not chosen arbitrarily. They are chosen so that the effective areas touch each other. Figure 3.7 shows how a three-scale version of the $\mathrm{LBP}_{8,R}$ operator is constructed. The $\mathrm{LBP}_{8,1}$ calculated from the original image is used as the basis of the feature vector. Two low-pass filtered images are created, and the LBP operators with the neighborhood radii shown in the figure are calculated from these images. The final feature vector is created by concatenating the results from the different resolutions. To distinguish between the low-pass filtered and the "original" versions of the LBP, the former is later denoted by $\mathrm{LBPF}_{P,R}$.

The length of the final feature vector soon becomes a limiting factor in using a large number of different scales. Considering only "uniform" patterns helps to some extent, but not enough. Memory consumption and computational burden in classification still make the use of many different scales impractical. Another possibility is to sum up the histograms instead of concatenating them, which fixes the length of the feature vector. The drawback is that each scale must use the same number of neighborhood samples. Important information may also be lost. The ability to encode a large neighborhood compactly while preserving important information is a crucial issue in building a multi-scale extension of the LBP operator. Cellular automata, detailed in the next section, may provide a solution to this problem.

3.6.3 Cellular automata

Cellular automata can be generally described as discrete dynamic systems completely defined by a set of rules in a local neighborhood. The state of a system is represented as a regular grid, on which the rules are applied to produce a new state. An interesting property of cellular automata is that very simple rules can result in very complex behavior.

The concept of a cellular automaton was first introduced by John von Neumann in the 1940's. In his model, the state of a system was presented with a two-dimensional grid of ones and zeros — just like a digital binary image. The value of a slot in the state grid at time instant $t+1$ was determined by the values of its nearest neighbors at time instant t. Wolfram (1983) later considered a one-dimensional case in which the value of a slot at time instant $t+1$ is dependent only on the value of the slot itself and its two neighbors at time instant t. If the one-dimensional state is binary, there is a total of $2^3 = 8$ different neighborhoods. The rules determine whether a certain neighborhood produces one or zero at the next iteration. There is therefore a total number of $2^8 = 256$ different rules.

Figure 3.8 shows one cellular automaton rule. Diverging from the convention adopted by Wolfram (2002), ones are represented with white and zeros with black squares. The rule in the figure is interpreted as follows. Whenever three adjacent slots are ones at time instant t, the value of the center slot at time instant $t+1$ remains one (bit 7). Similarly, if the two leftmost slots are ones and the rightmost slot is zero (bit 6), the value of the center becomes zero. The rest should be quite obvious. Evidently, the rule can be encoded with eight bits, and this number uniquely determines the cellular automaton. The automaton number of this particular automaton is $10001001_2 = 137_{10}$.

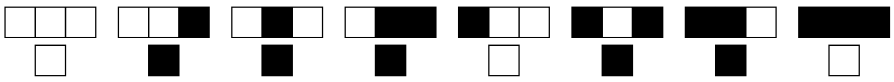

Figure 3.8. A one-dimensional cellular automaton.

Figure 3.9 illustrates how the automata work. The topmost row contains an arbitrary input signal. At each iteration, the rule of Figure 3.8 is applied, producing a new row. Each row is regarded as a circular signal so that the left and right borders are actually adjacent to each other. The outcome can be conveniently presented as a two-dimensional pattern.

A few people have used cellular automata in image processing, especially with parallel computation (Dyer & Rosenfeld 1981). Hernandez & Herrann (1996)

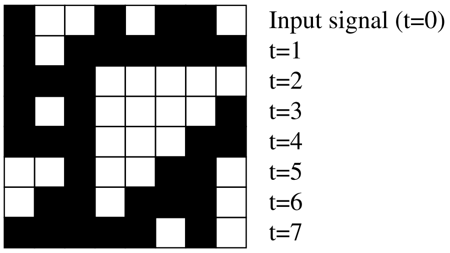

Figure 3.9. Seven iterations of a cellular automaton starting with an arbitrary input signal.

present cellular automata algorithms for sharpening and smoothing images. Ikenaga & Ogura (2000) introduce an algorithm for performing complex binary morphology operations in real time with cellular automata. Khan *et al.* (1997) have developed a 2-D cellular automata machine (CAM) that is applied to the zooming and thinning of image elements. Daemi *et al.* (1994) propose genetically evolving cellular automata as a solution for many image processing tasks, and successfully use them in detecting edges. It does, however, seem that the use of cellular automata in extracting features for image analysis has been largely overlooked.

3.6.3.1 Cellular automata as texture features

The relation between multi-scale LBP and cellular automata may not be quite obvious. It becomes so if we think of the thresholded circular neighborhoods as one-dimensional circular binary signals. In Figure 3.10, two $\mathrm{LBP}_{8,R}$ codes are dressed to form the two topmost rows of a two-dimensional pattern. The notation of "time" in the evolution of the cellular automaton pattern is thus replaced by the radius of the circular neighborhood. Since the rows are treated circularly, the breakpoint (dashed line) plays no role. This property has an important consequence: the cellular automata are always invariant with respect to rotation, irrespective of the LBP version used as the input signal. Any number of LBP scales can be added to the pattern. The problem of encoding a neighborhood now becomes one of finding a rule that could have produced the pattern, starting from the input signal at the topmost row. Arbitrarily large neighborhoods can thus be encoded with the input signal (P bits) and a cellular automaton rule (eight bits). The joint distribution of the input signal and the cellular automaton rule code can be used as a texture descriptor.

Given a complete cellular automaton pattern, it is easy to deduce the rule which produced it. All one needs to do is to find an occurrence of each of the eight possible neighborhoods, and see what it produced (Wolfram 2002, p. 1089). In

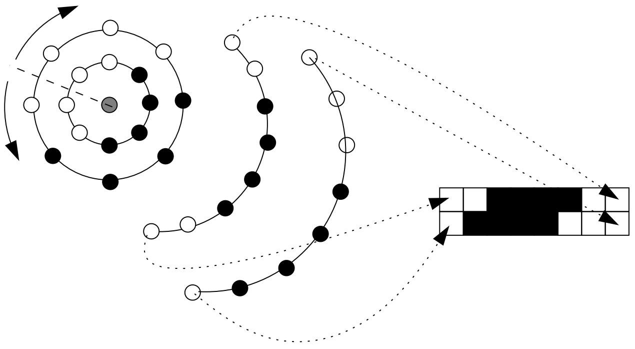

Figure 3.10. Turning a multi-resolution LBP into a two-dimensional pattern.

general the pattern does not, however, follow exactly any rule, and one needs to find the most likely one given incomplete and contradictory information. Wolfram (2002, p. 590) writes: "In general the problem of working out what model is most appropriate for any given set of data is an extremely difficult one." In reproducing large patterns inversion is an obvious problem because the behavior of a cellular automaton is in some sense chaotic. Small changes in the rules produce large changes in the outcome after a few iterations. Nevertheless, if one can deduce the most likely cellular automaton rule for a binarized neighborhood, it no doubt contains useful information about the structure of the neighborhood.

A straightforward way of finding the most appropriate cellular automaton rule is to try out every possibility, and compare the result to the observed pattern. The Hamming distance can be used as a goodness-of-fit measure. Ideally, the observed and reproduced patterns match, resulting in zero distance. At the other extreme, the Hamming distance equals the number of slots in the pattern.

Searching through all possible rules is a time consuming operation. With one-dimensional automata and three-neighbor rules the problem is however solvable. For more complex cellular automata, evolutionary mechanisms have been traditionally used (See e.g. Bossomaier *et al.* 1999, Billings & Yang 2003a). The method of Billings & Yang (2003b) identifies both the rules and the neighborhood with a modified orthogonal least squares algorithm.

Probabilistic cellular automata can be used to avoid exhaustive searching. In a probabilistic automaton, the decision on whether the value of a slot at time instant $t+1$ will be one or zero is based on the neighborhood at time instant t and two probabilities assigned to the possible outcomes. If both probabilities are 0.5, then a certain neighborhood is equally likely to produce either one or zero. The "ordinary" cellular automaton is a special case of a probabilistic one in which the probabilities are zeros and ones.

Figure 3.11 displays a random neighborhood for which the most likely probabilistic cellular automaton rule is to be found. It is easy to see that no consistent rule can be found. For example, the neighborhood "111" occurs 13 times. The

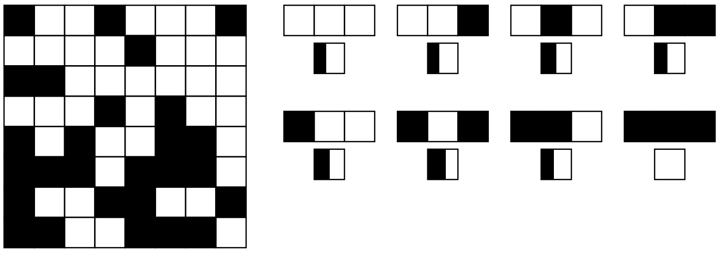

Figure 3.11. A random pattern and the probabilistic cellular automaton that most likely produced it.

outcome is “0” five times, and “1” eight times. The best one can guess is that “111” produces “0” with a probability of 5/13. This is illustrated with a partially black square in the rightmost part of the figure. The problem is that each pattern must now be encoded with eight real numbers instead of eight bits. These vectors must be clustered somehow to reduce the dimensionality of the resulting texture descriptor.

An easy way of converting probabilistic rules to conventional ones is to binarize the probabilities. This way, the larger probability always “wins”, and the rule can again be represented with eight bits.

Having defined the concepts, we can now proceed with an analytical definition of the cellular automaton rule for a pixel. For each pixel, the pattern from which the cellular automaton rule is derived, is formed by S concentric LBP codes, denoted by $\mathrm{LBP}_{P,n}$, $n \in \{1, \dots, S\}$. Each bit in the $\mathrm{LBP}_{P,n}$ codes with $n \in \{2, \dots S\}$ is seen as a result of applying a cellular automaton rule r to its nearest neighbors in $\mathrm{LBP}_{P,n-1}$. Let us define a set R_b^h that contains each occurrence of neighborhood h with b as the output value on the next scale:

$$\begin{aligned} R_b^h = \;& \{(n,i),\ n \in \{1,\dots,S-1\},\ i \in \{0,\dots,P-1\} | \\ & F_3(\mathrm{LBP}_{P,n}, i) = h,\ F_1(\mathrm{LBP}_{P,n+1}, x) = b,\ i+1 \equiv x \pmod{P}\}, \end{aligned} \tag{3.30}$$

where (n, i) denote the scale and bit index of the center of a three-bit binary neighborhood. The functions F_3 and F_1 are defined in Eq. 3.18.

The most likely cellular automaton rule (r) is built by combining the most common outcomes for each of the eight different neighborhoods:

$$r = \sum_{h=0}^{7} s(|R_1^h| - |R_0^h|)2^h \tag{3.31}$$

In cases where $|R_1^h| = |R_0^h|$ neither outcome is more probable. The $s(x)$ function (Eq. 3.6) produces a unity, but zero can be used as well.

As a texture feature, the marginal distributions of $\mathrm{LBP}_{P,1}$ codes and cellular automata rules deduced from S consecutive scales is used, and denoted by

$\mathrm{LBPCA}_{P,S}$. The approach is similar to that used in building the “original” multi-resolution version. For each pixel, S LBP codes are calculated so that the radius of the local neighborhood grows from 1 to S. The automaton rule is deduced from these codes for each pixel, and these are treated just like the LBP codes. Their distribution is used as a multi-scale texture descriptor, and it is concatenated to the $\mathrm{LBP}_{P,1}$ feature vector.

The computational overhead of the $\mathrm{LBPCA}_{P,S}$ is over S times that of $\mathrm{LBP}_{P,R}$. Therefore, the operator may not be well suited for applications where fast feature extraction is needed. Its ability to describe large neighborhoods with a relatively short feature vector is, however, a very important feature in image retrieval, for example.

3.6.3.2 Statistical stability

The simple method of thresholding probabilistic cellular automaton rules has the drawback that with small patterns, not enough information can be collected to draw statistically sound decisions. Therefore, the number of scales in the multi-scale LBP operator should be large. Due to border effects, a large neighborhood means that a large texture must be used. Since the correlation between the center and a neighbor decreases with distance, very large neighborhoods may not provide much useful information. With small textures, only unreliable measurements can be made, which may render the joint distribution of LBP codes and cellular automaton rules both too sparse and too noisy.

To fight the imminent problems with statistical stability, the number of “accepted” cellular automaton rules can be reduced. Based on knowledge of the structure of digital images, one may argue that certain rules are more likely to occur than others. For example, due to the high correlation between neighboring pixels, it is likely that a sequence of three “ones” at scale 1 is likely to produce a “one” at scale 2. Based on this argument, one of the most likely cellular automata should be the one shown in Figure 3.12 in which a majority rule is used to judge the outcome. When the distance between scales grows, the resulting cellular automaton rules probably no longer adapt to the correlation between pixels. In a sense, the cellular automation rule can be seen as a measure of correlation between the LBP codes at different scales.

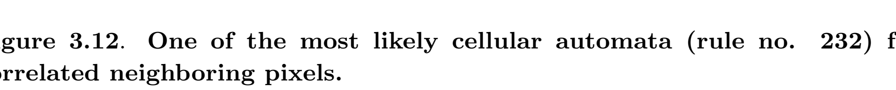

Figure 3.12. One of the most likely cellular automata (rule no. 232) for correlated neighboring pixels.

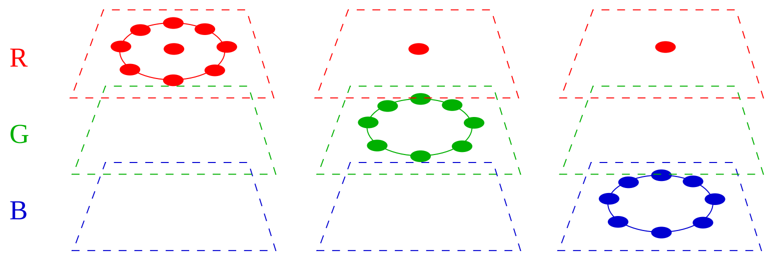

Figure 3.13. Creating opponent color LBP codes with a red center. The three planes illustrate color channels.

An easy way of purging unlikely cellular automata is to employ the strategy described in Section 3.4. With a large amount of natural textures, reliable statistics on the occurrences of different cellular automata can be derived. Then, the number of applicable automata can be reduced by discarding the least frequently occurring ones. An even more compact descriptor can be achieved by considering only the marginal distributions of the LBP codes and cellular automaton rules. It has emerged that the latter approach works much better.

Statistics calculated from the Outex textures (see Section 2.6) show that 32 of the 256 cellular automaton rules have a 94.6% share of all pixels. Thus, a 87.5% compression of the feature vector can be achieved by discarding only 5.4% of the information. Another advantage is that the purging of the most infrequently occurring codes is likely to decrease the statistical instability in their distribution. In the statistical analysis, rule 232 (Figure 3.12) ranked the fourth most common one. In the experiments reported in Section 3.8.3, all 256 rules are, however, used.

3.7 Opponent color LBP

The opponent color LBP (OCLBP) operator was developed as a joint color-texture operator for comparing gray-scale and color texture features in Paper VI. The mechanism with which the OCLBP feature is extracted is motivated by Jain & Healey (1998). Also the use of the term "opponent color" here follows the convention adopted by them: all pairs of color channels are called "opponent colors". Precisely, opponent colors are colors perceived as opposing pairs by humans: red-green and yellow-blue. The "generalization" of the term is however justified, because with computer vision there is no need to restrict oneself to the color processing model of the human eye.

In the opponent color LBP, the LBP operator is applied on each color channel separately. In addition, each pair of color channels is used in collecting opponent color patterns so that the center pixel for a neighborhood and the neighborhood itself are taken from different color channels. Figure 3.13 illustrates the three situations in which the center pixel is taken from the red channel. In total, three inter-channel LBP histograms and six intra-channel histograms are extracted and concatenated into a single distribution. This version was used in the experiments presented in Paper VI. Later, it turned out that three of these can be dropped without loss of accuracy. Since opposing pairs, like R–G and G–R, are highly redundant, either will suffice for analysis. Even then, the resulting texture descriptor is six times longer than the gray-scale version, which may make it unusable in some applications.

3.8 Experiments

3.8.1 Texture segmentation

The multi-scale extension of the LBP operator was first presented in Paper I. The main differences to the generalized multi-scale LBP were that only eight samples were allowed at each scale and that a square neighborhood was utilized. No interpolation was used for the diagonal neighbors.

To show the performance of the multi-scale approach, a comparative study was conducted. As a test bed, the supervised texture segmentation problems of Randen & Husoy (1999) were chosen. The images in the segmentation problems come from three sources: the Brodatz album, the VisTex database, and the MeasTex database. The segmentation problems contain twelve artificial mosaics generated from arbitrarily chosen textures. For each texture class in a test mosaic there is a 256×256 training image that is extracted from a different area in the source image. Global histogram equalization has been applied to the source images.

The original experiments were replicated as closely as possible to obtain comparable results. An independent sample of each texture class was used in training, and the segmentation itself was made on a pixel-by-pixel basis. A description of local texture was generated for each pixel by calculating an LBP (without interpolation) histogram on a circular area around it. The classification for a pixel was obtained with the nearest neighbor principle.

In their study, Randen & Husoy compared a number of analysis methods, including different filtering schemes, co-occurrence features and multi-resolution autoregressive models. The performance of the LBP was compared to the best result obtained with any of these methods with any parameter combination. Furthermore, the results reported for the LBP were obtained with a fixed set of parameters, although better performance could have been achieved, for example, by a varying disk size. In all but two of the twelve problems, $\text{LBP}_{8,1}$ was better than any of the methods evaluated by Randen & Husoy. Simple multi-resolution versions were even better with all but three problems. On average, $\text{LBP}_{8,1}$ obtained an accuracy of 87.7%. With $\text{LBP}_{8,1}+_{8,2}+_{8,3}$, an 89.9% score was achieved. The average of the best scores with any of the other methods was 81.6%.

The performance difference between the “basic” and multi-scale versions was most prominent with the last problem in which textures with a large scale difference

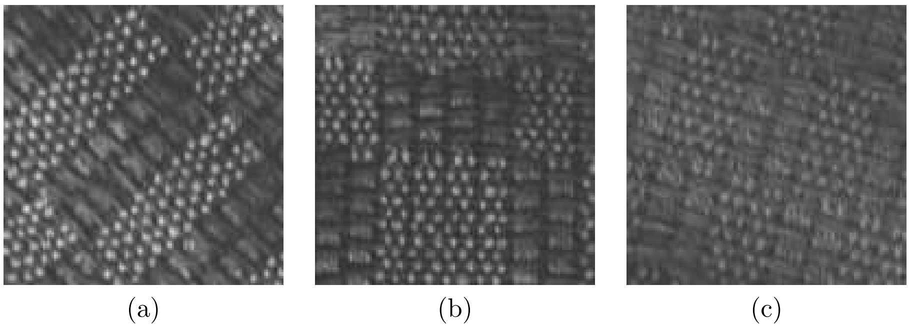

Figure 3.14. Three samples of canvas033 under three different illuminations and rotation angles: 2856K incandescent CIE A 45° (a), 2300K horizon sunlight 0° (b), and 4000K fluorescent TL84 15° (c).

were to be separated. The result indicated that a larger neighborhood could be successfully utilized in enhancing the performance of the LBP operator.

Much of the performance difference between the LBP and the other methods can be attributed to the gray-scale invariance of the LBP. Despite the histogram equalization, local illumination level changes in some source textures were clearly visible. Methods dependent on gray scale cannot cope with such situations. Nevertheless, the results were a clear indication of the performance of the operator.

3.8.2 Gray scale and rotation invariant texture analysis

To evaluate the performance of the LBP operator in rotation invariant texture analysis, two experiments were conducted as reported in Paper IV. First, the study of Porter & Canagarajah (1997) was replicated as closely as possible in order to obtain comparable results to other methods. The image data consisted of 16 textures digitized from Brodatz' album. The images had been digitally rotated to obtain material for the analysis of rotation invariance. In their original experiments, Porter & Canagarajah used training samples in many rotation angles, thereby simplifying the problem. For this reason, Paper IV also presents a more challenging setup in which the texture classifier is trained at only one particular rotation angle and tested with the others.

The second experiment involved 24 textures from the Outex database. The image set contained textures in three different illuminations and at nine different rotation angles, presenting a more realistic and challenging problem for gray scale and rotation invariant texture analysis. Two setups were considered: first, rotated versions of training samples were used for evaluation. In the second case, the rotated textures were imaged under two different light sources.

The changes caused to textures by varying the viewpoint or light source position are not taken into account in either of the image databases. With the Outex textures, however, there is slight variability in the positions of different light sources, causing changes in local shadowing. Even then, the effects are not very significant because the textures are illuminated at about 60°slant and because most of the textures are relatively flat. Textures samples from the same class may however look quite different under changing illumination, as illustrated in Figure 3.14. The goal of the experiments was to show that the LBP can be made invariant against rotation of the image domain and changes in the color and intensity of illumination. Invariance with respect to varying viewpoints and illumination directions is a clearly more complicated task that needs to be solved with different tools (Pietikäinen *et al.* 2003).

In the study by Porter & Canagarajah, three feature extraction schemes for rotation invariant texture classification were utilized: wavelet transform, circularly symmetric Gabor filters, and a GMRF model with a circularly symmetric neighbor set. The authors concluded that the wavelet-based approach was the best one in terms of accuracy, noise invariance, and computational complexity. Using a Mahalanobis distance classifier, they reported a classification accuracy of 95.8% as the best result. With a three-scale LBP/VAR distribution, all the texture samples could however be faultlessly classified. Even when only one rotation angle was used for training, the accuracy of the multi-scale LBP/VAR was 99.7%.

With the Outex textures, the multi-scale LBP/VAR achieved a 97.8% accuracy in the rotation invariance test. With illumination invariance added, the experiment proved to be much more difficult, as the accuracy dropped to 86.9%. Even though the results are far from perfect, they are clearly better than those obtained with the wavelet-based methods. With wavelets, the corresponding scores were 80.4% and 71.8%.

3.8.3 Multi-scale extensions

As a test bed for the low-pass filtered multi-scale LBP and the cellular automata, a classification experiment with 24 natural textures from the Outex database was arranged, as reported in Paper V. The same textures were utilized in Paper IV, but this time the test set included textures with two different spatial scales. This allows one to inspect the robustness of a method with respect to slight variations in the size of textural structures.

For the $\mathrm{LBP}_{8,1}$, the test suite was a hard problem, as only an accuracy of 92.7% could be achieved. The three-scale $\mathrm{LBP}^{u2}_{8,1} +^{u2}_{16,2} +^{u2}_{24,3}$ was clearly better with a 96.3% score. With a three-scale $\mathrm{LBP}^{u2}_{8,R}$ accompanied by Gaussian low-pass filtering, an accuracy of 99.0% was obtained. Filtering did indeed seem to help, as the result without it was consistently slightly worse. The increase in classification accuracy may however not be large enough compared to the increased computational burden. Furthermore, the size of the LBP distribution cannot be made shorter with filtering.

The LBPCA emerged as a very powerful measure of image texture. Consistent

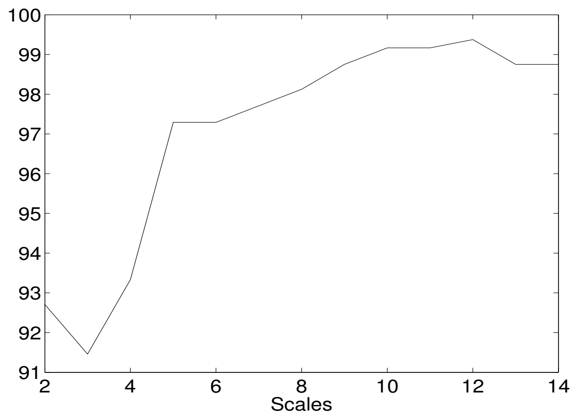

Figure 3.15. The accuracy of LBPCA$_{8,S}$ with growing S.

with the hypotheses, the accuracy of the method increased with the number of LBP scales in use, until a certain limit. Figure 3.15 shows how the accuracy of LBPCA$_{8,S}$ increases with the number of scales (S). With a small number of scales, the cellular automaton rules cannot be reliably deduced from the data. The increase in accuracy stops because the analyzed portion of the textures becomes smaller, and because the values of the neighborhood samples far away from the center and the center itself may no longer have much of a relationship. The best score, 99.4%, was obtained with a twelve-scale LBPCA$_{8,12}$. The same score was achieved by a version with only "uniform" LBP codes enabled. The result falls just 0.2 percentage units short of that of the three-scale LBPF. The length of the LBPCA feature vector is however only half of that of the LBPF.

3.8.4 Color texture discrimination

As discussed in Section 2.4, the problem of combining color and texture has been approached in a variety of different ways. The question now arises as to which of the proposed methods is the preferred one. In Paper VI, an answer to this question was sought through empirical evaluation. The performance of color indexing methods was compared to well-known texture and color-texture methods, and to combined color and texture methods in static and varying illumination conditions. In addition, a color version of the LBP operator was developed for joint color-texture analysis. It turned out that color-texture was outperformed by either color or texture in all experiments; joint color texture features dit not seem to provide much value for their increased complexity.

Many studies have concluded that adding color information to texture measures

Table 3.1. Correct classification percentages of different color, texture, and color-texture descriptors in three experiments.

Feature	VisTex	Outex 13	Outex 14	Mean
RGB cube (32^3)	99.5	94.3	32.1	75.3
$\text{LBP}^{u2}_{16,2}$	97.0	80.4	69.0	82.1
Opponent color $\text{LBP}_{8,1}$	98.8	91.2	46.9	79.0
$\text{LBP}^{u2}_{16,2}$ + RGB marginals	98.8	90.0	59.1	82.6

increases accuracy. This is what Padmapriya *et al.* (2003), for example, found when applying the LBP channel-wise to color textures. An important issue overlooked by this and many other studies is that the increase in accuracy is obtained with a tripled amount of information.

In Paper VI, three experiments with two different texture sets were arranged. The sets included 54 color textures from the VisTex database (MIT Media Lab 1995), and 68 color textures from the Outex texture database (see Section 2.6). In the first two experiments, only static illumination conditions were considered. In the last one, however, three different light sources were utilized.

RGB color histograms were used as simple color features. As gray-scale texture operators, the Gabor filter design of Manjunath & Ma (1996) and the LBP were chosen. The opponent color Gabor filtering technique of Jain & Healey (1998) and the OCLBP were utilized in color-texture description. In addition to joint color-texture features, two different methods of fusing separate color and texture measures were evaluated. The first of these combined the normalized similarities between corresponding feature vectors by summing them together. The second one, called the Borda count (Ho *et al.* 1994), combined class rankings given by separate classifiers.

Under constant illumination, the simple color indexing methods gave the best performance. The performance of the color descriptors however dropped drastically when the color of the illumination changed, and gray-scale texture yielded the best results. Therefore, if it is possible to determine whether illumination will change or not, one can decide which feature space to prefer.

The OCLBP performed clearly better than the other texture measures in static illumination, and its accuracy was almost equivalent to that of the color histograms. Opponent color texture was however defeated by either color or gray-scale texture in all three experiments, so its use is not easily justified. The best average result over the three experiments was obtained with $\text{LBP}^{u2}_{16,2}$ combined (using the Borda count) with the three marginal distributions of the RGB color channels, quantized to 256 levels. The results for the best method in each group of features (color, color-texture, gray-scale texture, color and texture) are shown in Table 3.1.

The results were later confirmed in a yet unpublished work in which a more comprehensive set of experiments was performed. Five different color spaces were investigated: RGB, rg chromaticity coordinates, the approximated K-L expansion based color system introduced by Ohta *et al.* (1980), the perceptually uniform

CIE 1976 ($L^*a^*b^*$), and HSV. The color constant color indexing method of Funt & Finlayson (1995) was included, and gray-scale texture measures were applied to color images channel-wise. A more comprehensive set of gray-scale texture features was also considered, and more methods combining separate color and texture features were evaluated. Again, joint color-texture descriptors were always outperformed by either color or texture alone. Furthermore, separate color and texture measures combined on a higher level also resulted in a better accuracy than joint color-texture descriptors.

Based on the results, Paper VI proposes that color and texture should be treated as separate phenomena. The preferable way of using them both seems to be to separate them. If needed, the separate measures can be combined on a higher level, possibly sequentially rather than in a parallel manner. The same conclusion was drawn by Soriano *et al.* (2001) with a coral reef classification application. In Section 4.1.1 this finding is utilized with the optimization of color and texture features for real-time visual inspection purposes.

4 Applications

As a very simple texture operator, the LBP is ideally suited for applications requiring fast feature extraction. Due to its simplicity and performance, many people have applied it to a number of different applications. These are detailed in the following sections. First, the real-time issues of the operator are discussed in Section 4.1. Section 4.2 presents a survey of the visual inspection applications the LBP has been applied to. In Section 4.3, some image retrieval applications with the LBP involved are presented. Finally, Section 4.4 briefly summarizes various other applications of the operator.

4.1 Real-time optimization

The time consumed by an analysis method consists of two factors: the time used for feature extraction and the time used for classification. If the features used are powerful enough, only a few of them are needed, and classification can be made quickly. Thus, when implementing an operator for real-time use, one must pay attention to its implementation and to the length of the feature vectors it produces. In addition, different classification principles have a great impact on the computational overhead. This section introduces techniques with which the LBP can be made to meet demanding time requirements without sacrificing performance.

4.1.1 Reducing features for robustness and speed

In the literature, a plethora of methods for selecting a sub-optimal subset of features is presented. In comparative surveys conducted by Zongker & Jain (1996), and later by Jain & Zongker (1997), the sequential forward floating search (SFFS) method of Pudil *et al.* (1994) was shown to perform best in most situations, with a reasonable computational burden. The survey did not however consider methods that handle multiple feature sets at the same time, like genetic algorithms and the beam search.

As a statistical measure of image texture, LBP distribution needs special attention when features are pruned. To ensure that the sum of each feature vector remains unity, an extra bin needs to be appended to the histograms. Another important point is that the time consumed in extracting the LBP distribution is not affected by the number of enabled features.

Paper II reports a study in which the beam search was used in optimizing LBP feature vectors. An experiment with tilted CUReT (Dana *et al.* 1999) textures was arranged. Two versions of the textures were considered: one at 0° and the other at 11.25° tilt.

As a reference result, the non-tilted textures were used as both training and testing data. The maximum possible number of non-overlapping 64×64 sub-images was taken from each texture. Half of the samples were used in training, the other half serving as a test set. With $\mathrm{LBP}_{8,1}$ (no interpolation), an accuracy of 98.5% was achieved, and $\mathrm{LBP}_{8,1}^{u2}$ (no interpolation) did not do much worse with a 98.1% score.

In optimization, the non-tilted textures were used as training samples, and non-overlapping 64×64 sub-images were extracted from the tilted textures. Half of these samples were used for optimization, and the other half for validation. The beam search with a beam width of six was exploited. The mean accuracy in classifying the validation set with six different 15-pattern sets was 91.8%. As a final result, the patterns that were present in at least five of the six sets were selected. There were 14 such patterns, most of which were uniform. With these patterns enabled, the classification accuracy was 91.6%. Compared to the 88.5% score of the full LBP histogram, the enhancement was clear.

The results show that it is possible to find an application-specific subset of LBP codes that performs even better than the whole histogram. In this particular application, the optimized feature vector was over 18 times shorter than the original one, resulting in an 18-fold increase in classification speed. At the same time, the shorter feature vector results in clearly greater accuracy, showing that purging unreliable LBP codes enhances the robustness of the texture descriptor.

The SFFS algorithm of Pudil *et al.* (1994) is an enhanced version of the forward search. On each iteration, all remaining features are added to the feature set in turn, and the one that resulted in the largest performance increase is selected. In a backward step, each of the already selected features is disabled in turn, and the performance of the remaining features is measured to see if removing the feature could increase performance.

In Paper IX, the SFFS algorithm was successfully utilized in pruning LBP codes in a paper inspection application. An attempt was also made to utilize it in a more general texture discrimination problem with a set of Outex textures. The results further confirm the discovery that only a small subset of LBP codes may be enough for high accuracy in certain applications.

4.1.2 Optimized software implementation

When implementing the LBP operator for a general purpose CPU, there are a number of issues affecting the computational performance that must be addressed. A decent microprocessor can nowadays perform hundreds of millions of instructions in a second. The full capability cannot however be utilized if proper care is not taken. In CISC processors, like the present PC processors, the number of micro-operations per instruction varies — some instructions consume more clock cycles than others. Also, to exploit the full capability of the pipeline, the number of conditional jumps in the code must be minimized. Finally, the number of memory accesses should be kept as low as possible. At least it must be ensured that only very few memory references miss the first-level cache.

An apparent problem with the LBP operator is that each pixel value in a neighborhood must be compared to the value of the center. With P neighbors this causes, on average, $P/2$ pipeline stalls per pixel, because the comparison result cannot be predicted. The operator can, however, be implemented without conditional jumps, as will be explained in the following. The CISC instructions used are all one-cycle operations, with the exception of memory-to-register and register-to-memory transfers.

The trick used to eliminate conditional branching is based on the processors' internal representation of integer values, called two's complement. A negative number is created by inverting each bit of the corresponding positive number (one's complement) and then adding one to the result, ignoring possible overflows. In Table 4.1, some positive and negative binary numbers are shown.

Table 4.1. Small positive and negative binary numbers.

Decimal	4-bit binary	8-bit binary
4	0100	00000100
3	0011	00000011
2	0010	00000010
1	0001	00000001
0	0000	00000000
-1	1111	11111111
-2	1110	11111110
-3	1101	11111101
-4	1100	11111100

The difference between the four-bit and eight-bit representations is easily seen: the most significant bit (MSB) is extended to fill empty space. Independent of the number of bits, the MSB tells the sign of the number.

To build an LBP code, one needs to set a bit in a binary number depending on whether a value in a neighborhood is larger than the center pixel. Instead of comparing these values, one can utilize their difference. When the value of a neighbor is subtracted from the value of the center, the MSB of the result can be directly used in building the LBP code. Care must, however, be taken to ensure no overflows or underflows occur when calculating the difference. That is, the calculation must be performed with a sufficiently large number of bits. Furthermore, one must be subtracted from the difference so that a neighbor value equal to the center produces the sign bit.

Figure 4.1 shows an example of calculating a four-bit LBP code. The neighborhood on the left is traversed counter-clockwise, starting at "3". The signs of the differences are extracted by a logical AND operation with 1000_2. The sign is then shifted to the right $3 - Index$ times. Finally, the LBP code is obtained with a logical OR operation of the shifted signs.

	4	
2	3	3
	1	

Index	Difference	Binary	Shifted sign
0	$3-3-1=-1$	1111	0001
1	$3-4-1=-2$	1110	0010
2	$3-2-1=0$	0000	0000
3	$3-1-1=1$	0001	0000
		$\mathrm{LBP}_{4,1}$ (logical OR):	0011

Figure 4.1. Optimized calculation of $\mathrm{LBP}_{4,1}$.

Paper IX presents an optimized C code for calculating the $\mathrm{LBP}_{8,1}$ for an image array stored linearly in memory as 32-bit integers. The code does not employ bilinear interpolation for the diagonal neighbors, but uses nearest-neighbor interpolation instead for maximum performance. The compiled code uses only once-cycle instructions on an AMD Athlon$^{\mathrm{TM}}$, with the exception of memory-referencing instructions that need more, depending on the state of the cache. On a 1GHz processor, the implementation achieves video rates (24 images/second) with images as large as 960×960 pixels.

Figure 4.2 shows how the frame rate decreases with image size. A similar curve is drawn for a corresponding C++ version showing slight language overhead. The third curve shows the speed of a C++ implementation that uses bilinear interpolation. The difference between the interpolated and the non-interpolated versions is clear, even when all necessary interpolation factors are precalculated. The x axis in the figure represents the width and height of the input images. Thus, the number of pixels to be processed is x^2. The curves start at 128. On the y axis, the number of frames per second is shown.

4.1.3 Classification

As already noted, the most important factor affecting the computational cost of classification is the length of the feature vectors. Different classifiers also have dif-

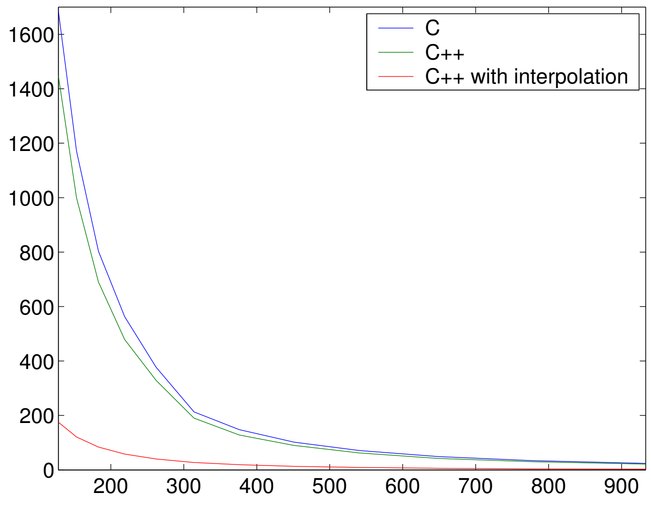

Figure 4.2. Frame rates of the different implementations of the LBP$_{8,1}$ operator as functions of image size on a 1GHz PC processor.

ferent performance characteristics. The computational performance of many non-parametric classifiers, like the k-NN classifier, decreases linearly with the number of training samples. Parametric ones, such as artificial neural networks, just incorporate the information about training samples in their internal model of the data. Their computational cost is thus unaffected by the amount of training data. Although the performance of nearest neighbor type parametric classifiers can be enhanced (See e.g. Dasarathy 1991), they are no match for the non-parametric ones with a large amount of training data. In Paper IX, the self-organizing map (SOM) was used as a fast classifier for LBP distributions.

A SOM consists of a finite amount of nodes representing the original high-dimensional data in a low-dimensional node grid (Kohonen 1997). The code vectors on the map are adapted and labeled with training data. Each node is labeled according to the most common class among the training samples closest to its code vector. In classification, the SOM works as a simple vector quantizer: an unknown sample is classified according to the code vector closest to it.

Typically, the size of a SOM is selected so that the number of training samples is 5-20 times the number of nodes. As a result, the time consumed in classifying an unknown sample with the SOM is approximately 5-20 times shorter than that of the k-NN classifier. With a relatively small number of training samples, the speed difference may, however, not be that large, as the size of the SOM cannot be made arbitrarily small without destroying its learning capabilities. The type of input data affects the size of the SOM. If the inspected data are complex and the features have no ability to discriminate them correctly, a larger SOM is needed. A number of methods of optimizing SOM-based classification exists (Niskanen *et al.*

2002). Also, methods for making k-NN classification faster have been presented (Dasarathy 1991). Since the speed finally comes down to the number of code vectors or training samples, the optimization methods were not considered in Paper IX. Moreover, the sizes of the self-organizing maps used were so small that the optimization methods would have been useless anyway.

In training a SOM and in classification, the Euclidean distance has been conventionally used. In the experiments reported here, the log-likelihood measure (Eq. 3.11) was used exclusively.

4.2 Visual inspection

In industrial processes with a large throughput, quality control is a critical issue. Defective material must be blocked before it gets to the customer. Even when there are no actual defects, the material may need to be processed according to its quality. In short, inspection results can be utilized in controlling the production process for greater yield, quality, or both.

In visual inspection, the goal is to classify products or detect their defects according to their visual properties. The traditional way of doing this is to use a trained human. Even professionals are slow, error-prone and inconsistent. For this reason, computer vision methods have long been applied to visual inspection, mostly with good results. For a thorough survey of automated visual inspection, one should see the paper by Newman & Jain (1995). A good overview is also provided by Davies (1990, chp. 17).

As discussed in Section 2.4, texture features can be used in visual inspection as a supplement to color information. In some applications, texture is the only information available, and needs to be used alone. Although many potential application areas for texture analysis exist in industry, only a limited number of successful exploitations have been reported (Song *et al.* 1992, Pietikäinen & Ojala 1996). One of the major problems is the non-uniformness of real-world textures, which is caused by changes in orientation, illumination, scale, etc. In addition, the computational requirements of many proposed texture features are high. (Silvén 2000)

Due to its gray-scale invariance and low computational overhead, the LBP is a good candidate for a feature with visual inspection. One of the first applications of the LBP was metal surface inspection (Pietikäinen *et al.* 1994, Ojala 1997, Xu *et al.* 1997), in which the LBP showed superior performance compared to Laws' masks (Laws 1979) and GLD features. Recently, Marzàbal *et al.* (2001) introduced a system in which a subset of the LBP codes was used in characterizing defects in automobile engine valves. A 99.5% recognition rate was achieved in real time. Yet another industrial application was reported by Paclík *et al.* (2002). They compared different feature extraction schemes in a segmentation process needed for the analysis of laundry detergents. It was concluded that "LBP, CM [co-occurrence matrix], SGLD and intensity features outperform the Gabor and DCT filters both in the discriminative power and in the computational time".

4.2.1 Wood inspection

One of the most commonly reported applications of the LBP is wood inspection, with which the operator has been used with color measures (Kyllönen & Pietikäinen 2000). A common approach has been to use color and texture features with non-supervised clustering methods like the SOM (Niskanen *et al.* 2001, Silvén *et al.* 2003). In Paper VIII, color and texture features, including the LBP, were optimized for a demanding real-time parquet inspection application.

Paper VIII describes a method of optimizing color and texture features for real-time visual inspection tasks. As a case study, a real-time parquet inspection application is presented. The paper presents no new optimization methods. Instead, a comparison of two existing search methods neglected by Zongker & Jain (1996) — the beam search and a genetic algorithm — are evaluated. Their use is extended to the simultaneous utilization of color and texture descriptors.

In the experiments, a set of 150 beech wood parquet slabs from ESPRIT Project 21023 (1994) were used (Kauppinen 1999). In addition to healthy wood, the slabs contain 15 different defect types which needed to be separated. For color features, percentiles calculated from RGB histograms were used. For texture description, the selected features included the LBP, alongside the reduced multi-dimensional co-occurrence and signed difference histograms (Valkealahti & Oja 1998, Ojala *et al.* 1999, 2001). Also, a set of features based on contrast edges and on the Hough transform were evaluated.

Color and texture features were extracted independent of each other and optimized with the beam search and a genetic algorithm. The evaluation of the optimization result was performed by classifying the 21491 samples that were not used in optimization. The color percentile features achieved an accuracy of 89.8% after optimization with the genetic algorithm. The beam search did even better with less features as the classification accuracy was 90.3%. With all color features enabled, the accuracy was 89.3%.

The beam search resulted in more compact sets of features than the genetic algorithm. In most cases, it also produced a better classification accuracy. The utility attained with texture information was relatively small. With the LBP, only one percentage unit was gained. It might, however, be quite an important unit in the wood industry where huge volumes are processed. With the testing material used in the experiments, one percentage unit corresponds to over 200 samples. The measured difference is therefore statistically reliable.

4.2.2 Paper inspection

Paper inspection has been a successful application area with the LBP. Iivarinen used the LBP and GLCM texture features with a SOM-based non-supervised segmentation method in detecting defects in surface images. Paper samples were used in experiments (Iivarinen 2000a,b). The performance of the LBP was found to be equivalent to that of the GLCM methods, but its computational simplicity was seen as a clear advantage. Recently, the LBP was applied to the measurement of paper quality by Turtinen *et al.* (Turtinen *et al.* 2003a,b). Non-supervised learning was used in classifying paper samples to four quality classes. With the LBP, excellent results were reported. Paper IX reports a study in which the same problem is solved in real time, utilizing the techniques detailed in Section 4.1.

On-line paper characterization is a good example of a real-time inspection application. The paper web moves about 30 m/s, posing extreme requirements for an analysis system. The image acquisition itself is a difficult problem which must be considered carefully. Requirements for image analysis are simple: faster is better. With a spatial image width of about 0.6 m, a frame rate of 50 fps is needed to ensure no material is missed. Therefore, taking an image, transferring it to the analysis system, extracting features, and classifying may take only about 20 ms. In analyzing the quality of paper it probably causes no harm to miss a few centimeters between successive frames. The situation would be quite different if defects should be detected. In any case, a continuous estimate of paper quality is of high value to the control system.

The problem used in judging the performance of the proposed system was to classify paper samples into four quality classes. The testing data contained back-lit paper images in which brighter areas indicate thinner paper. The data to be analyzed were 8-bit gray-scale images with 756×566 pixels.

The SFFS algorithm was utilized in reducing the length of the LBP feature vector. As a fast classifier, a SOM with 10×8 nodes was used. With only three LBP codes, an accuracy of 99.2% was achieved, which is more than sufficient. Compared to the result of k-NN classification with all LBP codes enabled (99.8%), the performance drop is not bad, given a 65-fold increase in speed. The time consumed in determining the quality class of a previously unseen sample of paper was successfully reduced to 20 ms on a 1GHz PC. If image acquisition and transfer can be done in parallel to the analysis task, this means that practically no paper is missed between successive frames.

The real-time inspection system presented in Paper IX was shown to work also in a more general context. In an experiment with natural textures from the Outex database, an eight-fold increase with speed was achieved compared to k-NN classification, with no noticeable accuracy penalty.

4.3 Image retrieval

During the past few years, digital multimedia has become ubiquitous. Especially the Internet has produced huge amounts of image and video material that is not organized in any meaningful manner according to content. Content-based image retrieval (CBIR) techniques try to tackle this problem by automatically analyzing the semantics of images or video. A number of different methods have been proposed for this task. Usually, many different cues including color and texture are used. Different features have been evaluated for example by Ma & Zhang (1998). IBM's QBIC (Flickner *et al.* 1995) and the Photobook system of Pentland *et al.*

(1996) might be the best known studies in the area.

A number of researchers have used LBP as part of their CBIR systems. Lew (2000) presented a system called "Imagescape" that used distribution-based texture features as texture models. These features included the LBP, LBP/C, and "Trigrams" that was a variant of the LBP calculated from an edge image. Also Berman & Shapiro (1999) have used LBP as a texture feature in their "Flexible Image Database System". Schaefer (2001) in turn applied the operator to JPEG-compressed images for image retrieval. He found that with compressed images, sub-sampling does not decrease indexing accuracy. LBP has also appeared as a texture feature in the studies of Tao & Dickinson (1996), Huijsmans *et al.* (1997), and Kauniskangas *et al.* (1997).

Liao & Chen (2002) and later Yao & Chen (2003) reported studies in which retrieval experiments were performed with distorted color textures. A new texture feature, called "local edge patterns", was derived from the LBP. The operator works just like the LBP, but it is applied to an edge image instead of directly to the gray values. Compared to LBP/VAR, the new operator was shown to yield better retrieval rates with distorted color textures, although no information was given about the parameters of the LBP/VAR operator used.

One of the major problems in indexing multimedia, despite its huge amount and unorganized structure, has been the lack of a universally accepted way of creating content descriptions. The ISO/IEC MPEG-7 standard (Martinez 2001) is an attempt to create a standardized description for digital audiovisual content. It is also known as the "Multimedia Content Description Interface". The standard specifies a set of descriptors that are used in describing multimedia content. It also specifies description schemes, which define the structure and semantics of the descriptions. On a meta-level, description schemes can be used in describing relationships between description schemes. Descriptions and description schemes are delineated with an XML-based description definition language. (Chang *et al.* 2001).

MPEG-7 is composed of seven parts, of which "Visual" (Sikora 2001) addresses the description of visual elements, such as video or still images. The "eXperimentation Model" (XM) of the Visual standard (Yamada *et al.* 2001) in turn describes the extraction and matching of visual multimedia descriptors. For texture, this document presents three descriptors: "edge histogram", "homogeneous texture" and "texture browsing". The first of these captures the spatial distribution of edges. The two other features are based on Gabor filtering. In Paper VII, a large-scale empirical experiment with these standardized texture descriptors and the LBP was conducted.

As a test bench for the aforementioned "standard" texture description methods and the LBP, a texture retrieval experiment was arranged with all the 319 textures (split into 20 non-overlapping parts) in the Outex database so that each of the 6380 images served as a query image in turn. Each texture method was used in finding the twenty images they considered the best matches for each query.

With this data, the accuracy of the LBP was clearly better than that of the other descriptors. The best result obtained with the "standard" texture methods was the 47.4% score of homogeneous texture. In its very basic form, LBP achieved a 57.9% retrieval rate. A simple two-resolution version was even better with a 63.7% score. The speed of the LBP was from 2.5 up to 5770 times greater than that of the other methods.

The important thing discovered in this study was that with very low computational complexity, good accuracy can be achieved with the LBP. The retrieval rates are comparable to or even better than those of the "standard" methods, even with a very short feature vector.

4.4 Other applications

Many people have chosen the LBP for applications requiring image segmentation. Especially the LBP/C and LBP/VAR operators have been popular. Rubio *et al.* (2002) write: "After testing several different textons, we chose the local binary pattern and contrast measure." They use the LBP/C with a split-and-merge type unsupervised segmentation algorithm to quickly divide large aerial images into meaningful regions. Also the segmentation algorithm seems to be influenced by that presented by Ojala & Pietikäinen (1999b). Recently, Lucieer *et al.* (2003) used the LBP for the segmentation of aerial images. This time, LBP/VAR was utilized with a segmentation algorithm combining aspects of split-and-merge and fuzzy c-means methods. Wang *et al.* (2001) in turn present a system with which some preliminary results on the unsupervised classification of endoscopic images are obtained. The paper presents a variation of the LBP called LBP/I, which incorporates intensity information into the LBP distribution. A compacted version of this distribution is used in segmenting colonoscopic images, with "promising" results. Iivarinen (2000b) used the LBP in detecting defects on surface textures (Section 4.2.2). The LBP-based "local edge patterns" (Section 4.3) were used by Chen & Chen (2002) in segmenting color images.

One of the most exotic uses of the LBP is the hair-based person detection system of Cohen *et al.* (2000). In this application, the LBP is used as a texture feature "because of the real-time nature of the problem". Among the texture features tested, the LBP was reported to yield the best accuracy.

Garcia *et al.* (2001a) have developed a system for the creation of visual mosaics of the seabed. In the experiments, they evaluated a number of different texture methods, including GLCM, energy filters, MRF models, and the LBP. The contrast measure of the LBP/C operator and the energy filters were selected due to their performance. The results were utilized in estimating the motion of an underwater robot (Garcia *et al.* 2001b). The LBP distribution itself was also used in a similar task (Garcia *et al.* 2001c).

Finally, Soriano *et al.* (2001) try to combine color and texture (LBP) in classifying images of coral reef. The LBP alone provided the greatest accuracy. The authors hypothesize that a better approach would be to use color and texture sequentially.

5 Summary

The LBP operator is a theoretically simple yet a very powerful method of analyzing textures. Through the extension developed during this thesis, the LBP operator was made into a really powerful measure of image texture, showing excellent results in terms of accuracy and computational complexity in many empirical studies. Its rotation invariance properties were enhanced, and the main shortcoming of the operator — its inability to detect large-scale structures — was removed. As a side-product, cellular automata were applied to texture analysis, presumably for the first time ever. The cellular automaton LBP is still in its infancy, and active development is needed to maximally utilize its capabilities.

The LBP operator is proposed as a truly unifying approach to the traditionally divergent statistical and structural models of texture analysis. Texture is described in terms of micro-primitives (textons), and their statistical placement rules. Optionally, the primitives may be coupled with a complementary measure of local image contrast, which measures the strength of the primitives.

An empirical study was performed to see whether texture and color information should be used jointly or separately. The results show that color and texture should be treated as separate phenomena in general. This result was used in applications with both color and texture information. The real-time applicability of the LBP was enhanced via a fast software implementation, reduced feature sets, and SOM-based classification.

Currently, the LBP operator is relatively invariant with respect to changes in illumination and image rotation. It can resist even texture scale changes. Still, there are problems that need to be solved. In natural scenes, textures can appear in arbitrary positions, requiring analysis methods invariant against 3-D distortions. The next challenge could be using the LBP approach in analyzing not only pure texture images but a wide range of different natural scenes. In fact, this work has already started (Pietikäinen *et al.* 2003).

As of writing, the LBP methodology seems to be gaining acceptance among texture researchers all around the world. Due to its simplicity and unquestionable accuracy, the methodology has found its way into numerous different applications, surveyed in Chapter 4. It should be noted that most of these utilize the original version of the operator. It is likely that with the extension presented here, the operator will find many other application areas.

Some people have also found creative ways of extending the operator. Schael (2002), for example, has developed powerful features based on a “relational kernel function”. The features are invariant with respect to 2-D Euclidean motion and strictly increasing gray scale transformations. The approach is strongly based on the LBP. The LBP operator applied to edge images (LBPEdge or LEP) in turn is shown to perform well in retrieving distorted color textures (Liao & Chen 2002).

Many researchers have long credited Gabor filtering as the state-of-the-art in texture analysis. With the LBP, the status quo ought to change. Despite being simple it is also fast, and in many applications very accurate. These are properties that application developers cannot overlook. The future will show whether the research community does.

References

Adams R & Bischof L (1994) Seeded region growing. IEEE Transactions on Pattern Analysis and Machine Intelligence 16(6): 641–647.

Aleksander I & Stonham T (1979) Guide to pattern recognition using random access memories. Computers and Digital Techniques 2(1).

Baykut A, Atalay A, Erçil A & Güler M (2000) Real-time defect inspection of textured surfaces. Real-Time Imaging 6(1): 17–27.

Beck J, Prazdny S & Rosenfeld A (1983) A theory of textural segmentation. In: Beck J, Hope B & Rosenfeld A (eds) Human and Machine Vision. Academic Press, New York.

Berman A & Shapiro L (1999) A flexible image database system for content-based retrieval. Computer Vision and Image Understanding 75(1/2): 175–195.

Billings S & Yang Y (2003a) Identification of probabilistic cellular automata. IEEE Transactions on Systems, Man, and Cybernetics – Part B. In press.

Billings S & Yang Y (2003b) Identification of the neighborhood and CA rules from spatio-temporal CA patterns. IEEE Transactions on Systems, Man, and Cybernetics – Part B. In press.

Bossomaier T, Cranny T & Schneider D (1999) A new paradigm for evolving cellular automata rules. Proc. 1999 Congress on Evolutionary Computation, 1: 169–176.

Boukouvalas C, Kittler J, Marik R & Petrou M (1999) Color grading of randomly textured ceramic tiles using color histograms. IEEE Transactions on Industrial Electronics 46(1): 219–226.

Boyer K, Mirza K & Ganguly G (1994) The robust sequential estimator: a general approach and its application to surface organization in range data. IEEE Transactions on Pattern Analysis and Machine Intelligence 16(10): 987–1001.

Brodatz P (1966) Textures: a Photographic Album for Artists and Designers. Dover Publications, New York.

Caelli T & Reye D (1993) On the classification of image regions by colour, texture and shape. Pattern Recognition 26(4): 461–470.

Campbell F & Robson J (1968) Application of Fourier analysis to the visibility of gratings. Journal of Physiology 197: 551–566.

Çesmeli E & Wang D (2001) Texture segmentation using Gaussian-Markov random fields and neural oscillator networks. IEEE Transactions on Neural Networks 12(2): 394–404.

Chang SF, Sikora T & Puri A (2001) Overview of the MPEG-7 standard. IEEE Transactions on Circuits and Systems for Video Technology 11(6): 688–695.

Chaudhuri B & Sarkar N (1995) Texture segmentation using fractal dimension. IEEE Transactions on Pattern Analysis and Machine Intelligence 17: 72–77.

Chaudhuri B, Sarkar N & Kundu P (1993) Improved fractal geometry based texture segmentation technique. IEE Proceedings Part E 140: 233–241.

Chellappa R, Kashyap R & Manjunath B (1999) Model-based texture segmentation and classification. In: Chen C, Pau L & Wang P (eds) Handbook of Pattern Recognition and Computer Vision, pp 249–282. World Scientific, Singapore, 2nd edition.

Chellappa R & Manjunath B (2001) Texture classification and segmentation: tribulations, triumphs and tributes. In: Davis L (ed) Foundations of Image Understanding, pp 219–240. Kluwer.

Chen JL & Kundu A (1995) Unsupervised texture segmentation using multichannel decomposition and hidden Markov models. IEEE Transactions on Image Processing 4: 603–619.

Chen KM & Chen SY (2002) Color texture segmentation using feature distributions. Pattern Recognition Letters 23(7): 755–771.

Chen P & Pavlidis T (1979) Segmentation by texture using a cooccurrence matrix and a split-and-merge algorithm. Computer Graphics and Image Processing 10: 172–182.

Cohen I, Garg A & Huang T (2000) Vision-based overhead view person recognition. Proc. 15th International Conference on Pattern Recognition, Barcelona, Spain, 1: 1119–1124.

Conners R & Harlow C (1980) A theoretical comparison of texture algorithms. IEEE Transactions on Pattern Analysis and Machine Intelligence 2(3): 204–222.

Cross G & Jain A (1983) Markov random field texture models. IEEE Transactions on Pattern Analysis and Machine Intelligence 5: 25–39.

Cula O & Dana K (2001a) Compact representation of bidirectional textures. Proc. Computer Vision and Pattern Recognition, Kauai, Hawaii, pp 1041–1047.

Cula O & Dana K (2001b) Recognition methods for 3D texture surfaces. Proc. SPIE Conference on Human Vision and Electronic Imaging VI, 4299: 209–220.

Daemi M, Elliman D & Sahota P (1994) Training genetically evolving cellular automata for image processing. Proc. International Symposium on Speech, Image Processing and Neural Networks, 2: 753–756.

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

Dasarathy B (ed) (1991) Nearest Neighbor (NN) Norms: NN Pattern Classification Techniques. IEEE Computer Society Press, Los Alamitos, California.

Daugman J (1988) Complete discrete 2D Gabor transforms by neural networks for image analysis and compression. IEEE Transactions on Acoustics, Speech, and Signal Processing 36(7): 1169–1179.

Davies E (1990) Machine Vision: Theory, Algorithms, Practicalities. Academic Press, London.

DeValois R, Albrecht D & Thorell L (1982) Spatial-frequency selectivity of cells in macaque visual cortex. Vision Research 22: 545–559.

DeValois R & DeValois K (1990) Spatial Cision. Oxford University Press.

DeYoe E & van Essen D (1996) Concurrent processing streams in monkey visual cortex. Trends. Neurosci. 11: 219–226.

Dubuisson-Jolly MP & Gupta A (2000) Color and texture fusion: application to aerial image segmentation and GIS updating. Image and Vision Computing 18: 823–832.

Duda R, Hart P & Stork D (2001) Pattern Classification. John Wiley & Sons, New York.

Dyer C & Rosenfeld A (1981) Parallel image processing by memory-augmented cellular automata. IEEE Transactions on Pattern Analysis and Machine Intelligence 3(1): 29–41.

ESPRIT Project 21023 (1994) Catie - colour and texture inspection equipment. http://www.ee.oulu.fi/Catie/.

Faugeras O (1978) Texture analysis and classification using a human visual model. Proc. 3rd International Conference on Pattern Recognition, Tokyo, Japan, pp 549–552.

Faugeras O & Pratt W (1980) Decorrelation methods of texture feature extraction. IEEE Transactions on Pattern Analysis and Machine Intelligence 2: 323–332.

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

Francos J, Meiri A & Porat B (1993) A unified texture model based on a 2-D Wold-like decomposition. IEEE Transactions on Signal Processing 41: 2665–2678.

Funt B & Finlayson G (1995) Color constant color indexing. IEEE Transactions on Pattern Analysis and Machine Intelligence 17(5): 522–529.

Gabor D (1946) Theory of communication. Journal of Institution of Electrical Engineers 93(26): 429–459.

Garcia R, Batlle J & Cufi X (2001a) A system to evaluate the accuracy of a visual mosaicking methodology. Proc. OCEANS 2001, Honolulu, Hawaii.

Garcia R, Cufi X & Battle J (2001b) Detection of matchings in a sequence of underwater images through texture analysis. Proc. IEEE International Conference on Image Processing, 1: 361–364.

Garcia R, Cufi X & Carreras M (2001c) Estimating the motion of an underwater robot from a monocular image sequence. Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems, Maui, Hawaii, pp 1682–1687.

Gevers T & Smeulders A (2001) Color constant ratio gradients for image segmentation and similarity of texture objects. Proc. Computer Vision and Pattern Recognition, Kauai, Hawaii, pp 1–8.

Granlund GH (1978) In search of a general picture processing operator. Computer Graphics and Image Processing 8: 155–173.

Haralick R (1992) Computer vision theory: the lack thereof. Computer Vision, Graphics and Image Processing 36: 372–386.

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

Haralick R & Shapiro L (1985) Image segmentation techniques. Computer Vision, Graphics and Image Processing 29(1): 100–132.

Harwood D, Ojala T, Pietikäinen M, Kelman S & Davis S (1993) Texture classification by center-symmetric auto-correlation, using Kullback discrimination of distributions. Technical report, Computer Vision Laboratory, Center for Automation Research, University of Maryland, College Park, Maryland. CAR-TR-678.

He DC & Wang L (1990) Texture unit, texture spectrum, and texture analysis. IEEE transactions on geoscience and remote sensing 28(4): 509–512.

Heikkinen J (1993) Applications of texture analysis and classification methods to metal surface inspection problems. Diploma thesis, University of Oulu.

Hepplewhite L & Stonham T (1996) Texture classification using N-tuple pattern recognition. Proc. 13th International Conference on Pattern Recognition, Vienna, Austria, 4: 159–163.

Hernandez G & Herrann H (1996) Cellular automata for elementary image enhancement. Graphical Models and Image Processing 58(1): 82–89.

Ho T, Hull J & Srihari S (1994) Decision combination in multiple classifier systems. IEEE Transactions on Pattern Analysis and Machine Intelligence 16(1): 66–75.

Huijsmans D, Leq M & Denteener D (1997) Quality measures for interactive image retrieval with a performance evaluation of two 3×3 texel-based methods. Proc. 9th International Conference on Image Analysis and Processing, Florence, Italy, 2: 22–29.

Iivarinen J (2000a) Surface defect detection with histogram-based texture features. Proc. SPIE Vol. 4197 Intelligent Robots and Computer Vision XIX: Algorithms, Techniques, and Active Vision, pp 140–145.

Iivarinen J (2000b) Unsupervised segmentation of surface defects with simple texture measures. In: Pietikäinen M (ed) Machine Perception and Artificial Intelligence, volume 40, pp 231–238. World Scientific.

Ikenaga T & Ogura T (2000) Real-time morphology processing using highly parallel 2-d cellular automata CAM$_2$. IEEE Transactions on Image Processing 9(12): 2018–2026.

Jain A & Farrokhnia F (1991) Unsupervised texture segmentation using Gabor filters. Pattern Recognition 24(12): 1167–1186.

Jain A & Healey G (1998) A multiscale representation including opponent color features for texture recognition. IEEE Transactions on Image Processing 7(1): 124–128.

Jain A & Karu K (1996) Learning texture discrimination masks. IEEE Transactions on

Pattern Analysis and Machine Intelligence 18: 195–205.

Jain A & Zongker D (1997) Feature selection: evaluation, application and small sample performance. IEEE Transactions on Pattern Analysis and Machine Intelligence 19(2): 153–158.

Jain R & Binford T (1991) Ignorance, myopia, and naivete in computer vision systems. CVGIP: Image Understanding 53: 112–117.

Julesz B (1981) Textons, the elements of texture perception, and their interactions. Nature 290: 91–97.

Kass M, Witkin A & Terzopoulos D (1988) Snakes: active contour models. International Journal of Computer Vision 1: 321–332.

Kauniskangas H, Sauvola J, Pietikäinen M & Doermann D (1997) Content-based image retrieval using composite features. Proc. 10th Scandinavian Conference on Image Analysis, Lappeenranta, Finland, pp 35–42.

Kauppinen H (1999) Development of a color machine vision method for wood surface inspection. Dr. tech. dissertation, University of Oulu. http://herkules.oulu.fi/isbn9514254244/.

Khan A, Choudhury P, Dihidar K, Mitra S & Sarkar P (1997) VLSI architecture of a cellular automata machine. Computers & Mathematics with Applications 33(5): 79–94.

Kohonen T (1997) Self-organizing maps. Springer-Verlag, Berlin, Germany.

Kullback S (1968) Information theory and statistics. Dover Publications, New York.

Kumar A & Pang G (2002) Defect detection in textured materials using Gabor filters. IEEE Transactions on Industry Applications 38(2): 425–440.

Kyllönen J & Pietikäinen M (2000) Visual inspection of parquet slabs by combining color and texture. Proc. IAPR Workshop on Machine Vision Applications, Tokyo, Japan, pp 187–192.

Laws K (1979) Texture energy measures. Proc. Image Understanding Workshop, pp 47–51.

Leung T & Malik J (1999) Recognizing surfaces using three-dimensional textons. Proc. International Conference on Computer Vision, 2: 1010–1017.

Leung T & Malik J (2001) Representing and recognizing the visual appearance of materials using three-dimensional textons. International Journal of Computer Vision 43(1): 29–44.

Levine M (1985) Vision in Man and Machine. McGraw-Hill.

Lew M (2000) Next-generation web searches for visual content. IEEE Computer 33(11): 46–53.

Liao CJ & Chen SY (2002) Complementary retrieval for distorted images. Pattern Recognition 35(8): 1705–1722.

Lucieer A, Fisher P & Stein A (2003) Fuzzy object identification using texture based segmentation of high-resolution DEM and remote sensing imagery of coastal area in England. Proc. Second International Symposium on Spatial Data Quality, Hong Kong.

Ma W & Zhang H (1998) Benchmarking of image features for content-based retrieval. Proc. 32nd Asilomar Conference on Signals, Systems and Computers, 1: 253–257.

Mäenpää T (1999) Texture analysis using Fourier spectra and local binary patterns. Diploma thesis, University of Oulu.

Malik J, Belongie S, Shi J & Leung T (1999) Textons, contours and regions: cue integration in image segmentation. Proc. 7th International Conference on Computer Vision, Kerkyra, Greece, 2: 918–925.

Malik J & Perona P (1990) Preattentive texture discrimination with early vision mechanisms. Journal of the optical society of America A 7(5): 923–932.

Manjunath B & Ma W (1996) Texture features for browsing and retrieval of image data. IEEE Transactions on Pattern Analysis and Machine Intelligence 18(8): 837–842.

Martinez J (2001) Overview of the MPEG-7 standard. ISO/IEC JTC1/SC29/WG11 N4031.

Marzàbal A, Torrens C & Grau A (2001) Texture-based characterization of defects in automobile engine valves. Proc. IX Symposium Nacional de Reconicimiento de Formas y Analisis de Imagenes.

McLean G (1993) Vector quantization for texture classification. IEEE Transactions on Systems, Man, and Cybernetics 23(3): 637–649.

Mirmehdi M & Petrou M (2000) Segmentation of color textures. IEEE Transactions on Pattern Analysis and Machine Intelligence 22(2): 142–159.

MIT Media Lab (1995) Vision texture – VisTex database. http://www-white.media.mit.edu/vismod/imagery/VisionTexture/vistex.html.

Mojsilovic A, Kovacevic J, Kall D, Safranek R & Ganapathy S (2000) Matching and retrieval based on the vocabulary and grammar of color patterns. IEEE Transactions on Image Processing 9(1): 38–54.

Newman T & Jain A (1995) A survey of automated visual inspection. Computer Vision and Image Understanding 61(2): 231–262.

Niskanen M, Kauppinen H & Silvén O (2002) Real-time aspects of SOM-based visual surface inspection. Proc. SPIE Machine Vision Applications in Industrial Inspection X, San Jose, CA, USA, 4664: 123–134.

Niskanen M, Silvén O & Kauppinen H (2001) Color and texture based wood inspection with non-supervised clustering. Proc. 12th Scandinavian Conference on Image Analysis, Bergen, Norge, pp 336–342.

Ohlander R, Price K & Reddy D (1978) Picture segmentation using a recursive region splitting method. Computer Graphics and Image Processing 8: 313–333.

Ohta Y, Kanade T & Sakai T (1980) Color information for region segmentation. Computer Graphics Image Processing 13: 222–241.

Ojala T (1997) Nonparametric texture analysis using spatial operators, with applications in visual inspection. Dr. tech. dissertation, University of Oulu. http://herkules.oulu.fi/isbn9514246187/.

Ojala T & Pietikäinen M (1999a) Texture classification. In: Fisher R (ed) CVonline – Compendium of Computer Vision. University of Edinburgh. http://www.dai.ed.ac.uk/CVonline/.

Ojala T & Pietikäinen M (1999b) Unsupervised texture segmentation using feature distributions. Pattern Recognition 32: 400–486.

Ojala T, Pietikäinen M & Harwood D (1996) A comparative study of texture measures with classification based on feature distributions. Pattern Recognition 29: 51–59.

Ojala T, Pietikäinen M & Kyllönen J (1999) Gray level cooccurrence histograms via learning vector quantization. Proc. 11th Scandinavian Conference on Image Analysis, Kangerlussuaq, Greenland, pp 103–108.

Ojala T, Valkealahti K, Oja E & Pietikäinen M (2001) Texture discrimination with multidimensional distributions of signed gray level differences. Pattern Recognition 34: 727–739.

Paclík P, Duin R, van Kempen G & Kohlus R (2002) Supervised segmentation of textures in backscatter images. Proc. 16th International Conference on Pattern Recognition, Québec, Canada, 2: 490–493.

Padmapriya N, Ghita O & Whelan P (2003) Experimentation on the use of chromaticity features, local binary pattern and discrete cosine transform in colour texture analysis. Proc. 13th Scandinavian Conference on Image Analysis, Göteborg, Sweden. Accepted.

Panjwani D & Healey G (1995) Unsupervised segmentation of textured color images. IEEE Transactions on Pattern Analysis and Machine Intelligence 17(10): 939–954.

Papathomas T, Kashi R & Gorea A (1997) A human vision based computational model for chromatic texture segregation. IEEE Transactions on Systems, Man, and Cybernetics B 27(3): 428–440.

Paschos G (2001) Perceptually uniform color spaces for color texture analysis: an empirical evaluation. IEEE Transactions on Image Processing 10(6): 923–937.

Patel D & Stonham T (1991) A single layer neural network for texture discrimination. Proc. IEEE Int. Symposium on Circuits and Systems, pp 2657–2660.

Patel D & Stonham T (1992) Texture image classification and segmentation usin rank-order clustering. Proc. 11th International Conference on Pattern Recognition, Hague, Netherlands, 3: 92–95.

Pavlidis T (1992) Why progress in machine vision is so low. Pattern Recognition Letters 13: 221–225.

Pentland A, Picard R & Sclaroff S (1996) Photobook: content- based manipulation of image databases. International Journal of Computer Vision 18: 233–254.

Phillips P & Bowyer K (1999) Empirical evaluation of computer vision algorithms. IEEE Transactions on Pattern Analysis and Machine Intelligence 21: 289–290.

Pietikäinen M, Nurmela T, Mäenpää T & Turtinen M (2003) View-based recognition of 3-D textured surfaces. Proc. 12th International Conference on Image Analysis and Processing, Mantova, Italy. Accepted.

Pietikäinen M & Ojala T (1996) Texture analysis in industrial applications. In: Sanz J (ed) Image Technology – Advances in Image Processing, Multimedia and Machine Vision, pp 337–359. Springer-Verlag.

Pietikäinen M, Ojala T, Nisula J & Heikkinen J (1994) Experiments witih two industrial problems using texture classification based on feature distributions. Proc. SPIE Vol. 2354 Intelligent Robots and Computer Vision XII: 3D Vision, Product Inspection, and Active Vision, pp 197–204.

Pietikäinen M, Ojala T & Xu Z (2000) Rotation invariant texture classification using feature distributions. Pattern Recognition 33: 43–52.

Pietikäinen M & Rosenfeld A (1981) Image segmentation by texture using pyramid node linking. IEEE Transactions on Systems, Man, and Cybernetics 11: 822–825.

Poirson B & Wandell B (1996) Pattern-color separable pathways predict sensitivity to simple colored patterns. Vision Research 36(4): 515–526.

Porter R & Canagarajah N (1997) Robust rotation-invariant texture classification: wavelet Gabor filter and GMRF based schemes. IEE Proceedings on Vision, Image, and Signal Processing 144: 180–188.

Price K (1986) Anything you can do, i can do better (no you can't). Computer Vision, Graphics, and Image Processing 36: 387–391.

Pudil P, Novovičová J & Kittler J (1994) Floating search methods in feature selection. Pattern Recognition Letters 15: 1119–1125.

Puzicha J, Buhmann J, Rubner Y & Tomasi C (1999) Empirical evaluation of dissimilarity for color and texture. Proc. 7th International Conference on Computer Vision, Kerkyra, Greece, pp 1165–1172.

Randen T & Husoy J (1999) Filtering for texture classification: a comparative study. IEEE Transactions on Pattern Analysis and Machine Intelligence 21(4): 291–310.

Rao A & Lohse G (1993) Identifying high-level features of texture perception. CVGIP: Graph. Models Image Process. 57(3): 218–233.

Rosenfeld A, Ye-Wang C & Wu A (1982) Multispectral texture. IEEE Transactions on Systems, Man, and Cybernetics 12(1): 79–84.

Rubio T, Bandera A, Urdiales C & Sandoval F (2002) A hierarchical context-based textured image segmentation algorithm for aerial images. Proc. 2nd International Workshop on Texture Analysis and Synthesis, Copenhagen, Denmark, pp 117–121.

Sánchez-Yáñez R, Kurmyshev E & Cuevas F (2003) A framework for texture classification using the coordinated clusters representation. Pattern Recognition Letters 24: 21–31.

Schaefer G (2001) JPEG compressed domain image retrieval by colour and texture. Proc. Data Compression Conference DCC '01, Snowbird, USA.

Schael M (2002) Invariant texture classification using group averaging with relational kernel functions. Proc. 2nd International Workshop on Texture Analysis and Synthesis, Copenhagen, Denmark, pp 129–133.

Shapiro L & Stockman G (2001) Computer Vision. Prentice-Hall, Upper Saddle River, NJ.

Shi J & Malik J (2000) Normalized cuts and image segmentation. IEEE Transactions on Pattern Analysis and Machine Intelligence 22(8): 888–905.

Sikora T (2001) The MPEG-7 visual standard for content description — and overview.

IEEE Transactions on Circuits and Systems for Video Technology 11(6): 696–702.

Silvén O (2000) Applying texture analysis to industrial inspection. In: Pietikäinen M (ed) Texture Analysis in Machine Vision, pp 207–217. World Scientific.

Silvén O, Niskanen M & Kauppinen H (2003) Wood inspection with non-supervised clustering. Machine Vision and Applications. In press.

Smith G & Burns I (1997) Measuring texture classification algorithms. Pattern Recognition Letters 18: 1495–1501. Http://www.cssip.uq.edu.au/staff/meastex/meastex.html.

Sokal R & Rohlf F (1969) Biometry. W.H. Freeman.

Song K, Petrou M & Kittler J (1992) Texture defect detection: a review. Proc. SPIE Vol. 1708 Applications of Artificial Intelligence X: Machine Vision and Robotics, pp 99–106.

Soriano M, Marcos S, Quibilan M, Alino P & Saloma C (2001) Image classification of coral reef components from underwater color video. Proc. OCEANS 2001, Honolulu, Hawaii, pp 1008–1013.

Tamura H, Mori S & Yamawaki T (1978) Textural features corresponding to visual perception. IEEE Transactions on Systems, Man, and Cybernetics 8: 460–473.

Tan T & Kittler J (1993) Colour texture classification using features from colour histogram. Proc. 8th Scandinavian Conference on Image Analysis, Tromso, Norge, 2: 807–813.

Tao B & Dickinson B (1996) Image retrieval and pattern recognition. Proc. SPIE Multimedia Storage and Archiving Systems, 2916: 130–139.

Theodoridis S & Koutroumbas K (1999) Pattern Recognition. Academic Press, San Diego, CA.

Tomita F & Tsuji S (1990) Computer Analysis of Visual Textures. Kluwer.

Tuceryan M & Jain A (1999) Texture analysis. In: Chen C, Pau L & Wang P (eds) Handbook of Pattern Recognition and Computer Vision, pp 207–248. World Scientific, Singapore, 2nd edition.

Turtinen M, Pietikäinen M, Silvén O, Mäenpää T & Niskanen M (2003a) Paper characterization by texture using visualization-based training. International Journal of Advanced Manufacturing Technology. In press.

Turtinen M, Pietikäinen M, Silvén O, Mäenpää T & Niskanen M (2003b) Texture-based paper characterization using non-supervised clustering. Proc. 6th International Conference on Quality Control by Artificial Vision, Gatlinburg, Tennessee, USA. In press.

Unser M (1986) Sum and difference histograms for texture classification. IEEE Transactions on Pattern Analysis and Machine Intelligence 8(1): 118–125.

Valkealahti K & Oja E (1998) Reduced multidimensional cooccurrence histograms in texture classification. IEEE Transactions on Pattern Analysis and Machine Intelligence 20: 90–94.

Varma M & Zisserman A (2002) Classifying materials from images: to cluster or not to cluster. Proc. 2nd International Workshop on Texture Analysis and Synthesis, pp 139–143.

Wandell B (1995) Foundations of Vision. Sinauer Associates, Sunderland, Mass.

Wang L & He DC (1990) Texture classification using texture spectrum. Pattern Recognition 23: 905–910.

Wang P, Krishnan S, Kugean C & Tjoa M (2001) Classification of endoscopic images based on texture and neural network. Proc. 23rd Annual EMBS International Conference, Istanbul, Turkey, pp 3691–3695.

Weszka J, Dyer C & Rosenfeld A (1976) A comparative study of texture measures for terrain classification. IEEE Transactions on Systems, Man, and Cybernetics 6: 269–285.

Wolfram S (1983) Statistical mechanics of cellular automata. Reviews of Modern Physics 55: 601–644.

Wolfram S (2002) A New Kind of Science. Wolfram Media, Inc., Champaign, IL, USA.

Woods J (1972) Two dimensional discrete markov random fields. IEEE Transactions on Information Theory 18: 232–240.

Xu Z, Pietikäinen M & Ojala T (1997) Defect classification by texture in steel surface inspection. Proc. International Conference on Quality Control by Artificial Vision, Le Creusot, Burgundy, France, pp 179–184.

Yamada A, Pickering M, Jeannin S, Cieplinski L, Ohm J & Kim M (2001) MPEG-7 visual part of experimentation model version 9.0. ISO/IEC JTC1/SC29/WG11 N3914.

Yao CH & Chen SY (2003) Retrieval of translated, rotated and scaled color textures. Pattern Recognition 36: 913–929.

Zhu S & Yuille A (1996) Region competition: unifying snakes, region growing and Bayes/MDL for multiband image segmentation. IEEE Transactions on Pattern Analysis and Machine Intelligence 18(9): 884–900.

Zongker D & Jain A (1996) Algorithms for feature selection: an evaluation. Proc. 13th International Conference on Pattern Recognition, Vienna, Austria, 2: 18–22.