Multi-Resolution Image Processing and Learning for Texture Recognition and Image Enhancement

Thesis by
Hayit Greenspan

In Partial Fulfillment of the Requirements
for the Degree of
Doctor of Philosophy

California Institute of Technology
Pasadena, California

1994

(Defended May 20, 1994)

Publication History:

- Thesis approved by defense committee (defense May 20, 1994):

  Dr. Rodney Goodman (Chair)
  Dr. Charles Anderson
  Dr. Christof Koch
  Dr. Pietro Perona
  Dr. Dimitri Psaltis

- Final thesis delivered to Graduate Office May 27, 1994

Acknowledgements

This work benefited from a collaboration with many people. First I would like to thank my advisor, Prof. Rodney Goodman, for giving me the opportunity to come to Caltech for the Ph.D. and for providing continuous support and guidance throughout the years. I especially appreciate the freedom that he has given me to pursue the variety of my interests, and to collaborate with many people on the way. This was a crucial element in the ability to combine research fields, which forms the basis for this thesis.

Special warm thanks to Dr. Charlie Anderson with whom I have interacted for several years while he was working at the Jet Propulsion Laboratory. It is Charlie who first introduced me to the field of pyramid image representations. We have had many interesting discussions on pyramids and texture, and it is through these discussions that the idea for the image enhancement scheme was conceived.

My main collaboration recently was with Prof. Pietro Perona. I would like to thank him for his general support as well as for sharing his expertise in vision, and specifically filtering. It is through this interaction that the pyramid steerability was established, thus augmenting one of the major building blocks of the proposed system.

Many thanks to all the members of the defense committee for their interest in my work and their valuable comments.

To Dr. Padhraic Smyth, many thanks for the stimulating discussions and helpful advice throughout the years, including the review of the thesis. Also, for the support in organizing the related NIPS workshop on "Learning in Computer Vision."

I would also like to thank Dr. Chellappa with whom I have interacted while he was at USC.

To Serge Belongie - much appreciation for being such a great SURF student (Summer Undergraduate Research Fellow) and for continuing the research during the school year. Thanks for becoming such an active, helpful and enjoyable partner in the work! To my colleagues Subrata Rakshit and Ming-Chieh Lee - thanks for working together on many interesting issues, some of which are incorporated in the presented work.

For their companionship and continuous support, I would like to thank all the members of the MicroSystems Group, both past and present. Special thanks to Chuck Higgins and

Bhusan Gupta for helpful discussions, Chris Ulmer for the guidance on the many updates of the ITRULE program, and Jeff Dickson for his valuable LaTeXsupport. More special thanks to Bob Freeman for his on-line support as our very friendly system manager, and bearing with me through some of the more stressful times. Finally, to Bette Linn, our devoted secretary, for all the help throughout the years.

I would like to take a special moment to acknowledge Prof. Edward Posner's support and encouragement throughout my years at Caltech. I have always found him willing to address any technical or personal problem. Throughout the years he has also professionally collaborated with my husband and has become a family friend. Dr. Posner's premature death is a great loss to us all. I deeply regret that I can not share with him these finishing moments and continue a collaboration in the future as we had planned.

To my parents, I am forever grateful for your love and support. To my son Dror, thanks for becoming a part of my life, with so much joy. And finally, many many thanks to my husband Moshe. I cherish the many helpful discussions that we have had on the way, both technical and otherwise. Your continuous support and encouragement throughout the years have made this work possible.

This work was supported by an Intel graduate fellowship (many thanks to my mentor there, Dr. Dan Seligson); by the Army Research Office under the contract DAAL03-89-K-0126; by ARPA and ONR under grant no. N00014-92-J-1860 and grant no. AFOSR-90-0199, and partly by Pacific Bell.

Abstract

A general recognition framework is presented that consists of multi-resolution pyramidal feature-extraction and learning paradigms for classification. The system is presented in the context of the texture recognition task.

In the feature extraction part of the system, an oriented Laplacian pyramid is used as an efficient filtering scheme to transform the input image to a more robust representation in the frequency and orientation space. An optimal technique is presented for computing a steerable representation of the pyramid. Steerability is used to generate a rotation-invariant input representation.

In the learning stage of the system we focus on a rule-based probabilistic learning scheme. This information-theoretic technique is utilized to find the most informative correlations between the attributes and the output classes while producing probability estimates for the outputs. Both unsupervised and supervised learning are utilized. Apart from the rule-based approach we experiment with other non-parametric classifiers, such as the k-nearest neighbor classifier and the Backprop neural-network.

We demonstrate experimentally that our scheme improves significantly upon the state-of-the-art both in rotation-invariant classification and in orientation estimation. A variety of applications are presented, including autonomous navigation scenarios and remote-sensing, as possible extensions for the texture recognition system. A generalization of the system to face-recognition is discussed.

In the latter part of the thesis, a procedure for creating images with higher resolution than the sampling rate would allow is described. The enhancement algorithm augments the frequency content of the image by using a non-linearity that generates phase-coherent higher harmonics. The procedure utilizes the Laplacian pyramid image representation. Results are presented depicting the power-spectra augmentation and the visual enhancement of several images. Simplicity of computations and ease of implementation allow for real-time applications such as high-definition television (HDTV). An initial investigation is pursued to combine the enhancement scheme with pyramid coding schemes.

Table of Contents

List of Figures

List of Tables

Chapter 1
Introduction

MULTI-RESOLUTION image processing, specifically the pyramid image representation, is proving to be valuable to the image processing community, with applications in a wide variety of domains, such as edge-detection [PM90], texture recognition [Tur86, ACBG90, KG83], motion [AB85, Hee87], stereo and more. Two major advantages can be associated with the pyramid representation:

- The pyramid provides for an hierarchical image processing scheme which enables the image analysis to be pursued sequentially from low to high-resolution versions of the image. This enables efficient image analysis systems. [1]

- The pyramid can be used as a computationally efficient filtering mechanism to shift the input representation (of pixels) to a representation in frequency and orientation space. There is both biological and computational evidence supporting the use of a bank of orientation-selective bandpass filters to achieve this shift as the front end of many image-processing tasks.

The multiresolution analysis of images is the common theme throughout this thesis. In the major part of the work we utilize the pyramid as an efficient filtering scheme and investigate the pyramid filters characteristics as related to image analysis tasks; specifically the texture recognition task. In the latter part of the thesis we use the multiresolution representation to investigate the behavior of images features (specifically image edges) across scale. We use this to an advantage in a new image-enhancement algorithm.

LEARNING in computer-vision and image-understanding has gained an increasing amount of interest recently, both from researchers in the learning community and from researchers involved with the computer vision world. The field is characterized by a shift away from the classical, purely model-based computer vision techniques, towards data-driven learning paradigms for solving real-world vision problems. Classical computer-vision techniques have, to a large extent, neglected learning, which is an important component for robust and flexible vision systems. Meanwhile, there is real-world demand for automated image

[1] Real-time implementations are now starting to emerge [van91].

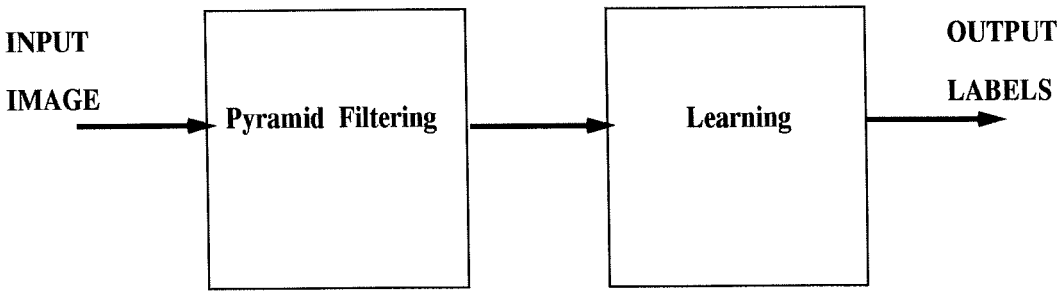


Figure 1.1: A general recognition framework

handling for scientific and commercial purposes, and a growing need for automated image understanding and recognition, in which learning can play a key role. Some of the applications include difficult recognition tasks, remote-sensing imagery analysis, automated inspection, autonomous navigation systems which use vision as part of their sensors, and the field of automated imagery data-base analysis.

In the major part of the thesis we concentrate on a general recognition system which combines pyramid filtering of the input image with learning schemes for classification. These two building blocks are presented in Fig. 1.1. We are interested in both a good preprocessing stage and in the following up classification schemes.

IMAGE CODING is just recently being combined with image processing for advanced image compression schemes (so called "second generation" image coding). This is a second research topic which is shortly discussed in this thesis. In the growing field of multimedia applications, image coding is shifting from classical compression schemes, which are based on sampling the image on fixed tiling, to adaptive coding where coding schemes and image partition is done locally with the help of a first stage of image analysis (see Fig. 1.2).

In the latter part of the thesis we use the pyramid concepts to an advantage in an image enhancement scheme. An investigation into combining the enhancement scheme with pyramid coding schemes is initiated.

Due to the variety of topics covered in this work, we choose to introduce the underlying concepts and related literature, within each chapter, as we go along. A brief outline of the thesis chapters follows.

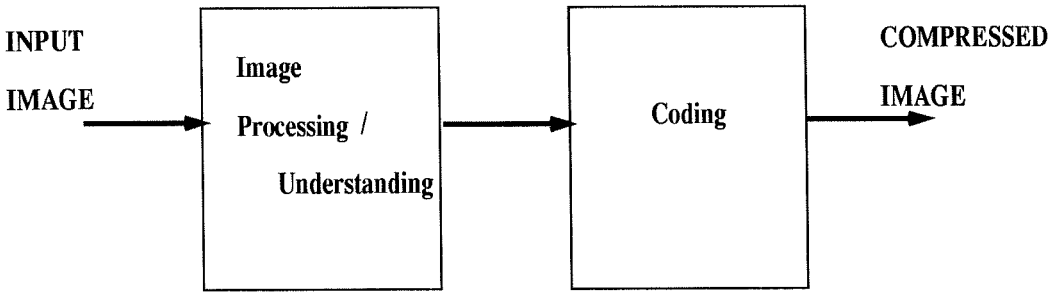


Figure 1.2: Combining image processing with image compression

1.1 Thesis Outline

In the main part of the thesis we present a general recognition framework that combines multiresolution pyramidal feature-extraction with learning techniques. The system is presented in the context of the texture recognition task. In **Chapter 2** the multiresolution pyramid filtering scheme is introduced, and extended, to include an oriented pyramid. This pyramid allows for a computationally efficient filtering scheme to transform the input image (pixels) to a more robust representation in the orientation and frequency space. We prove that the defined oriented pyramid spans the orientation space. The recognition system is described in **Chapter 3**, within the context of the texture recognition task. We start by introducing the texture analysis challenge, together with reviewing some of the many texture classification approaches available in the literature. We then introduce a rule-based probabilistic learning scheme. This information-theoretic technique is utilized to find the most informative correlations between the attributes and the output classes (in the form of "readable" rules) while producing probability estimates for the outputs. We focus on the rule-based approach and investigate some of its characteristics as well as utilize other non-parametric classifiers, including the k-nearest neighbor classifier and the Backprop neural-network. The recognition system is described and analyzed. State-of-the-art texture classification results on large texture databases are shown. The texture recognition system is extended upon in the following chapters. In **Chapter 4** we prove that the oriented pyramid is steerable. We present an optimal technique for deriving the set of interpolation functions ("steering coefficients") for a given overcomplete discrete representation, which enable the shift to a steerable representation. We then show how the oriented

pyramid, which is 8/3 redundant (and thus more compact than other pyramids previously used in the literature) can have the property of steerability. The steerability characteristic enables a transition to a rotation-invariant input representation. State-of-the-art results for rotation-invariant recognition, as well as high-accuracy results in detecting the orientation angle are presented in **Chapter 5**. Finally, **Chapter 6** presents a collection of possible future applications for the texture recognition system. These include autonomous navigation scenarios and remote-sensing imagery analysis. Texture is shown to have a role in these application domains. Learning is also advantageous, as it provides an adaptive system, which is robust to noise and changing environments, as well as provides a means for automated rule generation in the classification tasks. Additionally, we propose that the recognition system can be generalized to other recognition domains. Initial experiments in the shape-recognition arena, specifically face-recognition, are presented.

We conclude the thesis with an additional field of image-processing in which we have found the pyramid representation to be advantageous. The application of the multiresolution image representation to image enhancement and coding is covered in **Chapter 7**. First, a new enhancement scheme is described, which is based on the Laplacian pyramid. We conclude with an attempt to combine the enhancement scheme with pyramid coding schemes and present some initial results in this challenging new field.

Chapter 2
Multiresolution Image Processing

2.1 Introduction

Multiresolution image representations based on pyramid or wavelet representations are the topic of much recent research, towards a compact image representation space and an efficient basis for image analysis schemes. Wavelets are complete and orthonormal representations, and thus can be advantageous in image data compression [Mal89]. The pyramid representation, which is a more redundant representation within the wavelet family, allows for a computationally efficient filtering scheme and provides a framework for fast computation of image measures, and for implementing coarse-to-fine analysis in many application domains, including motion, stereo, pattern-matching and more.

In this work, the pyramid filtering scheme is utilized to shift from the pixel representation of the input image to a more informative and robust representation in the orientation and frequency space. There is both biological and computational evidence supporting the use of a bank of orientation-selective bandpass filters, such as the Gabor filters, to achieve this shift as an initial phase of many image-processing tasks. These tasks include edge-detection [Can86, PM90], texture recognition [Tur86, ACBG90, KG83], motion-detection [AB85, Hee87] and more. Orientation and frequency responses are extracted from local windows of the input image and the statistics of the coefficients characterizing each window form a local representative feature vector. In the application domains listed above, we are interested in utilizing the extracted feature vectors as an intermediate step towards orientation analysis, or other higher-level analysis. Our constraints are therefore in computational efficiency and memory requirements (especially important for real-world applications), as opposed to achieving a complete self-inverting representation which is important for coding and reconstruction purposes. It is this distinction which motivates us into using the Oriented Laplacian pyramid, described below, which is computationally efficient and compact.

We start this chapter with a short review of the log-Gabor filters (Section 2.2). We then proceed to describe the pyramid scheme, as a computationally efficient filtering strategy (Section 2.3). We review the classical Burt pyramid, and its variation, the FSD pyramid,

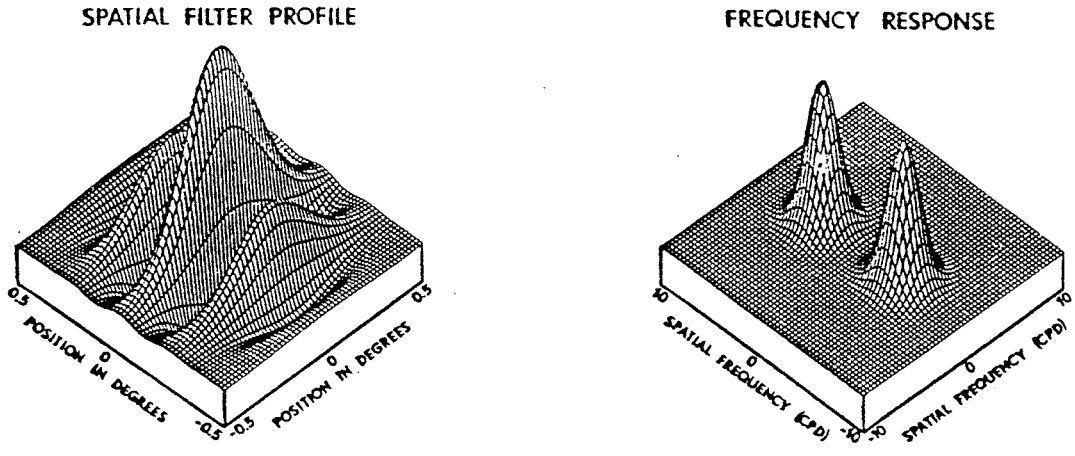


Figure 2.1: log-Gabor filters

in sections 2.3.1 and 2.3.2, respectively. We extend on the classical pyramids to incorporate oriented filtering, in what we term the Oriented Laplacian pyramid. This is presented in Section 2.3.3. Finally, we review some of the oriented pyramid characteristics in Section 2.3.4. These include good low-band rejection and computational efficiency. In our final analysis we use the singular-value decomposition to prove that the oriented pyramid that we have defined, with 4 oriented components per scale, spans the orientation space. Section 2.4 summarizes this chapter and motivates the use of the pyramid as a filtering strategy, to shift an input pixel representation to a more robust representation, in the orientation and frequency domains. This preprocessing step will be utilized in the following chapters.

2.2 The Log-Gabor Filters

Gabor functions are Gaussians modulated by complex sinusoids. In its general form the 2-D Gabor function and its Fourier transform can be written as:

$$g(x,y;u_0,v_0) = e^{-(x^2/2\sigma_x^2+y^2/2\sigma_y^2)+2\pi i(u_0x+v_0y)} \; ; \; G(u,v) = e^{-2\pi^2(\sigma_x^2(u-u_0)^2+\sigma_y^2(v-v_0)^2)} \quad (2.1)$$

where σ_x and σ_y define the widths of the Gaussian in the spatial domain and (u_0,v_0) is the frequency of the complex sinusoid. The above functions are schematically shown in Fig. 2.1.

The biological motivation for these filters lies in their goodness of fit to the receptive-field profiles of simple cells in the striate cortex. Computationally, the Gabor filters have received much attention as they achieve optimal joint resolution in both space and spatial frequency [Dau85]. The Gabor functions from a complete but non-orthogonal basis set and any given function can be expanded in terms of these basis functions. Such an expansion provides a localized frequency description and has been used in image compression.

In order to optimally detect and localize features at various *scales*, filters with varying support rather than a fixed one are required. The *log-Gabor* functions are complex sinusoidal gratings modulated by 2-D Gaussian functions in the space domain, and shifted Gaussians in the spatial frequency domain, where the variance of the Gaussian *scales with* the spatial frequency of the sine wave. The orientation space is discretized into equidistant intervals while the frequency bands are distributed in octave steps, such that the radial bandwidth doubles for each next high-frequency band. This octave-like organization is advocated in the case of natural images [Fie87].

The log-Gabor filters form a family of self-similar filter profiles. In this respect they are related to the wavelet representation. Wavelets are families of basis functions obtained through dilations and translations of a basic wavelet and such a decomposition provides a compact data structure for representing information. Here the basic wavelet is a Gabor function and hence we refer to this decomposition as the Gabor wavelet decomposition. The log-Gabor filters form a complete but non-orthogonal basis set for the wavelet decomposition [Woo91].

2.3 The Pyramid Scheme

The use of multiresolution pyramid techniques is rapidly becoming a standard in many areas of image processing and computer vision. The pyramid provides both a compact image representation and a structure in which to implement efficient analysis algorithms.

In a pyramid representation the original image is decomposed into sets of low-pass and band-pass components via Gaussian and Laplacian pyramids, respectively. The Gaussian pyramid consists of low-pass filtered (LPF) versions of the input image, with each stage

of the pyramid computed by low-pass filtering of the previous stage and corresponding subsampling of the filtered output. The Laplacian pyramid consists of band-pass filtered (BPF) versions of the input image, with each stage of the pyramid constructed by the subtraction of two corresponding adjacent levels of the Gaussian pyramid.

The history of multiresolution representations of images began in a serious manner with Burt and Adelson [BA83]. Several variations have been introduced to the Burt and Adelson pyramid in the last several years. In this work we introduce one such variation (the *FSD pyramid* [And87]), extend to form an oriented pyramid (so called the Oriented Laplacian pyramid) and investigate this pyramid filters' characteristics.

2.3.1 The Burt and Adelson Pyramid

The Burt and Adelson pyramid (otherwise termed *Reduce Expand* pyramid) consists of the following set of operations (Figure 2.2):

$$\begin{aligned} G_{n+1} &= Reduce(G_n) \\ L_n &= G_n - Expand(G_{n+1}). \end{aligned} \tag{2.2}$$

The *Reduce* operation consists of low-pass filtering followed by subsampling the filtered image (removing every other pixel and every other line to produce an image half as big as the original in each dimension). The *Expand* operation involves creating from the reduced image an image the size of the original via interpolation. (e.g., reinserting the missing pixels with zeroes, multiplying by four and then low-pass filtering). The above loop continues until a final image G_{n+1} is created which is nominally around 8 by 8 pixels in size.

The sequence of low-pass images, G_n, is called the *Gaussian* Pyramid and the sequence of bandpass images, L_n, is called the *Laplacian* Pyramid. The final lowpass image, G_{n+1}, plus the set of bandpass images L_n form an overcomplete representation of the image. The pyramid is complete as it allows for an exact reconstruction of the original image:

$$G_n = L_n + Expand(G_{n+1}). \tag{2.3}$$

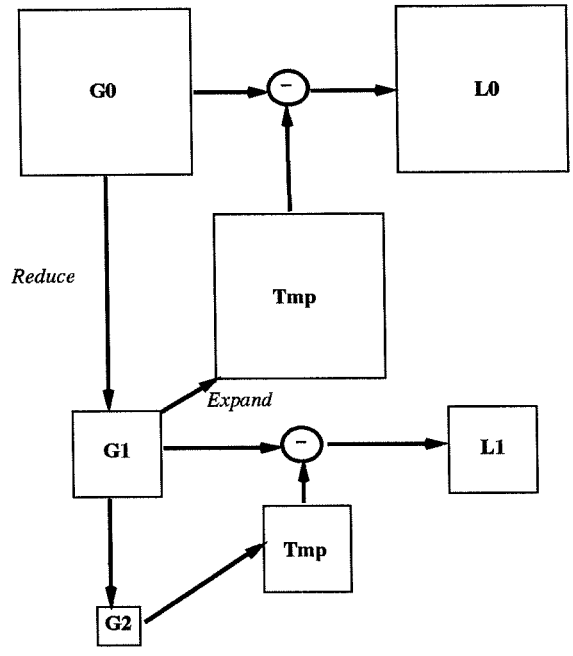


Figure 2.2: The Burt and Adelson pyramid

The pyramids are overcomplete in the sense that the total number of stored values is 4/3 the number of pixels in the original image (see Section 2.3.4).

2.3.2 The FSD Pyramid

A modified form of the Burt and Adelson pyramid, the FSD (Filter Subtract and Decimate) pyramid, was suggested by Anderson [And87]. It follows the following set of operations (see Figure 2.3):

$$G^0_{n+1} = W * G_n \; ; \; L_n = G_n - G^0_{n+1}$$
$$G_{n+1} = \text{Subsampled } G^0_{n+1}. \tag{2.4}$$

The low-pass filter (LPF), W, is Gaussian in shape, normalized to have its coefficients sum to 1. The values used in this work for W, which is a 5-sample separable filter, are (1/16, 1/4, 3/8, 1/4, 1/16).

The slight modification in the pyramid generation allows for several interesting filtering characteristics. First, the order of operations is such that the subtraction occurs before the decimation step, ensuring that aliasing does not get incorporated into the mid-band

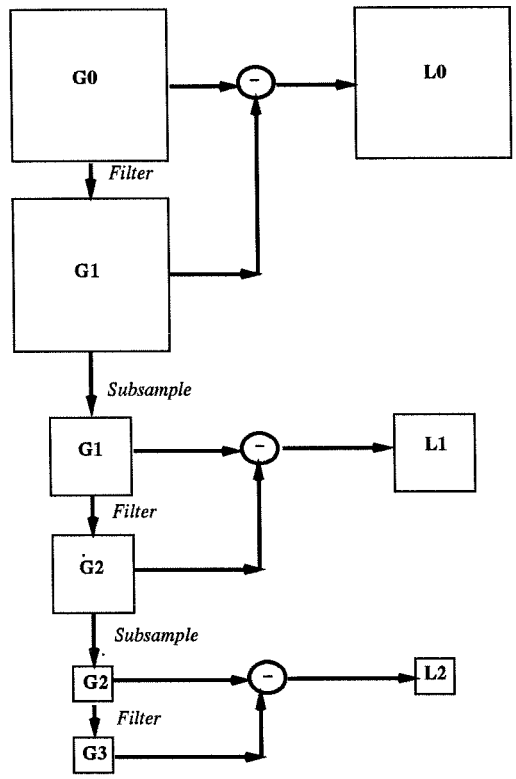


Figure 2.3: The FSD pyramid

regions of the bandpass images, L_n, and overall providing narrower bandpass characteristics. Second, the hardware implementation of the Gaussian and Laplacian pyramids is simplified since the data flow is simpler in a pipeline architecture and only one convolution element is required [And87]. In the remainder of this work the FSD pyramid is used as the initial image representation.

In order to extract the orientationally tuned bandpass filtering responses, the oriented pyramid is formed next. A computationally efficient scheme allows for the generation of the oriented pyramid based on the already existing Laplacian pyramid. In the remainder of this chapter, the generation of the oriented pyramid and its characteristics will be addressed.

2.3.3 The Oriented Laplacian Pyramid

The Oriented pyramid [1] is the result of modulating each level of the Laplacian pyramid with a set of oriented sine waves, followed by another LPF operation using a separable filter, and corresponding subsampling, as defined in equation 2.5:

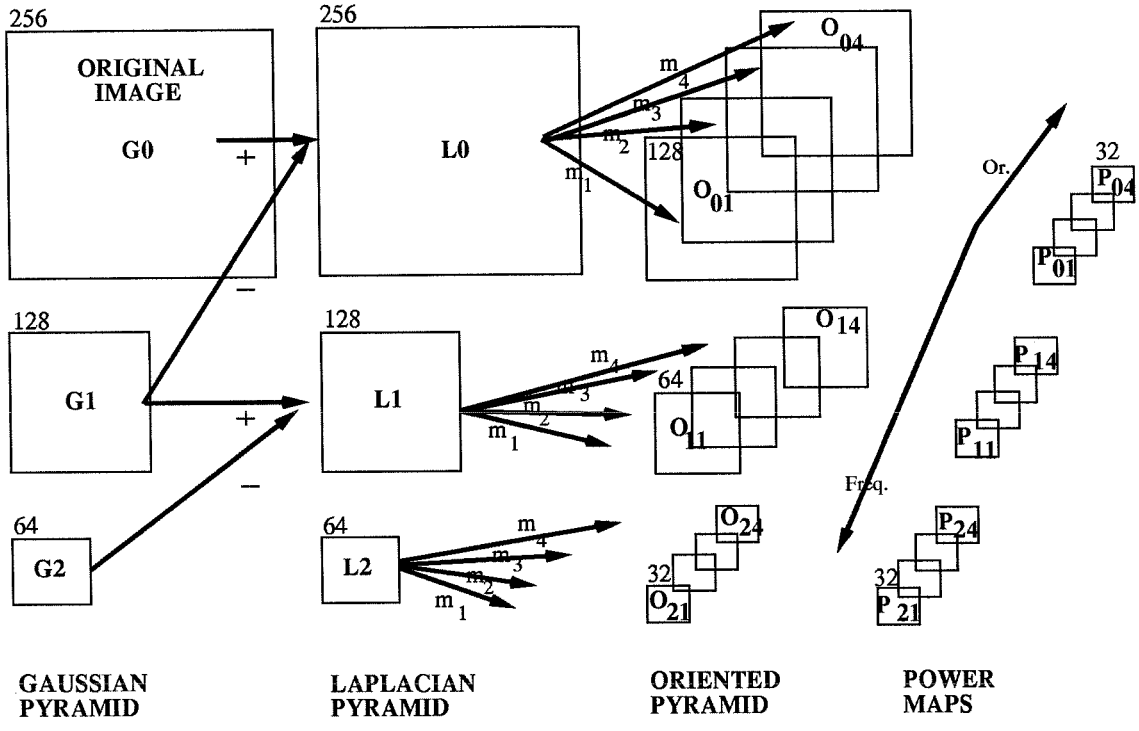
$$O_{n\alpha} = LPF[e^{(i\vec{k_\alpha}\cdot\vec{r})}L_n[x,y]], \tag{2.5}$$

where $O_{n\alpha}$ is the oriented image at scale n and orientation α, $\vec{r} = x\vec{\imath} + y\vec{\jmath}$ (x and y are the spatial coordinates of the Laplacian image), $\vec{k}_\alpha = (\pi/2)[\cos\theta_\alpha\vec{\imath} + \sin\theta_\alpha\vec{\jmath}]$ and $\theta_\alpha = (\pi/N)(\alpha - 1)$; $(\alpha = 1...N)$. For future reference, we term the pyramid an *ori5* pyramid when using a 5-tap filter as the LPF in (2.5). Using a 3-tap filter $(1/4, 1/2, 1/4)$ or a 7-tap filter
$(1/64, 6/64, 15/64, 20/64, 15/64, 6/64, 1/64)$, we get *ori3* and *ori7* pyramids, respectively. Unless otherwise noted we use the 5-tap LPF case.

In this work we use 4 oriented components ($N = 4$) . From equation 2.5, each level (n) of the pyramid is thus modulated by the following complex sinusoids where x and y are indices of the Laplacian image to be modulated:

$$m_1(x,y) = e^{i(\pi/2)x} \; ; \; m_2(x,y) = e^{i(\pi\sqrt{2}/4)(x+y)}$$
$$m_3(x,y) = e^{i(\pi/2)y} \; ; \; m_4(x,y) = e^{i(\pi\sqrt{2}/4)(y-x)}. \tag{2.6}$$

[1] Throughout the thesis we will use the terms *Oriented pyramid* and *Oriented Laplacian pyramid* interchangeably.



PYRAMID IMAGE REPRESENTATION

Figure 2.4: Block diagram of the oriented pyramid generation. The input image is represented via the Gaussian, Laplacian, and Oriented pyramids. The power maps represent the local statistics of the oriented pyramid's coefficients, which characterize the image local-area response to the different orientations and frequencies (see Section 3.3.2).

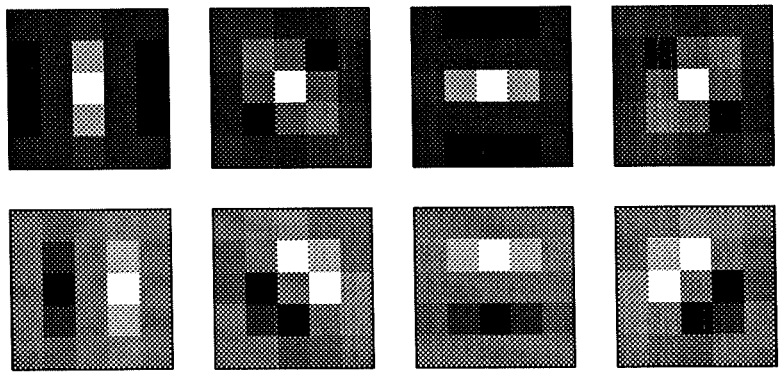


Figure 2.5: A set of oriented pyramid filters, $O_{n\alpha}$. Real and imaginary components are presented, top and bottom, respectively, for $n = 0$ and $\alpha = 1..4$.

These four modulators differ only in their orientations, which are $0°, 45°, 90°$ or $135°$ for m_1 through m_4, respectively. The origin of x and y is taken to be the center of the image being modulated. Note that the modulating frequency remains constant for each level of the pyramid.

After modulation, the Laplacian images are lowpass filtered and subsampled. At this point, the Laplacian images have effectively been filtered by the following set of log-Gabor filters:

$$\begin{aligned}
\psi_1(x,y) &= \frac{1}{2\pi} e^{-(x^2+y^2)/2+i(\pi/2)x} \\
\psi_2(x,y) &= \frac{1}{2\pi} e^{-(x^2+y^2)/2+i(\pi\sqrt{2}/4)(x+y)} \\
\psi_3(x,y) &= \frac{1}{2\pi} e^{-(x^2+y^2)/2+i(\pi/2)y} \\
\psi_4(x,y) &= \frac{1}{2\pi} e^{-(x^2+y^2)/2+i(\pi\sqrt{2}/4)(y-x)}
\end{aligned} \tag{2.7}$$

and then subsampled. Fig. 2.4 shows a block-diagram of the orientation-pyramid generation. A set of oriented-pyramid filters are displayed in Fig. 2.5

2.3.4 The Pyramid Filters' Characteristics

The filtering operation in equation 2.5 is not the standard one found in the literature. Usually, the original image is filtered with a set of oriented sinewave modulated Gaussian filters (as in equation 2.1). This straightforward approach, which is most commonly used, has two main problems. First, the real, or cosine component of each filter has a nonzero

value at 0 in the frequency domain. Given the $(1/f)^2$ power spectrum of natural images [Fie87], even a slight DC component can bias all filter outputs. In equation 2.5 the oriented filters are applied to the *bandpass* images of the pyramid, L_n. This ensures good low-frequency rejection. The second problem is a computational one. The diagonally oriented filters are non-separable (computational complexity scales as the filter size squared rather than linearly). In order to accomplish filtering at diagonal directions using separable filters, a reversal in the order of operations is performed (the image is first modulated by a sinewave and then LPFed, rather than modulating the LPF prior to convolving with the image). This reversal gives us separable filters, and it therefore allows for a computationally efficient filtering scheme.

We next investigate the redundancy of the generated pyramid. The redundancy in the nonoriented Laplacian pyramid representation is 4/3 . The sizes of the Laplacian bands go as $N^2, N^2/4, N^2/16...$ The sum of the series is $4/3N^2$. In the pyramid scheme defined above, we use four complex oriented filters to create *eight* oriented bandpass components from each nonoriented Laplacian level. The eight include the real and imaginary response maps from each complex oriented filter modulation. Since this involves lowpass filtering after the modulation, it is possible to subsample these oriented bands by a factor of 2 in each dimension. We thus create eight bands, each 1/4 of the size of the original nonoriented Laplacian. From each band of size $M \times M$, we hold 8 bands of size $M^2/4$. The total number of pixels at each level therefore increases from M^2 to $(M^2 \times 8)/4 = M^2 \times 2$ leading to an increase of redundancy by a factor of two. Overall, the redundancy of our oriented pyramid is $4/3 \times 2 = 8/3$. This pyramid is more compact than other oriented pyramids described in the literature which usually exhibit 16/3 redundancy [FA91, SFAH92, Per91]. The tradeoff to the above savings is that the pyramid formation is not self inverting due to the decimation which allows us to decrease the redundancy. Also, the filter kernels are not designed to guarantee a *perfectly* flat power spectrum across orientation. However, as we shall show in a later chapter (see Chapter 4), the pyramid can be shown to be steerable to a good approximation and good results are achieved for rotation invariant texture recognition. The lack of self inversion is not a problem since these oriented pyramids are utilized for information processing rather than image coding.

We conclude this chapter with an investigation into the independence characteristics

of the pyramid filter set. We show that the oriented pyramid as defined in the preceding section, with the selected oriented components of 45° bandwidth, spans the orientation space.

Following the work of Perona [Per91], we make use of the singular-value decomposition (SVD) to investigate the independence of the set of oriented pyramid kernels (as in equation 2.5). This procedure consists of the following steps:

- Generate 360 oriented pyramid kernels (at a single scale) via equation 2.5 with $N = 360$.
- Concatenate each of these 360 kernel matrices into column vectors, and combine these column vectors to form a large matrix, $\mathbf{A}$.
- Perform the SVD by finding the matrices $\mathbf{U}$, $\mathbf{V}$, and diagonal matrix Σ such that

$$\mathbf{A} = \mathbf{U}\Sigma\mathbf{V}^T. \tag{2.8}$$

The diagonal matrix Σ contains the square roots of the positive eigenvalues of $\mathbf{A}^T\mathbf{A}$. The number of nonzero eigenvalues in Σ is equal to the number of linearly independent column vectors in $\mathbf{A}$. Upon inspecting the results of the SVD, the first seven singular values, $\sigma_1..\sigma_7$, in Σ contain approximately 99.5% of the sum of all the singular values ($\sum \sigma_i$). This is shown in Fig. 2.6. The above result indicates that a set of eight filters, i.e., an orientation bandwidth of 45°, is sufficient to span the 360° of orientation space with more than 99% accuracy. The four filters, O_{n1} through O_{n4}, and their conjugate counterparts, which we hereon term O_{n5} through O_{n8}, satisfy this requirement. The chosen set of kernels are shown in Fig. 2.5. The filters' combined power spectra covers the 360° orientation space, as can be seen in Fig. 2.7.

2.4 Summary

In this chapter we have reviewed the concepts behind the pyramid multi-resolution image representation and processing. We extended on the Gaussian and Laplacian pyramids to define an Oriented Laplacian pyramid which allows for a computationally efficient log-Gabor filtering scheme. We have analyzed the defined set of 8 oriented filters per scale, to show that they span the orientation space with more than 99% accuracy.

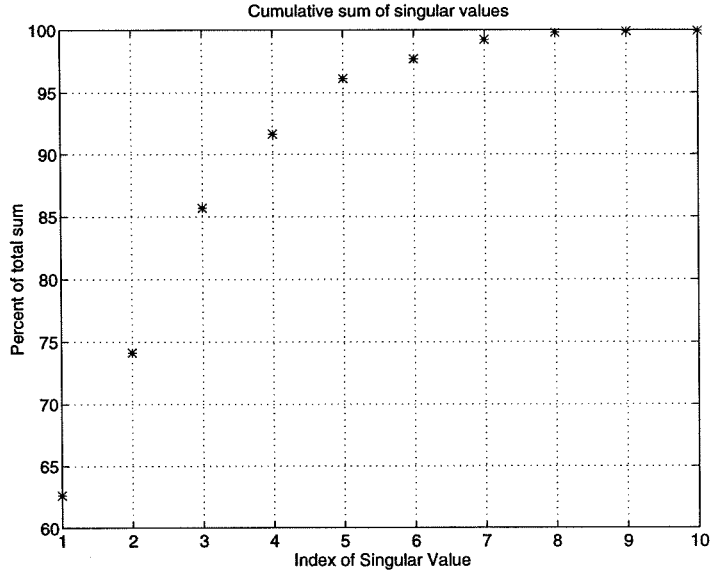


Figure 2.6: SVD decomposition for the oriented pyramid kernels. The first seven singular values contain approx. 99.5% of the sum of all the singular values.

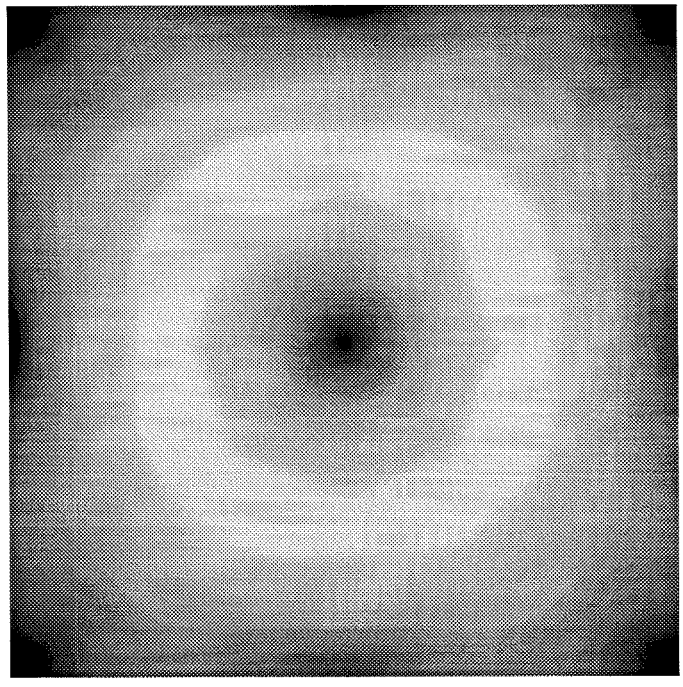


Figure 2.7: Power spectra characteristics for the chosen filter set (+ conjugate counterparts).

The set of 15 filters: 3 scales and 4 orientations per scale, together with the non-oriented Laplacian component, L_n, form our image representation space, which will be utilized in the following chapters. In Chapter 3 power maps are defined from the oriented pyramid maps. These power maps form the feature space for a texture recognition system. In Chapter 4 we extend the analysis of the pyramid filters and show that they form a steerable set of filters. This characteristic enables a rotationally invariant representation. Finally, in Chapter 7 we will revisit the pyramid as a multiresolution image representation scheme, for image enhancement and coding.

Chapter 3
Learning Texture Discrimination Rules in a Multiresolution System

3.1 Introduction

In this chapter we focus our interest on the texture analysis task and within the context of this task we present a recognition framework in which informative discrimination rules are learned from a multiresolution representation of the textured input [GGC92, GGCA94]. Our goal is to combine our knowledge about a good image representation space, using the multiresolution oriented pyramid (as described in Chapter 2), with the advantages of learning paradigms which will be introduced here.

Texture is one of the major modalities which help us understand the visual environment. Image texture, defined as a function of the spatial variation in pixel intensities (gray values), is useful in many application domains and has been a subject of intense study by many researchers. In the texture analysis field are included several different objectives. *Texture recognition* (or *classification*) is the task of correctly labeling homogeneous textured regions in an input image, thus producing a classification map of the input image, with each uniform region identified with the texture class it belongs to. The goal of *texture segmentation* is to obtain a boundary map of the textures present in the input (not necessarily identifying the textured surfaces). An additional interest is in *texture synthesis.* This is often used for image compression and for rendering object surfaces which are as realistic looking as possible. Finally, texture can be used in the problem domain of *shape from texture* in which texture features provide information about surface orientation and shape.

We start this chapter with an introduction to the texture analysis field. We review some of the many different approaches that have been used in the literature, specifically in the texture recognition and segmentation tasks. In Section 3.3.1 we present our texture recognition (and segmentation) system, which combines the multi-resolution image representation with probabilistic learning. The textured input is represented in the frequency-orientation space via the oriented Laplacian pyramid. In this chapter we focus our attention on the classification part of the system in which we associate a class label

to the extracted features. We are interested in probabilistic learning of the input domain. Within this learning paradigm we present the concept of a *rule-based network* and investigate its applicability in an imaging analysis task. Both unsupervised and supervised learning are utilized. In an unsupervised learning stage a statistical clustering scheme is used for the quantization of the feature-vector attributes. A supervised stage follows in which labeling of the textured map is achieved using the rule-based network. This information theoretic technique is utilized to find the most informative correlations between the attributes and the texture class specification while producing probability estimates for the output classes.

State-of-the-art results for the texture classification task and image segmentation results will be presented in Section 3.4. The generalization capability of the system to the identification of an unknown class, so called "pattern discovery", will also be shown. A summary of the system characteristics and its classification capability is the topic of Section 3.5.

3.2 The Texture Analysis Task

Visual texture is one of the most fundamental properties of a visible surface. It participates as one of the major modalities that help us in the understanding of our visual environment. As such it takes part in lower-level to higher-level tasks, from scene segmentation to object recognition. Texture-analysis methods can be utilized in a variety of application domains, such as remote sensing, automated inspection, medical image processing and advanced image-compression schemes. For an overview of the different application domains see [TJ92]. The different textures in an image are usually very apparent to a human observer (for example see Fig. 3.10), but no good mathematical definition can encapture the very diverse texture family. The difficulty in defining texture can be demonstrated by the number of different texture definitions attempted by vision researchers. A few examples are given next:

- "A region in an image has a constant texture if a set of local statistics or other local properties of the picture function are constant, slowly varying, or approximately periodic." [Skl78]
- "The image texture we consider is nonfigurative and cellular... An image texture is described by the number and types of its (tonal) primitives and the spatial organization or layout of its (tonal) primitives... A fundamental characteristic of texture: it cannot be analyzed without a frame of reference of tonal primitive being stated or implied. For any smooth gray-tome

surface, there exists a scale such that when the surface is examined, it has no texture. Then as resolution increases, it takes on a fine texture and then a coarse texture." [Har79]

- "Texture is an apparently paradoxical notion. On the one hand it is commonly used in the early processing of visual information, especially for practical classification purposes. On the other hand, no one has succeeded in producing a commonly accepted definition of texture. The resolution of this paradox, we feel, will depend on a richer, more developed model for early visual information processing, a central aspect of which will be representational systems at many different levels of abstraction." [ZK81]
- "The notion of texture appears to depend upon three ingredients: (i) some local 'order' is repeated over a region which is large in comparison to the order's size, (ii) the order consists in the nonrandom arrangement of elementary parts, and (iii) the parts are roughly uniform entities having approximately the same dimensions everywhere within the textured region." [Haw69]

The above collection of definitions demonstrates that there is no general agreed upon definition. Some are perceptually motivated, and others are driven by the application at hand. It is this lack of definition that makes automatic description or recognition of these patterns a very complex and as yet an unsolved problem.

Recognition and segmentation schemes

Much effort has been expended to automatically segment and recognize different types of texture. A variety of methods exists in the literature. We briefly go over a few of the classical computer-vision approaches, following [TJ92], and give references for further review. We should stress that it is not our intention to cover the entire texture literature, but rather to indicate its variety through the more classical and well-known schemes.

Co-occurrence Matrices

Gray-level co-occurrence matrices (GLCM) have become one of the most well-known and widely used methods. In this method, the characterizing quality of texture is taken as the spatial distribution of gray values. These matrices encapture a full representation of the second-order gray-level statistics. The gray level co-occurrence matrix $P_{\mathbf{d}}$ for a displacement vector $\mathbf{d} = (dx, dy)$ is defined as follows. The entry (i,j) of $P_{\mathbf{d}}$ is the number of occurrences of the pair of gray levels i and j which are a distance $\mathbf{d}$ apart. Formally, it is given as

$$P_{\mathbf{d}}(i,j) = |((r,s),(t,\imath)) : I(r,s) = i, I(t,v) = j|, \tag{3.1}$$

where $(r,s),(t,v) \in N \times N$, $(t,v) = (r+dx, s+dy)$, and $|.|$ is the cardinality of a set. Haralick [Har79] has proposed a number of useful texture features that can be computed from the co-occurrence matrix. These include properties such as *energy , entropy, corre-*

lation and more. Many additional properties have since been introduced in the literature (for a review see [Car92]).

The co-occurrence matrix features suffer from a number of difficulties. There is no well established method of selecting the displacement vector **d** and computing co-occurrence matrices for different values of **d** is not feasible. In addition, for a given **d**, a large number of features can be computed from the matrix, requiring an additional, usually ad-hoc, feature selection method to select the most relevant features.

Structural Methods

Although researchers approach texture differently, most would agree that the texture family can be categorized into two main categories - *structured* and *unstructured*, more *stochastic* textures. Methods that can handle the more structured textures use structural models of texture which assume that textures are composed of texture primitives. The texture is produced by the placement of these primitives according to certain placement rules [VNP86],[LF79]. One needs to be able to define a priori a good set of primitives and placement rules (a tree grammar is commonly used) in order to characterize the textured input. This approach can handle very regular patterns. Some textures which can be handled in this manner are shown in Fig. 3.10 (top row).

Model Based Methods

Model based texture analysis methods are based on the construction of an image model that can be used not only to describe texture but also to synthesize it. Markov random fields (MRFs) have been popular for modeling images. These models capture the local (spatial) contextual information in an image. They assume that the intensity at each pixel in the image depends on the intensities of only the neighboring pixels. MRF models have been applied to various image processing applications such as texture synthesis [CJ83], texture classification [CC85, KK86] image segmentation, image restoration and image compression.

Stochastic models, such as the Markov Random Field (MRF) models, can be used as methods to handle unstructured or stochastic textures. Here the image is seen as an instance of a random process, defined via the model parameters [CC85],[CJ83]. The model parameters need to be estimated in order to define adequately the perceived qualities of the texture. Synthetic textures can then be generated and compared to the original images. This model-based technique can capture certain textures very well (see bottom row of Fig.

3.10), but they fail with the more regular textures as well as inhomogeneous ones [CJ83].

Signal Processing Methods

The methods discussed above use the pixel-based domain as their input space. Other methods exist in the literature which compute texture features from *filtered* images, and use these filtered characteristics in the classification or segmentation tasks.

- *Spatial domain filters* - These include simple edge masks and more complicated masks which are based on spatial moments. The $(p+q)th$ moments over an image region R are given by the formula:

$$m_{pq} = \sum_{(x,y)\in R} x^p y^q I(x,y). \tag{3.2}$$

  If the region R is a local rectangular area and the moments are computed around each pixel in the image, then this is equivalent to filtering the image by a set of local masks. The resulting filtered images that correspond to the moments are then used as texture features.

- *Fourier domain filtering* - Psychophysical results indicate that the human visual system analyzes the textured images by decomposing the image into its frequency and orientation components. Along these lines, texture analysis systems have been developed that perform filtering in the Fourier domain to obtain feature images.

- *Gabor and Wavelet models* - The concept of using multi-resolution processing (channels tuned to different frequencies), has been extended further, to include localization in space and orientation selectivity. This can be found in the wavelet model, and specifically in the use of Gabor filtering (see Chapter 2).

Further review of the texture analysis field, its applications and the different methods available in the literature, can be found in [TJ92]. Although texture analysis has been a subject of intense study by many researchers, it is as yet an open challenge to achieve a high percentage classification rate on the varied texture family within a single framework. Fulfilling this challenge is our ultimate goal. In this chapter we will present a texture recognition system which is based on extracting features in the orientation and frequency domains together with using learning schemes for the recognition task. In this approach, the important characteristics of the input domain are learned from examples, rather than

specified a priori via model-based schemes such as the structured or stochastic models mentioned above.

Before describing our system we first review one particular work in the literature which had a strong motivation for this research.

3.2.1 Motivating Work

Malik and Perona proposed spatial filtering to model the preattentive texture perception in the human visual system. Their model consists of processing the input data through a bank of even symmetric filters followed by half-wave rectification and local interactions. The 3 stage model is depicted in Fig. 3.1. The image is first convolved with a bank of 96 even-symmetric linear filters (oriented Gaussian, Gabor-like filters), followed by half-wave rectification, to produce 192 "response" maps or "feature maps" (modeling outputs of V1 simple cells). A nonlinear inhibition stage, localized in space, within and among the feature maps follows, which suppresses the weak responses and strengthens the strong ones at nearby locations, to give an additional layer of 192 "post-inhibitory" response maps. Finally, at the third level of analysis, a texture gradient is defined by taking a maximum over the gradients of all smoothed post-inhibitory response maps. The texture boundaries are defined as corresponding to local peaks of the texture gradient magnitude. The proposed system was shown to match closely psychophysical experiments in preattentive texture discrimination. Quantitative predictions about the degree of discriminability of different texture pairs match very well with experimental measurements of discriminability of human observers.

Malik and Perona's work is biologically plausible and computationally attractive. Several open issues remain unanswered in the above described model, and these motivate the recognition system which will be described next.

- First, the initial representation of the input domain. **Can we define a finite set of filters that are representative of the general textured input domain?** In the literature we see extreme cases such as 96 filters acting in parallel (producing close to 200 filter response maps) and in a multi-level scheme [MP90], to the selection of 1 filter per texture based on its Fourier spectrum analysis [ACBG90]. We are interested in defining a set of filters which is global enough to represent the general input image domain (without specific input parameter "tweaking"), is

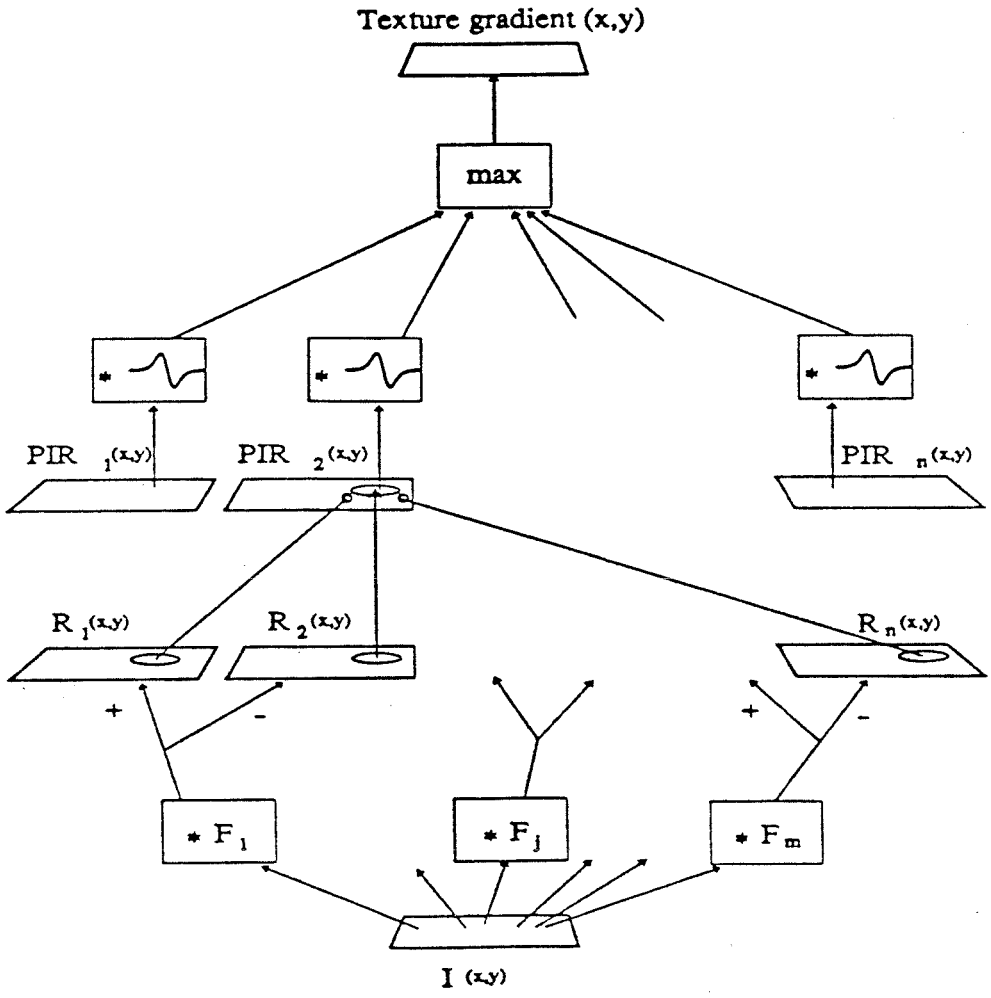


Figure 3.1: A model for preattentive texture discrimination

computationally plausible (can be used in real-time vision systems) and has desired characteristics in shifting the input pixel representation to a more robust, noise immune and invariant representation.

- Second, once we have all the information available (in the different processing layers) - **How can we utilize the existing information in the feature maps for higher-level analysis?**
  In addition to boundary finding, the visual system is able to discriminate and recognize the input domain. This higher-level processing is based on the information present in the extracted feature maps. We are interested in using information theoretic tools to extract the most relevant features (or filters) for discriminating between input textures while recognizing the different textures present in the input.

These two issues are the building blocks of the presented texture-recognition system. As introduced in Chapter 2 we have chosen the oriented Laplacian pyramid as the basis multi-resolution filter set. In this chapter we will discuss the set of 15 filters and their characteristics as applicable to the texture recognition task. Further analysis of the filter set characteristics for invariant recognition will be the topic of Chapter 4. Starting with the multiscale image representation we follow the multi-layered processing scheme (see Fig 3.4) and incorporate information-theoretic learning schemes to utilize the information present in the feature maps for the task of recognition. We should stress here that our interest is in the *recognition* of the input image, thus differing from the preattentive segmentation system of Malik and Perona.

3.3 A Texture-Recognition System

3.3.1 Introduction

We describe a hybrid texture analysis system that incorporates the advantages of learning paradigms, including statistical machine learning, knowledge-based systems and neural networks, in the context of multi-resolution feature extraction techniques. The main goal of the system is to learn a minimal representation for a given library of textures, based on which one can successfully classify and segment new mosaic test images into homogeneous textured regions. Of particular interest is to apply the system to noisy images arising in real-world computer-vision problems.

The system is composed of two main stages: a feature extraction stage followed by a learning stage. In the feature extraction stage we use the multi-resolution oriented pyramid to provide us with a computationally efficient filtering scheme and transform the input space into a set of 15 feature (filter) maps. In the next stage of the recognition system we learn a mapping between the extracted features and the output texture classes. The important characteristics of the input domain are learned from examples (rather than "hardcoded" via parametric modelling schemes).

Many techniques exist in the literature for analyzing sample spaces [DH73]. Each classification scheme has its own advantages and disadvantages, but on most problems one or more of these methods will prove successful in predicting the class output at a performance approaching the Bayes rate. Choosing a classifier is therefore based both on its performance and its special characteristics. In this part of the system we focus on an information based *rule-based* learning scheme (ITRULE). Some of the features of this learning paradigm, as will be described in this chapter, are the following:

- Both unsupervised and supervised learning are utilized.
- An information theoretic technique enables the characterization of the most informative correlations between the input features and the texture class specification.
- These learned correlations are specified as discrimination rules which are available to the user and can enhance his or her knowledge of the input domain and the classification task at hand.
- The learned rules can be mapped onto a rule-based neural network and thus the classification scheme is parallelizable and suitable for implementation using special purpose neural-network hardware.
- The rule-based network provides probability estimates for the output classes rather than just a hard-decision label as its output. These probability estimates can be used for higher-level analysis, such as feedback for smoothing and the learning of an unknown class, the so called "pattern discovery" problem.

We will demonstrate the use of the rule-based learning in the texture-analysis task. Two additional, more standard, non-parametric classifiers will be used along with the rule-based network (sometimes interchangeably). These include the k-nearest neighbor classifier andthe Backpropagation neural-network.

The two building blocks of the recognition system are described in more detail next.

3.3.2 The Feature Extraction Stage

The initial stage for a classification system is the feature extraction phase through which the attributes of the input domain are extracted and presented for further processing. The chosen attributes form a representation of the input domain, which encompasses information that should be useful in future tasks.

In the texture-analysis task there is both biological and computational evidence supporting the use of a bank of orientation-selective bandpass filters for the feature extraction phase [MP90, ACBG90, MC93, Fie87, PZ88]. Studies in neurophysiology [DAT82, CR68] have suggested that a multi-channel frequency and orientation analysis of the visual image formed on the retina is performed by the retina, LGN and the primary visual cortex. Neuronal responses of simple cells in the visual cortex of the macaque monkey to sinusoidal gratings of various frequencies and orientations have shown that the cells are tuned to narrow ranges of frequency and orientation [DAT82]. Psychophysical experiments [CR68] also support the conclusion that the visual system decomposes the image into filtered images of various frequencies and orientations. These studies have motivated multi-channel approaches to texture analysis. An open issue is the decision regarding the appropriate number of frequencies and orientations required for the representation of the input domain. As we have seen in the previous section, the literature presents for the filtering stage, extreme cases such as close to two hundred filters acting in parallel and in a multi-level scheme [MP90], to the selection of one filter per texture based on its Fourier spectrum analysis [ACBG90]. A systematic way to extract initial features is needed. In this work, we use the oriented Laplacian pyramid (a version of the Gabor wavelet decomposition), to define the initial finite set of 15 filters. The definition of the filters and their properties were discussed in Chapter 2. A computationally efficient filtering scheme using the Oriented Laplacian pyramid can also be found in the earlier chapter.

It is the local statistics of the oriented pyramid's coefficients which characterize the image local-area response to the different orientations and frequencies. The first order statistic is zeroed out because the filters have a zero-mean response. A non-linearity is thus needed to reach higher-order statistics. This non-linearity was found to be important in order to discriminate texture pairs with identical mean brightness and identical second-order statistics [MP90]. A measure of power or energy associated with each filtered map

can be defined as the nonlinear operation given below:

$$P_{n\alpha} = |O_{n\alpha}|, n = 0, 1, 2 \;\; \alpha = 1, 2, 3, 4. \tag{3.3}$$

The power maps form a pyramid of the local statistics of the oriented pyramid's coefficients, which characterize the image local-area response to the different orientations and frequencies. Levels 0 and 1 of the power-pyramid are lowpassed and subsampled to be the size of the smallest level of the pyramid (see Fig. 2.4). Each pixel in the resultant power maps thus represents an 8×8 window in the original image.

15 dimensional feature-*vectors* are formed from the extracted power maps. The 15 vector components represent the response at a particular location (x, y) across the 15 corresponding power maps. These vectors consist of the 4 oriented components per scale together with a non-oriented component extracted from the Laplacian pyramid. The resulting feature-vector is of the following form:

$$f = [\, P_{01} \quad P_{02} \quad P_{03} \quad P_{04} \quad L_0 \quad P_{11} \quad P_{12} \quad P_{13} \quad P_{14} \quad L_1 \quad P_{21} \quad P_{22} \quad P_{23} \quad P_{24} \quad L_2 \,]. \tag{3.4}$$

To illustrate the feature-representation space, an example of power maps extracted for the French-canvas ("frcanv") texture are shown in Fig. 3.2, followed by a display of a random feature vector which was extracted from the power maps in Fig. 3.3. In the presented power maps stronger responses are indicated in brighter regions (the Log of the power response is displayed). We first notice that most power is allocated to the highest resolution scale, (scale 0). This corresponds to the high-resolution nature of the input texture. Looking across the oriented components, we see that in the highest resolution scale most of the power is detected in the vertical direction (P_{01}). In the lower resolution scale (scale 1) the power shifts to indicate a strong horizontal direction in the texture. These characteristics are reflected in the displayed feature vector of Figure 3.3. In general, in the database of textures which we are interested in for this work, we will find one dominant oriented component per scale for the more "structured" (oriented) textures. A more uniform distribution of power across the oriented components is a characteristic of the "nonstructured" (non-oriented) textures. Examples of characteristic feature vectors for these two families of textures can be found in Fig. 3.11.

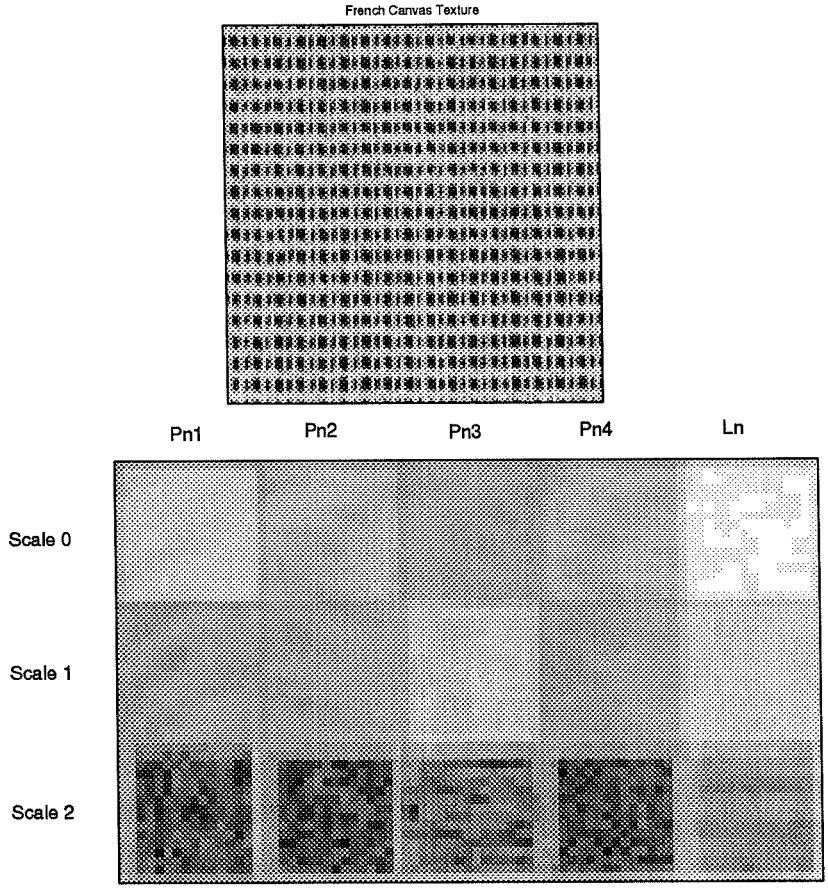


Figure 3.2: Power maps for the French Canvas ("frcanv") texture. Stronger power responses are brighter. Log of brightness level is displayed.

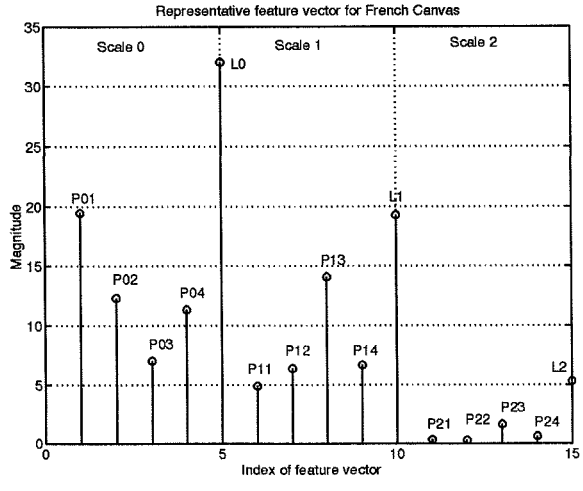


Figure 3.3: A representative feature vector extracted from the above power maps.

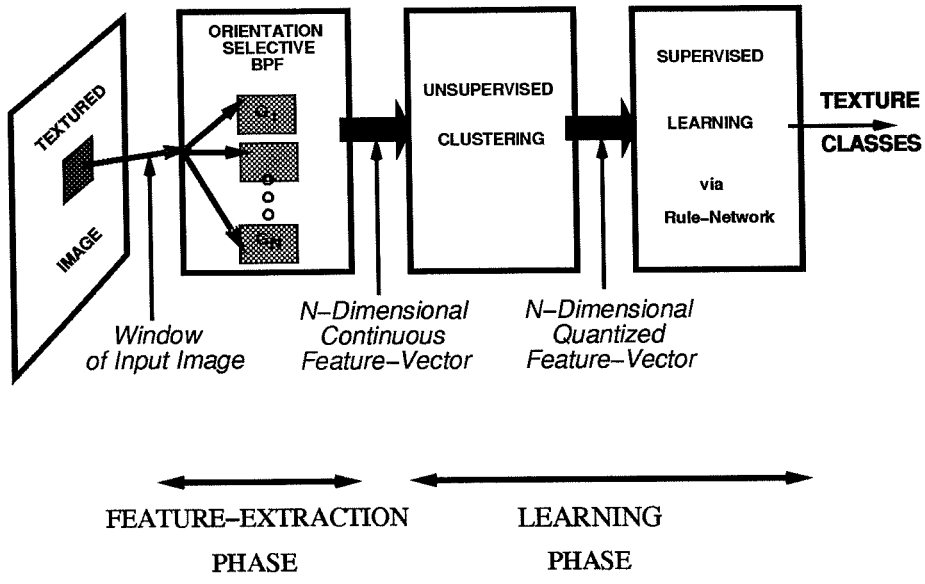


Figure 3.4: System Block Diagram

3.3.3 The Learning Stage

The Rule-Based Network Classifier

Two building blocks compose the learning phase of the rule-based classifier, as shown in Fig. 3.4 and as operationally outlined in Fig. 3.5. These consist of an unsupervised clustering stage followed by a supervised classification stage. The learning mechanism derives a minimal subset of the input feature maps (or filters) which conveys sufficient information about the visual input for its differentiation and labeling. The feature space is reduced in both unsupervised and supervised stages of analysis. In the unsupervised stage a machine-learning clustering algorithm is used to quantize the continuous input features. In the supervised stage the existing information in the feature maps is utilized for higher-level analysis, such as input labeling and classification, while providing probability estimates for the output classes. The two stages of the learning phase of the system are described next.

Unsupervised Clustering

The unsupervised learning stage can be viewed as a preprocessing stage for achieving a more compact representation of the filtered input. The goal is to quantize the continuous valued features which are the result of the initial filtering. The need for discretization

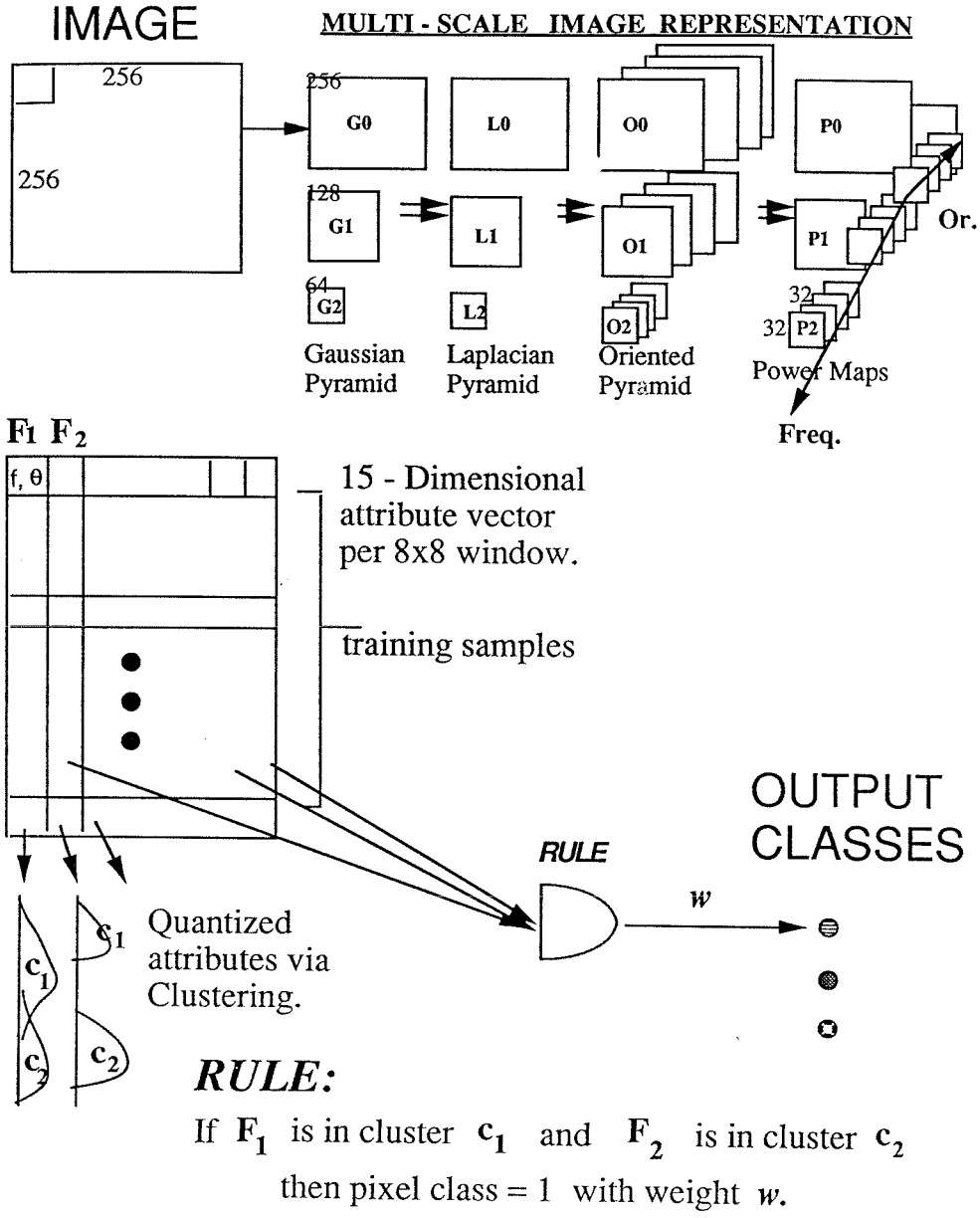


Figure 3.5: Outline of the rule-based system operation

becomes evident when trying to learn associations between attributes in a symbolic representation, such as rules. Moreover, in an extended framework, the dimensionality of the feature domain can be reduced.

The output of the filtering stage consists of 15 continuous-valued feature maps. Thus, each $(8*8)$ window in the input image is represented via a 15-dimensional feature vector. An array of such vectors, viewed across the input image, is the input to the learning stage (see Fig. 3.5). We wish to detect characteristic behavior, across the 15-dimensional feature space, for the family of textures to be learned.

In this work, each dimension, out of the 15-dimensional attribute vector, is individually clustered. All samples are thus projected onto each axis of the 15-dimensional space and one-dimensional clusters are found using the K-means clustering algorithm [DH73]. The K-means algorithm is a statistical clustering technique which consists of an iterative procedure of finding K means in the sample space, following which each input sample is associated with the closest mean in Euclidean distance. For details on the algorithm implemented see Appendix A. The found means, labeled 0 thru K minus 1 arbitrarily, correspond to discrete codewords. Each continuous-valued input sample gets mapped to the discrete codeword representing its associated mean. The output of this preprocessing stage is a 15-dimensional *quantized* vector of attributes which is the result of concatenating the discrete-valued codewords of the individual dimensions.

As some of the dimensions are more representative than others, it is the goal of the supervised stage to find the most informative dimensions for the desired task (with the higher differentiation capability) and to label the combined clustered domain.

Supervised Learning via the Rule-Based Network

The goal of the supervised stage is to classify the input image, while finding the most informative input dimensions, or attributes, for the desired task, thus reducing the dimensionality of the representation. We wish to learn a classifier which maps the output features of the unsupervised stage to the texture class labels. Using the rule-based network for this task, we have a probabilistic framework for the classification as well as the advantage of the "readability" of the extracted rules. The rule-based classifier defines correlations between input features and output classes as *probabilistic rules* of the form:

If $Y = y$ then $X = x$ with probability P.

Here, $Y = (Y_1, ..., Y_N)$ represents the attribute vector and X is the set $(x_1, ..., x_m)$ of m possible output classes. In this work, N=15 and m is the number of texture classes learned. Given an initial labeled training set of examples, where each example is of the form $(Y_1 = y_1, .., Y_N = y_n, X = x_i)$, the system is to learn a classifier such that when presented with future test attribute vectors, it will estimate the posterior probability of each class.

Information Theoretic Measure of Rule-Value

A data-driven supervised learning approach utilizes an information theoretic measure to learn the most informative links or rules between features and class labels. Such a measure was introduced by Smyth and Goodman [GS89] as the J measure, defined as follows:

$$J(\mathbf{X} = x_i; y) = p(y)\left(p(x_i|y)\log\left(\frac{p(x_i|y)}{p(x_i)}\right) + (1 - p(x_i|y))\log\left(\frac{(1 - p(x_i|y)}{(1 - p(x_i))}\right)\right).$$

Here, the information content of a rule is represented as the average amount of information that attribute values y give about the class X. The J measure has several desirable properties as a rule information measure. It is comprised of two main terms. The first is $p(Y = y)$, the probability that the particular set of attribute values will occur. The second term is a measure of the average change in bits necessary to specify X between the *a priori* distribution $p(X)$ and the *a posteriori* $p(X|y)$ distribution. Smyth and Goodman have shown that this measure can be interpreted as a special case of the cross entropy of the two distributions and satisfies all the properties of an information measure. Maximizing the product of the two terms is equivalent to simultaneously maximizing both the simplicity of the specific correlation vector, Y, and the goodness of fit to the perfect predictor of X. The simplicity of the rule, or the correlation vector Y, corresponds directly to the number of attribute-value conjunctions, the so-termed rule *order*. Lower-order rules have fewer conditions and thus have a higher probability of occurring. Higher-order rules are more specialized and are therefore better predictors. Maximizing the J measure results, therefore, in a tradeoff between accuracy and generality (higher-order and lower-order rules, respectively) in the prediction process.

The J-measure is next used in a search algorithm to search the space of all possible rules relating the attributes to the class, X, and produce a ranked set of the most informative

rules which classify X. For details about the rule-extraction algorithm refer to Appendix B.

Probabilistic Classification

The most informative set of rules via the J measure is learned in a training stage. Once the rule set is constructed, the rules may be used in parallel to compute the *posterior probability* of each class [GHMS92]. When presented with a new input evidence vector, Y, a set of rules can be considered to "fire." These are a subset of the correlations learned which the input attribute vector matches. Using Bayes' rule, the classifier estimates the log posterior probability of each class given the rules that fire. Let F be the set of rules which fire and $s_1, ..., s_{|\mathcal{F}|}$ be the actual attribute-value conjunctions corresponding to the fired rules. We get (for a derivation see Appendix B.2):

$$\log p(x_i|s_1, \ldots, s_{|\mathcal{F}|}) = \log p(x_i) + \sum_{j=1}^{|\mathcal{F}|} W_{ij}, \tag{3.5}$$

with

$$W_{ij} = \log\left(\frac{p(x_i|s_j)}{p(x_i)}\right),$$

where $p(x_i)$ is the prior probability of the class x_i, and W_{ij} represents the evidential support for the class as provided by rule j. In the absence of any rules firing, the estimate of each class is given by the bias value, namely the log of the prior probability of the class. Given a set of rules which fire, each rule contributes a weight to its corresponding output class. A positive weight implies that the class is true, while a negative weight implies it is false. The W_{ij}s provide the user with a direct explanation of how the classification decision was arrived at. Each class estimate can now be computed by accumulating the "weights of evidence" incident on it from the rules that fire. This can be done in a parallel manner. The largest estimate is chosen as the initial class label decision. The probability estimates for the output classes can now be used for feedback purposes for spatial smoothing and further higher-level processing.

The rule-based classification system can next be mapped into a three-layer feed-forward architecture as shown in Fig. 3.6. The input layer contains a node for each attribute except the class attribute. The hidden layer contains a node for each rule and the output layer contains a node for each class. Each rule (second layer node j) is connected to a class i via the multiplicative weight of evidence W_{ij}. The bias of each output node is $-log(p(x_i))$.

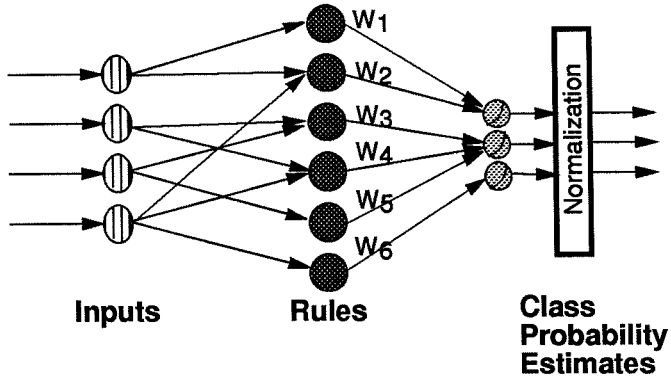


Figure 3.6: Rule-based network

Each output node sums its inputs subtracts the bias and exponentiates the result. Thus the output of each third-layer node is the posterior probability of the class it represents. A winner-take-all stage can be added to decide upon the most likely class. This hybrid rule-based neural model combines the explicit knowledge representation in the form of rules with the parallel implementation of neural-network architectures.

Additional Classifiers

Throughout the remainder of this chapter we will investigate into characteristics of the rule-based network to verify its effectiveness as a classifier and demonstrate its interesting characteristics in the image-analysis domain. We will also exemplify the classification performance of a few other, more standard classifiers, such as the k-nearest-neighbor classifier (k-nn) and the Back-Propagation neural-network classifier (Backprop). We next briefly describe each of the classification algorithms used.

The k-Nearest Neighbor Classifier

The nearest-neighbor classifier is one of the simplest learning methods. It is completely non-parametric, as nothing is apriori assumed about the population. Geometrically, the nearest-neighbor method can produce any arbitrary complex surface to separate the classes based only on the configuration of the sample points and their metric.

Let $\mathbf{x_1}, ..., \mathbf{x_n}$ be a set of n labelled samples, and let $\mathbf{x_m}$ be the sample nearest to $\mathbf{x}$ (we will be using the Euclidean distance); i.e.,

$$||\mathbf{x}_m - \mathbf{x}|| = min_i||\mathbf{x}_i - \mathbf{x}||, i = 1, .., n. \tag{3.6}$$

Then the *nearest-neighbor rule* for classifying $\mathbf{x}$ is to assign it the label associated with $\mathbf{x_m}$. An immediate extension of the nearest-neighbor rule is the *k-nearest-neighbor rule.* This rule classifies $\mathbf{x}$ by assigning it the label *most frequently represented* among the k nearest samples. In other words, a decision is made by examining the labels on the k nearest neighbors and taking a vote.

The neighborhood size (k) has to be smaller than the size of the feature space occupied by the smallest of the classes in the problem domain under consideration. If p = minimum number of samples from each of the training set classes, then k has to be in the range $1 < k << p$. [Das91] In our implementation, the algorithm is run with a variety of k values: $k = 1, 5, 10, 20, 50, 100$ (for N training samples per class, the maximum k value was taken to be $k \leq 0.25N$). In the results presented the classification rates for several k values will be presented as well as the average over the percentage accuracy.

In its standard form, the nearest-neighbor method involves no effort in *learning* from the samples. The tradeoff is the computational load in predicting the classification of a new case. Each new input sample must be compared with every sample in the prestored sample space.

The Backpropagation Neural-Network Classifier

The Backpropagation algorithm is well known in the neural-network literature [RHW86]. It has been successfully used in a variety of application domains. We are therefore motivated to check its performance in the texture recognition task. We use a three-layer network with the first layer containing a single node for each attribute (in our case = 15), the third layer containing a node for each class and the center layer containing hidden units that connect between the input and output units. A known problem associated with neural-network techniques is the definition of the network architecture. If one chooses too few hidden units, the network may have too limited a hypothesis space to learn the required concepts, while with too many it may overfit the training data. In the experiments of the following section we average over three runs. For the 15-dimensional input space that we have, we average over the cases of 30, 60 and 90 hidden units.

Earlier work by Goodman et al. [GHMS92] has demonstrated that the rule-based network is competitive in terms of classification accuracy when compared with alterna-

tive approaches, while achieving useful explicit rule sets about the task at hand. In the following section we apply the rule-based scheme to the texture recognition task, analyze its behavior with changing parameters and demonstrate some of its characteristics. We show that the extracted rules can extract an informative set of filters of the input representation, which are the more important ones for the classification task at hand. We use the probabilistic classification to an advantage in labeling noisy real-world scenes, and in determining what we define as "unknown" classes.

3.4 System Performance

3.4.1 Introduction

We present the results of applying the above-described system to textured images. The system's characteristics are demonstrated and the classification capabilities on large image databases are shown.

The database we use throughout the work consists of both unstructured and structured textures. Most of it consists of images taken from the Brodatz library of natural textures [Bro66] [1] which is the main source for textures in the literature. This set is augmented with miscellaneous textures that we scanned in, including jeans, newsprint, check-book cover and others (scanning resolution is 100 pixels/inch). We collected a large set of 30 such textures. The complete database is presented in Figure 3.8 followed by a corresponding label chart in Figure 3.9. Note that the Brodatz textures are labeled with their corresponding plate number from the original Brodatz book.

The experimental setup

In most of the following experiments the input to the system is taken as a $256 * 256$ size texture patch. The original image of size $256 * 256$ is input to the pyramid, resulting in feature maps (power maps) of size $32 * 32$ (refer to Figure 3.5). Feature vectors are extracted from the feature maps and are the input to the classification system. For a classification resolution of $8 * 8$ windows in the original image, we utilize the feature maps directly. Boundary effects are eliminated by extracting a $30 * 30$ patch from the feature maps (see Figure 3.7). We thus have a set of 900 feature vectors for the classification task. We use 300 vectors for training and 100 different ones for testing. These are chosen from small disjoint patches (chosen to be as far away as possible).

[1] database taken from a USC source [Web83]

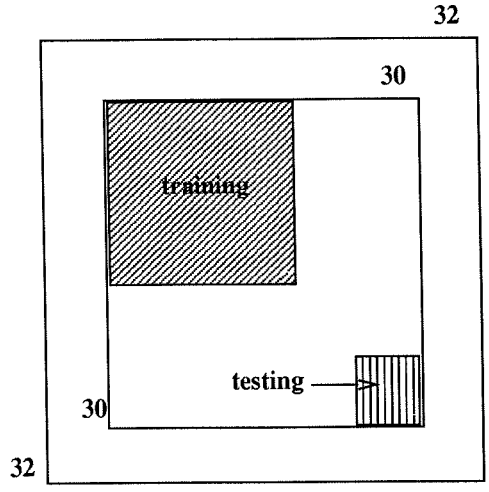


Figure 3.7: Experiment setup

window size	# training vectors/class	# testing vectors/class
8*8	300	100
16*16	125	100
32*32	40	9
64*64	12	4

Table 3.1: Training and testing sets for different classification windows

It is interesting to note the increase in classification accuracy as we vary the classification resolution. Different size windows in the feature maps are averaged to produce representative feature vectors corresponding to $16*16, 32*32$ and $64*64$ windows of the original image. To get the $16*16$ classification resolution we average over $2*2$ neighborhood blocks. Ignoring the boundary we have $15*15 = 225$ feature vectors. 125 are used for training and 100 disjoint ones are used for testing. Similarly, for $32*32$ resolution, $4*4$ windows are averaged over to give $7*7 = 49$ feature vectors. 40 are used in training and 9 are used for testing. Finally, for the $64*64$ resolution we need to average $8*8$ windows. Here, 12 feature vectors are used for training and 4 are used for testing. The above mentioned figures are summarized in Table 3.1.

In all the above cases no overlap exists between the training and testing sets. Moreover, no overlap exists amongst the training windows or the testing windows. In cases of limited

data (such as the $32 * 32$ and $64 * 64$ cases) 4 different runs are made (interchanging the testing vectors) and the average result is the one presented.

Comparing to the literature

There is a large variability in the experimental setups presented in the literature, which makes a comparison a very difficult task. The variety one finds includes the following:

- The data-set size and complexity. This issue includes the relative sizes of the training and testing sets and their relationship to each other (overlap etc). A database size of less than or equal to 10 textures has been the common ground for comparison for many years now, with no attempt to check the scalability of the presented systems with increasing the input space. The literature which we look at covers the wide range of methods, including Gabor filtering schemes, MRF schemes, co-occurrence matrices and others. In this variety of schemes we find databases of 5 [JF91, Car92], 8 [CJ85], 9 [CFP91] and 12 [Uns86] as the more common ones in the literature. It is only recently that we start to see an increase in the database size.

- Preprocessing of the data. There are many methods that use the raw input (e.g., [JF91]) while others use a variety of preprocessing methods. A common scheme is histogram equalization [Car92, Uns86]. In [Car92] it is claimed that histogram equalization has a significant impact on the performance of co-occurrence matrices, as it greatly reduces the results compared to the raw images. One should then note the effect of the input format on the classification results.

- Classification window size. Window sizes vary in the literature from $16 * 16$ to $128 * 128$. In some cases an overlap exists between the training and the testing windows. This parameter is of great significance in the classification results, with a substantial improvement usually evident as we increase the window size [HO88, Uns86].

Due to this large variability in experimental setups an exact comparison between different reported results is difficult. In the following set of experiments we address some of these experimental issues directly and we will point out relevant results for comparison as we go along.

List of experiments

In Experiment 1. we start our analysis on a 10 texture subset which is composed of

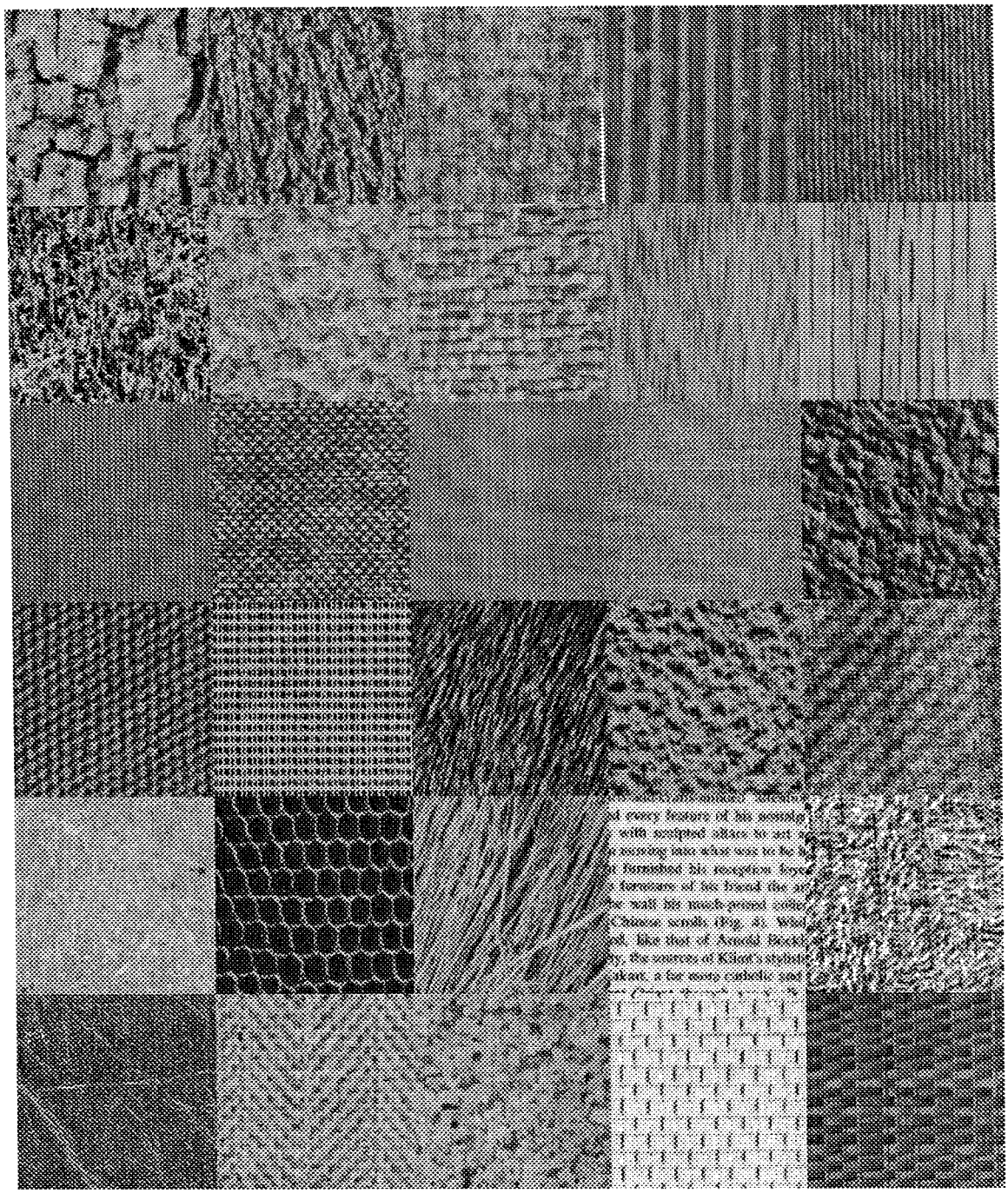


Figure 3.8: 30 texture database

bark (D12)	**calf** (D24)	**cloth** (D19)	**crdbrd**	**jeans**
grass (D9)	**pig** (D92)	**raffia** (D84)	**water** (D38)	**wood** (D68)
backpack	**bookbox**	**brownbag**	**chbkcover**	**cork** (D32)
cotcanv (D77)	**frcanv** (D20)	**fur** (D93)	**hmpaper** (D57)	**napkin**
prtboard	**reptile** (D3)	**straw** (D15)	**text**	**towel**
vinyl	**herringbone** (D16)	**sand** (D29)	**wire** (D6)	**strawmat** (D55)

Figure 3.9: 30 texture database - labels. Textures taken from the Brodatz book are labeled with the corresponding plate number. Others have been scanned in from natural objects. crdbrd = cardboard; chbkcover = check-book cover; hmpaper = handmade paper; prtboard = particle board.

both structured and unstructured textures, as shown in Figure 3.10. The difference in performance for the 2 sets of structured and unstructured textures is shown in Experiment 2. In Experiment 3 we look into the behavior of the rule-based network in detail on 2 texture pairs, one pair representing an easy discrimination task and the other pair representing a difficult task. In Experiment 4 we analyze the different system parameters on the 10 texture dataset. We demonstrate the system's sensitivity to increasing the number of input textures (so called "scalability"), and demonstrate high classification accuracy for the entire 30 texture database of Figure 3.8. In our results we compare several classification schemes to demonstrate the consistency across the schemes, while identifying some of the interesting features of the rule-based approach. The probability maps which we extract from the rule-based scheme are utilized in Experiments 5 and 6: image mosaic labeling and segmentation and the pattern discovery task.

Relevant Parameters

For each classification scheme we will list the set of parameters which determine its performance. In the **rule-based network** these include the following:

- K - The number of clusters in the unsupervised clustering stage.
- N - The maximum number of rules allocated.
- n - The maximum number of rules allocated per class.
- O - The maximum order of the rules. This parameter refers to the number of attribute conjunctions extracted in forming the rules.

Unless otherwise noted, we use the following default values:

$K = 10$; $N = 10{,}000$; $O = 3$; No pruning.

The default values were chosen based on experience. We allow for a large number of rules as our initial starting point. It is usually the case that the lower order rules ($O = 1, 2, 3$) are the most informative ones. As an increase in the order substantially effects the rule-extraction search, we limit the maximum order to 3. The sensitivity to the parameter K will later be shown to be low, thus allowing for a fixed default value over most of the experiments presented.

In the **k-nearest neighbor** scheme the relevant parameter is k - The number of neighbors used in the voting scheme. If not otherwise specified, the result presented is the average over $k = 1, 5, 10$ and 50.

In the **Backprop network** the relevant parameter is h - The number of hidden units

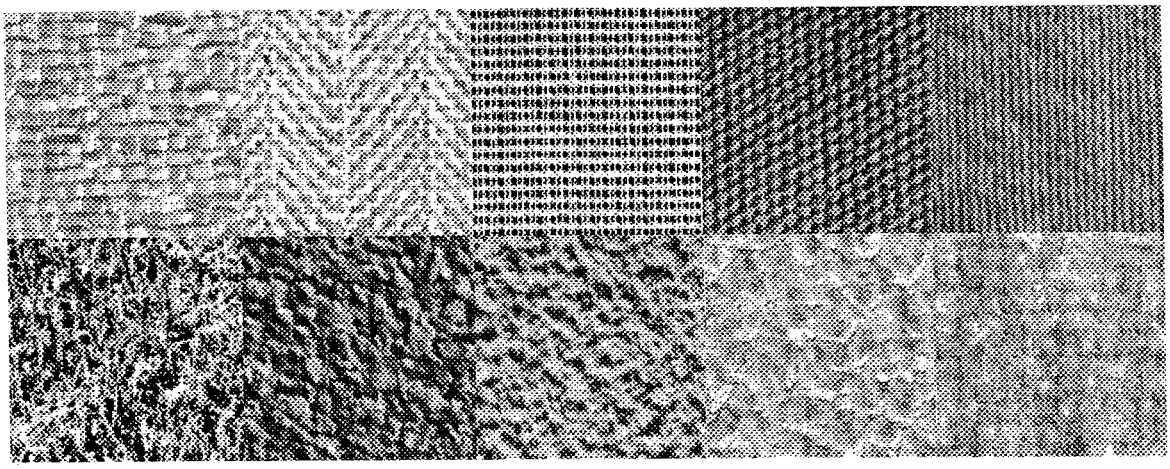


Figure 3.10: Structured and unstructured textures (top and bottom, respectively) - 10 texture case

in the network. Default is the average over $h = 30, 60, 90$. We use the conjugate gradient scheme. 500 training epochs are used with a threshold set at 10^{-6}. Each run is averaged over 5 different random seed values.

3.4.2 Results

Experiment 1: Classification of a 10 texture set

In this experiment we look at an 8*8 classification resolution for a 10 texture discrimination task. In the training phase, 300 training examples are given per texture class. The classification result given is based on labeling correctly a disjoint set of 100 $8 * 8$ windows. Table 3.2 presents the class-confusion matrix for the 10 textures of Figure 3.10, using the rule-based network classification scheme.

The overall classification rate is 94.3%. Very high percentage classification rates are achieved for both the structured and the unstructured textures which compose the set.

In Table 3.3 we summarize the classification results of the rule-based network, the k-nearest neighbor scheme and the Backprop network on the 10 texture case. We note the comparable high-performance across the 3 classifiers. These results compare favorably with the literature (e.g., [CC85, Uns86, CJ85]).

The classification resolution of $8 * 8$ is very high (as compared with the literature). The advantage of this high-resolution classification scheme is that we can achieve image segmentation via the recognition process (see Experiment 5).

	raffia	herring	frcanv	cotcanv	jeans	grass	cork	hmpaper	pig	cloth
raffia	**91.02**	0	0	0	0	0	0	0	8.98	0
herr	13.28	**84.76**	0	1.95	0	0	0	0	0	0
frcanv	0	0	**100**	0	0	0	0	0	0	0
cotcanv	0	0	0	**100**	0	0	0	0	0	
jeans	0	0	0	0	**100**	0	0	0	0	0
grass	0	0	0	0	0	**100**	0	0	0	0
cork	0	0	0	0	0	7.03	**92.58**	0.39	0	0
hmpaper	4.68	0	0	0	0	0	4.29	**91.02**	0	0
pig	3.52	0	0	0	2.34	0	0	0	**86.72**	7.42
cloth	1.17	0	0	0	0	0	0	0	1.56	**97.26**

Table 3.2: Class confusion matrix for a 10 texture case

10 TEXTURE CASE - 8*8 window	
Classification scheme	Classification accuracy
Rule-based network	
K=10, N=10000, O=3	**94.3%**
k-nn	
k=1	86%
k=5	86.4%
k=10	87.3%
k=20	87.2%
k=50	87.7%
k=100	85.3%
average result	**86.65%**
Backprop	
h=30	94.56%
h=60	94.78%
h=90	94.50%
average result	**94.6%**

Table 3.3: Classification results for 10 texture case

5 TEXTURE CASE	
	rule-based network
structured set:	97%
unstructured set:	81.8%

Table 3.4: Comparing classification results for a structured set of textures (top row of 10 texture set) and an unstructured set of textures (bottom row)

Experiment 2: Structured vs. Unstructured Textures

Structured textures are usually more easily discriminable than unstructured (or stochastic) textures, due to strong orientation components and regular arrangement of primitives. This can be seen in Fig. 3.10, in which the top row consists of structured textures and the bottom row is a set of unstructured textures.

Running the classification scheme on these two sets of textures separately exemplifies the difference in the task difficulty. We run the system on the 5-structured textures, consisting of: raffia, herringbone, frcanv, cotcanv and jeans (fig 3.10 top) , and a second, unstructured set of: grass, cork, hmpaper, pig and cloth (fig 3.10 bottom). In table 3.4 are the classification results using the rule-based network. As expected, the system has an easier time with distinguishing between the structured textures than the unstructured ones. We note the possible variability in classification results, as dependent on the actual textures used in the database.

Experiment 3: More Insight into the Rule-Based System Operation

To achieve a better understanding of the system operation we look at the discrimination of two pairs of textures. A simple case (oriented textures) will be chosen as the raffia and wood pair. A more difficult case (nonoriented) will be shown using the pig and cloth pair [2]. The texture pairs together with a corresponding set of feature vectors are shown in Fig. 3.11.

We note that the more difficult case of the pig and cloth discrimination task corresponds to visually similar textures. This difficulty is reflected in the feature space as well. Looking at the examples of feature vectors we note that the oriented textures have a dominant feature representing the preferred orientation of the texture, while the nonoriented

[2]Note that we have simplified the world by looking at only a pair of textures at a time. In this case, it suffices to characterize one class well. We will therefore see high classification results in both cases.

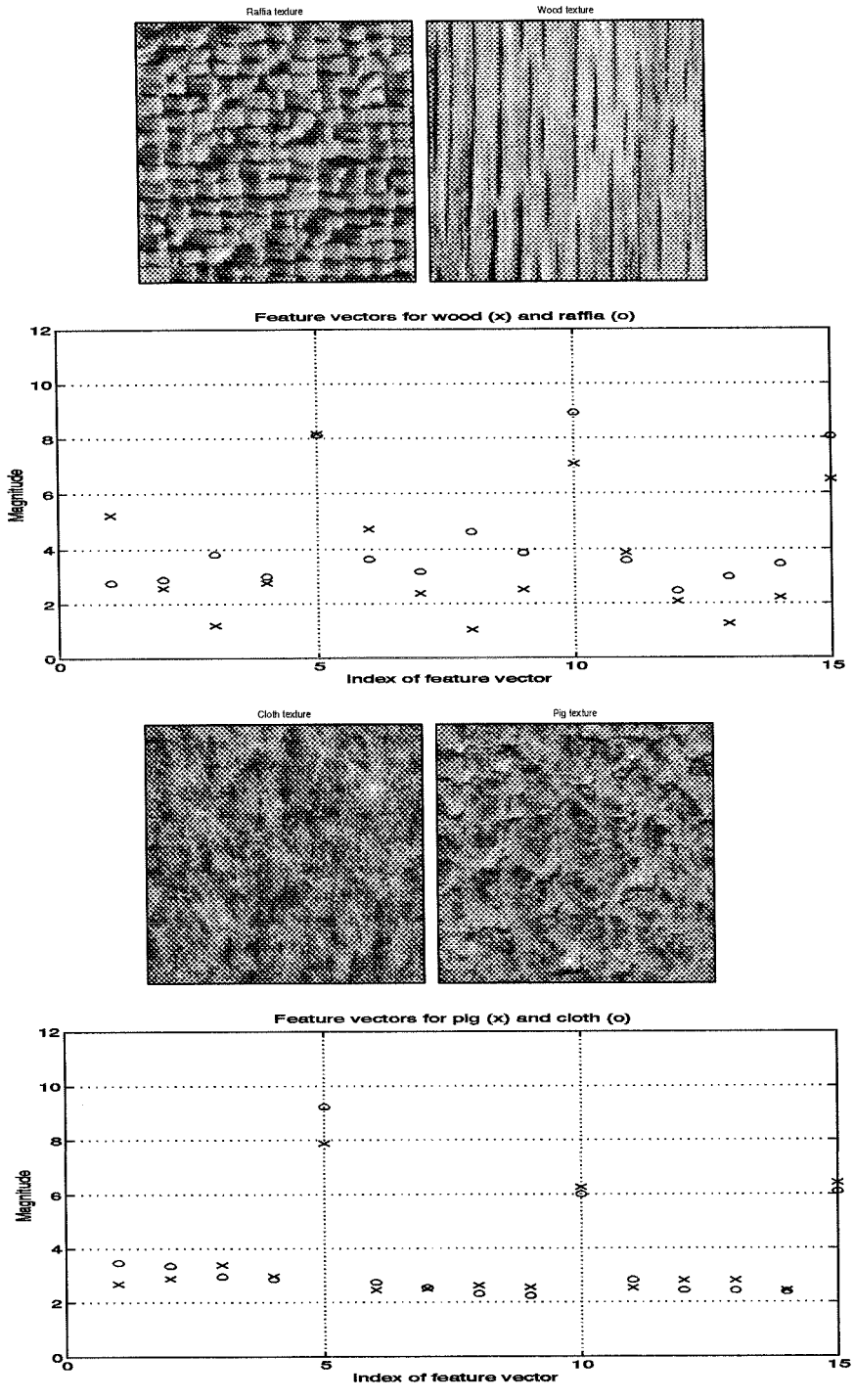


Figure 3.11: top: The raffia and wood texture pair with a corresponding set of feature vectors. bottom: Similarly for the cloth and pig textures.

1.	$IF\ f2 : 1.09 - 2.08$	$THEN$	$class = wood;$	$p = 1.0;$	$J = 0.281$
2.	$IF\ f7 : 0.99 - 1.87$	$THEN$	$class = wood;$	$p = 1.0;$	$J = 0.231$
3.	$IF\ f7 : 3.54 - 4.83$	$THEN$	$class = raffia;$	$p = 1.0;$	$J = 0.217$
4.	$IF\ f2 < 1.09$	$THEN$	$class = wood;$	$p = 1.0;$	$J = 0.199$
1.	$IF\ f1 < 1.9$	$THEN$	$class = pig;$	$p = 0.97;$	$J = 0.115$
2.	$IF\ f5 < 5.0$	$THEN$	$class = pig;$	$p = 0.99;$	$J = 0.0818$
3.	$IF\ f2 < 1.91$	$THEN$	$class = pig;$	$p = 0.96;$	$J = 0.0816$
4.	$IF\ f1 < 1.90\ and\ f15 > 4.96$	$THEN$	$class = pig;$	$p = 0.96;$	$J = 0.0722$

Figure 3.12: Set of most informative rules for raffia and wood discrimination (top). Similarly for the cloth and pig textures (bottom set).

textures have more uniform curves across orientation space which are less distinct.

A large set of feature vectors per class are presented to the system in the training mode and the most informative set of rules for the discrimination task are extracted (see Section 3.3.3). A set of the top most informative rules is presented (for each case) in Fig. 3.12. Shown (from left to right) is the rule, the probability $p(class|attribute\ vector)$ and the corresponding J measure, with the strength of the rule (via the J measure) decreasing with rule number.

It is interesting to see if the extracted rules match our understanding of the task at hand. In the case of the raffia and wood textures we note that the most informative rule identifies the wood texture based on attribute 3 of the feature vector which indicates a *low* response in the horizontal direction. For raffia the most informative rule identifies attribute 8, which is detecting a *strong* response in the horizontal direction. This is a main distinguishing characteristic across these 2 textures, as we see by examining the texture feature-vector curves. [3] In the case of the cloth and pig textures we see that the first 4 rules all relate to the pig class. Examining the feature vectors of Fig. 3.11 we see that the rules do not reflect the shown values (either they don't cover the example curves at all or they match both classes). Note the lower J measure values as compared to the structured case.

Using the extracted set of rules the raffia and wood pair are classified with 100% accuracy. The cloth and pig pair are classified with 90% accuracy. The system is able to cope with both the structured and unstructured pairs, but with reduced accuracy in the latter case.

[3] attributes are numbered in accordance to the index shown in the Figure.

input format	K-nn	Backprop
raw	87%	94.5%
histogram equalization	79%	91.4%

Table 3.5: Classification results with histogram equalization

The relative difficulty of the classification task can be detected via the *number* of rules which are required to cover the training sample space. We preserve the n most informative rules per class, with n varying ($n = 1, 2, ...$), and check classification results on the *training* set. In the raffia-wood case, **10** rules per class suffice to cover the space of examples with 100% accuracy. in the case of the pig and cloth texture pair **430** rules are needed per class, and these are only able to correctly label 546 out of 600 training set examples at 91% accuracy.

Experiment 4. System Analysis

- Sensitivity to the Input Image Format

  In the literature one finds a variety of preprocessing steps which are performed on the input image prior to its analysis. The most common one is *histogram equalization.* In this preprocessing step one tries to eliminate the influence of first-order statistics in the texture analysis task (these statistics are derived from the gray-level histogram), by making the gray-level histogram match a *uniform* distribution.

  Using a set of histogram-equalized images is considered to be a more challenging task for a classification system. In one particular recent work [Car92] it is claimed that using a set of 15 Gray-level Co-occurrence matrices (GLCM) features can discriminate among 15 Brodatz textures with close to 0% error rate. Using the same set of features on the set of 15 histogram equalized textures, Carstensen achieves an accuracy rating of 74.3%. Carstensen concludes that histogram matching of textures has a significant effect on the discriminatory performance of GLCM features.

  We wish to check the sensitivity of our system to the input representation. In Table 3.5 we compare classification results on the 10 texture set, with and without histogram equalization.

  Only a small decrease in performance of 3% is seen with the Backprop network. A larger decrease is evident with the k-nearest neighbor classifier. The range of

window size	#train/class	#test/class	K-nn	Backprop	Rule network
8*8	300	100	87%	94.5%	94.3%
16*16	125	100	93%	96.1%	95%
32*32	40	9	99%	100%	97.78%

Table 3.6: Classification results on different window sizes

3% decrease in performance will reappear as we increase our testing to a 30 texture case. These results suggest that the feature-space representation of the system is only minimally sensitive to the 1st-order statistics of the image, and that the combination with the learning schemes preserves the high-classification results.

- Sensitivity to Window Size

  Experiments presented in the literature differ in several of their parameters, one important one being the classification window size. The classification results are very much dependent on the classification window size and therefore care should be taken when judging across results. Larger windows produce a more homogeneous feature set (better SNR) and thus result in an easier classification task. To illustrate this point, the following experiment was run on the 10 texture database. Different size windows in the feature (filter) maps were averaged over to produce representative feature vectors corresponding to 8*8, 16*16 and 32*32 windows in the original input image (as explained in the introduction to this section). Table 3.6 presents the corresponding results. We see the increase in classification performance for the larger windows.

  The presented results compete with quoted results in the literature. We will quote a few. In [CC85] GMRF models are used on a set of 7 textures. Several features sets are used. Results presented are in the range of $93-99\%$ for $64*64$ window sizes, with two cases of 82% and 93% given for the $32*32$ window case. In [Uns86] sums and differences of histograms are used and compared to co-occurrence matrices on a set of 12 textures. Results include 88.2%(86.7%) for $16*16$ windows, 96.8%(93.23%) for $32*32$ windows and 97.92%(95.83%) on $64*64$ windows. (Figures in brackets are co-occurrence classification results quoted in [Uns86]). Finally, quoting one work which uses Gabor filters for the classification task we cite results from [CJ85] in

window size	standard deviation
8*8	1.86
16*16	0.08

Table 3.7: Sensitivity to the number of clusters for different window sizes

which a set of 8 textures are used with results of 98% for $64 * 64$ windows reducing to 91% for $32 * 32$ windows.

- Sensitivity to the Number of Clusters

  In this experiment we wish to check on the sensitivity of the rule-based system to the parameter K of the Clustering stage. We use the 10 texture database keeping constant the maximum number of rules ($N = 10,000$) and the maximum rule-order, $O = 3$. 5 runs were made with $K = 5, 7, 10, 15, 20$. The standard-deviation of the results is tabulated in Table 3.7.

  We note the small variation as we vary K, especially with the larger classification window sizes. This is related to the fact that as we increase the window sizes the feature set is more homogeneous, making the clustering an easier step and resulting in as easier classification task overall (as demonstrated above).

- Sensitivity to Number of Rules

  We next investigate the performance of the rule-based system as the number of rules per class is increased. The number of rules used in the classification task is a variable (N) of the system. Experience has shown that when the system is given a set of N rules to learn, dominant classes, which are easily distinguishable, have many rules associated with them, while the more problematic and difficult texture classes, have very few rules and sometimes are not included in the learned rule set at all. This leads to a problem since any class that is not represented by at least one rule can not be identified. It will be classified incorrectly or labeled as an unknown class (see pattern-discovery case at the end of this section). Following this observation, we define a modified parameter, n, to reflect the number of rules *per class*; i.e., the n most informative rules per class are saved in the learned rule set. We next check on the classification performance based on the parameter n. Fig. 3.13

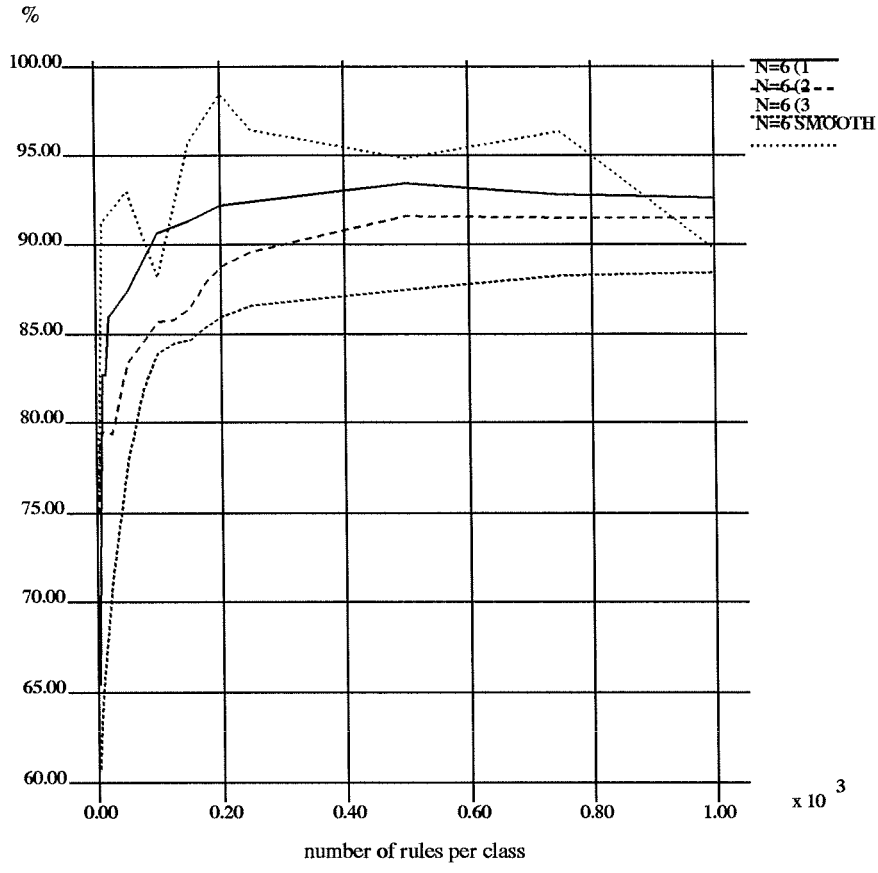


Figure 3.13: Classification performance with increasing the number of rules per class. Shown are classification curves for three 6 texture runs. The "smooth" curve is the result of first smoothing over the output maps of each 6 texture classification case and then averaging over the three cases.

displays 3 six-texture classification performance curves, with increasing the number of rules allocated per class. The fourth curve, labeled "smooth", is the result of first smoothing over the output maps of each 6 texture classification case (a majority vote is taken at a 3*3 window around each output map pixel), and then averaging over the three cases.

In all the curves plotted we get a high classification rate between 200 rules/class and around 750 rules/class. The numbers themselves are not strict, in the sense that there are many prunning techniques possible, which reduce the number of actual rules used. Some of these techniques are summarized in Appendix B.1.1. The general characteristics shown in the plot are the following:

- As long as there are a few rules per class there is already a strong jump towards

the high-performance rates.

- The performance is stable at a very wide range of rule numbers, thus the system is not overly sensitive to this parameter.

- If too many rules are allocated per class, we reach the point of overfitting the training set, and thus the reduced performance with the testing set.

- Sensitivity to Number of Input Textures - Scaling Up to 30

An interesting and important test of a system's performance is the *scalability* of the system with increasing the number of inputs. As mentioned in the introduction to this section, the literature of the past 10 to 20 years had mainly concentrated on the range of 10 texture classification. It is only in the past year that results on larger databases have been reported. We extend the results in the literature and check the scalability of the system's performance as we increase the input number to 30 textures.

In the following we present corresponding results to the ones described earlier for the 10 texture case, on the entire database of 30 textures, of Fig. 3.8. Three tables are presented: In Table 3.8 the classification performance for the $8 * 8$ resolution is given, for all the classification schemes we consider. In Table 3.9 the effect of the classification window size is shown. In Table 3.10 the effect of changing the input representation space via histogram equalization is shown.

Several things can be noted from the presented results.

1. There is a decrease in performance as we shift from the 10 texture case to the larger set of 30 textures.

2. The decrease in performance is reduced substantially (to less than 3%) as we increase the classification window size to $32 * 32$ and more so to $64 * 64$.

3. The results across the three classifiers are similar, especially with the $32 * 32$ and $64 * 64$ size windows.

4. Looking at the histogram equalization results we note the small reduction of $1 - 4\%$. This reinforces our earlier observation in the 10 texture case.

We have highlighted the result for the $64 * 64$ case which is the standard window size in the literature. As mentioned in the introduction, very few results can be found in the literature which deal with large size databases. A 25 texture case

30 TEXTURE CASE - 8*8 window	
Classification scheme	Classification accuracy
Rule-based network	**80%**
k-nn	
k=1	80.77%
k=5	81.4%
k=10	82.5%
k=20	83.3%
k=50	82.7%
k=100	82%
average result	**82.12%**
Backprop	
h=30	89.29%
h=60	89.57%
h=90	89.9%
average result	**89.58%**

Table 3.8: Classification results for 30 textures

30 TEXTURE CASE			
window-size	k-nn	Backprop	Rule-based network
8*8	82.12%	89.58%	80%
16*16	88%	93.4%	84%
32*32	96.6%	98.15%	94.4%
64*64	**95%**	**97.25%**	**97.5%**

Table 3.9: Changing window sizes

30 TEXTURE CASE - histogram equalization		
window-size	k-nn	Backprop
16*16	84%	91.27%
32*32	95%	97.4%

Table 3.10: Changing the input representation

can be found in [HO88] in which information trees are used for classification. The results quoted there vary from 70% on $20 * 20$ window sizes to 87% on $40 * 40$ size windows. The more recent results are related to the wavelet representation of textures. In [LF93] 25 textures are classified. Classification windows of size $128*128$ are used with an overlap between the extracted windows. High classification results are achieved. Another recent wavelet work is [CK93] in which 30 textures are classified. $256 * 256$ size windows are used in one experiment, $64 * 64$ windows with strong overlap are used in another. Our results compare favorably to the last two works mentioned. We cannot compare the higher-resolution classification results which we have. In addition, the proposed wavelet schemes can not be generalized to be rotation invariant. This is a major feature of our system which will be introduced in Chapters 4 and 5.

Experiment 5: Segmentation via the Recognition Process

One advantage of a high-resolution classification scheme (in our case using the $8*8$ window resolution), is the ability to combine *segmentation* of input scenes with the recognition process. This will be exemplified next using the rule-based network and will be extended as we look into real-world applications of the system, such as automated scenery analysis, in Chapter 6.

Experiment

In the training stage, a set of 6 textures was presented to the system, comprised of grass, raffia, wood, sand, herringbone weave and wool. The training input consists of a 128*128 image patch per texture. (i.e., 256 feature vectors per texture). In the testing phase, a 256*256 image is input to the system, which is a mosaic of 5 of the above textures as shown in Fig. 3.14 top left. The mosaic is comprised of grass, raffia, herringbone weave, wood and wool (center square) textures. Note that the patches forming the mosaic are *different* from the training patches, with no pixel overlap. The input poses a very difficult task which is challenging even to humans.

The test mosaic is input to the pyramid and feature vectors are extracted, corresponding to 8*8 windows in it. These vectors are quantized and labeled in the classification stage of the system, to give an output label map. Note that the $8 * 8$ classification windows are allowed to overlap boundaries of the input mosaic.

Results

In Fig. 3.14 we see the input test image (top left) followed by the labeled output map (top right) and the corresponding probability maps for the prelearned library of six textures (bottom). Based on the probability maps (with white indicating probability closer to 1) the very satisfying result of the labeled output map is achieved. The five different regions have been identified and labeled correctly (in different shades of gray) with the boundaries between the regions very strongly evident. We have achieved a segmentation of the image into homogeneous areas via the recognition process.

In this example no smoothing was performed on the output maps. This is evident looking at the output label map with isolated errors found in the interior of the mosaic and most errors located on the segmentation boundaries. The isolated errors can easily be eliminated by incorporating the 2D nature of the problem as a smoothing operation on the extracted probability maps or the final output maps.

Experiment 6: Pattern Discovery

We conclude this chapter with an additional result which demonstrates the capability of the rule-based learning system to generalize to the identification of an unknown class. In this task a presented pattern, which is not part of the prelearned library, is to be recognized as such and labeled as an unknown area of interest. This task is termed "pattern discovery" and its application is widespread, from identifying unexpected events to the selection of areas of interest in scene exploration studies. Learning the unknown is a difficult problem in which the probability estimates prove to be valuable. Our criterion for declaring an unknown class is when the sum of W_js (equation (4.5)) is negative for each class; i.e. there is negative evidence for each prelearned class. In the presented example, of Fig. 3.15, a three texture library was learned, consisting of the wood, raffia and grass textures. The input consists of wood, raffia and sand (top left). The output label map (top right) which is the result of the analysis of the respective probability maps (bottom) exhibits the accurate detection of the known raffia and wood textures, with the sand area labeled in black as an unknown class. This conclusion was based on the negative weights of evidence for each of the prelearned classes - indicated as zero probability in the corresponding probability maps. We have thus successfully analyzed the scene based on the existing source of knowledge.

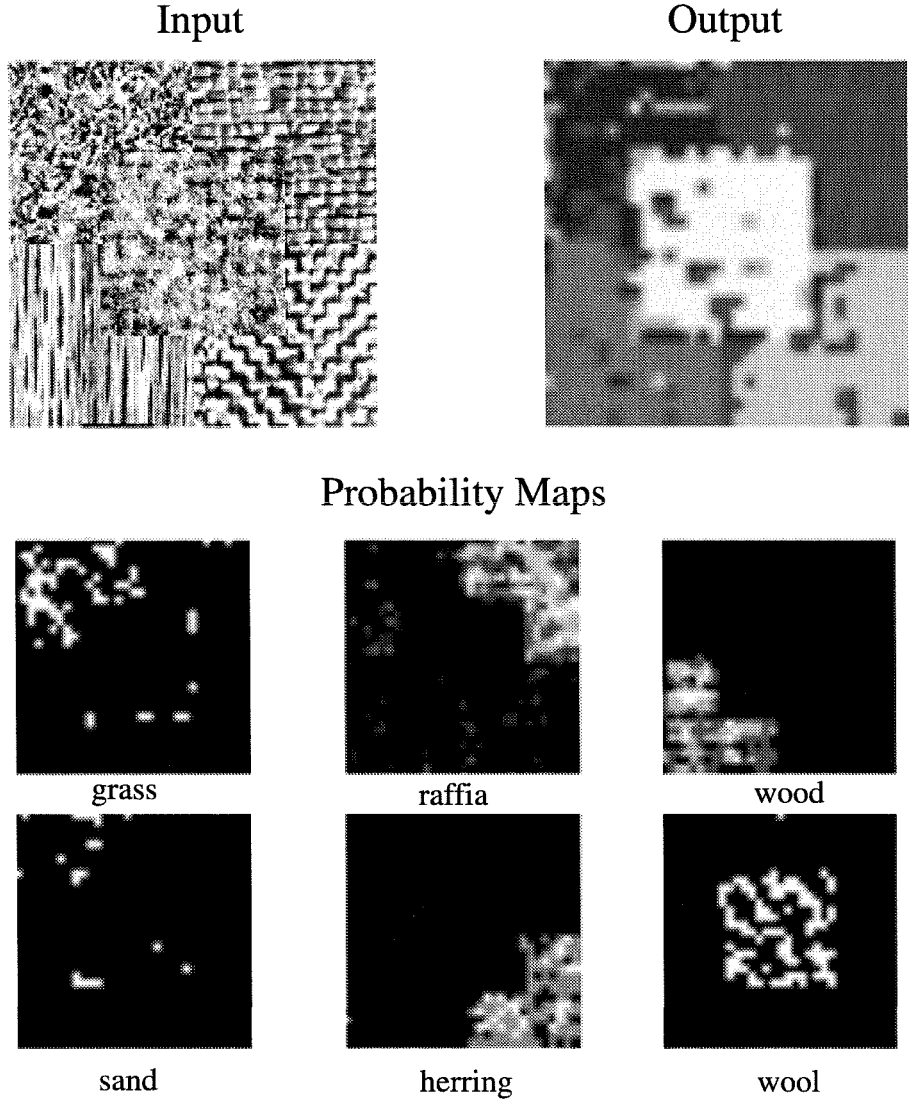


Figure 3.14: Five class natural texture classification. Input mosaic is presented (top left), followed by the labeled output map (top right) and probability maps (bottom). The probability maps correspond to a 6 - texture prelearned library, comprised of the (from top to bottom, left to right) grass, raffia, wood, sand, herringbone weave and wool textures. White areas indicate high probability. In the label map the different grey levels correspond to the 5 classes identified.

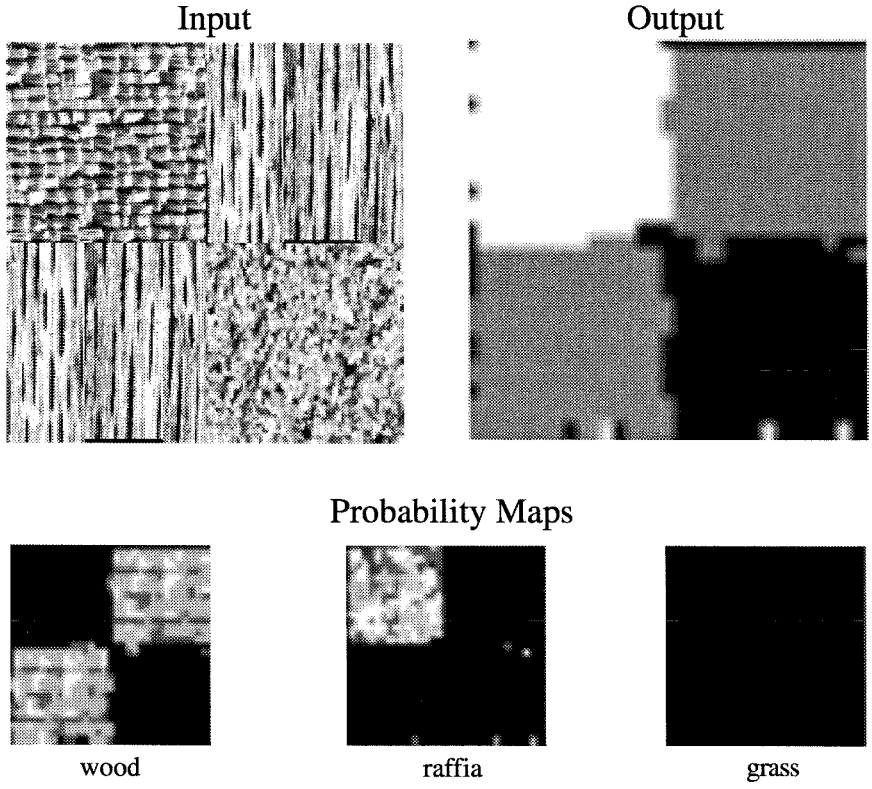


Figure 3.15: Generalization to an unknown class. Presented is the input mosaic (top left) which is comprised of the wood, raffia and sand textures, followed by the output labeled map (top right) and the probability maps (bottom). The probability maps correspond to a 3 - texture prelearned library, comprised of the (from left to right) wood, raffia and grass textures. The sand area is detected as an unknown class (labeled in black in the output labeled map) based on the negative weights of evidence for each of the prelearned classes - indicated as zero probability in the corresponding probability maps.

3.5 Summary and Discussion

We presented a texture recognition system which combines a strong input representation space, via the pyramid filtering, with learning paradigms for classification. The first stage of the system shifts the representation of the textured input from the pixel domain to the frequency-orientation space, via the multi-resolution pyramid decomposition. In the learning part of the system we focus on a rule-based classification network which we present along with the non-parametric k-nearest neighbor and Backprop learning schemes.

We have shown high-percentage classification rates for a wide variety of both structured and unstructured textures. The robustness of the system to changes in the input representation as well as the scalability of the system's performance as we increase the input space to a 30 texture database has been demonstrated. The classification results presented in this work are competitive in performance with other techniques widely used in the literature.

The strength of the system is largely due to the extracted feature set. One of the strong characteristics of the system is that unlike some of the existing techniques, the pyramid multi-resolution representation does not require a priori knowledge of the frequency content of the textures in the input image. We use the pre-defined set of 15 filters and are not required to "tweak" any of the filter's parameters in accordance with the specific inputs presented to the system.

An important issue is the resolution of the classification process. Using the pyramid representation is computationally efficient as the image is reduced in size by the filtering process. Two such reductions are used in the three-scale representation, resulting in reduced-size power maps which we classify. In our highest classification resolution each pixel that we classify represents an 8*8 local window in the original input. This high-resolution classification enables segmentation of mosaic images, where the segmentation is achieved via the recognition process. We should note that for some segmentation tasks it is sometimes desirable to increase the resolution even further (thus eliminating blockiness effects). In order to achieve even better classification accuracy, we would need to remain at the higher resolution levels of the pyramid (actually expanding the lower-resolution levels "upward"). A tradeoff exists between computational efficiency, both timewise and resourcewise, and achieving this high-accuracy performance.

Comparable results were presented across the three different classification schemes used. Some of the interesting characteristics of the rule-based network include the following: A minimal (most informative) feature set is learned. This provides a scheme to reduce the dimensionality of a given task. The extracted set of classification rules are available to the user. The system can thus enhance the user's knowledge of the input domain via its own extracted rule knowledge base. In addition, the *number* of rules required to cover the input sample space is indicative of the difficulty of the task, and reflects the relative difficulty of characterization among the texture classes. The output probability maps give more information about the decision process than does a hard-decision output. We have demonstrated the generalization capability of the system, based on the probability maps, to the identification of an unknown class, so-called "pattern discovery."

The rule-based network presented can require a long training time as it searches the input space and looks for informative rules of varied orders. Smart search algorithms and pruning algorithms are required in order to prevent the training time from growing exponentially with the number of examples presented to the system. We assume that this training phase can be done in a batch mode. The classification stage can then be achieved in real time as portions of the system are parallelizable and can be implemented in parallel hardware. We have thus automated the learning of a rule-based system while the inferencing stage can be performed in parallel when implemented on neural-network architectures.

Further investigation into the system's characteristics, both its feature-space representation, and the advantages of learning schemes, will be pursued in the following chapters. We start by proving that the pyramid filters form a steerable representation space. This will allow us to extend the recognition system to handle rotation invariant texture recognition (Chapters 4 and 5).

The application of the system to real-world problems (such as remote sensing) will be demonstrated in Chapter 6. Along with our specific interest in texture, we believe that the presented recognition system is a general one and could handle other visual modalities as well. We will touch upon this claim briefly, in the context of shape recognition (in particular, face recognition) in Chapter 6 as well.

Chapter 4
Steerability of the Pyramid Filters and Rotation Invariance

4.1 Introduction

In many image-processing tasks (such as the texture analysis, edge detection, and motion analysis tasks) it is useful to apply filters of arbitrary orientation and to examine the filter output as a function of orientation. One approach to finding the response of a filter at many orientations is to apply many versions of the same filter, which are rotated copies of each other. A more efficient approach is to apply a few filters corresponding to a few angles and interpolate between the responses. With the correct filter set and the correct interpolation rule it is possible to determine the response of a filter of arbitrary orientation without explicitly applying that filter. The term "steerable filter" has been defined in [FA91] to describe a class of filters in which a filter of arbitrary orientation is synthesized as a linear combination of a set of "basis functions". A function $f(x, y)$ is *steerable* when it can be written as a linear sum of rotated versions of itself:

$$f_\theta(x, y) = \sum_{k=1}^{M} \beta_k(\theta) f_{\theta_k}(x, y), \tag{4.1}$$

where $f_\theta(x, y)$ is $f(x, y)$ rotated through an angle θ about the origin and $\beta_k(\theta), k = 1..M$ are the M interpolation functions needed.

In this chapter we extend our analysis of the Oriented Laplacian pyramid filters, to show that the pyramid filters are a steerable set of filters [GBP+94]. In previous work, Freeman and Adelson [FA91] addressed the problem of synthesizing exactly steerable filters (via equation 4.1). Perona [Per91] addressed the problem of calculating the best steerable approximation to a given impulse response. We approach here a third related problem. It is sometimes desirable to use a particular set of filters, due to certain desired filter characteristics, computational complexity, existing hardware implementations or other constraints. The question arises: what is the best way to interpolate a given set of filters? In this chapter we present an optimal technique for deriving the set of interpolation functions (steering coefficients) for a given overcomplete discrete representation, which enable the shift to a steerable representation. We illustrate our technique using as a

sample case the oriented Laplacian pyramid.

We show how the oriented pyramid, which has 8/3 redundancy (this is a more compact representation than has previously been used in the literature [FA91], [SFAH92], [BA83]), can be transformed into a steerable one. We present the procedure to calculate the interpolation functions which give us a steerable representation.

We conclude this chapter by addressing the issue of a rotation invariant input representation via the extracted steerable representation. We present a scheme for generating rotationally-invariant feature-vectors for a given input, together with extracting the actual rotation information. Initial results for a variety of textured images will be shown throughout. These results will be extended upon in Chapter 5 in which we will proceed to use the steerability of the pyramid to achieve a large-scale rotation-invariant texture recognition system.

4.2 Interpolating in Orientation Space

In Chapter 2 we have introduced the Oriented pyramid filters which we term O_{n1} through O_{n8}, and have shown that the chosen set of 4 oriented filters per scale (together with their conjugate counterparts) span the 360° orientation space (see Fig. 2.7). In this chapter we wish to use the finite set of oriented filters to calculate the output of filters at *any* orientation in a continuum. We start by deriving the set of interpolation functions which give us the steerable pyramid. We then investigate further into some of the interpolation functions characteristics.

4.2.1 Interpolation Function Derivation

Having arrived at the set of oriented pyramid filters, O_{n1} through O_{n8}, we next wish to define interpolation functions (or steering coefficients) which allow us to use the finite set of eight filters (per scale) to synthesize oriented filters across the entire orientation space. Note that in this section we assume the input image to the pyramid to be a delta function, $G_0(x, y) = \delta(x, y)$. Let $\beta_{n,k}(\theta)$, $k = 1..8$, represent the interpolation coefficients in orientation space. We wish to calculate the filter output for any given angle θ, which we define as $\hat{F}_\theta(x)$, via a linear interpolation scheme as follows:

$$\hat{F}_{n,\theta}(x) = \sum_{k=1}^{8} \beta_{n,k}(\theta) O_{n,k}(x). \tag{4.2}$$

For clarity purposes we hereon avoid using the scale notation, with the understanding that the following derivation is performed at each scale, n, independently.

Our goal is to minimize the error between the filter output, $F_\theta(x)$, and the interpolated output, $\hat{F}_\theta(x)$, (in space for a particular orientation θ):

$$min_{\beta_{n,k}} \|F_\theta(x) - \hat{F}_\theta(x)\|^2_{\Re^2}. \tag{4.3}$$

We have

$$\|F_\theta(x) - \hat{F}_\theta(x)\|^2 = \|F_\theta(x)\|^2 + \|\hat{F}_\theta(x)\|^2 - 2\langle F_\theta(x), \hat{F}_\theta(x)\rangle, \tag{4.4}$$

where

$$\begin{aligned} \|\hat{F}_\theta(x)\|^2 &= \sum_{h,k} \langle \beta_k(\theta) O_k(x), \beta_h(\theta) O_h(x)\rangle \\ &= \sum_{h,k} \beta_h \beta_k \langle O_h, O_k\rangle_{\Re^2} \end{aligned} \tag{4.5}$$

and

$$\begin{aligned} \langle F_\theta(x), \hat{F}_\theta(x)\rangle &= \sum_h \beta_h \langle O_h(x), F_\theta(x)\rangle \\ &= \sum_h \beta_h \langle F_\theta, O_h\rangle_{\Re^2} \\ &= \sum_h \beta_h \cdot \Gamma_h(\theta) \end{aligned} \tag{4.6}$$

with $\Gamma_h(\theta) = \langle F_\theta, O_h\rangle_{\Re^2}$.

Using (4.4-4.6), we need to minimize the following expression with respect to β:

$$\underbrace{\sum_{h,k} \beta_h \cdot \beta_k \langle O_h, O_k\rangle_{\Re^2}}_{a} - \underbrace{2 \sum_h \beta_h \cdot \Gamma_h(\theta)}_{b}. \tag{4.7}$$

The derivative with respect to β_z of the left term (a) gives : $2\sum_k \beta_k \langle O_z, O_k\rangle$.

The derivative of the right term (b) gives: $-2\Gamma_z(\theta)$.

Equating the sum of the above two terms to zero leads to the following equation for the β's:

$$\sum_k \langle O_z, O_k\rangle_{\Re^2} \beta_k = \Gamma_z(\theta) \qquad z = 1..N, k = 1..N. \tag{4.8}$$

In matrix form we have:

$$\mathbf{O}\beta = \mathbf{\Gamma}, \tag{4.9}$$

with $\mathbf{O} = \langle O_h, O_k \rangle_{\Re^2}$, β = a column of the β_k's, $k = 1..N$, and $\mathbf{\Gamma}$ = a column of the Γ_z's, $z = 1..N$.

In the orthonormal case, $\mathbf{O}$ is a diagonal (identity) matrix since $\langle O_h, O_k \rangle = \delta_{hk}$.

In that case from 4.8 we get:

$$\beta_h(\theta) = \Gamma_h(\theta) = \langle F_\theta, O_h \rangle_{\Re^2}. \tag{4.10}$$

For the nonorthonormal case, the solution requires more computation – one method would be Gauss elimination method, the other would be to decompose $\mathbf{O}$ by SVD to $\mathbf{U\Sigma V}^T$ and use this to calculate $\mathbf{O}^{-1}$. Here, $\mathbf{UU}^T = \mathbf{I}$ and $\mathbf{VV}^T = \mathbf{I}$. The inverse matrix, $\mathbf{O}^{-1}$, can be found as:

$$\mathbf{O}^{-1} = \mathbf{V}[diag(1/(\lambda_j)]\mathbf{U}^T. \tag{4.11}$$

The solution for β can now be extracted as:

$$\beta = \mathbf{V}[diag(1/(\lambda_j)]\mathbf{U}^T \cdot \mathbf{\Gamma}, \tag{4.12}$$

where in the case of a zero eigenvalue, $\lambda_j = 0$, the corresponding $1/\lambda_j$ in $\mathbf{\Sigma}^{-1}$ gets replaced by a zero. The above scenario takes care of all possible $\mathbf{O}$ matrices, even if the matrix is not full rank. Overall, if $\mathbf{\Gamma}$ is in the range of $\mathbf{O}$ then the extracted β functions are exact. If $\mathbf{\Gamma}$ is *not* in the range then the β functions are the closest we can find in least-squares sense; i.e., minimizing $|\mathbf{O} \cdot \beta - \mathbf{\Gamma}|$.

Using the eight 45° bandwidth oriented filters, O_1 through O_8, we extract the eight steering coefficients (β_k for $k = 1..8$), as outlined above. A plot of $\beta_k(\theta)$ over the range $0° - 360°$ is shown in Fig. 4.1. Note that at each 45° increment, one of the $\beta_k(\theta)$'s peaks at 1, and the rest of the $\beta_k(\theta)$'s pass through zero. The curves for each $\beta_k(\theta)$ are cyclicly shifted copies of one another; for clarity, Fig. 4.1 bottom highlights the curve for $k = 5$.

With the interpolation functions in hand, we can now go back and calculate the oriented filters, $\hat{F}_\theta$, from the finite set of oriented filters that we hold, O_k, as:

$$\hat{F}_\theta = \sum_k \beta_k(\theta) O_k \tag{4.13}$$

across the continuous orientation space, $\theta = 0° - 360°$. Fig. 4.2 shows the percent error, $E(\theta)$, in the reconstruction of the oriented filters, for $\theta = 0° - 360°$ with steps of 5°. Here

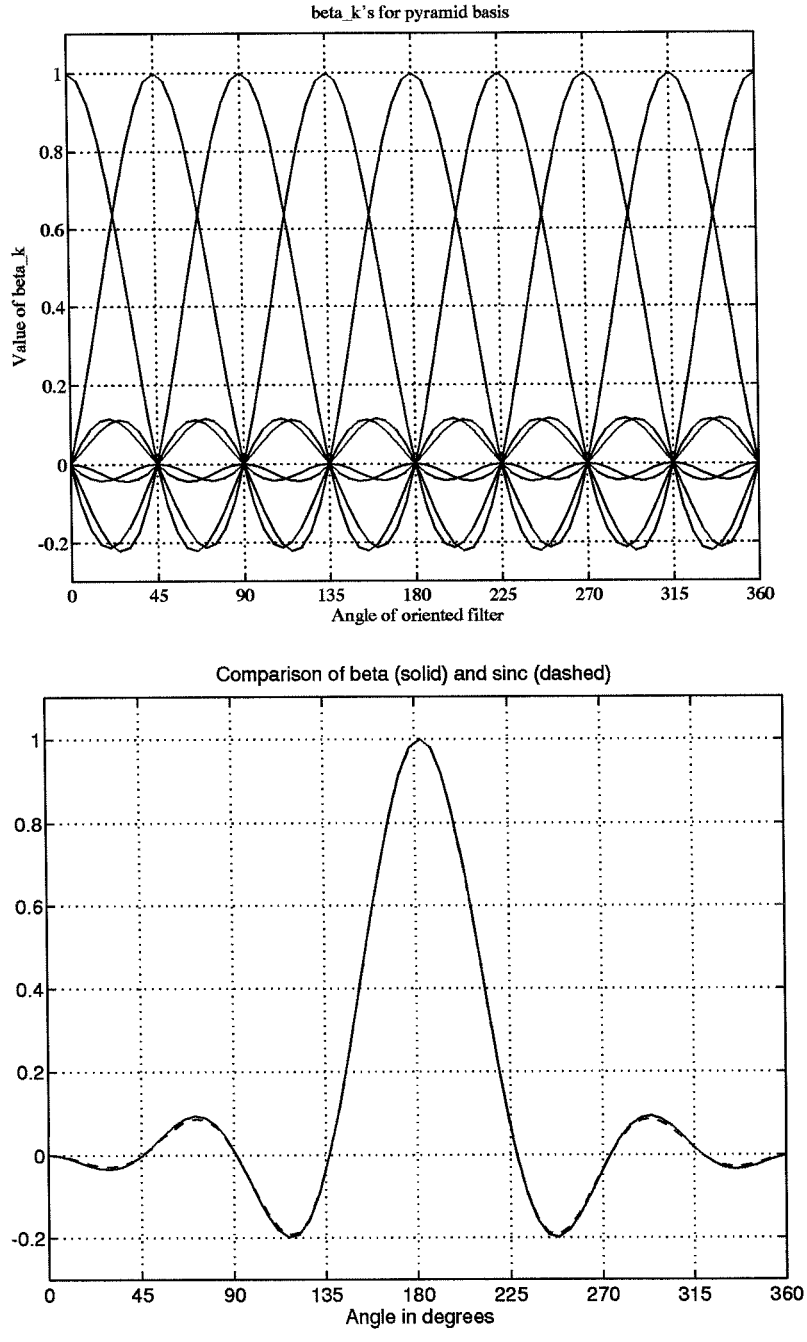


Figure 4.1: Top - Plot of the eight interpolation functions, $\beta_k, k = 1..8$. Bottom - Highlight of one characteristic interpolation function (solid), as compared with the Sinc function (dashed).

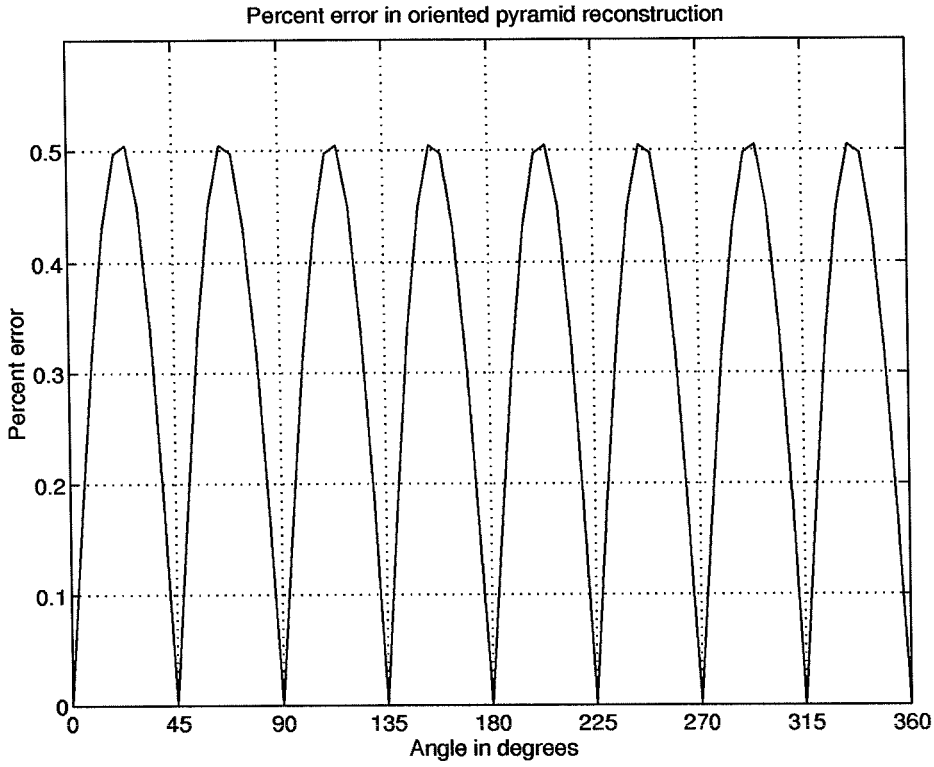


Figure 4.2: Percent error in the reconstruction of oriented filters across the continuous orientation space, $\theta = 0 - 360$ degrees, from the finite filter set, $O_k, k = 1..8$.

the interpolation error E is defined as:

$$E(\theta) = \frac{\left\| F_\theta - \hat{F}_\theta \right\|}{\|F_\theta\|} \times 100. \tag{4.14}$$

Note that the peak error is less than 1% (around 0.5%). This is in agreement with the SVD bound found in Chapter 2. We have thus completed the proof of the pyramid steerability.

An alternative scheme to the interpolation functions derivation of above, which does not involve matrix inversions, uses the Gram-Schmidt orthogonalization process. This scheme is outlined in Appendix C.1.

4.2.2 Steerability of the Filter Powers

We have so far shown the steerability of the oriented filters (equation 4.2). We next need to shift to investigate the behavior of the oriented filter *powers* which form the actual input representation. The power maps form a pyramid of the local statistics of the oriented pyramid's coefficients, which characterize the image local-area response to the different

orientations and frequencies (see Chapter 3). The exact interpolation equation for the filter powers is complex, and will not be derived here.

Given the fact that the energy is lowpass in orientation we hereon make the approximation that the filter output powers can be interpolated with the (sinc-like) β functions (this is confirmed by empirical observations):

$$\hat{P}_\theta \approx \sum_k \beta_k(\theta) P_k. \tag{4.15}$$

Here, $P_k, k = 1..4$, are the 4 oriented power components, $P_{n\alpha}, \alpha = 1..4$, and $P_k, k = 5..8$ is the duplicate set representing the power components of the conjugate counterparts. $\hat{P}_\theta$ represents the estimated power map for the texture rotated at an arbitrary angle, θ.

We test the estimation accuracy on a few texture examples. Fig. 4.3 presents the estimation error, $E(\theta)$, across orientation space (steps of 5°), for a set of 5 textures. Here:

$$E(\theta) = \frac{\left\| P_\theta - \hat{P}_\theta \right\|}{\|P_\theta\|} \times 100. \tag{4.16}$$

with P_θ representing the actual power map extracted from the input texture rotated to θ, and $\hat{P}_\theta$ representing the estimated response based on the original, nonrotated power maps. The error is less than 3%. These results demonstrate that the finite set of oriented filters which we chose for our representation gives us a steerable pyramid and indicates the validity of the interpolation functions in a real application. Moreover, the results demonstrate that the power pyramid is steerable, thus validating equation 4.15.

4.3 Achieving a Rotation-Invariant Representation via the Steerable Pyramid

For a given input (texture) we define a *feature curve* (per scale) across orientation space, $f_c(\theta)$, as the texture's response to any oriented filter in the 360° space (using symmetry considerations we will concentrate on the 180° space). Using the steerability property we note that the four components of the feature *vector*, i.e, P_{n1}, P_{n2}, P_{n3} and P_{n4}, allow us to reconstruct the continuous curve with a 45° sampling period.

As an input texture is rotated, its feature curve, $f_c(\theta)$, shifts across the orientation axis. Alternatively we can visualize the sample points cycling along the continuous curve.

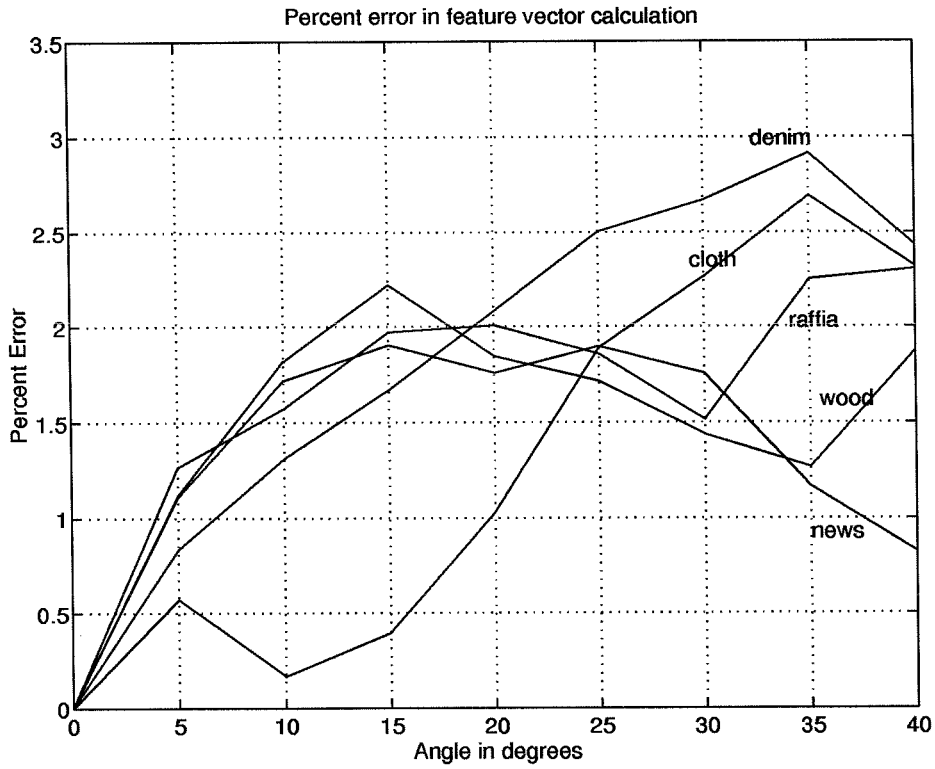


Figure 4.3: Percent error in the calculation of characteristic curves, 5 texture case.

It is our goal to find a rotation-invariant representation for the sampled curve. We will next describe how the Discrete Fourier Transform (DFT) can be used for this task.

4.3.1 Use of the DFT Representation for Rotation Invariance

Let $f(n), n = 0..3$, denote a sequence of four points taken from a single scale in a given feature vector. We define a *companion* feature vector $\hat{f}(k)$ to be the Discrete Fourier Transform of $f(n)$ as follows [1]:

$$\hat{f}(k) = \sum_{n=0}^{3} f(n) e^{-i\pi nk/2} \qquad k = 0, 1, 2, 3. \tag{4.17}$$

For illustrative purposes the above summation can be rewritten in the form of a matrix product, as follows:

$$\begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -i & -1 & i \\ 1 & -1 & 1 & -1 \\ 1 & i & -1 & -i \end{bmatrix} \begin{bmatrix} f(0) \\ f(1) \\ f(2) \\ f(3) \end{bmatrix} = \begin{bmatrix} \hat{f}(0) \\ \hat{f}(1) \\ \hat{f}(2) \\ \hat{f}(3) \end{bmatrix}. \tag{4.18}$$

In this form, it is clear that each term in the transformed sequence of points evaluates to:

$$\begin{aligned} \hat{f}(0) &= f(0) + f(1) + f(2) + f(3) \\ \hat{f}(1) &= (f(0) - f(2)) + i(f(3) - f(1)) \\ \hat{f}(2) &= f(0) - f(1) + f(2) - f(3) \\ \hat{f}(3) &= (f(0) - f(2)) - i(f(3) - f(1)). \end{aligned} \tag{4.19}$$

The four values of $\hat{f}(k)$ provide us the following information:

- $\hat{f}(0)$ is the DC component of $f(n)$, i.e., $\sum_n f(n)$,
- $|\hat{f}(1)| = |\hat{f}(3)|$ is the magnitude of the first harmonic,
- $\arg[\hat{f}(1)] = -\arg[\hat{f}(3)]$ is the phase of the first harmonic, and
- $\hat{f}(2)$ is the Nyquist component [2]

[1] The complete feature vector includes the DFT components for each scale together with the non-oriented components, L_n.

[2] We use the term "Nyquist" component to refer to the component of the highest harmonic present in a sequence, as dictated by the sampling rate.

Using the extracted coefficients above, we can represent the original feature curve, $f_c\theta$ as:

$$f_c(\theta) = A + B\cos(\theta + C) + D \tag{4.20}$$

with the coefficients given by

$$A = \hat{f}(0), B = |\hat{f}(1)|, C = \arg[\hat{f}(1)], D = \hat{f}(2). \tag{4.21}$$

We note that A and B do not change as a result of rotation, while $C/2$ is equal to the rotation angle on the input (The division by two is necessary since $f(n)$ goes through *two* complete cycles during a rotation of the input image by 360°). In this case we do not have a high-enough sampling rate to extract the phase component of the second harmonic. The coefficient D is therefore not rotation invariant. It will tend be very small and does not represent useful information with rotated inputs.

To begin investigating the prospect of an invariant feature waveform, the feature-vectors for an ideal sinusoidal grating texture at orientations from 0° to 45° with steps of 5° were set aside, for a total of 10 feature-vectors. Then a companion set of 10 feature-vectors was formed. Fig. 4.4 (top) shows magnitudes of the 10 DFT's for the ideal sinusoidal grating texture, and Fig. 4.4 (bottom) shows the phase of the DFT's. The magnitudes overlap onto a single characteristic curve. With regard to the phase, either $\hat{f}(1)$ or $\hat{f}(3)$ can be inspected to determine the amount of rotation on the input, as shown in Fig. 4.4. Only the positive and/or negative first harmonics $\hat{f}(1)$ and $\hat{f}(3)$ will lend insight to occurrences of a shift in the feature waveform resulting from a rotation of the input texture. Fig. 4.5 shows the DFT magnitudes for the denim texture. We note the variation in the Nyquist component, $\hat{f}(2)$, with the rotation of the input texture.

4.3.2 Derotation of the Feature Vectors

Given the rotation/phase-shift of a given (test) feature vector, we will now describe how to "derotate" the feature vector to approximate the response to the corresponding input texture which was prelearned and stored in the database. We will first treat the general case of a length-N bandlimited sequence $x(n)$ and then consider the special case of $N = 4$.

The General Case

Consider an arbitrary bandlimited, real, positive signal $x(t)$ which has been sampled at the Nyquist rate to yield a length N sequence $x(n)$ (where N is even). In our case $x(n)$

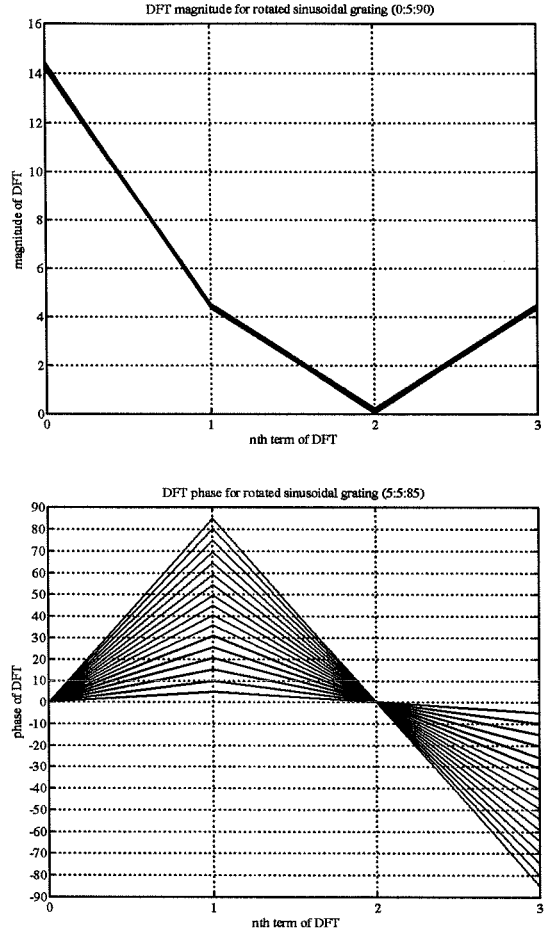


Figure 4.4: Top : DFT magnitudes for 10 rotated ideal sinusoidal-grating textures. Bottom: DFT phase for 10 rotated ideal sinusoidal-grating textures.

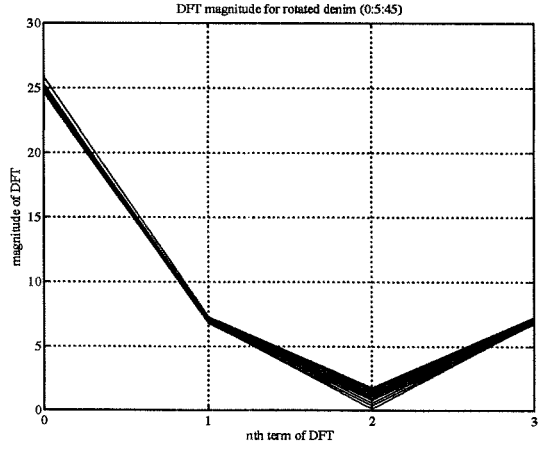


Figure 4.5: DFT magnitudes for 10 rotated versions of the denim texture

represents a sequence of oriented filter output powers for a local window in an input image.

This sequence can be expressed as

$$x(n) = \frac{1}{N} \sum_{k=0}^{N-1} \hat{x}(k) e^{i2\pi nk/N} \qquad n = 0, \ldots, N-1, \tag{4.22}$$

where $\hat{x}(k)$ is the DFT of $x(n)$. Since $x(n)$ has N evenly spaced samples, we can regard it to be samples of the curve $x(t)$ with a sampling period of $180°/N$. Making use of the conjugate-symmetry of $\hat{x}(k)$ (since $x(n)$ is real), we can rewrite equation 4.22 as

$$\begin{aligned} x(n) \;=\; & \frac{\hat{x}(0)}{N} + \frac{2}{N} \sum_{k=1}^{N/2-1} |\hat{x}(k)| \cos\left(\frac{2\pi nk}{N} + \arg[\hat{x}(k)]\right) + \\ & \frac{1}{N} \left|\hat{x}\left(\frac{N}{2}\right)\right| \cos(\pi n) \quad n = 0, \ldots, N-1. \end{aligned} \tag{4.23}$$

We will again consider what each component of $\hat{x}(k)$ represents in terms of orientation information. The $\hat{x}(0)$ term is the DC component of $x(n)$, i.e., $\sum_n x(n)$ (identical to $\hat{f}(0)$ in equation 4.19). We should expect that the sum of the filter powers remain invariant to rotations of input patterns. The terms that are of interest to us in terms of orientation and rotation information, however, are the harmonic terms $\hat{x}(k), k \neq 0$.

$\hat{x}(1)$ represents the first harmonic. A rotation of an input pattern by 180° would result in one full cycle of this harmonic. The next harmonic, $k = 2$, is tuned to patterns with two perpendicular orientations, such a "+" symbol, and would go through one full cycle during a 90° rotation of such a pattern. In general, the strength of the kth harmonic indicates the presence of k simultaneous orientations (evenly spaced in angle) and cycles once during each rotation of the input pattern by $180°/k$. The phase of the kth harmonic, meanwhile, indicates what the rotation of the pattern is, modulo $180°/k$. For example, the phase of the first harmonic for a wood texture (which is a single orientation pattern) could be inspected to detect a rotation of the woodgrain relative to vertical.

The last harmonic (corresponding to the Nyquist frequency), $k = N/2$, does not have useful phase information. A higher sampling rate would be necessary to obtain reliable phase and magnitude information for this harmonic. (For this reason, the summation in (4.23) can actually be written to only go from 1 to $N/2 - 1$ for tasks which involve rotated inputs).

As an example, consider the case for $N = 8$. Here we have

$$f_c(\theta) = A + B\cos(\theta + C) + D\cos(2\theta + E) + F\cos(3\theta + G) + H. \tag{4.24}$$

For this choice of N, we can detect up to three multiple orientations. The first, second and third harmonics would be tuned to the configurations "$-$", "$+$" and "$*$", respectively. The phases of these harmonics would indicate rotations of these multiple-orientation elements.

Derotation of the feature vectors is described next. We know that shift of $x(n)$ by an integer amount n_o (where $|n_o| < N$) results in a modulation of $\hat{x}(k)$:

$$x(n - n_o) \Longleftrightarrow e^{-i2\pi n_o k/N}\hat{x}(k) \tag{4.25}$$

that is, if we let $x_r(n) = x(n - n_o)$,

$$x_r(n) = \frac{1}{N}\sum_{n=0}^{N-1} e^{-i2\pi n_o k/N}\hat{x}(k)e^{i2\pi nk/N} \qquad n = 0, \ldots, N-1. \tag{4.26}$$

In general, this property only holds for integer n_o, but since $x(n)$ is bandlimited, the following relation correctly interpolates the samples of $x(n)$ for *any* shift n_o:

$$x_r(n) = \frac{1}{N}\sum_{n=-N/2+1}^{N/2} e^{-i2\pi n_o k/N}\hat{x}(k)e^{i2\pi nk/N} \qquad n = 0, \ldots, N-1. \tag{4.27}$$

The length N interval of summation has been changed here to be symmetric about zero to maintain conjugate symmetry between the positive and negative harmonics.

The algorithm for derotation follows. One harmonic is chosen from which we extract the phase. We then demodulate the phase components of all the harmonics present by that phase. Inversing the DFT encoding, we get back the desired feature vector, which can now be used in the classification task. The block-diagram for the derotation encoding scheme is shown in Fig. 4.6.

The Case for $N = 4$

Now consider the specific case when $N = 4$. Let $f_\phi(n)$ denote a prediction of $f(n)$ derotated by ϕ. We can relate this to the general case by letting $n_o = \phi/45°$, so that when ϕ is an integer multiple of 45°, n_o will be an integer. We can then predict f_ϕ using the formula

$$f_\phi(n) = \frac{1}{4}\sum_{n=-1}^{1} e^{-i2\pi k(2\phi/45°)}\hat{f}(k)e^{i\pi nk/2} \qquad n = 0, 1, 2, 3. \tag{4.28}$$

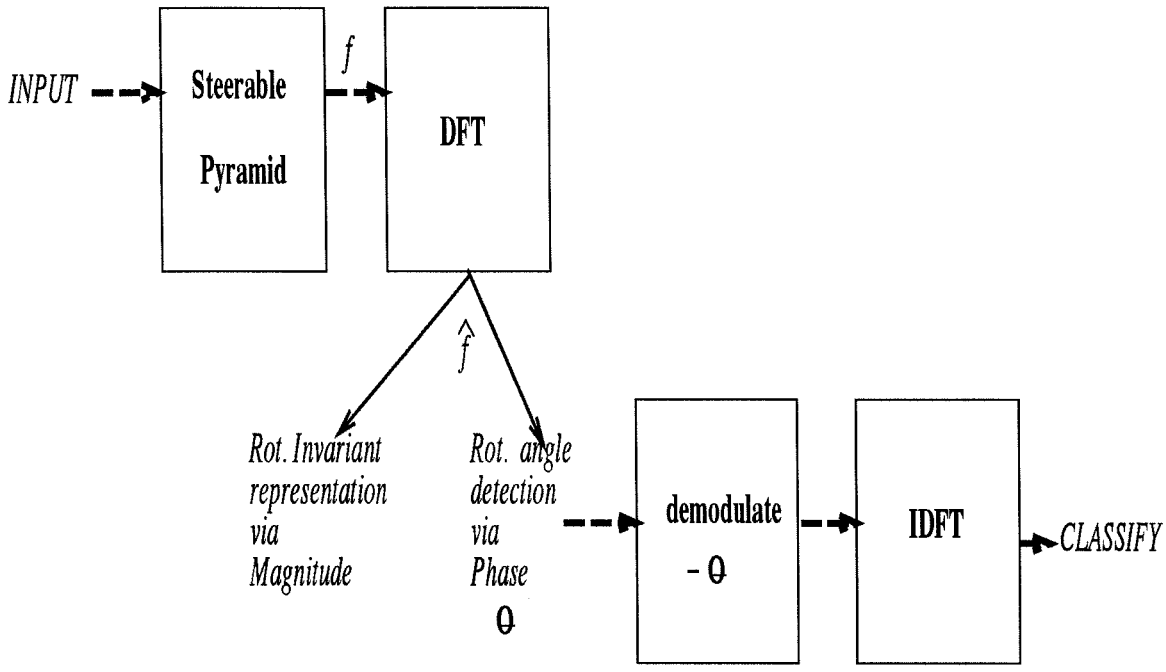


Figure 4.6: Block-diagram of the derotation-encoding scheme

Notice that ϕ appears multiplied by 2 here; this is again due to the fact that one entire rotation of an input texture by 360° results in *two* complete cycles of the values in $f(n)$.

Returning now to the task of derotation, any phase contained in the harmonics $\hat{f}(-1)$ and $\hat{f}(1)$ is zeroed by applying a modulator at a particular frequency. In this specific case of $N = 4$, zeroing the phase using demodulation is equivalent to simply taking the magnitude of each component in $\hat{f}(k)$. Recall that the DC component $\hat{f}(0)$ is necessarily positive (zero phase) since $f(n) \geq 0$ for all n, so it is unchanged when its magnitude is taken.

To summarize, in the case of $N = 4$, if we wish to predict the values of the zero-phase/derotated feature vector $f_\phi(n)$ using the information provided $f(n)$, we need only DFT $f(n)$, take its magnitude, and IDFT. Recall that the phase information is still available in $\arg[\hat{f}(-1)]$ and $\arg[\hat{f}(1)]$ should it be necessary to know the rotation angle of the input texture.

Summarizing the number of informative parameters in the texture's representation, we note the following: For an original K-point representation we have K informative values. In order to consider the rotation-invariance issue we lose 2 parameters, one is the Nyquist

parameter and the second one is the phase of the harmonic we chose as our reference harmonic - this phase is zeroed out via the modulation. We thus have $K-2$ parameters. Using the DFT magnitudes as the invariant representation we use: $1+(K-2)/2$ parameters.

We note that for the $K=4$ case, we get the same amount of information out from DFT encoding as with the demodulation procedure. In general, for $K>4$, more information is preserved by demodulating the feature vectors, as described above.

4.4 Initial Rotation Invariant Texture Recognition Results

We present a sample of results to validate the above analysis in a real-world noisy domain. A full-scope system will be presented in Chapter 5.

We present results of applying the above analysis to a 10-texture recognition task. The test consists of presenting different 128×128 images from the input set, with each image arbitrarily rotated at one of 5 angles: $(0, 10, 20, 30, 40)$ degrees. In the recognition process, feature-vectors are extracted and each component is averaged over the entire image, to produce one representative feature-vector per input. The extracted feature-vector, f, is next used to generate the companion feature-vector, $\hat{f}$, via the DFT transformation of the previous section (equation 4.17).

For each of the 10 textures we investigate the *magnitude* deviation of the representative feature-vector, $\hat{f}$, as the input texture is rotated. We compare the standard deviation within each class, c_i (in the 15 dimensional space), to the average (and minimum) distance between the mean of class c_i and the means of all other texture classes c_j, $j \neq i$; i.e., the average (/min) interclass distance. This is shown in the following table:

texture class	innerclass std.	avg. interclass distance	min. interclass distance
bark	0.99	10.89	5.92
calf	2.62	12.56	6.37
cloth	3.84	10.29	0.74
cardboard	2.76	12.67	6.45
denim	1.27	14.84	6.37
grass	1.33	18.17	7.86
pig	0.87	10.32	0.74
raffia	0.97	9.61	5.92
water	0.65	15.54	8.11
wood	0.96	10.59	6.08

In the above results we observe more than a factor of 10 difference between the innerclass and average interclass distances, for most textures. Looking at the minimum interclass distance for each texture, we again see that it is much larger than the innerclass standard deviation, except for the cloth and pig texture pair, for which the representative feature-vector means are very close. This difficulty is inherent in the similarity of the textures. The small innerclass standard-deviation strongly indicates the consistency of the DFT magnitude representation; i.e., the invariance of the response with the rotation of the input textures. To make this claim stronger, we next use the K-nearest-neighbor classification scheme on the set of 10 textures above. In the training stage, we use 16 128×128 examples per texture class, with no rotation. The test set consists of a new set of textures, which are rotated arbitrarily in one of the 5 angles: $(0, 10, 20, 30, 40)$ degrees. In this 10 texture recognition case we get 100% correct classification.

Once the identity is found we utilize the *phase* information from the DFT representation, to estimate the orientation of the test input, relative to the original texture from the training set (as described in section 4.3.1). Here we are interested in the error, in degrees, between the true rotation angle and the estimated orientation angle. The mean error, across the 5 rotation angles and for each of the 10 textures, is depicted in the following table:

texture class	mean error (in degrees)
bark	1.54
calf	0.57
cloth	1.56
cardboard	0.83
denim	0.76
grass	1.71
pig	0.46
raffia	0.37
water	0.26
wood	0.36

The average rotation-angle estimation error for the 10 textures is 0.84 degrees.

Both the perfect class identification and the high-resolution orientation estimation, as presented above, are very encouraging results in the difficult domain of rotation invariant natural texture identification.

4.5 Discussions and Conclusions

In this chapter we have presented an optimal technique for deriving the set of interpolation functions (or steering coefficients) which enable us to convert a given overcomplete oriented filter set into a steerable representation. We have shown the general concepts in the specific case of the Oriented Laplacian pyramid. We described the characteristics of the 8/3 redundant pyramid and have shown that the pyramid is steerable by defining a set of eight 45° bandwidth oriented kernels and deriving the corresponding steering coefficients. Properties of the kernels and interpolation functions have been investigated. Finally, we demonstrated highly encouraging results in using the pyramid for defining a rotation-invariant texture representation.

We have so far addressed the issue of one dominant orientation. In the case of multiple orientations at a single location (such as in a cross pattern) we need additional harmonics in our representation, or additional filters. Thus, to handle multiple orientations, we extend the above analysis, with a larger set of (narrower frequency tuned) filters. A similar framework to the one presented here can be applied next to scale invariance. The combination with scale is currently being investigated. In the following chapter the results presented here are utilized to achieve a state-of-the-art rotation-invariant texture recognition system.

Chapter 5
A Rotation-Invariant Texture Recognition System and Orientation Estimation

5.1 Introduction

We are now in the position to utilize the rotation-invariant representation of Chapter 4 to present state-of-the-art results on large database rotation-invariant texture recognition. Here we are interested in learning a set of textured inputs, following which we would like to recognize new test inputs as belonging to one of the prelearned classes, even if the new input is rotated relative to the original input. Furthermore, we wish to state with high accuracy the *orientation* of the test input relative to the original one [GBPG94].

We start by revisiting the texture recognition literature (and other general recognition literature), as we look for rotation-invariant recognition systems. In section 5.3 we demonstrate the inadequacy of a non rotation-invariant recognition system to handle rotated input data. We suggest the use of the steerable pyramid of oriented filters for generating rotation-invariant features. We demonstrate our ideas with experiments on rotation-invariant recognition (section 5.4) and on orientation estimation (section 5.5). A summary of the presented results and discussion conclude this chapter.

5.2 Background and Motivation

A major challenge in the texture recognition arena is a shift to rotation and scale invariant recognition systems. Having an invariant classifier is of great importance in object recognition from texture and inspection applications, where controlling the environment to ensure that the samples tested have the same orientation and scale as the training samples is either costly, difficult or altogether impossible. In this work we concentrate on the rotation invariance issue. Most of the methods presented in the literature address the classification problem under the assumption that the test samples from a given texture possess the *same* orientation as the training samples. Most methods therefore perform poorly when the orientation is quite *different* from the orientation of the training samples

(as will be exemplified in section 5.3). We revisit the texture recognition literature, with special attention to texture-invariant recognition.

In Chapter 3 we have reviewed the variety of structural and statistical approaches which have been adopted in the texture recognition task. The *structural* approach assumes the texture to be formed of primitives following a placement rule. This structural approach suffers from the complications of determining the primitives and placement rules that operate on these primitives. As a result, textures suitable for structural analysis have been confined to quite regular textures. Some of the structural approaches are rotation-invariant, especially with macrotextures [VNP86]. However, they are usually very complex and do not perform well for unstructured random microtextures. In the *statistical* approach the texture is regarded as a sample from a probability distribution on the image space and defined by a stochastic model or characterized by a set of statistical features. The most commonly used technique is the gray level co-occurence matrix and its variations [Har79, DJA79]. Some problems associated with the co-occurrences matrices and gray-level run-length (GLRL) statistics [Gal74, CH80] are the following: 1) They require a fair amount of computation - many co-occurence matrices need to be computed; 2) in order to reduce the dimensionality of the feature vector detailed first-stage classification experiments are to be performed to determine the best reduced set of features to use for successful classification; 3) the features are *not* invariant to rotation (except to multiples of 45°) or scale change in the texture. There have been attempts at making rotation-invariant features by averaging over directional matrices (such as the four GLCM matrices computed for $0°, 45°, 90°$ and 135° [DJA79]). In this case, in addition to the remaining problems outlined above, we lose the orientation information, which is one of the major features in the work presented here.

One model that has been well investigated for modeling the texture as a stochastic process is the Markov random field model (MRF) (e.g., [CJ83, KCK82]). Here again, although high classification results have been achieved on a large family of textures, most of the works are not rotation or scale invariant. One work that *is* well known for handling the rotation-invariance issue is [KK86]. In this work a new random field model model called "circular autoregressive" is presented whose parameters could be taken as rotation-invariant features for classification. A high-accuracy result is reported. The model however suffers from several limitations: 1) Interpolation is required for values of the circular

autoregressive field that do not fall on the rectangular grid. 2) A small database of textures is used (9,12 Brodatz textures), with a small set of samples per texture. 3) In several of the experiments the training actually includes all the rotations of the test set. 4) There is no immediate generalization to scale invariance. Another, more recent work that attempts to classify rotated and scaled textured images using Gaussian MRF models is [CFP91]. Strong classification results are presented with high-accuracy in the parameter estimation. Again, the results are shown on only a small database of 9 Brodatz textures, with a single $64 * 64$ image patch used for training and another single $64 * 64$ image patch used in testing. The scalability of the system to a larger database is not shown. In addition, only 2 orientations (0° and 60°) and 3 scales are actually used and it is not clear how the system can cope with higher resolution of the rotation angles. Classification of rotated and textured images based on Law's texture masks [Law80], is presented in [YC93]. Whereas Law's masks are fixed and not invariant, the classifier used in this work is the texture-energy associated with a mask that has been "tuned" to be both discriminant between different textures and invariant to rotation and scale changes. 15 Brodatz textures are used in this work, with 45° steps in orientation, and the three scales: $(1:2), (1:1), (2:1)$ [1].

Rotation and scale-invariant pattern recognition systems can also be found in the *learning* literature (e.g., [WWB88, PL92, FOTK92]). Most of the works found in the neural-network literature which try to "learn" the invariances, require special architectures or special constraints on the weights of the networks. In [FOTK92] a *circular* array of input neurons and corresponding weights is proposed. The responses of the cyclic-shifted input is summed thus allowing for rotation-invariance, with a resolution depending on the number of neurons in the input circular array (e.g., 12 neurons are needed for 30° resolution,24 for 15° and so forth). This architecture is applied successfully to coin recognition. A similar, more primitive concept was proposed earlier in [WWB88]. There, only 90° resolution is actually detected. A different approach has the invariance built into the architecture of the network through the imposition of appropriate constraints on the synaptic weights (e.g., so-called "Higher-order neural-networks" [PL92]). In these architectures special care needs to be incorporated to reduce the size of the network and the number of required

[1] Note that 45° increments is a simple case of *cyclic-shifting* the attribute values across the feature vector.

weights. Common features in the above-mentioned works is that once a certain network is formulated, it is restricted to recognize rotated input patterns at a specific rotation resolution. In addition, the actual rotation-angle information of the input is lost.

Finally, some very recent published works which involve wavelet decomposition of the input domain (e.g., [CK93, LF93]) have shown good classification results on large databases of textures. These results are very encouraging in the field of large database texture recognition. Still, one of the major drawbacks of the wavelet transforms (critically sampled Subband transforms) is their lack of translation invariance and in the 2D domain, their instability with regard to rotations of the input signal [SFAH92]. Overall, these works can *not* be generalized to include rotation invariance.

Summarizing the above-presented diversity in the literature we note that most of the results are given for a small set of input textures, and furthermore results are usually given for a discrete set of rotation angles with no generalization capability to the continuum of orientations. An additional important point is that many schemes shift to an invariant representation domain in which rotation information is eliminated altogether in order to achieve the rotation-invariance goal. In this chapter we show that our system can be generalized to become a rotation-invariant texture recognition system. Via the steerability characteristic we will demonstrate high classification rates on a large database of natural textures. In addition, the recognition system extracts with high accuracy the rotation-angle of the test image relative to the corresponding prelearned image in the database.

5.3 The Need for Rotation-Invariant Recognition

The need to accommodate rotation-invariance in a recognition system can be shown by the decline in performance of a "classical" recognition framework, which is not designed with rotation-invariance in mind. Fig. 5.1 shows a mis-classification curve for the texture recognition system of Chapter 3. The classification results are averaged over the entire 30 texture database (see Fig. 3.8). The degradation in performance with increasing the rotation angle of the input test images, is evident.

Although in general classification clearly declines with the deviation of the test inputs from the training inputs, it is of interest to note that the behavior is related to the texture's orientational characteristics. Nonoriented or unstructured textures are less affected by rotation. An example of a texture in this category is the pigskin texture, which is shown

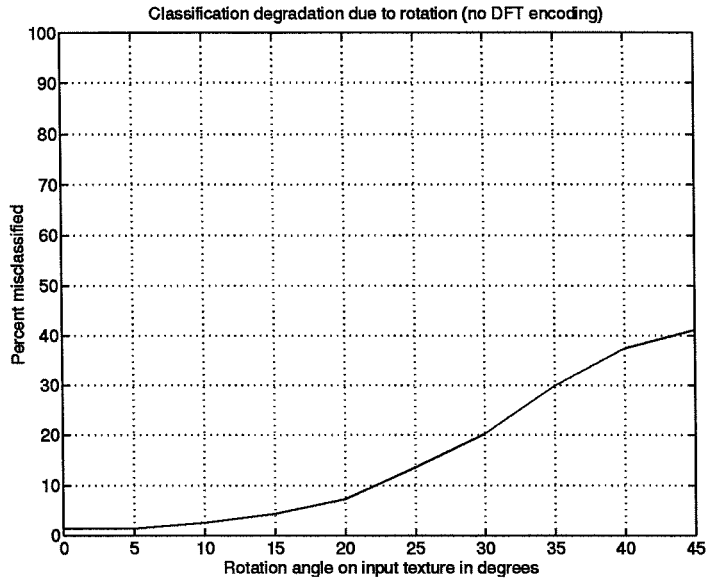


Figure 5.1: Average degradation in the classification performance of a non rotation-invariant texture classifier with respect to rotation angle of the sample texture patch, 30 texture database.

rotated at 5 degree increments in the bottom of Fig. 5.2. The perceptual similarity of the texture as it is rotated allows for high-accuracy in the classification process. The mis-classification curve of the pigskin texture is therefore of uniform slope, as can be seen at the bottom of Fig. 5.3.

Textures that exhibit strong orientation (such as the wood, canvas and herringbone textures) are the most affected by the rotation. Rotated versions of the wood texture are displayed at the top of Fig. 5.2. Here the rotation has a major effect on the perceived texture. A characteristic mis-classification curve for the wood texture is presented at the top of Fig. 5.3. The sharp decline in performance with rotation of the input is evident.

It is clear that rotation invariance is critical for a general texture recognition system. Rotation invariance can be achieved in one of two ways, either by extracting rotation-invariant features or by appropriate training of the classifier to make it 'learn' invariant properties. Achieving rotation invariance in the classifier is extremely difficult (as indicated by the limitations of the learning literature quoted above). In this work we use the first approach. We use the rotationally-invariant representation via the steerable pyramid, as introduced in the previous chapter, as the key element in the system in coping with the invariance challenge.

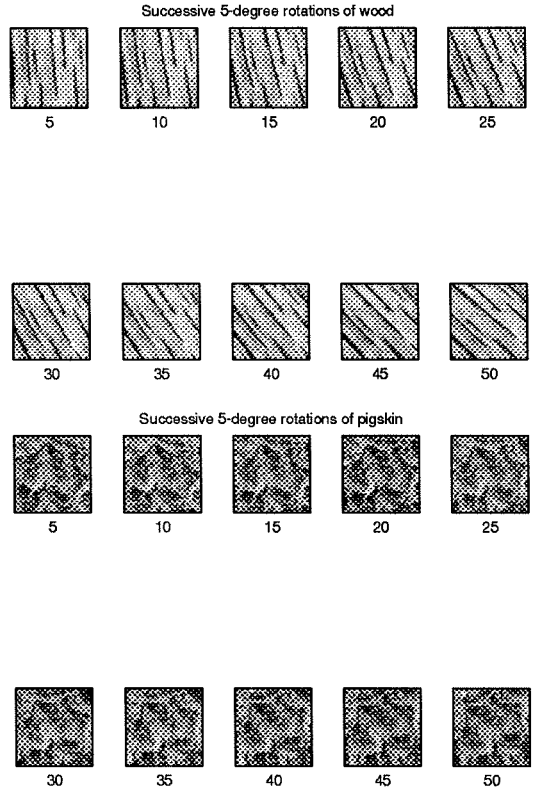


Figure 5.2: Example of rotating textures by 5 degree increments. The wood texture (top) is an example of an oriented texture. The pig texture (bottom) is an example of a nonoriented texture.

5.4 The Rotation-Invariant Texture Recognition System and Results

The recognition system we now hold is very much the same as the system we had presented and analyzed in Chapter 3. Two major stages comprise the system - the feature extraction stage and the classification stage. Feature vectors are extracted via the Oriented Pyramid. Following the analysis of the previous chapter we know that this pyramid is steerable and that a rotation-invariant representation for the feature vectors can be extracted via DFT encoding. We add the DFT encoding step as part of the feature-extraction phase of the system. In the classification part we continue to use the Rule-based learning algorithm together with the Backprop neural-network algorithm and the K-nearest neighbor algorithm. The updated system block diagram is depicted in Figure 5.4.

In order to visualize the effect of the DFT-encoding scheme we next revisit the example textures of section 5.3. In Fig. 5.5 the classification curve for a rotated input pattern is displayed for both textures. Using the DFT representation, the slope of the curve is close to being uniform, for both the nonoriented (pig) as well as the *oriented* (wood) texture, as the textures are rotated. This is in sharp contrast to the original classification curve in Fig. 5.3 and strongly demonstrates the rotation-invariance of the DFT-encoding representation.

The shift to the DFT encoded vectors is not without a price. It is interesting to note that the actual classification results decrease for the pig texture, as the representation is shifted to the DFT encoding (compare to Fig. 5.3 bottom). This representation is very useful for handling rotated databases, yet, for non-rotated databases performance can decrease relative to the nonDFT-encoding case. This issue will be addressed shortly (see also Chapter 4).

We next demonstrate the performance of the rotation-invariant system on the difficult 30 texture database.

Experimental Setup

For each texture a 256×256 patch is used from which a set of 16 non-overlapping 64×64 size windows are extracted. 12 of the windows are used for training and 4 *different* ones are used for testing. The test set consists of rotating each of the test windows by 5°

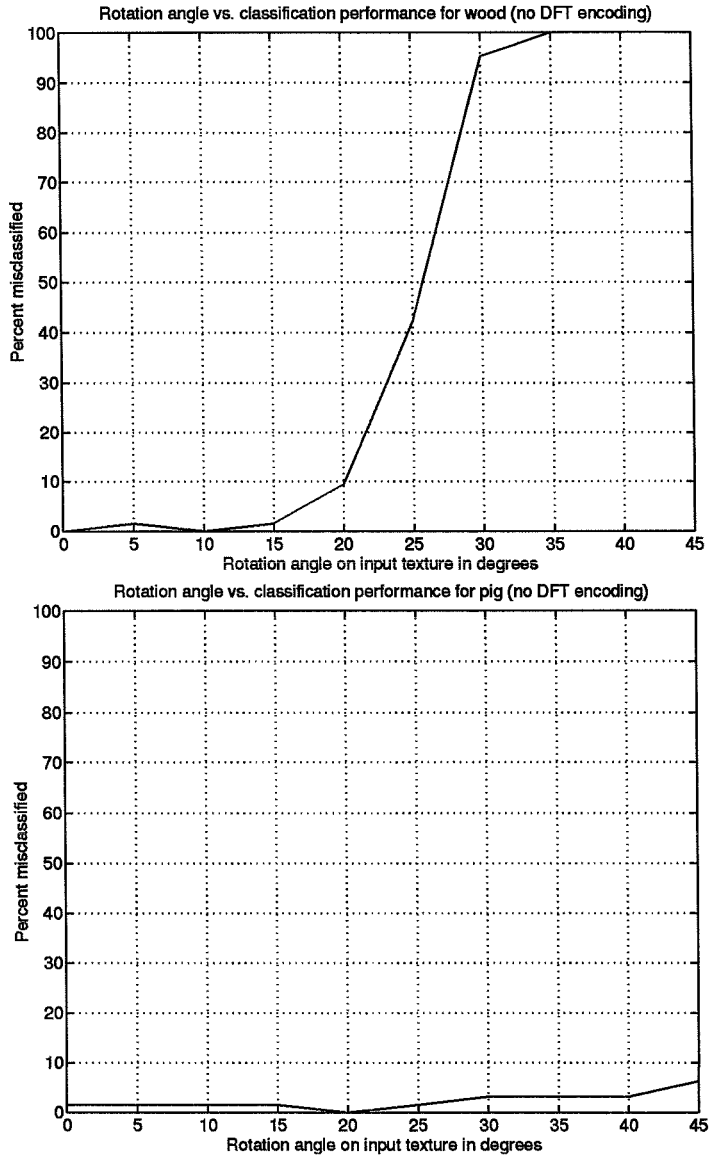


Figure 5.3: Degradation with rotation. The wood texture (top) is an example of an oriented texture. The pig texture (bottom) is an example of a nonoriented texture.

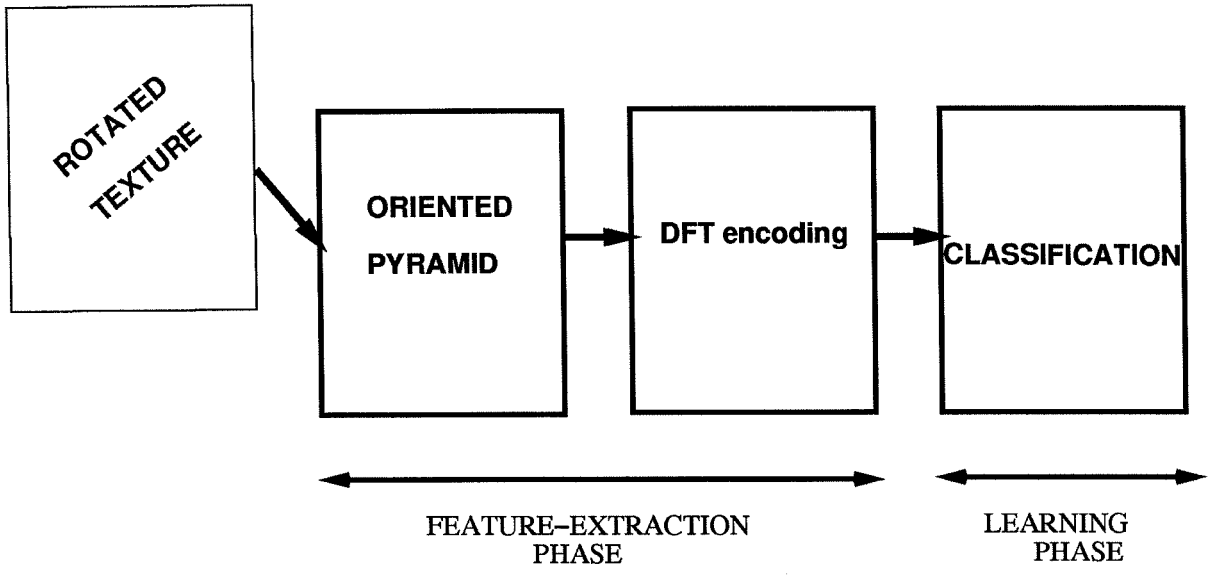


Figure 5.4: System Block Diagram

increments, between 5° and 50°; thus we have 40 test vectors per texture (For an example of two rotated textures refer back to Fig. 5.2. Note that the training inputs are *not* rotated.

The recognition process entails the following steps:

1. The 64×64 texture patch is passed through the steerable pyramid to result in a set of 15 8×8 filter maps.
2. The 8×8 filtermaps are averaged, to produce one representative feature vector per 64×64 input window.
3. The extracted feature vector, f, is DFT encoded to generate the companion feature vector, $\hat{f}$, of the previous chapter.
4. The magnitudes of the set of DFT-encoded feature-vectors are next presented to the classification system for recognition. Three classifiers are used: the K-nearest neighbor classifier (K-nn), the Back-Propagation (BackProp) classifier, and the Rule-based learning system.

To augment the testing results 4 different runs are made, each with a different set of 4 testing windows, and the classification results are averaged over these runs.

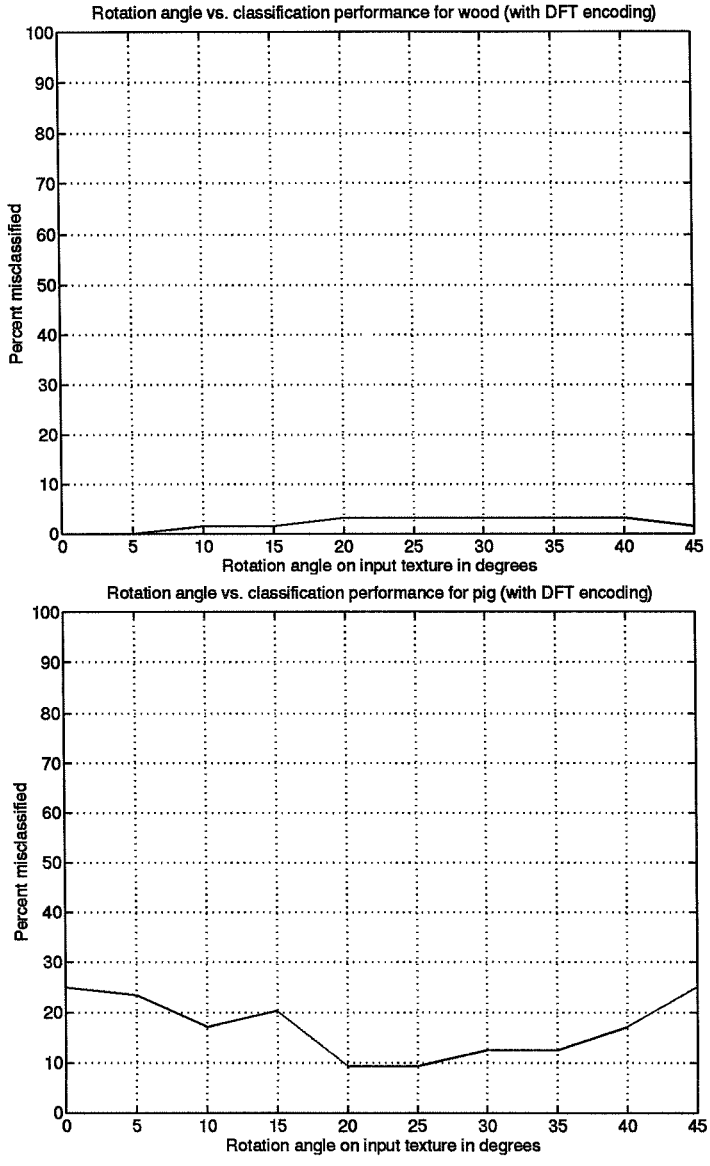


Figure 5.5: Rotation with DFT encoding. The wood texture (top) is an example of an oriented texture. The pig texture (bottom) is an example of a nonoriented texture. Uniform classification curves are evident in both cases.

Results

The results of classifying the 30 textures of Fig. 3.8 via the process outlined above, using the rule-based network classifier as an example (following the process outlined above), are depicted in the following table:

30 TEXTURE CLASSIFICATION - rule based network		
class	correct classification	confused with
1.bark	97.50%	hmpaper
2.calf	100.00%	
3.cloth	27.50%	pig
4.crdbrd	100.00%	
5.jeans	100.00%	
6.grass	62.50%	hmpaper,reptile
7.pig	97.50%	cloth
8.raffia	77.50%	sand
9.water	100.00%	
10.wood	100.00%	
11.backpack	100.00%	
12.bookbox	100.00%	
13.brownbag	100.00%	
14.chbkcover	67.50%	brownbag
15.cork	47.50%	calf,grass,hmpaper,reptile
16.cotcanv	100.00%	
17.frcanv	100.00%	
18.fur	97.50%	strawmat
19.hmpaper	67.50%	bark,reptile,text
20.napkin	90.00%	strawmat
21.prtboard	100.00%	
22.reptile	97.50%	grass
23.straw	35.00%	calf,wood
24.text	100.00%	
25.towel	60.00%	grass
26.vinyl	100.00%	
27.herring	100.00%	
28.sand	77.50%	bark,raffia,wire
29.wire	95.00%	hmpaper
30.strawmat	100.00%	

An overall classification rate of 86.58% is achieved. The use of the steerable representation for the input space has enabled us to enhance the recognition system to be rotation-invariant.

Several interesting points can be seen in the above table. First, the more structured

textures, such as jeans(#5), wood(#10) and canvas(#17), are mostly classified at 100% accuracy. Confusion occurs with the more unstructured textures, such as the cloth(#3) and grass(#6) textures. Second, it can be seen that the classification errors occur with what are visually similar (i.e., difficult to distinguish) textures. The cloth and pig textures constitute an example of such a texture pair. In these results it seems that we have successfully handled the rotation-invariance issues for the structured textures and we are now left with the original difficulty of the unstructured data (see analysis in Chapter 3).

To illustrate the performance of the recognition system further, we next compare results for 4 different classification scenarios, as given in the following table:

30 TEXTURE CLASSIFICATION - 4 cases				
rotation	DFT	Knn	Backprop network	Rule-based network
N	N	95%	97.25%	97.5%
Y	N	80%	67.5%	77.42%
N	Y	90%	83%	85.83%
Y	Y	91.5%	84.67%	86.58%

The 4 cases reflect the performance of the system as related to the state of the input test patterns (rotated vs. non-rotated), and the input representation space (rotation-invariant (via DFT encoding) vs. non rotation-invariant).

- In case 1 the data is nonrotated and no DFT conversion is performed. High classification rates are achieved. These rates indicate the strength of the recognition system on non-rotated inputs.

- In case 2 the test data is rotated and no DFT conversion is performed. Here the strong decline in performance is evident, as is expected for a non rotation-invariant system (see section 5.3). The result presented is averaged over both the oriented and non-oriented textures in the database, with the non-oriented cases (which are less affected by the rotation) actually augmenting the performance. In addition, rotations of $5° - 50°$ were included. A more severe drop in performance would be evident if only the large rotations were included.

- In case 3 the test data is nonrotated but a DFT representation is used. Here we get high classification results, though somewhat reduced from case 1. This is the price paid for shifting to a rotation-invariant representation which makes fewer assumptions about what is known. In using the *magnitude* of the DFT-encoded vectors for

the classification task, we ignore the phase information. In the 4 dimensional case we are actually left with only 2 invariant components (we zero out the single phase component and we cannot rely on the Nyquist component - see analysis in Chapter 4). This loss of information results in the reduced performance.

- Finally, in case 4 we have a rotated test set analyzed by the rotation invariant system. The increase in the results from case 2 are evident. As expected the results are similar to case 3. We are able to classify the varied database of 30 textures rotated at 5° resolution at an accuracy of close to 90%.

The above results represent state-of-the-art recognition results in the domain of large-database rotation-invariant natural texture recognition.

5.5 Orientation Estimation

In this section we study a method for computing the *rotation angle* of a given test patch relative to a reference texture in the database.

In the context of the recognition stage, only the magnitude of the DFT-encoded feature vector is used. Upon identification, the *phase* of the DFT can be inspected to determine the amount of rotation of the input texture relative to a prestored sample of that texture class. We find that general information about a texture's behavior in orientation space allows us to associate a certainty measure (or variance) to the extracted orientation. In one extreme, the input texture is non-oriented in which case no meaningful rotation angle can be detected.

The typical sequence of events in our system for the rotation-invariant texture recognition and rotation *detection* is as follows:

Training stage

For each texture in the database (prelearned set) we find general characteristics in orientation space. These include: defining a peak orientation (hereon called the "relative" orientation) at the three scales of the system, choosing a "dominant" scale based on the distribution of angles around the peak, looking at the behavior of the orientations across scale thus defining the texture as either single-oriented, double-oriented or non-oriented. Based on the above, a reliability measure for the texture's orientation is extracted.

Production stage

1. An input test image is classified as "wood", for example, based on the magnitude of a set of DFT encoded feature vectors.
2. The phase of the DFT-encoded feature vectors is inspected at the "dominant" scale (predetermined for the "wood" texture), from which the relative rotation of the input texture is determined.
3. A variance or reliability measure is assigned to the orientation calculation based on the general characteristics extracted in the training stage.

We start with a description of our method for extracting general orientation characteristics for the prelearned database of textures. We then proceed to use this information as we estimate the rotation angles of new test patterns which are presented to the system.

5.5.1 Orientation Characteristics of Textures

We start with a general orientation analysis of a prestored library of textures. In this analysis we wish to gain information about the following:

- General category of the texture: non-oriented, one dominant orientation per scale, two or more dominant orientations per scale.

- Defining one scale (out of the 3 scales in the system) to be a "dominant" scale – the scale at which the rotation will be extracted.

- Extracting information about the dominant orientation(s) per scale and the reliability of the orientation estimation.

The above goals are achieved using an histogram analysis scheme, as is described in the following experiment.

Experimental setup

The 30 texture database is comprised of 256×256 size images. Each textured image is run through the pyramid to produce a set of $32 \times 32 = 1024$ feature-vectors (i.e., each

feature-vector represents an 8×8 block in the input image) which are subsequently DFT-encoded. Only the core $30 \times 30 = 900$ feature-vectors are kept in the interest of avoiding edge effects.

The phase of the first harmonic of the DFT components of each of the 900 feature-vectors (namely, elements 2, 7, and 12) for each texture is next extracted, to produce a total of $3 \times 900 = 2700$ phase estimates for each of the 30 textures. (We will refer to each element of this collection of phase estimates as having the form $\rho_k e^{i\theta_k}$, where ρ_k is the response strength along a particular orientation θ_k.)

We examine next the histograms of the phase estimates of each texture, at each of the 3 scales of the system. Several histograms and plots were found informative for the orientation analysis. Figs. 5.6, 5.7 and 5.8 show the plots for the wood, pig and herring textures, respectively. A histogram of the phase estimates is presented top left, a second "polar"-histogram is presented top right. In each of these representations only the θ_k's are taken into account while the ρ_k's or "weightings" associated with these estimates are ignored. For this reason, two alternate representations of the measured orientation information which take both ρ_k and θ_k into account are of interest. The first of the two alternate representations, shown bottom left, is a scatter plot of all of the $\rho_k e^{i\theta_k}$'s. The second representation, shown bottom right, is a plot of $\log \rho_k e^{i\theta_k}$. Since

$$\log \rho_k e^{i\theta_k} = \log \rho_k + i\theta_k \tag{5.1}$$

the log plot is essentially a rectangularized version of the polar plot. In this representation the high-energy orientations are represented by the rightmost points in the plot. By inspecting the plot it is possible to determine the peak angle in the strongest-power region.

Results

We investigate the above-described histograms for estimating general orientation characteristics of the texture patches. We are looking for the following information:

- The number of peak orientations per scale
- A definition of the texture as a single-oriented, double-oriented or non-oriented texture.
- Extracting a reliability measure for the dominant orientation.

Three examples are presented. An example of a strongly (single) oriented texture is the wood texture (see Fig. 5.6). We note the strong peak at 90° at all three scales. An

example of a non-oriented texture is presented in the following figure (Fig. 5.7), for the pigskin texture. Here we note the large variance in angles in each of the three scales. Looking at the log plot we note the large variance in the power axis as well, with votes for the angles spread out in the entire angle axis. Most of the textures in the 30 texture database fall into one of the above categories. One extreme case of interest is the herring texture which consists of two dominant orientations per scale. This case is presented in Fig. 5.8. Scale 2 has in this case the most power in the oriented components, and the least variance. It is also this scale in which we see two strong peaks in orientation, at 90° and 135°. The strongest of the two peaks is used as the "reference" peak. Still, the information about the second peak is very important in the reliability of the estimated rotation angles. With the 4-dimensional DFT representation we extract a single phase component (as described in Chapter 4). In a single oriented texture, we are thus able to retrieve rotations in the 180 degree range. In the case of two dominant orientations per scale, our estimate can be accurate to ±90 degrees. This reliability measure is valuable information for the results as presented below.

Table 5.5.1 summarizes the orientation analysis for each of the 30 textures in our database. For each texture is indicated:

- The dominant scale from which to determine orientation. This scale was chosen as the one with smallest standard deviation around the peak.
- The mean angle (or peak) in that scale.
- The standard deviation of the angle distribution with a ±40 degree span around the peak.
- The number of points captured in the ±40 degree window around the peak.
- In cases where the number of points is less than 80% of the total (900), two possibilities exist: the texture is nonoriented or the texture has two dominant orientations per scale. A second mean is indicated in the table for those textures with multiple orientations in the dominant scale.

In Table 5.1 we have incorporated the highest resolution information available on the texture database. Table 5.2 presents a similar table for the case of averaging over the filtermaps of $64 * 64$ texture patches. Histogrammed are inputs from 64 such windows. Due to the averaging step we reduce the noise in the phase estimation. This is evident in the lower standard deviation results for each texture.

5.5.2 Rotation Angle Estimation

The goal of the orientation analysis is to calculate the rotation angle of each rotated test input texture. To each estimated rotation-angle we associate a reliability measure derived from the tables of the previous section.

Texture patches of size 64×64 are presented to the system with each patch rotated to ten different angles (5° to 50° with steps of 5°), as described in section 5.4. Here we are averaging over the filtermaps to produce one representative feature vector per $64 * 64$ window. The averaging step increases the reliability of our phase estimates, which we compare to the results in Table 5.2. We investigate the accuracy of rotation estimation for all 30 texture classes.

Three different methods are used for producing rotation error estimates. The three methods differ only in the choice of reference angle from which all successive rotation estimations are judged. In **method 1** the reference phase for all test patches of a particular texture is taken from one randomly chosen 5° rotated test patch in the set. In **method 2** the reference phase for all test patches of a particular texture is taken to be the *average* of the phases for all the 5° test patches of that texture. **Method 3** uses the phase of the 5° version of each texture patch as its reference phase for all rotations. In this case the training and test patch are the same and thus the error measured is not affected by changes in orientation across different patches of the same texture. Method 3 gives best accuracy measurements, with methods 1 and 2 presenting more real-world like scenarios to the system.

Experimental results are summarized in Table 5.3. In general we note that the rotation angle estimations are highly accurate for oriented textures (e.g., wood, jeans), in all 3 methods and across all scales, with much lower accuracy evident for the non-oriented textures (such as particle-board of brown bag). An interesting case is the herringbone texture. In this case poor error measures are found for both methods 1 and 2 but not for method 3. This is due to the fact that the texture has 2 dominant orientations 90° apart. In this case methods 1 and 2 accumulate error (an average between approximately 45° and 135°). In method 3 the rotation of each patch is found relative to its own 5° phase, thus no averaging takes place and the rotation accuracy found is high.

The above rotation-angle estimation results are summarized in Figure 5.9. Data from

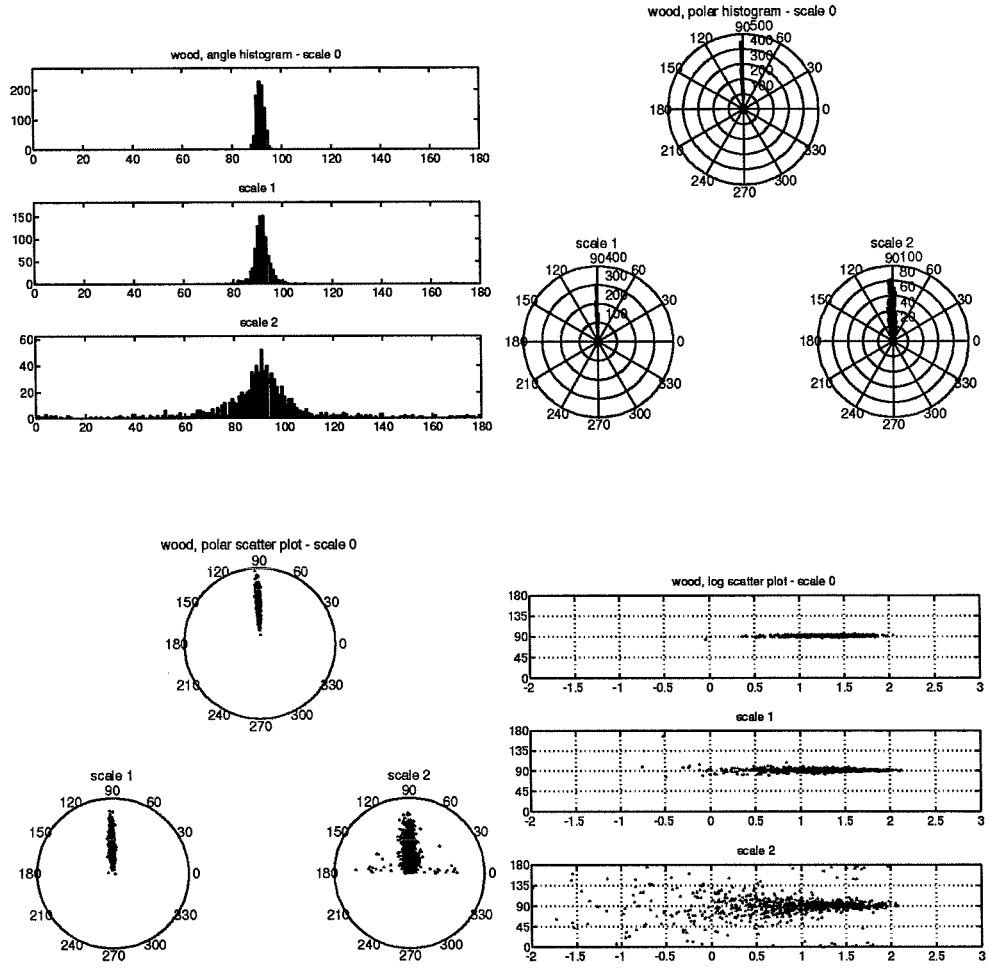


Figure 5.6: Wood texture characteristics. At top left and top right we have the regular and polar histograms, respectively. These consist of of 900 θ_k's representing the local orientations of 8×8 blocks in a 256×256 patch of the wood texture. In both histograms, a strong peak is visible at 90° at all three scales. The smallest standard deviation is found at scale 0 (as is typically the case). The bottom two plots incorporate both the angle certainties (the ρ_k's) and the orientation angles. The left polar scatter plot is simply a polar plot of $\rho_k e^{i\theta_k}$ for all k. In this representation, the strongest angles are furthest from the origin. In the second, log scatter plot, we have plotted $\log \rho_k e^{i\theta_k} = \log \rho_k + i\theta_k$ for all k, so that the strongest angles are the rightmost points on the plot.

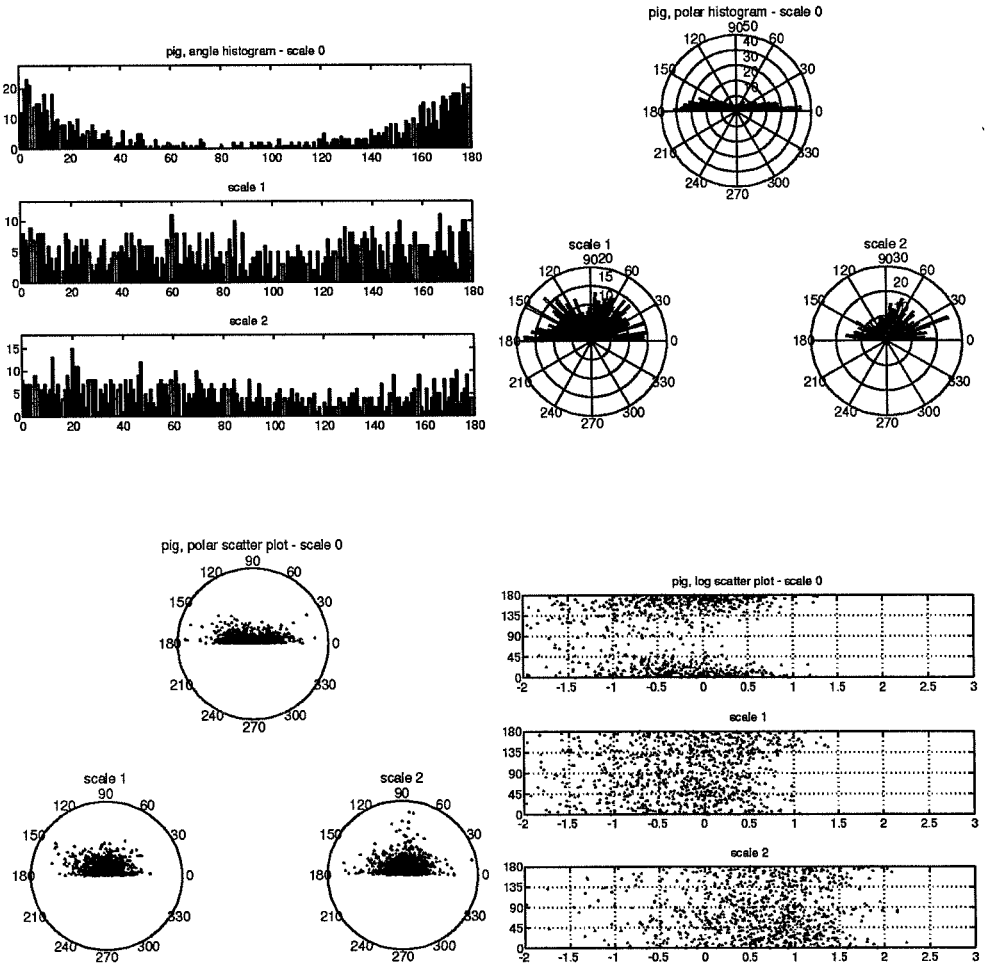


Figure 5.7: Pigskin texture characteristics. See description of plots intext and Fig. 5.6. Here we see an example of a non-oriented texture's behavior in orientation space.

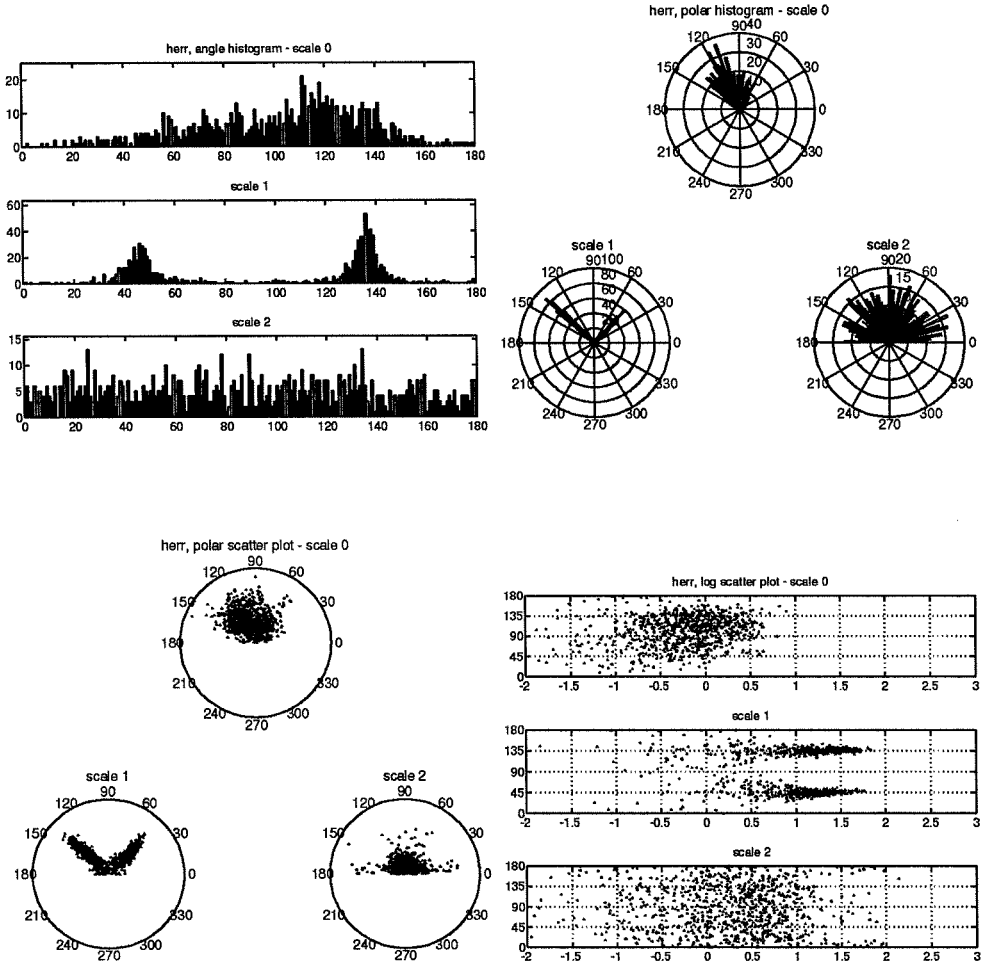


Figure 5.8: Herringbone texture characteristics. Similar plots are shown as in Fig. 5.6. Note the strong bi-modal angle distribution at scale 1, indicating the presence of two dominant orientations.

	dom. scale	mean	std.	num. points (out of 900)	mean2
1.bark	1	100.01	17.70	663	
2.calf	0	81.70	10.71	900	
3.cloth	0	70.15	17.71	794	
4.crdbrd	1	91.37	5.26	896	
5.jeans	0	86.54	1.86	900	
6.grass	0	106.45	17.15	823	
7.pig	0	178.23	17.62	756	
8.raffia	0	176.34	11.34	832	
9.water	0	92.36	3.40	898	
10.wood	0	91.46	1.44	900	
11.backpack	0	91.20	0.93	900	
12.bookbox	0	179.34	5.65	900	
13.brownbag	0	86.20	17.39	726	
14.chbkcover	0	182.27	12.96	738	
15.cork	0	62.95	9.43	900	
16.cotcanv	0	132.81	5.71	900	
17.frcanv	0	88.96	1.18	900	
18.fur	0	110.76	7.78	900	
19.hmpaper	0	60.92	18.38	734	
20.napkin	2	135.73	11.80	867	
21.prtboard	0	90.10	17.53	726	
22.reptile	0	75.44	14.05	845	
23.straw	0	143.66	11.66	892	
24.text	0	84.14	8.74	873	
25.towel	0	145.05	21.64	765	
26.vinyl	0	48.58	19.15	802	
27.herring	1	135.01	8.80	507	46.94
28.sand	0	106.72	16.68	790	
29.wire	0	94.60	2.58	899	
30.strawmat	1	165.46	6.92	892	

Table 5.1: Orientation characteristics of textures via a histogram analysis. Shown for each texture are the dominant scale (0, 1 or 2), the mean (peak) orientation angle, the standard deviation of the distribution around the peak angle within a $\pm 40°$ window, and the number of estimates that fall inside that window. The dominant scale (which is 0 for 83% of the textures) is defined as the scale with the smallest standard deviation about the measured mean. The standard deviations for highly structured textures such as wood and jeans tend to be very small while those of unstructured textures such as particle-board and handmade-paper are large. Note that in the chosen window span of $\pm 40°$, the max standard deviation is around 20. The herringbone texture (number 27) displays orientation preference to two angles $90°$ apart; the second mean is indicated for this special case.

	dom. scale	mean	std.	num. points (out of 64)	mean2
1.bark	0	116.68	12.23	64	
2.calf	0	80.07	4.29	64	
3.cloth	0	70.23	7.99	64	
4.crdbrd	1	91.31	0.90	64	
5.jeans	0	86.57	0.61	64	
6.grass	0	96.07	11.87	64	
7.pig	0	178.80	4.93	64	
8.raffia	0	175.82	2.13	64	
9.water	0	91.85	0.58	64	
10.wood	0	90.68	0.68	64	
11.backpack	0	91.27	0.65	64	
12.bookbox	0	179.82	1.48	64	
13.brownbag	0	85.08	8.56	64	
14.chbkcover	0	182.52	2.77	64	
15.cork	0	63.11	2.93	64	
16.cotcanv	0	132.31	2.08	64	
17.frcanv	0	88.94	0.38	64	
18.fur	0	110.74	6.64	64	
19.hmpaper	0	60.14	8.09	64	
20.napkin	2	134.83	3.58	64	
21.prtboard	0	90.85	8.27	64	
22.reptile	0	78.48	5.89	64	
23.straw	1	122.42	7.83	64	
24.text	0	85.27	1.56	64	
25.towel	1	139.22	19.87	53	
26.vinyl	0	49.54	13.52	64	
27.herring	2	135.90	7.63	40	
28.sand	0	105.88	6.16	64	
29.wire	1	91.16	0.54	64	
30.strawmat	1	166.01	2.27	64	

Table 5.2: Orientation characteristics of textures via a histogram analysis - 64*64 window case.

	average error (in degrees)								
	method 1			method 2			method 3		
	s0	s1	s2	s0	s1	s2	s0	s1	s2
1.bark	4.81	6.98	18.74	4.03	5.95	20.24	2.66	4.66	12.31
2.calf	2.56	3.19	3.75	2.32	3.00	3.88	0.58	1.07	2.47
3.cloth	15.74	16.04	16.98	8.47	7.04	13.75	2.46	4.81	18.30
4.crdbrd	0.66	0.40	1.36	0.66	0.52	0.63	0.30	0.44	1.06
5.jeans	0.24	1.26	27.88	0.23	1.37	60.04	0.10	1.29	23.66
6.grass	2.30	30.20	31.33	1.92	34.28	62.91	0.85	5.66	21.01
7.pig	7.15	37.31	32.69	6.95	52.75	38.67	2.74	9.26	26.06
8.raffia	1.83	5.66	11.29	2.17	5.77	9.17	1.59	1.41	7.49
9.water	1.99	3.70	11.25	1.47	3.41	12.06	0.25	0.57	6.37
10.wood	0.55	1.15	6.83	0.54	1.13	3.12	0.27	0.24	2.60
11.backpack	0.46	9.54	20.34	0.44	9.12	14.83	0.41	6.43	19.49
12.bookbox	2.86	4.97	37.17	1.62	4.55	53.38	1.17	2.95	12.71
13.brownbag	15.57	11.93	35.78	7.44	8.70	37.32	1.84	6.67	20.66
14.chbkcover	4.26	8.95	8.89	45.45	42.81	14.75	1.81	3.62	13.70
15.cork	5.57	6.53	6.39	4.49	5.68	6.10	0.98	1.52	3.43
16.cotcanvas	1.01	1.86	18.10	1.08	1.99	12.13	0.81	1.73	9.35
17.frcanv	0.97	1.40	13.66	0.92	1.35	12.17	0.78	1.34	9.88
18.fur	2.44	1.55	3.82	1.13	1.61	3.63	0.34	0.46	2.07
19.hmpaper	8.89	6.26	15.46	10.61	44.34	12.27	4.89	1.74	3.30
20.napkin	9.50	3.29	1.58	43.21	2.31	0.90	2.71	1.80	1.03
21.prtboard	5.50	12.29	32.81	5.46	13.66	34.51	2.38	9.57	12.60
22.reptile	1.21	12.68	20.76	1.32	12.67	16.59	1.15	8.70	13.34
23.straw	7.79	9.52	13.11	6.02	6.87	8.47	0.44	0.45	3.36
24.text	1.18	1.25	2.51	1.15	0.99	1.44	1.18	0.42	1.20
25.towel	23.62	26.21	32.07	13.86	13.46	22.96	3.90	3.24	14.49
26.vinyl	5.76	26.35	13.92	5.76	41.89	11.35	1.82	3.53	7.50
27.herring	32.91	45.13	40.90	28.08	44.94	39.26	1.66	1.94	8.63
28.sand	3.00	8.03	23.87	2.30	7.73	39.78	1.12	3.02	9.02
29.wire	0.58	0.59	8.52	0.42	0.62	9.84	0.37	0.53	8.66
30.strawmat	6.83	2.29	5.72	6.76	1.19	5.57	4.30	0.93	5.20

Table 5.3: Rotation angle estimation analysis. Shown are the calculated errors in orientation angle prediction for each of the 30 textures using the 3 methods outlined in section 5.5.2. The phase information from the DFT-encoded feature vectors for 10 different rotations of each texture were compared against the ideal phase values and then averaged to produce the above error measurements.

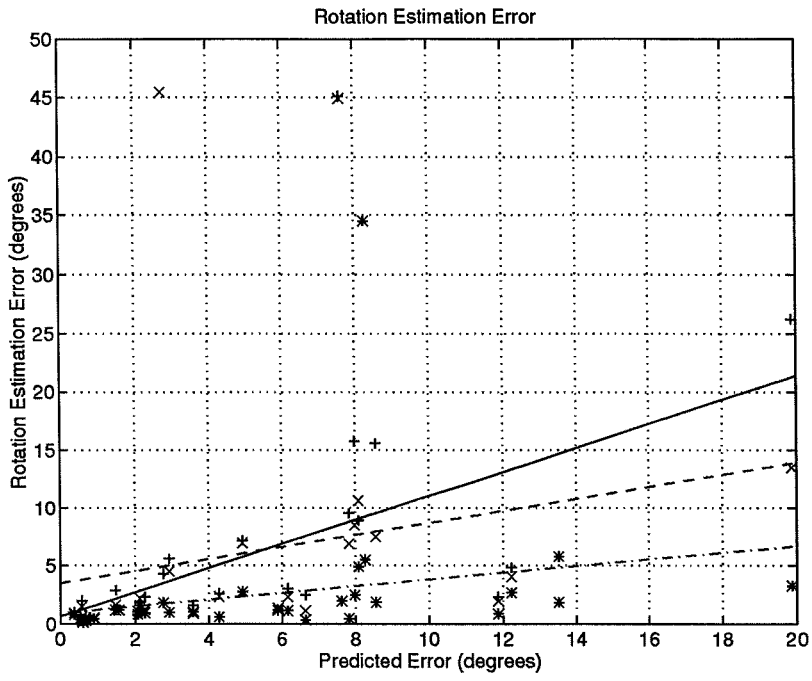


Figure 5.9: Rotation Error Estimation - Predicted vs. Calculated error for three methods. (solid:method1, dash:method2, dashdot:method3)

the three orientation estimation methods is plotted against the data of Table 5.2 which corresponds to the predicted error per texture. The slopes of all three curves indicate that the predicted error bound, which is based on the variability of the orientation within a texture patch, gives an upper bound to the actual estimated rotation angle.

Our final result is the average rotation-estimation error across all 30 textures for each of the three methods above. For each texture the dominant scale is read out from Table 5.2 and the corresponding rotation-estimation error is extracted from Table 5.3. The average error (in degrees) is given in the following table:

method 1	method 2	method 3
5.7	6.0	1.5

The results of the above table reflect the average error over both oriented and *non*oriented textures.

We next look at the subset of textures for which the rotation angle estimate has the most meaning. In the following table we present the same error measures for the subset of the 30 texture database, which we have determined to have a single dominant orientation.

The characterization of the textures, based on Table 5.1, was made as follows: Textures with a standard deviation greater than 12 were considered to be "non-oriented." The Herring texture was also omitted due to its bimodal distribution. The remaining textures are labeled as "oriented" (with one dominant orientation).

method 1	method 2	method 3
1.90	1.44	0.67

We note the high-accuracy in our rotation estimation across all three methods. The results summarized in the above two tables demonstrate high accuracy orientation estimation on the large database of all 30 textures, with even higher accuracy for the subset of oriented textures.

5.6 Conclusions

We have presented a rotation-invariant texture recognition system together with a method for estimating rotation of textures. The methods are novel in that features are obtained from oriented pyramid filters which present particularly good properties of "discriminability" for texture classification and are computationally efficient. The orientation estimation method is particularly reliable in that confidence measures are estimated along with the orientation. We have demonstrated state-of-the-art results both in classification and orientation estimation on a set of 30 (natural and real-world) textures.

Chapter 6
Future Extensions of the Texture Recognition System

6.1 Introduction

In this chapter we suggest possible future extensions of the texture recognition system in a variety of domains. First, we are interested to know if we can use the texture classification capability for natural scene analysis and segmentation. Fusing together several visual modalities (such as intensity-based segmentation, texture, stereo and color), will be required in real-world tasks such as automated scene analysis for autonomous navigation, remote-sensing and more. In this chapter we consider the use of texture alone and its possible contribution in these domains. We conclude that texture can be a major contributor and that the proposed system is robust in real-world noisy environments [GG93].

We also look upon some of the advantages of incorporating learning paradigms in the classification process. These include learning from examples, rather than from the human-experts, automated rule generation and more.

Finally, we suggest an extension to the texture recognition system and generalize to the analysis of *shape*. We propose that the recognition framework is a general one and suggest minor modifications for its use in the 2D, more object oriented domain. Initial encouraging examples in the face recognition domain are given.

6.2 Natural Scene Analysis

An application of the texture discrimination system to natural scene analysis is given next. In the remaining experiments, the training images are extracted as small subregions of the test images. The classification results shown are therefore partly on the training data but mostly exhibit generalization to new input. Larger data-bases will enable totally separate training and testing images. Initial simulation results are presented in Fig. 6.1 which presents a sand-rock scenario. The input images are photographs of the moon landscape, taken from the Jet Propulsion Laboratory (JPL) image database. The training examples are presented, followed by two input images and their corresponding output label maps,

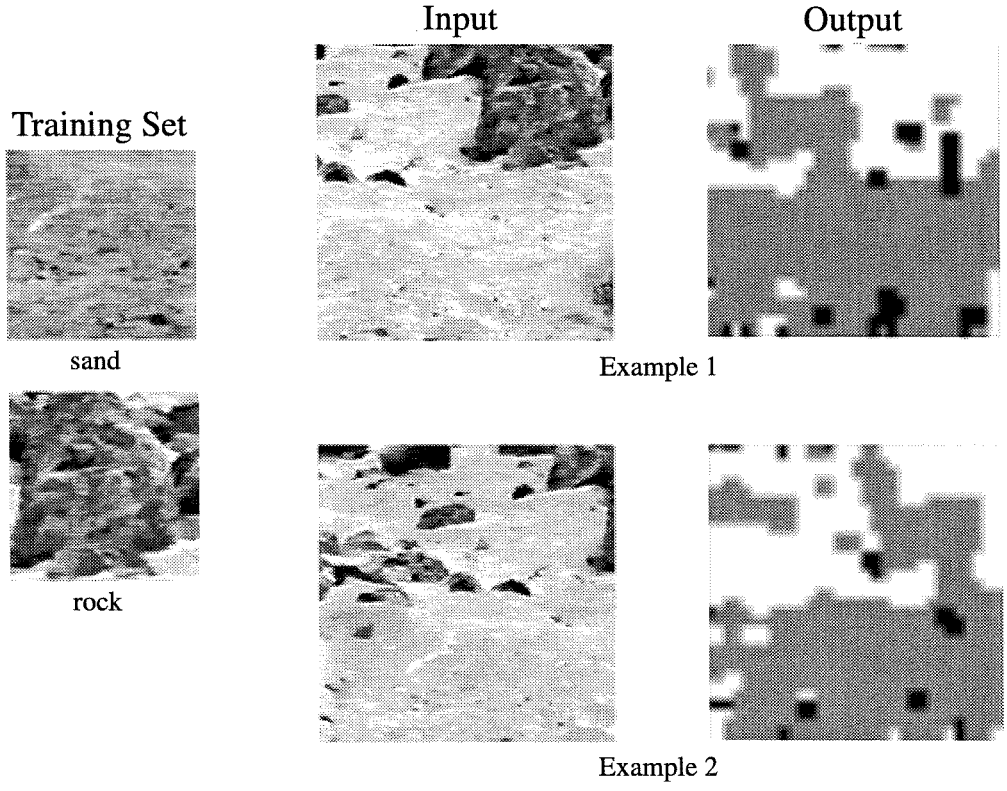
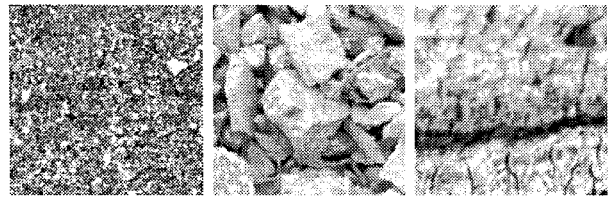


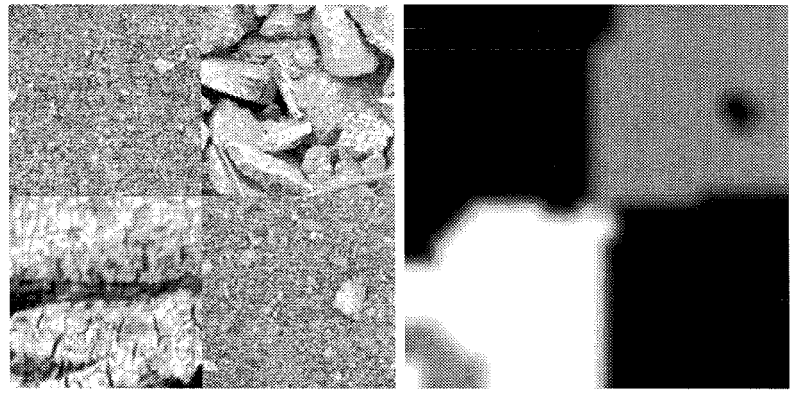
Figure 6.1: Natural scene analysis - rock sand scenario

left to right, respectively. Here, white represents rock, gray represents sand and black regions are classified as unknown. The probability maps are not displayed in this and the following examples. The system copes successfully with this challenge. We can see that a distinction between the regions has been made and for a possible mission such as rock avoidance (landing, navigation etc.) reliable results were achieved.

Fig. 6.2 presents a more difficult task. Gravel, rock and wood were learned (top) and a new mosaic test image was presented for recognition and labeling (bottom left). Note that the test image differs (except in the top left corner) from the training image set. Here, the images are very noisy, taken using a 35mm film camera at JPL. The input image is successfully segmented and labeled as can be seen in the resulting label map (bottom right). Note that black represents a class label in this figure. Generalization in a noisy environment as well as an advantage over intensity-based schemes are demonstrated in this example. Recall that the segmentation is based on classifying each $8*8$ local window



Training Set



Input Output

Figure 6.2: Natural scene analysis - 3 texture case

of the input image. At this resolution, the differences between the rock and gravel are subtle. Overall, these initial results are very encouraging and indicate the robustness of the system to cope with difficult real-world cases.

6.3 Autonomous Navigation Scenario

Autonomous vehicles require an automated scene analysis system to avoid obstacles and navigate through rough terrain. Fusion of several visual modalities, such as intensity-based segmentation, texture, stereo, and color, together with other domain inputs, such as soil spectral decomposition analysis, will likely be required for this challenging task. In Fig. 6.3 we present preliminary results on outdoor photographed scenes taken by an autonomous vehicle at JPL. The presented scenes (left) are segmented into bush and gravel regions (right). The training set consists of 4 $64 * 64$ image samples from each category. In the top example (a 256*256 pixel image) light gray indicates gravel while black represents bushy regions. We can see that intensity alone can not suffice in this task (for example, top right corner). The system has learned some textural characteristics which guided the segmentation in otherwise similar-intensity regions. Note that this is also probably the cause for identifying the track-like region (e.g., center bottom) as bush regions. We could learn track-like regions as a third category, or specifically include such examples as gravel in our training set.

In the second example (a 400*400 input image, bottom) light gray indicates gravel, dark gray represents a bush-like region, and black represents the unknown category. Here, the top right region of the sky, is labeled correctly as an unknown, or new category. Note that intensity alone would have confused that region as being gravel. Overall, the texture classification system succeeds in achieving a correct, yet rough, segmentation of the scene based on textural characteristics alone. These are encouraging results indicating that the learning system has acquired informative characteristics of the domain.

6.4 Remote-Sensing Image Analysis

6.4.1 Introduction

Our most recent results pertain to the application of the system to the noisy environment of satellite and airborne imagery. The goal is to segment the input image into homogeneous

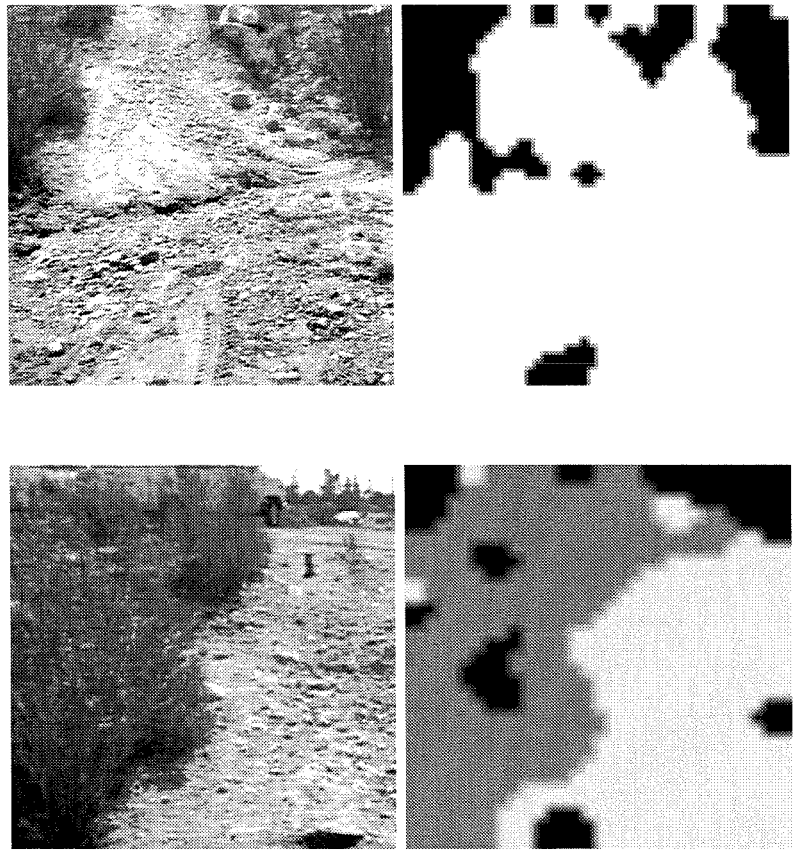


Figure 6.3: Image Analysis for Autonomous Navigation

textured regions and identify each region as one of a prelearned library of textures, e.g., tree area and urban area distinction. Classification of remote sensing imagery is of importance in many applications, such as navigation, surveillance and exploration. It has become a very complex task spanning a growing number of sensors and application domains. The applications include: land-cover identification (with systems such as the AVIRIS and SPOT), atmospheric analysis via cloud-coverage mapping (using the AVHRR sensor), oceanographic exploration for sea/ice type classification (SAR input) and more.

Much attention has been given to the use of the spectral signature for the identification of region types [Wha87, LP91]. Only recently has the idea of adding spatial information been presented [TSJ91]. In this work we investigate the possibility of gaining information from textural analysis. Texture can play a major role in segmenting the images into homogeneous areas and enhancing other sensors capabilities, such as multi-spectra analysis, by indicating areas of interest in which further analysis can be pursued. Fusion of the spatial information with the spectral signature will enhance the classification and the overall automated analysis capabilities.

Most of the work in the literature focuses on human expert-based rules with specific sensor data calibration. Some of the existing problems with this classic approach are the following [TSJ91]:

- Experienced photo-interpreters are required to spend a considerable amount of time generating rules.
- The rules need to be updated for different geographical regions.
- No spatial rules exist for the complex Landsat imagery.

An interesting question is if one can automate the rule generation. In this section we demonstrate that the learning framework can automatically learn spatial rules from a given database of examples.

6.4.2 Results

Initial results of applying the texture recognition system to remote-sensing images are given next. Fig. 6.4 presents two such examples. The first example (top) is an image of Pasadena, California, taken via the AVIRIS system (Airborne Visible/Infrared Imaging Spectrometer). The AVIRIS system covers 224 contiguous spectral bands simultaneously, at 20 meters per pixel resolution. The presented example is taken as an average of several

bands in the visual range. In this input image we can see that a major distinguishing characteristic is urban area vs. hilly surround. These are the two categories we set forth to learn. The training consists of a 128*128 image sample for each category. The test input is a noisy 512*512 image. In the presented output (top right), the urban area is labeled in white, the hillside in gray and unknown, undetermined areas are in darker gray. We see that a rough segmentation into the desired regions has been achieved. The probabilistic network's output allows for the identification of unknown or unspecified regions, in which more elaborate analysis can be pursued (see Section 3.4). The dark gray areas correspond to such regions; one example is the hill and urban contact (bottom right) in which some urban suburbs on the hill slopes form a mixture of the classes. Note that in the initial results presented, the blockiness perceived is the result of the analysis resolution chosen. Fusing into the system additional spectral bands as our input, would enable pixel resolution as well as enable detecting additional classes (not visually detectable), such as concrete material, a variety of vegetation etc.

A higher resolution Airborne image is presented at the bottom of Fig. 6.4. The input image (left) is of much higher resolution and it is evident that segmentation based on texture is of importance. The classes learned are bush (output label dark gray), ground (output label gray) and a structured area, such as a field or the man-made structures (white). Here, the training was done on 128*128 image examples (one example per class). The input image is 800*800. In the result shown (right) we see that the three classes have been found and a rough segmentation into the three regions is achieved. Note in particular the detection of the three main structured areas in the image, including the man-made field, indicated in white. These results demonstrate the network's capability of generalization and robustness to noise in two complex real-world images.

6.5 Extension to Shape Recognition

6.5.1 Introduction

Classical computer vision techniques, although researched for many years, have not yet proven successful in the shape recognition domain. Such techniques model the desired shapes via predefined models (e.g. geometrical). These are very frequently computationally intensive and are very sensitive to noise and changes in brightness or rotation of the

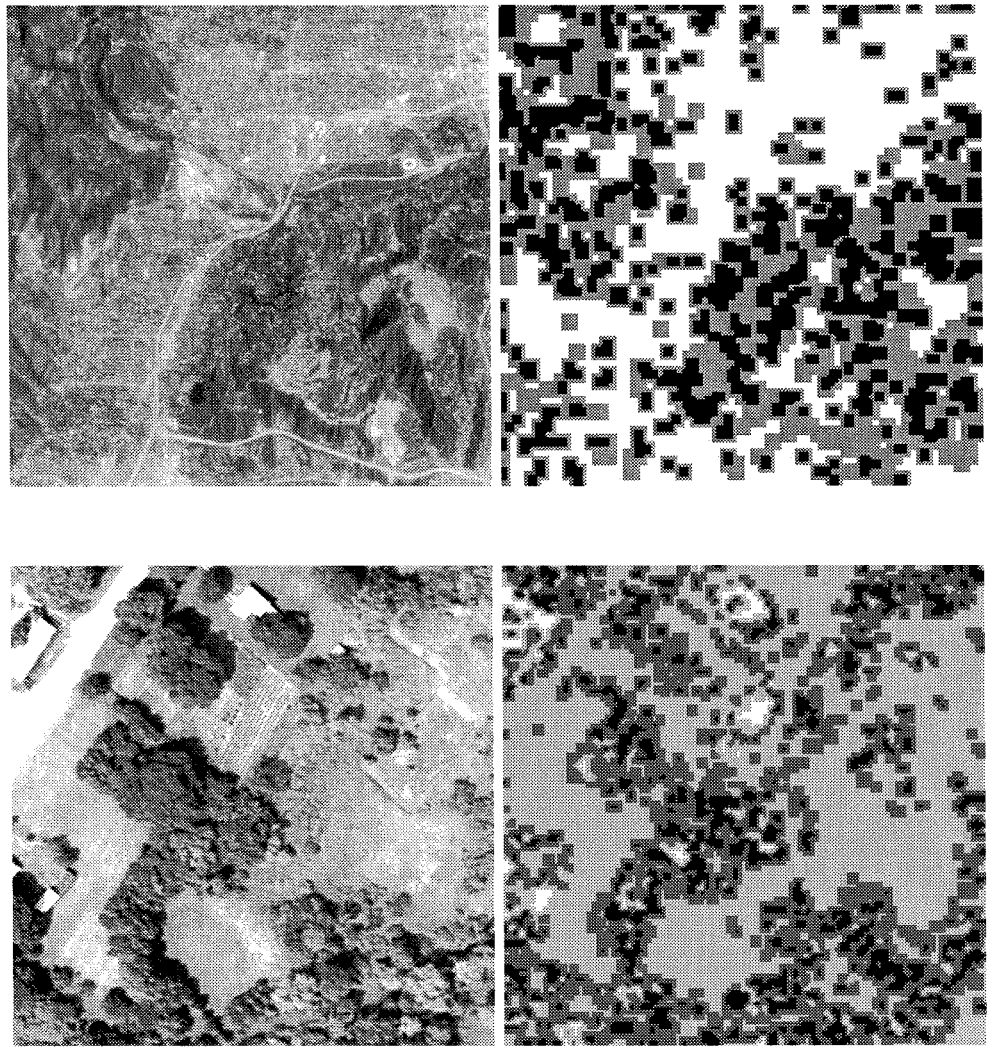


Figure 6.4: Remote sensing image analysis results. The input test image is shown (left) followed by the system output classification map (right). In the AVIRIS (top) input, white indicates urban regions, gray is a hilly area and dark gray reflects undetermined or different region types. In the Airborne output (bottom), dark gray indicates a bush area, light gray is a ground cover region and white indicates man-made structures. Both robustness to noise and generalization are demonstrated in these two challenging real-world problems.

input. Moreover, the model chosen works on a predetermined family of 2D shapes and usually cannot be generalized to any other input domain.

We are interested in *learning* the characteristics of the input domain from examples, rather than using parametric modeling of the input. In this section we extend and generalize the texture recognition system, to incorporate learning in more structured domains. This requires the preservation of the 2D spatial information in the recognition process. The proposed scheme is a general one which can adapt to different input domains. Initial results on face recognition are presented.

6.5.2 The Generalized Recognition System

Fig. 6.5 displays the generalized recognition system. The three layers of processing correspond to the original system processing stages (see Chapter 3). Parallel layers of processing are incorporated, each one extracting a higher level symbolic representation of the input domain. In the shift to a more symbolic representation we compress the amount of information that is encoded, while preserving the essential information for the classification task.

The main difference from the texture recognition system is in the unsupervised clustering and coding stage (center layer). Coding the input domain is the more data-driven stage. In the texture recognition system we transformed the image space into an array of 15-dimensional feature *vectors*, each vector corresponding to a local window in the original image. In the shape recognition task, the 2D spatial information within the filter maps is a strong characteristic which we need to preserve and identify in the recognition process.

We preserve the output of the filtering stage as a set of 15 continuous valued feature *maps*; each representing a filtered version of the original input. In this higher-level processing stage, the filter responses are quantized into three categories: low, medium and strong response. The K-means statistical clustering algorithm is utilized for this task, on the entire data set and across all filter maps. The categories learned are therefore general ones and will be appropriate for the testing data as well. Areas of strong response are valuable in distinguishing among different classes. To efficiently compress the representation further, we next code 5*5 windows in each quantized filter map according to their relative response strength. The number of strong response pixels (out of 25) are counted and coded at 5 equally spaced levels. The original input image is now represented as a set

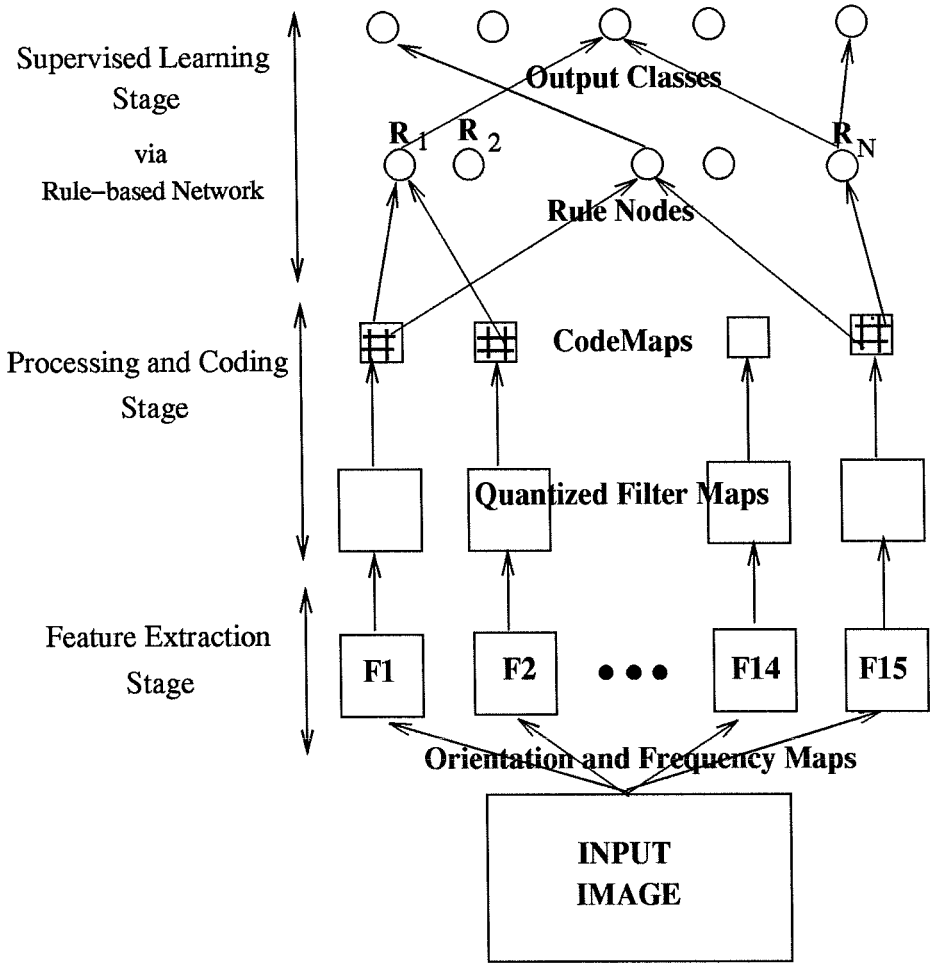


Figure 6.5: System Block Diagram

of 15 6*6 codemaps.

6.5.3 Face Recognition Results

We present initial results of applying the recognition system to the task of face recognition. An example of the input domain is presented in Fig. 6.6. Three faces are shown (left) followed by a subset of their corresponding filter response maps. Presented are the non-oriented component, followed by a horizontal oriented filter which detects *vertical* information, a filter sensitive to 45 degree orientation and one sensitive to 135 degrees, left to right, respectively. Stronger response is represented in black. The original image size is 240*240. The subsampled filter maps are 30*30. We see that even in this reduced resolution filter domain, distinguishing spatial response characteristics are present.

The processing and learning stages follow next. First, a quantization of the filter responses into the three categories of weak, moderate and strong response is enacted. This is followed by coding of the residual response maps into corresponding 6*6 codemaps, as depicted in Fig. 6.7 (left to right, respectively). In the above, unsupervised processing of the feature maps is performed on the entire data-base of examples. As shown in Fig. 6.7, the coding of the original image into a more symbolic representation via the codemaps (right) entails sufficient information for the following stage of learning. Note that the resultant code-maps are displayed in the original image size.

In the supervised learning stage, the rule-based network (see Chapter 3) is to learn the spatial characteristics of the codemaps [1]. Examples of learned rules are shown in Fig. 6.7. Here, the spatial differentiating characteristics across a specific filter map (the non-oriented component) are learned. Second order rules are presented.

An initial simulation was run as follows. The training set consisted of 11 people with 2 images per face. The testing set consisted of 2 new images of each person. The data-set of faces was taken in a lab setting with the people instructed to change their expressions and appearances. Variations included hair styles, smile vs. nonsmile, orientation of head etc. Note that no attempt was made to normalize or align the images via features, such as nose, eyes etc. This reduces the computational complexity and produces a more noisy data base. We have correctly labeled all faces of the training data and have 100% classification results on the testing set. Some extreme cases which were successfully labeled are presented in Fig. 6.8.

6.6 Summary and Discussion

In this chapter we have utilized the texture recognition system to an advantage in several application domains. An application to natural scenery analysis, with initial attempts at remote-sensing image analysis, are shown. These initial results are very encouraging and indicate the robustness of the system in coping with noisy real-world applications.

Future work in this direction should include a larger database of examples with more variability between the training and testing images; such as variability in the location and time at which the images are taken. Also, ideally we would like to have several apriori-

[1] In the current implementation rules are learned *within* each codemap. A generalization would be to learn rules *across* the codemaps as well.

labeled images to which we could compare our results, thus getting a more quantitative measure of success.

We have discussed a generalized recognition system and have demonstrated the learning system's capability to handle the varied input domain (of texture and shape) within a single framework. This is very different from the standard thought of specializing algorithms to the specific tasks at hand, which leads to ad-hoc schemes that are tuned to the specific task requirements. The learning scheme learns the most informative spatial correlations within the filter maps, while indicating the most informative filters for the task. The system is robust to the environment in several aspects. The use of orientation and frequency tuned maps (vs. original pixel values) allows for brightness invariance. The multi-resolution approach and additional coding schemes allow for translation and rotation invariance.

The areas of automated navigation, remote sensing and of-course, face recognition, are each a whole separate field of research in its own right. In all these domains, much larger databases are needed in order to achieve more conclusive results and identify the strengths and weaknesses of the learning system. We have only touched upon the different topics briefly and the presented results should be viewed mainly as encouraging and initiating further research in each of the domains.

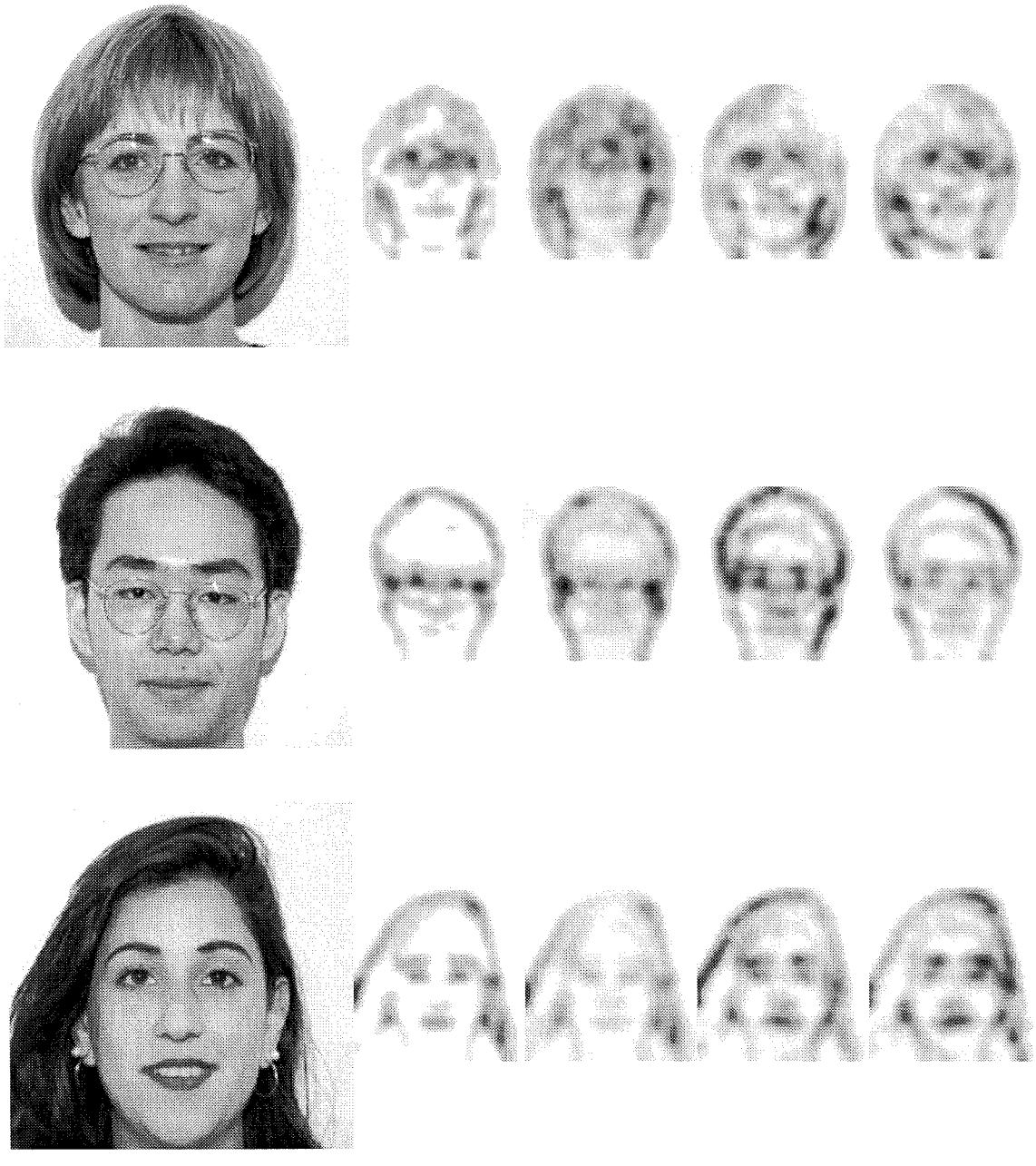
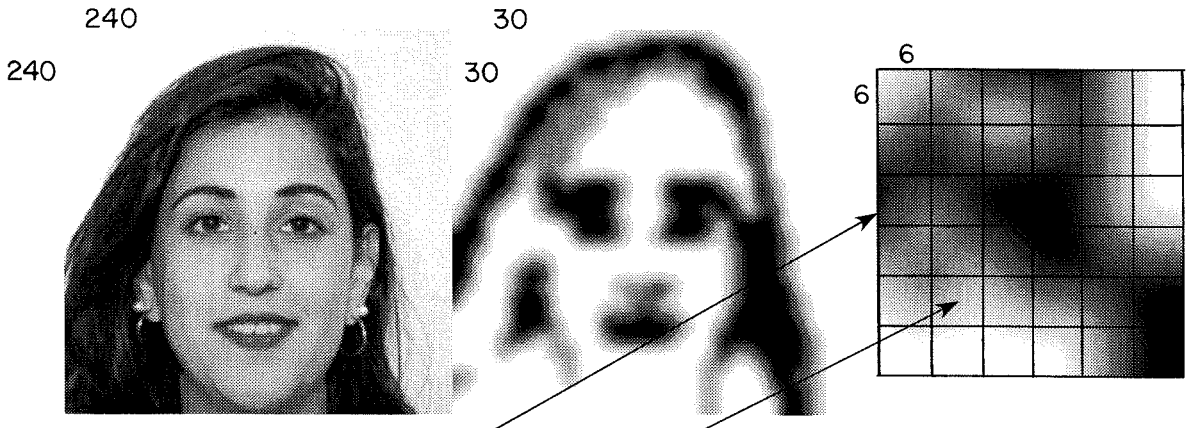


Figure 6.6: Input images and corresponding filter maps. Shown are the input images followed by the non-oriented filter response map, and response maps which are sensitive to the vertical, 45 degrees and 135 degrees, left to right, respectively.



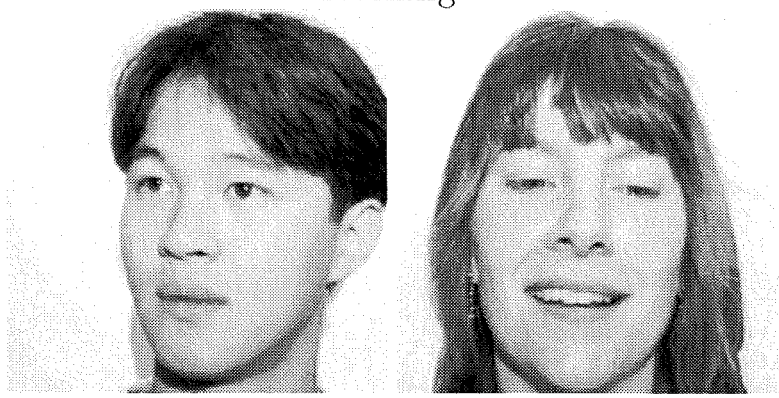
if AT13 1.0 AND AT26 2 THEN ILLIANA.

if AT22 2.0 AND ATT28 1.0 THEN THOMAS.

if AT15 3.0 AND ATT13 2.0 THEN ART.

Figure 6.7: Processing and learning stages. Presented are three input images and their corresponding quantized filter response maps and codemaps, left to right, respectively. The filter response maps and the codemaps are enlarged to match the original image size. An example of a second order rule is presented following each example image. The rules correlate spatially between pixels of the codemaps and vote to the corresponding output classes.

Training



Testing

Figure 6.8: Extreme cases in the training and testing data sets (top and bottom, respectively).

Chapter 7
The Multiresolution Representation for Image Enhancement & Coding

In this chapter we investigate additional domains of image analysis, including image enhancement and image coding schemes. The underlying common theme is the use of the multiresolution pyramid representation.

Image *enhancement* attempts to improve the quality of an image for human or machine interpretability, where quality is measured subjectively. Most enhancement filters are heuristic and problem oriented, and models of the degradation are generally not used in deriving them. This differs from image *reconstruction* where the goal is to restore an image based on some knowledge of the degradation it has undergone. The interest in image enhancement schemes has resurged lately in the digital communication world of HDTV (high-definition television) and Multimedia. In this market the challenging task is to transfer imagery data in a constrained bandwidth (BW) environment, and the outcome is judged subjectively by the human observer. Enhancing the images following the various compression schemes is a natural (though not always simple) road to take.

Enhancement algorithms have been around for many years and classically entail a linear operation of adding power to the existing high-frequencies in the image. A brief description of several such algorithms will be given in Section 7.1. In this chapter we present a new enhancement algorithm (sections 7.1 and 7.2) [GA94, GA]. This enhancement algorithm *augments* the frequency content of the image using shape-invariant properties of edges across scale, by using a non-linearity that generates phase-coherent higher harmonics. The procedure utilizes the Laplacian pyramid image representation. Results are presented depicting the power-spectra augmentation and the visual enhancement of several images.

In the second part of this chapter (Section 7.3) we present an initial attempt to combine the image enhancement scheme with existing image compression schemes [GL94]. This idea is part of a new trend developing in the image processing and image compression fields which has to do with the convergence of the two fields. The pyramid representation as a means for image coding will be briefly described, followed by the combination with

image enhancement for additional compression. The combination of image enhancement with progressive transmission, for savings in analysis time, will be discussed as well.

7.1 Image Enhancement by Non-Linear Extrapolation in Frequency Space

7.1.1 Introduction

We present a procedure for creating images with higher resolution than the sampling rate would allow. The procedure outlined here is applicable in several domains. In one case we assume that a given input image is blurred and no degradation model is known. If the degradation model exists, restoration techniques can be applied, together with other frequency enhancement techniques present in the literature (see below). The enhancement scheme described here can then be applied as an additional enhancement utility. A second application domain relates to expanding an image up by a factor of two in size (so called "zoom in"). This is desirable in many applications (e.g., HDTV, video-phone), but generally results in an image which appears blurred because there is no power in the highest spatial frequency band.

Classical enhancement algorithms usually add power to existing high frequency components, i.e., the edges. We start by giving a brief review of some well known enhancement filters. The presented scheme concentrates on creating *new* high-spatial frequencies and thus can augment existing (linear) high-frequency enhancement techniques available in the literature. In the proposed scheme the given frequency content is augmented using shape-invariant properties of edges across scale. The augmentation procedure is based on the pyramid image representation and can be described using the scale-space formalism [Wit83, YP83b]. The edge properties across scale are described following which we formalize the enhancement procedure. This procedure includes a simple extrapolation across scale representations using a non-linearity that generates phase coherent higher harmonics. The enhancement algorithm is schematically summarized in Fig. 7.1. It shares the basic structure of other high-frequency enhancement methods, except that the linear filter is replaced by a nonlinear filter operation. Experimental results depicting the power-spectra augmentation and the visual enhancement of several images will be presented.

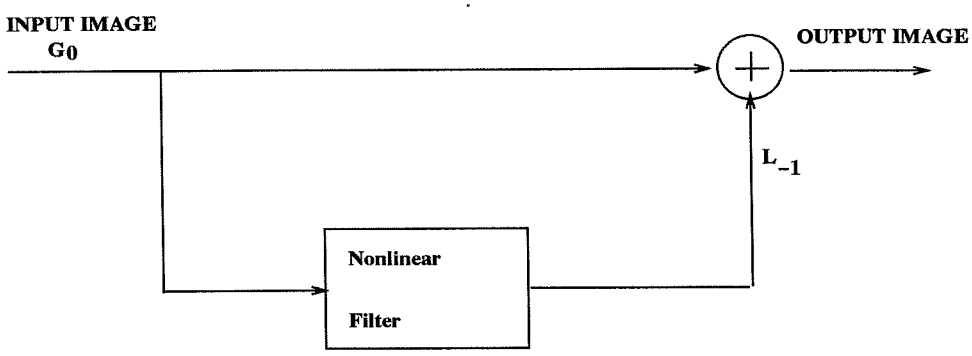


Figure 7.1: Basic diagram of the image enhancement algorithm

7.1.2 The Image Enhancement Field - Background

Image enhancement (or "edge enhancement") techniques focus on sharpening image edges. Edge enhancement filters are high-pass filters and their effect is to boost edges. Examples of edge enhancement filters include the following:

1. Gradient operators. A simple way to use them is to keep only the magnitude. Other methods keep both the magnitude and the angle. Moving across an edge, the gradient will start at zero, increase to a maximum, and then decrease back to zero. This produces a broad edge. Thinning methods are often used to thin down the edges.

2. Laplacian operators [MH80].

3. Adding a high-pass filtered image (such as the Laplacian) back to the original image to boost edges yet maintain the underlying grey level information. This procedure is analogous to the photographic process of "unsharp masking". In this process, a film is exposed through a negative superimposed on a slightly defocused positive transparency, thus subtracting the local mean, and the result is an image with improved edges.

4. $\nabla^2 G$ operators ("Mexican hat" operators [Mar82]). ∇^2 is the Laplacian and G is the two dimensional Gaussian distribution. The idea behind these filters is to first smooth the image with a Gaussian shaped filter, and then find the edges (using the Laplacian) in the smoothed image.

5. Enhancement in the direction of the gradient. An example of this type of filter is to compute the gradient at (i, j) and apply a one-dimensional Laplacian operator in the direction of the gradient.

The previous filters are fairly simple. More complex methods are available which try to handle the issue of noise (e.g., apply high-pass filters selectively to suppress noise). Overall, we have a set of high-pass filters which boost up the existing high-frequencies in the given image by concentrating on the image edges. In the proposed algorithm we also concentrate on the image edges, but extend our interest to investigate the behavior of edges across scale. We use the scale-space characteristics in augmenting the frequency content of the given image. We next give a short background on the image representation across scale.

Image Representation Across Scale

Edges are an important characteristic of images, since they correspond to object boundaries or to changes in surface orientation or material properties. An edge can be characterized by a local peak in the first derivative of the image brightness function, or by a zero in the second derivative, the so called zero crossings (ZC) [MH80]. An ideal edge (a step function) is scale invariant in that no matter how much one increases the resolution, the edge appears the same (i.e., remains a step function). This property provides a means for identifying edges and a method for enhancing real edges.

We concentrate on the edge representation of an image across different image resolutions. For this we view the image in a multi-resolution framework via the Gaussian and Laplacian pyramids (see Chapter 2). The Laplacian pyramid preserves the shape and phase of the edge maps across scale. An example is presented in Fig. 7.2.

The application of the Laplacian transform to an ideal edge transition results in a series of self-similar transient structures as illustrated in Fig. 7.3 left. An edge of finite resolution would produce a decrease in amplitude of these transients with increasing spatial frequency, with the magnitude of the edge going to 0 at frequencies above the Nyquist limit (see Fig. 7.3 right). An edge of finite resolution can be created by starting with a low resolution Gaussian image and then adding on all the bandpass transient structures. To create an edge with twice the resolution requires the creation of a self-similar transient at the next level, hereby referred to as L_{-1}. The most essential features of these transient

Figure 7.2: Multi-scale sequence of edge maps. Presented from left to right are the Laplacian pyramid components: L_0, L_1 and L_2 respectively.

structures is that they are of the same sign at the same position in space, hence their ZCs line up, and they all have roughly the same amplitude. The precise shape of the structures need not necessarily be maintained so long as their scaled spatial frequency response is similar. The simple procedure described next creates localized transients for L_{-1} that satisfy all these constraints except for the maintenance of constant amplitude. While more complicated procedures could handle the amplitude constraint, it was found that sharpening the stronger value edges produces in itself visually pleasing results.

The pyramid representation can be viewed as a discrete version of the scale-space description of ZC which has been introduced in the literature [Wit83, YP83b, YP83a]. The scale-space formalism gives the position of the ZC across a continuum of scales. One of the main theorems [Wit83] states that ZC of an image filtered through a Gaussian filter have nice scaling properties, one of which is that ZC are not created as the scale increases. If an edge appears at lower resolutions of the image it will consistently appear as we shift to higher resolutions (see Fig. 7.3). Although theoretically defined, not much work has yet taken advantage of the image representation across scale. In the algorithm presented next we utilize the shape invariant properties of edges across scale based on the pyramid representation and in agreement with the consistency characteristic of the scale-space formalism.

7.1.3 The Enhancement Scheme

Our objective is to form the next higher harmonic of the given signal while maintaining phase. Fig. 7.4. illustrates a 1-dimensional high-contrast edge scenario. The given input,

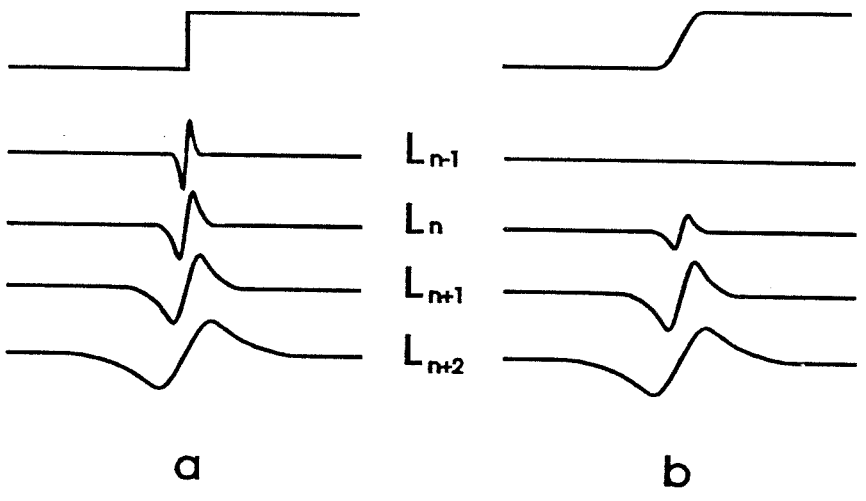
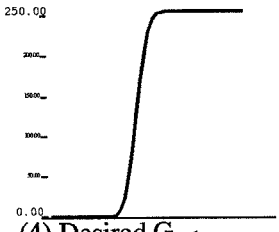


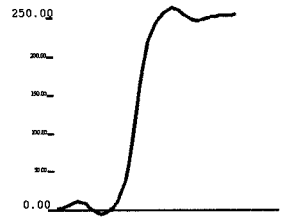
Figure 7.3: Laplacian transform on an edge transition

G_0, is shown in (0), together with its pyramid components, L_0 and G_1, shown in (1) and (2) respectively. From the pyramid reconstruction process we know that adding the high-frequency component L_0 to the G_1 component can sharpen G_1 to produce the input G_0. Ideally, we would like to take this a step further. We would like to predict a higher-frequency component, L_{-1}, preserving the shape and phase of L_0, as shown in (3), so that we can use the reconstruction process to produce an even sharper edge, which is closer to the ideal-edge objective, as shown in (4). The L_{-1} component can not be created by a linear operation on the given L_0 component; i.e., it is not possible to create a *higher*-frequency output by a linear enhancement technique.

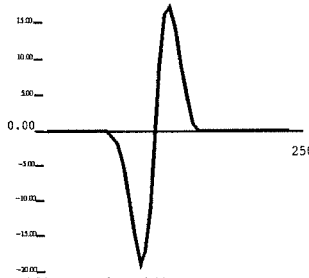
It remains to be shown how the L_{-1} component of the pyramid can be predicted. We extrapolate to the new resolution (L_{-1}) by preserving the Laplacian-filtering waveform shape, together with sharpening via a non-linear operator. The waveform as in (5) is the result of bounding the L_0 response, multiplying the resultant waveform by a constant and then removing the low-frequencies present in order to extract a high-frequency response. It was found experimentally that clipping L_0 with a threshold of 0.04 times the maximum signal's amplitude (i.e., 10 out of 256), and then multiplying by a factor of 6 gives the closest resemblance to the ideal-edge output without much ringing side-effect. The enhanced edge output is presented in (6).



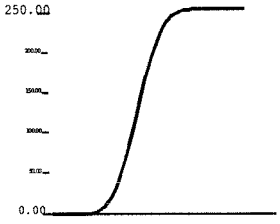
(4) Desired G_{-1}.



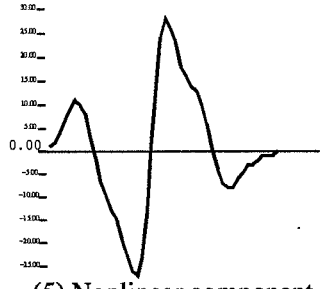
(6) Nonlinear edge enhancement.



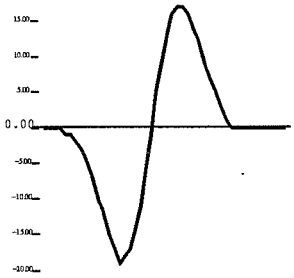
(3) Desired L_{-1}.



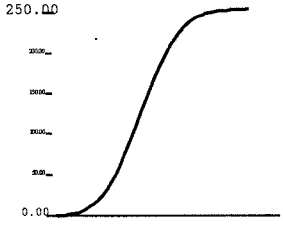
(0) Input image, G_0.



(5) Nonlinear component, L_{-1}.



(1) Pyramidal component, L_0.



(2) Pyramidal component, G_1.

Figure 7.4: One-dimensional ideal-edge scenario

Equation 7.1 formalizes the generation of L_{-1}.

$$L_{-1} = const(BOUND(L_0)) \tag{7.1}$$

where $BOUND(S)$ is the following function:

$$BOUND(S) = \begin{cases} T & \text{if } S > T \\ S & \text{if } -T \leq S \leq T \\ -T & \text{if } S < -T. \end{cases}$$

Here, T = $0.04(G_0)_{max}$.

Generating the new output image entails taking the L_{-1} image as the high-frequency component of the pyramid representation. Based on the reconstruction capability of the pyramid representation (see Chapter 2), the new output is generated next as the sum of the given input, G_0, and L_{-1}, as in equation 7.2.

$$OutputImage = L_{-1} + G_0. \tag{7.2}$$

Using the above algorithm, a "zoom in" application includes the following 5 steps:

1. Extract the high frequency components of an image $L_0 = F_{bp} * G_0$. We have used $F_{bp} = 1 - W$, as utilized in the formation of the Laplacian pyramid (Chapter 2).

2. Create a double sampled (along both dimensions) version, L_0^e, of L_0 through an interpolation procedure. The standard pyramid expand technique is to insert 0's at alternate pixels and lines, smooth the result with the lowpass filter, and then multiply the result by a factor of 4, which can be combined in an efficient subroutine.

3. Clip L_0^e, which amounts to setting the magnitude of the signal to a predetermined level if it exceeds that value (equation 7.1).

4. Create L_{-1} by bandpass filtering the clipped output to reshape the new transients so they have the desired spatial frequency components. This was done using the same bandpass filter as in step 1, but other variants are possible. It has been found that this step can be eliminated and still produce pleasing results.

5. Add a scaled version of L_{-1} to an expanded version of G_0 created in the same fashion as L_0^e in step 2 (equation 7.2).

The approach as presented above, together with the non-linear operator characteristics which were found for the high-contrast ideal edge scenario, are used in the examples which follow.

7.1.4 Computational Cost

Specific consideration is given for simplicity of computations and ease of implementation. In the following results, a 5*5 filter is used in the extraction of the L_0 edge map. A separable LPF is used of the form: [1/16, 1/4, 3/8, 1/4, 1/16]. The BPF is defined next as (1 - LPF). This initial BPF is part of any enhancement algorithm. If fewer multiplications are required a 3*3 filter can be used. The non-linearity stage of the proposed algorithm involves bounding the L_0 map followed by scalar multiplication of the resultant image. The scalar multiplication can be incorporated as a filter gain, or as part of the look-up table. It is therefore the look-up table which is the core of the non-linearity operation. If resources allow, a second filtering stage can be added at this time in order to remove any low-frequencies present in the resultant L_{-1} map, thus adding only the high-frequency response to the given input G_0. Experiments have shown that the second filtering operation is not critical for achieving good enhancement results. Thus, it can be ignored for real-time application domains.

7.2 Enhancement Results

In this section we show experimental results which indicate that the enhancement routine augments the frequency content of an input image achieving a visually enhanced output.

The first result exemplifies a zoom-in application. Here, an image is zoomed-up by a 2 to 1 ratio using the expand operation described above (Section 3). The absence of the high-spatial frequencies makes the image appear soft or blurry. We wish to see if the system can enhance the image sufficiently, thus saving in the required bandwidth for the transmission of a full resolution image. The zoomed-in input is shown in Fig. 7.5. This is part of a monkey's head. The output of the proposed enhancement technique is presented in Fig. 7.6. We can see much more detail in the hair region and perceive more texture in

the nose and eye regions. Fig. 7.7 presents the given and enhanced images, left to right respectively, together with their corresponding power spectrum characteristics (bottom). It is evident that the input power spectra is augmented. The enhancement process actually extrapolates to higher frequencies, thus producing the enhanced result.

In the following result we wish to exemplify the difference between high-frequency enhancement techniques available in the literature (sometimes referred to as the unsharp masking method), and the nonlinear analysis scheme presented in this work. Fig. 7.8 exhibits the monkey image example (part of this image was used in Fig. 7.5). A blurred input is presented at the top-left corner. Augmenting the high-frequency components present in the given image results in an enhanced image, as shown top-right. The result of applying the algorithm presented in this work is depicted in the bottom of the figure. We get an overall enhancement perception. The differences are evident in the hair, whiskers and eye regions. Fig. 7.9 displays the corresponding power-spectra characteristics. The power spectra at the bottom of the figure has its higher frequencies augmented.

Our final example is a rockscene image displayed in Fig. 7.10. The top figure presents the enhancement results. The bottom figure displays the corresponding power characteristics. The blurred input, which can be the result of cutting-off high-frequencies due to bandwidth considerations or a "zoom-in" application, is presented at the top-left corner. The original image, which we are assuming is not available to the system and which we wish to reproduce, is presented on the top right. The result of applying the algorithm presented to the blurred input is depicted in the bottom of each figure. We get an overall enhancement perception. The enhanced image very closely matches the original one and the power spectra of the enhanced image is very close to the original power spectra.

7.2.1 Comparison With Other Work

In the described enhancement scheme the focus is on high-frequency augmentation with phase-coherent characteristics. We are not aware of any other work in the literature that follows similar objectives. A different nonlinear filter for image enhancement has been proposed recently by Mitra [MLLY91]. The filter behaves like a local-mean-weighted highpass filter. The basic diagram of the image enhancement algorithm is the same (see Fig. 7.1). Still, the motivating background is very different resulting in interesting differences. In our algorithm a homogeneous filter is used for the extraction of the high-frequency L_0

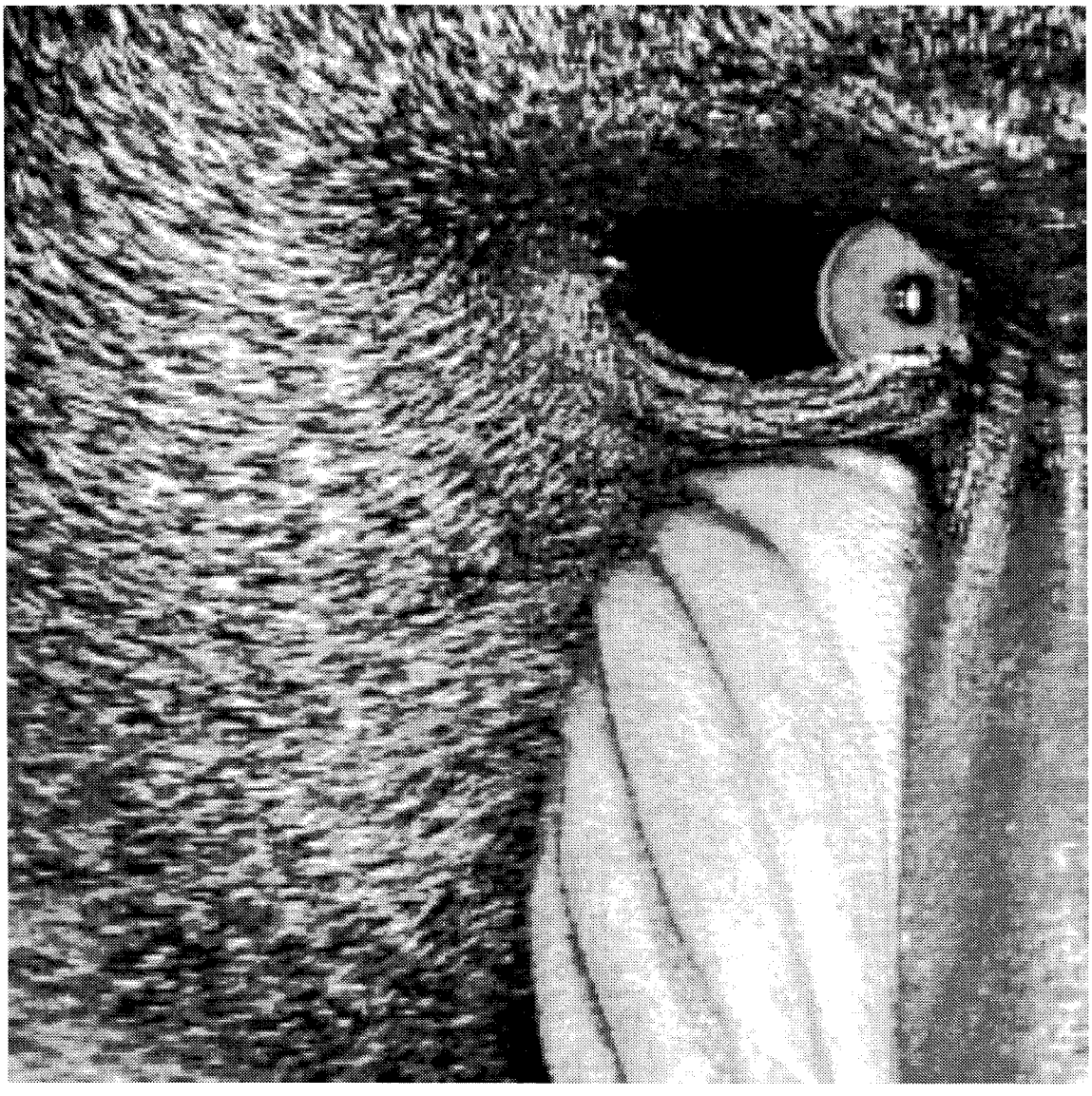


Figure 7.5: Zoom-in application - the zoomed-in input

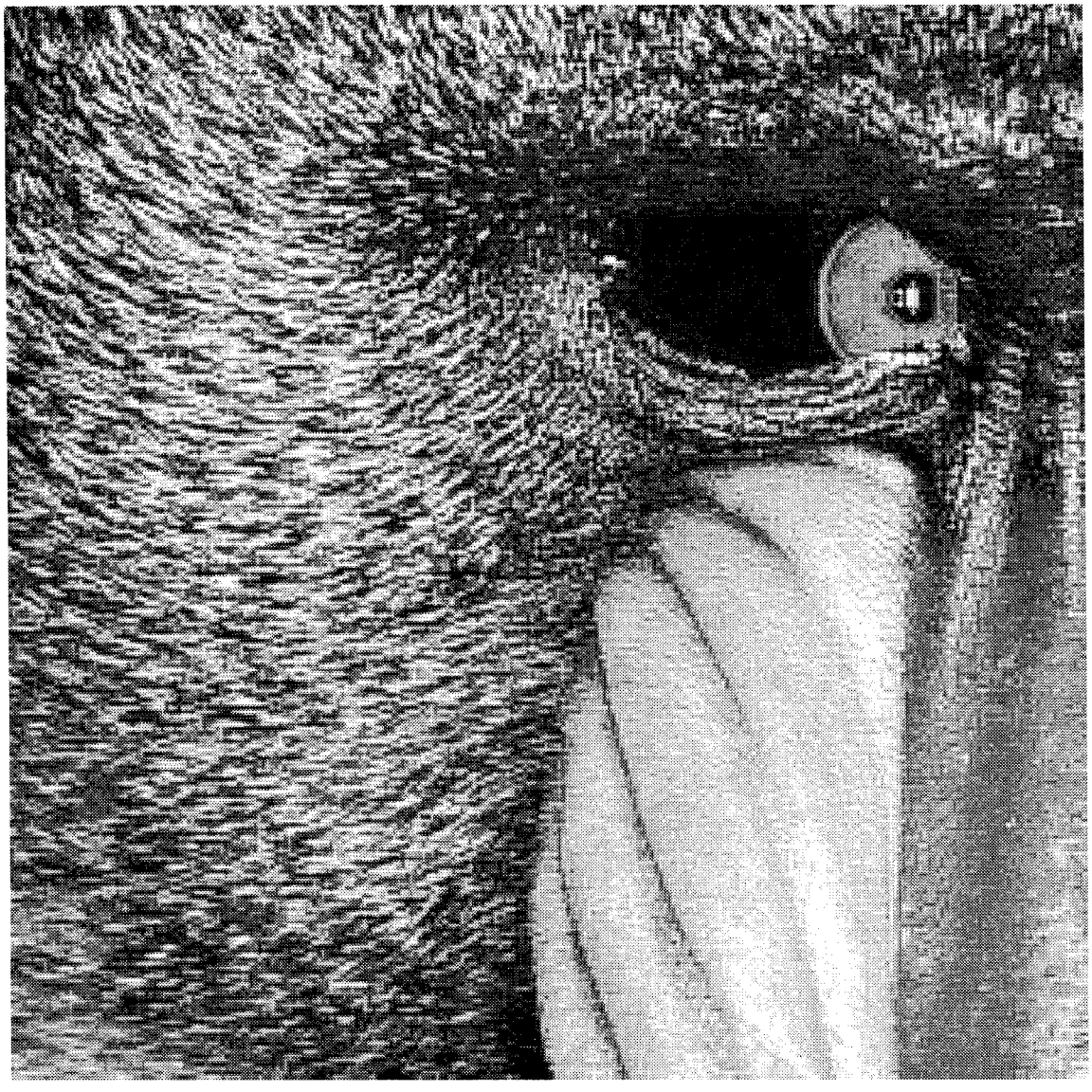


Figure 7.6: Enhanced output

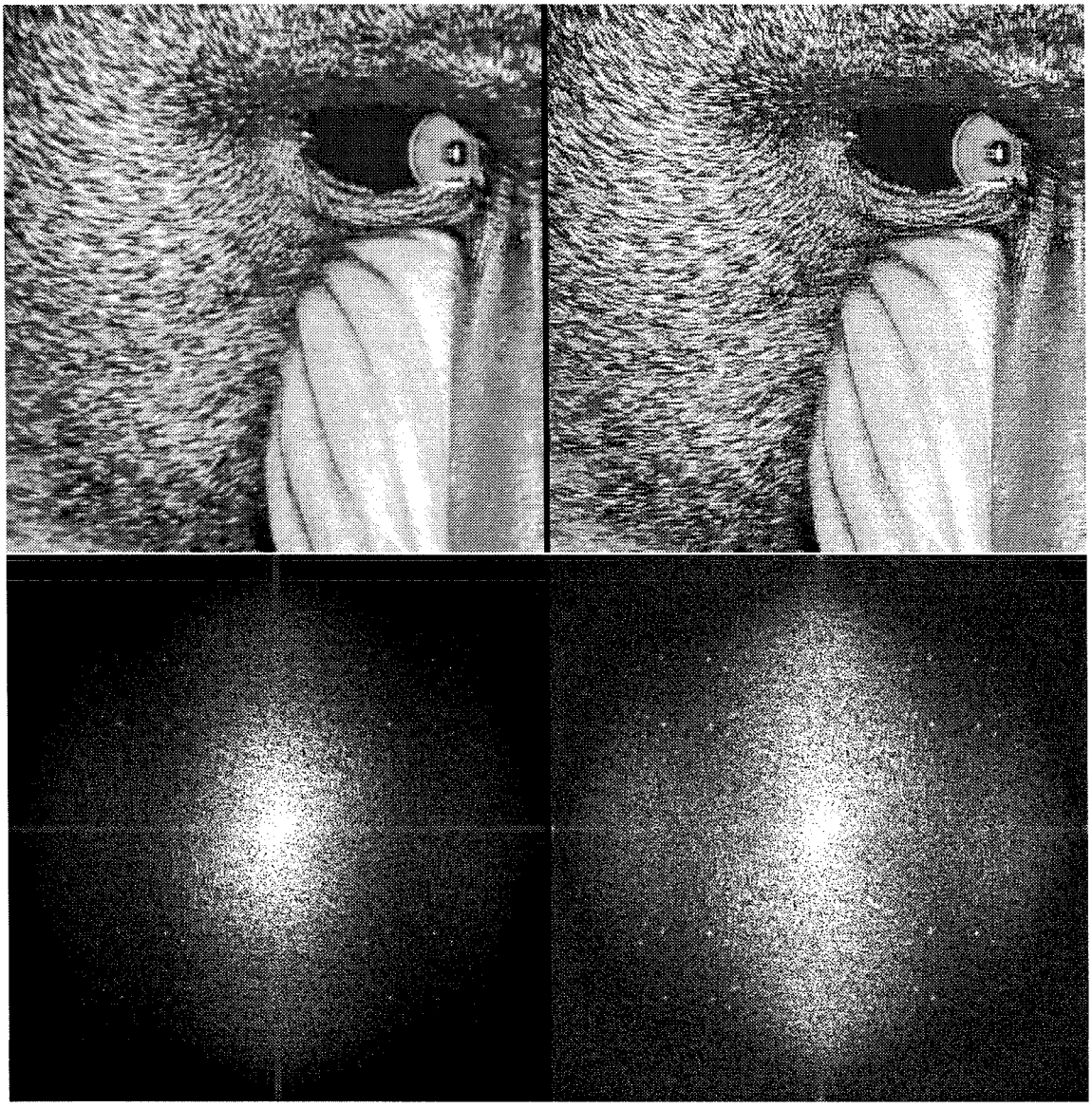


Figure 7.7: Given and enhanced images, left to right respectively, together with their corresponding power spectrum characteristics (bottom). It is evident that the input power spectra is augmented. The enhancement process actually extrapolates to higher frequencies, thus producing the satisfying enhanced result.

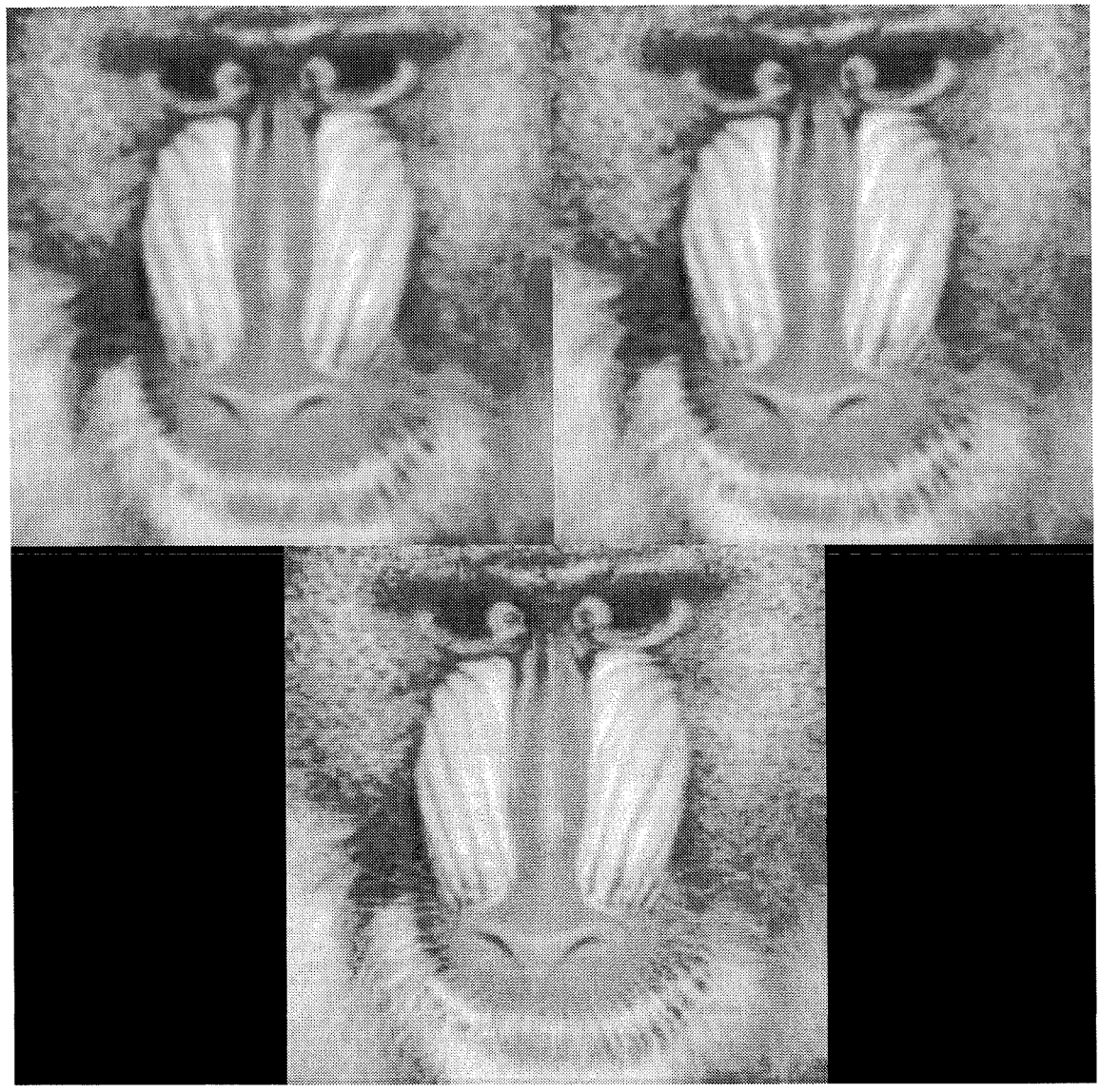


Figure 7.8: Enhancement algorithm results, monkey image. A blurred input is presented at the top-left corner. Augmenting the high-frequency components *present* in the given image results in an enhanced image, as shown top-right. The result of applying the algorithm presented in this chapter is depicted in the bottom of the figure.

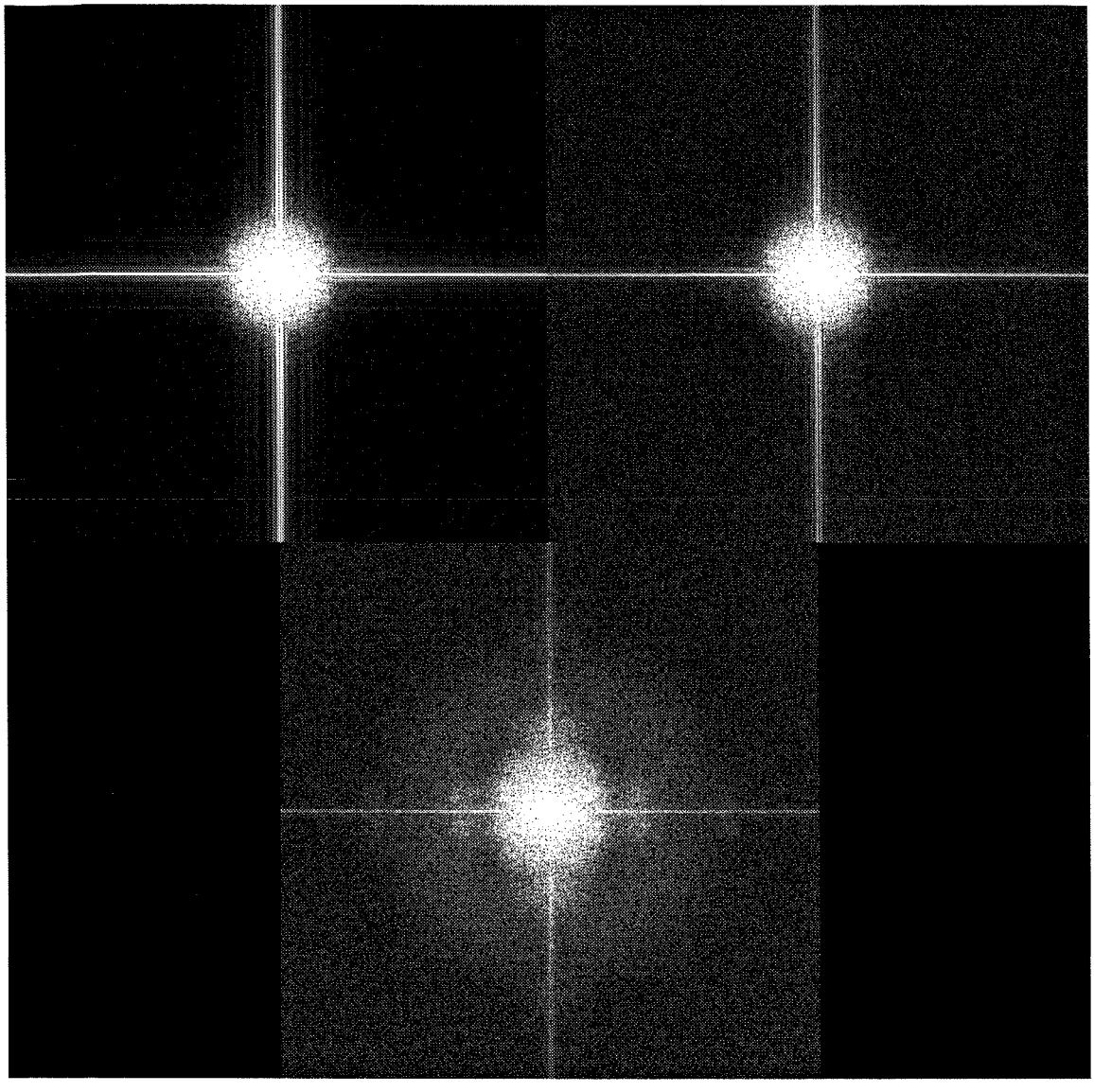


Figure 7.9: Corresponding power spectra characteristics of the monkey image.

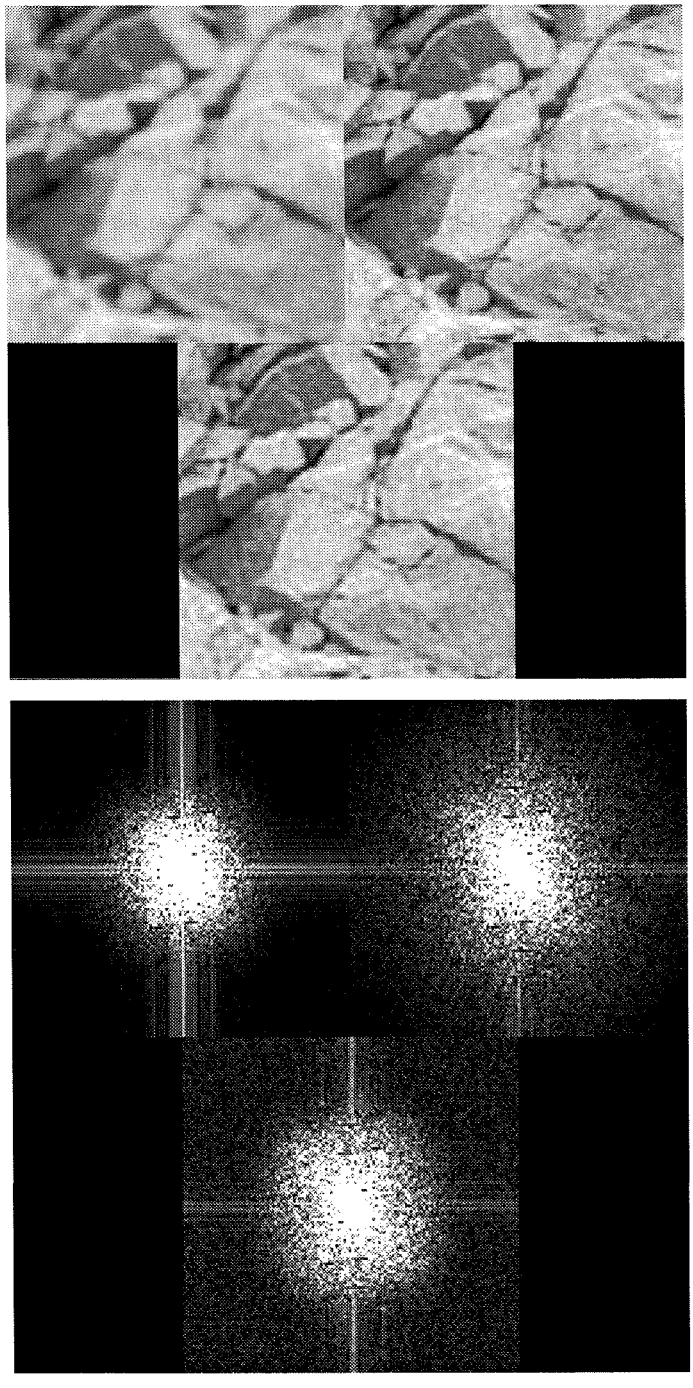


Figure 7.10: Enhancement results (top figure). Corresponding power-spectrum characteristics (bottom figure). In each of the above figures the blurred input and original image are presented (top left and top right, respectively) followed by the enhanced output (bottom). Both visual perception enhancement and power-spectrum augmentation are evident.

map. This is part of the pyramid generation. In Mitra's work, the filtering is biased either in the vertical and horizontal directions or along the diagonal directions - with the latter chosen as giving better performance. The difference between the 2 filter performances can be detected in the example of Figure 7.11. The input image is presented in Fig. 7.11. The enhanced output of Mitra's algorithm is shown in Fig. 7.12. The output of our enhancement algorithm is shown in Fig. 7.13. Both outputs are nice enhancement results. The differences can mainly be detected in the pants region, where our algorithm has less aliasing effects, and in the scarf (ribbon area) - where an undesired zig-zag effect is detected in Fig. 7.12. These, we believe, are directly related to the filtering characteristics.

The non-linearity of the enhancement process is introduced in Mitra's work via a multiplication of the highpass filter by the local mean. This has the effect of adding less of the high-frequency components to the dark regions and more to the brighter ones, and can be desirable for a smoother perception of the enhanced result. In our approach, the non-linearity is introduced via a bounding function. This also has the effect of introducing a stronger high-frequency component to brighter areas than to darker ones. A major difference between the two non-linearities is that the phase of the edges is preserved in our algorithm, following the scale-space formalism. The procedure outlined in Mitra's algorithm, however, has the effect of shifting the phase towards the brighter region. It remains to be investigated if this might cause any undesired effects.

Computationally, the main difference between the approaches is in the chosen non-linearities. Here, the core of our enhancement algorithm is the look-up table. This competes with the multiplication of the highpass filter output with the local mean in Mitra's algorithm.

7.2.2 Summary of Results

The presented enhancement technique is only a first step towards what can be accomplished by extrapolation across scale. Edges are only one major scale invariant features. Lines and dots, for example, require additional analysis. It may be desirable to use an adaptive threshold rather than the constant value used, although this has a side-effect of introducing phase shifts whose effects on the perceived sharpness are unknown at the present. As is always the case, a tradeoff exists between high-frequency enhancement and noise generation. The enhancement scheme will work best on input images which have

been previously processed for noise. One possibility is to use reconstruction schemes which remove noise while unavoidably blurring the image, and then enhancing the resultant image with the algorithm described in this work.

In conclusion, we have described a enhancement scheme that could very well address the most important features required in producing visually pleasing enhanced resolution versions of existing images. The simplicity of the computations involved and ease of implementation enable it to be incorporated in real-time applications such as high-definition television (HDTV).

7.3 Combining Image Enhancement with Pyramid Coding

In this section we present an initial attempt to combine the image enhancement scheme, described in the preceding part of the chapter, with the pyramid coding scheme. The convergence of the two fields of image processing and image compression (otherwise called "second generation" image coding), is the result of a growing need to handle large amounts of image data, be it in transmission or in automated image handling (such as image database query and retrieval), with the classical compression schemes reaching their limits. It is now accepted that in order to achieve more advanced compression schemes we need to use our knowledge about images and about their characteristic behavior, to an advantage.

At the image processing end we use our knowledge about the behavior of edges across scale (across different resolutions) in order to extrapolate in scale and increase the resolution of a blurred input image. The ability to extrapolate in scale is very useful for compression. We can think about saving bandwidth by not transmitting certain frequencies, and trying to reconstruct the information back at the receiver's end; we can think about combining the enhancement scheme with existing image compression schemes, such as the pyramid coding schemes, to achieve additional savings; and finally, we can use this ability in progressive transmission applications, whereby the lower resolution images get enhanced and thus information can be extracted at earlier stages of the transmission, saving in analysis time.

The pyramid representation as a means for compression will be briefly described, followed by the combination with image enhancement for further coding. The combination of image enhancement with progressive transmission will be suggested as well.

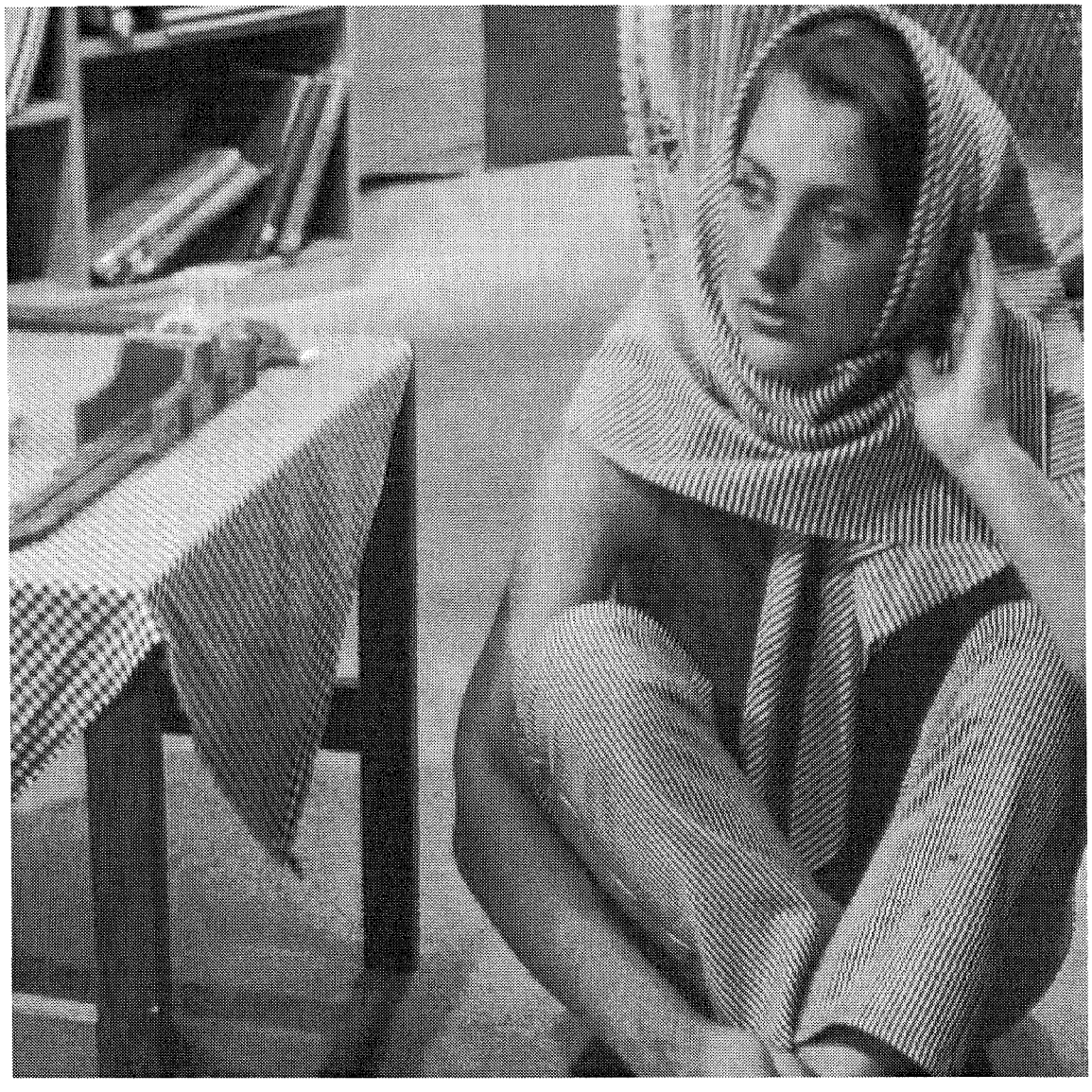


Figure 7.11: Input image

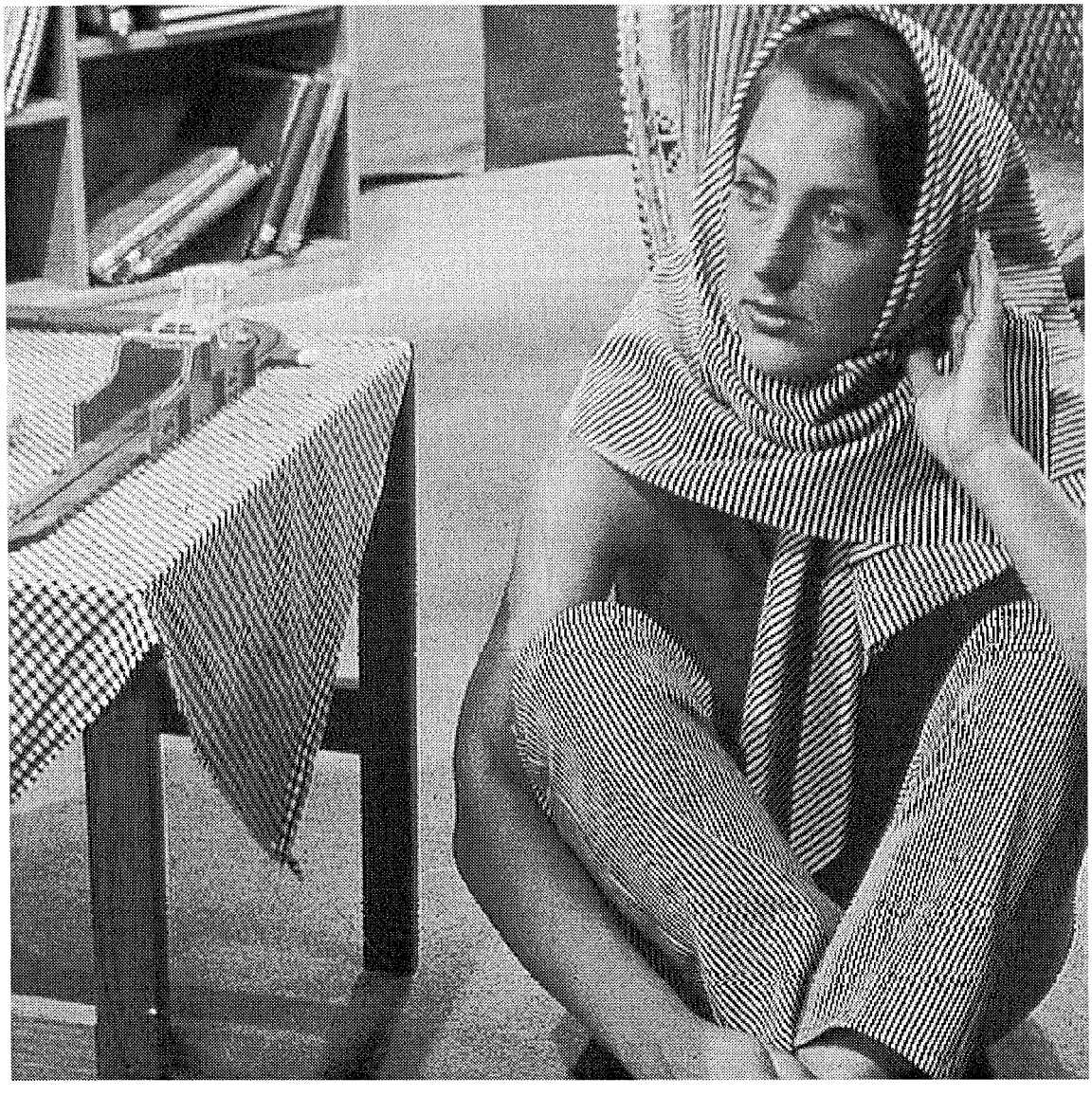


Figure 7.12: Enhanced output of Mitra's algorithm

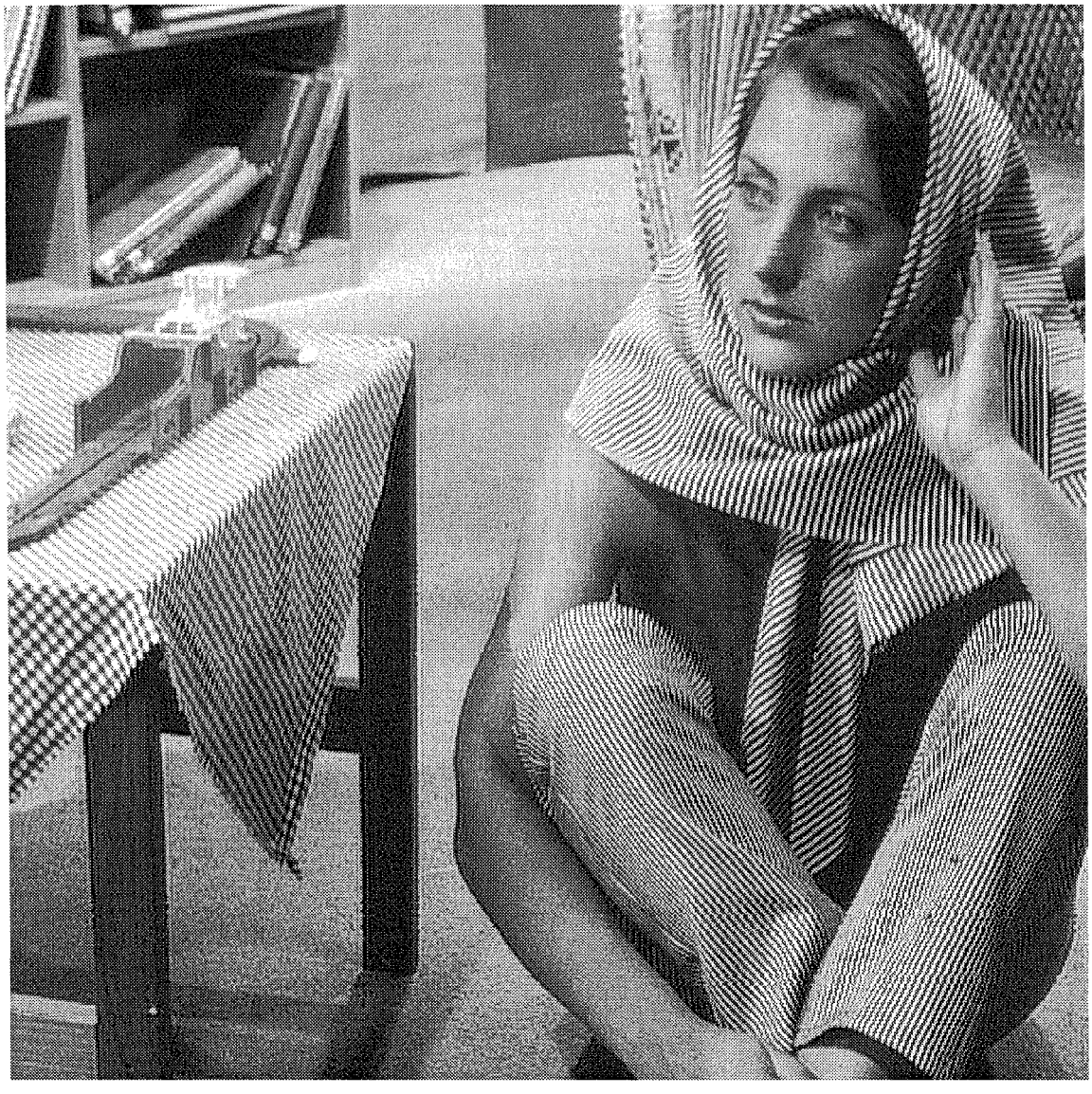


Figure 7.13: Output of our enhancement algorithm

7.3.1 Compression via the Pyramid Representation

The pyramid scheme codes an input image in a mutiresolution representation via the generation of subimages of various scales, as was shown in Fig. 2.2. Low-resolution subimages G_k are created by passing G_{k-1} through a low-passed filter, which we will term H, and a decimation box. In coding the input image we can use subimages $\{L_0, L_1, \cdots, L_K, G_{K+1}\}$ which are obtained by

$$\begin{aligned} L_0 &= G_0 - \widehat{G_1}, \\ L_1 &= G_1 - \widehat{G_2}, \\ &\cdots \\ L_K &= G_K - \widehat{G_{K+1}}, \\ G_{K+1}. & \end{aligned} \tag{7.3}$$

Here, L_k is the difference subimage of the kth level, G_k is the low-resolution subimage of the kth level, and $\widehat{G_k}$ is the interpolated version of G_k (using an interpolation filter F). In order to reconstruct the original G_0 image back we reverse (7.3).

The pyramid representation has been introduced in the literature for coding purposes [BA83], as it was shown to be a complete representation. In the procedure outlined above, perfect reconstruction is guaranteed if there is no quantization of the transmitted data, regardless of the choice of filters H and F. Different quantization and encoding strategies can be applied to the different subimages depending on the signal characteristics. For example, in a linear (e.g., Laplacian) pyramid, the signal variance in different subimages tends to be different. Usually, lower-frequency subimages have higher variance. Therefore, we would allocate a different number of bits to the subimages (more bits per pixel for the higher variance subimages).

Using the pyramid schemes for compression one should realize that the pyramid is an oversampled system (An example a 4/3 overcomplete pyramid was shown in Section 2.3.4). Compared to subband and transform coding schemes which are critically sampled systems, we expect lower compression rates due to the need to transmit more data. Still, there is great interest in the literature in using the pyramids for compression and image transmission purposes. Recent works have taken advantage of the fact that no matter how one designs the decimation filter H and the interpolation filter F, perfect reconstruction

is obtained. Therefore, specially tuned filters for H and F can be incorporated such that the signal characteristics in every level is better suited for compression. Examples include the use of a nonlinear filter scheme for image compression and a motion-compensation filter for video compression [Lee94]. This pyramid characteristic implies that we can take advantage of some nonlinear characteristics of images to help the compression and the results can be competitive with subband and transform coding, both of which have many constraints on designing perfect-reconstruction systems.

7.3.2 Applying Image Enhancement to Pyramid Coding

In this section we combine the image enhancement scheme, described above, with the pyramid coding scheme. We have shown the possibility of predicting the L_0 level of the Laplacian pyramid using lower-resolution edge maps. The next step is to code an image with and without the L_0 component and evaluate the corresponding rate-distortion performance, i.e., investigate the compression savings vs. the output image quality that we can achieve.

The Rate-Distortion Criteria

We decompose the original image, G_0, into $\{L_0, L_1, L_2, G_3\}$. We scalar quantize L_2, L_1, and L_0 and then compute the entropy of the quantized signals. For G_3, we first apply differential Pulse-code modulation (PCM), then compute the entropy. The average entropy of these subimages represents the rate (bits per pixel).

The most popular and widely used distortion criterion is the MSE (mean squared error) or PSNR (peak signal-to-noise ratio), which is defined as:

$$\text{PSNR} = 10 \log_{10} \frac{255^2}{\frac{1}{XY} \sum_{i=1}^{X} \sum_{j=1}^{Y} (I_{ij} - \hat{I}_{ij})^2}, \tag{7.4}$$

where I_{ij} is the original pixel value at position (i, j), $\hat{I}_{ij}$ is the quantized pixel value, and X and Y represent the horizontal and vertical dimensions of the input image, respectively.

The PSNR value is used to tell the quality of the quantized image compared to the original one. Generally, this criterion does give us the desired measure. However, as we will see later on in this chapter, not always does it match the human perception (and subjective judgment). Two additional cases challenging the PSNR as a quality measure

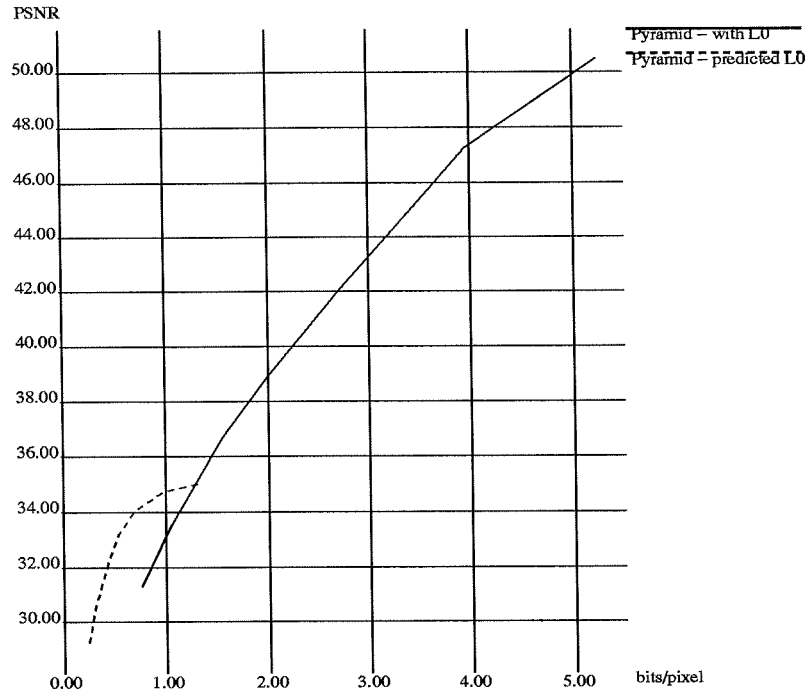


Figure 7.14: Rate distortion curve for Lenna image

(one in which two different image sets were generated by the same compression scheme and compared, the other when comparing the same image set with two different compression schemes), can also be found in [Lee94]. The PSNR distortion measure should overall be used with caution for judging image quality.

Results

The Lenna image is used for this coding task. Fig. 7.14 presents the rate-distortion curves for the pyramid coding of Lenna, with and without the L_0 component. We note that in using the L_0 component we have all pyramid levels and thus the reconstruction would be exact apart from the quantization errors induced. Using a *predicted* L_0 (i.e., the actual L_0 component is not being used) we introduce additional noise in the reconstruction process. In general, we note the slow degradation of the rate-distortion curve using the enhancement scheme, as opposed to the almost linear drop of the original (non-enhanced) curve. Even of more interest is that at very low bit-rates, the ability to estimate the L_0 component from the given L_1 component, or the ability to extrapolate in frequency space, allows for *better* PSNR.

An example of two images, with and without the L_0 component (top left and top right, respectively) is shown in Fig. 7.15, as compared to the original Lenna image (bottom). Both images are coded with approximately 1 bit/pixel. We have 0.99 bits/pixel with PSNR=34.77dB for the enhanced image and 1.053 bits/pixel with PSNR=33.54dB for the image decompressed with all its L_i components. In this case we get better PSNR, and better perceived similarity to the original, for the enhanced image with the predicted L_0 component, than for the image with *all* components present. This is a very interesting and encouraging result.

Next, we compare the pyramid compression with the discrete cosine transform (DCT). Fig. 7.16 presents the rate-distortion curves including DCT. The DCT clearly "wins" the PSNR comparison. We note that at the very low bit rates the differences are quite minimal. In addition, we need to compare the actual images, as opposed to the PSNR ratios, as is shown in Fig. 7.17. We note that the blockiness with the DCT is very evident and possibly more distracting to the eye than the artifacts introduced by the pyramid+enhancement scheme. A zoom-in image taken from Fig. 7.17 is presented in Fig. 7.18. In the DCT coding scheme we can see strong blocking effects in the quantized image (this is the case especially when the bit rate is low or when we zoom the image up). This phenomenon results from the independent quantization of blocks. This enforces the claim that the PSNR does not in all instances match our visual perception.

A similar investigation is done on a moon image, whose rate distortion curves are shown in Fig. 7.19.

As before, we note the slow degradation of the rate-distortion curve with the enhancement. We see again that at low bit-rates, better PSNR is achieved by *predicting* the L_0 component via the enhancement processing stage. When comparing with the DCT rate-distortion curve (Fig. 7.19 bottom) we notice that at very low bit rates we actually achieve *better* performance than the DCT.

We conclude this section with a few of the moon images. Fig. 7.20 displays the slow degradation phenomenon. Two images are displayed. The left one has 1.27 bits/pixel with PSNR=31.69dB. The right image has almost *half* the bit rate, at 0.65 bits/pixel with a very similar PSNR value of 31.1dB. The two images look identical. In Fig. 7.21 we compare the pyramid scheme to DCT at the low bit rate of 0.47 bits/pixel. The PSNR ratio is larger for the pyramid coding in this case, PSNR=30.53dB, where the DCT case

Figure 7.15: Lenna image comparison, with and without L_0. Top left: Including L_0 in the compression, Top right: Using a predicted L_0, Bottom: Original Lenna image.

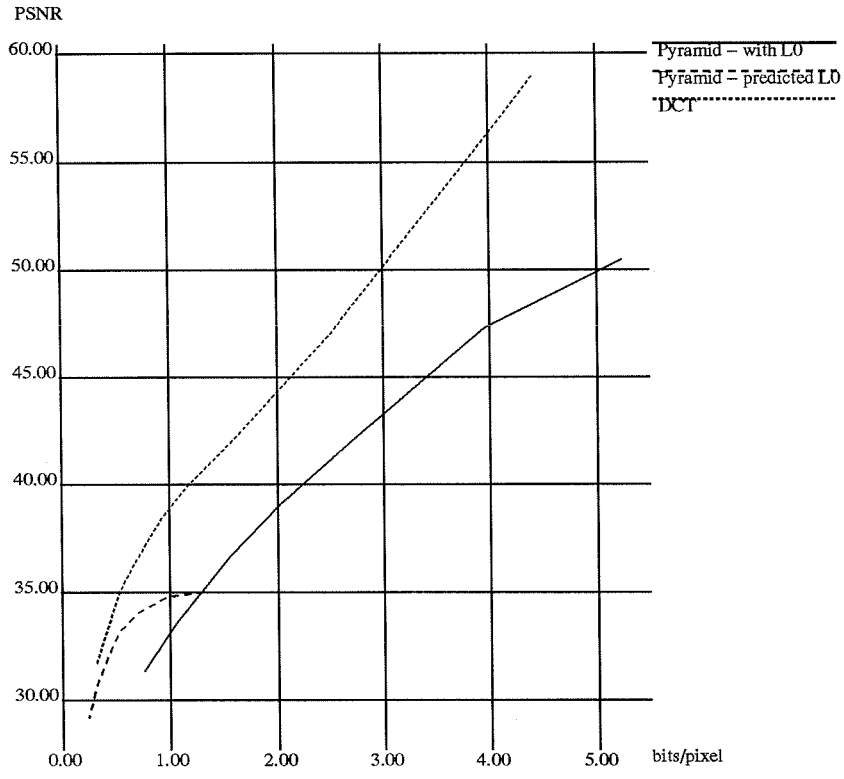


Figure 7.16: Rate distortion curves for Lenna, including DCT

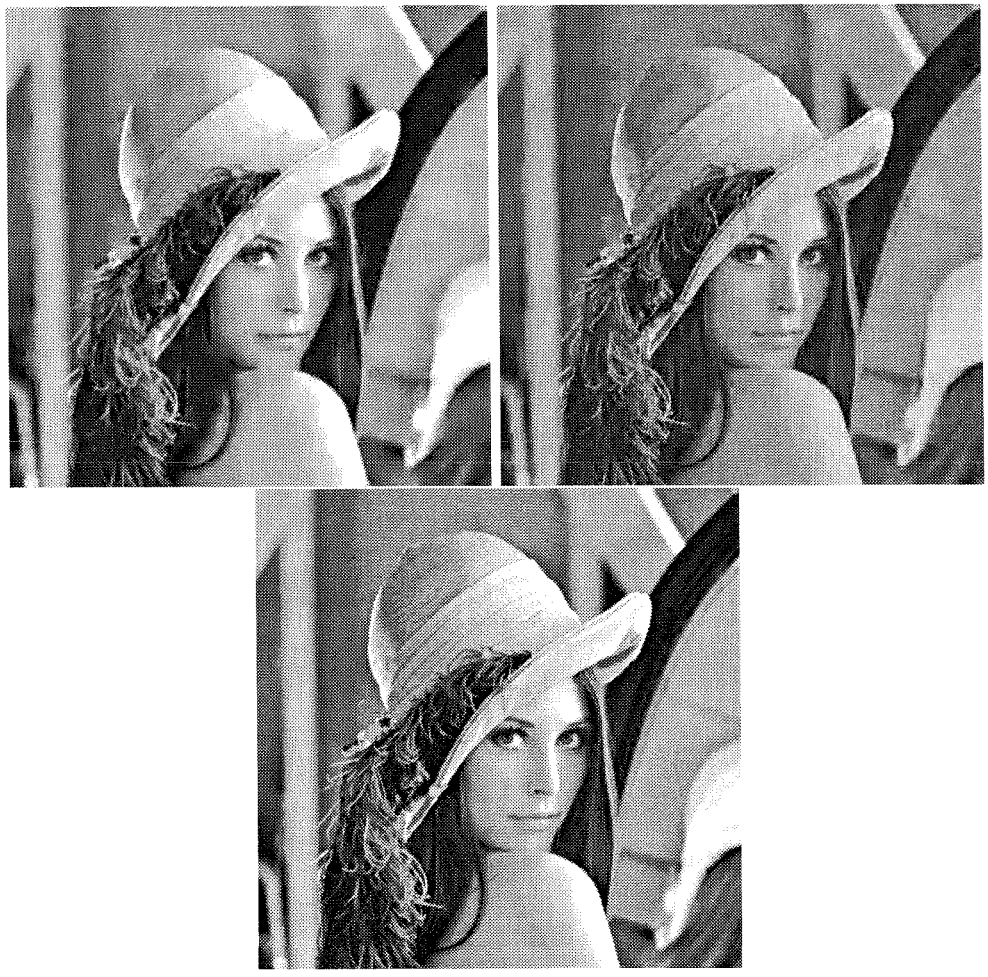


Figure 7.17: Lenna image compressed with pyramid scheme + enhancement (top left) and with DCT (top right). The original image is on the bottom.

Figure 7.18: Pyramid vs. DCT

has PSNR=30.49dB. The blockiness of the DCT is certainly visible here. Note the blocks on the main rocks which actually degrade the possibility of identifying rock boundaries etc.

7.3.3 Image Enhancement and Progressive Transmission

An additional interest is the combination of image processing with progressive image transmission schemes. Here, instead of looking for additional bit compression capabilities, we are interested in achieving a compression in *time*. By this we are referring to progressively transmitting information, from low resolution to high resolution, with the desire to extract information during the transmission, without waiting to receive the high-resolution image. More so, we would like to determine at an early stage of the transmission process if the image is of interest, so as to determine if the high-resolution image is to be transmitted at all.

In Fig. 7.22 we demonstrate the combination of an Integer Subband Coding (ISBC) scheme of the GASPARA image, with the enhancement scheme. This result is part of an image compression effort being pursued at JPL (Jet Propulsion Lab, NASA). We note the possibility to detect craters and other points of interest much more clearly in the enhanced images, even at extreme compression ratios. To the scientist this can be a tool to determine his interest in the region. If it does look interesting, the full resolution image can be transmitted, without any loss.

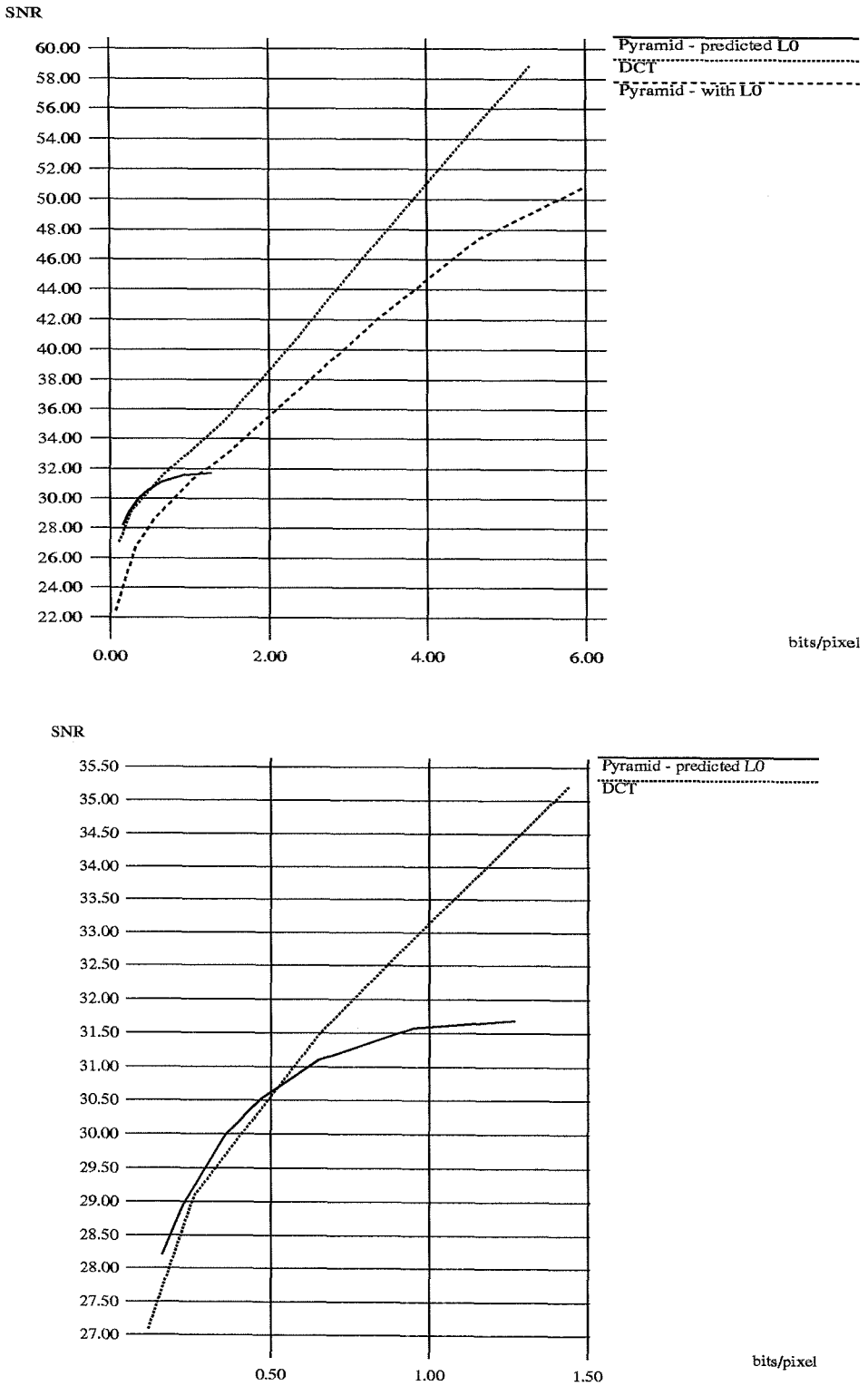


Figure 7.19: Moon image - Rate distortion curves for moon image (top); Comparison with DCT - zoom in (bottom)

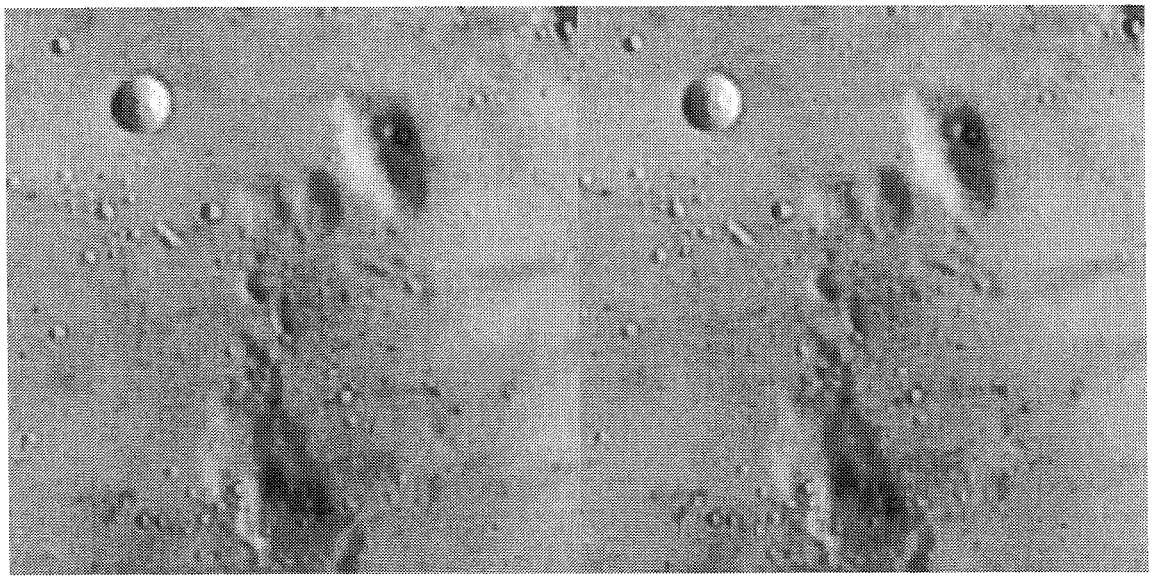


Figure 7.20: Slow degradation phenomenon at low bit-rates

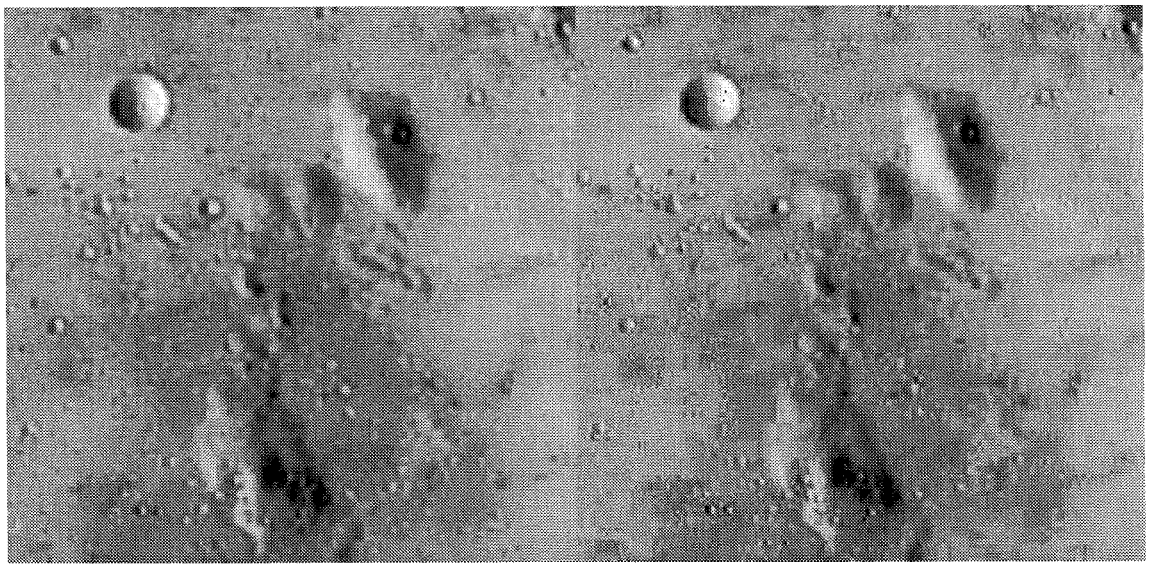


Figure 7.21: Comparison of Pyramid vs DCT at low bit-rates

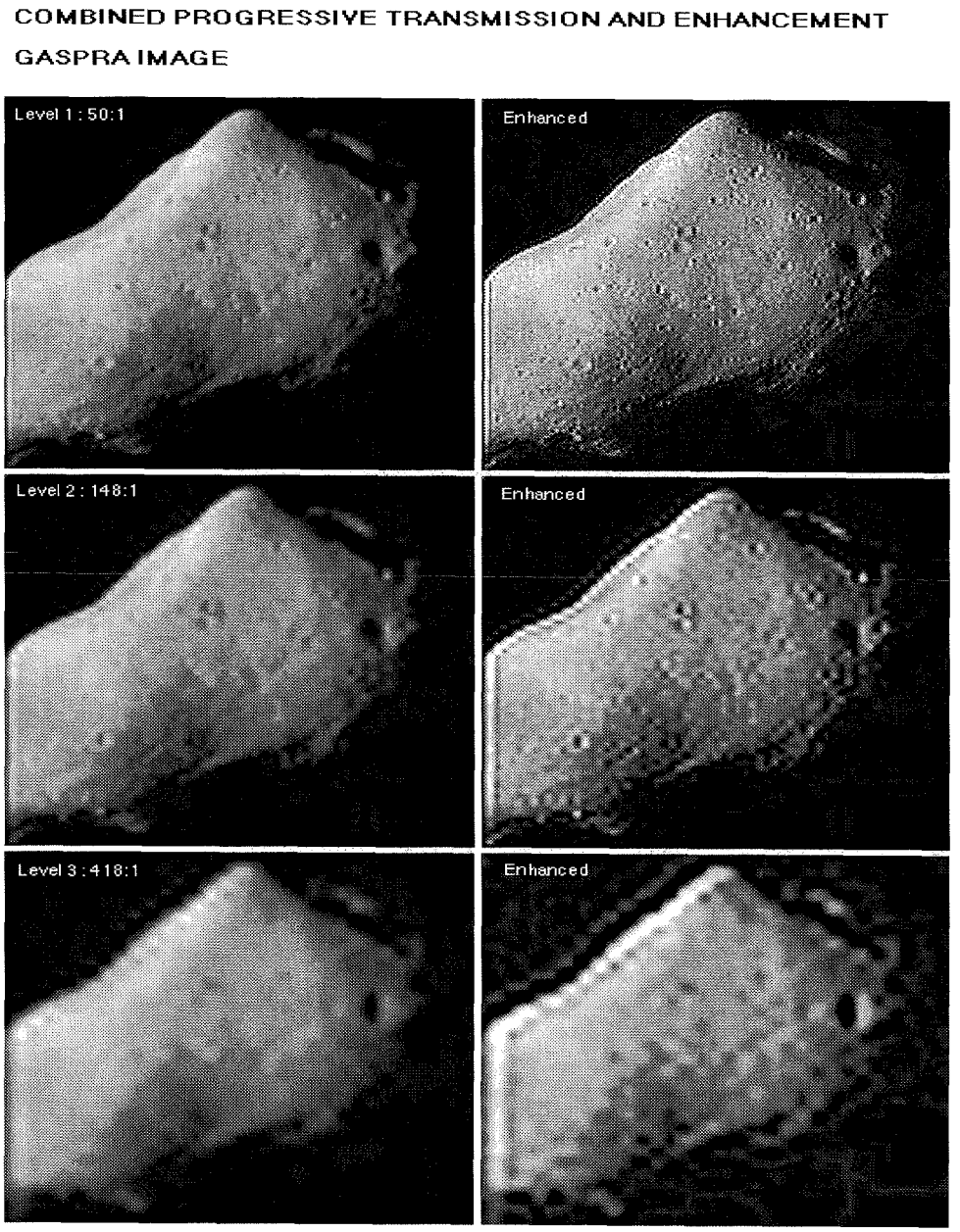


Figure 7.22: Combining image enhancement with progressive transmission

7.3.4 Summary and Conclusions

We have done a preliminary analysis on the combination of image enhancement with image compression schemes. Encouraging results have been achieved with the pyramid compression scheme, especially at low bit rates (which is the challenging frontier in the compression field). Finally, the case for including enhancement in progressive image transmission was made, with results indicating the enhanced visual perception at low resolutions.

Following are several issues which stem out of this work, for further exploration:

Extending the Enhancement scheme

- It is interesting to extend the prediction capability to lower-resolution scales (higher levels of the pyramid). This will enable extending the compression from the L_0 level to L_1 level etc. Initial investigation indicates that this is not a simple extension to the existing algorithm. As the resolution is decreased substantially, it is more difficult to locate the edges which are the important starting point in an enhancement scheme (but see possible extensions to the pyramid below). In addition, the sharpening process will require more investigation as to how to "fill-in" the regions in the image which have been blurred and now have been sharpened. Overall, pursuing this idea requires further research.

- In order for the enhancement scheme to take part in any real-time video application several additional characteristics need to be added on. These include enhancing color images, motion sequences, real time considerations etc. The simplicity of the algorithm and its low computational cost (see Section 7.1) ensure that real-time applications are possible. Colored images can be separated into three channels termed YUV. We assert that enhancing the Luminance channel (Y) and adding it back to the color-difference signals (U,V), will give the desired enhancement effect. This claim still needs to be shown in practice.

- The initial motivation for the enhancement algorithm was a simple and effective procedure. The final goal was the subjective human perception. When combining with compression, we use the PSNR criterion which actually shifts the goal to one of reconstruction, i.e., the comparison is to the original image pixel values. We

have demonstrated that this criteria is not always in agreement with the human perception. Still, if we do choose to use it, an additional step needs to be incorporated into the enhancement algorithm to preserve the average amplitude of the original image as it is being enhanced.

Chapter 8
Conclusion

In this thesis we have shown a variety of uses for the multi-resolution pyramid, both in the texture recognition task and in image enhancement and coding.

The usefulness of the oriented Laplacian pyramid as a feature extraction stage in a recognition system was demonstrated. Additionally we proved that the pyramid spans the orientation space and can be made steerable, thus allowing for a rotation-invariant representation space. This part of the work has produced a computationally efficient pyramid filtering scheme, as well as a more compact steerable pyramid than has previously been shown in the literature.

A texture recognition system has been presented which combines the pyramid filtering stage with learning for the classification stage of the system. We have focused on an information theoretic rule-based scheme, which extracts probabilistic rules between the attributes and the output classes. This framework combines the advantages of Baysian probabilistic analysis and the parallel nature of neural-networks. State-of-the-art results were presented for the texture recognition task. The generalization of the system to rotation-invariant recognition as well as high-accuracy orientation angle detection were demonstrated.

We have concentrated in this work on the texture classification application. Texture is one of the main visual modules which help us analyze the 2D world around us. A system that recognizes texture can take part in automated scene analysis, remote-sensing applications, medical imagery analysis and more. The building blocks of the presented system are general ones. Thus we feel that the system can be generalized to other domains, such as shape recognition. The application to face recognition is suggested for future research.

The use of the multi-resolution pyramid for image enhancement has been the topic of the latter part of the thesis. The field of image enhancement is gaining renewed interest in the communication community. As the industry is heading towards high-compression digital systems the ability to augment the visual perception of images to the user is a very important one. In this work we have utilized the pyramid concepts to an advantage in a

new enhancement scheme. Caltech is currently patenting this system. The future work in this domain can include short term goals, such as the application to color images and video, as well as longer-term goals such as the combination with image-coding schemes.

Appendix A
The K-Means Clustering Algorithm

The K-means algorithm is a statistical clustering technique which consists of an iterative procedure of finding K means in the sample space, following which each input sample is associated with the closest mean in Euclidean distance.

The iterative procedure:

1: Choose random initial values for the means, $\mathbf{m}_1, ..., \mathbf{m}_K$.

2: **Loop**- Classify the given n samples by assigning them to the class of the closest mean; i.e., for each sample $\mathbf{x}$ choose mean i such that:

$$||\mathbf{x} - \mathbf{m}_i||^2 = min_k||\mathbf{x} - \mathbf{m}_k||^2, k = 1..K. \tag{A.1}$$

3: Recompute the means as the average of the samples in their class.

4: If any mean changed value go to **Loop**; otherwise stop.

In our implementation, K means are independently extracted in each dimension of the 15-dimensional space. Each floating point sample from the filter maps gets associated with corresponding bins in each dimension, resulting in a 15 dimensional quantized feature vector.

Picking the parameter K correctly is usually of great importance when clustering the entire space. In our case, the fact that the clustering stage is a preprocessing step, prior to the rule-based network classification, seems to reduce substantially the difficulty of picking the appropriate number of clusters. The sensitivity of the system's performance on the parameter K is shown in Experiment 4. of Section 3.4.

Appendix B
More Details of the ITRULE Classifier

B.1 Finding the Initial Rule Set

The ITRULE classification algorithm takes sample data in the form of discrete attribute vectors and generates a set of K rules, where K is a user-defined parameter. The set of generated rules are the K most informative rules from the data which predict the class variable as defined by the J-measure, i.e.,

$$J(\mathbf{X} = x; y) = p(y)\Big(p(x|y)\log\Big(\frac{p(x|y)}{p(x)}\Big) + (1 - p(x|y))\log\Big(\frac{(1 - p(x|y)}{(1 - p(x))}\Big)\Big),$$

where y is the left-hand side value of the variable $\mathbf{Y}$, and $\mathbf{X}$ is the class variable, where x is a particular class.

The algorithm proceeds by first finding K rules, calculating their J-measures, and then placing these K rules in an ordered list. The smallest J-measure, that of the$K$th element of the list, is then defined as the running minimum J_{min}. From that point onwards, new rules which are candidates for inclusion in the rule set have their J-measure compared with J_{min}. If greater than J_{min} they are inserted in the list, the Kth rule is deleted, and J_{min} is updated with the value of the J-measure of whatever rule is now Kth on the list. For each of m possible right-hand sides (corresponding to m possible output classes), the algorithm employs *depth-first* search over possible left-hand sides, starting with the first-order conditions and specializing from there. The algorithm systematically tries to specialize all $m * N * 2m$ first-order rules, with N = total number of attributes (here $N = 15$), and terminates when it has determined that no more first-order rules exist which can be specialized to achieve a higher J-measure than J_{min}. For details on the various bounds used to constrain the search see [SG92].

B.1.1 Pruning the Rule Set

By using the J measure, ITRULE finds the best rules with each rule in *isolation*, not necessarily the best *rule set*. Two pruning algorithms are mostly used for reducing the extracted rule set while preserving high classification accuracy (for other pruning methods see [GHMS92]). In the first, *subsumption* pruning, higher order rules that have lower

information content than a corresponding lower order (more general) rule are omitted from the rule list. The second, *independence* pruning, enforces the conditional independence assumption (for a *given* class, the input attributes are independent). The algorithm starts with all rules in the set and checks each pair with the same class output for dependence. Dependence is defined as follows: if the two rules have condition sides y_1 and y_2, the rules are dependent if

$$\frac{|p(y_1, y_2) - p(y_1)p(y_2)|}{p(y_1, y_2)} > T \tag{B.1}$$

where $0 < T < 1$ is a user-set threshold. The dependent rule with the lower J-measure is removed from the rule set. This algorithm has proven to be computationally efficient and has a high generalization performance.

B.2 Class Probability Estimation and Classification

Let $|\mathcal{F}|$ be the set of rules which fire and $s_1, \ldots, s_{|\mathcal{F}|}$ be the actual attribute-value conjunctions corresponding to the fired rules.

For any particular class x_i, we have by Bayes' rule:

$$\begin{aligned} p(x_i|s_1, \ldots, s_{|\mathcal{F}|}) &= \frac{p(s_1, \ldots, s_{|\mathcal{F}|}|x_i)p(x_i)}{p(s_1, \ldots, s_{\mathcal{F}})} \\ &= \frac{\prod_{j=1}^{|\mathcal{F}|} p(s_j|x_i)}{p(s_1, \ldots, s_{|\mathcal{F}|})} p(x_i) \\ &\qquad \text{(assuming conditional independence given the class)} \\ &= \frac{\prod_{j=1}^{|\mathcal{F}|} p(s_j)}{p(s_1, \ldots, s_{|\mathcal{F}|})} p(x_i) \prod_{j=1}^{|\mathcal{F}|} \frac{p(x_i|s_j)}{p(x_i)} \\ &\qquad \text{(by Bayes' rule).} \end{aligned}$$

Let us define the weights W_{ij} as

$$W_{ij} = \log \frac{p(x_i|s_j)}{p(x_i)},$$

a bias term for each class as

$$t = \log p(x_i)$$

and an (as yet) undetermined constant

$$C = \log \frac{\prod_{j=1}^{|\mathcal{F}|} p(s_j)}{p(s_1, \ldots, s_{|\mathcal{F}|})}.$$

Hence, we get that

$$\log p(x_i|s_1, \ldots, s_{|\mathcal{F}|}) = C + t + \sum_{j=1}^{|\mathcal{F}|} W_{ij}.$$

Classification occurs simply by picking the maximum over these estimates for all possible classes x_i, while actual probability estimates are derived by normalizing these probability estimates to sum to 1, hence eliminating the constant C. Note that there is an implicit assumption here that the constant C is the same for each class x_i. For additional details see [GHMS92].

Appendix C
Derivation of the Interpolation Functions via Gram-Shmidt

C.1 Steering the Pyramid Filters

Rather than using equation 4.8 directly, we first use a derivation based on Gram-Shmidt and then show the compatibility of the two approaches. Our steps include finding the steering coefficients after orthonormalizing the basis kernels via Gram-Shmidt, finding the interpolation functions as the inner-product solution in equation 4.9 and then adjusting back to the original basis set.

Let us term the extracted pyramid basis as the eight functions O_1 thru O_8. The first step towards obtaining the steering coefficients is to employ the Gram-Schmidt process to orthonormalize the O_k's as follows:

$$\varphi_1 = O_1 \ , \ \varphi_{r+1} = O_{r+1} - \sum_{k=1}^{r} \frac{\langle O_{r+1}, \varphi_k \rangle}{\|\varphi_k\|^2} \varphi_k \text{ for } r = 1, 2, ..., k-1 \tag{C.1}$$

followed by the normalization step

$$\phi_k = \frac{\varphi_k}{\|\varphi_k\|}. \tag{C.2}$$

It is desirable to express the orthonormalization process in terms of a single matrix transformation, in the form

$$\phi_k = \sum_j \gamma_k^j O_j, \tag{C.3}$$

where γ is the matrix of orthonormalized Gram-Schmidt coefficients acting on O_k. The γ matrix can be arrived at by first forming a matrix of or*thog*onalized Gram-Schmidt coefficients, which we will call ζ, and then normalizing each row of ζ. By itself, ζ will accomplish orthogonalization of the O_k's as in Equation C.4:

$$\varphi_k = \sum_j \zeta_k^j O_j. \tag{C.4}$$

The ζ matrix is formed by filling the entries below the diagonal of a $k \times k$ identity matrix with Gram-Schmidt coefficients given by the recursive formula:

$$\zeta_m^n = \sum_{h=0}^{m-2} -\mu_{n+h}^m \zeta_{n+h}^n \text{ for } m > n, \tag{C.5}$$

where

$$\mu_n^m = \frac{\langle O_m, \varphi_n \rangle}{\|\varphi_n\|^2}. \tag{C.6}$$

The elements of kth row of ζ are then each divided by $\|\varphi_k\|$ to result in γ. As an example, the 3×3 gamma matrix is:

$$\gamma = \begin{bmatrix} \frac{1}{\|\varphi_1\|} & 0 & 0 \\ \frac{-\mu_1^2}{\|\varphi_2\|} & \frac{1}{\|\varphi_2\|} & 0 \\ \frac{-\mu_1^3 - \mu_2^3(-\mu_1^2)}{\|\varphi_3\|} & \frac{-\mu_2^3}{\|\varphi_3\|} & \frac{1}{\|\varphi_3\|} \end{bmatrix}. \tag{C.7}$$

Having obtained a means of orthonormalization of the set of original functions O_k, we can proceed to describe a method of obtaining the set of steering coefficients $\beta_k(\theta)$ for O_k. First we will consider a set of coefficients $\alpha_k(\theta)$ for use with the orthonormalized basis ϕ_k. With F_θ representing the ideal log-Gabor filter at angle θ, as given in Chapter 2, the $\alpha_k(\theta)$'s will simply be given by the inner product of the ideal filter and each ϕ_k (equation 4.8):

$$\begin{aligned} \alpha_k(\theta) &= \langle F_\theta, \phi_k \rangle \\ &= \langle F_\theta, \sum_h \gamma_k^h O_h \rangle \\ &= \sum_h \gamma_k^h \langle F_\theta, O_h \rangle. \end{aligned} \tag{C.8}$$

Returning now to $\beta_k(\theta)$, we can write:

$$\begin{aligned} \sum_k \beta_k(\theta) O_k &= \sum_k \alpha_k(\theta) \phi_k \\ &= \sum_k \alpha_k(\theta) \sum_h \gamma_k^h O_h \\ &= \sum_h \left(\sum_k \alpha_k(\theta) \gamma_k^h \right) O_h. \end{aligned} \tag{C.9}$$

For equality to hold, the quantity in parentheses in Equation C.9 must equal$\beta_k(\theta)$:

$$\begin{aligned} \beta_k(\theta) &= \sum_m \alpha_m(\theta) \gamma_m^k \\ &= \sum_m \sum_j \gamma_m^j \langle F_\theta, O_j \rangle \gamma_m^k \\ &= \sum_j \langle F_\theta, O_j \rangle \sum_m \gamma_m^j \gamma_m^k. \end{aligned} \tag{C.10}$$

We can simplify the expression for β_k further by observing that the summation over m is the dot product of the jth and kth columns of γ:

$$\beta_k(\theta) = \sum_j \langle F_\theta, O_j \rangle (\gamma^j \cdot \gamma^k) \tag{C.11}$$

C.2 Checking on the Equivalence Between the Two Approaches

We start with the following equations for $\beta_k(\theta)$

$$\sum_k \beta_k(\theta) \langle O_z, O_k \rangle = \langle F_\theta, O_z \rangle \qquad k = 1, .., N, z = 1, .., N.$$

Let $a_{ij} = \langle O_i, O_j \rangle$ and $d_i(\theta) = \langle F_\theta, O_i \rangle$. Then we can rewrite the above eqn. as

$$\begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{bmatrix} \begin{bmatrix} \beta_1(\theta) \\ \beta_2(\theta) \\ \vdots \\ \beta_n(\theta) \end{bmatrix} = \begin{bmatrix} d_1(\theta) \\ d_2(\theta) \\ \vdots \\ d_n(\theta) \end{bmatrix}.$$

For $n = 2$ we have

$$\begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \begin{bmatrix} \beta_1(\theta) \\ \beta_2(\theta) \end{bmatrix} = \begin{bmatrix} d_1(\theta) \\ d_2(\theta) \end{bmatrix}.$$

For $a_{11}a_{22} \neq a_{12}a_{21}$ we get

$$\beta_1 = \frac{d_1 a_{22} - a_{12} d_2}{a_{11} a_{22} - a_{12} a_{21}} \tag{C.12}$$

$$\beta_2 = \frac{d_2 a_{11} - a_{21} d_1}{a_{11} a_{22} - a_{12} a_{21}}. \tag{C.13}$$

This is the solution for β as given by Section 4.2.

We shall now use the Gram-Shmidt approach as above to formulate the problem in terms of the orthogonal functions φ, orthonormal function ϕ_k and their steering functions $\alpha(\theta)$.

$$\varphi_1 = O_1 \tag{C.14}$$

$$\begin{aligned} \varphi_2 &= O_2 - \frac{\langle O_2, \varphi_1 \rangle}{\langle \varphi_1, \varphi_1 \rangle} . \varphi_1 \\ &= O_2 - \frac{\langle O_2, O_1 \rangle}{\langle O_1, O_1 \rangle} . O_1 \\ &= O_2 - \frac{a_{21}}{a_{11}} . O_1. \end{aligned} \tag{C.15}$$

Normalizing these functions we get (using $a_{12} = a_{21}$)

$$\phi_1 = \frac{O_1}{\sqrt{a_{11}}} \tag{C.16}$$

$$\phi_2 = \frac{O_2 - \frac{a_{21}}{a_{11}}.O_1}{\sqrt{a_{22} - \frac{a_{12}.a_{21}}{a_{11}}}}. \tag{C.17}$$

As in Eqn. C.3), let $\phi_i = \sum_j \gamma_{ij} O_j$

$$\Rightarrow \gamma_{ij} = \begin{bmatrix} 1/a_{11} & 0 \\ \frac{-a_{21}/a_{11}}{\sqrt{a_{22}-\frac{a_{12}a_{21}}{a_{11}}}} & \frac{1}{\sqrt{a_{22}-\frac{a_{12}a_{21}}{a_{11}}}} \end{bmatrix}. \tag{C.18}$$

Now we can solve for the steering coefficients of ϕ_i using Eqn. C.8 to evaluate the $\alpha_i(\theta)$

Eq C.8 $\alpha_i = \langle F_\theta, \phi_i \rangle$

$$\Rightarrow \alpha_1(\theta) = d_1/\sqrt{a_{11}} \tag{C.19}$$

$$\Rightarrow \alpha_2(\theta) = \frac{d_2 - \frac{a_{21}}{a_{11}} d_1}{\sqrt{a_{22} - \frac{a_{12}a_{21}}{a_{11}}}}. \tag{C.20}$$

The $\beta_i(\theta)$ as given above are solutions of Eqn. C.10, which for $N = 2$ is

$$[\beta_1, \beta_2] = [\alpha_1, \alpha_2] \begin{bmatrix} \gamma_{11}, \gamma_{12} \\ \gamma_{21}, \gamma_{22} \end{bmatrix}. \tag{C.21}$$

Using Eqn. C.20 we get (where β_i and d_i are functions of θ and $a_{ij} = a_{ji}$)

$$\begin{aligned} \beta_1 &= \frac{d_1}{a_{11}} - (\frac{a_{21}}{a_{11}}(d_2 - \frac{a_{21}}{a_{11}} d_1))/(a_{11}a_{22} - a_{12}a_{21}) \\ &= \frac{d_1 a_{22} - d_2 a_{21}}{a_{11}a_{22} - a_{12}a_{21}} \\ &= \frac{d_1 a_{22} - d_2 a_{12}}{a_{11}a_{22} - a_{12}a_{21}} \end{aligned} \tag{C.22}$$

$$\begin{aligned} \beta_2 &= (d_2 - a_{21}d_1/a_{11})/(a_{22} - a_{12}a_{21}/a_{11}) \\ &= \frac{d_2 a_{11} - d_1 a_{21}}{a_{11}a_{22} - a_{12}a_{21}}. \end{aligned} \tag{C.23}$$

Hence the $\beta_i(\theta)$ as derived above (Eqns. C.22) are identical to those derived in Section 4.2. (Eqn. C.12).

References

[AB85] E. Adelson and J. Bergen, "Spatiotemporal energy models for the perception of motion," *Journal of the Optical Society of America A*, 2(2):284–299, 1985.

[ACBG90] M. Clark A. C. Bovik and W. S. Geisler, "Multichannel texture analysis using localized spatial filters," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12:55–73, 1990.

[And87] C. H. Anderson, "A filter-subtract-decimate hierarchical pyramid signal analyzing and synthesizing technique," United States Patent 718,104, 1987.

[BA83] P. J. Burt and E. A. Adelson, "The Laplacian pyramid as a compact image code," *IEEE Transactions on Communications*, 31:532–540, 1983.

[Bro66] P. Brodatz, *Textures*, Dover, New York, 1966.

[Can86] J. Canny, "A computational approach to edge detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 8:679–698, 1986.

[Car92] J. M. Carstensen, *Description and Simulation of Visual texture*, PhD thesis, The Technical University of Denmark, Lyngby, Denmark, 1992.

[CC85] R. Chellappa and S. Chatterjee, "Classification of textures using Gaussian Markov Random Fields," *IEEE Transactions on Acoustics, Speech and Signal Procesing*, 33(4):959–963, 1985.

[CFP91] F. S. Cohen, Z. Fan, and M. A. Patel, "Classification of rotated and scaled textured images using Gaussian Markov Random Field models," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(2):192–202, 1991.

[CH80] R. W. Conners and C. A. Harlow, "A theoretical comparison of texture algorithms," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2:204–222, 1980.

[CJ83] R. Cross and A. Jain, "Markov random field texture models," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 5:25–39, 1983.

[CJ85] J. Coggins and A. Jain, "A spatial filtering approach to texture analysis," *Pattern Recognition Letters*, pages 195–203, 1985.

[CK93] T. Chang and C. C. J. Kuo, "Texture analysis and classification with tree-structured wavelet transform," *IEEE Transactions on Image Processing*, 2(4):429–441, 1993.

[CR68] F. W. Campbell and J. G. Robson, "Application of Fourier analysis to the visibility of gratings," *Journal of Physiology*, 197:551–566, 1968.

[Das91] B. V. Dasarathy, editor, *Nearest neighbor pattern classification techniques*, IEEE Computer Society Press, Los Alamitos CA, 1991.

[DAT82] R. L. Devalois, D. G. Albrecht, and L. G. Thorell, "Spatial-frequency selectivity of cells in macaque visual cortex," *Vision Research*, 22:545–559, 1982.

[Dau85] J. G. Daugman, "Uncertainty relation for resolution, spatial frequency, and orientation optimized by 2d visual cortical filters," *Journal of the Optical Society of America A*, 2:1160–1169, 1985.

[DH73] R. O. Duda and P. E. Hart, *Pattern Classification and Scene Analysis*, John Wiley and Sons Inc., 1973.

[DJA79] L. S. Davis, S. Johns, and J. K. Aggarwal, "Texture analysis using generalized co-occurance matrices," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1:251–259, 1979.

[FA91] W. T. Freeman and E. H. Adelson, "The design and use of steerable filters," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(9):891–906, 1991.

[Fie87] D. H. Field, "Relations between the statistics of natural images and the response properties of cortical cells," *Journal of the Optical Society of America A*, 4:2379–2394, 1987.

[FOTK92] M. Fukumi, S. Omatu, F. Takeda, and T. Kosaka, "Rotation-invariant neural pattern recognition system with application to coin recognition," *IEEE Transactions on Neural Networks*, 3(2):272–279, 1992.

[GA] H. Greenspan and C. H. Anderson, "Image enhancement by nonlinear extrapolation in frequency space," United States Patent, submitted 1994.

[GA94] H. Greenspan and C. H. Anderson, "Image enhancement by nonlinear extrapolation in frequency space." In *Proceedings of SPIE on Image and Video Processing II*, volume 2182, pages 2–13, 1994.

[Gal74] M. Galloway, "Texture analysis using gray-level run lengths," *Computer Graphics and Image processing*, 4:172–179, 1974.

[GBP+94] H. Greenspan, S. Belongie, P. Perona, R. Goodman, S. Rakshit, and C. H. Anderson, "Overcomplete steerable pyramid filters and rotation invariance," In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, June 1994.

[GBPG94] H. Greenspan, S. Belongie, P. Perona, and R. Goodman, "Rotation invariant texture recognition using a steerable pyramid," In *Proceedings of the 12th International Conference on Pattern Recognition*, 1994, to appear.

[GG93] H. Greenspan and R. Goodman, "Remote sensing image analysis via a texture classification neural network," In *Advances in Neural Information Processing Systems 5*, San Mateo, CA, 1993. Morgan Kaufmann.

[GGC92] H. Greenspan, R. Goodman, and R. Chellappa, "Combined neural network and rule-based framework for probabilistic pattern recognition and discovery," In *Advances in Neural Information Processing Systems 4*, San Mateo, CA, 1992. Morgan Kaufmann.

[GGCA94] H. Greenspan, R. Goodman, R. Chellappa, and C. Anderson, "Learning texture discrimination rules in a multiresolution system," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, July 1994.

[GHMS92] R. M. Goodman, C. Higgins, J. Miller, and P. Smyth, "Rule-based networks for classification and probability estimation," *Neural Computation*, 4:781–804, 1992.

[GL94] H. Greenspan and M. C. Lee, "Combining image enhancement with pyramid coding," Technical report, Jet Propulsion Laboratory, 1994, in preparation.

[GS89] R. Goodman and P. Smyth, "The induction of probabilistic rule-sets, the ITRULE algorithm," In *Proceedings of the 1989 International Workshop on Machine Learning*, pages 129–132, San Mateo, CA, 1989. Morgan Kaufmann.

[Har79] R. Haralick, "Statistical and structural approaches to texture," *Proceedings of the IEEE*, 67(5):610–621, 1979.

[Haw69] J. K. Hawkins, "Textural properties for pattern recognition," In *Picture Processing and Psychopictorics*. Academic Press, 1969.

[Hee87] D. Heeger, "Optical flow from spatiotemporal filters," In *Proceedings of the First International Conference on Computer Vision*, pages 181–190, 1987.

[HO88] D. R. Hougen and S. M. Omohundro, "Fast texture recognition using information trees," Technical Report UIUCDCS-R-88-1409, University of Urbana Champaign, 1988.

[JF91] A. Jain and F. Farrokhnia, "Unsupervised texture segmentation using gabor filters," *Pattern recognition*, 24(12):1167–1186, 1991.

[KCK82] R. L. Kashyap, R. Chellappa, and A. Khotanzad, "Texture classification using features derived from random field models," *Pattern Recognition Letters*, 1:43–50, 1982.

[KG83] H. E. Knutsson and G. H. Granlund, "Texture analysis using two-dimensional quadrature filters," In *Proceedings of the IEEE Workshop on Computer Architectures and Pattern Analysis and Image Database Management*, Pasadena CA, 1983.

[KK86] R. L. Kashyap and A. Khotanzad, "A model-based method for rotation invariant texture classification," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 8(4):472–481, 1986.

[Law80] K. I. Laws, *Textural image segmentation*, PhD thesis, University of Southern California, 1980.

[Lee94] M. C. Lee, *Still and moving image compression schemes using multirescale techniques*, PhD thesis, California Institute of Technology, 1994.

[LF79] S. Y. Lu and S. Fu, "Stochastic tree grammar inference for texture synthesis and discrimination," *Computer Graphics and Image Processing*, 9:234–245, 1979.

[LF93] A. Lain and J. Fan, "Texture classification by wavelet packet signatures," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15(11):1186–1191, 1993.

[LP91] J. Lee and W. Philpot, "Spectral texture pattern matching: A classifier for digital imagery," *IEEE Transactions on Geoscience and Remote Sensing*, GE-29(4):545–554, 1991.

[Mal89] S. G. Mallat, "A theory for multiresolution signal decomposition: The wavelet representation," *IEEE Transactions on Pattern Analysis and Machine Intelligenece*, 11:674–693, 1989.

[Mar82] D. Marr, *Vision*, W. H. Freeman and Co., San Francisco, 1982.

[MC93] B. S. Manjunath and R. Chellappa, "A unified approach to boundary perception: Edges, textures and illusory contours," *IEEE Transactions on Neural Networks*, 1993.

[MH80] D. C. Marr and E. C. Hildreth, "Theory of edge detection," In *Proceedings of the Royal Society of London B*, volume 207, pages 187–217, 1980.

[MLLY91] S. K. Mitra, H. Li, I . Lin, and T. Yu, "A new class of nonlinear filters for image enhancement," In *Proceedings of ICASSP M5.1*, pages 2525–2528, 1991.

[MP90] J. Malik and P. Perona, "Preattentive texture discrimination with early vision mechanisms," *Journal of the Optical Society of America A*, 7:923–932, 1990.

[Per91] P. Perona, "Deformable kernels for early vision," In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 222–227, June 1991.

[PL92] S. J. Perantonis and P. J. G. Lisboa, "Translation, rotation, and scale invariant pattern recognition by high-order neural networks and moment classifiers," *IEEE Transactions on Neural Networks*, 3(2):241–251, 1992.

[PM90] P. Perona and J. Malik, "Detecting and localizing edges composed of steps, peaks and roofs," In *Proceedings of the Third International Conference of Computer Vision*, pages 52–57, Osaka, 1990. IEEE Computer Society.

[PZ88] M. Porat and Y. Y. Zeevi, "The generalized gabor scheme of image representation in biological and machine vision," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 10:452–468, 1988.

[RHW86] D. Rumelhart, G. Hinton, and R. Williams, "Learning internal representations be error propagation," In D. Rumelhart and J. McClelland, editors, *Parallel Distributed Processing*, chapter 8. MIT Press, Cambridge MA, 1986.

[SFAH92] E. P. Simoncelli, W. T. Freeman, E. H. Adelson, and D. J. Heeger, "Shiftable multi-scale transforms," *IEEE Transactions on Infromation Theory*, 1992.

[SG92] P. Smyth and R. Goodman, "An information theoretic approach to rule induction from databases," *IEEE Transactions on Knowledge and Data Engineering*, 4(4):301–316, 1992.

[Skl78] J. Sklansky, "Image segmentation and feature extraction," *IEEE Transactions on Systems, Man, and Cybernetics*, 8:237–247, 1978.

[TJ92] M. Tuceryan and A. K. Jain, *The Handbook of Pattern Recognition and Computer Vision*, chapter 11, World Scientific Publishing Company, 1992.

[TSJ91] J. Ton, J. Sticklen, and A. Jain, "Knowledge-based segmentation of landsat images," *IEEE Transactions on Geoscience and Remote Sensing*, GE-29(2):222–232, 1991.

[Tur86] M. R. Turner, "Texture discrimination by gabor functions," *Biological Cybernetics*, 55:71–82, 1986.

[Uns86] M. Unser, "Sum and difference histograms for texture classification," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 8(1):118–125, 1986.

[van91] G. S. vanderWal, "The sarnoff pyramid chip," In *Proceedings of Computer Architecture for Machine Perception*, volume CAMP91, December 1991.

[VNP86] F. M. Vilnrotter, R. Nevatia, and K. E. Price, "Structural analysis of natural textures," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 8:76–89, 1986.

[Web83] A. G. Weber, "Image database," Technical Report 101, University of Southern California, Image Processing Institute, 1983.

[Wha87] S. Wharton, "A spectral-knowledge-based approach for urban land-cover discrimination," *IEEE Transactions on Geoscience and Remote Sensing*, GE-25(3):272–282, 1987.

[Wit83] A. Witkin, "Scale-space filtering," In *Proceedings of IJCAI, Karlsruhe, West Germany*, pages 1019–1021, 1983.

[Woo91] J. W. Woods, *Subband Image Coding*, Kluwer Academic Publishers, 1991.

[WWB88] B. Widrow, R. Winter, and R. Baxter, "Layered neural nets for pattern recognition," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 36(7):1109–1118, 1988.

[YC93] J. You and H. A. Cohen, "Classification and segmentation of rotated and scaled textured images using texture 'tuned' masks," *Pattern Recognition*, 26(2):245–258, 1993.

[YP83a] A. L. Yuille and T. Poggio, "Fingerprint theorems for zero crossings," Technical Report 730, MIT AI Memo, 1983.

[YP83b] A. L. Yuille and T. Poggio, "Scaling theorems for zero-crossings," Technical Report 722, MIT AI Memo, 1983.

[ZK81] S. W. Zucker and K. Kant, "Multiple-level representations for texture discrimination," In *Proceedings of the Conference on Pattern recognition and Image Processing*, pages 609–614, Dallas, Texas, 1981.