Terms and Conditions of Use of Digitised Theses from Trinity College Library Dublin

Copyright statement

All material supplied by Trinity College Library is protected by copyright (under the Copyright and Related Rights Act, 2000 as amended) and other relevant Intellectual Property Rights. By accessing and using a Digitised Thesis from Trinity College Library you acknowledge that all Intellectual Property Rights in any Works supplied are the sole and exclusive property of the copyright and/or other IPR holder. Specific copyright holders may not be explicitly identified. Use of materials from other sources within a thesis should not be construed as a claim over them.

A non-exclusive, non-transferable licence is hereby granted to those using or reproducing, in whole or in part, the material for valid purposes, providing the copyright owners are acknowledged using the normal conventions. Where specific permission to use material is required, this is identified and such permission must be sought from the copyright holder or agency cited.

Liability statement

By using a Digitised Thesis, I accept that Trinity College Dublin bears no legal responsibility for the accuracy, legality or comprehensiveness of materials contained within the thesis, and that Trinity College Dublin accepts no liability for indirect, consequential, or incidental, damages or losses arising from use of the thesis for whatever reason. Information located in a thesis may be subject to specific use constraints, details of which may not be explicitly described. It is the responsibility of potential and actual users to be aware of such constraints and to abide by them. By making use of material from a digitised thesis, you accept these copyright and disclaimer provisions. Where it is brought to the attention of Trinity College Library that there may be a breach of copyright or other restraint, it is the policy to withdraw or take down access to a thesis while the issue is being resolved.

Access Agreement

By using a Digitised Thesis from Trinity College Library you are bound by the following Terms & Conditions. Please read them carefully.

I have read and I understand the following statement: All material supplied via a Digitised Thesis from Trinity College Library is protected by copyright and other intellectual property rights, and duplication or sale of all or part of any of a thesis is not permitted, except that material may be duplicated by you for your research use or for educational purposes in electronic or print form providing the copyright owners are acknowledged using the normal conventions. You must obtain permission for any other use. Electronic or print copies may not be offered, whether for sale or otherwise to anyone. This copy has been supplied on the understanding that it is copyright material and that no quotation from the thesis may be published without proper acknowledgement.

Hybrid Feature-Based Transform Coding of Grey-Level Images

Ivan Fox

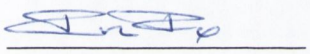
A thesis submitted for the degree of
Doctor of Philosophy in Computer Science
University of Dublin, Trinity College
Department of Computer Science

31st October 2000

Declaration

I declare that the work described in this thesis has not been submitted for a degree at any other university, and that the work is entirely my own.

Signature,



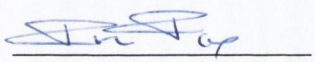
Ivan Fox,

31st October 2000

Permission to lend and/or copy

I agree that the library in Trinity College may lend or copy this thesis upon request.

Signature,



Ivan Fox,

31st October 2000

Acknowledgements

I would like to extend my most sincere gratitude to the many people who helped and supported me in the completion of this thesis. My supervisor Prof. John Byrne for his generous support in enabling me to pursue this research. Prof. David Vernon for enlightening me as to the value of philosophical awareness in the pursuit of science. Dr. Gerard Lacey for his guidance and advice. Damian Gordon for his generous help in all manner of things. The members of the Computer Vision and Robotics Group for their encouragement, and my (ex)office-mates Gloria and Abdo. Dr. Nikla Duffy for all her support. I would also like to thank my family and friends for all their good humoured tolerance. Final thanks goes to the examiners for agreeing to read this thesis.

Abstract

The objective of image compression is to achieve the minimum possible bit-rate, within certain bounds of computational complexity, and with a tolerable degree of distortion. In *very low bit-rate coding* it is generally accepted that the distortions introduced will be *visible* to some degree. The most widely used image coding techniques today (JPEG,MPEG-2,H.263), are based on *block transform coding* with the *discrete cosine transform* (DCT). At low bit-rates however, DCT based methods tend to introduce a number of particularity annoying forms of distortion, namely *blocking effects, ringing noise* and *blurring*. This has been the motivation behind the development of *second generation* image coding methods. Second generation coding methods adopt source models based on the human visual system (HVS), as opposed the statistical *Markov-1* model on which the DCT is based. By doing so, they strive to better encapsulate the *visually important* aspects of an image in a more compact form. This approach is typified by *sketch based coding* and *segmented image coding*, collectively referred to as *feature based coding* (FBC) methods.

One of the principal drawbacks of second generation methods in general however, is their computational complexity and strong signal dependence (i.e. they tend to work well with some images but not with others). In this thesis we describe a novel hybrid coding scheme that seeks to combine the computational efficiency and robustness of transform coding, with the subjective benefits of FBC. This is achieved by combining a block based approach to texture coding with a feature based approach to contour coding. Essentially, an image is reconstructed by the proposed scheme, using two pieces of information: a subsampled version of the original image, which we call the *smooth-component*, and the geometrical contour information extracted at the encoder. We call this the *contour-component*. The contour-component may be thought to provide the spatial bounds for an interpolation procedure on the smooth-component. The proposed scheme is evaluated and compared with JPEG. We demonstrate superior low bit-rate performance to JPEG, in terms of both PSNR and subjective image quality.

Acronyms

2-D	Two Dimensional
3-D	Three Dimensional
BTC[1]	Block Transform Coding
BPP	Bits per pixel
BPCC	Bits per Contour Point
CR	Compression Ratio[†]
DCT	Discrete Cosine Transform
DFT	Discrete Fourier Transform
DMS	Discrete Memoryless source
DPCM	Differential Pulse Code Modulation
DVD	Digital Versatile Disk or Digital Video Disk
DWT	Discrete Wavelet Transform
FBC	Feature Based Coding
FBTC	Feature Based Transform Coding
HVS	Human Visual System
IDCT	Inverse Discrete Cosine Transform
KLT	Karhunen-Loève Transform
LGO	Laplacian-Gaussian Operator
MSE	Mean Square Error
PCM	Pulse Code Modulation
PSNR	Peek Signal to Noise Ratio
SBC	Subband Coding
SIC	Segmented Image Coding
VLC	Variable Length Coding
VQ	Vector Quantization
JPEG	Joint Photographic Experts Group
MPEG	Moving Picture Experts Group

[1] We use the acronym BTC here to refer to *Block Transform Coding* and not *Block Truncation Coding* [19] as it is also used.

Contents

Detailed Chapter Contents

Chapter 1

INTRODUCTION TO THESIS

1.1 Context

The field of image coding has been an active area of research since as early as 1920, when the Bartlane cable picture transmission system was developed to enable more rapid transmission of images via transatlantic cables [15]. Since then, the internet has spread its web across the world, and the bandwidth of public telecommunication systems has increased dramatically. Despite these improvements however, the same concerns that motivated the development of the Bartlane system remain present today, namely, the relatively large *cost per bit* of uncompressed images. The field of image coding has grown to address this problem.

In recent years, renewed stimulus for the development of more efficient image compression technologies has come from the success of the internet as a medium for information delivery, and the emergence of many new consumer oriented audio-visual technologies. Within the last few years video delivery systems have begun the conversion from analog to digital, with videotape being replaced by digital video disk (DVD), and analog broadcasts with digital television. Digital alternatives now exist to traditional film based still cameras, camcorders and even movie cameras. Digital images are estimated to account for as much as 90% of the traffic on the internet [42 p3] and digital video is delivered over the internet in both pre-recorded and live form. Inexpensive *webcam* devices

are now available that enable affordable PC based video-conferencing, while mobile video-phone systems for wireless digital telecommunications networks are under development[2].

All of these digital systems either store or transmit images or video at some stage. Digital images and video, in their *raw* or uncompressed format, have memory requirements that would render many of these applications infeasible were compressed representations unavailable. Compression is thus an enabling technology, whose goal is to represent an image or image sequence with as few bits as possible. In pursuit of this goal however, real-world concerns must be addressed, and solutions developed that lie within certain bounds of complexity[3].

1.2 Approaches to Image Compression

In this thesis we shall be concerned primarily with compression techniques for *very low bit-rate* coding (>20:1 compression ratio). Numerous approaches have been taken to address the problem of image and video coding within this bit-rate, but some broad classes predominate in the literature. If methods are classified in terms of how they model the source image, then two general classes emerge. Methods that model the source as a stochastic† process are often described as *waveform coders,* while those that adopt higher level models are known as *model-based coders.* In the following paragraphs we provide a very brief overview of the predominant approaches to image compression. Two specific approaches are elaborated in chapters three and four; *transform coding* (closely related to subband coding) and *feature based coding* (a second generation approach) respectively.

Waveform coding methods are currently dominated by a group of techniques that may be described as *multiresolution methods.* These include techniques such as *transform coding* [20][22][27][32], *subband coding* [8][30][43][65][66][67], *pyramid coding* [70], *hierarchal coding* [11], and *wavelet coding* [41][42][43][45](see chapter 3). These methods all share the common approach of decomposing the source image into different spatial frequency bands (levels of spatial resolution/detail), and coding each band according to their individual statistics.

[2] At time of writing, high bandwidth wireless digital telecommunications networks are being tested in the United Kingdom but sufficiently compact mobile devices for video-conferencing have not been made public.

[3] In this thesis we use the term *complexity*, as it commonly used in the image coding literature, to refer to memory and computing requirements.

Another approach to image coding, that could be described as purely statistical, is known as *vector quantization* (VQ)[13] (chapter 2). In VQ an image or transform thereof is divided into smaller collections known as vectors and each vector tested for similarity with members from a predefined codebook. The index into this codebook is then coded rather that the original vector itself. The concept of VQ is extended in *fractal coding* [14]. In fractal coding the codebook is generated and transmitted for each new image. In addition to encoding codebook indices however, instructions for the transformation (e.g. scaling, translation, rotation, shear) of the codebook members are also transmitted.

In recent years, a collection of methods that may be loosely described under the heading of *model based coding* [11] have emerged. Methods in this category share commonality in modelling some "higher level" aspects of the signal *content.* Model based approaches divide into two broad classes, those that employ 2-D or 3-D structural models of the source content, (e.g. human face models [11]), and *second generation methods* [4] [80][103][119] that model *features* in the image signal, such as contours and textures. Model based approaches often achieve superior compression, when applied to specific types of images, but tend to fail when the source data cannot be fitted to their model. In the next section we provide a brief overview of the solution investigated in this thesis.

1.3 Investigated Approach: Overview of Solution

The most widely employed image compression technique is the block based discrete cosine transform (DCT)[20] in which source images are modelled as stochastic processes (see §3.4). The DCT however, is generally regarded to have significant failings for very low bit-rate coding. In particular, the *types* of distortion introduced at very low bit-rates are perceptually unfavourable (see §3.2.7). *Second generation* techniques [85] have been developed for very low bit rate coding, that strive to better model human visual perception in the encoding process (see chapter 4). By doing so, it is hoped that the distortions introduced will be more favourable to the human visual system (HVS). One of the principal drawbacks of second generation methods in general however, is their computational complexity and strong signal dependence (i.e. they tend to work well with some images but not with others).

[4] The introduction of the phrase "second generation" to describe these approaches, implies that the earlier waveform based methods may be described as "first generation" approaches.

In this thesis we describe a novel hybrid coding scheme that seeks to combine the computational efficiency and robustness of transform coding, with the subjective benefits of a second generation technique known as *feature based coding*. This is achieved by combining a block based approach to texture coding with a feature based approach to contour coding. A three component image model has been defined that represents images in terms of contours and textures. Textures are further classified as either *smooth texture* or *highly textured*, which we simply refer to as *smooth* or *texture* respectively. The three components collectively represent a *simplified* view of an image. This model is used as the basis for the proposed solution (see Fig. 1.1).

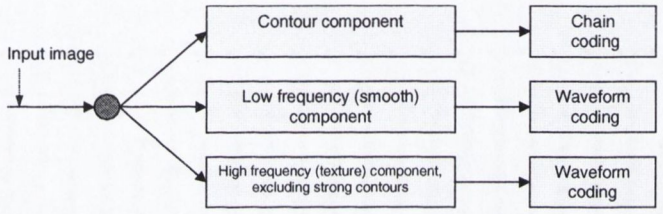


Fig. 1.1 Illustration of image representation using the three component model.

In the proposed image model, we use the assumption that for *simple* images[5], it is sufficient to know only the *locations* of contours, and the nature of the intensity distributions in their *general vicinity,* to reconstruct a visually meaningful approximation of significant edges. We stipulate therefore, that for such images, there are two *types* of information present:; a high frequency component that *locates* contours, and a low frequency component that *fills in* between contours. These two *features* are represented by the *contour-component* and the *smooth-component* respectively. The smooth component is encoded as an *edge sensitive,* subsampled version of the original image, and the contour-component is represented in the spatial domain as a set of chain-coded pixels (see Fig. 1.2).

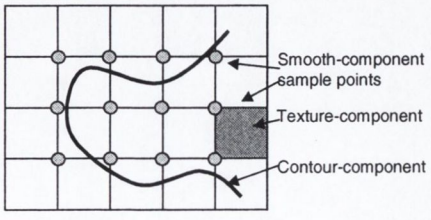
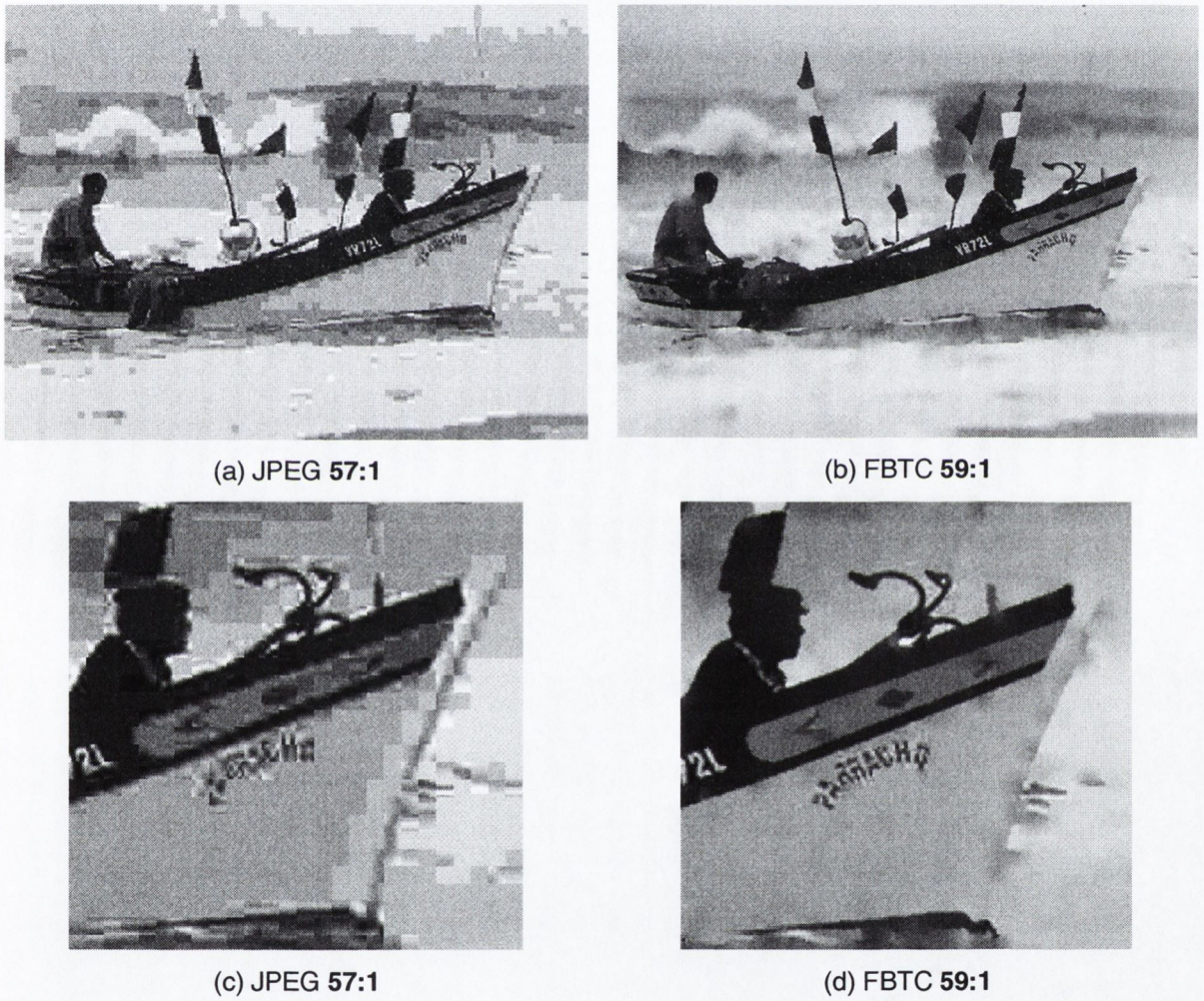


Fig. 1.2 Illustration of smooth, texture and contour components.

[5] i.e. images that are dominated mostly by large smoothly textured objects

The last component of the model is the *texture-component*. The texture-component is used to selectively encode image regions that contain spatially compact edge features. For such regions, we cannot reasonably predict the intensity profiles from the smooth-component and contour-components alone. Fig. 1.3 presents some sample results of the proposed scheme in comparison with JPEG.



(a) JPEG **57:1** (b) FBTC **59:1**

(c) JPEG **57:1** (d) FBTC **59:1**

Fig. 1.3 Example of JPEG compressed image (a) and image compressed using the proposed scheme (b). A Zoom of (a) is presented in (c) and zoom of (b) is displayed in (d).

1.4 Main contributions

The original contributions made in this thesis may be summarized as follows:

1. The formulation and evaluation of a novel three component image model for very low bit-rate image coding.

2. Development of three different codec† architectures to evaluate the proposed image model.

3. Development of a novel approach for the amalgamation of block based transform coding and feature based coding.

4. Design and implementation of a perceptually motivated texture extraction method based on efficient morphological operators. Three types of edge features may be efficiently classified and spatially segmented: strong edges, busy-edges, and texture edges.

5. Development of novel technique for *edge sensitive* image subsampling, with a method for *relocating* sample points that intersect edges.

6. Development of a novel method for the interpolation of image data in the regions about strong edges.

All of the above contributions have been investigated experimentally and documented in this thesis.

1.5 Organisation of the Thesis

The organisation of this thesis is as follows. Chapter two provides a brief introduction to the fundamental concepts in image coding. The basic principles of image compression are introduced, two classes of coder model are presented; waveform and model-based, and the fundamental information theoretic concepts are presented.

Chapter three discusses the topic of transform coding. The objective of this chapter is to provide an appreciation for the means by which compression is achieved through transform coding, and to outline the principal strengths and weakness of the approach for very low bit-rate coding. As compression is only achieved in transform coding by post-transformation techniques, we also discuss the concepts of quantization, bit-allocation, and zonal and threshold coding. The principal forms of distortion in transform coding, and techniques to address them are also described. We present the optimal but impractical Karhunen-Loève transform (KLT) from which we then derive the more practical discrete

cosine transform (DCT). The properties of both these transforms are discussed. Finally, we introduce subband coding and relate it to the DCT.

Chapter four discusses the more recent field of *feature based coding* (FBC), where image models that more accurately reflect the characteristics of the human visual system (HVS) are used. The motivations for the approach are presented, and the two basic feature types, *contours* and *textures,* are described. We proceed to review the state of the art in *sketch based coding* and briefly describe the principles of *segmented image coding.* This chapter, together with chapter three, provides the context for the proposal of the novel method described in chapter five.

Chapter five provides a detailed description of a hybrid feature-based transform coding method. The principal strengths and weaknesses of transform coding and FBC are reviewed, and the basis for the proposed method established in terms of a complimentary amalgamation of these two schemes. Theoretical and practical issues relating to the proposed scheme are discussed, and three codec architectures presented. Chapter 6 presents the results of the proposed scheme in comparison with JPEG, and evaluates the three codec architectures. Finally chapter 7 draws some conclusions and suggests directions for future work.

Appendix A presents the original test images used in the evaluation of the codec. Appendix B briefly discusses the principle of bilinear and bicubic image interpolation. Appendix C describes the contour coding method known as chain coding. Appendix D presents some common gradient based edge detectors, and appendix E provides a brief review of some matrix concepts made use of in chapter three.

A glossary of terms is provided following the appendices. Terms in the text that have a glossary entry are indicated by a post-fixed superscript cross symbol thus: term†.

Chapter 2

INTRODUCTION TO IMAGE CODING

2.1 Introduction

2.1.1 Context

This chapter introduces the field of image coding and provides some definitions that are made use of elsewhere in this thesis. This chapter is by no means a comprehensive review of image coding techniques. The discussion is limited to those topics that serve as background material to contributions made in this thesis. We therefore focus primarily on subjects relevant to *block based transform coding* and *feature based coding*. Both of these topics are expounded on further in subsequent chapters.

2.1.2 Objectives and Overview

We begin with a brief look at the principal factors that are considered when approaching the problem of image compression. The subsequent section (§2.2) provides a basic description of image digitization, some properties of the digital representation and the principal forms of mathematical notation for image representation. We proceed to introduce the topic of image compression, noting the basic principles that make compression possible, and outlining two general classes of image coding schemes: waveform and model-based. Some common metrics for evaluating the performance of

compression schemes and some relevant information theoretic principles are also presented.

2.2 Approaching the Problem

The operation of image compression is often referred to as *low bit-rate coding*[6] or *bandwidth reduction*. There is no known single coding scheme that works equally well for all image types and under all circumstances. The approach taken will usually depend on a number of different factors, such as the format of the source image (binary, monochrome, colour, etc), the content of the source image (if predictable), the target bit-rate, the amount and type of distortion tolerable, the application (off-line/real-time, scalable), the transmission medium (error-proneness), and available computing power[7]. In general however, there are trade-offs to be made between lower bit-rates on one hand and higher distortion, computation, error-susceptibility and content restrictions on the other.

One of the first considerations when approaching the problem of image coding is often the *source image format*. Most coding schemes are designed to operate on specific input image formats. In particular, the methods used to code binary and monochrome images usually adopt quite different approaches. Another principal consideration is whether distortion of the image data is tolerable or not. Coding schemes that introduce some degree of distortion in the reconstructed image, even visually imperceptible, are classified as *lossy coding* schemes. Schemes that allow for perfect reconstruction of the image data are said to be *information preserving* or *lossless* schemes. Lossy coding schemes generally achieve superior compression and are employed when transmission or storage requirements are the primary concern, or when the image is intended for direct human inspection, i.e. the image will not undergo further processing or analysis. Lossless methods are generally chosen when the preservation of the original pixel values are of paramount importance (e.g. computer vision, medical images). In this thesis we shall be concerned exclusively with lossy coding schemes.

[6] Where context permits, we shall simply use the term *coding* to imply low-bit rate coding.

[7] The need for image compression arises from very practical concerns, many theoretically superior coding schemes are not used in practise due to their computational complexity.

Before proceeding to discuss the problem of digital image compression further, it is perhaps pertinent to briefly discuss the process by which digital images are produced, known as *digitization*. This is the topic the following section (§2.3).

2.3 Image Representation

2.3.1 Image Digitization

An *analog* still image may be represented by a *continuous* two-dimensional intensity function $f_c(x,y)$, where x and y represent spatial coordinates[8], $x \in \Re^{+X}$, $y \in \Re^{+Y}$ and X and Y are the horizontal and vertical dimensions of the image respectively. This representation is termed the *spatial domain* representation. A monochrome (or greyscale) image yields a single intensity (brightness) value for a given set of coordinates while a colour image yields a range of spectral intensities. We shall restrict the remainder of this discussion to natural[9] monochrome images.

While all natural images have their origin in the continuous domain computers may only store and process data of a *discrete* form. A discrete or *digital image* is generated from a continuous image by periodic sampling in the spatial domain and the *quantization* of those samples. We assume that the analog image has been band-limited[10] prior to sampling, and that sampling is performed at or above the Nyquist rate (twice the highest spatial frequency present). Assuming a band-limited signal, the sampled signal is an exact signal representation according to the sampling theorem [1].

Sampling at regular spatial intervals in both the x and y directions yields the discrete function $f_d(x,y)$ where $x \in \mathrm{Z}_M, y \in \mathrm{Z}_N$ and M and N are the number of horizontal and vertical sample points respectively. Image digitization imposes a finite spatial resolution on the image which will determine the degree of discernible detail in the image. Subsequent discussions assume an orthogonal sampling lattice where horizontal and vertical sampling frequencies are equal. The sampled values of a digital image are commonly referred to as picture elements (pels) or *pixels*. The pixels of monochrome images are typically

[8] Points on the image plane are specified with respect to the orthogonal x and y axes where x values increase from left to right and y values increase from top to bottom.

[9] Images acquired by optical devices of the natural world as opposed to computer synthesised images.

[10] i.e. filtered to remove all spatial frequencies greater than half the sampling frequency.

represented using 8-bit values where 0 represents black and 255 represents white. Images represented using this basic *raw* or *uncompressed* data format are sometimes said to be in *canonical form* [38].

2.3.2 Mathematical Notations

Images may be represented using a number of different forms of mathematical notation. The various forms used in this thesis are listed below.

- 2-D discrete function representation $f_d(x,y)$, $x \in \mathrm{Z}_M, y \in \mathrm{Z}_N$, $M \times N$ pixels.
- 1-D Sequence of N pixels $x(n)$, $n \in \mathrm{Z}_N^+$, increasing values of n cause the image to be scanned from left to right and top to bottom
- $m \times n$ matrix representation $\mathbf{X}_{mn}$, row and column indices equate to horizontal and vertical positions within the image.
- Matrix row or column vector representation, $\mathbf{x}_N = [x_0, x_1, ..., x_{N-1}]$ or $\mathbf{x}_N = [x_0, x_1, ..., x_{N-1}]^T$ respectively. Similar to the 1-D sequence, pixels are represented by the elements x_i of $\mathbf{x}$.

2.4 Objective of Image Compression

The goal of image compression is to represent an image with as few bits as possible, with a tolerable degree of *distortion* (§2.7). As we are concerned exclusively with image compression for the benefit of human viewers in this thesis, the human visual system (HVS) will factor largely in our consideration of what is meant by *tolerable* distortion. In this respect, we shall be mindful of the suitability of simple statistical distortion measures in evaluating coder performance (see §2.7).

2.5 Image Compressibility

There are two principal reasons why images of natural scenes are compressible.

1. **Statistical**: all meaningful images exhibit some form of internal structure that manifests itself through statistical dependencies between pixels. This property is referred

to as *spatial redundancy* (also *interpixel*, *interframe* & *geometric redundancy*) and pixels are said to be *correlated*.

2. **Subjective**: the human visual system (HVS) is not equally sensitive to all types of *image features*, certain types of intensity distributions are more perceivable by the HVS than others. Intensity characteristics, or distributions, to which the HVS is less sensitive are said to be *visually redundant* (or *psycho-visually redundant* or *perceptually irrelevant*) and may often be coarsely approximated without introducing noticeable distortions.

Image compression methods therefore seek to remove or minimise two principal forms of data redundancy: *statistical* and *subjective*. In addition to reducing these two forms of redundancy, image compression schemes also generally incorporate strategies to reduce *coding redundancy* that arises when some symbols (§2.6.1) occur more frequently than others.

2.6 Coder Models

An image coding scheme has two principal components: an encoder which takes an image in its raw format and generates a compressed representation, and a decoder which takes a compressed representation and *reconstructs* the original image. A computer program that performs these functions is referred to as a *codec* (compressor/decompressor). Coding schemes may broadly classified, in terms of how they model the source image, as either *model based* or *waveform* coders.

2.6.1 Waveform Coding

Coding schemes that model source images as random signals and exploit their stochastic properties may be described as *waveform coders*. A block diagram of a general waveform coder is presented in Fig. 2.1.

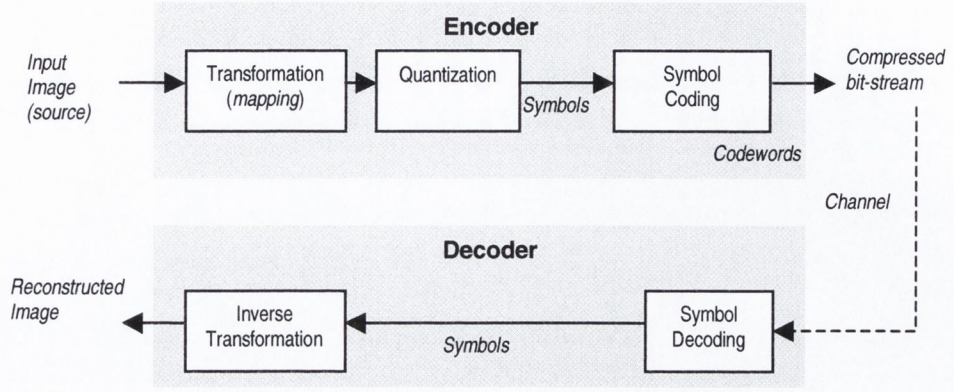


Fig. 2.1 Block diagram of a waveform image coder.

A waveform encoder incorporates three main stages.

1. Transformation (mapping): A one-to-one transform is applied over the source image. This stage is usually lossless and simply exploits statistical redundancies in the source image. This operation is generally reversible and may or may not reduce directly the amount of data required to represent the image. Typical transforms include linear predictive mappings, unitary mappings such as the discrete cosine transform (DCT), and multiresolution mappings such as subband and wavelet transforms.

2. Quantization: The data produced by the transform are mapped to a finite set of *symbols* by the quantizer. This stage is lossy, usually exploiting the psycho-visual properties of the HVS. *Scalar quantizers* perform an element-by-element quantization of the data whereas *vector quantizers* consider a number of elements collectively, generating a single symbol for each collection. The is the first stage that actually *compresses* the image.

3. Symbol Coding: The symbols output by the quantizer are assigned *codewords* by the coder. The symbol coder may employ fixed or variable length codes (VLC), also known as entropy coding (e.g. Huffman codes).

A decoder performs two principal operations: *symbol decoding* and *inverse transformation.* The codewords produced by the encoder are decoded to produce symbols that can then be inverse transformed to reconstruct the original image, albeit with some distortion.

2.6.2 Model Based Coding

Model-based coding schemes form a class of coding approaches that do not model images directly as stochastic processes but rather consider them as having structural features such as contours and regions (Fig. 2.2). Methods in this category vary widely in their approaches, but most can be classified as either 2-D feature based coding, 3-D feature based coding or 3-D model-based coding. Given the diversity of model based coding approaches, further general discussion is beyond the scope of this thesis. In chapter 4 we discuss 2-D feature based coding in more detail.

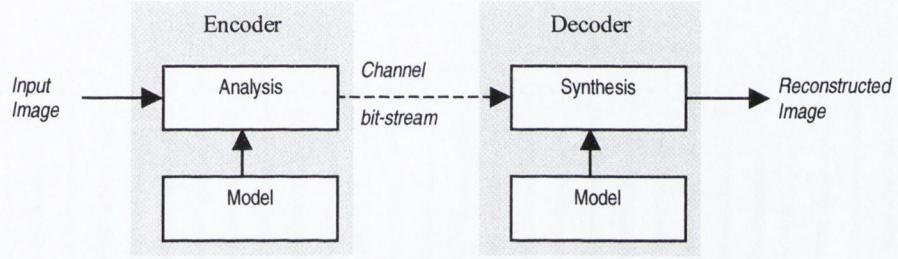


Fig. 2.2 Block diagram model-based coder.

2.7 Bit-rate and Distortion

The performance of an image coder is usually evaluated in terms of two key parameters; *bit-rate* and *distortion*. The bit-rate of a coder refers to the average number of bits required to represent a pixel in the compressed representation, measured in *bits per pixel*. The term distortion refers to the alteration of pixel values between an original and reconstructed image. Lossy compression schemes, by their nature, introduce some form of distortion. A *distortion measure* is a value that quantifies the difference between an original and reconstructed image and thus serves as a form of *quality metric* for comparing compression schemes. A commonly used distortion measure is the *mean square error* (MSE) defined by

$$D = E\left\{\left(X - \hat{X}\right)^2\right\} \tag{2.1}$$

where $E\{\cdot\}$ is the expectation operator, X represents the original image and $\hat{X}$ represents the image reconstructed from the compressed representation. For a discrete number of samples $\{x_i\}$ $i = 0,1,...,N-1$ (pixels) the MSE may be calculated by [42 p71].

$$MSE = \frac{1}{N}\sum_{i=0}^{N-1}(x_i - \hat{x}_i)^2 \tag{2.2}$$

In evaluating practical coding schemes a logarithmic metric known as the *peek signal to noise ratio* (PSNR) is often used [42 p79]. It is defined for N pixels by

$$PSNR = 10\log_{10}\frac{1}{N}\sum_{i=0}^{N-1}\frac{x_{\max}^2}{(x_i - \hat{x}_i)^2} \tag{2.3}$$

where $x_{\max}$ represents the maximum possible value that a pixel may assume[11] and x_i and $\hat{x}_i$ are the original and reconstructed pixel values respectively. Other statistical distortion measures are presented in [20 p209].

While the PSNR is a widely quoted distortion measure for lossy coding schemes, it can only very crudely indicate the *visibility* of any distortion introduced. It does not accurately model the characteristics of the HVS, nor indicate the *type* of distortion (§3.2.7, §3.3.2) present, and hence whether one type of distortion is less favourable than another. Furthermore, a single number is a poor descriptor for a quantity that may occur to varying degrees, in terms of its perceptibility, across the surface of an image.

Ultimately, subjective tests may be required to evaluate the visibility of distortion, particularly when comparing two codecs that employ quite different approaches (e.g. VQ and transform coding). There are however a number of HVS characteristics that are well understood that guide the design and explain the failings of various coding techniques. We discuss these characteristics in chapters 3 and 4, with respect to the particular coding approaches. Some important features of the HVS that should be considered in the design of any codec include [109]:

[11] Generally equal to the value 255.

- The HVS is more sensitive to changes in contrast than to overall illumination. Thus edges play an important role in visual perception. In fact, the HVS amplifies the visibility of edges in an image, giving rise to *Mach Bands* where two different intensity distributions abut. This characteristic of the HVS is responsible for the particular visibility of *blocking effects* in block-based coding (see chapter 3, section 3.2.7).

- The HVS is not equally sensitive to all spatial frequencies. This sensitivity may be expressed by a contrast sensitivity function (CSF) which estimates the visibility of stimuli at different spatial frequencies. Generally speaking the HVS is less sensitive to higher spatial frequencies which may therefore be more coarsely quantized without introducing noticeable distortion (see section 2.9 for a discussion on quantization).

- The HVS performs *masking*. Essentially this means that the HVS is less sensitive to distortion on a spatially complex background. Thus distortion is often said to be masked in highly textured regions or in the immediate vicinity of edges.

- One final feature of the HVS that we remark on here is its ability to detect small departures from the co- linearity of two line segments. This is referred to as *Vernier offset* and necessities the smooth continuity of line/edge features in a reconstructed image if this particularly visible form of distortion is to be avoided (see chapter 3, section 3.2.7).

2.8 Information Theory and Symbol Coding

In this section we address the problem of symbol coding. In the context of image coding, symbols are usually the values output by a quantizer. The mathematical framework of *Information Theory,* first formally presented by Shannon in 1948 [2], defines the framework and the language of source coding and provides the tools for the analysis and evaluation of coding methods.

2.8.1 The Information Content of a Symbol

In the following discussion we consider the problem of coding, with optimal bit-rate efficiency, the symbols generated by a random discrete amplitude source (e.g. a quantizer, §2.6.1). Let X represent a random source that emits vectors $\mathbf{x}_i = (x_1, x_2, ..., x_N)$. Each vector $\mathbf{x}_i \in X$ is called a *symbol* and may assume a value from the alphabet $A = \{a_0, a_1, ..., a_{M-1}\}$, of M letters. We assume the source is *stationary* so that the probability density function[12] (pdf) of the source remains the same regardless of the size of N or what point in time a symbol is emitted. We consider two source models: a *discrete memoryless source* (DMS) and *Markov-K* source.

A source X, is described as a DMS if the symbols produced by X are statistically independent (i.e. uncorrelated). The likelihood of any one letter occurring is determined solely by its probability $p(a_i)$ where

$$\sum_{i=0}^{M-1} p(a_i) = 1 \tag{2.4}$$

If a symbol occurs that is very probable then we say that the information content of the symbol is low. Similarly, if an unlikely symbol occurs then it is said to have a high information content. The concept of information conveyance may be expressed mathematically by

$$I(a_i) = \log_2 \frac{1}{p(a_i)} \tag{2.5}$$

where the base of the logarithm determines the units of measurement (in this case bits). If the probability of a symbol occurring is 1, then the information content in the symbol will be zero. As $p(a_i)$ approaches 0, $I(a_i) \rightarrow \infty$.

[12] The *pdf* of an image defines the probability (likelihood) of any one pixel value occurring.

2.8.2 Entropy

The *entropy* (or *uncertainty*) $H(X)$, of a DMS source X is defined as the average information content per symbol in the source (usually measured in bits per symbol)[15 p326][10 p351],

$$H(X) = \sum_{a \in A} p(a) \log_2 \frac{1}{p(a)} = -\sum_{a \in A} p(a) \log_2 p(a) \tag{2.6}$$

If X is defined in terms of N dimensional vectors[13] then it is often more meaningful to refer to the entropy per source letter defined [11 p15][8 p7]

$$H_N(\mathrm{X}) = \frac{1}{N} H(X)$$

If a source is modelled as a Markov-K signal, where the probability of a symbol occurring is dependent on the values of the preceding K symbols, then we define its entropy in terms of the conditional probabilities $p(x_j = a_i | x_{j-1}, \ldots, x_{j-K})$ $\forall j$, $a_i \in$ A [10 p351].

$$H(\mathrm{X}) = \sum_{S^K} p(x_{j-1}, \ldots, x_{j-K}) H(\mathrm{X} | x_{j-1}, \ldots, x_{J-K}) \tag{2.7}$$

where S^K denotes all possible realisations of $x_{i-1}, \ldots, x_{j-K}$ and

$$H(\mathrm{X} | x_{j-1}, \ldots, x_{j-K}) = \sum_{a \in \mathrm{A}} p(a_i | x_{j-1}, \ldots, x_{j-K}) \log_2 p(a_i | x_{j-1}, \ldots, x_{J-K}) \tag{2.8}$$

A more simple expression may be formulated by arranging the symbol x_i and the preceding K symbols in a vector $\mathbf{x}$. We denote the probability of any given vector $\mathbf{x}_i$ occurring by $p(\mathbf{x}_i)$. The entropy per symbol in $\mathbf{x}$ may be expressed by [2]

$$H_K(\mathrm{X}) = -\frac{1}{K} \sum_{\forall x} p(\mathbf{x}_i) \log_2 p(\mathbf{x}_i) \tag{2.9}$$

[13] i.e. each symbol that X emits is a vector with N components

2.8.3 Rate-distortion Function

Given a distortion measure (e.g. PSNR), it is often desirable to know the minimum possible bit-rate attainable for a given distortion. A function that specifies the minimal attainable bit-rate for a given distortion is known as a *rate-distortion function* [3]. Lossy coding methods exploit the rate-distortion ratio to predict the minimum bit-rate attainable for a given reconstruction quality. Similarly, a *distortion-rate* function specifies the minimal possible distortion for a given rate. If the distortion measure employed incorporates the subjective properties of the HVS, then we may find from the rate-distortion function the minimum bit-rate at which distortion is imperceptible. This bit-rate is known as the *visual entropy*. A large part of information theory is devoted to rate-distortion theory.

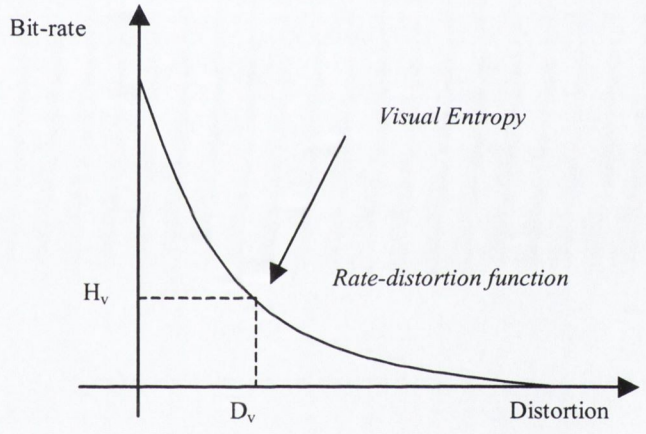


Fig. 2.3 Rate-distortion function of a discrete amplitude source.

2.8.4 Variable Length Coding

In variable length coding (VLC), the number of bits used to represent a letter from the symbol alphabet is dependent on the probability of that letter occurring. The codewords of the alphabet are of variable bit-length and must therefore be uniquely identifiable by virtue of their bit pattern. More frequently occurring (probable) letters are assigned shorter codewords (i.e. less bits used) while less probable letters are represented by longer codewords. In VLC the optimum bit-length for a codeword is proportional to the *information* it conveys. Thus when an improbable symbol occurs it is said to convey more

information than the occurrence of a more probable symbol, and this is reflected in terms of the number of bits used to convey the information.

2.8.4.1 Huffman Coding

Huffman coding [7] is an optimal VLC method for coding symbols individually when the symbol probabilities are powers or two [10 p354] [15 p334] [42 p66]. It is optimal in the sense that no other instantaneously decodable integer-length VLC method can be found to generate a smaller average codeword length [17][10 p354]. Each Huffman code is a variable length, uniquely identifiable, instantaneously decodable *block code*, so called because each source symbol is mapped to a fixed sequence of bits [15 p345]. The codewords produced by a Huffman coder are said to be *instantaneous* because each codeword may be decoded without reference to successive symbols. The average bit-rate produced by a Huffman coder will always be ≥ 1, thus efficiency will be lost if the source entropy is less than 1 bit per symbol [11, p235] (without forming vectors). An adaptation to the standard Huffman coding approach known as *modified Huffman coding*, assigns an "escape" code to a set of less probable symbols [16]. These symbols are in turn assigned codes which are appended to the escape code. The maximum codeword size may thus be limited [11 p235].

2.8.4.2 Arithmetic Coding

Where the source entropy is less that 1 bit per pixel, *arithmetic coding* may be used to overcome the limitations of Huffman coding. Arithmetic coding does not generate block codes, i.e. small bit sequences that may be mapped directly to a symbol, but rather an entire sequence of symbols is mapped to a single codeword. This codeword is a real (floating point) number representing a binary fraction in the interval [0,1]. For a large number of samples, arithmetic coding approaches the entropy of a memoryless source [11 p235] [42 p66]. A disadvantage of arithmetic coding however, is the fact that symbols cannot be instantaneously decoded.

2.8.5 Fixed Length Coding

In fixed length coding, an equal number of bits are assigned to represent each symbol. Fixed length coding is optimal when the probabilities of all symbols are equal, i.e. the probability distribution of the source is flat, and the number of symbols is equal to a power of two. The entropy of the source will then be equal to the length of the codewords.

2.8.6 Run-Length Coding

Both Huffman and arithmetic coding are entropy based techniques, no account is taken of *where* symbols are likely to occur in a sequence. If the source is likely to emit long sequences of repeated symbols however, *run-length* coding may outperform entropy based methods. A run-length code consists of an integer tuple (*run*, *length*), meaning the value *length* is repeated *run* times. A variation on the approach, known as *run-level* coding may be used when only a specific symbol is likely to be frequently repeated, e.g. zero. In this case it may be advantageous to code (*run*, *level*) tuples where *run* represents the repetition of the expected value and *level* represents the value following it. Compression efficiency may be further improved by variable length entropy coding of the run-length codewords [8 p201].

2.9 Quantization Strategies

Quantization is the process of representing a set of continuous-valued samples with a finite number of states. A quantizer thus maps a large number of input levels into a smaller number of output levels. Quantization may be performed, for example, directly on the pixels of an image, on the coefficients of a transform applied to an image, or the parameters of a image model. Due to the differing natures of the various data types that may be quantized in an image coder, this discussion assumes a general nature.

Quantizers may be categorised as either *scalar* or *vector* quantizers, *uniform* or *non-uniform*, *symmetric* or *non-symmetric*, and *memoryless* or with memory. The three latter properties are described briefly below while sections 2.9.1 and 2.9.2 treat scalar and vector quantization respectively.

- A *uniform quantizer* is so called because the source domain is partitioned into intervals of equal step-size (characteristic step function). See sections 2.9.1.1 and 2.9.1.2 and Fig. 2.6 below.
- A *symmetric quantizer* is symmetric about the origin i.e. $Q(-x) = -Q(x)$.
- The quantization of samples in a *memoryless quantizer* is not influenced by the quantization history, i.e. each sample is quantized independently.

2.9.1 Scalar Quantization

The simplest form of quantization is scalar quantization where each source sample is quantized individually. Let x denote a continuous scalar signal sample. We view a scalar quantizer $Q(\cdot)$, as a mapping from $x \in \Re$ to a finite set $Y = \{y_0,..., y_{L-1} | y_i \in \Re\}$. The domain (inputs) of $Q(\cdot)$ is partitioned into L non-overlapping *decision levels* or *decision thresholds* while the range (output) of $Q(\cdot)$ is partitioned into L non-overlapping *reconstruction levels* or *representation levels* (Fig. 2.4) The reconstruction levels are referenced by integer indices. Thus while a representation level r_i, might equate to a large floating point number, only the index i need be communicated to the symbol coder (we assume r_i may be reconstructed from i at the decoder) .

$$Q : \Re \rightarrow Y$$

$$y_i = Q(x), \; x \in [d_i, d_{i+1}], \; i = 0,1,..., L-1 \tag{2.10}$$

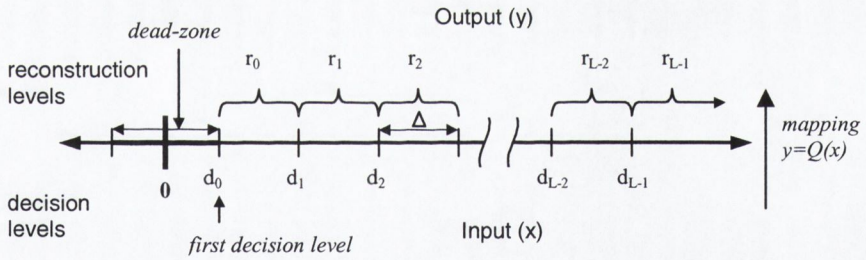


Fig. 2.4 Partition of a continuous domain into decision and reconstruction levels.

The difference between the original and quantized sample is called the *quantization error* or *quantization noise.* Distortion arising from the quantization of source samples from the interval $[d_0, d_L]$ is called *granular distortion* while distortion due to the quantization of samples lying beyond d_L (i.e. $d_{L-1} + \Delta$) is called *overload error* or *overload distortion.* The reconstruction and decision levels are often determined by minimising some error criterion based on the quantization error.

If there is a decision level d_i such that $r_i = 0$, then the quantizer is said to be *midtread.* If there is no reconstruction level r_i that equals zero but a d_i that equals zero, the quantizer is called a *midriser.* Quantizers that have their first decision level d_0 at a non-zero value are said to have a *dead-zone.* In this case, all absolute values less than d_0 are mapped to zero. Similarly, values greater than d_L or less than $-d_L$ are quantized to r_{L-1} and $-r_{L-1}$ respectively.

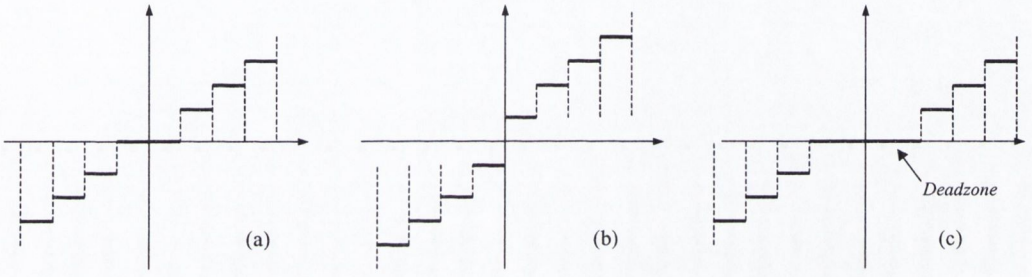


Fig. 2.5 Examples of (a) midtread, (b) midrise, and (c) dead-zone quantizers.

2.9.1.1 Uniform Quantization

The most straightforward method of quantization is uniform quantization (Fig. 2.6), in which the reconstruction levels and decision are uniformly spaced. For a uniform quantizer,

$$\left.\begin{aligned} d_i - d_{i-1} &= \Delta \\ r_i &= \frac{d_i + d_{i-1}}{2} \end{aligned}\right\} 1 \le i \le L$$

where Δ is the step-size equal to the spacing of two consecutive reconstruction or decision levels. In uniform quantization only three parameters need to be considered: the initial decision level, the step-size & the number of levels. Uniform quantization is optimal (in the MSE sense) for scalar quantization when the source has a uniform *pdf* (i.e. flat) [8 p129][19 p27].

2.9.1.2 Non-uniform Quantization

For sources with non-uniform probability densities *non-uniform* quantization will be optimal [9 p594]. In non-uniform quantization the reconstruction and decision levels are not evenly spaced (Fig. 2.6-b). As the *pdf* becomes less uniform, the gain of non-uniform over uniform quantization will increase [9 p597]. The determination of the optimum spacing of decision and reconstruction levels will depend on the error criterion used [9 p593].

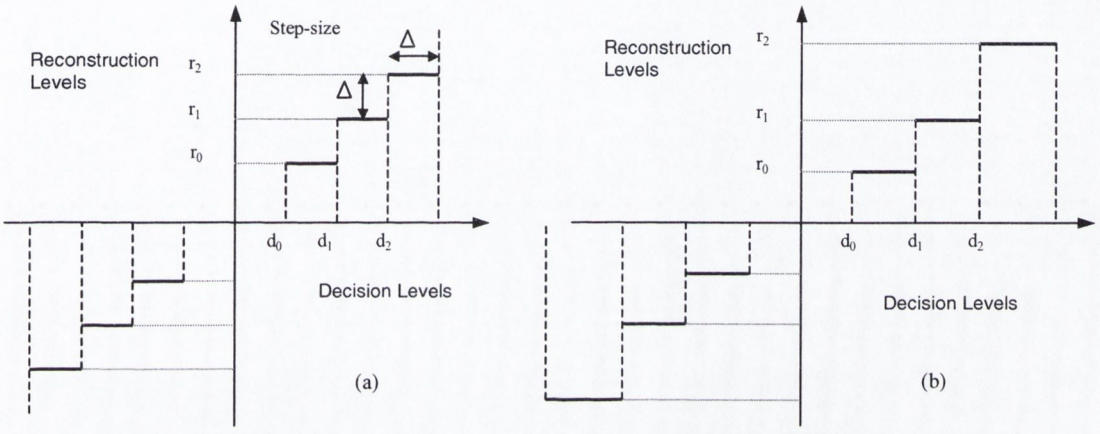


Fig. 2.6 (a) Uniform quantization and (b) Non-uniform quantization.

2.9.1.3 Lloyd-Max Quantization

Quantizers that minimise quantization noise in the MSE sense, for a fixed number of reconstruction levels are referred to as *Lloyd-Max quantizers* after Lloyd [4] and Max [5]. Lloyd-Max quantizers exploit the 1st order *pdf* of the signal to be quantized and are therefore also referred to as *pdf-optimised quantizers* [8, p130]. In a Lloyd-Max quantizer decision levels are selected to be midway between neighbouring reconstruction levels and reconstruction levels are chosen to be the centroids of the *pdf* in the appropriate interval [8 p131].

2.9.1.4 Companding

Given that uniform quantization is optimal when the *pdf* is uniform, another approach to optimal quantization may be taken. If x is first mapped to z by a non-linear function such that the pdf of z is uniform, then with z uniformly quantized the same effect may be achieved (Fig. 2.7). This non-linear mapping is called *companding* [9, p597].

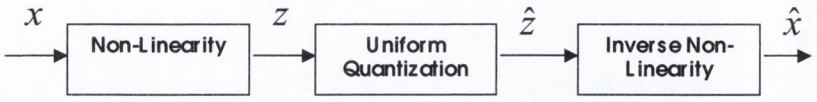


Fig. 2.7 Non-uniform quantization by companding.

2.9.2 Vector Quantization

Thus far, we have considered scalar quantization where each source sample is independently coded. In general however, the performance of an encoding/decoding system can only approach the rate-distortion bound if a number of samples are coded collectively [6, p9] [8 p137]. This form of quantization is known as *vector quantization* (VQ) [13]. Vector quantization may be defined similarly to scalar quantization, but for an N-dimensional space, as a mapping of $\mathbf{x} \in \Re^N$ to $\mathbf{Y} = \left\{ \mathbf{y}_0, \mathbf{y}_1, ..., \mathbf{y}_{L-1} \middle| \mathbf{y}_i \in \Re^N \right\}$.

$$Q : \Re^N \rightarrow \mathbf{Y}$$

In vector quantization $\Re^N$ is partitioned into non-overlapping N-dimensional cells $\{C_i\}$ where

$$\bigcup_{i=0}^{L-1} C_i = \Re^N \text{ and } C_i \cap C_j = 0 \text{ for } i \neq j$$

Thus, $\mathbf{x}$ is quantized to $\mathbf{y}_i$ if it belongs to the cell C_i.

$$\mathbf{x} \in C_i \Rightarrow Q(\mathbf{x}) = \mathbf{y}_i \ , \ i = 0,1,...,L-1$$

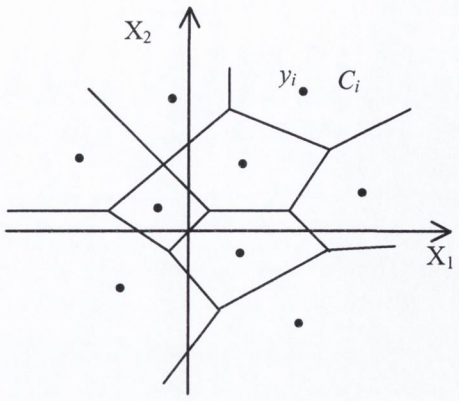


Fig. 2.8 Example of a partitioning of 2-Space into *Voronoi* partitions.

This can be viewed as an extension to the concept of scalar quantization where the vectors $\mathbf{y}_i$ correspond to the reconstruction levels and the cells C_i correspond to the decision levels/regions (Fig. 2.8). The cells C_i are also called *Voronoi regions* or *Dirichlet partitions.* In scalar quantization the mapping of a sample from a decision level to a reconstruction level is straightforward but in VQ, due to the multidimensionality of the samples, a *fidelity criterion* or *distortion measure* $d(\cdot,\cdot)$ is generally used (see [20 p209]). The quantization process therefore involves finding a $\mathbf{y}_i$ to which $\mathbf{x}$ is closest.

$$Q(\mathbf{x}) = \mathbf{y}_i \Leftrightarrow d(\mathbf{x},\mathbf{y}_i) \leq d(\mathbf{x},\mathbf{y}_j),\ j = 0,1,...,L-1$$

In vector quantization both the encoder and decoder must maintain the set of vectors $\{\mathbf{y}_i\}$, also called the *codebook* [8 p136]. When the encoder finds a best-match vector in the codebook, its index is transmitted to the decoder which may then retrieve the same vector from its copy of the codebook. The problem of partitioning the *N*-dimensional space such that the average distortion is minimised is called the *codebook design problem* [10 p405]. The codebook used by the quantizer must be generated *a priori* by analysis of a number of *typical* input data sets (e.g. images). The vectors in the codebook will generally be representative of commonly occurring sample distributions (e.g. pixel intensity patterns).

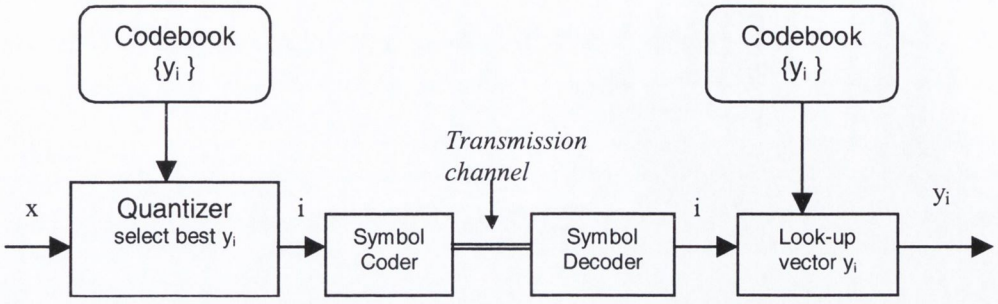


Fig. 2.9 Example of a vector quantizer.

One way in which VQ is applied in image coding is to encode the pixels of an image directly without prior transformation. In practice, the image is partitioned into sub-images or blocks (block-based coding). The dimensions of these blocks will equate to the dimension of the vectors. As the dimension of the vectors is increased VQ approaches the rate-distortion bound, even if the source is memoryless [10 p405][11 p197]. In practice however, the dimension of the vectors in VQ is constrained by coding complexity and memory requirements[11 p197]. The encoder selects a vector from the codebook to which the sample vector is closest (in terms of some distortion measure). The index of this codebook vector is then entropy coded and communicated to decoder. The decoder then performs a look-up in the code book and retrieves the appropriate vector (Fig. 2.9). This approach is advantageous as most of the computation required by the scheme occurs at the encoder while the decoder essentially performs a simple look-up.

Notable variants to the VQ approach described here include *VQ with memory* where predictions are formed from previous vectors in a block based scheme [11 p197] and *adaptive VQ* where the coding rule or codebook is changed over time to track changes in the short-term statistics of the input vectors [11 p202]. Vector quantization has also been applied extensively in the transform domain to code coefficients produced by block-based transforms [11 p206]. Due to the independent manner in which blocks are treated in block-based coding, further statistical redundancies often remain to be exploited by VQ following transformation. VQ is advantageous over scalar quantization due to the freedom with which the vector space may be partitioned and thus the *pdf* of the transform coefficients exploited for improved quantization efficiency. In chapter 3 we discuss the use

of VQ for the coding of transform coefficients in a hybrid scheme employing both VQ and predictive coding (§2.9.3).

2.9.3 Predictive Coding

One of the most simple ways to represent a sequence of pixels is to use *pulse code modulation* (PCM) where each pixel is quantized individually, neglecting the correlation between neighbouring pixels. A slightly more complex approach that does seek to exploit the correlations between adjacent pixels is *differential PCM* (DPCM). In DPCM a predictor estimates the current pixel value based on preceding pixel values and then quantizes and codes the error of the prediction rather than actual pixel value. Predictive coding is thus most efficient if there is a high correlation between adjacent pixels, resulting in a small or zero prediction error. As signals are in general highly correlated, a good estimation of the present sample can often be made from the previous sample or samples [20 p178]. The encoder and decoder must utilise the same prediction strategy so that the decoder may reconstruct pixel values from the error of the prediction. Further discussion on this topic may be found in [20 p178] and [21].

While DPCM is a compression strategy in its own right, it is more often employed in image coding in conjunction with other strategies such as transform coding (chapter 3). For example, in the popular DCT (§3.4) based JPEG [50] coding standard DPCM is employed to code the DC coefficients (§3.2.6).

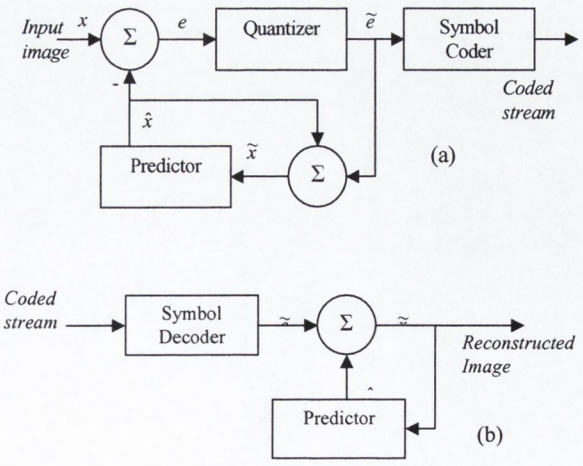


Fig. 2.10 Block diagrams of a (a) DPCM encoder and (b) decoder.

2.10 Summary

In this chapter we have discussed some of the basic principles of image compression and laid the foundation for the more advanced discussions of subsequent chapters. While we have not discussed the details of any particular coding scheme, we have briefly described the two predominant coder models (waveform & model based) and introduced the fundamental concepts of entropy and quantization.

In the next chapter (chapter 3) we introduce the topic of *transform coding*. We shall define the fundamental concepts and outline the main strengths and weaknesses of this approach.

Chapter 3

TRANSFORM CODING

3.1 Introduction

3.1.1 Context

The previous chapter introduced the two principal approaches to image coding: *waveform* and *model based.* In this chapter we focus on the *transformation* stage (§2.6.1) of waveform coding. The image coding literature describes three predominant waveform coding schemes: *transform coders*, *subband coders*, and *wavelet coders.* In fact, both transform and wavelet coding may be viewed as special cases of subband coding [8] (§3.5.5). Although these three approaches are fundamentally related, they are generally discussed from different perspectives using distinct terminologies. In this thesis we adopt the perspective and language of transform coding, but many of the ideas are equally applicable to subband coding in general.

3.1.2 Objectives and Overview

In this chapter we discuss the use of discrete linear transforms for image compression. The objective here is to provide an appreciation for the means by which *image transforms* facilitate compression in transform coding. Any attempt to realise this objective must necessarily provide an overview of the "big picture". Thus, we begin with a discussion of the fundamental ideas and techniques employed in transform coding (§3.2). We proceed to

focus on two specific transforms that occupy places of significant importance in the field of transform coding; the Karhunen-Loève transform (§3.3), and discrete cosine transform (§3.4). The Karhunen-Loève transform (KLT) is an optimal transform (in the MSE sense) that, while usually impractical, serves as a useful benchmark against which to compare other transforms. We employ the theory of the KLT as a basis for the derivation of the discrete cosine transform (DCT) which yields nearly optimal energy concentration/compaction for images modelled as Markov-1† processes yet is considerably easier to implement than the KLT [40][19 p45]. Testament to the appeal of the DCT for image and video coding is its presence at the heart of all the principal image (JPEG [50]) and video coding standards (MPEG-1 [51][52], MPEG-2 [53], H.261 [54], H.263 [55]). We complete our treatment of transform coding with a brief comment on its relationship to subband and wavelet coding (§3.5). Wavelet coding in particular has attracted considerable interest within the research community in recent years, culminating in the selection of the discrete wavelet transform (DWT), as the basis for the next major still image coding standard known as JPEG2000 [41].

3.2 Fundamentals of Transform Coding

This section discusses the fundamental ideas and techniques employed in transform coding. Subsections 3.2.2 through 3.2.4 introduce the mathematics and language of *linear transforms* for image compression, describe the motivations for their use, and investigate the general properties that are desirable in such transforms. In transform coding however, the image transform forms only part of an overall coding scheme. While the transform is central to the compression process, it does not generally achieve compression in itself. Rather, it "re-arranges" the image data to facilitate more effective treatment by subsequent processes and techniques. To fully appreciate the means by which compression is achieved therefore, it is necessary to also address the complimentary topics of *quantization & bit-allocation* (3.2.5) and *zonal & threshold coding* (3.2.6).

Subsection 3.2.7 proceeds to describe the types of distortion that arise in images reconstructed from highly compressed representations, while subsection 3.2.8 identifies some distortion reduction techniques. The final topic treated in this section is hybrid coding (§3.2.9) in which the amalgamation of transform coding and other coding techniques is briefly discussed.

3.2.1 Introduction

While the spatial domain representation of an image is one that humans are inherently familiar with, for purposes of image compression a number of alternative representations[14] are attractive. Mappings from one domain to another are most often implemented in image coding by means of *linear transforms*. The type of transforms that we shall be concerned with in this thesis are those which map image data form the spatial domain to the spatial frequency domain. Such transforms are said to perform *signal decomposition*. The *Fourier transform* is an example of a signal decomposition that transforms an input signal to its constituent frequency components, represented by a linear combination of various sinusoids of different frequency, phase, and magnitude.

The principal motivation for the decomposition of a signal is to make it more amenable to subsequent quantization and coding [8 p23]. The advantages of working with a frequency domain representation of an image are twofold. Firstly, it is generally easier to estimate the perceptual importance of frequency components than it is to characterise the perceptual contribution of individual pixels. It is therefore more straightforward to design perceptually tuned bit-allocation strategies in the frequency domain. Secondly, transforms are generally designed to decorrelate[15] the pixels of an image and compact the image energy into fewer coefficients. Both of these properties are desirable for compression as shall be elaborated in section 3.2.3. The next section (§3.2.2) introduces the mathematics of signal decomposition by series expansion.

3.2.2 Series Expansion of Discrete Time Signals

In simple terms, we wish to be able to represent an image as the sum of a number of "basic building blocks". Furthermore, by selecting building blocks that are fundamental[16] to all images, then different images could be represented/synthesised by combining these building blocks in different proportions, i.e. weighting factors in the summation. This representation has the form of a series expansion. We proceed to express this idea in more formal terms.

---

[14] By *representation*, we mean a "description" of an image.

[15] *Decorrelation* means that each coefficient in the transform domain carries more or less separate information.

[16] It should be noted that while most practical image transforms employ fixed sets of "building blocks", the optimal KLT does not.

Let a sequence of pixels from an image be represented by a discrete-time† finite length signal $x[n]$, $n = 0,1,...,N-1$. We wish to find a set of elementary signals $\{\phi_i\}_{i\in Z}$ such that $x[n]$ may be written as a series expansion of the form [8 p24]

$$x[n] = \sum_{i=0}^{N-1} a_i \phi_i[n] \tag{3.1}$$

where i is an integer index, a_i are the real-valued expansion coefficients and ϕ_i are a set of complex vectors (functions of n) called the expansion set [41 p2].

If the expansion is *complete*[17] and unique, then the vectors of the expansion set $\{\phi_i\}$, are *linearly independent* and form an *orthogonal basis* for the class of signals that may be expressed by the expansion [41 p2]. The vectors of the expansion set are then generally referred to as *basis vectors/functions.* If the expansion is complete but the vectors are not linearly independent, then they do not form a basis and we have an *overcomplete* representation called a *frame* [43 p4,26], i.e. there is redundancy in the expansion set (see Fig. 3.1).

Fig. 3.1 Examples of (a) an orthogonal basis and (b) non-orthogonal (overcomplete) basis for R^2.

Assuming orthogonal (linearly independent) basis vectors, i.e.

$$\langle \phi_i[n], \phi_j[n] \rangle = 0, \; i \neq j \tag{3.2}$$

the coefficients $\{a_i\}$, of the expansion may be calculated by the inner product of the basis functions $\phi_i[n]$ with the input signal $x[n]$

[17] By complete we mean that all signals in the signal space can be represented by the expansion.

$$a_i = \langle \phi_i[n], x[n] \rangle \tag{3.3}$$

Thus, the expression of equation (3.1) can be rewritten as

$$x[n] = \sum_{i=0}^{N-1} \langle \phi_i[n], x[n] \rangle \phi_i[n] \tag{3.4}$$

3.2.2.1 Matrix Representation

It is often more convenient to express a series expansion/signal decomposition in matrix notation. As this is the notation we employ in our discussion on linear transforms (§3.2.3), we shall introduce it here. We begin by arranging the N signal samples in a vector $\mathbf{x}$,

$$\mathbf{x} = [x[0], x[1], ..., x[N-1]]^T$$

and correspondingly for the basis vectors

$$\boldsymbol{\varphi}_i = [\phi_i[0], \phi_i[1], ..., \phi_i[N-1]]^T$$

so that we can rewrite equation (3.1) as [8 p24]

$$\mathbf{x} = \sum_{i=0}^{N-1} a_i \boldsymbol{\varphi}_i \tag{3.5}$$

Incidentally, the orthogonality property may now be expressed by

$$\langle \boldsymbol{\varphi}_i, \boldsymbol{\varphi}_j \rangle = \boldsymbol{\varphi}_i^H \boldsymbol{\varphi}_j = c_i \delta_{ij} \tag{3.6}$$

where δ_{ij} is the Kronecker† delta, $\{c_i\}$ are constants and H denotes the *Hermitian*† transpose (equal to the transpose for real-valued basis functions).

To complete the reformulation of equation (3.1) in matrix notation, we form an $N \times N$ matrix $\boldsymbol{\Phi}$, with the basis vectors $\boldsymbol{\varphi}_i$, as <u>columns</u>

(3.7)

$$\mathbf{\Phi} = [\boldsymbol{\varphi}_0, \boldsymbol{\varphi}_1, ..., \boldsymbol{\varphi}_{N-1}]$$

and write the expansion coefficients as a column vector

$$\mathbf{a} = [a_0, a_1, ..., a_{N-1}]^T \tag{3.8}$$

to arrive at [8 p24]

$$\mathbf{x} = \mathbf{\Phi a} \tag{3.9}$$

Thus the expansion is expressed as the multiplication of a $N \times N$ matrix of basis vectors with a $N \times 1$ matrix of expansion coefficients. An expression for the expansion coefficients may be obtained by pre-multiplying equation (3.9) by $\mathbf{\Phi}^{-1}$, yielding[18]

$$\mathbf{a} = \mathbf{\Phi}^{-1}\mathbf{x}, \tag{3.10}$$

which is more commonly expressed by introducing the matrix[19] $\mathbf{\Psi} = \left(\mathbf{\Phi}^{-1}\right)^H$, and writing

$$\mathbf{a} = \mathbf{\Psi}^H \mathbf{x} \tag{3.11}$$

We view the expansion as a transform/mapping of $\mathbf{x}$ from one coordinate system/domain to another in which pixels have been replaced by expansion coefficients. The expansion/transform may be viewed as a change of basis or simply as another way of "looking at" the image data. In the new representation, the expansion coefficients effectively carry all the information about the signal, while the basis vectors provide a "frame of reference". Many expansions will also have the *successive approximation* property which means that by selecting only a subset of the basis functions, a good (meaningful) approximation of the signal (image) can be reconstructed [43 p93]. This is a very useful property for compression. We elaborate this discussion in section 3.2.3.

[18] Recall that the columns of $\mathbf{\Phi}$, i.e. the basis vectors, are linearly independent.

[19] The columns of $\mathbf{\Psi}$ are called the *reciprocal* basis vectors [8 p25].

3.2.2.2 Unitary Transforms

A particularly interesting class of expansions are those with *orthonormal* basis vectors, i.e. orthogonal basis vectors of unit length such that $\boldsymbol{\varphi}_i^H \boldsymbol{\varphi}_j = \delta_{ij}$. A matrix of basis vectors with this property is said to be *unitary* [8 p25]. Orthonormal expansions preserve both the first order (linearity) and second order (correlations) structure of the vector (signal) space [42 p17] and conserve signal energy [43 p96] [11 p226]. For a unitary matrix

$$\boldsymbol{\Phi}^H = \boldsymbol{\Phi}^{-1} \tag{3.12}$$

and so the matrix of reciprocal vectors is equal to the matrix of basis vectors

$$\boldsymbol{\Psi} = \boldsymbol{\Phi} \tag{3.13}$$

and we may obtain the expansion coefficients by

$$\mathbf{a} = \boldsymbol{\Phi}^H \mathbf{x} \tag{3.14}$$

or any single coefficient by

$$a_i = \boldsymbol{\varphi}_i^H \mathbf{x} \tag{3.15}$$

For completeness we return to equation (3.1) and substitute equation (3.15) for a_i to express the series expansion, for the unitary case as

$$x[n] = \sum_{i=0}^{N-1} (\boldsymbol{\varphi}_i^H \mathbf{x}) \phi_i[n] \tag{3.16}$$

The question arises as to what makes a good basis for the expansion when image compression is the objective. While there is no clear answer to this question, in general a good basis is one that allows a compact representation or less complex processing [43 p5], see section 3.3 for additional comments. In the following section we explore this question further from the perspective of transform coding.

3.2.3 Linear Transforms for Image Compression

This section introduces the mathematics and language of linear transforms for image compression, and notes the properties desirable in such transforms. Let the vector $\mathbf{x}$ represent a discrete time signal of length N, $\mathbf{x}^T = [x_0, x_1, ..., x_{N-1}]$. Each x_i represents a sample of the source signal. We define a linear operator $\mathbf{H}$, on $\mathbf{x}$, that yields a vector $\mathbf{y}$

$$\mathbf{y} = \mathbf{H}\mathbf{x} \tag{3.17}$$

where $\mathbf{y}^T = [y_0, y_1, ..., y_{N-1}]$. We call the matrix $\mathbf{H}$, the *forward transform kernel*[20] (or *transform matrix*) and the elements y_i, *transform coefficients*[21].

$$\mathbf{H} = \begin{bmatrix} \phi_{00} & \phi_{01} & \cdots & \phi_{0,N-1} \\ \phi_{10} & \phi_{11} & & \\ \vdots & & \ddots & \\ \phi_{N-1,0} & & & \phi_{N-1,N-1} \end{bmatrix} \tag{3.18}$$

The rows[22] of $\mathbf{H}$ are the *basis vectors* of the transform, forming an orthogonal basis for the N-tuples over the real field. Any transform coefficient is thus calculated by

$$y_i = \sum_{j=0}^{N-1} \phi_{ij} x_j \tag{3.19}$$

Given a set of transform coefficients we may recover the original signal, or an approximation thereof, by application of the *inverse transform*. Assuming $\mathbf{H}$ is non-singular, we may recover $\mathbf{x}$ by

$$\mathbf{x} = \mathbf{H}^{-1}\mathbf{y} \tag{3.20}$$

[20] A transform kernel is referred to as a *filter bank* in subband coding. A forward transform kernel is called an *analysis* filter bank and inverse transform kernel is called a *synthesis* filter bank [8 p187].

[21] Transform coefficients are referred to as *subband signals* in subband coding.

[22] If the basis vectors form the columns of the kernel then it must be transposed to perform the transform.

A transform with an exact inverse is said to have the *perfect reconstruction* (PR) property [8 p23][43 p93], i.e. the original signal may be reconstructed, without distortion, by application of the inverse transform on the transform coefficients

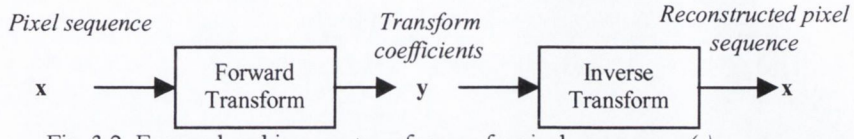


Fig. 3.2 Forward and inverse transforms of a pixel sequence *x(n)*.

For a real[23] orthogonal transform, the inverse transform will simply be the transpose of the forward transform matrix..

$$\mathbf{x} = \mathbf{H}^T \mathbf{y} \tag{3.21}$$

It should be noted that if $\mathbf{H}$ is unitary then $\mathbf{H}\mathbf{H}^T = \mathbf{I}$, the identity matrix.

3.2.3.1 Separable Transforms

Separable transforms represent an important class of transforms in image coding. A transform is said to be separable if it can be applied to a two-dimensional data set (e.g. an image) as a succession of one-dimensional transforms. A two-dimensional transform $a(n_1, n_2)$, is separable if it can be expressed as

$$a(n_1, n_2) = g(n_1) f(n_2) \tag{3.22}$$

where $g(n_1)$ is a function only of n_1 and $f(n_2)$ is a function of only n_2. Furthermore, a transform is *symmetric* if g is functionally equal to f so that

$$a(n_1, n_2) = g(n_1) g(n_2) = f(n_1) f(n_2). \tag{3.23}$$

A separable transform is applied to an image matrix $\mathbf{X}$, by first transforming each row of the matrix independently. The results are stored in a matrix of equal dimension and over it a second transform is applied to each column. We assume that both transform matrices

[23] By real, we mean that the basis vectors are real valued rather than complex.

are real orthogonal matrices such that $\mathbf{H}^T = \mathbf{H}^{-1}$, i.e. the matrix inverse is equal to its transpose. If $\mathbf{H}_R$ and $\mathbf{H}_C$ are the row and column transforms respectively then $\mathbf{X}$ is transformed to Y by

$$\mathbf{Y} = \mathbf{H}_R \mathbf{X} \mathbf{H}_C^T \tag{3.24}$$

If a discrete orthogonal transform is separable, then so is its inverse [20 p92]. The original image matrix $\mathbf{X}$ may be recovered from $\mathbf{Y}$ by the inverse transform

$$\mathbf{X} = \mathbf{H}_R^T \mathbf{Y} \mathbf{H}_C \tag{3.25}$$

Numerous transforms have been developed with the property of separability [20], these include the *discrete cosine transform* (DCT), *Walsh-Hadamard transform* (WHT), *discrete Fourier transform* (DFT), *slant transform*(ST), and *Harr transform* (HT).

If a transform is both separable and symmetric ($\mathbf{H} = \mathbf{H}^T$ and $\mathbf{H}_R = \mathbf{H}_C$) then we may express the forward and reverse transforms by $\mathbf{Y} = \mathbf{HXH}$ and $\mathbf{X} = \mathbf{HYH}$ respectively, where $\mathbf{H} = \mathbf{H}^T = \mathbf{H}^{-1}$.

3.2.3.2 Desirable Properties

There are a number of key properties that are desirable in linear transforms for image compression. We have discussed some of these elementary properties in the preceding sections, but reiterate them here for clarity.

Inversion: Transforms employed for image compression must be invertible, i.e. it must be possible to recover the original signal by application of an inverse transform. In a lossy coding scheme, *perfect reconstruction* of the original image is not required, thus need not be a requirement of the transform either. In such cases transforms with the *almost perfect reconstruction* property are also appropriate.

Separability. Separable transforms are desirable for the ease with which they may be applied to two-dimensional data, i.e. images. They are also advantageous from a computational viewpoint. If fast algorithms exist for the one-dimensional transform, they may also be exploited in the 2-D case.

Orthogonality: Orthogonal transforms are advantageous for a number of reasons:

1. As the basis functions of an orthogonal transform are linearly independent, there will be no data redundancy in the transform domain[24].

2. The inverse of an orthogonal transform (or unitary, with appropriate normalisation) may be produced by simply transposing[25] the forward transform kernel.

3. Unitary transforms preserve signal energy. If $\hat{\mathbf{x}}$ and $\hat{\mathbf{y}}$ are the approximate versions of $\mathbf{x}$ and $\mathbf{y}$ respectively (see quantization, section 3.2.5) then [43 p375]

$$\|\mathbf{x}-\hat{\mathbf{x}}\| = \|\mathbf{y}-\hat{\mathbf{y}}\| \tag{3.26}$$

This means that small errors in the transform domain will equate to small errors in the signal domain [43 p276], i.e. they won't become magnified in the reconstructed image. This property facilitates the design of simple quantizers for use in the transform domain, with predictable results in the spatial domain. For orthonormal transforms, distortion introduced in the signal domain may be predicted from the distortion in the transform domain due to the conservation of energy property of unitary transforms. Assuming a stationary signal [11 p226][8 p160][11 p267]

$$\sigma_x^2 = \frac{1}{N}\sum_{i=0}^{N-1}\sigma_i^2 \tag{3.27}$$

where σ_i^2 is the variance of y_i and σ_x^2 is the variance of the input signal.

Energy compaction & correlation reduction. A transform that yields a sequence of uncorrelated (or less correlated) coefficients is said to have the property of *correlation reduction.* This essentially means converting statistically dependent pixels to more or less independent coefficients. This property is not in itself sufficient for image compression[26] however, it is also desirable that the image energy be redistributed amongst a smaller set of transform

[24] This is a distinct concept to that of correlation reduction.

[25] Restricting our discussion to images, i.e. real valued signals.

[26] White noise is uncorrelated yet doesn't present any useful properties for compression.

coefficients. Such transforms are said to have the *energy compaction property* (or produce a *compact* representation) [42 p33]. Furthermore, if many of the coefficients are very small or zero, the representation is said to be *sparse* [41 p8]. While there is no general formal proof that decorrelation results in more efficient compression [43 p372], practical results demonstrate an advantage.

3.2.3.3 Ordering the basis vectors

For reasons of coding efficiency (see § 3.2.6) it is advantageous to arrange the basis vectors of a transform such that, for a typical image, the transform coefficients occur in a *monotonic decreasing sequence*[27]. In practice, this generally means that basis vectors will be ordered in terms of the spatial frequencies that they represent. As most of the information in natural images is concentrated in the lower frequencies, the ordering is generally one of increasing frequency.

As the transform coefficients generated by a 2-D transform will form a 2-D matrix, a scanning pattern must be defined to reduce the matrix to a one dimensional sequence. A common approach is to use a *zigzag* scanning pattern (Fig. 3.3)[19 p29, p76] (although other patterns are also used [19 p304]). Zigzag scanning assumes that increasing horizontal spatial frequencies (vertical edges) are represented from left to right in the coefficient matrix, and increasing vertical frequencies (horizontal edges) from top to bottom.

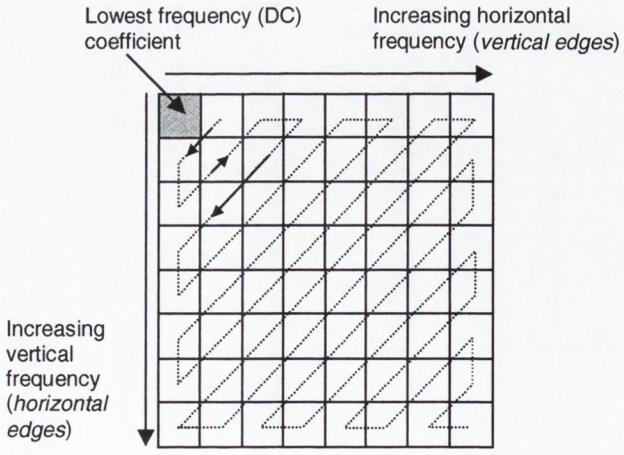


Fig. 3.3 Zigzag scanning for an 8x8 coefficient matrix.

[27] i.e. a non-increasing sequence.

3.2.4 Block Transform Coding (BTC)

In practise, rather than apply a single transform over an entire image, the image is often divided into a number of smaller consecutive sub-images or *blocks*, and each block transformed separately. This is called *block decomposition* [8 p24] and is advantageous primarily for reasons of computational[28] and coding efficiency. In fact, the phrase *transform coding* [40] or *block transform coding* (BTC[29]) [85] is generally understood to refer to this approach. The block based approach reduces considerably the number of calculations required to generate the transform coefficients. Furthermore, there is generally no significant coding efficiency for block sizes greater than 16×16 [11 p226][20 p171], and block sizes 8×8 are typical [9 p647]. BTC is also advantageous in that it facilitates the introduction of adaptive features based on block activity [20 p166](§3.2.6.2).

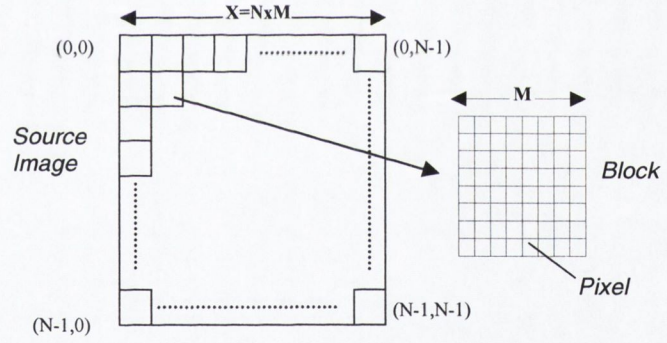


Fig. 3.4 Block-partitioning of an image
An image of dimensions $X \times X$ (square for simplicity), divided into $N \times N$ blocks of width M pixels such that $X = N \times M$.

3.2.5 Quantization and Bit Allocation

In a *lossy* coding scheme, where perfect reconstruction is sacrificed to achieve compression, transform coefficients undergo *quantization* prior to coding (§2.9). In fact, this is one of the principal means by which compression is achieved in transform coding.

[28] In addition to reducing the total number of calculations required, the block based approach facilitates parallel transformation of each block on suitable hardware.

[29] We use the acronym BTC here to refer to *Block Transform Coding* and not *Block Truncation Coding* [19] as it is also used.

The quantized sequence of transform coefficients is denoted $\hat{\mathbf{y}}$, and the reconstructed sequence $\hat{\mathbf{x}}$.

$$\hat{\mathbf{x}} = \mathbf{H}_N^{-1}\hat{\mathbf{y}} \tag{3.28}$$

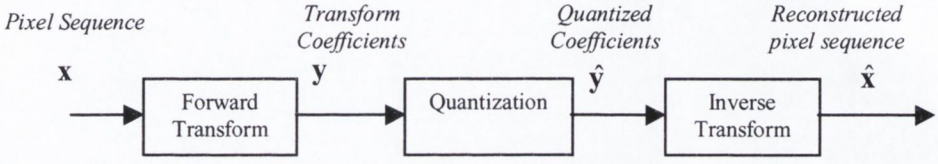


Fig. 3.5 Forward and inverse transforms of a pixel sequence *x(n)* with quantization.

Assuming a transform has the PR property, the quantization of transform coefficients will be the sole[30] cause of distortion in the reconstructed image. The quantization strategy therefore deserves considerable attention.

As different transform coefficients contribute different degrees of visual information in an image, it can be advantageous for compression to consider representing different coefficients with different degrees of precision (and hence number of bits). This is the problem of *bit-allocation.*

One approach to the problem is to allocate bits proportionally to the variance σ_i^2, $i = 0,1,...,N$ of each coefficient [19 p28] [8 p188], where N is the number of retained coefficients. Bits will thus be allocated according to the dynamic range of each coefficient. As the variances represent the energy or information content of the corresponding transform coefficients [20 p123], coefficients with large variances may be interpreted as containing the significant features of an image. In fact, assuming a zero mean source, coefficients with relatively small absolute values may often be discarded completely without seriously affecting the appearance of the reconstructed image.

[30] The limitation of finite precision for the representation of coefficients is assumed to be of negligible influence when reconstructing integer valued pixels.

The average reconstruction error variance will be equal to the quantization error of the transform coefficients [19 p28] (3.27). If a total of B bits are available to store N coefficients then the bits allocated for any coefficient is given by [10 p381]

$$b_i = \frac{B}{N} + \frac{1}{2}\log_2 \sigma_i^2 - \frac{1}{2N}\sum_{i=0}^{N-1}\log_2 \sigma_i^2 \qquad (3.29)$$

where rounding to integer values will be required, satisfying

$$B = \sum_{i=0}^{N-1} b_i \qquad (3.30)$$

For the 2-D case a *bit allocation matrix* (Fig. 3.6-a) may be generated for the $M \times M$ coefficient matrix, and the bits allocated to any coefficient at location i, j calculated by [19 p28]

$$b_{ij} = \frac{B}{M^2} + \frac{1}{2}\log_2 \sigma_{ij}^2 - \frac{1}{2M^2}\sum_{k=0}^{M-1}\sum_{l=0}^{M-1}\log_2 \sigma_{ji}^2 \qquad (3.31)$$

Once the number of bits to allocate to each coefficient has been determined, a different Lloyd-Max (§2.9.1.5) quantizer may be defined for each coefficient [10 p382] [8 p194]. This approach, of course, requires some knowledge of the *pdfs* of the various coefficients.

Tescher and Cox [47] have shown that, for the DCT, coefficient variances are strongly correlated along the zigzag scan pattern [20 p174], approximated by a linear relationship in the logarithmic domain [48]. By transmitting only the slope and intercept of this variance distribution, and the average bit rate, the decoder can reconstruct the bit-allocation matrix [19 p29].

Assuming uniform quantization, as is the case in most practical transform coders, the bit-allocation problem becomes one of determining the most appropriate quantization step size for the various coefficients. In this case, a *quantization matrix* may be produced to represent different quantization step sizes (Fig. 3.6-b). The principal aspect of the information a quantization matrix conveys is the *relative* size of the quantization steps [43 p402]. In BTC, a single quantization matrix may be generated for all blocks and scaled, i.e.

multiplied by a constant, to adaptively suit the rate-distortion function of individual blocks [43 p403][10 p383]. In this case, only the scaling factor need be transmitted for each block.

8	7	6	4	3	0	0	0
7	6	5	4	0	0	0	0
6	5	4	0	0	0	0	0
4	4	0	0	0	0	0	0
3	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

(a)

16	11	10	16	24	40	51	61
12	12	14	19	26	58	60	55
14	13	16	24	40	57	69	56
14	17	22	29	51	87	80	62
18	22	37	56	68	109	103	77
24	35	55	64	81	194	113	92
49	64	78	87	103	121	120	101
72	92	95	98	121	100	103	99

(b)

Fig. 3.6 Bit-allocation (a) and quantization matrices (b).
Note that we assume a specific ordering of the basis functions. Here, horizontal frequencies increase from left to right, and vertical frequencies increase from top to bottom.

In addition to statistical and information theoretic considerations when allocating bits to coefficients, the contributions made by different coefficients to the image in perceptual terms should also be considered. Schemes that incorporate aspects of the HVS in quantization are said to employ *HVS weighting*[31] [11 p238]. At its most simple, this usually involves reducing the precision of higher spatial frequencies [19 p78], i.e. coarser quantization. More perceptually tuned quantization matrices rely heavily on experimental results [43 p402]. The quantization matrix presented in Fig. 3.6-b is the quantization matrix used in the popular JPEG[32] [50] coding standard.

3.2.6 Zonal and Threshold Coding

Thus far we have assumed that all transform coefficients could be candidates in the bit-allocation process. For reasons of coding efficiently, this may not always be the case. We must consider how a coefficient selection might be communicated to the decoder, and what overhead in terms it bits, might be generated. There are two principal issues: how to select the coefficients to be coded and how to efficiently communicate this selection to the

[31] HVS Weighted quantization may also be employed in other schemes such as DPCM [63].

[32] The JPEG standard get its name from the group responsible for its specification – the **J**oint **P**hotographic **E**xperts **G**roup.

decoder. The two principal approaches to these interrelated problems are *zonal coding* and *threshold coding* [10 p381][9 p647].

3.2.6.1 Zonal Coding

In zonal coding, a binary mask known as a *zonal mask* is used to indicate which coefficients have been retained and which have been discarded. Let us consider an $n \times n$ coefficient matrix $\mathbf{Y}$. We may generate an $n \times n$ binary zonal mask $\mathbf{Z}$ for $\mathbf{Y}$. Each bit in $\mathbf{Z}$ will indicate if the coefficient at the corresponding location in $\mathbf{Y}$ will be retained or discarded (Fig. 3.7).

In BTC a single zonal mask may be generated for all blocks based on the global characteristics of the image. This approach is efficient in that the locations of retained coefficients need not be communicated to the decoder for each block. By using a zonal mask in conjunction with a bit-allocation matrix and assumed scanning order, coefficients of variable bit-length may be concatenated and recovered without ambiguity. The approach will be locally sub-optimal however, where the block characteristics deviate from the norm by any significant degree. In some blocks, large coefficients lying outside the zone will be discarded while zero valued coefficients lying within the zone are coded.

$$\mathbf{Y} = \begin{bmatrix} 180 & 58 & 15 & 2 \\ 68 & 45 & 3 & 0 \\ 10 & 5 & 0 & 1 \\ 1 & 0 & 0 & 1 \end{bmatrix} \quad \mathbf{Z} = \begin{bmatrix} 1 & 1 & 1 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \quad \hat{\mathbf{Y}} = \begin{bmatrix} 180 & 58 & 15 & 0 \\ 68 & 45 & 0 & 0 \\ 10 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

Fig. 3.7 Zonal Coding
Example of a 4×4 coefficient matrix $\mathbf{Y}$, with zonal mask $\mathbf{Z}$, and truncated coefficient matrix $\hat{\mathbf{Y}}$.

3.2.6.2 Threshold Coding

Threshold coding is a locally adaptive technique where coefficients are retained on the basis of their magnitude rather than their locations within a block. Threshold coding relies on the premise that the basis functions have been ordered such that, for a typical image, the transform coefficients occur in a monotonic decreasing sequence (when zigzag scanned for example). Assuming the magnitude of a coefficient is proportional to the information

it conveys, the coefficients are likely to be ordered in terms of their importance for image reconstruction (see § 3.3.3). We would therefore typically expect the biggest, and hence most important, coefficients to occur towards the start of the sequence, followed by a run of zero or near zero coefficients towards the end [8 p201]. This increasing proportion of zero values towards the end facilitates efficient representation through *run-level* coding [21] [19 p76] [44]. Coefficients are coded in tuples that represent the magnitude of a coefficient and the number of preceding zeros [10 p383] (in the scanning order). An *end-of-block* (EOB) code may be inserted following the last non-zero coefficient. Finally, the run-length code words may be entropy coded.

Whichever method is used to select and code transform coefficients, where block based frequency decompositions are concerned, the lowest frequency component of each block, often called the DC[33] coefficient[34], is generally coded separately from the other (AC) coefficients due to inter-block correlations between them [8 p188, p200][19 p113]. Furthermore, small errors in the quantization of the DC coefficients can produce significant perceptual distortions in the reconstructed image. This fact necessitates the quantization of DC coefficients with high precision.

3.2.7 Distortion in Transform and Subband Coding

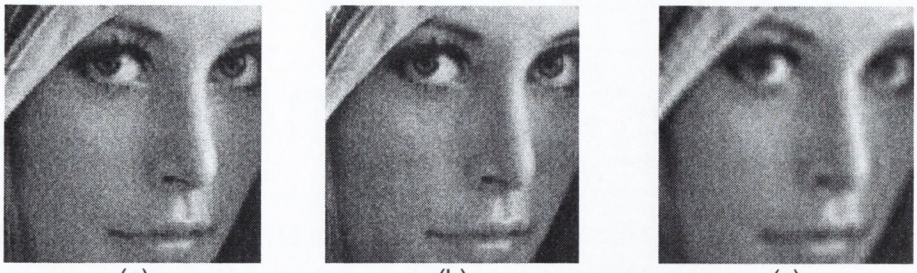
There are two classes of distortion introduced in highly compressed images reconstructed from frequency domain transforms: *blurring* (or lack of detail) and *image artifacts*. Blurring is generally seen as a less annoying form distortion as it occurs "naturally" in many images and therefore tends to attract less attention [8 p219]. Image artifacts by contrast, can be particularly annoying.

3.2.7.1 Blurring Distortion

In both transform and subband/wavelet coders, blurring (Fig. 3.8-c) is caused by the truncation of high frequency coefficients [8 p207].

[33] DC meaning *direct current* is a term inherited from the field of electronics. The DC coefficient is simply the mean value of the image/block.

[34] The DC coefficients are equivalent to the *low-low* band in subband coding.



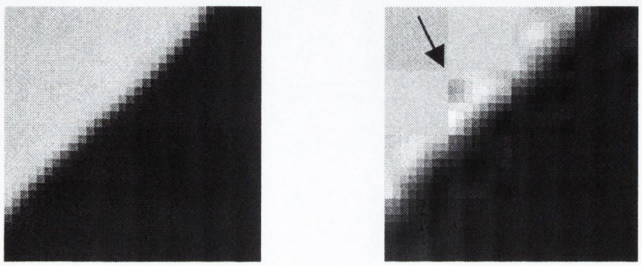
(a) (b) (c)

Fig. 3.8 Examples of distortion in transform coding (not block based). Original image (a), graininess due to coarse quantization (b) and blurring due to truncation of high frequency coefficients (c).

The amount of blurring distortion introduced in a reconstructed image will be determined by the choice of threshold value, t (i.e. the dead-zone, see §2.9.1). By making t too small, many high frequency coefficients will be retained, requiring more bits for their representation, and making the run-length coding strategy (§2.8.6) less efficient [8 p207] Furthermore, if coefficients from weak high frequency textures are retained, but then coarsely quantized, artifacts may become visible in visually smooth image areas [8 p207]. This type of distortion can be described as *granular noise* or *graininess* [9 p651] [10 p385][20 p180] (§Fig. 3.8-b). If t is made larger, graininess is reduced but the degree of blurring increases.

3.2.7.2 Ringing Noise

Ringing noise, closely related to *Gibbs phenomenon* [8 p209], manifests as artifacts around regions of sharp intensity transitions, i.e. edges. It is most visible in transitions from one smooth region to another and is effectively masked in highly textured regions. While ringing noise is the predominant type of distortion in traditional subband coders (§3.5) [8 p218], it generally only occurs in BTC at very low bit rates (see Fig. 3.9). In such cases however, it may be particularly annoying.



(a) (b)

Fig. 3.9 Illustration of *ringing* in BTC at a very low bit-rate. Note that the effect is contained within each block, and does not impact "globally" to any extent. Original (a), DCT coded (b)

3.2.7.3 Blocking Effects

Another undesirable side effect of BTC at very low bit-rates, is a distortion known as *blocking effects* (or *blocking artifacts*)[19][20]. This distortion arises due to the independent treatment of blocks during transformation and quantization, and manifests as the appearance of block boundaries, or grid structure, in the reconstructed image. Blocking effects are generally most visible in smooth image areas and areas with strong contours. The particular visibility of blocking effects has its origin in two perceptual phenomena; *Mach bands* and *Vernier offset.*

(a) (b)

Fig. 3.10 Example of blocking effects in transform coding. Original (a) and reconstructed image (b), enlarged for clarity.

Mach Bands

Ernst Mach was an Austrian physicist who noted that when two different uniform intensity distributions abut (§Fig. 3.11), their perceived intensity, or lightness, changes in the region about the boundary [81]. The same phenomenon[35] is responsible for the "exaggerated" visibility of blocking effects in BTC. At high compression ratios, small differences in pixel intensity, introduced across block boundaries during coding, are amplified by the HVS, giving rise to the perception of *false contours.* In this respect, the independent treatment of blocks in BTC is a fundamental weakness for low bit-rate applications.

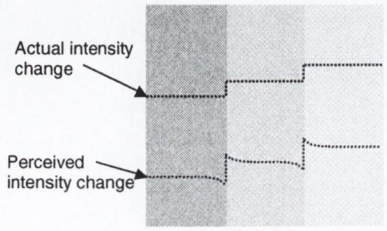
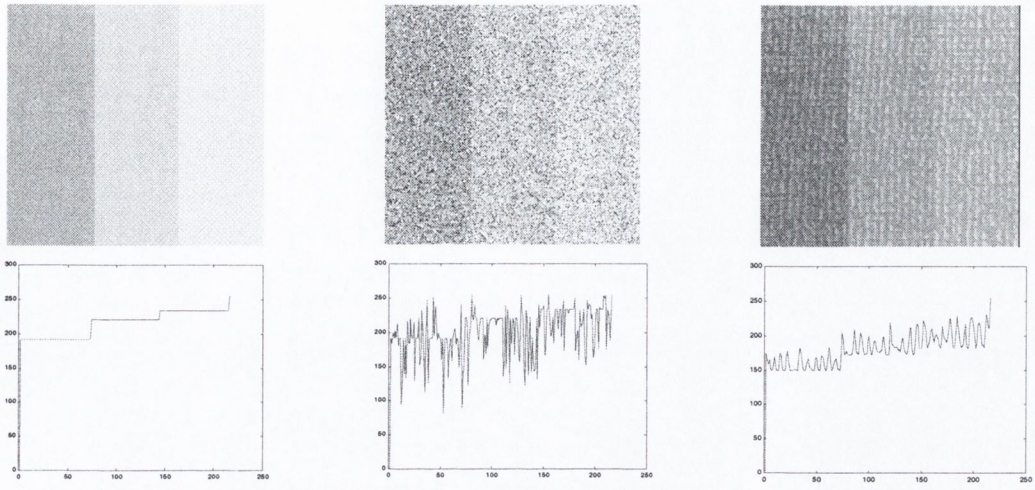


Fig. 3.11 Illustration of Mach bands, actual and perceived intensity changes.

[35] See also chapter 4, section 4.2.3.



(a) (b) (c)

Fig. 3.12 Illustration of Mach bands with intensity profiles of horizontal cross section. Uniform (a), in the presence of noise (b), and with texture (c). Notice that even with a very noisy image, the boundaries are visible.

Vernier Offset

Another perceptual phenomenon responsible for the appearance of blocking effects is the human visual system's sensitivity to *Vernier offset.* As shall be discussed in chapter four, the HVS is particularly sensitive to abrupt changes in light intensity, characteristic of the outlines or *contours* of objects. Furthermore, the HVS is acutely sensitive to *discontinuities* in contours, i.e. human vision is very good at detecting small departures from the co-linearity of two line segments [75][92][93].

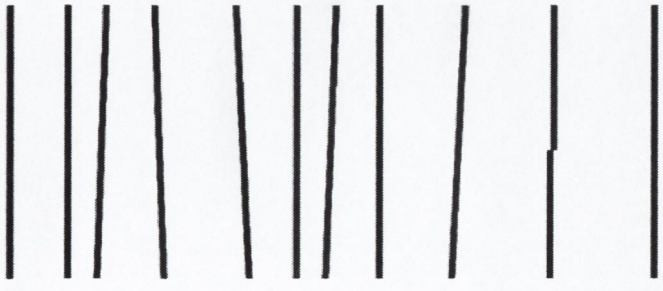


Fig. 3.13 Illustration of HVS sensitivity to Vernier offset. The line second from the right has been broken and the bottom part shifted to the left. The attention of the observers is quickly brought to this line.

At high compression ratios, BTC schemes are faced with the task of representing arbitrarily oriented contour details, across block boundaries, with a small number of "directionally limited" basis functions. The consequence of this in the reconstructed image is the introduction of artificial contour discontinuities (Vernier offset) at block boundaries. Fig.

3.13 illustrates the perceptual attraction (or distraction) of Vernier offset, and Fig. 3.14 demonstrates its manifestation in JPEG at a high compression ratio.

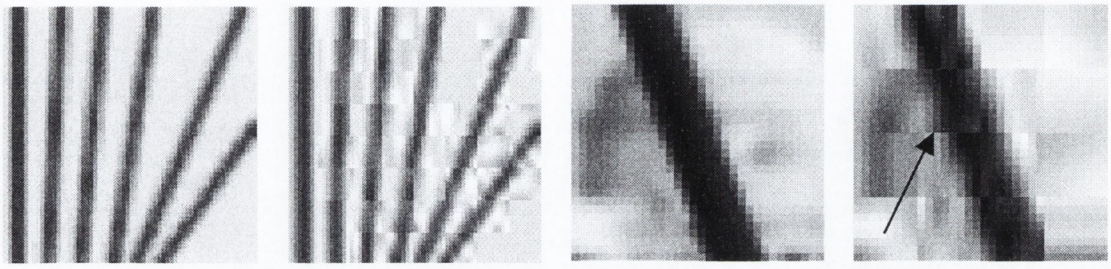


Fig. 3.14 Introduction of edge discontinuity artefacts introduced by JPEG. as a result of forcing a high compression ratio (zoomed). Original images (a) and (c) and reconstructed images (b) and (d).

A number of approaches have been taken to address the problem of blocking effects in transform coding, most of which fall into one of two categories; those that take place at the encoder, and those that are applied subsequent to decoding, when the distortion has already occurred. The following section addresses the reduction of blocking effects.

3.2.8 Reduction of Blocking Effects

3.2.8.1 Block Overlapping

One of the most simple encoder-side solutions for the reduction of blocking effects is to overlap the perimeter of adjacent blocks. Pixels lying within overlapping regions have their values encoded and transmitted more than once. These pixels are then averaged at the decoder, thus producing a smoothening effect. This approach however, increases the number of operations required to encode and decode the image and results in an increased bit-rate, an overlap of more than one pixel is therefore seldom considered for image coding [9, p653].

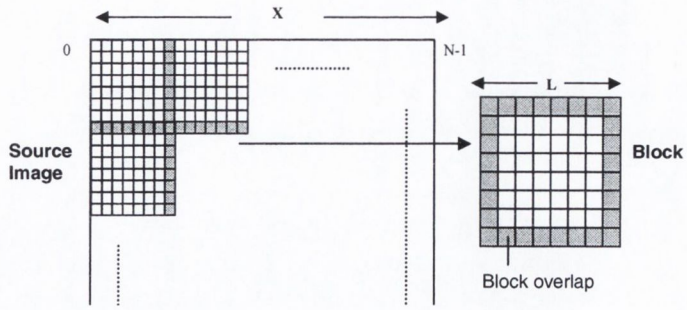


Fig. 3.15 Block overlapping
An image of dimension X, divided into N overlapping blocks of width L pixels, and overlap factor $N_0 = \frac{L}{M}$, such that $X = \frac{NL}{N_0}$.

3.2.8.2 Lapped Orthogonal Transforms

In 1988 Cassereau et al. [49] and Malvar and Staelin [24][25][26][29] introduced a new class of transforms, known as *lapped orthogonal transforms*[36] (LOT), that overlap adjacent blocks, but without generating additional transform coefficients, and hence bit-rate. The overlapping basis functions of the LOT decay smoothly towards zero at their boundaries [20,p287]. Typically, an overlap of one block is used [45 p186] which means the basis functions are of length L=2M, where M is the number of blocks. In this respect LOTs may be viewed as an intermediate case of transform and subband coding[37] [8 p210]. Each block is represented with a fewer number of transform coefficients than would be used for normal BTC. As a result, individual blocks cannot be reconstructed fully from their own transform coefficients alone, but require the overlapped regions to be combined. LOTs have been shown to reduce blocking effects [24] but other artifacts, such as ringing around edges, tend to appear due to the longer basis functions [43 p404]. While the LOT may be considered an improvement over standard block based coding it has not replaced it in any major image coding standards. Reasons for this have been cited as

- Its increased computational requirements, approximately twice the number of operations over the DCT [20 p288] (Existing fast algorithms require in the order of $N \log N$ operations [43 p404]).

[36] In subband coding, the LOT would be described as a filter bank with relatively short filter responses [8 p182]
[37] In fact, they may be considered a special kind of critically subsampled paraunitary filter banks [45 p186]

- Its complexity for VLSI implementation [43 p404]

3.2.8.3 Low Pass Filtering

A decoder-side approach to the reduction of blocking effects is to filter block boundaries once the image has been reconstructed [9 p643][61]. The visibility of blocking effects is due to the high frequency nature of abrupt intensity variations at the block boundaries. A low pass filter is therefore generally applied to block boundary regions to reduce the abruptness of intensity changes across the block boundaries (see Fig. 3.16). In this approach, the encoding process is unchanged.

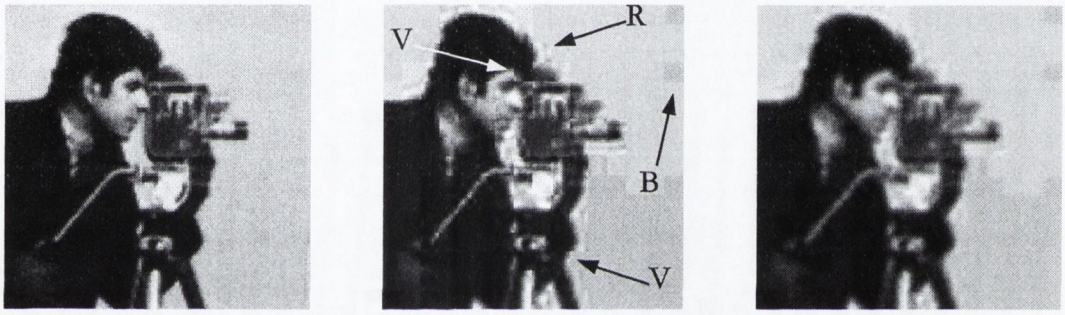


Fig. 3.16 Illustration of distortion reduction by low pass filtering (blurring). Original image (a), distorted image (b) and blurred image (c). Cases of particular distortions are labelled by B,R,&V indicating blocking in smooth areas, ringing, and Vernier distortion respectively.

3.2.9 Hybrid Coding

Transform coding may be combined with other techniques, such as DPCM and VQ, in *hybrid* schemes. The motivation for the use of hybrid techniques is either improved performance or reduced complexity [8 p279]. These aims are sought to be achieved by exploiting the strengths of one scheme to offset the weaknesses of another [20 p178].

3.2.9.1 Transform/DPCM Coding

Hybrid transform/DPCM coding, first proposed by Habibi [62], has a number of different "flavours". One approach [64] is to apply a one-dimensional transform to the lines of an image, and then code the resulting coefficients, in columns, by DPCM (Fig. 3.17). This

method seeks to combine the robustness and compression efficiency of transform coding, with the simplicity of DPCM.

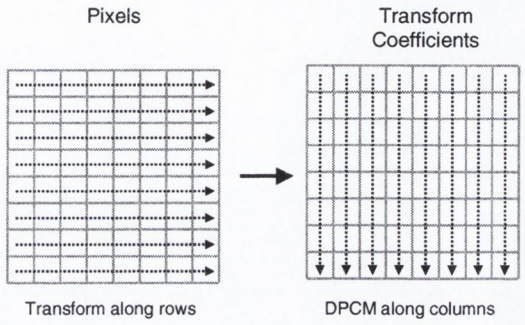


Fig. 3.17 1-D Transform/DPCM hybrid coding

This idea may be also be extended to two dimensional transforms. Due to the independent manner in which blocks are treated in transform coding, there is often correlation remaining between the coefficients of adjacent blocks. Following block decomposition, coefficients are collected into blocks, as shown in Fig. 3.21 (page 11). Each block may then be DPCM coded.

3.2.9.2 Transform/VQ Coding

As with Transform/DPCM coding, Transform/VQ has many variants. In a manner analogous to the 1-D Transform/DPCM coding described in section 3.2.9.1, Transform/VQ coding may be implemented by performing a 1-D DCT on the rows of an image, and them VQ along the columns of the resulting coefficients (see Fig. 3.18) [19 p114].

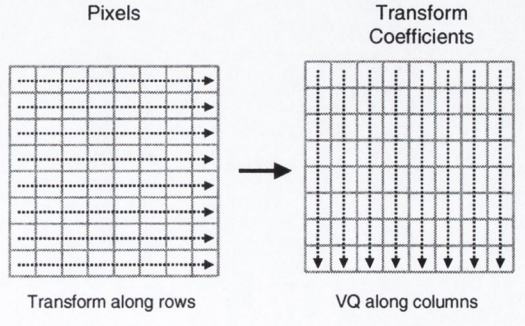


Fig. 3.18 1-D Transform/VQ hybrid coding

Another approach is to simply apply VQ to the blocks of transform coefficients following a 2-D DCT. A block may be partitioned into regions of similar variance and then these regions independently VQ coded [19 p113]. The codebook for each region, in this case, would consist of typical coefficient value distributions rather than pixel intensity distributions (as in the spatial domain).

3.3 The Discrete Karhunen-Loève Transform (KLT)

The most suitable transform for image coding should ideally produce completely uncorrelated coefficients and compact the maximum signal energy into the fewest transform coefficients [40][43 p5][10 p379]. The first property is desirable to avoid representing the same information more than once [8 p165] and thus justify scalar quantization [10 p379], while the second allows many coefficients to be discarded without seriously degrading the reconstructed image quality. The Karhunen-Loève[38] transform (KLT), first discussed by Karhunen [33] and later by Loève [34], is an optimal transform in terms of the signal decorrelation and energy compaction achieved in the transform domain [9 p644][40][43 p5].

3.3.1 Formulating the KLT

The KLT[39] treats an image as a realization of a homogeneous random field. We seek the best representation of a random function in the mean squared error sense (MSE). Let us consider a zero mean random vector $\mathbf{x} \in \Re^N$, $\mathbf{x} = \{x_0, x_1, ..., x_{N-1}\}^T$. If $\{\boldsymbol{\varphi}_i\}$ is a set of linearly independent vectors spanning the N-dimensional vector space (i.e. they form a basis for the vector space), then $\mathbf{x}$ may be represented in terms of a linear combination of the elements of $\{\boldsymbol{\varphi}_i\}$ (see section 3.2.2, equation (3.5))

$$\mathbf{x} = \sum_{i=0}^{N-1} a_i \boldsymbol{\varphi}_i \qquad (3.32)$$

[38] We refer to the *discrete-time Karhunen-Loève transform*, also known as the *Hotelling transform[43 p373]*, *eigenvector transform* or *principal component analysis (PCA)*.

[39] The following discussion is based largely on [15 p148] and [20 p28].

where a_i are the coefficients of the expansion $a_i = \langle \mathbf{x}, \boldsymbol{\varphi}_i \rangle / \langle \boldsymbol{\varphi}_i, \boldsymbol{\varphi}_i \rangle \quad i = 0,1,...,N-1$, and $\langle \cdot, \cdot \rangle$ denotes the *inner product*. The vector $\mathbf{x}$ may be represented by the N numbers x_i or alternatively as the N numbers a_i in the space spanned by the basis vectors $\{\boldsymbol{\varphi}_i\}$.

If the vector space spanned by $\{\boldsymbol{\varphi}_i\}$ is chosen, and the basis vectors ordered, such that only the first D coefficients a_i are significantly different to zero then the vector $\mathbf{x}$ may be approximated by only these terms. Let the approximated $\mathbf{x}$ be denoted by $\tilde{\mathbf{x}}$. We seek the set of basis vectors $\{\boldsymbol{\varphi}_i\}$ that minimises the difference between $\mathbf{x}$ and $\tilde{\mathbf{x}}$ in the MSE sense.

3.3.2 Finding the basis vectors

Firstly, we express the truncated representation (i.e. only the first D weighted vectors) of $\mathbf{x}$ by

$$\tilde{\mathbf{x}} = \sum_{i=0}^{D-1} a_i \boldsymbol{\varphi}_i \tag{3.33}$$

The MSE of the truncation is given by

$$\varepsilon = E\left\{(\mathbf{x} - \tilde{\mathbf{x}})^2\right\} \tag{3.34}$$

(effectively the contribution of the last N-D terms)

$$\varepsilon = E\left\{\left\langle \sum_{i=D}^{N-1} a_i \boldsymbol{\varphi}_i, \sum_{i=D}^{N-1} a_i \boldsymbol{\varphi}_i \right\rangle\right\} \tag{3.35}$$

It should be noted that by making D=N we get perfect reconstruction of $\mathbf{x}$ from the linear combination of the terms $(a_i \boldsymbol{\varphi}_i)$, $i = 0,1,...,D-1$.

Assuming $\{\boldsymbol{\varphi}_i\}$ forms an orthonormal basis for the vector space we can write $\langle \boldsymbol{\varphi}_i, \boldsymbol{\varphi}_k \rangle = \delta_{ik}$ where δ_{ik} is the Kronecker† delta. Now

$$\varepsilon = E\left\{\sum_{i=D}^{N-1}|a_i|^2\right\} \tag{3.36}$$

$$= E\left\{\sum_{i=D}^{N-1}\left|\langle \mathbf{x}, \boldsymbol{\varphi}_i \rangle\right|^2\right\} \tag{3.37}$$

$$= E\left\{\sum_{i=D}^{N-1}\boldsymbol{\varphi}_i^T \mathbf{x} \cdot \mathbf{x}^T \boldsymbol{\varphi}_i\right\} \tag{3.38}$$

$$= \sum_{i=D}^{N-1}\boldsymbol{\varphi}_i^T E\left\{\mathbf{x} \cdot \mathbf{x}^T\right\}\boldsymbol{\varphi}_i \tag{3.39}$$

Recalling the definition of the auto-covariance matrix of the zero mean random variable $\mathbf{x}$ as [15 p149]

$$\mathbf{A}_x = E\left\{\mathbf{x} \cdot \mathbf{x}^T\right\} \tag{3.40}$$

we can substitute $\mathbf{A}_x$ into equation (3.39) and write

$$\varepsilon = \sum_{i=D}^{N-1}\boldsymbol{\varphi}_i^T \mathbf{A}_x \boldsymbol{\varphi}_i \tag{3.41}$$

To minimize the MSE between $\mathbf{x}$ and $\tilde{\mathbf{x}}$ we must find a suitable set of basis vectors $\{\boldsymbol{\varphi}_i\}$, ordered such that the expression in equation (3.41) is minimised. The covariance matrix $\mathbf{A}_x$, is a real, Toeplitz†, symmetric and positive definite[40] and thus finding a set of n orthogonal eigenvectors always is possible [35][15 p150][45 p104]. Minimising the value of equation (3.41) subject to the orthonormality condition, we get [20 p29]

$$\left(\frac{\delta}{\delta \cdot \boldsymbol{\varphi}_i}\right)\left(\varepsilon - \lambda_i\langle \boldsymbol{\varphi}_i, \boldsymbol{\varphi}_i \rangle\right) = 0 \tag{3.42}$$

[40] A symmetric matrix $\mathbf{A}$ is positive definite if the quadratic form $\mathbf{x}^T\mathbf{A}\mathbf{x}$ is positive for all nonzero vectors $\mathbf{x}$. This is equivalent to the condition that all the eigenvalues of $\mathbf{A}$ are positive [39].

where the Lagrange multiplier λ_i has been introduced for the orthonormality constraint. We may proceed to [20 p29][45 p105]

$$(\mathbf{A}_x - \lambda_i \cdot \mathrm{I}_N)\boldsymbol{\varphi}_i = 0 \text{ for } i = 0,1,\ldots,N-1 \tag{3.43}$$

where I_N is the identity matrix. Clearly this is in the form of the *eigenvalue problem*. The set of eigenvectors obtained $\{\boldsymbol{\varphi}_i\}$, will be the basis vectors of the transform and they will diagonalise the auto-covariance matrix $\mathbf{A}$ [20 p30].

Let $\mathbf{P}$ be the matrix of orthonormal eigenvectors vectors $\boldsymbol{\varphi}_i$ given by

$$\mathbf{P} = [\boldsymbol{\varphi}_0, \boldsymbol{\varphi}_1, \ldots, \boldsymbol{\varphi}_{N-1}].$$

The covariance matrix $\mathbf{A}_x$, may be diagonalised as $\mathbf{A}_x = \mathbf{P}^T \boldsymbol{\Lambda} \mathbf{P}$ where $\boldsymbol{\Lambda}$ is a diagonal matrix of the corresponding eigenvalues $\{\lambda_i\}$. Thus, the orthogonal basis functions of the KLT are obtained from the statistics of the source image, as the eigenvectors of the corresponding auto-covariance matrix.

The KLT of a vector $\mathbf{x}$ to a vector $\mathbf{y}$ is given by $\mathbf{y} = \mathbf{P}\mathbf{x}$, and its inverse by $\mathbf{x} = \mathrm{P}^T \mathbf{y}$. For a non-zero mean source, we need to factor in the *mean vector*[†] $\mathbf{m}_x$, of the vector population. The covariance matrix will then become $\mathbf{A}_x = E\{(\mathbf{x} - \mathbf{m}_x) \cdot (\mathbf{x} - \mathbf{m}_x)^T\}$ and the transform represented by $\mathbf{y} = \mathrm{P}(\mathbf{x} - \mathbf{m}_x)$. The mean of the $\mathbf{y}$ vectors resulting from this transform will be zero.

3.3.3 Ordering the basis vectors

We seek to approximate the reconstruction of $\mathbf{x}$ from $\mathbf{y}$ (and the basis functions) by

$$\tilde{\mathbf{x}} = \mathbf{P}^T \tilde{\mathbf{y}} \tag{3.44}$$

where $\tilde{\mathbf{y}}_i = \mathbf{y}_i$ for $i = 0,1,\ldots,D-1$ zero otherwise , i.e. the last $N-D$ transform coefficients are discarded. The ordering of the basis vectors becomes important for the

minimisation of the MSE. To find a suitable arrangement for the basis vectors we examine $\mathbf{A}_x = \mathbf{P}^T \mathbf{\Lambda} \mathbf{P}$ and note that

$$\mathbf{P}^T \mathbf{A}_x \mathbf{P} = \mathbf{\Lambda} = \begin{bmatrix} \lambda_1 & & 0 \\ & \ddots & \\ 0 & & \lambda_N \end{bmatrix} \tag{3.45}$$

We consider this expression in conjunction with equation (3.41) and see that the MSE will be equal to the sum of the eigenvalues of the remaining (*N-D*) eigenvectors [20 p30, p122]

$$\varepsilon = \sum_{i=D}^{N-1} \lambda_i \tag{3.46}$$

which is minimised by ranking the eigenvalues λ_i, in descending order so that $\lambda_i \geq \lambda_{i+1}$ for $i = 0,1,...,N-1$. The eigenvectors must be arranged correspondingly. As a result of this ordering the first transform coefficient will carry the largest part of the signal energy with each successive coefficient making a non-increasing contribution.

3.3.4 Properties of the KLT

Some important properties of the KLT follow.

- It completely decorrelates the signal in the transform domain [20 p30].
- It minimizes the MSE for data compression [43 p374].
- It contains the most energy in the fewest transform coefficients [43 p374].

While the KLT is interesting theoretically it has serious practical limitations. To begin with the basis functions used in the KLT are derived from the auto-covariance matrix of the source image. As different types of images are likely to have different covariance functions, a new set of basis functions must be generated for each image i.e. the KLT is signal dependent. Furthermore, the KLT assumes the covariance function of the source image is known, where in fact this is not the case, it must be estimated. If the estimation is not accurate then the optimal theoretical properties of the transform will no longer hold.

Another practical limitation of the KLT is that given an auto-covariance matrix for an image, the solution to equation (3.43) is generally not straightforward. The KLT is therefore an ideal but impractical transform for image compression [20 p31]. In summary

- The basis functions of the transform are signal dependent and so cannot be predetermined but must be computed and transmitted for every image.

- There are no fast algorithms for the determination of the basis vectors (diagonalisation of the auto-covariance matrix). It is a computationally complex transform requiring in the order of N^2 operations [43 p374].

- Perfect decorrelation is generally not possible as natural images can rarely be modelled as realizations of homogeneous random fields.

Although the KLT is usually not practical for image compression in real world applications, it does serve as a useful benchmark for comparison with other linear transforms. The principal failing of the KLT, but also its primary strength, lies in the dependence of the basis functions on the source image. It is reasonable to speculate on the existence of a *fixed set* of image independent basis functions, that while being sufficiently general could still approach the efficiency of the KLT. Indeed many discrete transforms have been investigated for image compression that employ predetermined basis functions [36]. These include the discrete Fourier transform (DFT), discrete cosine transform (DCT). Hadamard, and Harr transforms [37]. Of these, and numerous others, the DCT has emerged as the most widely adopted transform for image and video compression, employed by all the predominant coding standards (JPEG, MPEG-1, MPEG-2, H.263).

3.4 The Discrete Cosine Transform (DCT)

3.4.1 The DCT from the KLT

Like the majority of linear transforms used for image coding, the discrete cosine transform (DCT) maps pixels in the spatial domain to coefficients in the frequency domain. The DCT was developed in 1974 by Ahmed, Natarajan and Rao [22], as an approximation of

the KLT for a first-order Gauss-Markov[41] process (Markov-1) with a large positive adjacent (inter-sample) correlation coefficient $\rho(\rho \to 1)$. In this case, the auto-covariance matrix of the source is a symmetric Toeplitz matrix of the form[42] [43 p375][19 p54]

$$\mathbf{A}_x = \begin{bmatrix} 1 & \rho & \rho^2 & \cdots \\ \rho & 1 & \rho & \cdots \\ \rho^2 & \rho & 1 & \cdots \\ \vdots & \vdots & \vdots & \ddots \end{bmatrix} \tag{3.47}$$

The first-order Markov process is one of the few cases where, given the auto-covariance matrix, an analytical solution to equation (3.43) is available [20 p30]. Ray and Driver [46] provided a solution for the discrete case. The eigenvectors of $\mathbf{A}_x$, which form the basis of the KLT, were found to be [20 p30]

$$\phi_{mn} = \boldsymbol{\varphi}_m(n) = \sqrt{\frac{2}{N+\lambda_m}} \cdot \sin\left(w_m \left[n+1-\frac{N+1}{2} \right] + \frac{\pi(m+1)}{2} \right) \tag{3.48}$$

$$m,n = 0,1,\ldots,N-1$$

where $\boldsymbol{\varphi}_m(n)$ denotes the nth component of the mth eigenvector. Arranging these eigenvectors in an $N \times N$ matrix $\mathbf{G}$, as in illustrated in (3.18), the term ϕ_{mn} denotes the entry at row m and column n. The terms λ_m are the eigenvalues

(3.49)

[41] A first-order Markov process is generated by exciting a first order recursive filter with a zero-mean, stationary white noise process [45 p109].

[42] Assuming unit variance and zero mean

$$\lambda_m = \frac{1-\rho^2}{1-2\rho\cos(w_m)+\rho^2}$$

and the terms w_m are the real positive roots of the transcendental† equation

(3.50)

$$\tan(Nw) = -\frac{(1-\rho^2)\sin(w)}{\cos(w)-2\rho+\rho^2\cos(w)}.$$

We can see that as $\rho \to 1$, equation (3.50) becomes

(3.51)

$$\tan(Nw) = 0$$

giving

(3.52)

$$w_k = \frac{k\pi}{N}, \quad \text{for } k = 0,1,\ldots,N-1.$$

The eigenvalues λ_m vanish when w_m are non-zero. In fact the only non-zero eigenvalue occurs for $m=0$ where $\lambda_0 = N$ [20 p37], all other eigenvalues tend to zero as $\rho \to 1$, giving

(3.53)

$$\phi_{0n} = \sqrt{\frac{2}{N+N}} \cdot \sin\left(\frac{\pi}{2}\right) = \frac{1}{\sqrt{N}}$$

and

(3.54)

$$\phi_{mn} = \sqrt{\frac{2}{N}} \cdot \sin\left(\frac{m\left(n+\frac{1}{2}\right)\pi}{N} + \frac{\pi}{2}\right)$$

(3.55)

$$= \sqrt{\frac{2}{N}} \cdot \cos\left(\frac{m\left(n+\frac{1}{2}\right)\pi}{N} \right) \quad m \neq 0 .$$

Equation (3.55) may be generalised by introducing $\alpha(m)$, defined as

(3.56)

$$\alpha(m) \begin{cases} \frac{1}{\sqrt{2}}, & m = 0 \\ 1, & m \neq 0 \end{cases}$$

and writing

(3.57)

$$\phi_{mn} = \sqrt{\frac{2}{N}} \cdot \alpha(m) \cos\left(\frac{m\left(n+\frac{1}{2}\right)\pi}{N} \right).$$

Equation (3.57) defines the basis vectors of the so called *type two* DCT, or DCT-II. Three other variants of the DCT have been defined [56], but we shall confine our discussions to the DCT-II in this thesis, as this is the form employed in all the major standards [45 p95] (see [20][45] for others DCT forms). The basis vectors $\boldsymbol{\varphi}_m$, arranged as the rows of the forward transform kernel $\mathbf{G}$, allow the DCT-II to be computed for a one-dimensional source, as in (3.17), by

(3.58)

$$\mathbf{y} = \mathbf{G}\mathbf{x} .$$

The DCT-II is a real unitary transform [20], thus the inverse transform kernel $\mathbf{G}^{-1}$, is obtained by simply transposing $\mathbf{G}$, $\mathbf{G}^{-1} = \mathbf{G}^{T}$. A numerical evaluation of the one dimensional DCT-II basis functions for $N=8$ is tabulated in Table 3.1, and their plots presented in Fig. 3.19.

Departing from matrix notation momentarily, we present an equation for the DCT, expressed as a series summation (3.59). The operations are identical to that of equation (

3.58), only the representation has changed. Here we use $C^{II}(u)$ to denote the sequence of DCT-II transform coefficients, and $\alpha(u)$ is defined as in (3.56).

DCT-II :

$$C^{II}(u) = \sqrt{\frac{2}{N}}\alpha(u)\sum_{n=0}^{N-1} x(n)\cos\left(\frac{(2n+1)u\pi}{2N}\right), \quad u = 0,1,...,N-1 \tag{3.59}$$

and its inverse

IDCT-II:

$$x(n) = \sqrt{\frac{2}{N}}\sum_{u=0}^{N-1}\alpha(u)C^{II}(u)\cos\left(\frac{(2n+1)u\pi}{2N}\right), \quad n = 0,1,...,N-1. \tag{3.60}$$

Fig. 3.19 DCT-II Basis functions for $N=8$

Table 3.1 DCT-II basis functions (rows) for $N=8$. See Fig. 3.19 for plot.

	n=0	n=1	n=2	n=3	n=4	n=5	n=6	n=7
m=0	0.3536	0.3536	0.3536	0.3536	0.3536	0.3536	0.3536	0.3536
m=1	0.4904	0.4157	0.2778	0.0975	-0.0975	-0.2778	-0.4157	-0.4904
m=2	0.4619	0.1913	-0.1913	-0.4619	-0.4619	-0.1913	0.1913	0.4619
m=3	0.4157	-0.0975	-0.4904	-0.2778	0.2778	0.4904	0.0975	-0.4157
m=4	0.3536	-0.3536	-0.3536	0.3536	0.3536	-0.3536	-0.3536	0.3536
m=5	0.2778	-0.4904	0.0975	0.4157	-0.4157	-0.0975	0.4904	-0.2778
m=6	0.1913	-0.4619	0.4619	-0.1913	-0.1913	0.4619	-0.4619	0.1913
m=7	0.0975	-0.2778	0.4157	-0.4904	0.4904	-0.4157	0.2778	-0.0975

3.4.2 The Two-dimensional DCT

The separability property of the DCT [20] facilitates simple extension of the transform to two dimensions, based on a series of one-dimensional transforms (see section 3.2.3.1). No modification of the 1-D transform kernel is required. If $\mathbf{H}$ represents a one dimensional DCT transform kernel then the DCT of an image matrix $\mathbf{X}$ can be found by[43]

$$\mathbf{Y} = \mathbf{H}\mathbf{X}\mathbf{H}^T. \tag{3.61}$$

While the kernel $\mathbf{H}$ that implements the 2-D DCT is in fact composed of 1-D basis vectors (Fig. 3.19), it should be understood that net effect of (3.61) is a mapping to a 2-D basis. Each coefficient produced by the transform will represent the weighting factor of the respective 2-D basis vector, or basis image, as shown in Fig. 3.20 (for the 8×8 case). Thus, an 8×8 DCT will generate 64 transform coefficients, each representing the weighting of an 8×8 basis vector.

The graphical representation of the 2-D DCT basis vectors, presented in Fig. 3.20, illustrates the "frequency" based ordering described in section 3.2.3.3. The first row represents only horizontal frequencies while the first column represents only vertical frequencies. The remaining basis vectors represent different combinations of both horizontal and vertical frequencies.

[43] We assume that all the matrices are of equal rank.

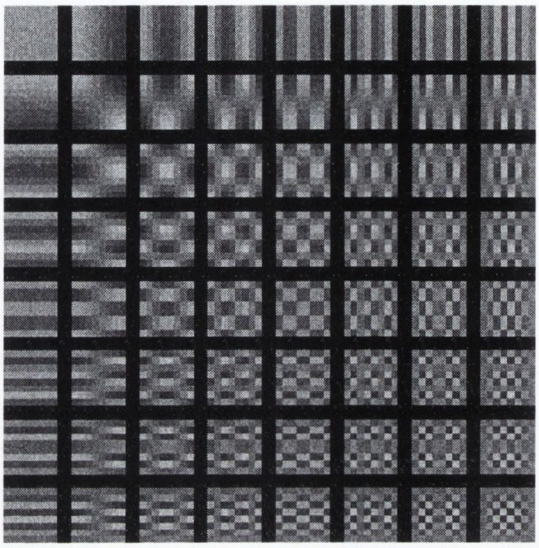


Fig. 3.20 Basis images (eigen images) of the 2D 8×8 DCT

The 1D DCT defined in (3.59) may be extended to two dimensions as follows:

2D DCT-II :

$$C^{II}(u,v)=\sqrt{\frac{2}{NM}}\alpha(u)\alpha(v)\sum_{n=0}^{N-1}\sum_{m=0}^{M-1}x(n,m)\cos\left(\frac{(2n+1)u\pi}{2N}\right)\cos\left(\frac{(2m+1)v\pi}{2M}\right) \tag{3.62}$$

$$u=0,1,...,N-1, \qquad v=0,1,...,M-1.$$

and its inverse

2D IDCT-II:

$$x(n,m)=\sqrt{\frac{2}{NM}}\sum_{u=0}^{N-1}\sum_{v=0}^{M-1}\alpha(u)\alpha(v)C^{II}(u,v)\cos\left(\frac{(2n+1)u\pi}{2N}\right)\cos\left(\frac{(2m+1)v\pi}{2M}\right) \tag{3.63}$$

$$n=0,1,...,N-1, \qquad m=0,1,...,M-1.$$

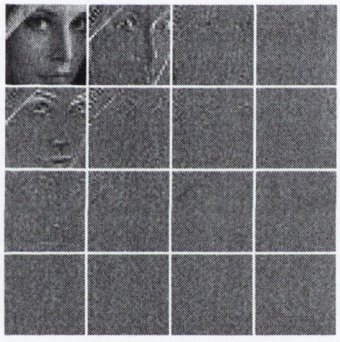
where $\alpha(k)$ is defined as in (3.56).

3.4.3 Properties of the DCT

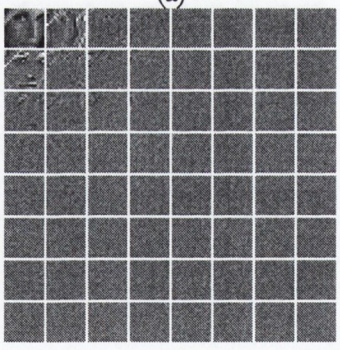
3.4.3.1 Energy Packing

In section 3.2.3.3 we stated that *energy packing* is a desirable property of linear transforms for image compression. Yip and Rao [20][60] have evaluated the performance of the DCT and other discrete transforms in this respect, and have demonstrated the excellent energy packing properties of the DCT. In Fig. 3.21 we have illustrated the energy compaction resulting from a 4×4 and an 8×8 DCT on a section of the "Lena" image. Note that most of the energy is compacted in the top left corner of the grid (DC), with the DC coefficient effectively representing a low pass filtered and decimated[†] version of the original image.

(a)



(b)



(c)

Fig. 3.21 Energy packing in the DCT. Original image (a), 4×4 (b) and 8×8 (c) block based DCT, with coefficients of the same basis vectors arranged together in sub-images which are then ordered corresponding to their basis vectors. Coefficient values have been scaled and translated, for display purposes, to range within the interval 0 to 255. DC and AC components scaled independently, with 128 representing zero for the AC coefficients.

3.4.3.2 Fast Recursive Algorithms

As stated in section 2.2, the computational requirements[44] of an image coding scheme is often an important practical concern when determining its usefulness for a given task. The

[44] By *requirements* we mean complexity, number of operations, types of operations, and memory requirements.

requirements for the encoder and decoder may often be considered independently. For example, an image compression scheme intended for "web page graphics" would require a low complexity decoder for fast rendition on a variety of computers. The encoder of such a scheme, by contrast, could be considerably more complex. The opposite case may also be true. A low power mobile robot, sending images to a remote base station, would benefit from a low complexity encoder, while the base station decoder, with more computing resources, could be more complex. The necessity for fast encoder/decoder algorithms is demonstrated further in real-time video applications, where delays between encoding and decoding must be minimised.

For the DCT, both the forward and inverse transforms require theoretically the same number of mathematical operations, although clearly the decoder need not complete calculations for zero valued coefficients (of which there may be many). Considering the 1-D case, equation (3.59) shows that an N-point DCT requires approximately N^2 multiplications and N^2 additions. This however, is a "brute force" method for computing the DCT. A number of algorithms have been developed for the N-point DCT, many of them based on the fast Fourier transform, that reduce the number of arithmetic operations to approximately $N \log_2 N$ [57][58][59][74]. One of the primary advantages of the DCT for image and video coding is the existence of these fast algorithms.

3.4.3.3 Progressive Image Transmission & Scalability

Progressive image transmission is the transmission of an image in such a way that the whole image may be reconstructed in stages, with each successive "chunk" of data adding further clarity to the image. Progressive transmission is useful in a number of low bandwidth circumstances where an initial, "rough" idea of an image is transmitted first, followed by a succession of refinements. A user browsing images (e.g. internet, image database) may choose to terminate image retrieval mid-transmission, based on the rough approximation.

The DCT lends itself naturally to progressive transmission, as do subband coding schemes in general [19 p177]. In one approach, known as *spectral selection*, the coefficient matrix is partitioned into groups, usually on the basis of their variances [19 p145], and the

coefficients form each block, lying within these groups, are then transmitted together[45] as "stages" (usually from low frequencies to high frequencies) [19 p83] (Fig. 3.22).

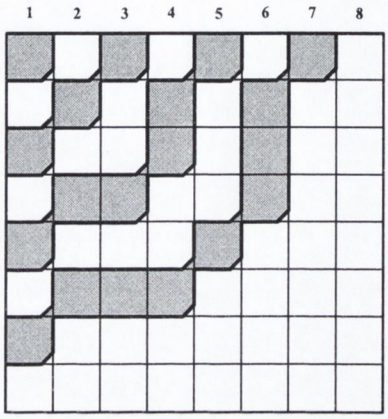


Fig. 3.22 Seven-stage spectral selection for progressive transmission [19 p145].

Alternatively, in a method known as *successive approximation*, the individual bit planes of all the coefficients together are transmitted as stages. Combination of the two techniques can deliver a good approximation of an image during the early stages of transmission [19 p83]. The popular JPEG [50] image coding standard includes support for progressive transmission.

The concept of *scalable coding* is quite similar to that of progressive image transmission. An image represented in a scalable format may be decoded to different levels of quality/clarity depending of the requirements of the application, or nature of the platform on which the decoder resides. The DCT is also well suited to applications requiring scalable representations. (§Fig. 3.23). We can say that the DCT lends itself to scalability because a good approximation of the "whole" image may be generated from only a subset of transform coefficients.

[45] This is equivalent to transmitting subbands separately in subband coding (§3.5).

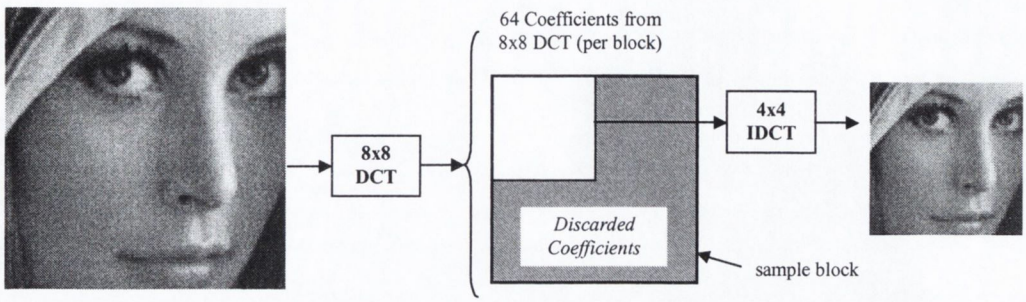


Fig. 3.23 Example of an original image coded using an 8×8 DCT, then reconstructed at half size by application of a 4×4 IDCT on only the first 4 rows and columns of coefficients.

Fig. 3.23 illustrates the potential offered by the DCT for spatial scalability. In this case, an image has been reconstructed at half size by performing a 4×4 IDCT on only a quarter of the coefficients. Such an operation would be useful to cater for the display characteristics of a lower resolution display device.

The topics of progressive transmission and scalability introduce the notion of multiple spatial resolutions within a single image representation. In the next section we develop this concept of *multiresolution* coding further

3.5 Subband & Wavelet Coding

In section 3.1.1 we alluded to the fact that transform coding may be viewed as a special case of *subband coding* (SBC). In this section we briefly describe the principles of subband coding and its relationship to transform coding. We also take a look at a *multiresolution* instance of subband coding known as *wavelet coding*. All three methods are closely related as they decompose a source image into different spatial frequency[46] bands (*subbands*), and then code each band according to the statistics and subjective importance of the information it conveys.

[46] The term frequency refers to *spatial frequency* unless otherwise indicated.

3.5.1 Introduction to Subband Coding

Subband coding was first applied to image compression by Vetterli [66], Woods, and O'Neil [67]. In one-dimensional subband coding, a signal is ideally decomposed into relatively narrow non-overlapping spatial frequency bands (Fig. 3.24). These *subbands*[47] are then decimated† and coded with a coder and bit-rate accurately matched to the statistics and subjective importance of each particular subband [11 p266, p91][10 p411][72 p31]. The frequency partitioning is generally computed by convolving the input image with a set of bandpass filters [65 p144]. The original image is reconstructed by upsampling, filtering, and summating the subband signals.

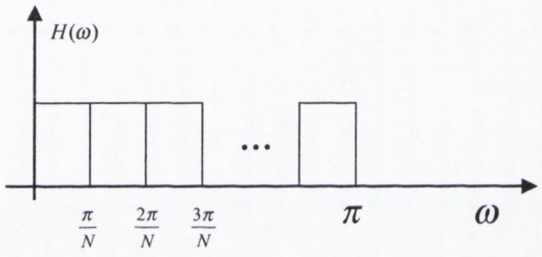


Fig. 3.24 Ideal "brick wall" frequency responses of the analysis filters, with uniform partitioning.

The set of bandpass filters that implement the signal decomposition, are collectively referred to as an *analysis filter bank* while the set of reconstruction filters is called a *synthesis filter bank*[48].

3.5.2 Two-channel Filter Banks

The most simple form of subband decomposition partitions the frequency spectrum into two uniform subbands; the low band and the high band. If we let the normalised sampling frequency $f_s = 1$, then the highest possible spatial frequency present in the digital signal will be $\frac{1}{2} f_s$, or π radians (assuming band limited digitization, and recalling the Nyquist sampling theorem). Equal partitioning of the frequency spectrum will therefore be centred about $\frac{1}{4} f_s$. In practice, ideal bandpass filterbanks cannot be realised, so must be approximated by filterbanks with overlapping pass bands (Fig. 3.25) [10 p411]. This

[47] Subband signals are equivalent to *transform coefficients* in the language of transform coding [8 p187].

[48] The analysis and synthesis filter banks of a subband coding scheme are equivalent to the forward and reverse transform kernels of a transform coding scheme respectively.

overlap causes *aliasing*[49] when each subband is decimated and so strategies must be employed to minimise or eliminate this distortion. *Quadrature mirror filters* (QMFs) have been developed [68][69], exhibiting symmetry about $\frac{1}{4}f_s$ that avoid aliasing in the reconstructed image. This is achieved by designing analysis-synthesis filters such that the aliasing introduced by the analysis section is cancelled out by the synthesis section [10 p411][11 p91].

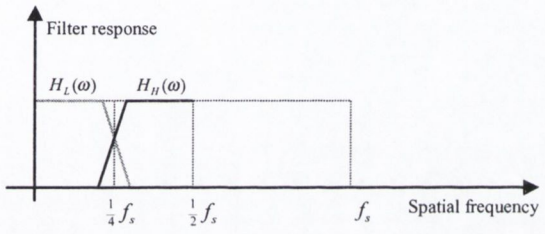


Fig. 3.25 Two-channel filter bank with analysis filters $H_L(\omega)$ and $H_H(\omega)$, splitting the frequency spectrum into high and low bands respectively.

A two-channel filter bank is depicted in Fig. 3.26. Note that by partitioning the frequency spectrum into two equal sized bands, the bandwidth in each subband is effectively halved. Both channels are thus subsampled by a factor of two prior to quantization, causing the total number of subband signals to remain equal to the number of original samples (quantization not indicated in Fig. 3.26).

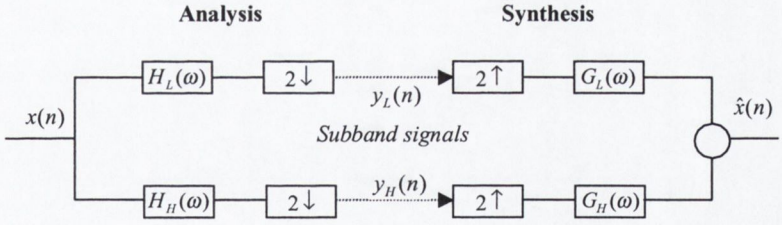


Fig. 3.26 Block diagram of a two-channel analysis-synthesis filter bank system.

The simple two-channel scheme may easily be extended to multiple channels (*multi-channel filter banks*) . The direct approach is simply to partition the frequency spectrum into the desired number of subbands. For a critically[50] sampled *N*-channel filter bank, the

[49] See [73 p303] for an excellent description of signal sampling and aliasing.

[50] For a non-critically sampled filter bank, the subsampling factor will be greater than the number of analysis filters. A non-critically sampled filter banks is equivalent to having an overcomplete basis [43 p171].

subsampling factor applied to each filter output will be equal to the number of analysis filters (Fig. 3.27).

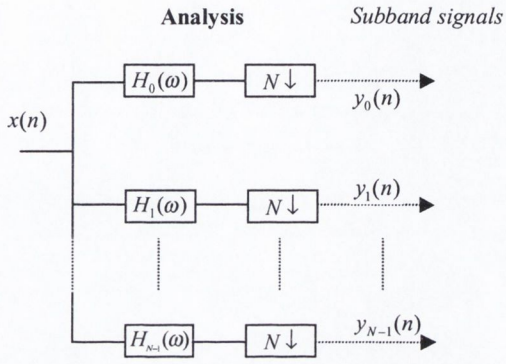


Fig. 3.27 N-channel analysis filter bank with critical subsampling by N.

Another way to construct a multi-channel filter bank is to simply cascade a number of two-channel filter banks [43 p142] as shown in Fig. 3.28.

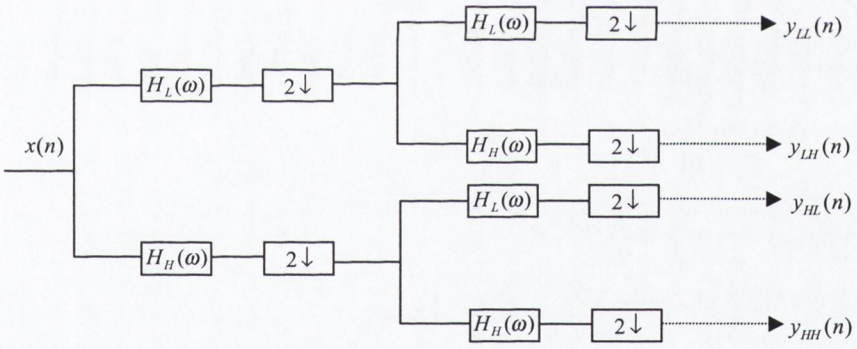


Fig. 3.28 Multi-channel filter bank implemented by cascading two-channel filter banks.

If the cascading is applied to all the intermediate filter outputs (as shown in Fig. 3.28), then the system is called a *uniform* cascade system, otherwise it is referred to as a *non- uniform* or *pyramid* cascade [11 p59][65 p150]. The full potential of subband coding may be exploited when non-uniform band splitting is used [40]. This is the topic of the next section.

3.5.3 Multiresolution Decomposition

A *multiresolution* decomposition is a subband coding scheme in which non-uniform (unequal) band splitting is employed. One of the most useful multiresolution

decompositions for image compression is known as a *octave-band* decomposition. In an octave-band decomposition, only the lowpass band of a cascaded two channel filter bank is recursively split until a desired depth or *level* has been reached (Fig. 3.29). Subband coders with critically sampled octave-band decomposition are also known as *discrete wavelet transform* (DWT) coders, or *wavelet* coders [40][71].

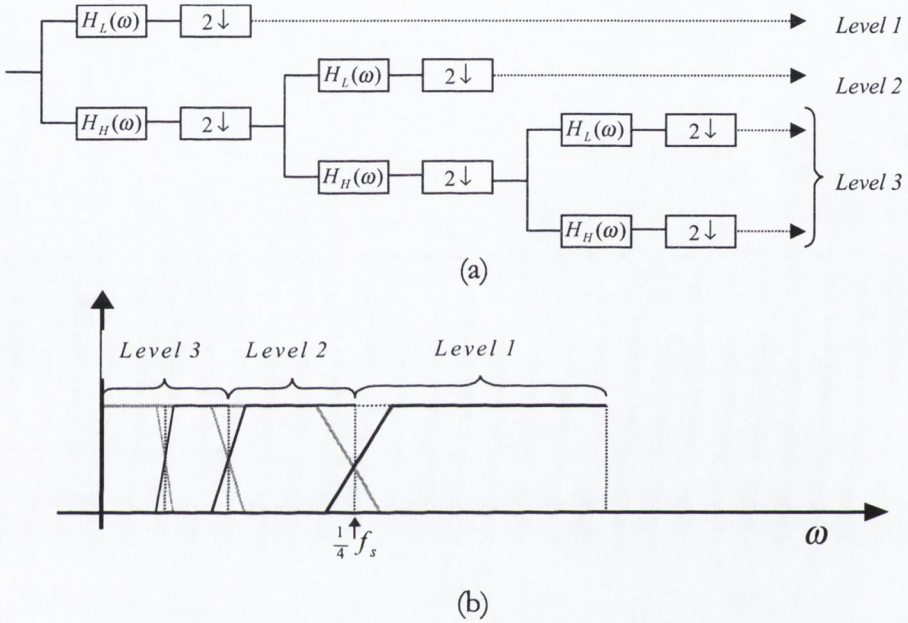


Fig. 3.29 Octave-band decomposition (a) and resulting partitioning of the frequency spectrum (b). In this case we have 4 subbands and 3 levels.

In an octave-band configuration, the logarithmic partitioning of the frequency spectrum, reflects both the fall off of the *power spectral density* of typical images, and the frequency sensitivity of the HVS [43 p5][71]. It is therefore advantageous for image compression from both a perceptual and MSE standpoint. The coding gains that may be achieved over the DCT by the DWT depend very much on the whole codec architecture and not just the use of the transform itself. In [119], a recent comparative study, the DWT was shown to outperform the DCT typically by the order of about 1db in peek signal to noise ratio. The coding gains resulting from the use of the DWT come at the cost of increased computational complexity however [42 p95][119]. A 2-D 8x8 DCT can be computed using only 54 multiplications whereas the computational requirements of the DWT depend on the length of the wavelet filters, which is at least one multiplication per coefficient [119].

The one-dimensional scheme presented in Fig. 3.29 may be extended to two dimensions for image coding. The resulting subbands are indicated in Fig. 3.30 for a level 3 decomposition on a section of the *Lena* image.

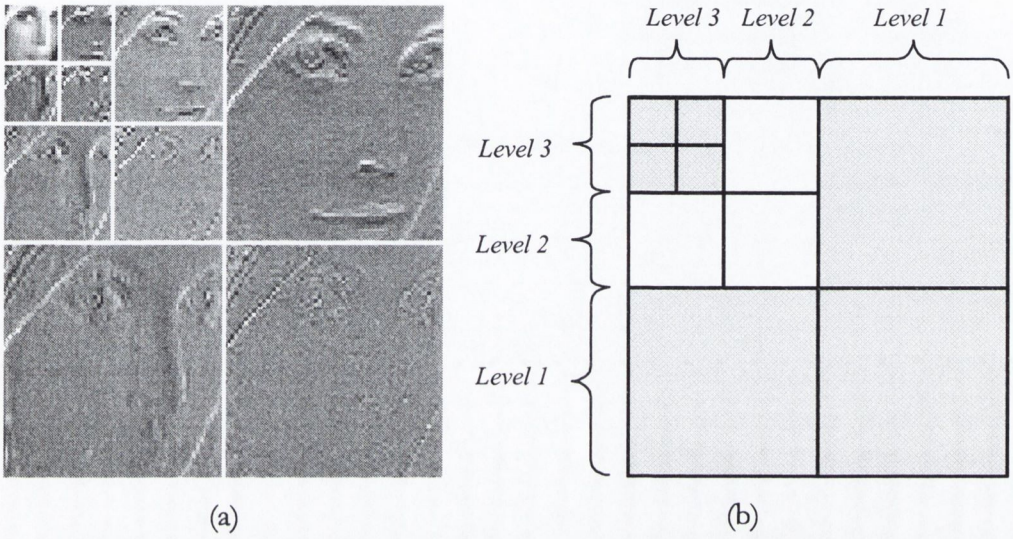


Fig. 3.30 2-D Wavelet decomposition with 3 levels (a). Spatial frequencies increase from left to right and top to bottom (b).

3.5.4 Pyramid Coding

Another approach to multiresolution coding, known as *pyramid coding*, was proposed by Burt and Adelson in 1983 [70]. Pyramid coding schemes are conceptually very close to octave-band decompositions. The term *pyramid* essentially implies the representation of image data at a hierarchy of resolutions. One such representation is known as a *Laplacian pyramid*. An image is represented in a Laplacian pyramid as follows. The source image is first low pass filtered and subsampled to produce a "low resolution" version of the original image. From this image, a "coarse approximation" of the original is generated by upsampling with interpolation to the original image dimensions. This approximation (or prediction image) is then subtracted from the original image to generate a difference image (i.e. error image) (Fig. 3.31). The process may be applied recursively, generating a hierarchy, or pyramid, of difference images at each resolution. The final representation will consist of a small image at the top of the pyramid, with a number of difference images at successively greater resolutions.

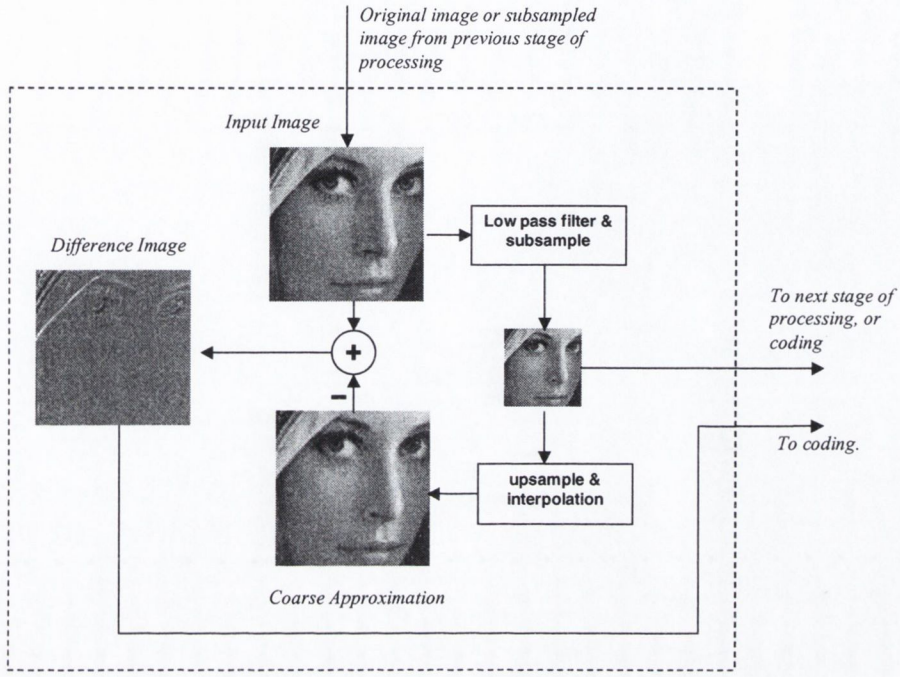


Fig. 3.31 Pyramid decomposition of an image.

The general[51] pyramid scheme produces an overcomplete (§3.2.2) representation [43 p94][72 p36], which would seem to diminish its potential for image compression. If however, the system is linear and the low pass filter is orthogonal to its even translates, then the difference image may be down sampled after filtering. In this case, the pyramid decomposition becomes equivalent to a critically downsampled orthogonal subband coding scheme [43 p174].

3.5.5 Relationship to Transform Coding

BTC may be viewed as a special case of subband coding [8][40][71]. An $N \times N$ transform kernel is equivalent to an N-channel filter bank, with filters of length N, and critical subsampling by a factor of N [43 p157]. Fig. 3.21 (page 11) presents a subband interpretation of the DCT, in which similar subbands/transform coefficients have been grouped together. Subband coding schemes in general, including transform and wavelet coding, all achieve compression through the same basic technique: the distribution of bits to different subbands/transform coefficients based on their relative importance [8 p2]. Thus, the advantages of one frequency decomposition over another is often viewed in terms of the opportunities offered for subsequent subband/coefficient treatments.

[51] Where non-linear decimation and interpolation operators are used.

3.6 Summary & Conclusions

This chapter has presented an overview of the principles and techniques of transform coding. The theory behind signal decomposition for image compression was discussed, and the properties of image transforms that are attractive for image coding were described. The optimal KLT and *near* optimal, but more practical, DCT have been presented, and their advantages/disadvantages for image compression discussed. The block based architecture that distinguishes transform coding from subband coding was discussed, and the implications in terms of computational efficiency and distortion discussed. In summary, we note the block based approach yields computational performance benefits, but introduces annoying distortions at very low bit-rates. These distortions are characterised by ringing noise, blocking effects, and blurring. Furthermore, the independent treatment of blocks (and AC coefficients), imposes a minimum redundancy in the representation. In the next chapter, we discuss *feature based coding*, an approach that seeks to address the weaknesses of BTC at low bit-rates, by employing HVS based image models, and arbitrary segmentation of the source.

Chapter 4

FEATURE BASED IMAGE CODING

4.1 Introduction

4.1.1 Context

In the previous chapter we discussed the principal of *block transform coding* (BTC), a *waveform coding* method based on the *block based frequency decomposition* of a source image. The popular DCT was shown to closely approach the rate-distortion characteristics of the optimal KLT, for images modelled as *stationary stochastic processes.* It has long been recognised however, that BTC schemes have reached a saturation point in terms of compression efficiency for low bit-rate coding [80][85]. Severe distortions such as *blocking effects* and *ringing noise,* that are particularly perceptually undesirable, become manifest at low bit rates. While a certain degree of visible distortion is generally accepted in very low bandwidth coding, the *type* of distortion present is an important issue (§3.2.7). In recent years, a number of methods, known as *second generation coding techniques* [80][103], have been investigated that avoid the more subjectively displeasing forms of distortion characteristic of BTC [80][83-87][91][94][100][101][107][108]. The objective of these schemes is to produce more *visually pleasing* images at very low bit-rates by incorporating higher level perceptual factors in the coder model. The MSE as a performance indicator is less important than subjective opinion in such schemes.

4.1.2 Objectives and Overview

In this chapter, we describe the principle of *feature based coding* (FBC), a second generation approach, in which images are not modelled as stochastic processes, but rather considered to have complex structural *features*. The goal of FBC is to exploit this structure, together with the properties of the HVS, to achieve superior "perceptually weighted" compression efficiency at very low bit rates. We review a number of *2-D feature based* schemes in which images are generally modelled in terms of feature *primitives* such as *contours* and *textures*.

In this chapter, we seek to establish a context for the novel scheme described in chapter five. Section 4.2 describes the rational behind FBC and describes the two major types of feature primitives; contours and textures. We proceed, in section 4.3, to review the major contributions to the field of FBC, from both historical and technological perspectives. Finally, in section 4.4, we summarise the principal points emerging from the review and draw some conclusions.

4.2 Fundamentals of Feature Based Coding (FBC)

4.2.1 Why Feature Based Coding?

The ultimate goal of low bit-rate image and video coding has always been the minimisation of the representation, within certain bounds of computational cost and complexity, while retaining an acceptable reconstructed image quality. In recent years, with the historical limitations in processing power being gradually eroded, new opportunities have emerged for the realisation of more complex coding strategies, offering improved compression. There is therefore greater flexibility today in terms of the directions that image and video coding strategies may take, while remaining *feasible* solutions.

One direction that has been explored with notable success has been the adoption of alternative source models. Traditional waveform coding techniques[52], such as the DCT based JPEG and MPEG I & II standards, have based their image models on stationary stochastic processes. If images were to be truly best characterised as realisations of stochastic processes, with time invariant probability density functions, then we could consider the existing waveform based methods to be near optimal, and there would be little

[52] Which may be described as *first generation* coding techniques.

motivation to pursue alternative source models. Fortunately, natural images are in general, far from being realisations of homogenous random fields [10][43 p375][77]. This fact prompts us to explore alternative source models. The motivation for this investigation however, is not merely to find more advanced statistical models, but rather to find representations that more compactly capture the essence of natural images with respect to human visual system (HVS).

On first inspection, the practicality of somehow modelling the infinite variety of image intensity distributions that may be produced by such a complex world would seem forbidding. Indeed, model based schemes that do attempt to model the *content* of a scene are generally restricted to very specific source models (e.g. a human face) [11 p341]. More tractable, for unrestricted content, is to model *features* of the intensity distribution (i.e. the image) itself as they impact on the HVS. While the HVS is a considerably complex system which remains only partially understood [81][88], its sensitivities to a number of commonly occurring intensity distributions have been recognised. In particular, the HVS has evolved to place specific emphasis on patterns of light arising from the structural features of natural objects. It is these patterns, to which the HVS is "tuned", that we call *features*, or *feature primitives*.

If these features can be suitably described, recognised by a computer algorithm, extracted and efficiently represented, then we have the basis for an image coder that is, at least primitively, HVS based. It is generally accepted that image models based on the HVS offer the way forward for very low bit rate coding [83]. A principal advantage of the feature based approach becomes evident at very high compression ratios [82], typically 50:1, where the bits available for image representation are severely curtailed. In very low bit rate applications, images are often compressed solely for the benefit of human viewers. In other words, visual information that might be relevant, for example, in the context of a computer vision application, or pre-press[53] processing, but is not particularity *visible* or *significant*[54] to the low level HVS, may be removed or degraded. It is in this way that a feature based scheme strives to better utilise the available bandwidth.

[53] We refer to the publishing industry.

[54] In this context, the terms visible and significant refer to low level visual processes and should not be confused with the results of higher level cognitive functions, assigning significance to content.

In the following section, we elaborate further on the concept of image *features*. Before doing so however, we pause to state an assumption on the nature of the images under scrutiny, namely that the content of the image to be coded is visually *meaningful*. Feature based coders, being HVS based, are likely only to perform to their full potential if the input image has structure (i.e. is not random).

4.2.2 Feature primitives and the HVS

In the field of psychology, experiments in visual perception have sought to understand the mechanisms of the HVS. The fundamental findings of such investigations have been used in the design of many first generation coders. HVS characteristics such as frequency sensitivity and contrast sensitivity have motivated the use of frequency based transforms and guided the design of quantization tables. In colour images, the design of various colour models has been strongly based on the chromatic sensitivities of the eye, while in video coding, the perception of object motion and correspondence between frames is conveyed by exploiting the temporal characteristics of the HVS. While it is true that many first generation image coding techniques have already been fundamentally influenced by the properties of the HVS, there are many more perceptual characteristics and phenomena to be exploited.

Failure to adequately account for some of these characteristics in the coder model is the cause of many types of visual distortion in highly compressed images. By characterising and incorporating some of these features in the coder model, it is hoped that the more "annoying" forms of distortion may be remedied. Findings from the field of visual perception therefore underpin many of the principal strategies employed in feature based coding. In the following subsection we briefly discuss some aspects of visual perception relevant to feature based coding and describe the two dominant types of feature in FBC; *contours* and *textures*.

A more detailed study of visual perception in general is beyond the scope of this thesis. An excellent text by Sekuler and Blake [81] provides an interesting introduction to visual perception, Wolfe [92] discusses feature detection, and Raff [88, chapter 7] treats the topic of visual perception with respect to digital images.

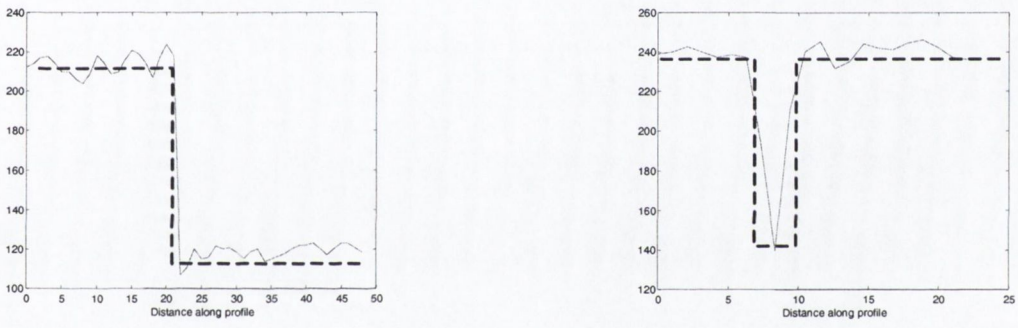
4.2.3 Contours and Textures

It has long been recognised, that the outlines, or *contours*, of objects play an important role in the process of vision [76][81]. Outside of a scientific context, it is interesting to note that graphical artists have been experimenting with theories of image representation and visual perception for centuries. A persistent and dominant technique in the process of creating pictures involves the initial representation of the principal structural features by contours, followed at a later stage by the addition of light and shade [89 p40]. A line drawing, or sketch alone can often communicate all the essential structural information in a scene. This is a remarkable fact when we consider that the intensity profiles of an image and its sketch can be quite different indeed. It would seem that the visual system is naturally capable of *seeing* a close correspondence between an image and its representation by contour lines. Furthermore, experiments in visual search (using eye-tracking) have shown that when examining images, the eyes spend a significant proportion of their time tracing the outlines of objects and fixating discontinuities in contours [90].

The special significance of contours in vision is not without a physiological basis. In the human eye, there are approximately 130 million photoreceptors (*rods* and *cones*) populating the *retina*, which converge[55] to a complex network of approximately one million neural cells (*collector cells* and *retinal ganglion* cells)[81]. In this convergence, information from groups of neighbouring photoreceptors is collected (by collector cells) and conveyed to the retinal ganglion cells. The vast majority of retinal ganglion cells are designed to detect *differences* in light, or *contrast*, rather than overall light level [80][81]. Thus, the neural machinery of the eye extracts *information* about the intensity distributions within locally receptive fields. In particular, a strong response is produced by discontinuities in illumination, characteristic of the *change* in light intensity produced by the structural features of objects. The higher level result of such cell responses, is a neural code that is based primarily on the shape and contrast of objects in an image [88 p55][81 p65]. Thus, contours emerge as prime candidates for feature primitives in FBC. The question naturally arises as to what exactly defines a contour.

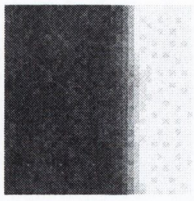
[55] The degree of convergence varies systematically with retinal location [81 p71].

An abrupt change in pixel intensity (or grey level) is commonly referred to as an *edge* [80]. A set of connected edge pixels defines a *contour*. A contour generally delineates the boundary between two *visually distinct* image regions, often corresponding to the structural features of objects represented in the scene. The terms *strong* and *weak* are used to further indicate the degree of abruptness of a contour [80]. In computer vision, where images are often examined with the aim of identifying objects, two principal edge types are defined; *step edges* and *line edges* (see Fig. 4.2)[15]. We do not consider *junctions* or *corners* explicitly, as we are not concerned with the additional structural information conveyed by such edges.

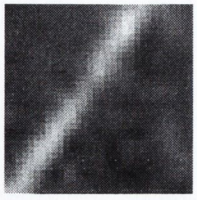


(a) (b)

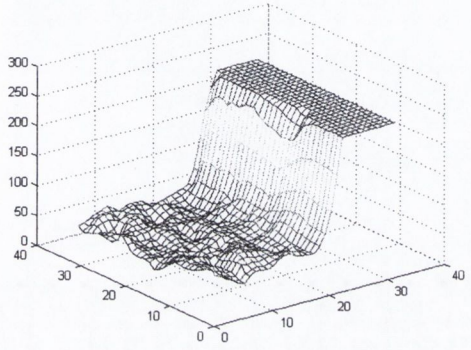
Fig. 4.1 Examples of the profile across a step edge (a) and a line edge (b). Ideal approximations of the edge profiles have been overlaid (dashed lines).



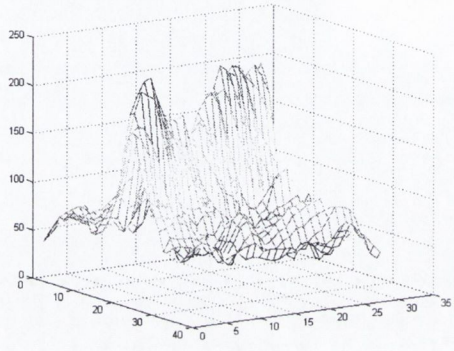
(c) 2-D step edge image



(d) 2-D line edge image



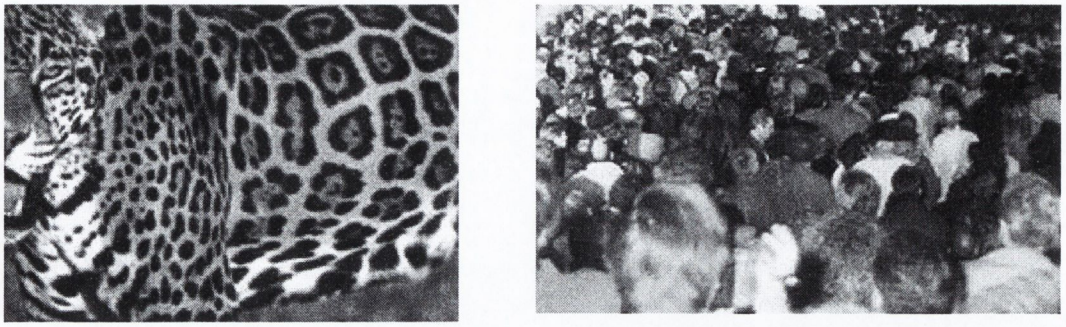
(e) 2-D step edge plot



(f) 2-D line edge plot

Fig. 4.2 Illustrations of *step* and line *edges* for the 1-D case (a) and (b) respectively and the 2-D case, images (c) and (d) with pixel value plots (e) and (f).

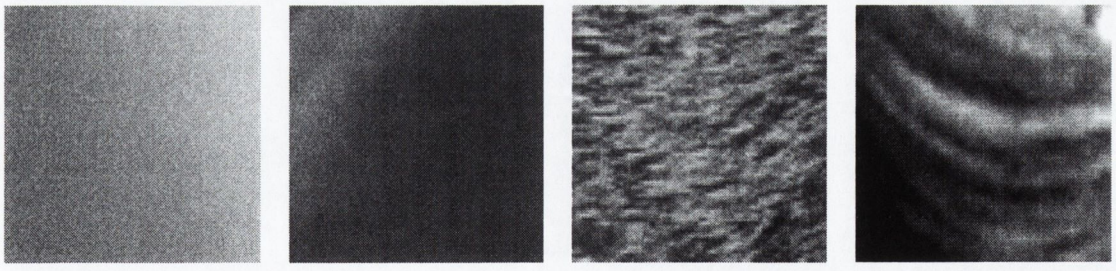
Abrupt changes in intensity however do not always correspond to the structural features of objects, nor do they always elicit a strong visual response [91]. Image regions that represent the surfaces of objects may also be characterised by abrupt changes in grey level. These areas are generally referred to as *textured* regions. While the term *texture* is often used loosely, in a visual sense, to refer to structured patterns of light, it has no formal definition [30 p506][80][88]. This is principally due to the fact that, in this context, texture has a strong subjective component. Not only do many textures require world knowledge to correctly classify, but their intensity characteristics change with scale (see Fig. 4.10). The natural world exhibits textural intensity distributions that are mostly random while manmade surfaces often have texture with a periodic nature [88].



(a) (b)

Fig. 4.3 Example of texture (a) and objects (b) both exhibiting similar intensity characteristics.

In FBC the term texture does not communicate any assumption of structure, it simply refers to any non-contour intensity distribution [80]. Texture constitutes the second major type of feature primitive in FBC. Textured regions may be further described as either *smooth* or *highly textured.* These descriptions are vague however, and serve only to approximately indicate the nature of the intensity distribution. A smooth texture is one in which any change in intensity from one pixel to the next is gradual, i.e. the energy in the region is concentrated in the lower frequencies. A highly textured region, by contrast, is characterised by high energy in the high frequencies, often with abrupt grey level changes indicative of edges (Fig. 4.4). Some authors choose to reserve the term texture to describe highly textured regions and simply refer to smoothly textured regions as smooth [91].



(a) smooth texture (b) smooth texture (c) highly textured (d) highly textured (zoomed)

Fig. 4.4 Examples of smooth and highly textured image regions.

Given that highly textured regions and contour regions both share the characteristic of abrupt changes in grey level, a further means of differentiating the two is needed if feature classification is to be successful. Ran and Farvardin [91] have studied the sensitivities of the HVS in this respect and noted the *strength* of an edge is diminished in the presence of other edges with close proximity. We may apply this knowledge, although simplistically, in

feature detection by considering an abrupt intensity change to only "really" represent an edge if it surrounded by mostly smooth texture on at least one side, otherwise it forms part of a texture.

Adopting this strategy means that highly textured regions are less likely to have contours classified within them. Unfortunately, complex scenes in which relevant information is present at small *scales* pose a problem for feature classification based on this strategy. In such cases, the structural features of an object may be closely spaced and thus classified as texture. This problem is difficult to overcome however without employing higher level cognitive models which are likely to be considerably more complex.

The descriptions of contour and texture presented above represent the two predominant types of feature primitive in FBC. It should be noted however, that these descriptions are by no means definiti ve. A variety of FBC schemes have presented alternative characterizations of features in their models, and these shall be discussed in the following section 4.3.

4.3 Principal Approaches to Feature Based Coding

There are two general classes of second generation image coding methods that shall be discussed here: *segmented image coding* (*a.k.a partition coding* and *region based coding*) and *sketch based coding (a.k.a. contour coding*[56]*)*. The methods of these two classes differ primarily in how they approach the problem of extracting and modelling the perceptually relevant data. Segmented image coding (SIC) methods [80][85][94][101][103][107][108] segment an image into regions of homogenous texture enclosed by contours. The coded message of a SIC scheme consists of the shapes of these regions, defined by their contours, and a description[57] of the texture (intensity variation) within them. Sketch based coding methods [75][76][79][83][84][86][87][100] by contrast, focus primarily on the extraction of contours, and some information about the nature of the grey level intensity across them. Thus, SIC schemes generally find contours as a "by-product" of merging homogenous texture, whereas sketch based schemes specifically model and extract contour information. We shall review sketch based coding first.

[56] Not to be confused the distinct process of coding contour data.

[57] We refer to a coded mathematical description, such as a weighted sum of basis functions.

4.3.1 Sketch Based Approaches

The methods described in this section, while all employing rather different terms to describe themselves, are essentially based on the same principle; the specific characterisation of *edge information* in the coded representation. We describe these methods as *sketch based coding* methods, an intuitive title adopted by Carlson [87]. In sketch based coding, a source image is processed to extract the *locations* of strong edges or contours. This information could be represented by a binary *edge map*[58] as it is essentially two-dimensionally geometric in nature. Typically however, there are coding gains to be made by extracting the shapes of each contour and coding them separately. Furthermore, sketch based schemes generally need to represent more than purely geometric information about contours. Information regarding the intensity distribution across these contours is also extracted. The exact nature of this information and the form of its representation varies between schemes. In general however, an approximation of the original image may be reconstructed from the contour information alone. Texture is therefore often represented in the form of a *residual image*[59]. We proceed to discuss the evolution of sketch based coding methods over the years. These methods have all been motivated by the desire to realize a more "perceptually tuned" approach to image compression.

4.3.1.1 Synthetic Highs System

As early as 1959, Schreiber et al [75] proposed an image coding technique that sought to better model both the properties of the source and of human vision. This early method was called the *synthetic highs system*. The objective of this approach was bandwidth reduction for analog television broadcast, and thus processing was scan-line based. As a consequence, edges rather than contours were modelled by this approach. In fact, edge information was the only component of this scheme represented digitally. The specifics of the scheme are therefore not particularly relevant in the context of this review. What is relevant however is the concept introduced by Schreiber, of separately encoding edges and texture information.

[58] The earliest schemes [75][77] to model edges in image coding were scan-line based, due to their envisaged application to TV image compression, and thus did not represent edges in two dimensions.

[59] An image generated by subtracting a reconstructed or prediction image from the original image, a.k.a. error image.

In his original paper [75], Schreiber stressed the perceptual significance of edge information, citing works describing vision as "*mainly a border or contour phenomenon*". The implications of this "phenomenon" for image coding were expounded further in a later paper of his [76], where the principle of *contour coding* [79] was discussed in the context of digital images.

In a synthetic highs encoder (Fig. 4.5), a source image is processed to generate two separate signal components. In one branch of the coder, the source image is low pass filtered and coded by an analog means. This is the so called *low-pass* signal, which approximates the intensity distribution in the original image, without preserving high-frequency contour or texture information.

In the other branch of the coder, a *high-pass* signal is produced by the operation of an edge detection[60] procedure on the original image. The location and quantized amplitude of each edge points is digitally coded. A non-linear quantizer is employed such that the quantization error remains proportional to the strength of the edge. Thus, the coded representation consists of two *channels* or *components*; one which represents the low-frequency information in the image, and the other which represents high frequency or edge information.

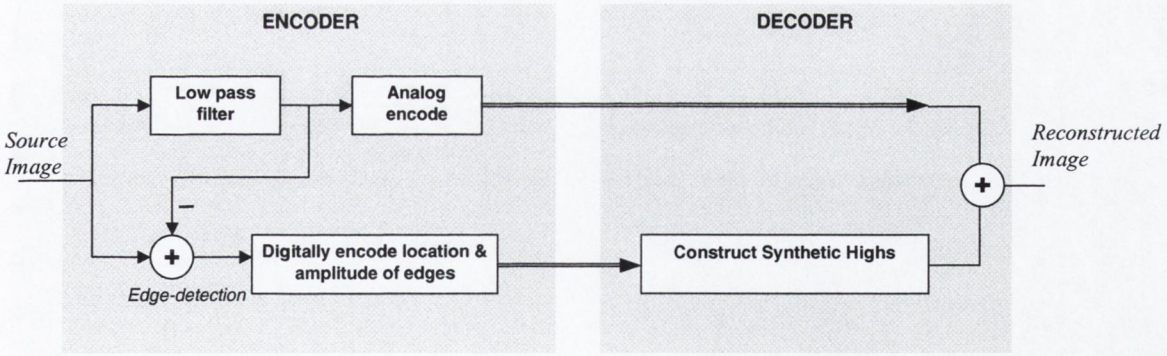


Fig. 4.5 Block diagram of a *synthetic highs* encoder/decoder.

In the decoder, a *synthetic highs* signal is generated from the digitally coded edge information (location and amplitude). This signal is then added to the *lows* signal to reconstruct an approximation of the original signal.

[60] A method referred to as *edge-taking* is employed, a.k.a. cell-to-cell differencing or linear prediction.

Schreiber's scheme paved the way for the more advanced two-dimensional sketch based coders that would follow. It is also interesting to note that the synthetic highs approach is in many ways similar to a two-level *pyramid coder* (see §3.5.4), in that the residual image in pyramid coding corresponds to the synthetics highs part of Schreiber's scheme.

4.3.1.2 Two-Dimensional Contour Coding

While the synthetic highs approach introduced a novel concept for image representation, it was not until 1967 that the principle was extended, by Graham [79], to digital images. With the limitations of the analog representation removed, Graham was first in explicitly representing the geometric structure of the image through two-dimensional contours. This was an important step towards realising the potential of the sketch based approach for image compression. In Graham's implementation, a low pass filtered version of the original image is transmitted along with contour information. Only contours whose gradients exceed a minimum threshold are coded. Contour points are traversed, and at each point the *changes* in the following parameters are recorded: gradient magnitude, gradient direction, and contour direction. All of these pieces of information are Huffman coded. Graham reported compression ratios of between 4:1 and 23:1, depending on image complexity, with "good quality" reconstructions.

4.3.1.3 Two-Component Source Model

In 1972 the theme of the contour/texture modelling was revisited by Yan and Sakrison [77], but with a fundamentally different approach. Their paper is interesting because of their early rejection of the stochastic model for image coding. They proposed a *two-component* image model, intended to better model both the characteristics of the HVS, and the structure of natural images. They argued that typical image signals are made up of essentially two different characteristic components; a stationary component, corresponding to textured regions, and a discontinuous component corresponding to the edges of objects. Based on this structural view of the source, they proposed a representation based on two components.

It should be mentioned at this point that processing is essentially scan-line based, and so, like the synthetic highs method, one-dimensional edges are recorded rather than contours. While they don't explicitly attempt edge detection, they generate a *cubed root* representation

of the source image, in which the locations of sharp intensity changes are accentuated. They call this image the *root-image*. Each scan-line of the root-image is approximated by a number of connected straight line segments (Fig. 4.6). These line segments form the first component of the model, preserving the spatial locations of edge points in the source.

Texture fidelity however is lost in the coarse signal approximation. To enable restoration of texture fidelity, a residual image is generated by subtracting the approximated root-image from the original root-image. This is the second component of their scheme, and may be coded using a standard waveform coding technique (they chose the Fourier transform at that time).

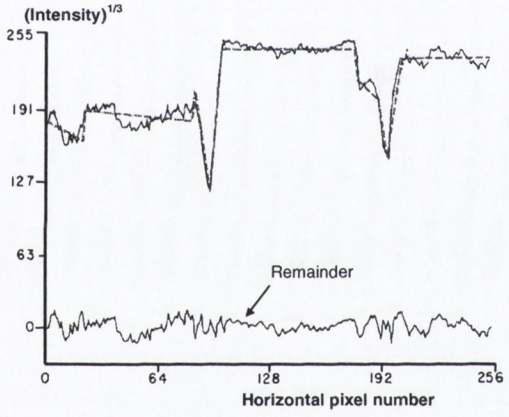


Fig. 4.6 Two-component model coder – fitting line segments (dashed) scan line of a root-image. Reproduced from [77].

Sample reconstructed images presented in their report demonstrated superior rendition of contours when compared to Fourier transform coded images at the same bit-rate (approximately 0.8 bpp.). While the two-component model failed to prompt further investigation of their approach, the rationale behind its conception remains the motivation for many second generation schemes today.

4.3.1.4 Sketch Based Coding

In 1988 Carlson [87] proposed a sketch based coding scheme with some novel advancements. In this work, the aim was to develop a scheme which emphasised the *perceptual acceptability* of the reconstructed images, more so than their mathematical accuracy.

The information extracted from the original image is in the form of contour coordinates, and grey values either side of the contours (Fig. 4.7). This is called the *sketch data.*

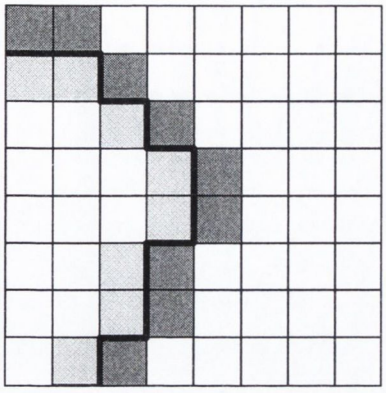


Fig. 4.7 Grey level values recorded for a contour specified geometrically by the heavy black line. This is the so-called *sketch data.*

Contours are extracted using a Laplace-Gaussian operator[61] (LGO), followed by the application of gradient operators across zero-crossings. The LGO locates edges as the zero-crossing of the second derivative [91], while the gradient of the LGO output at a zero-crossing indicates the strength of an edge. A threshold operation is applied to the absolute value of the output to retain only significant edges. The threshold value is the main parameter to control the amount of contour data being retained. This is an important parameter in all sketch based coders, as it directly influences the compression efficiency.

Due to the directional nature of the edge detection operators employed, edge linking is necessary to recover connected contours. The contours of Carlson's scheme actually correspond to the *cracks* (or gaps) between pixels (Fig. 4.7), rather than actual pixel locations, and are represented by shifting contour coordinates ½ pixel relative to the image coordinates. Following contour extraction, the grey level values on both sides of each contour point are coded. Rather than code actual grey level values however, the differences and sums of corresponding points across the contour are approximated by two second order polynomials. It is in fact the coefficients of these polynomials that are coded. In addition to coding the grey level data across contours, the geometry of the contours must be coded. A method based on chain coding was suggested, and likely bit rates of 9 bits per start point and 1.3 bits per contour point estimated.

As the contour data is only intended to reconstruct the visually significant regions of abrupt intensity change, textured regions must be coded separately. A residual image is

[61] The LGO is more generally referred to as the Laplacian-of-Gaussian (LoG). This operator has a "Mexican hat" profile that combines a smoothing effect (the Gaussian) with a second derivative operator (the Laplacian), see also page 96.

generated for this purpose by subtracting the image reconstructed from contour data alone, from the original image. This is then coded by a Laplacian pyramid coder.

In the decoder the sketch data is decoded and the texture between contours approximated by interpolating the grey level data. The problem of texture reconstruction is formulated as a constrained optimisation† (or energy minimisation) problem, governed by the heat or diffusion equation. This method interpolates the values between contour points by means of an iterative reconstruction algorithm that converges rather slowly to the final solution, particularly if the distance between contours is large. To speed up convergence, Carlson used multiresolution grids to initially reduce the distance between contours. The lowest resolution approximation of the sketch data is reconstructed first, and then interpolated to the next resolution and used to seed the next iteration of the interpolation process. This process is completed until full resolution has been reached. To complete reconstruction, the residual image is decoded and added to the reconstructed image.

Carlson concludes that sketch based coding only offers an advantage over waveform coding at high compression ratios (30-75). His method, in common with all sketch based methods, is undermined by the presence of perceptually irrelevant structural detail. It is thus not a good "general purpose" coding scheme such as JPEG. The principal benefit of this approach becomes evident at low bit-rates, where strong contour information is more faithfully reproduced than is typically seen in waveform coders. We note however that edges are considered to be only one pixel wide and thus every retained edge will be rendered as a sharp intensity transition. This is likely to introduce an artificial *look* in the reconstructed image by the abuttal of low and high frequency components without the presence of mid-range frequencies. These errors in reconstruction however may be tolerated in low-bandwidth applications where waveform coders often introduce more visually disturbing distortions.

Probably the main weaknesses of Carlson's sketch based coding are the difficulties faced in selecting the *right* edge information i.e. the contours that are visually important, and the computational complexity of the iterative interpolation process. Some efforts were made to reduce the computational burden of interpolating between contours, but this remains a principal drawback of the scheme (addressed in [83][86]).

4.3.1.5 Three-Component Model

In 1992 Ran and Farvardin [100] revisited sketch based coding with a number of proposed improvements. They remarked on two drawbacks of Carlson's scheme [87]. Firstly, they stated the LGO operator has poor edge localisation properties, giving rise to the extraction of incorrect grey levels values across contours. Secondly, they suggested that Carlson's assumption of edges being one pixel wide was unrealistic. To address these weaknesses, they proposed an alternative edge representation based on edge *brims* (Fig. 4.8). Additionally, they proposed an amalgamation of FBC and waveform coding, based on a three component source model. The three components of their model[62], are described as *primary*, *texture* and *smooth*. These components are generated as follows.

A source image is first filtered with a space-variant, edge preserving low pass filter (the same principle is employed in [80]). The resulting image is called the *stressed image* and contains mainly sharp edges with smooth texture elsewhere. The strong edge information is subsequently extracted from the stressed image by identifying pixels of local maximum *curvature energy*[63] [91] (in horizontal and vertical directions). Unlike Carlson's method, contours are defined in terms of *edge brims*, which may be thought of as the lower and upper coordinates of an edge, corresponding to the maxima and minima of the second derivative (Fig. 4.8). In this way, edge transitions that span more than one pixel may be represented more accurately. Thus for each edge, two contour components are generated, one for each brim (in Carlson's scheme a single contour is coded with values either side).

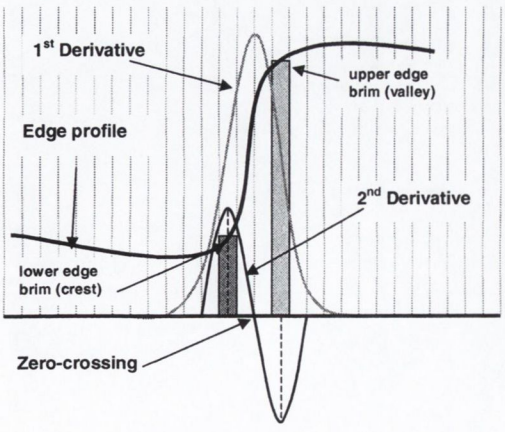


Fig. 4.8 Illustration of upper and lower edge brims, corresponding to the crests and valleys of the second derivative operator.

[62] Described in detail in [91].

[63] The curvature energy of a point is calculated by subtracting twice the value of the pixel from the sum of the pixel values to its left and right and then squaring the result.

The *primary* component of the scheme is represented in coded form by the contour information. For each contour, initial coordinates, geometric structure (chain-coded), and mean intensity value, are encoded. The *textured* component is generated by subtracting the stressed image from the original image, and so does not contain any strong edges, but only texture data. The *smooth* component is generated by subtracting the reconstructed primary component from the stressed image (Fig. 4.9). The smooth and textured components are added together and coded using a waveform technique such as subband coding or adaptive DCT. The reconstructed image is obtained by decoding the smooth/texture component, reconstructing the primary (strong edge) component, from the contour information, and combining them together.

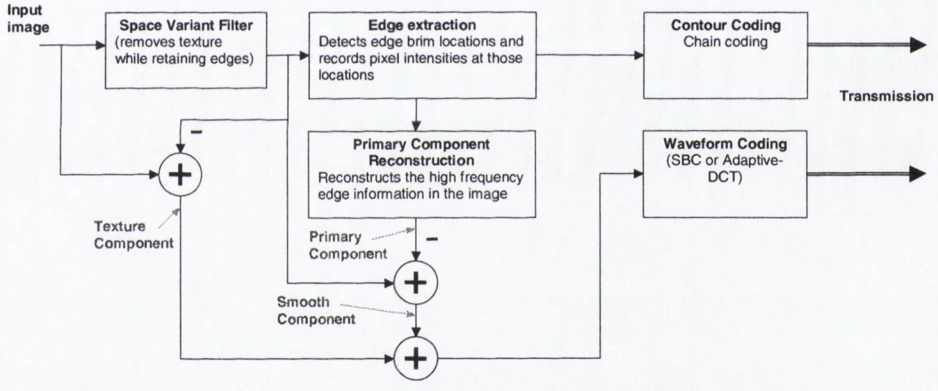


Fig. 4.9 Image encoder based on a three component image model

While the three component model is an interesting source model, perhaps Ran and Farvardin's most interesting contribution to sketch based coding is their representation of the contour component using edge brims. As pointed out by Robinson later [84], these lines are less noisy than the Laplacian zero-crossing that represent edge midpoints. Edge brims do not show as many "random" fluctuations because they do not represent a very rapid change in grey value with respect to edge position, as edge midpoints do [83][84]. They also enable edges of different widths to be elegantly accommodated in the representation.

The extraction of *separate* smooth and textured components seems to yield few if any coding benefits, as they are summated prior to waveform coding. In fact, it is most likely that any improved performance in coding the source/texture components would be solely

due to the absence of strong edge information, as represented in the primary component. No remarks are made on the computational efficiency of the scheme, but it is perhaps reasonable to assume that like Carlson's [87] scheme the solution to the energy minimization problem, used to generate the primary component, is computationally complex.

4.3.1.6 Image Coding with Ridge and Valley Primitives

In 1995 Robinson [84] proposed a sketch based scheme that he described as a "one-component" scheme. The inspiration for his method came from earlier work in cartoon generation which sought a very economical representation of moving images for sign language conversations. The principle of grey-level image coding with ridge and valley has much in common with both [87] and [100].

Robinson's method is similar to Ran and Farvardin's, in terms of his representation of edge brims, although his edge extraction method is somewhat different. A source image is first filtered with a Laplacian-of-Gaussian (LoG) operator. The output of the filter is fed to a ridge/valley follower that locates strings, or threads, of pixels along valleys and ridges. Only the strongest outputs of the Laplacian-of-Gaussian operator, forming threads of at least a minimum length, are retained. The geometric properties of each thread are encoded by chain coding. The threads are used to sample the original image, generating an intensity *profile* along the thread, which is coded using a one-dimensional *yardstick* (fractal) method. Although Robinson describes his method as "one-component", a small regular grid of samples, typically 32×32 pixels, is also coded and transmitted with the ridge/valley component. This is done to aid texture reconstruction in smooth image areas.

Like similar previous schemes [87][100], reconstruction is a scattered data interpolation problem. The "known" pixel values are reconstructed from the yardstick compressed ridge and valley data. This requires interpolation of the profile values lost by yardstick coding. The remaining known pixel values are those in the uniform spaced sample grid. *Natural Neighbour Interpolation* was deemed most suitable for reconstructing the unknown samples between the known points.

Reconstructed images, coded at bit-rates of between 0.1 bpp and 0.4 bpp were presented. Encouraging results were demonstrated for low complexity images, i.e. images with mostly

smooth texture, such as close facial images. A problem characteristic of many second generation coders, namely difficulty in coping with highly textured regions, is also characteristic of this scheme.

4.3.1.7 Edge and Mean Based Image Compression

Desai et al [86] proposed *edge and mean* based image compression in 1996. This scheme is based on the ideas of Carlson [87] but introduces a number of novelties. A primary objective of their approach is a reduction in the computational complexity typically associated with sketch based schemes (also investigated by [83]). Envisaged applications include low power portable video telephones.

Edge extraction is performed on the original image using two different edge operators. Sharp edges are extracted using the *Sobel* operator and smoother edges are extracted by a *Level* edge operator[64]. The combined *edge-intensity map* is formed as a weighted average of these two filter outputs. A binary edge map is obtained by thresholding the edge-intensity map, and thinning the result to narrow the width of each edge to one pixel. Contours are coded by first converting edge pixel relationships from 8-connected to 4-connected, eliminating short contours, and chain-coding the remaining contours.

Contour intensity data is sampled by a novel and interesting method that circumvents one of the problems of Carlson's scheme; namely where pixel values immediately next to an edge are not good representatives of the average value a short distance from the edge. It was noticed that taking contour intensity samples from the pixels directly next to the contours themselves could lead to large reconstruction errors. The approach was thus taken to fit straight line segments to the intensity distribution between contours (horizontally and vertically). The end-points of these line segments, where they intersect contour points, are used as the contour intensity points (Fig. 4.10). Less precise edge detection is facilitated by this approach. Redundancy between successive contour intensity points is exploited by coding the profile using a reduced set of *mean values,* obtained from short, fixed length, line segments fitted to the profile.

[64] The level edge detection kernel is essentially like a "spaced out" Sobel kernel, e.g. $\begin{bmatrix} -1 & 0 & 0 & 1 \\ -2 & 0 & 0 & 2 \\ -1 & 0 & 0 & 1 \end{bmatrix}$

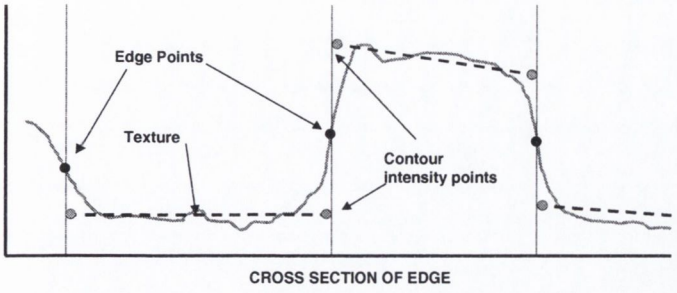


Fig. 4.10
Extraction of contour intensity points by fitting straight line segments to the texture between contours.

It is suggested in [86] that a residual image should be coded to improve the quality of the reconstruction. Rather than do this however, the authors chose to add an additional "component" to the scheme, which they called *mean* coding. In mean coding, the input image is partitioned into 10×10 blocks, and the average value of each block encoded[65] and transmitted.

At the decoder, the contour data is decoded and the textured regions between contours are reconstructed by linear interpolation in horizontal and vertical directions. This is in contrast to Carlson's [87] more complex iterative interpolation method. The intensity of each *non-edge* block (10×10 blocks) is set to its corresponding mean value.

Superior interpolation performance is achieved by the use of simple linear interpolation. This leads to "bleeding" from broken contours however, which may extend across large regions of an image. Addition of the mean value data removes bleeding in non-contour blocks, but introduces additional distortions in the form of blockiness.

4.3.1.8 Strong Edge Features for Image Coding

In 1996 Casas and Torres [83] proposed a two-component sketch based coding method with some novel advancements. The principal advancement of their method is the use of an efficient *morphological interpolation* algorithm to reconstruct smooth texture from the sketch data. We briefly discuss their scheme.

Strong edges are extracted as follows. The *morphological Laplacian*[66], an approximation to the second derivative, is computed on an input image. Upper and lower edge brims (ridges and valleys) are found by computing the *watershed* of the Laplacian and of its dual. The

[65] They refer to simulations, encoding the mean values by statistical coding.

[66] Defined as the residue of the gradient by dilation, and the gradient by erosion.

intensity values corresponding to edge brims (after thresholding) are extracted from the source image and each edge profile approximated by a polynomial. The geometric structure of each edge brim is encoded by derivative chain coding. Texture information is represented by a residual image, calculated by subtracting the image reconstructed from the edge brims, from the original image. The residual image is encoded with a linear subband coding scheme.

It is the application of the morphological interpolation technique [116] for smooth texture reconstruction that is of most interest in this scheme. Starting from the initial set of edge brim pixels, the morphological interpolation technique approximates the intensities of the unknown pixels by a two step algorithm, namely a *geodesic propagation* step followed by a *smoothing* step [83]. In the first step the values of known pixels are propagated by geodesic dilation to fill the empty areas of the image. False contours are created where the "fronts" of the propagation meet. The smoothing step assigns the pixels on either side of these false contours to the mean level. These pixels are then described as secondary pixels and used at the starting points (with the original values) for the net iteration of the two steps. The process is iterated until the algorithm has converged to the final interpolated image (after 9 iterations for example), at which time the residual image is added to it

The problem of computational complexity in the previous methods for sparse data interpolation is addressed by this method. Pervious attempts to reduce the computational burden of the interpolation process [87][100] (space variant linear diffusion filters/multi-resolution grids) reported executions times about two orders of magnitude higher than this scheme. While the subjective quality of the reconstructed images compares very favourably with JPEG at high compression ratios, a certain smoothening effect is visible. Small objects may be removed completely. These forms of distortion are believed to be more favourable to the types introduced by transform and subband coders at similar bit-rates.

4.3.2 Segmented Image Coding (SIC)

In segmented image coding [80][85][94][101][108][117], an image is segmented into closed regions of homogeneous or slowly varying texture. The shapes/borders of these regions are described by contours and constitute the first component of the coded representation.

Unlike sketch based schemes, contours serve only to describe the geometry of textured regions, i.e. contour intensity profiles are not explicitly represented. The second component of a SIC scheme describes the textures within each region. The emphasis placed on texture by SIC schemes reflects the sensitivities of the HVS to homogenous regions of texture [103]. Thus the SIC approach, while still a contour/texture method, approaches the problem from a slightly different perspective.

There are four general steps in a segmented image coding scheme:

1. **Pre-processing**. Pre-processing is generally required prior to segmentation to make the image more amenable to compression. This usually involves noise removal and a certain degree of texture smoothing [103].

2. **Segmentation.** The goal of segmentation is to obtain an optimal partition of the image under some proposed constraints [85] (e.g. desired compression ratio, region of interest). The compression performance of SIC schemes is highly dependent on the success of the segmentation algorithm [101]. There are three principal, sometimes conflicting, requirements for good performance. The segmented regions should correspond to objects as they are perceived by the HVS, they should be homogenous, and there should not be "too many" of them.

3. **Contour Coding**. Contours are purely geometrical in nature in SIC schemes. They locate the boundaries between textured regions. Both lossy and lossless contouring coding methods may be used, depending on the requirements of the application.

4. **Texture Coding**. Texture coding is the final step in a SIC scheme. Typically, the texture within each region is represented by a weighted sum of basis functions [101]. Texture coding is generally seen as on the most computationally expensive step in a SIC scheme, as for each region a new basis must be constructed [101] (different size and shape).

In segmented image coding, the majority (about 60%) of the encoded message is generally devoted to contour coding [85]. The number of regions into which an image is segmented will largely influence the overall compression efficiency. In fact, if too many small regions

are segmented, block based transform coding may well produce better results [118]. Thus, SIC schemes are an interesting solution to low bit–rate coding if the meaningful data in an image is either characterised by large objects, and/or a few small objects. They do not offer interesting solutions where the number of regions in an image must be large to retain an acceptable degree of comprehensibility. An example of a suitable type of image for SIC would be a head-and-shoulders image of a human, against a plain background. In this case the eyes and mouth would represent small but important objects, and the remaining regions could be relatively large.

4.4 Summary & Conclusions

This chapter has provided an overview of the motivations for feature based coding, introduced the fundamental concepts, and discussed the two major feature based coding approaches: sketch based coding and segmented image coding. The majority of the discussion was devoted to sketch based coding, as this approach is more relevant in the context of the solution proposed in chapter five.

FBC schemes are perceptually motivated coding schemes that make inherent assumptions about the nature of the source content, and how that content is likely to be perceived by the HVS. Traditional FBC schemes make no attempt to analyse the *semantics* of the content, but rather focus solely on detecting and coding features primitives. Generally, small features are viewed as being less significant than large features, and may often be removed. This is illustrated by the elimination of short contours in sketch based coding, and the merging of small regions in SIC. This approach has a perceptual basis [109][110], but may fail dramatically when relevant information is present at small scales (e.g. facial features).

One of the principal drawbacks of feature based methods is their computational complexity. Although few authors provide a detailed analysis of the computational requirements of their schemes, it is generally indicated that they exceed transform based methods, in terms of complexity, by an appreciable margin [101]. Furthermore, while it is difficult to generalise the error robustness of FBC schemes, due to the diversity of approaches, they would seem inherently more sensitive to data loss, particularity the loss of contour information.

Chapter 5

HYBRID FEATURE BASED TRANSFORM CODING

5.1 Introduction

5.1.1 Context

In chapters three and four, the principles of block transform coding (BTC) and feature based coding (FBC) respectively were discussed. In this chapter, we introduce a novel hybrid coding scheme that exploits the strengths of both schemes, while lessening their respective weaknesses. Specifically, we seek to combine the computational efficiency and robustness of transform coding, with the subjective benefits of feature based coding at very low bit-rates. We describe this scheme as *feature based transform coding* (FBTC) The proposed scheme has both statistical and perceptual motivations.

5.1.2 Objectives and Overview

The objective of this chapter is to present the investigated solution and to provide a reasoned argument for the solution, with respect to existing knowledge in the field. Section 5.2 begins by describing some of the strengths and weaknesses of BTC and FBC for very low bit-rate coding, and draws some conclusions as to where they might offer complimentary advantages in a combined scheme. Section 5.3 proceeds to make some observations on the perceptual and statistical nature of contours and texture, and proposes a component model based on these observations. Section 5.4 provides a detailed description of a hybrid coding scheme incorporating the proposed component model and

section 5.5 discusses the properties of the scheme. Finally, 5.6 briefly summaries the chapter.

5.2 Problem Description

The rationale for the scheme investigated in this thesis is based on a number of observations regarding the strengths and weaknesses of both *block based transform coding* and *feature based coding* for low bit-rate image compression. We propose that these two schemes may be amalgamated in such a way that some of the weaknesses of one scheme may be offset by the strengths of the other. This is a classic motivation for the design of hybrid schemes. In this section we identify a number of problems with BTC and FBC for low bit-rate coding and state the properties of each scheme that offer advantages with respect to the other.

5.2.1 BTC and FBC at Low Bit-rates

The most popular block based transform for image and video compression is the discrete cosine transform (DCT) as discussed in chapter three, section 1.4. The success of the DCT may be attributed to the simplicity of its implementation, its sound theoretical foundation, and its relatively good performance [8 p2]. In the case of JPEG [50], the goal is to achieve moderate to good quality at 0.25 to 0.5 bits per pixel [19]. This scheme works well with a wide range of image types and is therefore often described as *general purpose*. At high compression ratios however, the JPEG scheme, in common with many block based codecs, tends to exhibit a number of perceptually distracting distortions (§3.2.7). These include *blocking effects* and *ringing noise* (§3.2.7), and have been the main motivation behind the development of second generation coding schemes [80]. Furthermore, while block based coding is advantageous for computational efficiency, it imposes a minimum redundancy in the representation. This redundancy can become significant in large *simple*[67] images.

In feature based coding, the common distortions mentioned above are largely avoided, albeit with the introduction of some alternative forms of distortion, and redundancy in simple images is significantly reduced by content dependent segmentation. The practical

[67] A *simple* image is one that contains no more than a few small objects or complex textures. It is characterised principally by large objects with smooth textures.

application of these schemes however, is undermined by their increased computational complexity, and strong signal dependence (i.e. compression works well with some image types but poorly with others [103]). In the following sub-sections we look more closely at these problems.

5.2.2 Segmentation and Coding Redundancy

In BTC, an image is segmented into equal sized blocks whereas in SIC, image segmentation is dependent on the source content. Both approaches have advantages and drawbacks.

Signal homogeneity. In SIC, by segmenting an image into homogeneous regions, the texture within each region will generally lend itself to more efficient representation. In the most simple representation for example, the texture within a region may be approximated by its mean value. Even with this simple approach, intelligible images may often be produced. In block based coding by contrast, segmentation is *arbitrary* with respect to the signal content, and does not generally reflect the spatial variation in the signal statistics. An attempt to adopt a strategy similar to the one above for texture representation, would result in severe blocking effects. On the other hand, in BTC no information about the shape/location of blocks needs to be communicated to the decoder. This is particularity advantageous in *busy* images where there is little texture homogeneity. In such circumstances, the performance of SIC schemes may be severely degraded due to the overhead in communicating the shapes and textures of many small regions [85] (or many contours in sketch based coding). Thus, in general, FBC achieves coding gains with large homogeneous areas but does not tend to perform as well with complex structures [103], while for BTC the opposite is true.

Minimum Redundancy. In block based coding, the segmentation of a source image into uniform blocks establishes a *minimum redundancy* in the representation (without the use of hierarchical structures such as quad-trees). While the DCT may be efficient at coding highly correlated signals within each block, the block based segmentation of BTC fails to *efficiently* exploit correlations across block boundaries. At most, the correlation in mean intensity level from one block to the next may be exploited (by DPCM), but even if the same "pattern" is repeated from one block to the next, separate AC coefficients will still be

generated for each block[68]. Thus while the DCT is generally efficient at coding arbitrary intensity distributions within a block, the block based segmentation prevents signal characteristics that span multiple blocks from being exploited in a more global sense. FBC schemes by contrast, segment an image according to the nature of its content. Large areas of homogeneous texture may generally be represented efficiently with little coding redundancy. SIC schemes operate on the principle that the overhead in communicating the *shape* of regions is offset by the coding advantages of texture homogeneity within each region. It should be noted however that the minimum redundancy of BTC offers advantages for error robustness, as shall be discussed in section 5.2.6.

5.2.3 Signal Dependency

While it is often said that the DCT offers near optimal energy packing [19 p45], this is only true if the source may be best characterized by a first-order Markov process. Over an image as a whole, structure is generally evident which undermines this assumption. For example, by employing a feature based image model, a much more compact representation of *edge information* may be generated in comparison to the DCT, which generally requires many non-zero DCT coefficients to sufficiently represent [103].

For highly textured images the DCT's performance benefits extend across a large area of the image. It is with such images however that SIC and sketch based schemes are typically less successful, in terms of coding efficiency. This essentially means that the DCT is very good at efficiently coding highly textured blocks [82][107], i.e. where structure is present at *small* scales, whereas FBC techniques work well with simple images dominated by *large* structural features [85].

5.2.4 Distortion & Subjective Picture Quality

One of the principal failings of BTC at low bit rates is the appearance of blocking effects (§3.2.7.3). Blocking effects take the form of two perceptual distractions; abrupt discontinuities in grey level, in which a distinct block boundary may be visible, and Vernier offset, in which continuous contours or intensity variations appears broken and spatially displaced. Furthermore, while ringing noise does not occur to any significant degree in BTC at moderate bit rates (§3.2.7.2), at low bit rates it tends to become particularly visible

[68] It should be noted that in this instance, hybrid BTC/VQ could reduce AC coefficient redundancy.

about sharp intensity transitions, i.e. strong edges. Given the crucial role played by edges in the perception of visual information, these forms of distortion make BTC less favourable for low bit-rate applications.

As feature based methods don't employ block segmentation, blocking effects are generally absent in FBC. Furthermore, sharp contours are generally reproduced clearly and without ringing noise. Thus we conclude that, from a perceptual viewpoint, FBC emerges as a more favourable approach for coding edges and smooth texture.

5.2.5 Computational Complexity

A number of fast recursive algorithms exist for the DCT (§3.4.3.2), and this has certainly been a significant factor in its selection for major image and video coding standards. Its suitability for parallel execution through block based processing has also been exploited in the development of VLSI solutions [19][20][43]. Despite its poor subjective performance at low bit-rates, image and video coding solutions based on the DCT are likely to remain in use for many years to come. This is apparent by the recent emergence of DVD, an MPEG-2 based system, as a medium for mass distribution of consumer video. Furthermore, while JPEG2000[69], the next major still image coding standard, will address JPEGs weaknesses at low bit-rates, it is not intended to replace JPEG, but rather compliment it [41][85]. Thus we observe, that in addition to the DCT being computationally efficient in itself, hardware support for DCT based codecs is widespread and growing rather than diminishing[70].

The computational complexity of FBC schemes by contrast, remains a considerable drawback. While a number of schemes have employed techniques to reduce computational complexity [87][83][86][102] it would still seem to remain a significant disadvantage of FBC schemes in comparison with the DCT. There seem to be a number of key areas where, in terms of computational efficiency, FBC schemes require improvement:

- *Encoder*: Segmentation: Contour extraction/region segmentation.

[69] Recall that JPEG2000 is a wavelet based standard.

[70] Many manufacturers of mainstream computer display adaptors now incorporate IDCT hardware in their products (ATI Technologies, for example).

- *Encoder*: Generation of basis functions for each segmented regions in SIC.

- *Encoder*: Necessity, in some schemes, to decode part of the encoded signal, in the encoder, in order to a) find lost features or b) generate smooth/texture components or residual images.

- *Decoder/Encoder*: In sketch based coding, the reconstruction of the edge component of the signal, through interpolation (solving of energy minimization problem) remains a computationally complex process.

One could of course argue that computational complexity should not be such a significant factor in codec design, as even the most complex solutions are likely to become feasible as CPU processing power advances over time. In some cases, computational complexity is not a primary concern, as bandwidth limitation is seen as a bigger obstacle. In many other cases however, processing power is as much a *limited resource* as bandwidth, and complexity concerns are equally justified (e.g. mobile devices).

5.2.6 Error Robustness

There are two types of error that are handled well by BTC; coding errors and transmission errors. Treating coding errors first, if a block is badly reconstructed, i.e. is visually quite different to the original, the extent of the "damage" is spatially constrained. This essentially means that one bad block will only ever *look like* one bad block. In SIC, and sketch based coding in particular, coding errors[71] in one part of an image may have visible effects across large areas. These may be catered for by coding a residual image, in which case the encoder must also incorporate the decoder, and incur the necessary computational overhead [83][87][100].

The second type of error that block based coding caters well for is transmission errors. If a block is lost during transmission, it may often be reasonably approximated from adjacent blocks [104][105][106]. This is in contrast to FBC where loss of any one contour or region may cause significant distortion across a large region of the image. Thus, while BTC does

[71] Errors due to broken or badly located contours for example.

introduce a minimum redundancy between blocks, this redundancy has benefits for error robustness.

5.2.7 Conclusions

Based on the discussions of the preceding sections, we suggest where BTC and FBC might offer complimentary advantages in a combined scheme. The specific details of the proposed amalgamation however shall be discussed in later sections.

1. For low bit-rate coding the DCT is inefficient at representing sharp edges where edge blurring and ringing noise are often perceived. The *problems* associated with strong edges are not present to the same degree in FBC. Furthermore, in the context of a complete scheme, FBC techniques offer improved coding efficiency for edge features[72]. Thus, we propose a contour oriented approach to strong edge representation.

2. The DCT is good at representing highly textured regions while FBC schemes don't generally offer any advantages for texture coding in a highly textured region (within complete blocks). Given the DCT's competence for texture representation [82], and its computational advantages, we propose the DCT for texture coding.

3. Segmenting the source image into uniform blocks offers several advantages; efficient computation, parallel encoding, transmission error robustness, and the spatial containment of reconstruction errors. There are also disadvantages however; distortion & redundancy in particular. We propose that in the context of a *component model*, the problem of blocking effects can be satisfactorily addressed, and the advantages mentioned above may be considered to offset the loss in coding efficiency.

Several questions arise from the proposals above. In particular, it is not clear how BTC and FBC might be combined in such a way that the compression benefits of each scheme are retained. We begin to address these questions in the following section.

[72] It remains to be seen if these benefits are retained in a hybrid coder.

5.3 Towards a Solution

5.3.1 Observations, Assumptions, & Approximations

In this section (§5.3), we describe a number of observations, assumptions and approximations that fundamentally underpin the hybrid coding strategy described in this chapter. We proceed to introduce a novel component model based on these assumptions. The observations and assumptions described in this section however, are clearly intended to only represent approximations to the *true* characteristics of images in general. Thus, the applicability and benefits of the approach will be limited to specific classes of images. Furthermore, the proposed scheme is inherently lossy. It is argued that the assumptions presented below are justified in the context of the application, i.e. very low bit rate coding, where a certain degree of perceptually based image simplification may promote superior compression.

5.3.2 Contour Intensity Profiles

We propose that the boundaries of objects in *simple* images may generally be characterised by one of the following types of intensity distributions: a *step* edge, a *line* edge, or a combination of step and line edges[73]. We propose that, for low bandwidth applications, where a certain degree of object simplification may be tolerated, the boundaries of "structurally significant" objects in a scene may be adequately approximated by one of the aforementioned edge types (see Fig. 5.1).

[73] Note that a *staircase* edge may be viewed as a combination of step and line edges.

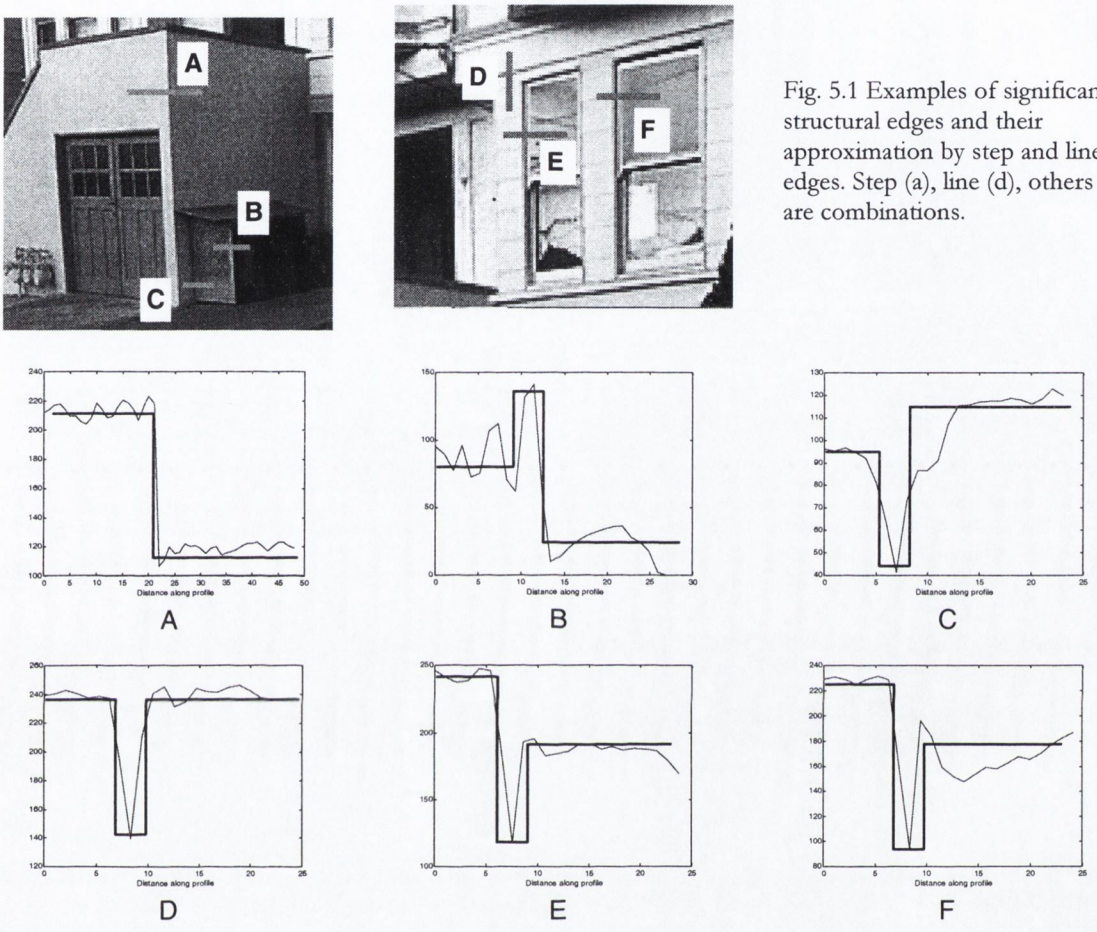


Fig. 5.1 Examples of significant structural edges and their approximation by step and line edges. Step (a), line (d), others are combinations.

5.3.3 Statistical Aspects of Contours

We propose that the boundaries of objects in images of natural scenes often separate significantly large regions of smooth texture. In other words, there is often little change in the perceived intensity (values) of pixels, extending some distance from the edges of objects. Where this is the case we suggest that if the grey level value of some smooth texture *near* an edge is known, then we can often reasonably predict an approximation of the intensity profile at the edge. Assuming this to be true (which we investigate in chapter 6), we propose that for strong edges we need to know two things:

1. The geometrical *location* and *shape* of the edge.

2. The nature of the intensity distribution in the "general vicinity" of the edge.

We seek to exploit this "simplified view" of edges and the intensity distributions about them in an efficient way. This is one of the principal motivations for the component model proposed in this thesis.

5.3.4 Smooth Texture

Let us assume that an image is characterised solely by smooth texture, i.e. low spatial frequencies. In such a case the image may be *subsampled* without loss according to the Nyquist sampling theorem. In the context of a block based coding scheme, subsampling such an image prior to encoding would be advantageous to reduce inter-block signal redundancy[74] and promote superior compression, i.e. the smaller image could be compressed more efficiently than the larger image. Ideally we would like to be able to subsample every image that could be more efficiently coded, without loss, by doing so. Few real images fall into this category however.

If, however, we decided that some loss could be tolerated, then we would have the basis for an image coding technique that achieves compression, in part, by subsampling. The question as to what "loss" could be tolerable naturally arises. Taking a feature-based view (see chapter 4), emphasis is placed on the importance of smooth texture and strong contours. To prevent contour information being lost due to subsampling it would have to be coded separately (i.e. outside of the original image). The subsampling factor would then determine the maximum spatial frequency of texture in the image. This is similar the theory behind the *synthetic highs* [75] coder described in chapter 4.

If the assumptions of section 5.3.3 are now considered, it may be seen that the contour information that must be extracted may take the form of an edge map[75] and the subsampled image should provide sufficient information to allow reconstruction of the intensity profiles across contours. If the subsampled image is to provide this information then there are implications for the subsampling procedure which shall be addressed later in this chapter. The reconstruction of coded images then takes the form of an "edge-assisted" image interpolation problem.

---

[74] Recall that AC coefficients are independently coded in a block based scheme.

[75] i.e. the edge map tells us *where* abrupt intensity changes occur, but nothing about the exact nature of the change.

5.3.5 The Component Model

Based on the assumptions of the previous sections being true for *many* simple images, we propose a *three component image model* consisting of *contour*, *texture*, and *smooth* components.

In the proposed model, contours are employed *solely* to define the spatial constraint of texture and smooth data either side of them. This is similar to the role played by contours in segmented image coding (SIC). Unlike SIC however, we do not require contours to be closed, nor do they define regions of texture. Their sole function in the proposed scheme is to mark the locations in the image where abrupt intensity changes occur. In this sense, they perform a similar function to the sketch *geometry* data in sketch based coding. These contours form the first component of the model and shall be represented in the spatial domain.

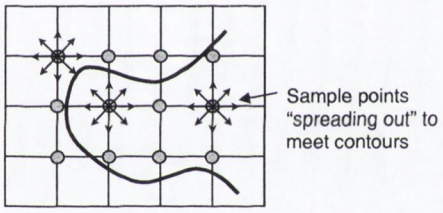


Fig. 5.2 Illustration of concept of sample points "providing" the information to reconstruct intensity distributions about contours.

The two remaining components of the proposed model are *smooth* and *texture*, both of which are coded using waveform techniques. The smooth component represents a simplified view of the intensity distribution over the surface of the image, but is not simply a low pass filtered image. The sample points that make up the *smooth image* are extracted from the source image with respect to the locations of contours. In other words, sample points that lie close to contours must be chosen such that the abrupt intensity change across the contour may be reconstructed by the decoder. In effect, this means that only *non-contour* regions may be low pass filtered prior to subsampling. This point will be elaborated in section 5.4.

The *texture component* is employed to represent high energy features, other than strong edges. Essentially, the texture component selectively restores the mid to high range frequencies lost in the subsampled smooth component. Recall from section 4.2 that a strong edge is only considered as such, if it is surrounded by smooth texture on at least one side. We call *edge like* features that are closely spaced, *busy edges*, and classify them as texture. We do this

partly for reasons of coding efficiency, and also because our assumption of strong edges being surrounded by mostly smooth texture (§5.3.3) is not valid for busy edges. In summary, an image is *modelled* in terms of the following three components:

1. A *contour component* representing the geometrical locations and shapes of strong edge features (other than busy edges).

2. A *smooth component* representing a subsampled, "edge sensitive" approximation of the low frequency information in an image.

3. A *texture component* representing areas of high energy, including busy edges, but not strong edges.

In the following section (§5.4), we present a coding scheme based on the component model.

5.4 Proposed Scheme

5.4.1 Objectives

In section 5.2, the advantages and disadvantages of BTC and FBC for low bit-rate coding were described. Section 5.3 established the foundation for a possible way forward. In this section we provide a detailed description of a hybrid coding scheme, motivated by the previous discussions. We begin however by stating the objectives of our solution. The following properties are specified.

1. **HVS based image model**. In particular, special emphasis should be placed on the "good" reconstruction of strong edges, and the minimisation or removal of blocking effects in smooth areas.

2. **Low complexity encoder and decoder**. Envisaged applications include low power remote monitoring where the encoder complexity, in particular, must be kept minimal.

3. **Component independence**. The solution should be flexible with respect to inter-component dependency. In other words, failure to code and transmit either the

contour or texture component should still allow the smooth component to reconstruct an approximation of the original image.

4. **Comparable performance**. The scheme should be capable of reconstructing images, with a subjective picture quality at *least* comparable to JPEG at the same bit-rate.

5. **Compatibility with block based DCT**. The potential to integrate with, and take advantage of existing DCT solutions is seen as a major bonus, but not strictly a requirement.

5.4.2 Overview

5.4.2.1 Encoder Overview

A block diagram of the proposed scheme is presented in Fig. 5.3. It should be mentioned at this stage that a number of alternative schemes were investigated, each with its own advantages under various circumstances. The scheme presented in Fig. 5.3 depicts the *first configuration* of the encoder that was investigated, and shall provide the reference model for initial discussions. Refinements shall be added to the model as specific issues are encountered in the discussion. This approach is taken to avoid "bombarding" the reader with all the issues at once.

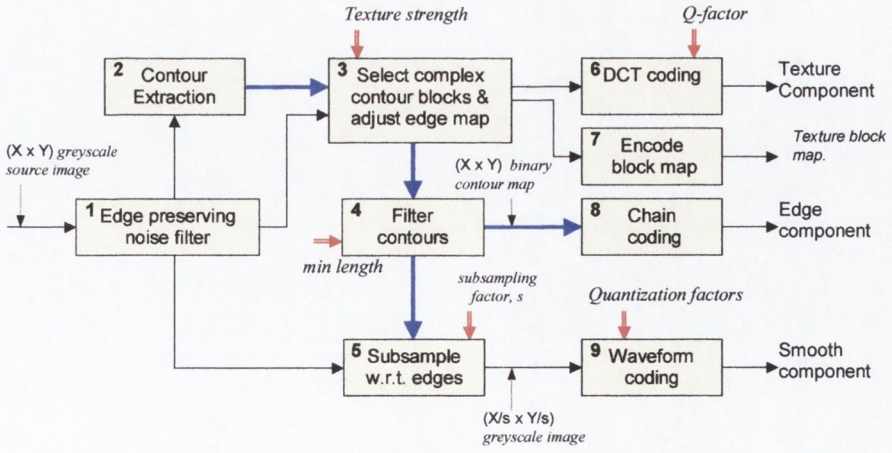


Fig. 5.3 Hybrid feature based transform coder (FBTC) – *first configuration*. The heavy blue lines indicate the transfer of binary edge maps, while the short double red arrows indicate where parameters of the model may be independently adjusted.

The principle of still image coding with the proposed scheme is as follows. A source image is first filtered to remove noise while leaving edges intact (stage 1 in Fig. 5.3). Using the filtered image, the locations of strong edges are extracted and a binary *contour-map* is produced. This contour-map, and the original filtered image, are passed to a texture extraction procedure which identifies areas of high signal variance. The contour-map is employed in this procedure to protect against strong edges being classified as texture and assist in the recognition of busy edges. During this stage, *blocks* of texture are extracted from the original image and any busy edges are removed from the contour-map. Remaining contours less than a minimum length are also removed (stages 1 though 4).

The contour-map and the original filtered image are then passed to an "edge sensitive" subsampling procedure. This procedure extracts sample points from the source image at regular grid intervals. In fact, for each sample point, the *mean*[76] value from a small pixel neighbourhood around the sample point is recorded[77]. If an edge point intersects a sample point, then that sample point is *relocated* to a pixel "near" the edge using simple logic that may be duplicated at the decoder. This is done because we do not wish to sample intensity values *on* contours, but rather just next to them.

All three components may be coded separately. The texture component consists of a $1 \times n$ arrangement of 8×8 blocks, where n equals the number of blocks. These blocks are coded by DCT (Fig. 5.4-b). In addition, a binary *block-map* is required to indicate the locations of texture blocks (Fig. 5.4-e). Assuming texture blocks make up a small fraction of the image, the block-map may be efficiently coded by run-length encoding. The contours from the contour-map are coded using *differential chain codes* (Appendix C), and the smooth component is coded, with little loss, by a waveform technique (we chose DCT).

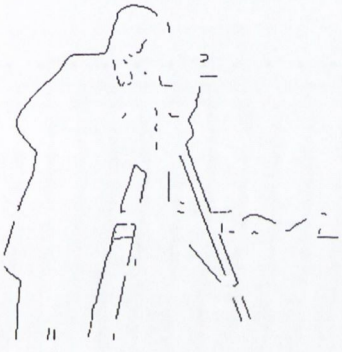
[76] Except near contours where the median value is taken.

[77] The reason for not sampling directly from a low pass filtered image is explained in section 5.4.3.4.

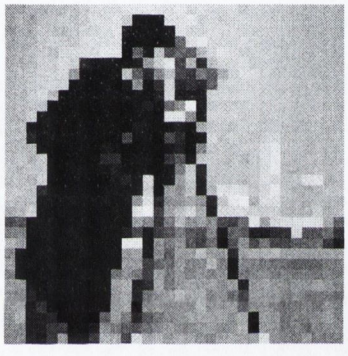
(a)

Fig. 5.4 Components of the scheme. Original image (a) , texture (b), contour-map (c) and smooth (d). Also displayed is the block-map (e) representing the locations of texture blocks in the image. For display purposes (a) and (c) have been reduced by a factor of 2 while (b) , (d), and (e) have been enlarged by a factor of 4 (in this case the subsampling factor is 8).

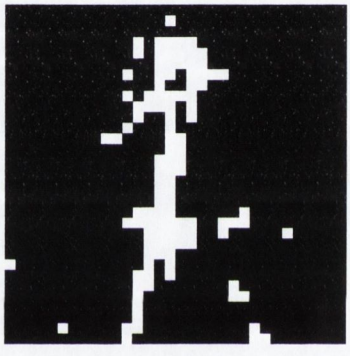
(b)



(c)



(d)



(e)

5.4.2.2 Decoder Overview

A block diagram of the decoder is presented in Fig. 5.5. An encoded image is reconstructed as follows. All three components, and the texture block map, are first decoded. An *empty* image, with dimensions equal to that of the final image, is created and populated with the sample points from smooth component. *Texture blocks* are then inserted into the image at the appropriate locations (Fig. 5.6).

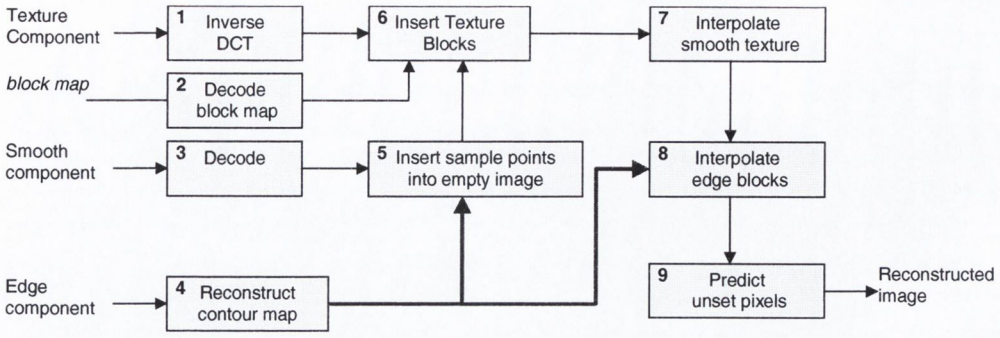


Fig. 5.5 Hybrid FBTC scheme decoder block diagram

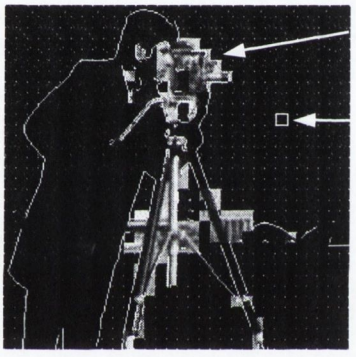


Fig. 5.6 Sample restoration after insertion of smooth component samples (very small dots) and texture blocks. Edge locations have also been indicated.

After insertion of the texture blocks (if any), the unknown (missing) pixel values between the smooth samples may be interpolated. The term *smooth block* is used to refer to cells within the partially reconstructed image with known sample points in each corner. Blocks that have contour pixels passing through any side, or fully contained within the block, are referred to as *contour blocks*. Simple *bilinear interpolation* may be used to reconstruct the pixels in isolated smooth blocks, i.e. blocks that are not adjacent to texture blocks. Where a smooth block abuts a texture block, extra samples from any abutting sides are also used in the interpolation process (Fig. 5.7).

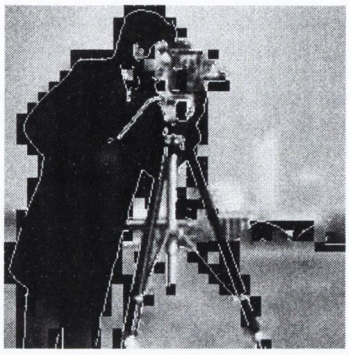


Fig. 5.7 Sample restoration after interpolation of smooth component. Edge locations have also been indicated.

Once the smooth blocks have been interpolated, contour blocks may be processed. This involves using the available sample points and taking information from adjacent blocks, most of which will have been generated by interpolation, and predicting the unset pixels, up to the contour boundaries (Fig. 5.8). Finally, any unset pixels, including pixels isolated from the interpolation process by contours, and contour pixels themselves, are predicted from their neighbouring pixels.

(a)

(b)

Fig. 5.8 Sample restoration after interpolation of contour blocks (a) and with contour pixels estimated (b).

5.4.2.3 Summary of Properties

From the preceding discussions, we can make some initial comments on the general properties of the proposed scheme.

1. In the absence of the contour component and/or the texture component, the smooth component may still be reconstructed, albeit with considerable distortion. Nevertheless, a "minimum service" is facilitated by the scheme. For a given encoder

quantization setting, the "minimum picture quality" obtainable from the smooth component alone will be dependent on the subsampling factor.

2. Interpolation will be contained to pixel cells (blocks) with dimensions at most equal to the subsampling factor plus one. If the subsampling factor is 8, then the problem is reduced to a number of 9×9 interpolation problems (with an overlap of one pixel between cells).

3. As the pixel values from reconstructed texture blocks are used to aid the interpolation process in neighbouring smooth and contour blocks, texture blocks should be encoded with a "good" quality level, i.e. with little distortion, or errors in their reconstruction will propagate to adjacent smooth or contour blocks.

4. The problem of multiple contours passing through a single block, and the implications for interpolation within that block have not been addressed by the scheme (addressed in section 5.4.4.2).

5. Contours will never be *truly sharp*, as contour points occupy actual pixel positions in the image (rather than inter-pixel spaces).

In the following sections, we describe the encoder/decoder stages in more detail and treat any issues overlooked by this brief overview.

5.4.3 The Encoder

In this section, the theory and implementation of the encoder is presented in more detail. We shall expand on the overview provided in section 5.4.2, and examine theoretical implications of the proposed design for the objectives listed in section 5.4.1.

5.4.3.1 Pre-processing

As with many FBC schemes, the performance of the coding scheme is influenced by the *quality* of the input signal (image). Noise in the image may cause errors in edge detection [115], and inaccuracies in the sampling of smooth and texture components. It is therefore advantageous to pre-filter the source image to remove noise, prior to encoding. In addition to noise removal however, it may also be advantageous to *simplify* or precondition

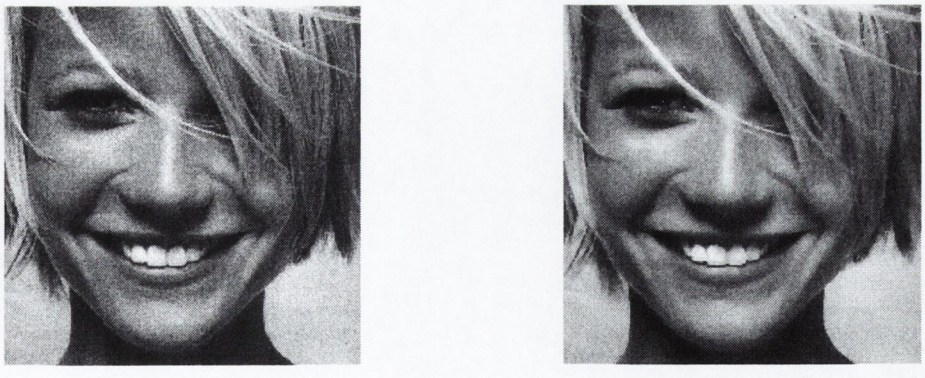
the input image to make it more amenable to subsequent processes. Simplification in this context may involve the removal of small objects, elimination of "non-contour" high frequencies, and/or the emphasis of contours. Noise removal and image simplification may be combined steps or carried out independently.

Based on the actions of the principal encoding procedures, the proposed scheme is likely to perform better, if the input image exhibits the following features:

1. **Distinct edges**. Any pre-processing must preserve, or preferably enhance, contour clarity. Blurred edges may give rise to gaps in contours, leading to texture bleeding in the decoder, and inefficiencies in contour representation (§5.4.3.2).

2. **Smooth texture close to contours**. In smooth image areas, it is assumed that there is little intensity fluctuation for a small distance from contours. Thus, if a pixel adjacent to a strong contour is sampled, it should have a similar value to pixels within a small neighbourhood from the contour.

Image pre-processing should therefore take the form of an edge preserving smoothening filter. This essentially means that the input image should be low pass filtered in smooth areas without "corrupting" contours. Fig. 5.9 illustrates edge sensitive smoothing.



(a) (b)

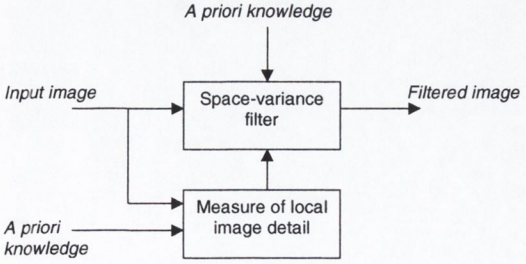
Fig. 5.9 Illustration of edge preserving smoothing filter. Original image (a) and filtered image (b). Notice that fine blurring is apparent but strong edges remain.

Noise filters generally fall into one of three categories [10]: linear shift invariant filters, non-linear filters, and adaptive filters. Linear filters, such as weighting averaging filters or linear minimum mean square error filters, are essentially low pass filters and thus attenuate

high frequencies. Such filters are unsuitable for our application as edges, being high frequency in nature, are blurred. Our attention therefore, turns to non-linear and adaptive methods.

In the selection of a noise filter, a balance needs to be struck between the benefits for subsequent processing and the additional computational complexity introduced by pre-processing. One of the most simple non-linear filters is the *median* filter, where the grey level of a pixel is replaced with the median of a neighbourhood around it [15]. This method effectively removes noise "spikes" in smooth areas while preserving the contrast of strong step edges. Textures and very thin line edges can become quite distorted however (lines can be lost). If such distortions are tolerable, or not relevant for a particular class of image, or given application then, median filtering may be appropriate as it is one of the less complex methods.

Another approach is employ space-variant [9 p536] filtering, where the filter characteristics *adapt* to the local characteristics of the input (Fig. 5.12). One such filter if the *adaptive Wiener filter* [9, p536]. Although, this filter may be implemented in different ways, generally, regions of low variance (i.e. smooth areas) are filtered more, while regions of high variance (edges and texture) are filtered less. While this approach effectively removes noise in smooth areas, the filter is effectively turned off across edges. Thus, noise in the vicinity of edges remains, and this is a problem for subsequent edge detection. Nevertheless, it is reasonably effective as a *one-pass* approach, i.e. as a solution that is applied in a single iteration.



Z_1	Z_2	Z_3
Z_4	Z_5	Z_6
Z_7	Z_8	Z_9

3x3 Sliding Kernel

Fig. 5.10 Block diagram of an adaptive image filter (reproduced from [9]) and a filter kernel

Operating on a *sliding neighbourhood*[78] the adaptive Wiener filter is computed as follows. First, the local mean and variance of the neighbourhood are estimated:

$$mean \quad \bar{z} = \frac{1}{N}\sum_{i=1}^{N} z_i \tag{5.1}$$

$$variance \quad \sigma_z^2 = \frac{1}{N}\sum_{i=1}^{N} (z_i - \bar{z})^2 \tag{5.2}$$

where the terms z_i represent the pixels of the neighbourhood, and N is the total number of pixels in the neighbourhood. These estimates are then used to calculate the output pixel $\hat{z}_c$:

$$\hat{z}_c = \bar{z} + \left(\frac{\sigma^2 - v^2}{\sigma^2} \right) (z_c - \bar{z}) \tag{5.3}$$

where v^2 is the noise variance and z_c is the centre pixel of source neighbourhood (corresponding to z_5 for a 3×3 neighbourhood, see Fig. 5.10). Note that the mean and variance are updated for each pixel. If the noise variance is not known, it may be estimated as the average of all the local estimated variances. The noise filtered image is used as the input image for the three main processes to follow: texture extraction, contour extraction, and smooth extraction.

5.4.3.2 Contour Extraction & Coding

Contour extraction, filtering and coding has the potential to be one of the most computationally expensive stages in the encoding process. A balance should be struck between the complexity of the edge detection, linking, and coding methods on one hand, and the benefits in terms of improved compression efficiency and/or improved reconstruction quality on the other. A number of edge detection operators were evaluated in the context of the proposed coding scheme, the results of which may be found in chapter 6. The *Sobel* operator emerged as a favourable option, for its simplicity and the "quality" of the resulting edge maps.

[78] In a sliding neighbourhood operation, each output pixel is generated by convolving a filter kernel with the corresponding pixel in the input image. The "neighbourhood" is determined by the size of the filter kernel, e.g. 3x3. The term "sliding" refers to the fact that each output pixel is generated by sliding the kernel over the image.

Contour Detection

Edges are detected with the Sobel operator by separately convolving the input image with the two Sobel kernels (see below). The two edge-maps produced by these operations approximate the first-order partial derivatives in the horizontal and vertical directions. Thus, horizontal edges are detected by:

$$H_h = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix} \tag{5.4}$$

and vertical edges are detected by:

$$H_v = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}. \tag{5.5}$$

These two edge-maps will consist of both positive and negative values representing rising or falling edges respectively. The *absolute* values of the "pixels" in the edge-map will reflect the strengths of the edges they correspond to. A combined edge map may be computed by [9]:

$$E = \sqrt{(I * H_v)^2 + (I * H_h)^2} \tag{5.6}$$

where H_h and H_v are defined in (3.1) and (5.5) respectively and I is the source image. A binary edge-map is generated from E, by thresholding and edge thinning (see [9 p476]). The threshold chosen for the generation of the binary edge map will directly influence the number of edge points produced in the edge-map, and hence the compression ratio of the scheme[79].

Ideally, all contours that *look* connected in the original image should be connected in the binary edge-map. Gaps in contours, arising due to noise in the input image, or the

[79] The total compression ratio will also be determined by smooth and texture components.

proximity of the edge strength to the threshold value, may lead to *bleeding*[80] in the reconstructed image (see Fig. 5.11). Furthermore, contour *start points* are much less efficient to code than *connected* contour pixels (appendix C). It is therefore advantageous for coding efficiency to have an edge-map containing mainly long contours rather than many short ones. Thus, in the interest of reconstruction quality and compression efficiency *edge-linking* is employed to connect broken contours.

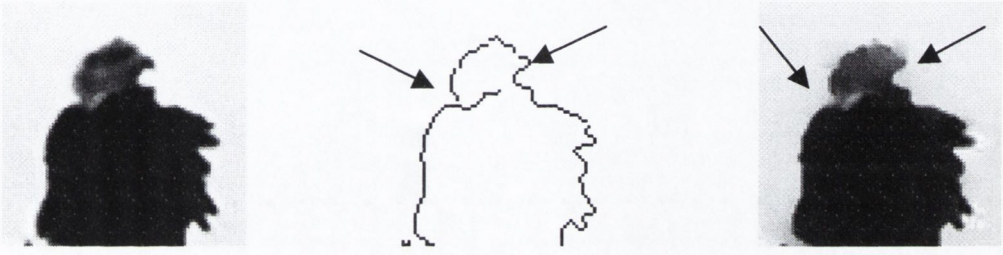


Fig. 5.11 Example of *bleeding* from gaps in contours. Original image (a), its edge-map *without* edge-linking (b) and its reconstruction (c), with obvious bleeding (arrows).

Edge-linking is performed by identifying all contour endpoints, and linking any endpoints separated by a single pixel[81]. To protect against false linking, pixels should only be linked if the gradient across the link pixel exceeds a threshold. This step was omitted in our investigations however, in the interest of complexity reduction.

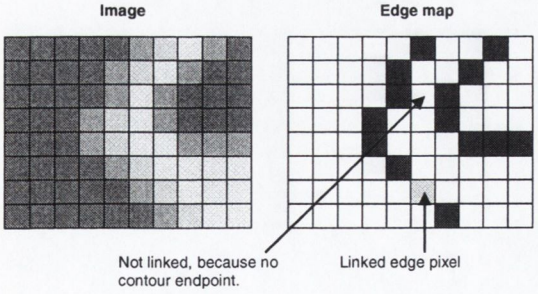


Fig. 5.12 Illustration of edge-linking.

Following edge-linking, all contours less than a minimum length are removed. This is done to remove short contours that require a comparatively large number of bits to code, yet

[80] Bleeding is where a grey level value appears to "leak out" from a break in a contour. See section 5.4.4.2.

[81] Diagonal distances are considered to be one pixel.

don't contribute very much in perceptual terms. The assumption of "perceptual significance" is based on the fact that the HVS is less sensitive to *small* objects than it is to big objects [109]. It is assumed that short contours may be removed without loosing perceptually significant image features[82].

The Sobel operator produces an edge-map in which *edges* occupy pixel positions rather than inter-pixel spaces. These *edge pixels* approximately locate the *steepest* part of their respective edges. Thus, no edge may be "perfectly sharp", as its transition must be at least one pixel wide. In practise, we do not consider this to be a significant problem. Our experiments (chapter 6) have indicated that the distortions introduced by this approach are perceptually insignificant in the context of the other distortions present. In fact, pixel sharp edges often have an *artificial* look about them, particularly when they are angled just off the horizontal or vertical axes[83], where *Vernier offset* (§3.2.7.3) may be perceived.

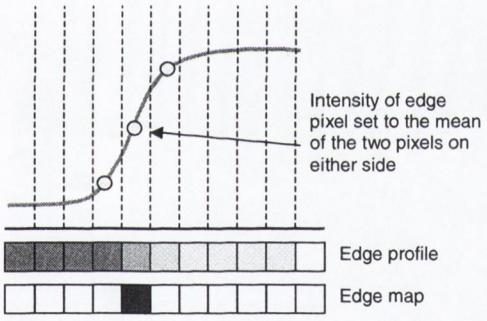


Fig. 5.13 Illustration of approximation of an edge pixel (1-D case) by the two pixels on either side.

In the proposed scheme, *all* edges are considered to be *one* pixel wide. It is assumed that the value of edge pixels themselves may be satisfactorily approximated by the average of the values at either side of them (see Fig. 5.13). The validity of this assumption on a section of the Lena image is demonstrated in Fig. 5.14. There are of course intensity distributions where this approximation yields distorted results, but as we are not concerned with perfect reconstruction in very low bit-rate coding, and reconstructions based on this assumption are generally satisfactory (see chapter 6), the advantages of its simplicity outweigh its fallibilities.

[82] Although, without knowledge of the image content, this assumption is precarious.

[83] This is a recognised problem in image synthesis, where steps are often taken to smoothen along sharp contours.

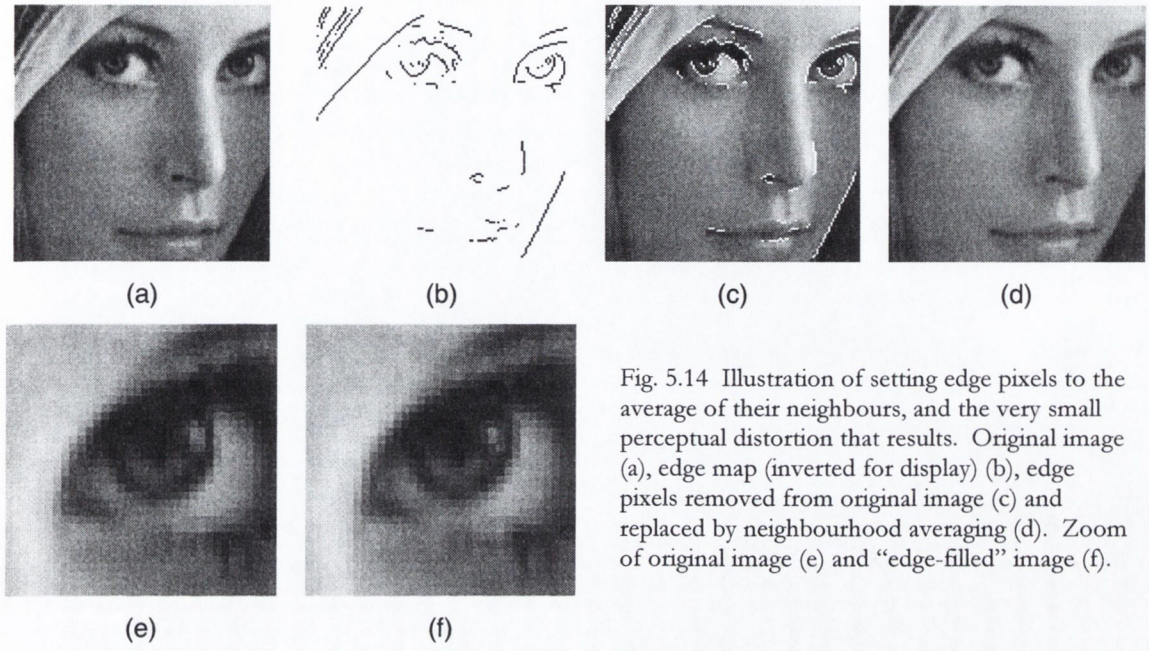


Fig. 5.14 Illustration of setting edge pixels to the average of their neighbours, and the very small perceptual distortion that results. Original image (a), edge map (inverted for display) (b), edge pixels removed from original image (c) and replaced by neighbourhood averaging (d). Zoom of original image (e) and "edge-filled" image (f).

Contour Coding

The contour map is coded by extracting each contour separately and coding them by *differential chain coding* (see appendix C). This method works best on long contours, as the cost of coding a start point is much larger than that of a contour point (see Fig. 5.15). Contours may be sorted according to their length, and shorter contours discarded should bit-rate requirements demand. Assuming that long contours originate from *large* objects in an image, and noting that the HVS is generally more sensitive to larger objects in a scene[84][109], rate-control based on this strategy has a rudimentary perceptual basis. Unlike sketch based methods, omission of contours in the proposed scheme simply results in a blurring of the associated edges.

The estimated bit rate for contour coding with differential chain codes is the same as that reported by Carlson [87] & Desai [86]; 9 bits per start point, and a conservative 1.3 bits per contour point (Fig. 5.15).

[84] Up to a point.

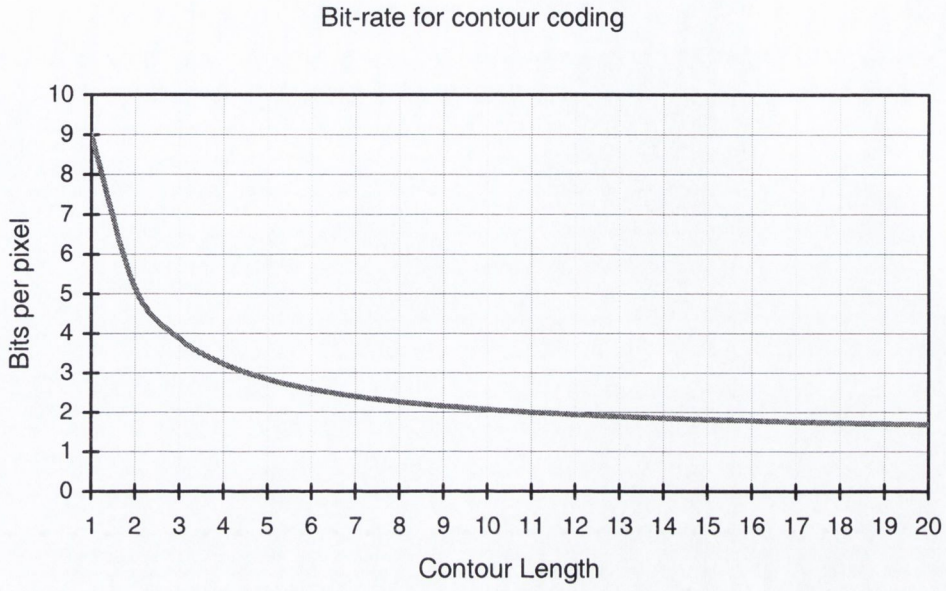
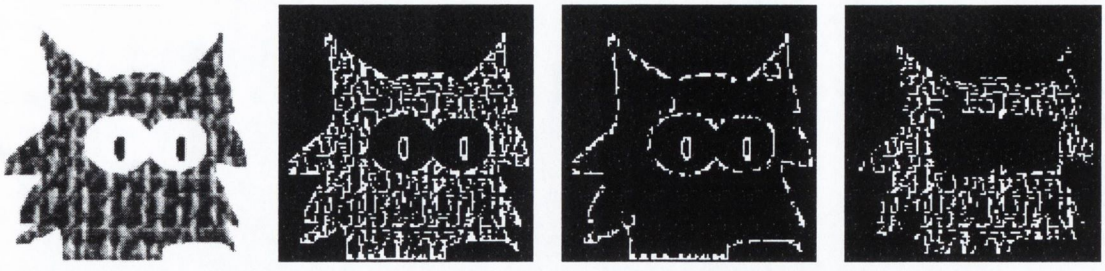


Fig. 5.15 Bit-rate for contour coding as a function of contour length.

5.4.3.3 Texture Extraction

In texture extraction, the objective is to locate regions of high variance that do not correspond to strong edges. The principal requirements of this procedure are fast execution and minimal false positives (including the misclassification of strong edges). In other words, we require a fast algorithm that will not incorrectly identify contours as texture. Such misclassifications would likely lead to coding inefficiencies where more bits would be required to represent a strong edge in "texture form", than would be required by the sketch representation.

In the context of the component model, we define texture as *closely spaced edge-like features.* We refine this definition to exclude edges that are surrounded by smooth texture on at least one side. Fig. 5.16 illustrates the distinction between *true* edge features and texture features.



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

Fig. 5.16 Illustration of edge/texture discrimination. Original image (a) and its edge-map (b) including both contours and texture. Contour-map (c) after texture extraction (d).

Thus texture classification needs to make use of two properties:

1. The local variance in the regions about edges.

2. The proximity of edges to each other.

To calculate the variance however would be unnecessarily complex and yield superfluous information. We therefore define a quantity called the *texture factor*, which may be considered related to the *local variance*, and computed by a number of simple steps, including *linear filtering* with a 3×3 neighbourhood kernel.

Z_1	Z_2	Z_3
Z_4	Z_5	Z_6
Z_7	Z_8	Z_9

3×3 kernel

$$\textit{mean: } \bar{z} = \frac{1}{9}\sum_{i=1}^{9} z_i$$

$$\textit{variance: } \sigma_z^2 = \frac{1}{9}\sum_{i=1}^{9} (z_i - \bar{z})^2$$

Fig. 5.17 Equation for calculation of *local variance* using a 3×3 kernel.

We define the texture factor α, below (see also Fig. 5.17).

$$\alpha_z = \frac{1}{9}\sum_{i=1}^{9} |z_i - \bar{z}_i| \qquad (5.7)$$

Note that rather than use the mean value of the neighbourhood, centred about z_5, we use $\bar{z}_i$, which is the mean value centred about the *ith* kernel entry (and thus extends outside the kernel for $i \neq 5$). In practical terms, the texture factor is calculated a follows. An averaging filter is applied to an image using a 3×3 neighbourhood, and the result is subtracted from the original image. We take the absolute values of the pixels in this image which gives us an *indication* of the energy nature of the signal, but it is very susceptible to noise[85]. The image is therefore filtered again with an averaging kernel. Thus, representing the averaging filter by H, and the source image by I we generate a *texture factor image* by

$$\alpha = \left| I - (I * H) \right| * H \tag{5.8}$$

where

$$H = \frac{1}{9} \times \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} \tag{5.9}$$

The texture factor image is then used to generate a binary image that may be efficiently processed with *morphological operators*, to extract spatial information about the texture. We define a threshold t, and generate a binary image from α according to

$$B(x,y) = \begin{cases} 1 & \alpha(x,y) > t \\ 0 & otherwise. \end{cases} \tag{5.10}$$

Set pixels in B will equate to pixels with a proportionally high texture, and conversely, unset pixels will equate to smooth areas. It is in fact smooth areas that we wish to identify as they effectively exclude edge features close to them from being considered as texture. We create a mask, to "exclude" contours, by inverting and *dilating* B (dilation is denoted by the symbol $\oplus$, and erosion by $\ominus$). We call this the *smooth mask*, M_s.

[85] We allow for imperfect noise reduction in the pre-processing stage (5.4.3.1).

$$M_s = B^{-1} \oplus S \tag{5.11}$$

where S is a 3×3 structuring element consisting of all ones. The objective here is to create a mask that envelops contours, thereby excluding them from consideration as texture. Any edge features remaining in the contour-map after masking with the smooth mask may be loosely deemed to lie within *busy* image regions. Fig. 5.18 illustrates the various stages of processing involved in texture extraction.

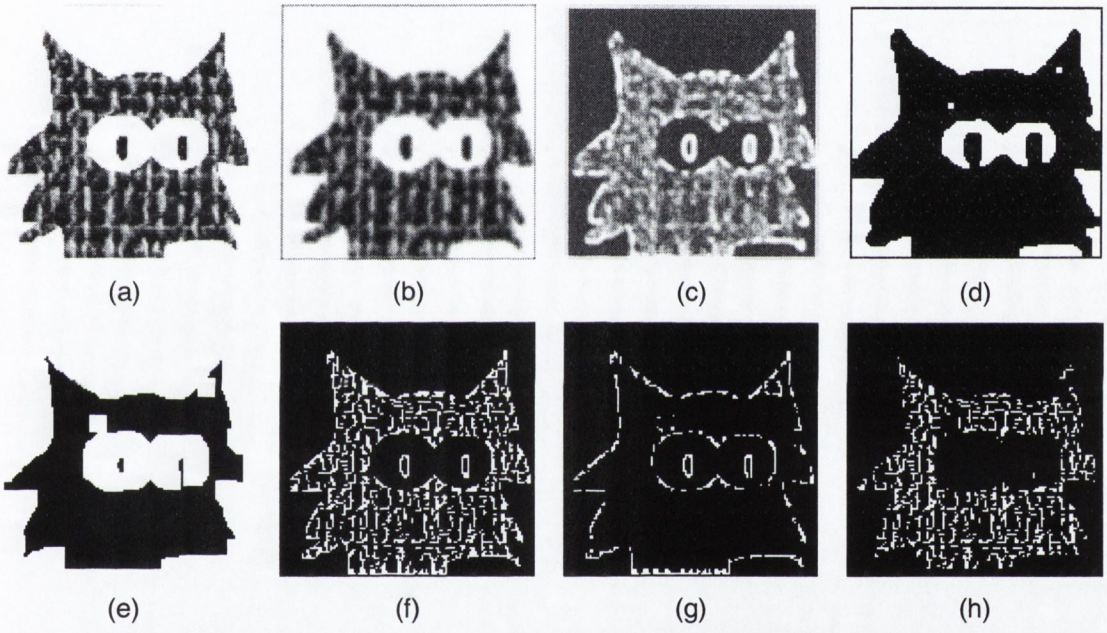


Fig. 5.18 Illustration of various stages of texture extraction Original image (a), after averaging filter (b), and texture factor image (c). Smooth mask (d), and after dilation (e). Original edge map (f), with contours (g) and texture (h) identified.

A logical (bitwise) AND of the smooth mask with the edge map is sufficient to separate contours from textures, while texture edges may be extracted by a logical XOR of the filtered contour map with the original edge map. In addition to finding texture edges we also wish to find *busy edges,* i.e. closely spaced contours. Busy edges will be classified as texture, but may not have been found by the above procedure. To find busy edges we perform *closure* on the original edge map (dilation followed by erosion), and XOR the result with the original edge map. This leaves behind the gaps between any closely spaced edges. This image may be logically ORed with the texture image to produce a final texture map (Fig. 5.19). Thus, where C is the filtered strong edge map, E is the original edge map

including texture, T is the extracted texture map, E_b is the busy edge map, T_f is the final texture map (inc. busy edges), and S is a 3×3 structuring element consisting of all ones:

$$C = (M_s) \text{ AND } (E) \tag{5.12}$$

$$T = (C) \text{ XOR } (E) \tag{5.13}$$

$$E_b = ((E \oplus S) \ominus (E)) \text{ XOR } (E) \tag{5.14}$$

$$T_f = (E_b) \text{ OR } (T) \tag{5.15}$$

(a) (b)

Fig. 5.19 Busy edges (a) and the final texture map (texture + busy edges) (b).

When a final binary texture map has been generated, it may be used to determine if blocks in the source image should be texture coded (stage 6 of the encoder in Fig. 5.3). This is simply a matter of counting the number of set pixels in each block, and coding the block as texture if that number exceeds a threshold.

Texture blocks are coded by extracting an 8×8 block from the source image and appending it to the texture block image. In the *first generation* coder model, the texture block image consists of a $1\times n$ array of blocks, where n is equal to the number blocks. This image is then coded by DCT. Additionally, a binary *block-map* is recorded to note the location of each extracted texture block. This image may be run-length encoded.

5.4.3.4 Smooth Component Extraction

To extract the smooth component, the source image must be subsampled on a regular grid. Usually, before subsampling an image, it is necessary to low pass filter the image to avoid aliasing [73]. This is the same principal as band-limiting an analog signal prior to sampling. Fig. 5.20 illustrates the presence of aliasing in an image reconstructed from a non-band limited source.

(a) (b) (c)

Fig. 5.20 Example of an original image (a) subsampled by a factor of 3 and then upsampled with bilinear interpolation to the original image dimensions. In the case of (b) no low pass filtering was performed on the source prior to subsampling, whereas (c) was low pass filtered prior to subsampling.

Subsampling in this way however poses problems for the reconstruction of edges using just the subsampled image and *geometrical* edge information. Low pass filtering an image causes blurring around edges, which means that sample values taken close to edges will not reflect the abruptness of the intensity change across the edge. Fig. 5.21 and Fig. 5.22 illustrate this for the 1-D case.

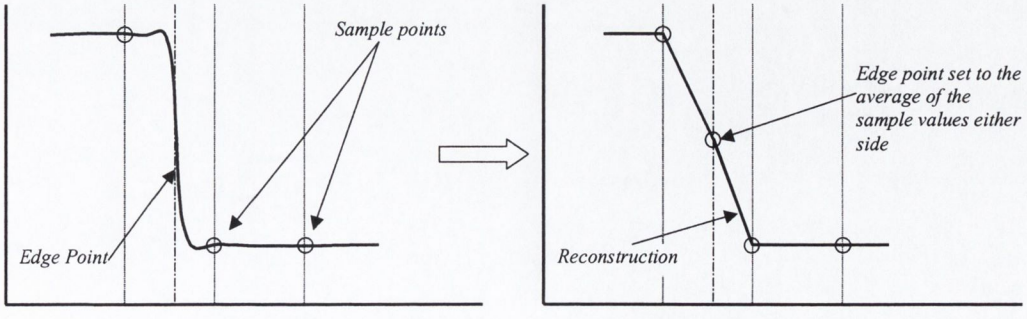


Fig. 5.21 Basic Reconstruction of a signal from an edge point and two sample points, *without* low pass filtering of source signal. The edge point itself has been set to a value midway between the sample points at either side (thus edges will never by *truly* sharp).

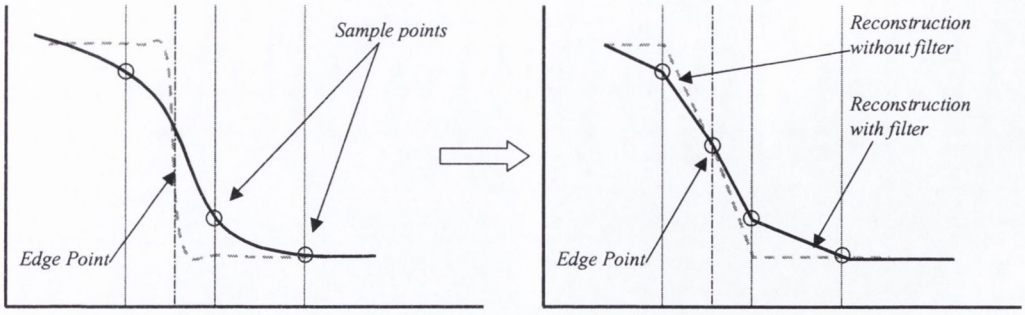


Fig. 5.22 Basic Reconstruction of original signal from Fig. 5.21, *with* low pass filtering of source signal. Notice the corruption of the edge *strength*.

While space-variant edge preserving low pass filters may be used to address this problem[86], in the interest of computational simplicity, a simple *local neighbourhood averaging* approach was taken that would allows for some degree of aliasing in smooth areas. Textured areas are effectively excluded from this problem as they are not subsampled.

Sampling

Sample points that lie within smooth areas have their values calculated as the average of a 3×3 neighbourhood about the sample point. For sample points where an edge pixel passes though the 3×3 neighbourhood, but not the centre of the kernel, edge pixels are simply masked in the averaging process. Where a sample point lies *on* a contour point, a special procedure must be employed to *relocate* that sample point to another location close to the edge. This action must be taken as edge values themselves are generally of little use in predicting the intensity of pixels either side of them, especially if the subsampling factor is large.

Relocating Sample Points that Intersect Edge Points

There are two principal decisions to be made in relocating sample points:

1. *Where* to relocate the sample point in the reconstructed image, as it cannot be located on the regular grid pattern.

2. *What* pixels to use in calculating the value of a relocated sample point.

[86] If an edge preserving low pass filter was used to pre-process the image then this problem doesn't arise.

The problem of *where* to relocate a sample point intersected by an edge pixel shall be treated first. On inspection of the problem we note that if a sample point intersects an edge point, then the next nearest sample point will be a distance equal to the subsampling factor from the edge. In other words, if a sample point is relocated to just *one* side of the contour, then the distance over which predicted values at the other side of the contour must span, will be comparatively large. Our confidence in such a prediction is likely to be quite low. Fig. 5.23 illustrates this for the 1-D case.

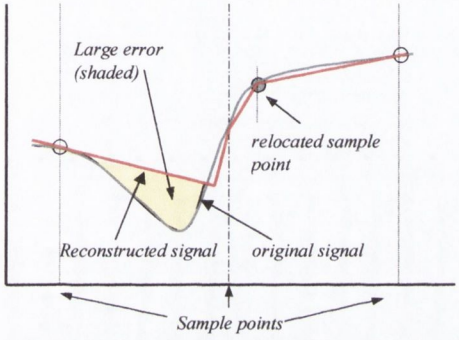


Fig. 5.23 Illustration of a large reconstruction error due to the relocation of a sample point lying on an edge, to just *one* side of the edge.

To remedy this, sample points should be recorded on *both* sides of a contour. Fig. 5.24 illustrates this for the 2-D case. The additional sample points cannot however be recorded in the subsampled image itself, and must therefore be stored separately. The modified encoder block diagram has been presented in Fig. 1.1 to reflect this change.

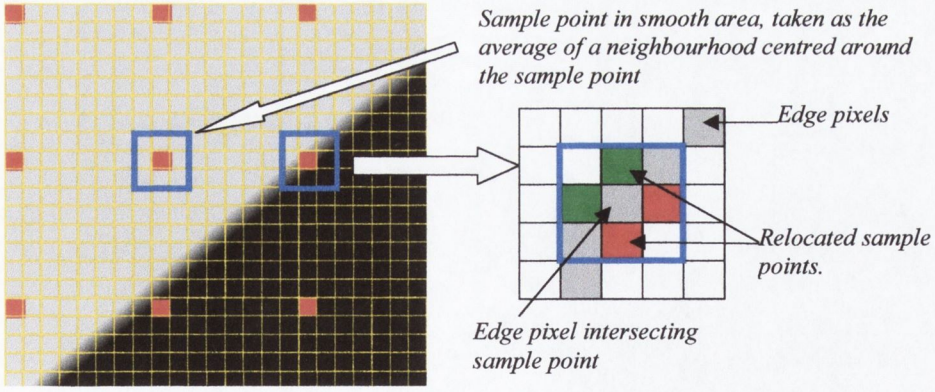


Fig. 5.24 Example of sample point relocation where an edge point intersects an sample point. Red pixels represent sample points on the same *side* of the contour, while the green pixels are on the other (opposite) side of the contour.

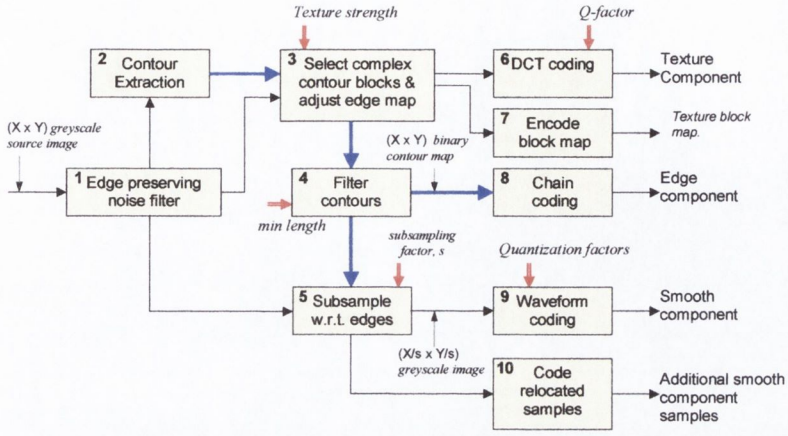


Fig. 5.25 Hybrid feature based transform coder (FBTC) – *first configuration*, with support for additional smooth component sample points outside the sampling grid.

As can be seen from the example in Fig. 5.24, there are four logical positions to which a sample point may be relocated (green and red squares). These sample positions lie in line with the next nearest sample points in each of the four directions (North, South, East, West). Depending on the placement of edge points however, only a subset of these positions may be valid for any given contour intersection. Furthermore, a strategy must be developed to decide which relocated sample points lie on the same side of a contour. We proceed to develop a strategy to relocate sample points without generating an overhead in storing their locations.

In the analysis of this problem, a number of possible pixel arrangements were found, where edge pixels intersect sample points. Some examples of the types of edge intersection that may occur are presented in Fig. 5.26. In most cases, horizontal and vertical sample points will be paired, representing the same side of an object in the image. To avoid redundancy, a single sample value should be recorded for all points lying on the same *side* of a contour.

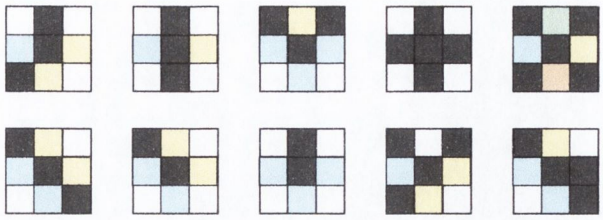


Fig. 5.26 Examples of different edge patterns in a 3×3 kernel (black) and possible sample point relocations. Sample point relocations of the same colour may be considered to belong to the same "object" in the image, and thus have only a single value recorded for all of them.

Thus, relocation is a matter of pairing sample points on the same side of a contour. Given the numerous permutations of contour patterns that may intersect the centre pixel of a 3×3 cell, a "perimeter scanning" method is used to find sample points on the same side of a contour (Fig. 5.27). Starting at *z4*, the perimeter of the 3×3 cell is scanned in a clockwise fashion. For each edge pixel encountered, a new group number is assigned to the cell position. The number of different group numbers occupying possible sample locations (Fig. 5.27) will indicate how many extra samples are required (0 to 3), and which samples belong to the same side of the contour. The decoder may apply this same algorithm, using the edge map, to retrieve extra samples and place them in the correct locations in the reconstructed image. Fig. 5.28 shows a section from a partially reconstructed image, where a sample point intersecting a contour has been relocated.

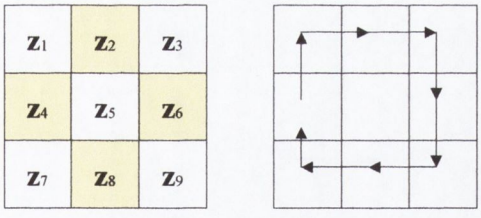


Fig. 5.27 Perimeter scanning in a 3×3 cell. Shaded positions are possible sample locations.

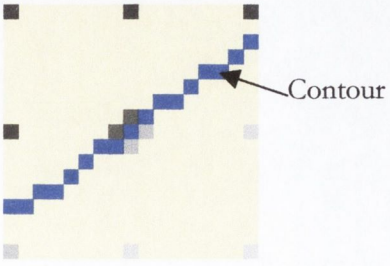


Fig. 5.28 Example of *sample relocation* in a section on an image. All four relocation positions have been used with two distinct groups. A total of one extra sample point is recorded outside of the subsampled image.

Choosing Values for Relocated Samples

In sub-subsection 5.3.3, we stated the assumption that if the location of an edge, and some sample values near the edge are known, we can often reasonably reconstruct a perceptually acceptable approximation of the edge profile. The term *near* of course, is vague. In practise, intensity variations in the immediate vicinity of edges tend to be *erratic*, but gradually become less so a small distance from the edge. If the subsampling factor is high (i.e. fewer sample points), then it would be preferable to avoid these localised characteristics and record a sample value more representative of the "general area away from the edge". This point is illustrated in Fig. 5.29, where the effects on the reconstructed signal of relocating a sample point lying *on* an edge to either *r1* or *r2*, one and two pixels from the edge respectively, are shown. In this case, *r2* represents a better option, as the gradient at *r1* is still quite large and unrepresentative of the signal further from the edge. On the other hand, if the sample points were more closely spaced, then sampling at *r1* would retain the *characteristic* of the edge.

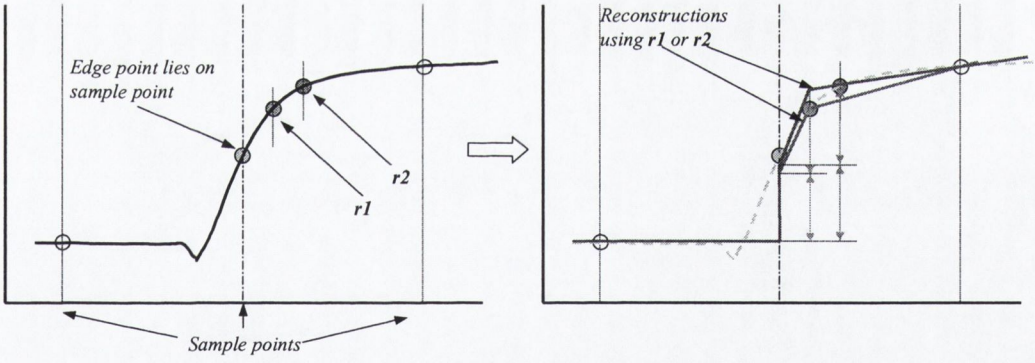


Fig. 5.29 Illustration of displacement of sample point lying on an edge point to *r1* or *r2*, and the effects on the reconstructed signal. Neighbourhood averaging has been omitted for simplicity. The edge pixel itself is set to the average of the sample values either side of it.

A logical way to estimate the "best" value for a relocated sample point would be to choose a value that minimises the reconstruction error. Unfortunately, this approach would not guarantee the most "perceptually pleasing" result. Due to *masking* by the HVS, errors in the immediate vicinity of strong edges tend to be less visible than those a little further away [103][109][110]. Thus, distortion *next* to an edge would have lower perceptual weighting than distortion a few pixels from the edge. This makes it difficult to estimate the

reconstruction error with the least *perceptible* distortion, i.e. we cannot simply minimise the *magnitude* the reconstruction error to arrive at the optimum choice for a sample value.

Furthermore, due to the possibility of the space between two sample points being intersected by more than one contour (see Fig. 5.35, page 144), the required data (to test reconstruction error) is not always available without performing reconstruction over a wider area. In the interest of simplicity and practicality therefore, we chose to simply mask edge pixels form the 3×3 neighbourhood of the relocated sample point, and take the *median* of the remaining pixels[87]. If a relocated sample point is paired with another sample point, then their values are averaged to arrive at the final sample value.

The canonical form of the smooth component is an approximation of a subsampled version of the source image. Due to the fact that no low pass filtering may have been performed on this image prior to subsampling, it is possible that with a large subsampling factor, and an averaging neighbourhood of only 3×3, that some image details will be lost completely.

(a) (b) (c)

Fig. 5.30 Example of an original 256×256 image (a) subsampled by a factor of 4 and then upsampled with bilinear interpolation to the original image dimensions (b). The same image, subsampled using edge sensitive subsampling and reconstructed using an edge map (356 bytes) and 159 additional (relocated) samples, is shown in (c).

The smooth component may be coded using a standard waveform coding technique. An important consideration in the coding of the smooth component is that very little distortion can be tolerated in the reconstructed smooth component, especially across edges. Each pixel in the smooth image may represent the intensity change associated with a significant structural feature. In a "normal" image, single pixels rarely contribute very

[87] This was found to produce better results than taking the *mean* of the masked neighbourhood.

much by themselves to the image as a whole. In the proposed scheme however, a single pixel in the smooth component may be the sole representative of a much larger area in the upsampled image (e.g. if it is enclosed by edges). Furthermore, while a small degree of distortion in the immediate vicinity of strong edges is usually masked in "normal viewing", if a distorted edge is "magnified", the distortions become more visible. The upsampling of the smooth component is akin to magnification, and thus preservation of edge integrity is more important than usual. The DCT was chosen to code the smooth component. This method was chosen to enable *re-use* of the DCT functions employed for texture coding, and maintain codec simplicity. Measures are taken in the decoder to remove the quantization noise introduced by the DCT at lower bit-rates (§5.4.4.1)

This concludes the description of the encoder. We proceed in the next section to discuss the reconstruction of the image form the coded components.

5.4.4 The Decoder

On decoding the three components[88], the decoder is faced with the task of reconstructing an approximation of the original image. In this section, discussions shall be based on the *first configuration* of the scheme, described in section 5.4.2. In this configuration, all three components are used to reconstruct an image. Depending on the source content, either texture or contour components may be absent, but every image will at least have a smooth component. Assuming all three components are used, there are five basic steps to reconstruction. The decoder block diagram of Fig. 5.5, has been modified indicate support for relocated samples in the smooth component, and presented in Fig. 5.31 for reference.

Decoder stages

1. Insertion of smooth component, and any relocated values, into empty image, R, except in texture blocks.
2. Insertion of texture blocks into R.
3. Interpolation of smooth component (using texture blocks if present)
4. Interpolation of *contour blocks.*
5. Prediction of unset pixels, including contour pixels.

[88] And any additional *relocated* sample points.

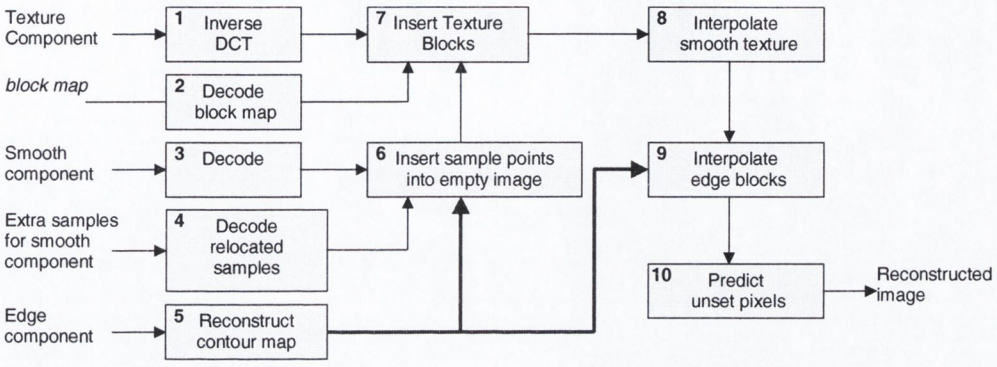


Fig. 5.31 Hybrid FBTC scheme decoder block diagram with support for additional, relocated samples in the smooth component.

5.4.4.1 Smooth and Texture Components

Following decoding of the smooth and texture components, it may be advantageous to filter either one or both images to remove compression artifacts. Fig. 5.32 shows a reconstructed smooth component image that has been compressed using JPEG. Compression artifacts are visible about strong contours. These artifacts become magnified following upsampling, and should therefore be removed if possible. Due do the small size of the smooth image, a space-variant (edge adaptive) filter may be used without incurring significant computational overhead (§5.4.3.1). The effects of pre-filtering the smooth component prior to upsampling and interpolation are investigated in chapter 6.

(a) (b)

Fig. 5.32 Example of a smooth component image (128x128) that has been compressed with JPEG (a). The same image after noise removal (b). Note that *information* is also lost by noise reduction.

Texture reconstruction is simply a matter of inserting texture blocks into an empty image. Texture should generally be compressed with a "good" quality setting, as both smooth and contour blocks that abut texture blocks, use the adjoining texture pixels to aid interpolation. Furthermore, highly compressed texture blocks are likely to introduce *blocking effects.*

Reconstruction of the smooth component involves interpolating between known sample values (see appendix B). Smooth blocks will always have at least four known pixels in each corner[89]. If no texture blocks abut, then reconstruction may be completed by simple *bilinear interpolation,* or more complex *bicubic interpolation.* The use of more sophisticated interpolation methods, that require larger regions of support, is problematic in the context of the proposed scheme, due to the fact that pixels should *not* be smoothly interpolated in the vicinity of contours. These areas are processed separately, using the edge map. There are two principal factors to consider in choosing an interpolation method:.

[89] We consider a block with a contour pixel in any one corner to be a contour block.

1. It is desirable to maintain the block based processing paradigm where possible, as this facilitates straightforward reduction of the problem to a number of independent parallel tasks.

2. It is typically around edges that bilinear and bicubic interpolation methods introduce their most visible artifacts. As it may be reasonably assumed that edges will be absent in the *reconstructed* smooth component[90], the advantages of superior but more complex methods are diminished in this context.

Another factor to consider in the selection of an interpolation method is the *availability* of pixel values in a support area larger than the current block. If any neighbouring block is a contour block, then some of its samples will represent different sides of a contour, and thus should *not* be smoothly interpolated. This fact may be accounted for by generating a modified contour block where the samples on the side closest to smooth block are duplicated across the contour. This approach incurs additional complexities that need to be weighed against the benefits in terms of picture quality. In the interest of computational efficiency and simplicity, only bilinear and bicubic methods were considered. In the case of bicubic interpolation, the 4 corner samples (i.e. 2×2 support) are duplicated to pad a 4×4 support. The true potential of bicubic interpolation is thus only partially realised by this approach. Ultimately this decision was a trade off between computational complexity and reconstruction *smoothness*.

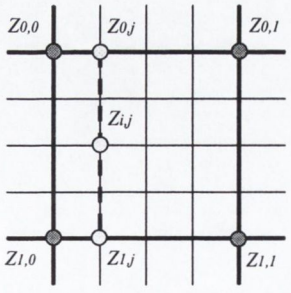


Fig. 5.33 Interpolation of unknown value $z_{i,j}$ from known samples $z_{0,0}, z_{0,1}, z_{1,0}$, and $z_{1,1}$.

For bilinear interpolation, an unknown pixel $z_{i,j}$ within a block is calculated according to the formulae:

[90] We refer to the upsampled smooth component and not the *smooth image*.

$$z_{0,j} = j \cdot z_{0,1} + (1-j) \cdot z_{0,0} \tag{5.16}$$

$$z_{1,j} = j \cdot z_{1,1} + (1-j) \cdot z_{1,0} \tag{5.17}$$

$$z_{i,j} = i \cdot z_{1,j} + (1-i) \cdot z_{0,j} \tag{5.18}$$

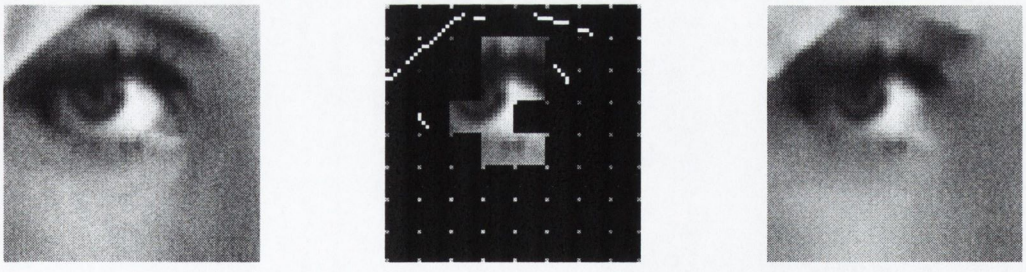
where the distance between known samples has been normalised . Thus for the example of Fig. 5.33, we have

$$z_{\frac{1}{2},\frac{1}{4}} = \tfrac{1}{2}\left(\tfrac{1}{4} z_{1,1} + \tfrac{3}{4} z_{1,0}\right) + \tfrac{1}{2}\left(\tfrac{1}{4} z_{0,1} + \tfrac{3}{4} z_{0,0}\right). \tag{5.19}$$

This simple separable calculation, which can be reduced to 4 multiplications and 8 additions, is evaluated for all the unknown pixels in the block. For details of bicubic formulae, see appendix B.

Neighbouring Texture Blocks

If a texture block abuts a smooth block then, in addition to the four corner points, there will be a complete boundary of additional known values to use in the interpolation of the smooth block. In fact, failure to smoothly "diffuse" the intensity distribution from a neighbouring texture block, into the smooth block, may cause blocking effects.



(a) (b) (c)

Fig. 5.34 Example of interpolating smooth blocks, using the intensity profile from the boundaries of neighbouring texture blocks where possible. Original image (a), empty image populated with sample points, texture blocks and contour lines (b), and reconstructed image (c). Subsampling factor is eight, images zoomed by 2.

5.4.4.2 Reconstructing Contour Blocks

A *contour block* is defined as any block that contains contour pixels. Like smooth blocks, the dimensions of a contour block are equal to the subsampling factor plus one. There will

therefore be an overlap of one pixel with all adjoining blocks. Contour block reconstruction is a problem of estimating, or predicting, the unknown pixels in the block, using the following information:

1. The geometrical contour information.

2. The four corner samples (more if relocated samples are present, less if relocation was not possible).

3. The pixels of any previously reconstructed adjoining blocks (i.e. the overlap).

There are numerous possible arrangements of contour pixels in a contour block. A number of representative arrangements are presented in Fig. 5.35.

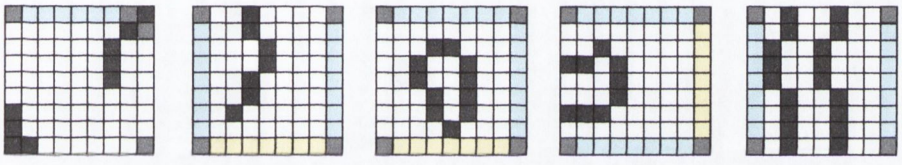


Fig. 5.35 Examples of different contour blocks. Grey squares represent known sample pixels, black squares represent contour pixels, yellow squares represent known pixels from adjacent (overlapping) texture blocks, and blue squares represent interpolated pixels from adjoining smooth blocks.

We shall treat the most simple kind first, illustrated by (a) and (b) in Fig. 5.35. In this case the side of a block is intersected by at most one contiguous contour. A contour might travel along the block boundary, but there will be at most two *unknown line segments* on any one boundary line.

The problem of "filling-in" the unknown pixels in a contour block essentially involves zero-order extension of the known pixels up to, but not through, the contour boundary (Fig. 5.36). The unknown pixel is then calculated as a weighted average of these projected pixels.

25	-1	-1	-1	100
-1	-1	-1	-1	-1
-1	-1	-1	-1	-1
-1	-1	-1	-1	-1
30	-1	-1	-1	80

(a)

0	0	1	0	0
0	0	1	0	0
0	0	1	0	0
0	0	1	0	0
0	0	1	0	0

(b)

25	25	-2	100	100
26	26	-2	95	95
27	27	-2	90	90
28	28	-2	85	85
30	30	-2	80	80

(c)

Fig. 5.36 Example of "filling-in" a contour block (a), with known values in the four corners, with respect to a contour map (b). Edge pixels in the filled block (c) are indicated with the value –2.

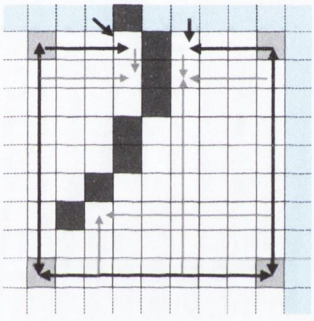


Fig. 5.37 Illustration of interpolating a contour block. Grey squares represent known sample pixels, black squares represent contour pixels, and blue squares represent interpolated pixels from adjoining blocks.

Fig. 5.37 illustrates the estimation of unknown pixels using neighbouring pixels. The black arrows indicate where "known" values are used to estimate a pixel, while the narrower grey arrows indicate where previously predicted pixels are used. Where a direct *line-of-sight* exists between any two known corner pixels, direct linear interpolation is used to predict the intermediate pixels. For all other pixels, a *weighting matrix* is generated to reflect the *confidence* of the prediction based on any one neighbouring pixel, and the final value calculated as a sum of these normalised weighted predictions. The confidence factor associated with any partial prediction is based on:

1. The distance of the pixel used in the prediction from the current pixel.

2. The confidence factor associated with that pixel.

Confidence Factors

For every predicted pixel, a confidence factor is calculated. If a previously predicted pixel is used to predict another pixel, its confidence factor is used to weight its contribution in the summation. The confidence factors of all pixels used in a prediction are scaled such that their sum is 1. The confidence that is placed on the estimate of an unknown pixel is inversely proportional to its distance from the nearest known pixel on which it is based.

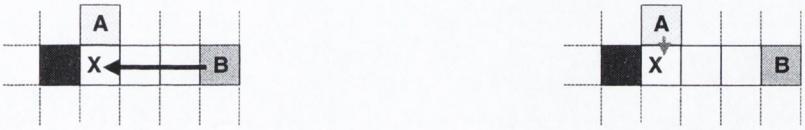


Fig. 5.38 Illustration of estimating a pixel X from a sample point B and a pixel from a neighbouring block. B is known and therefore has confidence factor of 1, while A is predicted and thus will have a lower confidence factor. A is closer to X than B however, which will "boost" its contribution to the estimate of X.

Fig. 5.38 illustrates the estimation of a pixel **X** from a pixel **A**, which is itself an estimate, and a pixel **B** which is a known sample point. As **B** is known it will have a confidence factor of 1. The pixel **A**, by contrast will have a confidence factor less than one, and so, despite its proximity to **X**, may even contribute less to the value of **X** than **B** will. The pixel will be calculated by:

$$X = \hat{s}_a A + \hat{s}_b B \tag{5.20}$$

where

$$p_a = c_a(1 - d_a) \tag{5.21}$$

$$p_b = c_b(1 - d_b) \tag{5.22}$$

and

$$\hat{s}_a = \frac{p_a}{p_a + p_b} \tag{5.23}$$

$$\hat{s}_b = \frac{p_b}{p_a + p_b} \tag{5.24}$$

and c_a, c_b, are the confidence factors associated with **A** and **B** respectively, and d_a, d_b are the distances of **A** and **B** respectively from **X**. Distances are measured in terms of the fraction of unit distance between two sample points. Thus in the example of Fig. 5.38, assuming a subsampling factor of 8, $d_a = \frac{1}{8}$, and $d_b = \frac{3}{8}$. Diagonal distances are approximated as one fractional unit.

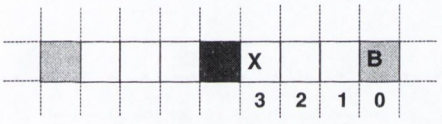


Fig. 5.39 Distances along a line. With a subsampling factor of 8, each block will be 9×9 pixels. The distance of X from B is 3 pixels or $\frac{3}{8}$ fractional units.

The values $\hat{s}_a$, and $\hat{s}_b$ represent normalised weighting factors for the two pixels on which **X** is based. Finally, a confidence factor c_x, must be calculated for the estimated value of **X**. This is calculated by

$$c_x = \max(p_a, p_b) \tag{5.25}$$

To illustrate the effect of these weighting calculations, we present an example below (see Fig. 5.40). Let us suppose, for simplicity, that **A** is based directly and solely on **C**, a known sample value. The confidence factor associated with **A** will be

$$c_a = c_c \cdot (1 - d_c)$$

$$c_a = 1 \cdot \left(1 - \tfrac{3}{8}\right) = \tfrac{5}{8}$$

Now, **X** will be calculated from

$$p_a = \tfrac{5}{8} \cdot \left(1 - \tfrac{1}{8}\right) = 0.55, \qquad p_b = 1 \cdot \left(1 - \tfrac{3}{8}\right) = 0.62$$

$$\hat{s}_a = \left(\frac{0.55}{1.17}\right) = 0.47 \qquad \hat{s}_b = \left(\frac{0.62}{1.17}\right) = 0.53$$

$$X = 0.47 \cdot A + 0.53 \cdot B$$

and the confidence factor for **X** will be:

$$c_x = \max(p_a, p_b) = \frac{5}{8}$$

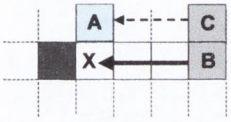


Fig. 5.40 Diagram for sample calculation above.

More Complex Contour Blocks

Contour block interpolation is further complicated if multiple block boundary intersections are permitted. Examples of such blocks are represented by Fig. 5.35-d & e. In such cases, *isolated* pixels are left unset following the interpolation process (see Fig. 5.41).

25	-1	-1	-1	100
-1	-1	-1	-1	-1
-1	-1	-1	-1	-1
-1	-1	-1	-1	-1
30	-1	-1	-1	80

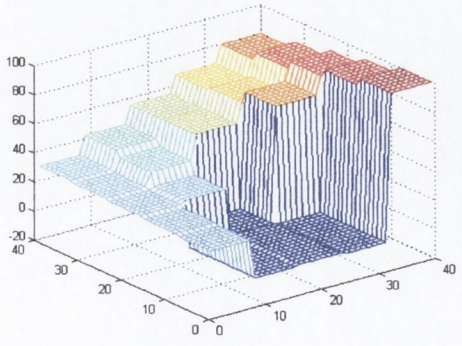
(a)

0	1	0	1	0
0	1	0	1	0
0	0	1	0	0
0	0	0	0	0
0	0	0	0	0

(b)

25	-2	-1	-2	100
26	-2	-1	-2	86
27	28	-2	58	72
28	32	38	47	58
30	33	37	41	45

(c)



(d)

Fig. 5.41 Example of filling-in a block where some pixels have been isolated from the interpolation process.

Isolated pixel may often be predicted by taking samples from a neighbouring block, but this is not always possible. Furthermore, the "enclosed" area may span multiple contour blocks. To rectify this situation, additional samples may be recorded by the encoder

wherever more than two separate contours intersect a block boundary (Fig. 5.42). This approach introduces a small computational overhead in the encoder, where the *grid-lines* of the sampling lattice must be scanned to detect potentially isolated pixels.

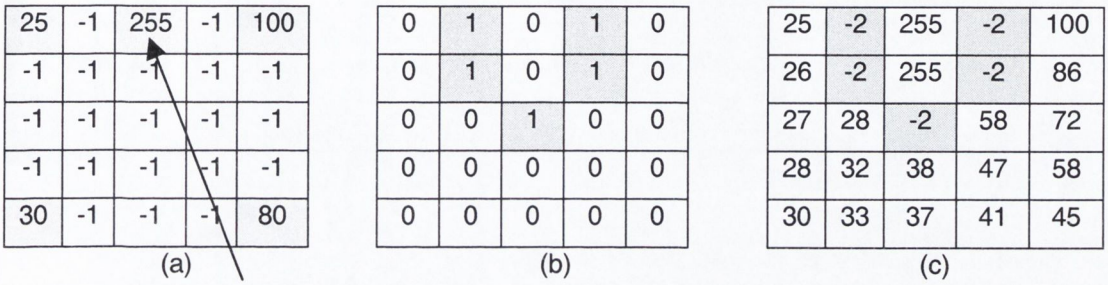
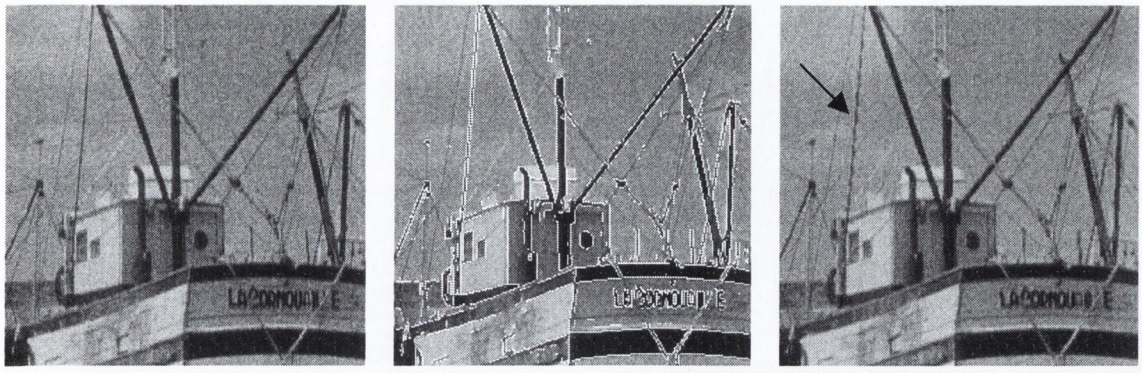


Fig. 5.42 Additional sample supplied for isolated pixels.

The final contour block type that presents additional complications is represented by Fig. 5.35-c. In this case, the contour lies totally within the block, creating a closed loop with isolated pixels. No block boundary crossing is made, or the crossing is such that there is no opportunity to record additional samples by the approach discussed above. Such situations require additional strategies to detect and cater for. This is the topic of the next section.

5.4.4.3 Filling unset pixels

Following interpolation of smooth and contour blocks, a number of pixels may remain unset. As edges occupy pixel positions rather than inter-pixel spaces (i.e. cracks), they must also be predicted. As stated in section 5.4.3.2, contour pixels are set to the average of their neighbouring pixels. A 3×3 kernel is centred about each contour pixel and after masking out any other contour pixels, the average of the remaining pixels is calculated. This approach produces a smoothening effect across contours, and may introduce apparent Vernier offset effect along contours that are slightly offset from the horizontal or vertical axes (see Fig. 5.43). It is suggested that the problem of Vernier offset distortion could be adequately removed by smoothing *along* contours.



(a) (b) (c)

Fig. 5.43 Illustration of Vernier offset effect in contour pixels predicted as the mean of neighbouring pixels. Original image (a), image with edge pixels unset (b), and set to the average of their neighbours (c).

Individual pixels or clusters of pixels isolated from interpolation by enclosing contours pose a particular problem for the proposed scheme. For these situations to be catered for, the encoder must predict their occurrence, and encode additional information for their solution. This information could take the form of explicit details regarding the locations of isolated pixels, plus additional sample points, or alternatively, only additional samples could be transmitted and an algorithm operating on the edge map could be used by both the encoder and decoder to determine where the extra samples are needed.

In the evaluation of the codec performance with various test image and subsampling factors (see chapter 6), the occurrence of *enclosed pixels,* of more than a few pixels, was noted to be relatively small. The additional computation required to detect such situations was therefore deemed to exceed the benefits. Any isolated pixels contained fully within a block are therefore set to the average of the surrounding pixels by the decoder. Features that are small relative to the subsampling factor are therefore lost by this approach. In practise, if the contours enclosing isolated pixels are two pixels or less from each other, and if the area is sufficiently large, these enclosed areas may be classified as *busy contours* and coded as texture (§5.4.3.3).

5.4.4.4 Post-Processing

As a final stage in reconstruction, the image may benefit from some post-processing. One of the notable characteristics of reconstructed images, coded with the proposed scheme

with a large subsampling factor (>4), is the absence of mid-range frequencies in smooth image regions. As most of the area in an image will generally be represented by the smooth component, large regions of the image may appear artificially smooth. A possible solution to this problem, as proposed by Kunt [80] for SIC, is to add a small amount of Gaussian white noise to the image. which has the effect of diminishing the synthetic appearance of smooth areas.

5.4.5 Alternative Configurations

In addition to the codec configuration described thus far, two additional configurations were considered. In this section, we briefly present these alternative configurations. These are not *failed* designs by any means, rather they each have their own advantages under different circumstances. The *relative performance* of the different schemes is generally influenced by the source image *size* and *content* (see chapter 6).

5.4.5.1 The Merged Architecture

The *first configuration* of the encoder uses a waveform coding method to code the subsampled smooth image. If the input image size is sufficiently large then the smooth-image may also be reasonably large[91] and thus exhibit the interpixel correlations typical of natural images. If the smooth image is very small however, it is more likely to be dominated by high frequency components that assume a more random nature. Such intensity distributions generally compress less efficiently. This is the classical case of *big* images compressing better than *small* images. Furthermore, assuming that a small smooth-image size reflects a large subsampling factor, every pixel in the smooth-image will represent information over a relatively large area of the reconstructed image. This means that distortion in the smooth image will have a relatively large spatial impact. This in turn prevents the smooth component from being compressed with a very lossy method. The advantages of the DCT for coding of the smooth component under these circumstances comes into question. In fact, DPCM may offer a more favourable approach. This led to the design of the merged architecture presented in Fig. 5.44.

[91] How large/small depends on the subsampling factor.

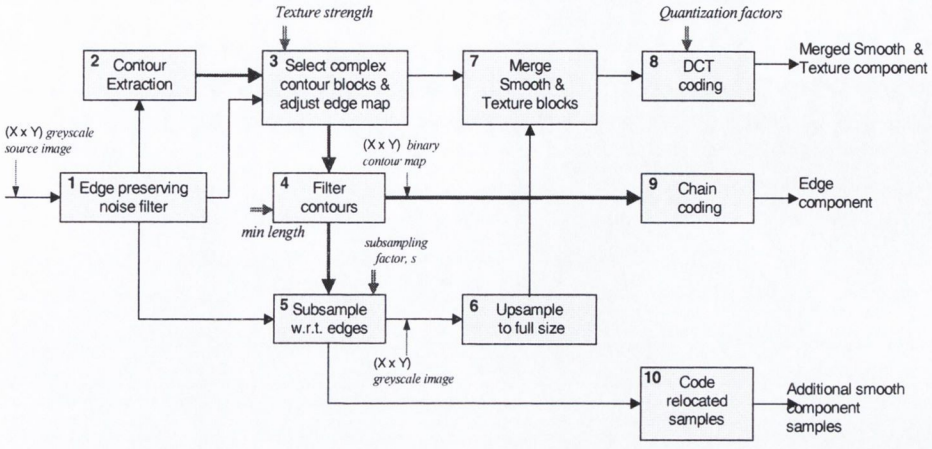
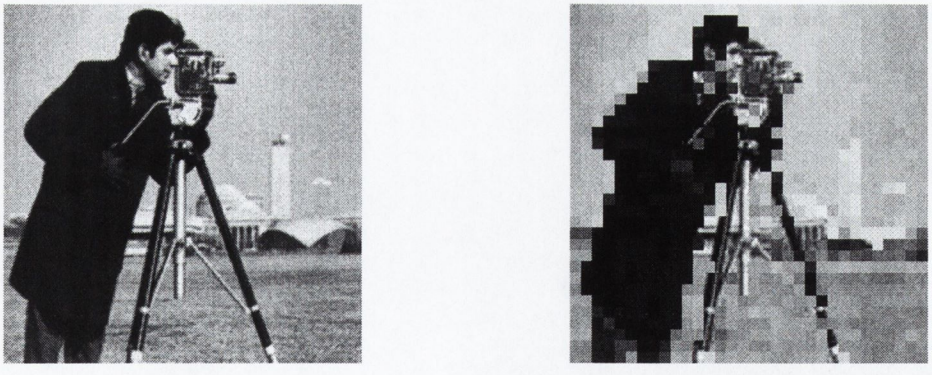


Fig. 5.44 Block diagram of the merged architecture.

The merged architecture was developed to more efficiently code small images (256x256 or less) with a subsampling factor equal to the DCT block size (usually 8). In the merged architecture, the samples from the smooth component are upsampled and inserted back into the full-size texture image. Thus, for a subsampling factor of 8, each sample from the smooth component will occupy 64 bits. Unlike the first configuration of the codec, there will be no redundancy between smooth and texture components as they share the same *merged* image (see Fig. 5.45).



(a) (b)

Fig. 5.45 Example of an original image (a) and a merged image representing *merged* smooth and texture components.

In the merged representation, with a subsampling factor of 8, it is not necessary to record a block-map, indicating the locations of texture blocks. The decoder may simply detect texture blocks by determining if all pixels in the same block share the same value. If they do, then the block is a smooth-block, otherwise it is a texture-block. Smooth-blocks will compress efficiently as they will not generate *any* AC coefficients. They will be DPCM

coded along with the DC coefficients of the texture blocks. The performance of this architecture in comparison with the first configuration is presented in chapter 6.

5.4.5.2 Contour-Smooth Scheme

The final architecture that was investigated may be viewed as a special case of the first configuration, but we discuss it separately here as this "mode" outperforms the scheme based on the complete three component model for particular types images. We call this architecture the *contour-smooth* scheme. In this approach, the texture component is omitted completely, and busy edges are classified as strong contours. A block diagram of the encoder model is presented in Fig. 5.46.

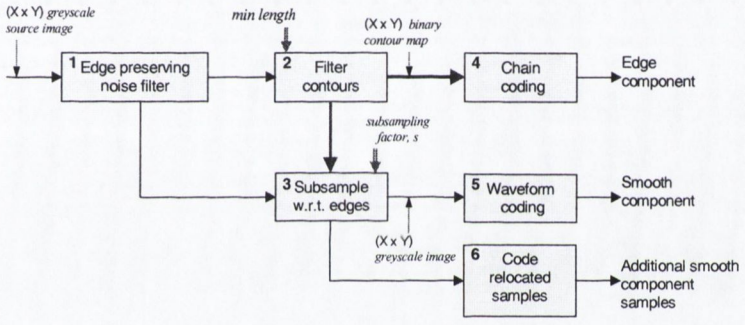


Fig. 5.46 Block diagram of the contour-smooth architecture.

Despite its simplicity, this approach may perform particularity well on large simple images containing large objects. No texture analysis is performed, and so there is explicit assumption that either texture is not present, or not important. The computational savings in omitting texture analysis are what makes this simplified scheme advantageous. Subsampling factors greater than 8 may be used where there is little variation in the intensity across objects. The total number of calculations required to interpolate pixels in the reconstructed image will be largely independent of the subsampling factor[92]. Coding results for this scheme are presented in chapter 6.

5.5 Properties of the Scheme

In this section we briefly state some principal properties of the proposed schemes, and examine whether or not the objectives of section 5.4.1 have been met. The discussions

[92] Different sampling grid sizes will influence the number of additional samples recorded however.

presented here shall be restricted to theoretical aspects. Chapter 6 expands upon these discussions with respect to experimental results.

5.5.1 Perceptual Basis

One of the principal motivations for the amalgamation of feature based and transform coding has been the good rendition of perceptually important strong edges. The three component model on which the proposed scheme is based explicitly models contour information. It was proposed that the shape/locations of contours communicate the most salient information for object recognition, where the texture at either side may be smoothly approximated. The validity of this proposal shall be investigated in chapter 6. In keeping with the perceptual model, the first configuration of the codec allows the smooth and texture components to be independently coded, with encoding methods and bit-rates matched to their individual statistics.

The "risk" of perceptually undesirable *blocking effects* has been dramatically reduced in smooth areas due to the smooth interpolation of reconstructed pixels. As the number of texture blocks in simple images is likely to small, those blocks that are retained may be coded with a "higher quality" setting, thereby reducing blocking effects in textured regions also. *Ringing* around strong edges is eliminated where they are coded by the contour component, but *bleeding* may occur if gaps in contours are not closed.

5.5.2 Computational Complexity

As discussed in chapter 4, one of the principal drawbacks of sketch based coders is their computational complexity. This complexity is present in both the encoder and decoder of such schemes. In the proposed scheme, the most computationally expensive component to process is the contour component (extraction, filtering, and coding). The initial noise filtering stage may be simplified or omitted if required. Smooth component sample extraction employs its own local low pass filtering in any case. It is principally for the benefit of contour detection that noise filtering is advantageous. The texture extraction process employs two stages of linear filtering which, operating on the full size image, cannot be overlooked in terms of computational complexity. The morphological operations are extremely efficient however, and operate on a data set only one eighth the size of the source image (bit oriented).

The smooth and texture components may be coded directly using the DCT. The complexity of this task is lessened by the fact that there will be relatively few texture blocks to code, and the smooth-image will be much smaller than the full size image. Thus the additional treatment of the smooth component in subsampling is offset by the fact that there will be much fewer pixels to code afterwards. There is no need to partially or fully reconstruct the image in the decoder, a requirement in most sketch based schemes.

The decoder complexity is also relatively small. The most complex stage of processing is the interpolation of contour blocks. In simple images, the number of contour blocks is likely to be relatively small, and so this complexity is not significant overall. Furthermore, in contrast with other sketch based coders, the regularity[93] of the interpolation problem greatly simplifies the estimation process.

The codec was implemented using *Mathworks™ Matlab R11* on a *Microsoft Windows 98™ SE* computer, with a 400MHz *Intel Celeron™* processor and 64MB of SDRAM. For a 640x480 image and a subsampling factor of 8, the first configuration of the codec encodes an image in between 2 and 5 minutes, while the decoder requires a similar time frame. Optimised implementations using a 3GL such as C++ would most likely be *considerably* faster, but no such investigations were made as part of this research.

5.5.3 Error Tolerance

The tolerance of the proposed scheme to transmission errors originates in the block based architecture, and independence of the three components. If a smooth block is lost, it may be approximated using the values of its neighbours and available contour information. If a texture block is lost, there will be a *smooth-sample* with which to approximate it (first configuration only), and if a contour is lost, there will simply be a blurring of the associated edge.

5.5.4 Comparison with JPEG

The proposed scheme may be configured to operate as a JPEG coder by simply selecting JPEG as the encode method in the *merged architecture*, and setting the texture sensitivity very low. In this way, JPEG comparable compression performance will always be available. At

[93] i.e. it is not a *scattered data* interpolation problem.

low bit-rates, with large simple images, the *first configuration* has theoretical advantages due to the *multiresolution* nature of the approach. Contour and texture blocks may be considered the lowest level (highest resolution) of a pyramid scheme, where the smooth component represents a higher level, and intermediate levels (mid-range frequencies) have been discarded. We provide results of all the schemes in comparison with JPEG in chapter 6.

5.5.5 Relationship to Existing Solutions

It is perhaps pertinent at this stage to comment on the relationship of the proposed scheme to similar existing solutions. As a hybrid scheme it adopts ideas from both its "parent" approaches, but the amalgamation of waveform and model based approaches is not novel in itself. In chapter 4 a scheme by Ran and Farvardin [100] (§4.3.1.5) is discussed that the authors describe as an amalgamation of waveform coding and feature based coding. Christopoulos [101] describes a scheme that he refers to as a hybrid SIC-DCT approach. In this approach an image is segmented into both regular blocks and arbitrarily shaped regions depending on the local image characteristics. Highly textured regions are coded by a block based DCT approach, while regions that may be better represented by a SIC approach are coded accordingly. These two approaches, while quite dissimilar to each other, both share commonality with the proposed scheme. The novelty and benefits of the proposed scheme may be appreciated in terms of the original manner in which waveform and feature based coding have been combined, yielding unique and advantageous properties for compression.

5.6 Summary & Conclusions

This chapter has presented a novel hybrid coding scheme, incorporating concepts from both feature based coding and block transform coding. We have discussed the motivations behind this amalgamation, and developed a solution to achieves the primary objectives of computational efficiency, good reconstruction quality, and error robustness. In the following chapter we seek to demonstrate the results produced by the proposed scheme for a number of test images. We will show that for a certain class of images, the codec outperforms JPEG at very low bit rates, in terms of both minimum attainable bit-rates, peek signal to noise ratio (PSNR) and subjective image quality.

Chapter 6

RESULTS

6.1 Introduction

6.1.1 Context

In the previous chapter, a novel hybrid coding scheme based on an amalgamation of transform coding and feature based coding was proposed. In this chapter, we evaluate this coding scheme and compare its performance with JPEG, in terms of image quality and compression efficiency.

6.1.2 Objectives and Overview

The hybrid coding scheme discussed in chapter 5, is based on a number of assumptions regarding the importance of *contours* in image perception, the statistics of their intensity distributions, and how these features may be reasonably approximated for the HVS. In this chapter, we seek to identify the circumstances[94] under which these assumptions may be deemed valid, by leading to subjectively acceptable reconstructed image quality. We also seek to identify the circumstances under which they fail to uphold. As the proposed coding scheme represents a novel combination of transform and feature based coding, we seek to characterise the particular types of distortion introduced by this approach.

[94] i.e. the types of images, or more precisely, the types of intensity distributions.

The remainder of this chapter is structured as follows. Section 6.2.1 examines the suitability of three different edge detection methods for the proposed scheme. The purpose of these tests is not to definitively determine the "best" edge detector, but rather gain insight to their suitability for the proposed scheme, by applying them to an appropriate *representative* image[95], and examining the reconstruction results. Section 6.2.2 explores the effects of different subsampling factors on the reconstructed image quality, and section 6.2.3 addresses distortion reduction in the smooth component through adaptive filtering. The final investigation looks at the benefits of including a texture component to cater for locally complex intensity distributions. Before proceeding to the results however, we briefly describe the parameters of the three codec configurations and make some brief comments on the meaning of the term *simple* with respect to images.

6.1.3 Parameters

The adjustable parameters of the three codec architectures are tabulated in table 6.1 below. In the following sub-sections, some of these parameters will be examined in terms of their effect on compression efficiency and reconstruction quality.

Table 6.1 – Hybrid Scheme Parameters
F = First Configuration, M = Merged Architecture, C = Contour-Smooth Scheme

Encoder		**F**	**M**	**C**
Parameter	*Settings*			
Noise pre-filter	Adaptive Wiener, Median, none	•	•	•
Edge Detection	Roberts, Sobel, Prewitt (gradient ops)	•	•	•
Edge Threshold	0→N, $N \in \Re$, 0<N<1	•	•	•
Edge Linking	On, off	•	•	•
Minimum Contour Length	1→N , $N \in Z$	•	•	•
Texture Sensitivity	0→N , $N \in \Re$	•	•	-
Texture Component Bit-rate	Set though quantization factor Q, in JPEG	•	-	-
Smooth Component Bit-rate	Set though quantization factor Q, in JPEG	•	-	•
Merged Components Bit-rate	Set though quantization factor Q, in JPEG	-	•	-
Subsampling Factor	2→N, $N \in Z$	•	•	•

[95] i.e. one that contains both the types of edges we would like to detect, and also those that we would prefer not to.

Decoder		F	M	C
Interpolation Method	Bilinear, Bicubic	•	•	•
Smooth Component Noise Filter	Adaptive Wiener, none	•	-	•
Post-Filter	Gaussian white noise, 0→N, $N \in \Re$	•	•	•

As table 6.1 shows, there are a number of different parameters that will influence the performance of the hybrid codec. While all of these parameters were investigated as part of this research, we focus primarily on the effects of two key parameters in this chapter: *subsampling factor* and *smooth component compression ratio* (bit-rate). The remaining parameters were generally preset to values obtained experimentally to produce visually acceptable results, or meet a given bit-rate requirement (for comparison with JPEG). Where not otherwise indicated the default parameter settings were as listed below (table 6.2).

Table 6.2 – Default parameter settings	
Noise pre-filter	None.
Edge Detection	Sobel
Edge Threshold	0.1
Edge Linking	On
Minimum Contour Length	3
Texture Sensitivity	10
Texture Component Bit-rate	20
Smooth Component Bit-rate	60
Merged Components Bit-rate	20
Subsampling Factor	8

In the decoder, adaptive Weiner filtering was performed by default, the interpolation method used was bilinear, and no noise was added post-reconstruction.

6.1.4 Image Classification

In this thesis we describe images, rather vaguely, in terms of their *simplicity* or *complexity*. These descriptions are generally employed to express the suitability of an image for

representation by the proposed scheme. It was considered that a mathematical metric might serve to give a more quantitative appreciation of this image description, but ultimately any simple metric was deemed to inadequately convey important spatial aspects. Various ratios of smooth to textured regions could be calculated, but some indication of the area and spatial distribution of these regions would also need to be incorporated into the metric. This is perhaps a problem that could be investigated as future work..

We define a *simple* image qualitatively as one characterised by the following features, which may be inspected visually:

- Mostly large homogeneous regions of smooth texture, typified by the *Peppers* image (Appendix A).

- Where small objects are present, they are in high contrast to their backgrounds, both of which are characterized by smooth texture.

Generally speaking, simple images tend to produce more "intelligible" edge maps which may be an additional aid in deciding whether an image should be described as simple or not. Fig. 6.1 demonstrates the edge maps produced by two images, one simple and one complex. Note that the many closely spaced structural features in the complex image, together with the complex textures makes it difficult to discern the content of the image from the edge map alone.

Fig. 6.1 Example of a simple image (a) and its edge map (c), and a complex image (b) and its edge map (d) .

6.1.5 Test Images

A number of test images were selected to evaluate the proposed scheme. These include a number of the "standard" test image images: Lena, Peppers, Boats, Zelda, and Camman, and some images chosen for their content *simplicity*. These images were selected as, once it became apparent that the performance of the scheme was optimal for images in this class, we wished to explore the potential of the scheme in the right context. We also test the scheme with more complex images however, such as the Pig and Boats images. All of the original source images are presented in Appendix A.

6.2 Experimental Results

6.2.1 Edge Detector Evaluation

Given the perceptual foundation of feature based coding, only contours that are *perceptually significant* should be sought by the edge detector. No effort should be made to *infer* contours where they are not locally visible[96]. In other words, we do not seek the outlines of objects as they may be inferred by higher level perceptual processes, but rather the primitives which may give rise to such inferences. The requirements of a contour detector for FBC therefore are somewhat different from those employed in classical computer vision problems.

The evaluation criteria for any potential contour detection method will, in this context, include a strong subjective component. A number of methods for the evaluation of contour detection have been developed [96][67][98]. Most methods however still rely on the HVS to judge the "quality" of results produced by contour detection. Lopez et al. [99] have defined a set of visual criteria with which to evaluate detector performance and Heath et al. [98] have formalized another process for human visual inspection of edge images. A recent proposal for the evaluation of contour detectors is based on *receiver operating characteristic* curves (ROC) [96]. In this method, a set of ground truths are manually specified. The presence of *true positives* and *false positives* in the edge map is then used to construct a ROC curve. Different detection methods may then be evaluated in terms of their ROC graphs.

An important aspect of any evaluation criteria however, is what "good" performance actually signifies, i.e. good in what respect? In FBC, a good edge detector is one that allows for the reconstruction of images with an acceptable perceptual correspondence to the original. It is not immediately clear how human efforts to judge the "quality" of edge maps may correlate with this outcome. The utility of such metrics should therefore be considered with caution. In FBC it is perhaps more appropriate to consider the performance of an edge detection method in terms of the perceived quality of the reconstructed image, rather than the edge map itself.

[96] In the sense that we should attempt to recover shape. Contour segments may however, be connected for coding efficiency.

In addition to perceptual considerations, a suitable edge map should be amenable to efficient coding. Where *chain coding* is used to code the resulting contours, an ideal contour map should have the following properties.

1. Smooth contours, i.e. no unnecessary *kinks* in contours.

2. Connected contours, small *gaps* in contours diminish compression efficiency.

3. No more than one pixel wide.

The results of three different edge detection methods are presented in this thesis: *Roberts*, *Sobel*, and *Prewitt* (see Appendix D). These are all simple and efficient operators that approximate the gradient of the signal at a point. The contour-smooth encoder architecture was employed to evaluate the performance of these methods in terms of the following characteristics:

1. The average number of bits required to chain code a contour pixel in the edge-map.

2. The subjective quality of the images reconstructed using the various methods to detect edges.

In the contour-smooth scheme, contours arising from both texture and busy-edges are encoded. Following edge-linking, all contours less than a threshold value of 3 pixels are removed. The "Lena" image was used in these tests as it contains a mix of both strong and weak contours, and texture at different scales. All edge detectors were configured[97] to produce an edge map that, following edge linking, resulted in an overall bit-rate of approximately 0.0515 bits per pixel when *chain coded.* This value was chosen as one that produced a "good" edge-map for the Sobel operator. Thus, we compare the output of the different detectors, matching this bit-rate.

Tables 6.3 through 6.5 present the results for the Sobel, Roberts, and Prewitt methods respectively. Figures 6.1 though 6.3 display reconstructed images where a subsampling factor of 8 was used.

[97] i.e. the edge strength threshold parameter(s).

Table 6.3 Sobel Method – Lena Image (512x512)

Bit-rates produced by the Sobel operator. Contours less than 3 pixels in length removed. Bit-rate calculations based on Sobel output *with* and *without* edge-linking.

SP = Contour start point, CP = contour point, RP = number of removed (i.e. <3) contours.

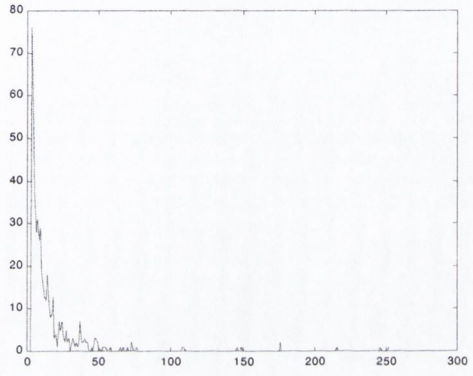
See Fig. 6.2 for reconstruction results using contour map *with* edge linking.

Without Edge-Linking: Average BPCC = **1.7876**, SP = 497, CP = 7352 , RP = 307

Total BPP = **0.0535**.

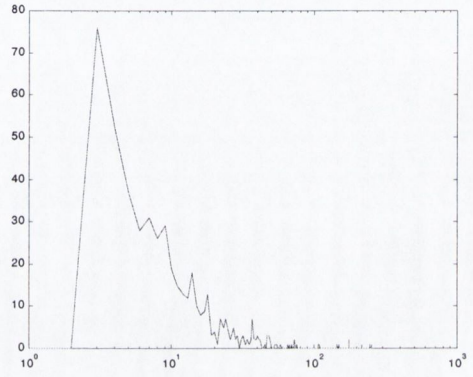
Histogram of Contour Lengths

Without Edge-Linking



Histogram of Log Contour Lengths

Without Edge-Linking

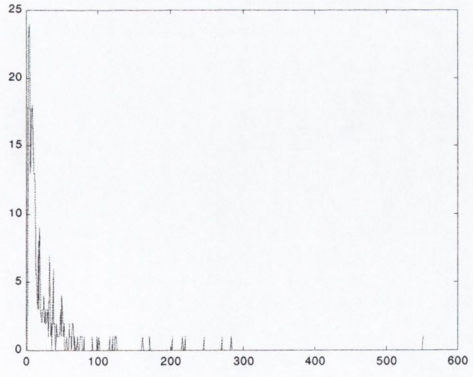


With Edge-Linking: Average BPCC = **1.5779**, SP = 308, CP = 8225, RP = 238

Total BPP = **0.0514**.

Histogram of Contour Lengths

With Edge-Linking



Histogram of Log Contour Lengths

With Edge-Linking

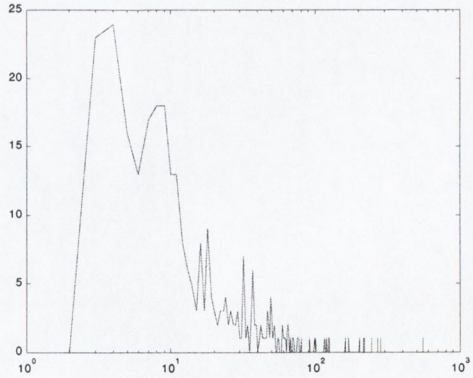


Table 6.4 Roberts Method – Lena Image (512x512)

Bit-rates produced by the Roberts operator. Contours less than 3 pixels in length removed. Bit-rate calculations based on Roberts output *with* and *without* edge-linking.

SP = Contour start point, CP = contour point, RP = number of removed (i.e. <3) contours.

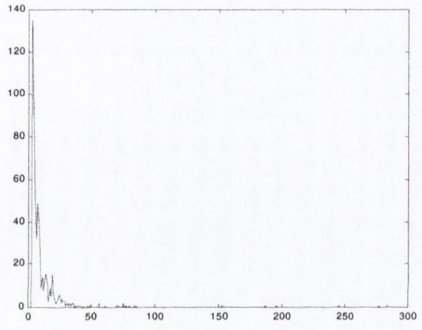
See Fig. 6.3 for reconstruction results using contour map *with* edge linking.

Without Edge-Linking: Average BPCC = **1.9257**, SP = 592, CP = 6693 , RP = 642

Total BPP = **0.0535**

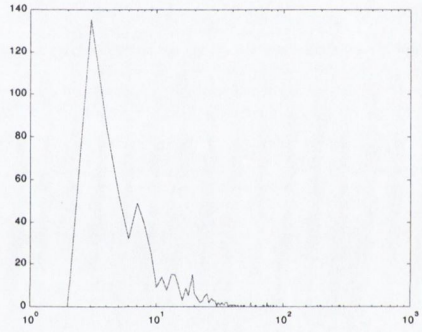
Histogram of Contour Lengths

Without Edge-Linking



Histogram of Log Contour Lengths

Without Edge-Linking

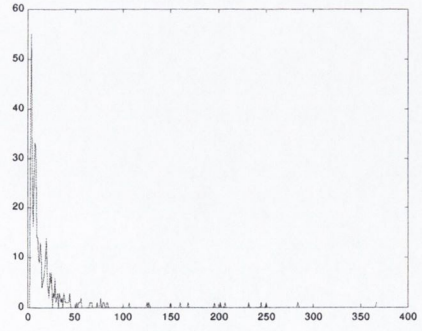


With Edge-Linking: Average BPCC = **1.6817**, SP = 405, CP = 7766, RP = 504

Total BPP = **0.0524**.

Histogram of Contour Lengths

With Edge-Linking



Histogram of Log Contour Lengths

With Edge-Linking

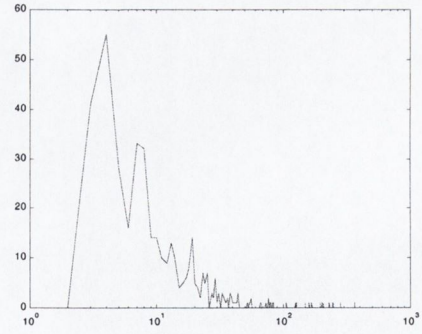


Table 6.5 Prewitt Method – Lena Image (512x512)

Bit-rates produced by Prewitt operator. Contours less than 3 pixels in length removed. Bit-rate calculations based on Prewitt output *with* and *without* edge-linking.

SP = Contour start point, CP = contour point, RP = number of removed (i.e. <3) contours.

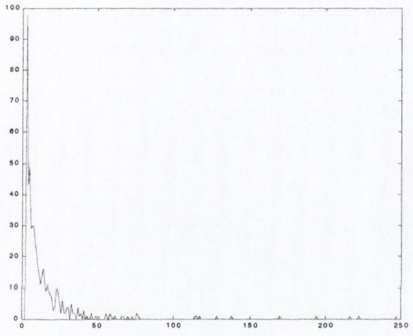
See Fig. 6.4 for reconstruction results using contour map *with* edge linking.

Without Edge-Linking: Average BPCC = **1.8019**, SP = 533, CP = 7673 , RP = 295

Total BPP = **0.0564**.

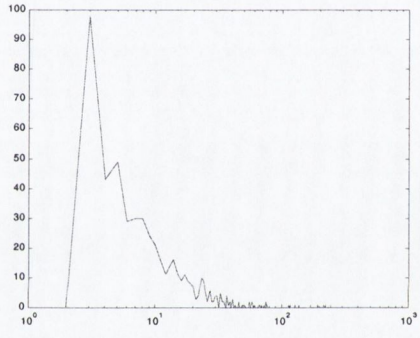
Histogram of Contour Lengths

Without Edge-Linking



Histogram of Log Contour Lengths

Without Edge-Linking

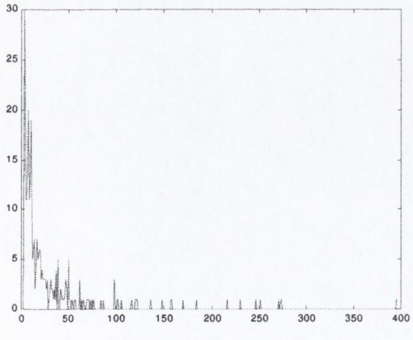


With Edge-Linking: Average BPCC = **1.5592**, SP = 302, CP = 8668, RP = 219

Total BPP = **0.0534**.

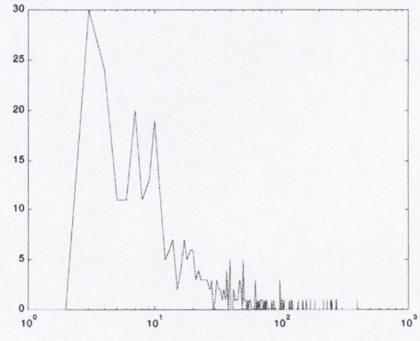
Histogram of Contour Lengths

With Edge-Linking



Histogram of Log Contour Lengths

With Edge-Linking



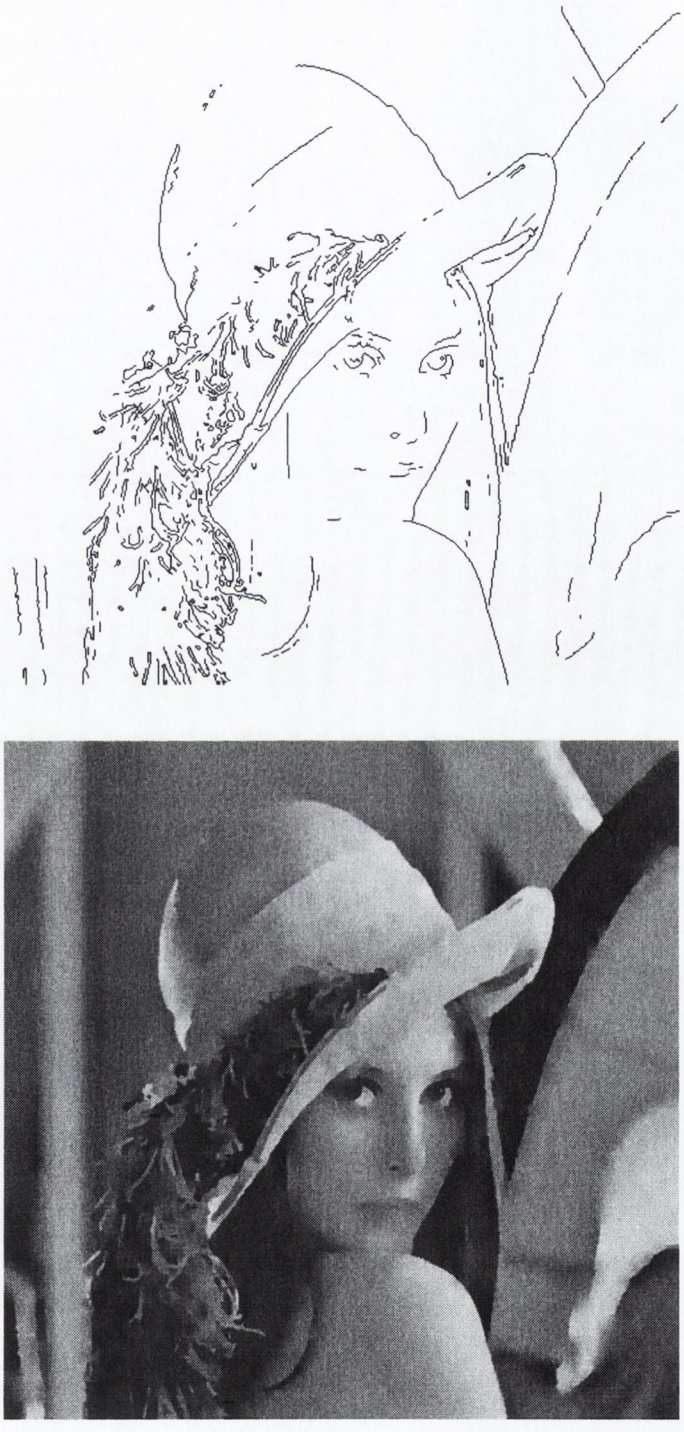


Fig. 6.2 Contour-map and reconstructed image for 512x512 "Lena" image, using a subsampling factor of 8, no smooth component compression, no texture, and the **Sobel** edge operator, and a contour bit-rate of 0.0514 bpp.

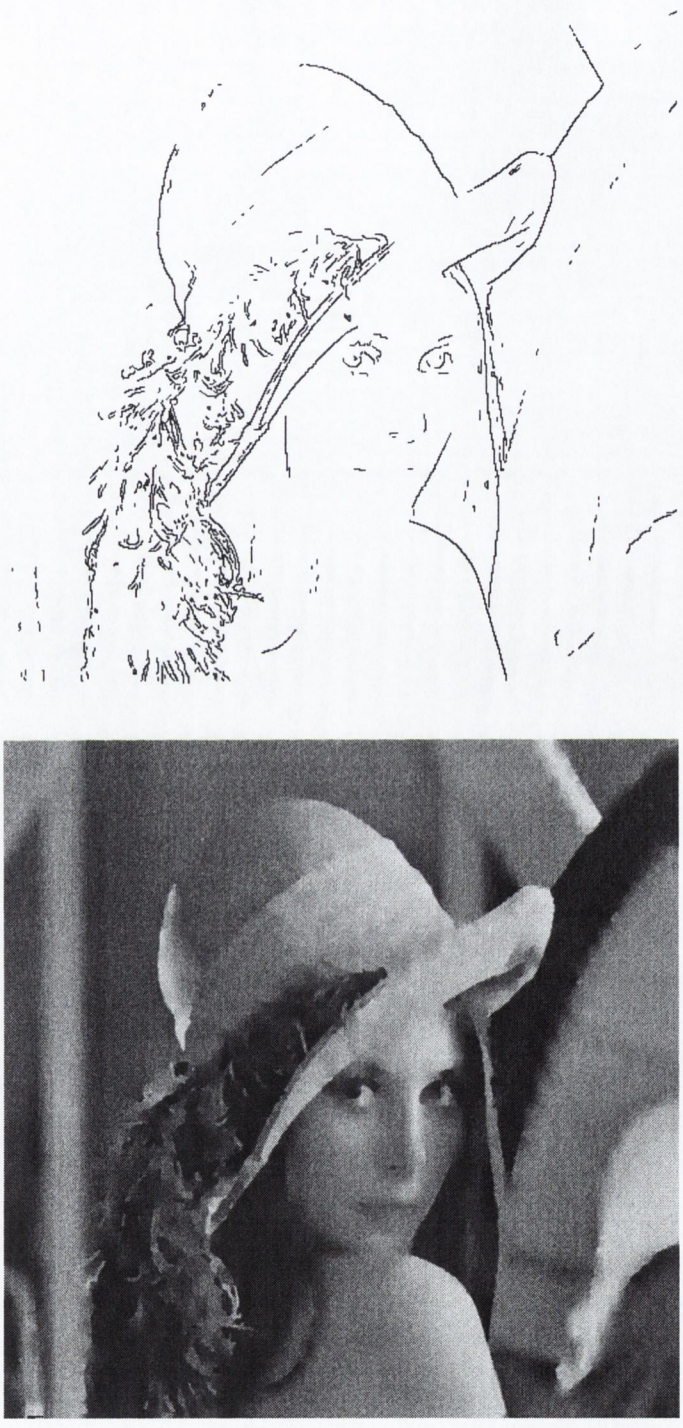


Fig. 6.3 Contour-map and reconstructed image for 512x512 "Lena" image, using a subsampling factor of 8, no smooth component compression, no texture, and the **Roberts** edge operator, and a contour bit-rate of 0.0524 bpp.

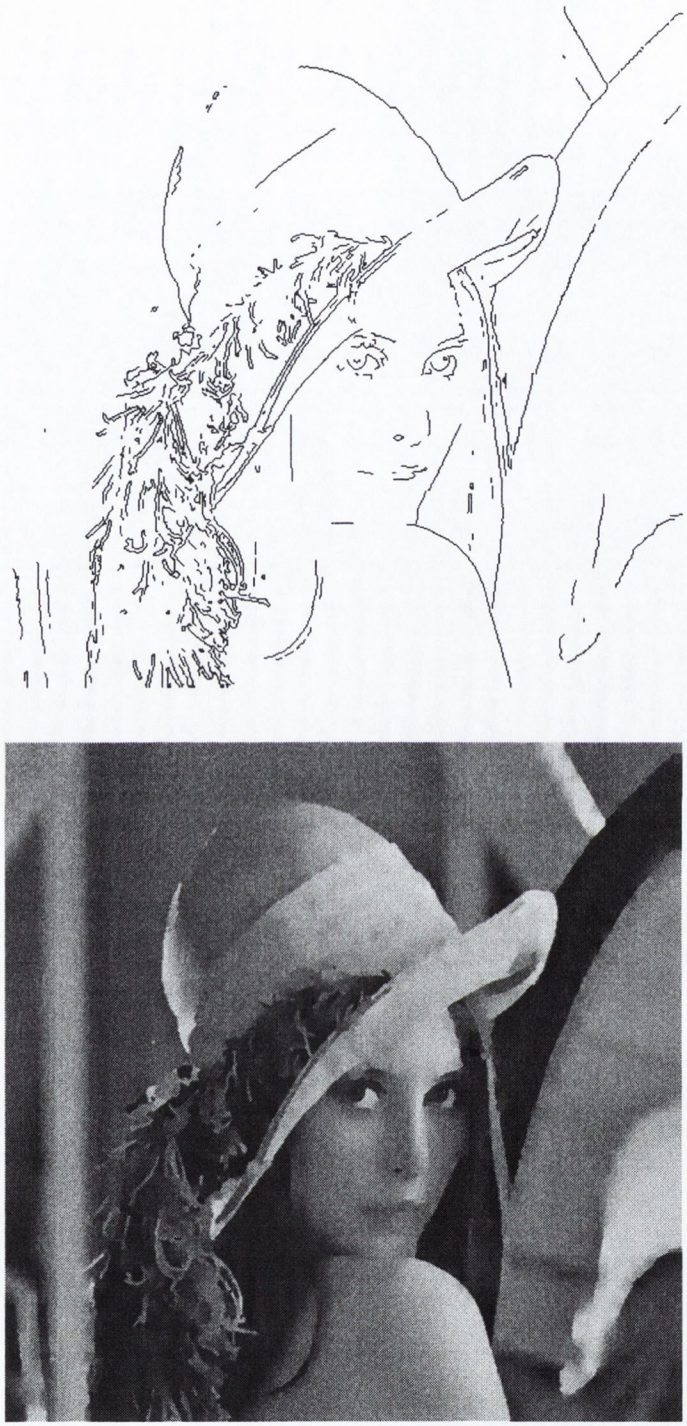


Fig. 6.4 Contour-map and reconstructed image for 512x512 "Lena" image, using a subsampling factor of 8, no smooth component compression, no texture, and the **Prewitt** edge operator, and a contour bit-rate of 0.0534 bpp.

The results of the three investigated methods, operating on the Lena image, are presented in Tables 6.2 through 6.4, and Figures 6.1 though 6.3. The contour coding statistics and reconstructed Lena images produced from the Sobel and Prewitt methods are almost identical. The Roberts method however is clearly inferior to both of the other methods.

Closer inspection of the results demonstrate that the Prewitt operator benefited more than the Sobel operator from edge-linking which suggests that its original edge map contained more unconnected pixels. This is evidenced by the high number of short contours in the original Prewitt edge-map. Further experiments with the Prewitt and Sobel operators on additional images have indicated that the Sobel operator tends to outperform the Prewitt with respect to our requirements. The remainder of the results presented in this section use the Sobel edge operator.

In addition to the simple gradient based methods the Canny and Laplacian-of-Gaussian methods were explored[98], but found to yield few benefits but considerable additional complexity. The principal failing of the gradient based methods is their inability to detect weak contours that give rise to mach bands. These contours should be recorded to remain faithful to the feature based principles but are not detected by the approach used to detect strong contours. Future work should address this failing.

6.2.2 Subsampling and Reconstruction Quality

In this subsection, we investigate the effects of subsampling on the quality of the reconstructed image. In particular, we are interested in exploring the validity of the assumptions stated in chapter 5, sections 5.3.3 and 5.3.4, namely that contours and smooth texture alone often communicate all the essential information in an image. To put this hypothesis to the test, we examine the effects of different subsampling factors on PSNR and subjective image quality. The *contour-smooth scheme* is employed for these tests. The smooth component is coded without loss, as we are primarily interested in the effects of texture loss here. The images tabulated in table 6.6 (below) were used in this evaluation.

[98] A number of sample source images were coded, reconstructed, and inspected visually using the alternative edge detection methods.

Table 6.6 – Images used in evaluation of subsampling

Image	Scale of Relevant information	Busy Edges?	Texture Content	Smooth Areas
Peppers (512x512)	An ideal image for FBC, contours are long, smooth, and connected. Most of the relevant structural information is present a large scales.			
	large	very few	very low	large (surface of peppers)
Plane (640x480)	An ideal image for FBC, dominated by large smooth areas, very simple texture, and high contrast structural information. The relevant information is mostly present at small scales, but is surrounded by smooth areas.			
	small & large (people & plane)	few (around stalks & spot reflections)	very low (some weak texture in snow)	large (snow, sky)
Lena (512x512)	An image containing relevant structural information at a number of scales. As this is an image of a human face, the eyes and mouth regions have the most perceptual relevance. The woman's hair contains many busy contours that rank low for perceptual importance but occupy a much larger region than the eyes and mouth			
	large and small (hat, shoulder, eye, mouth)	many (hair)	high in localized area (hair)	large (background, shoulder, hat)
Pig (352x512)	A deceptively complex image. The apparently smooth pig's face and body contain many busy contours. The ground around the pig is highly textured, with textured areas abutting strong edges (rail tracks). The smooth sky area is broken by busy edges (telephone pole)			
	small & medium (poles, face)	very many	high (ground)	few (sky, pig's body)

6.2.2.1 Results: Peppers

Results for the Peppers image presented in table 6.7, Fig. 6.5, and Fig. 6.6. Subsampling factors up to 24 were possible while still yielding intelligible results. The "Number of extra samples recorded" table entry refers to the number of *relocated* or *extra* samples recorded during subsampling.

Table 6.7 - Peppers Image (512x512): subsampling statistics.

Subsampling factor	Smooth-image size	PNSR (dB)	Number of extra samples recorded
3	170x170	21.6199	315
4	128x128	21.3372	168
6	85x85	20.9163	170
8	64x64	20.5264	129
12	42x42	19.9501	93
16	32x32	19.4801	115
24	21x21	18.7277	83

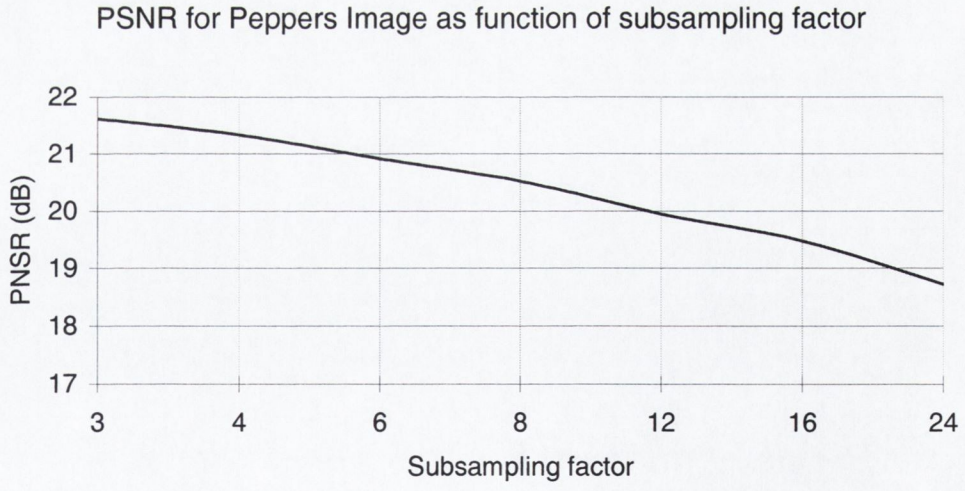
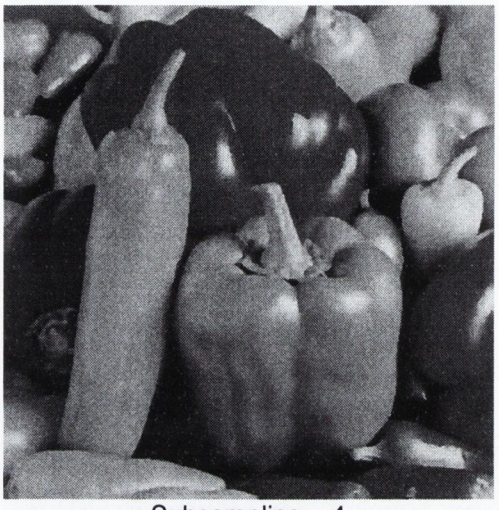
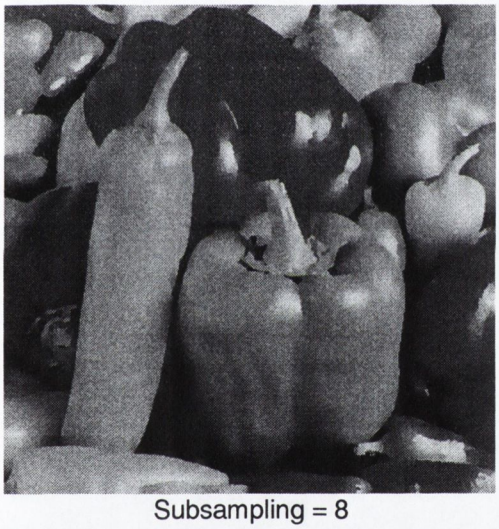


Fig. 6.5 The PSNR performance of the contour-smooth scheme with the Peppers image and varying degrees of subsampling.

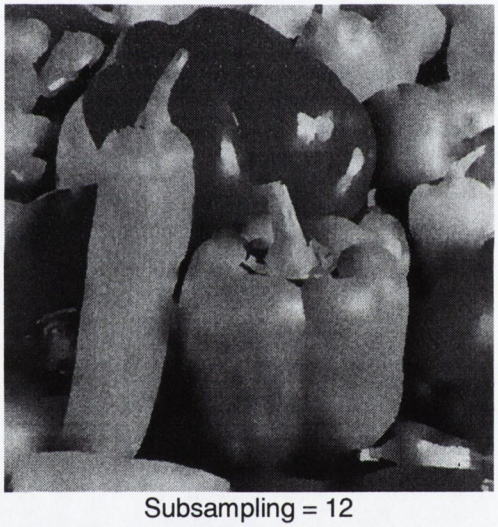
Edge map (same for all)



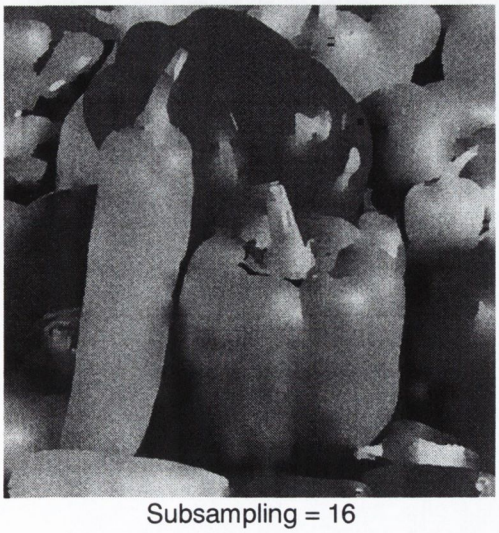
Subsampling = 4



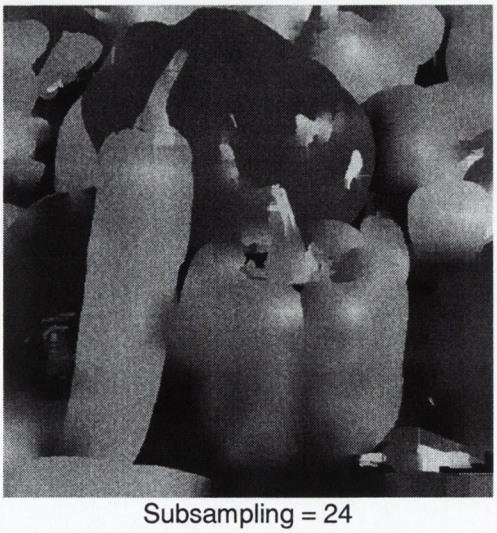
Subsampling = 8



Subsampling = 12



Subsampling = 16



Subsampling = 24

Fig. 6.6 Effects of subsampling on reconstruction quality for the Peppers image.

6.2.2.2 Results: Plane

Results for the Plane image presented in table 6.5, Fig. 6.7, & Fig. 6.8.

Table 6.8 - Plane Image: subsampling statistics.

Subsampling factor	Smooth-image size	PNSR (dB)	Number of Extra samples recorded
3	160x213	23.0163	597
4	120x160	22.8640	303
6	80x106	22.6206	199
8	60x80	22.4292	183
12	40x53	22.2287	190
16	30x40	22.0463	150
24	20x26	21.7450	144

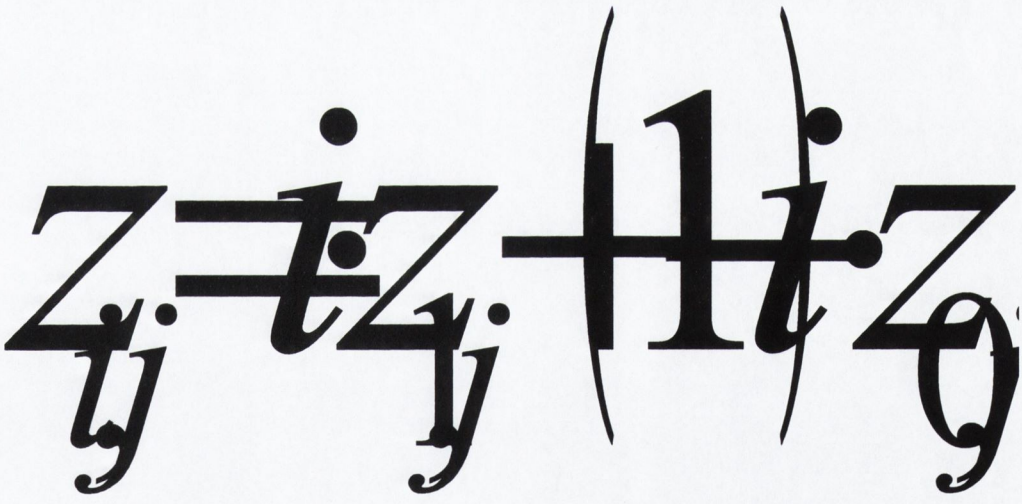
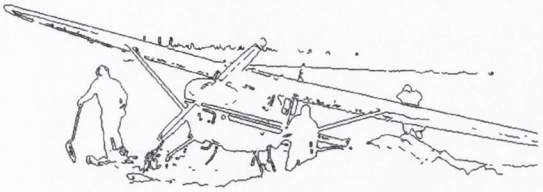
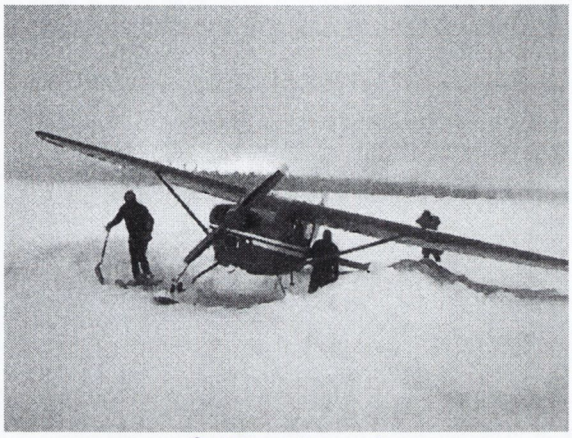


Fig. 6.7 The PSNR performance of the contour-smooth scheme with the Plane image and varying degrees of subsampling.

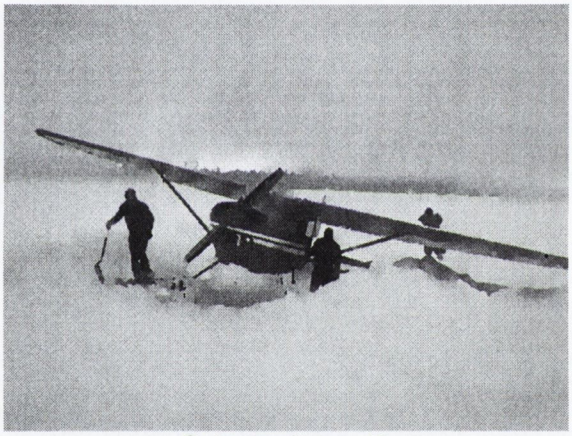


Edge map (same for all)



Subsampling = 4

Subsampling = 8



Subsampling = 12

Subsampling = 16

Subsampling = 24

Fig. 6.8 Effects of subsampling on reconstruction quality for the Plane image.

6.2.2.3 Results: Lena

Results for the Lena image presented in table 6.6, Fig. 6.9, & Fig. 6.10.

Table 6.9 - Lena Image: subsampling statistics.

Subsampling factor	Smooth-image size	PNSR (dB)	Number of Extra samples recorded
3	170x170	21.2196	752
4	128x128	20.9535	452
6	85x85	20.5578	282
8	64x64	20.1569	285
12	42x42	19.5803	263
16	32x32	19.1002	252
24	21x21	18.3645	207

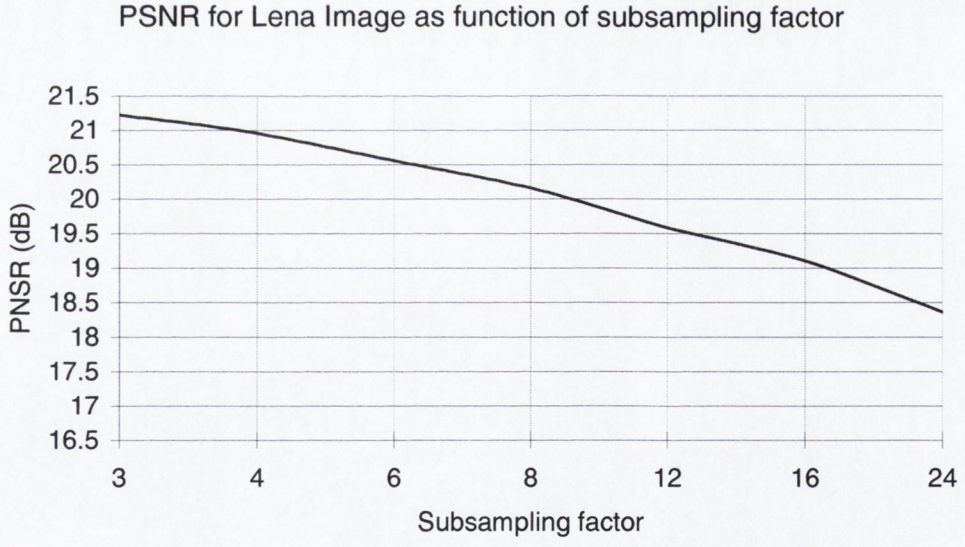


Fig. 6.9 The PSNR performance of the contour-smooth scheme with the Lena image and varying degrees of subsampling.

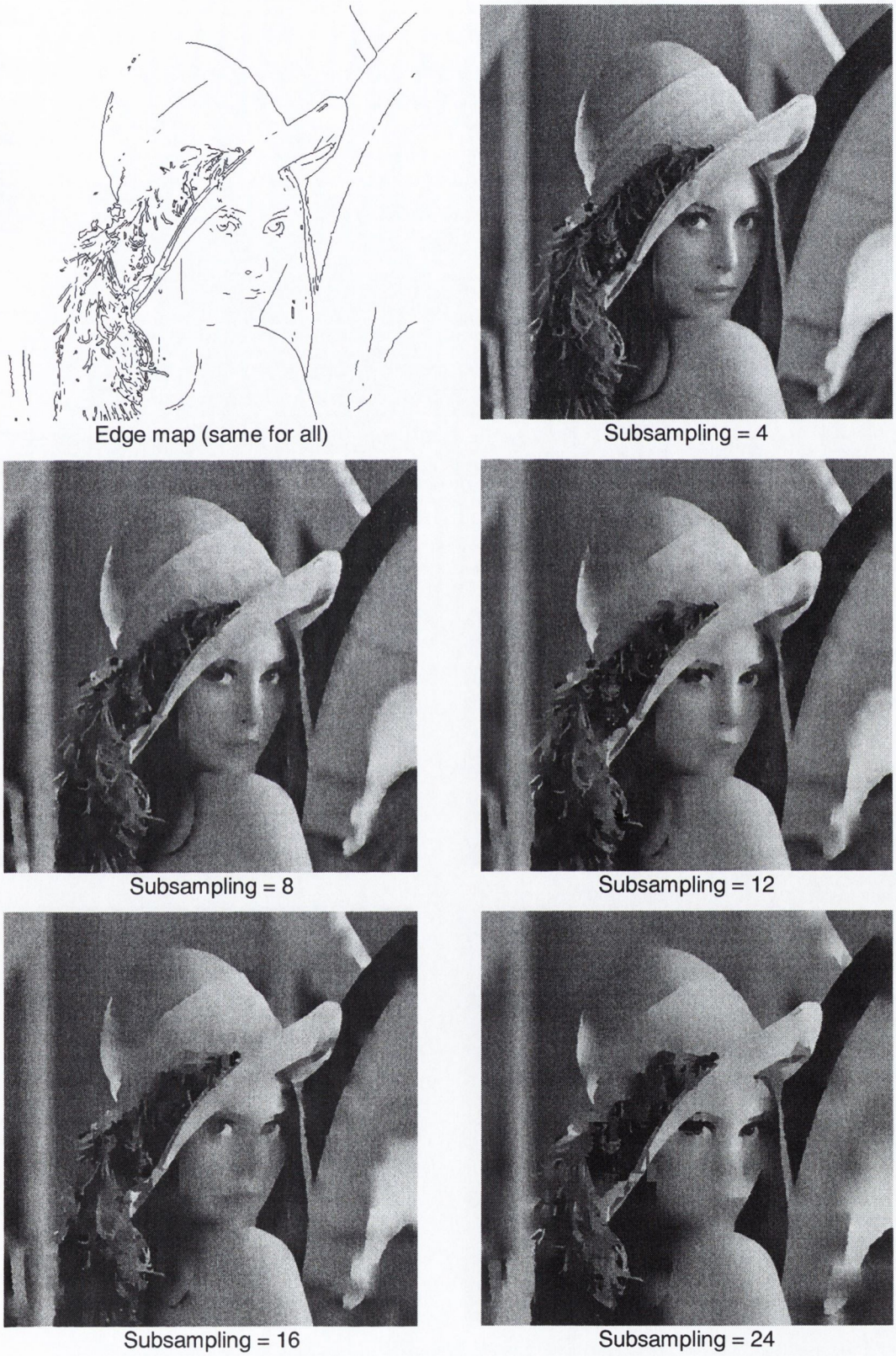


Fig. 6.10 Effects of subsampling on reconstruction quality for the Lena image.

6.2.2.4 Results: Pig

Results for the Plane image presented in table 6.7, Fig. 6.10, &.Fig. 6.11

Table 6.10 - Pig Image: subsampling statistics.

Subsampling factor	Smooth-image size	PNSR (dB)	Number of Extra samples recorded
3	170x117	21.7589	547
4	128x88	21.5590	292
6	85x58	21.2074	222
8	64x44	20.9978	229
12	42x29	20.5472	194
16	32x22	20.3789	174
24	21x14	19.6537	147

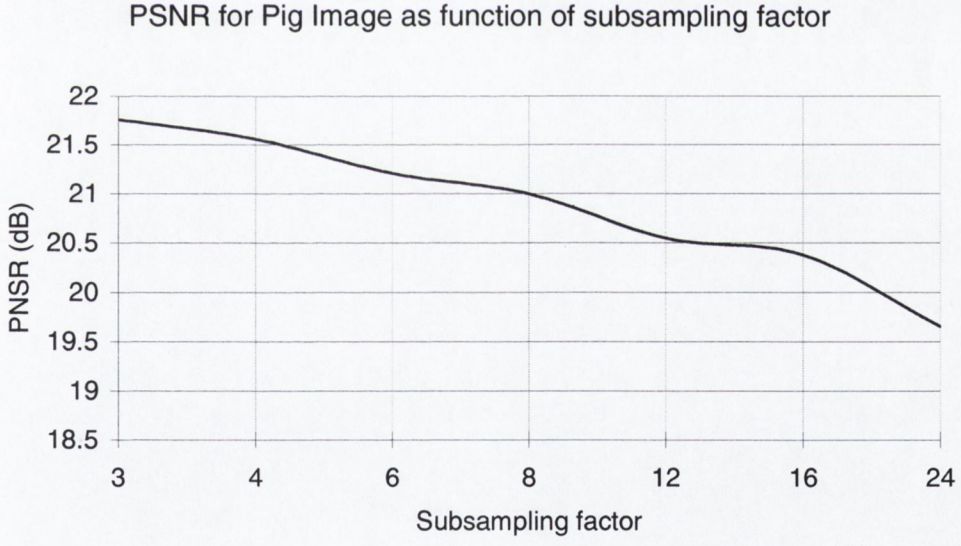
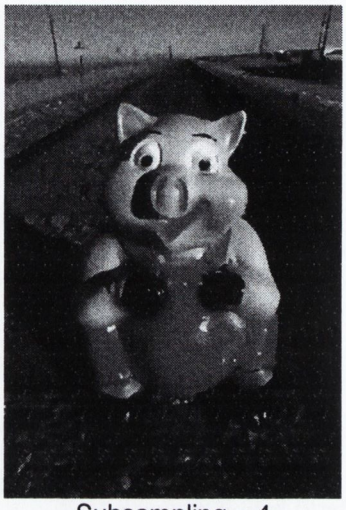


Fig. 6.11 The PSNR performance of the contour-smooth scheme with the Pig image and varying degrees of subsampling.

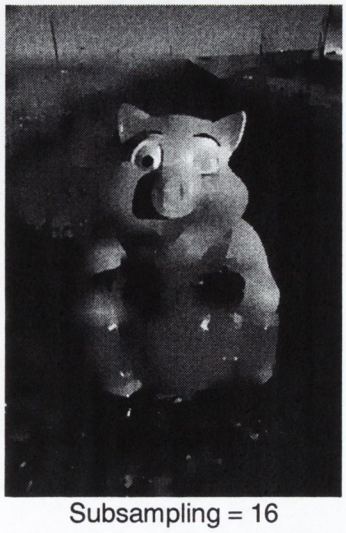
Edge map (same for all)



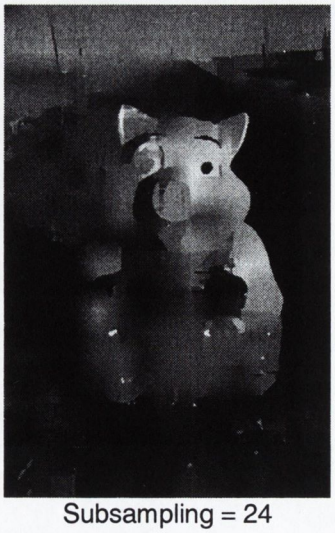
Subsampling = 4

Subsampling = 8

Subsampling = 12



Subsampling = 16



Subsampling = 24

Fig. 6.12 Effects of subsampling on reconstruction quality for the Pig image..

6.2.2.5 Conclusions

The results for the four test images; Peppers, Plane, Lena, and Pig, demonstrate the validity of the contour/smooth approximations for the Pepper and Plane images. In these images structure is present at large and small scales respectively. In both cases, strong edges are surrounded by very smooth texture, and reconstructed images are intelligible even with very large subsampling factors. For the Lena and Pig images by contrast, severe distortion is introduced in some image areas. There is a visually unpleasant "blending" of textures and smooth areas where subsampling factors greater than or equal to 8 have been used.

In all cases however, significant strong edges that abut largely smooth areas are rendered with good clarity. The figures of the people in the Plane image are clearly discernable, as is the outline of the pig in the Pig image, and the hat, shoulder, and iris outlines of the eyes in the Lena image. We conclude that the spatial accuracy and clarity of edges are well preserved by the scheme, and lead to good reconstruction quality where those features are sufficiently large and distinct from their surroundings.

It is important to note however that in these tests no compression was performed on the smooth component. It is likely that compression artifacts in the smooth component will have a large impact on the visual quality of reconstructed images, particularly so when the subsampling factor is high. In the following section we investigate the effects of compression artifacts in the smooth component, on the subjective quality of reconstructed images.

6.2.3 Smooth Component Compression

In this section we investigate the effects of compression artifacts in the smooth component, on the quality of the reconstructed image. We employ the *contour-smooth* scheme again for these tests. As preliminary tests indicated that the visibility of distortion arising from a coarsely quantized smooth component is most prominent around strong edges, we shall only provide results for the Plane image, which is characterized principally by strong edges abutting smooth texture.

Table 6.11 Plane Image: Smooth Component Compression Results

Contour component file size: 1070 bytes (for all images)

Number of *extra* and *relocated* samples: 303 – uncompressed, 303 bytes

Q-Setting in Smooth Image	Smooth-component file size (bytes)	Total Compression Ratio[99]	PSNR With adaptive Wiener filter	PSNR Without adaptive Wiener filter
100	11994	23	22.3451	22.8640
95	6414	39	22.3578	22.7230
90	4634	51	22.3270	22.5997
85	3804	59	22.2968	22.4867
80	3316	66	22.2658	22.3951
75	2956	71	22.2413	22.3365
70	2732	75	22.2094	22.2652
65	2534	79	22.1510	22.1909
60	2370	82	22.1180	22.1329
55	2221	85	22.1179	22.1188

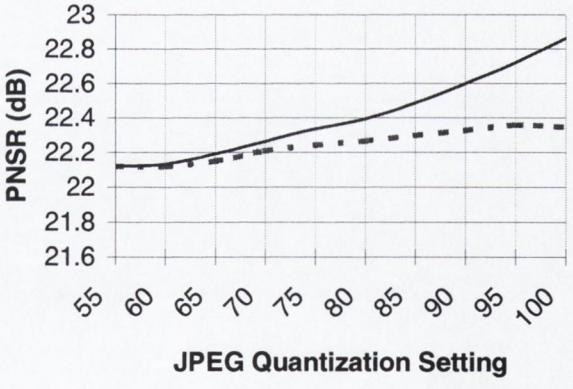


Fig. 6.13 PNSR Results for Plane image compressed with a subsampling factor of 4. The PSNR without adaptive Wiener, the solid line, filtering is higher (better) than the PSNR with filtering (dashed line) except at very low bit-rates. As may be seen from Fig. 6.15 and Fig. 6.16 however, the filtered image is subjectively better.

[99] The total compression ratio, to which we refer, is the compression ratio of all the scheme components combined.

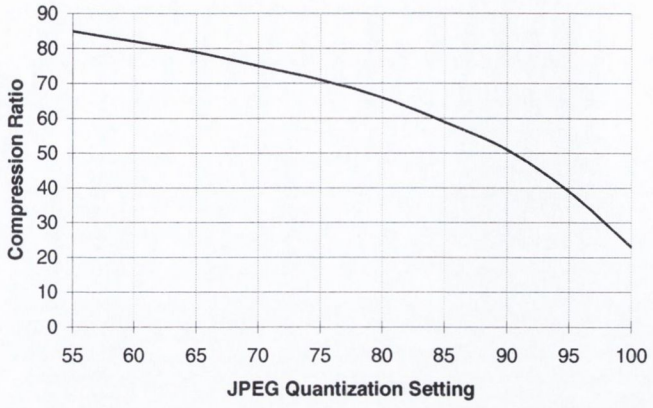


Fig. 6.14 Compression ratio for Plane image with subsampling factor of 4 as a function of smooth component compression. A quantization setting of 100 is equivalent to negligible distortion in the smooth component, while smaller numbers indicate successively coarser quantization.

Fig. 6.15 Reconstructed Plane image at a compression ratio of **85:1** *without* filtering of the smooth component prior to insertion in the sample grid. The smooth image corresponds to a JPEG Q factor of 55. Subsampling factor = 4.

Fig. 6.16 Reconstructed Plane image at a compression ratio of **85:1** *with* adaptive Wiener filtering of the smooth component (same as Fig. 6.15) prior to insertion in the sample grid. Distortion has been largely removed. Subsampling factor = 4..

Fig. 6.15 above demonstrates the appearance of quantization errors and their magnification in the reconstructed image. As may be seen from Fig. 6.16 however, this distortion may be satisfactorily reduced by adaptive Wiener filtering prior to reconstruction, albeit at the cost of texture fidelity. As the smooth image is generally relatively small, 120 x 160 in this case, the filtering process will not generally introduce a significant computational overhead. Fig. 6.17 presents the Plane image compressed with JPEG. The compression ratio for this image is 61. The quality of the compression results with the contour-smooth scheme is subjectively superior to JPEG, even at 83:1. As Fig. 6.8 suggests however, further subsampling may be possible without appreciable distortion, as an alternative means of achieving higher compression. In Fig. 6.18, a number of low complexity images have been compressed with compression ratios ranging from 81 to 120, and compared with JPEG. In the next section we investigate addition of the texture component to address the smooth-contour schemes' poor handling of textures and busy edges.

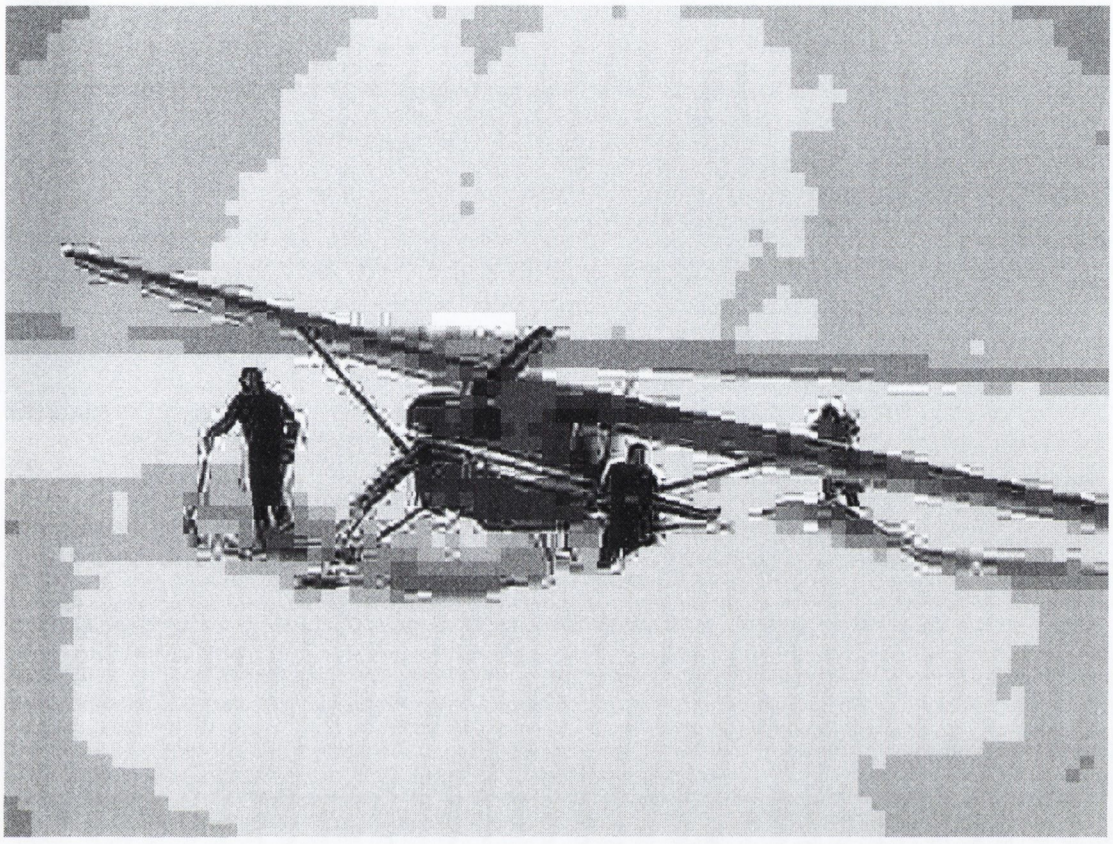
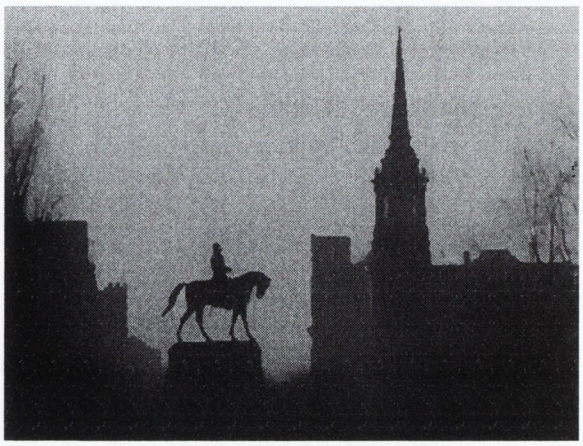


Fig. 6.17 The Plane image compressed using JPEG with a compression ratio of **61**:1.

JPEG CR=**69**, PSNR=17.77

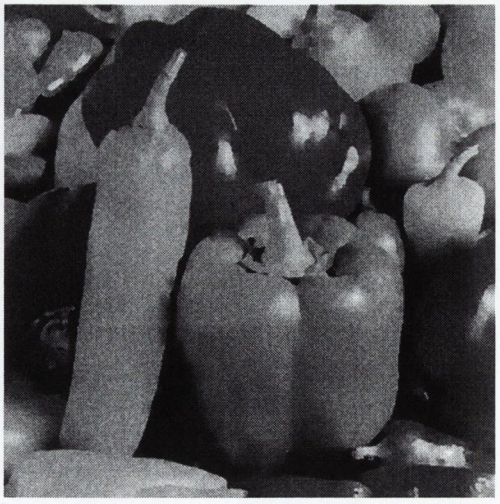


Smooth-Contour CR=**120**, PSNR = 21.78

JPEG CR=**52**, PSNR=17.5

Smooth-Contour CR=**81**, PSNR = 20.01

JPEG CR=**60**, PSNR=17.76



Smooth-Contour CR=**83**, PSNR = 19.54

Fig. 6.18 Examples of various low complexity images compressed with JPEG, and with the smooth-contour scheme. All original images provided in the appendix. A subsampling factor of 8 was used on all smooth-contour images, with adaptive Wiener filtering on the smooth component.

6.2.4 Adding Texture

The experimental results presented thus far, have reflected the performance of the most simple scheme, where just the contour and smooth components are employed. Image representation based on these two components alone yields good results for simple images, but may cause severe distortion in more complex images. This was evidenced by the effects of subsampling on the Pig and Lena images. These distortions manifest as a strong blurring and *smearing* across smooth edges, and often the loss of small objects completely. Furthermore, closely spaced narrow edges may become merged with their surroundings. Fig. 6.19 shows the Boatwave image reconstructed without texture. The subsampling frequency was 4, and the compression ratio is 85:1. Note that fine detail, such as the name on the boat, has been lost.

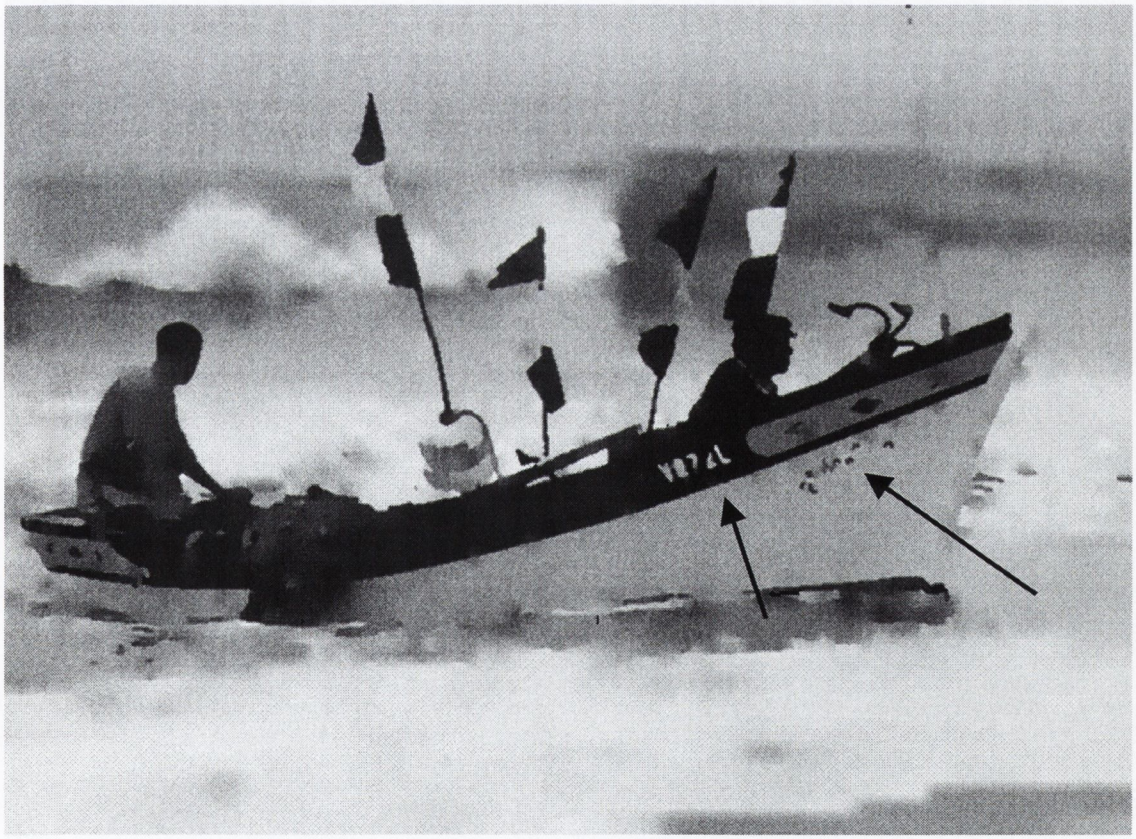


Fig. 6.19 Boatwave image reconstructed without texture, note the loss of detail.
CR=85:1 PSNR=20.00 before smooth component filtering, 18.22 afterwards.

Adding the texture component may remedy the problems of detail loss and edge merging to a certain degree. As the texture component adds information, however, there is

generally an increase in the overall bit-rate. Fig. 6.20 shows the Boatwave image again, this time with texture added. The compression ratio has been reduced to 59:1

Fig. 6.20 Boatwave image reconstructed *with* texture, fine detail has been recovered (although how well is dependent on the compression factor). PNSR = 19.91 before smooth component filtering, 19.30 afterwards.

This image also demonstrates how some *busy-edges* have been classified as texture; flag pole, anchor. Comparing these features, we note that the image in Fig. 6.19 displays more visually pleasing results. This is a clear shortcoming in the classification of texture. In many circumstances, how well *line edges* are rendered is very much dependent on how long they are, and their curvature. In the Lena image, the hair is a suitable candidate for texture coding, but many of the individual strands have characteristics on a small scale, similar to the flag pole above. If the flag pole were shorter, then perhaps we would not be so "lucky" in terms of the number of block boundary intersections that yield extra samples. The Boatwave image compressed with JPEG is presented in Fig. 6.21.

Fig. 6.21 Boatwave image compressed with JPEG, CR=52, PSNR= 18.2

6.3 Merged Architecture versus the First Configuration

For smaller images (256x256) the merged architecture may offer some advantages. In the merged architecture, smooth and texture components are coded together. This simplifies this encoding and decoding of the two components using the DCT. The merged components effectively represent a single image. A drawback of the approach however, is that the smooth component cannot be independently quantized, and this is where the significant compression benefits were found in the first configuration. On the other hand, the smooth component will be not suffer any coding distortion in the merged component, and will therefore not require filtering to reduce distortion.

6.4 When Things go Wrong

In this section we present a side by side comparison of the less favourable results obtained with the proposed scheme. We don't incorporate the texture component here as it generates too many irrelevant details, while reducing the compression ratio (See Fig. 6.22 below)

Fig. 6.22 Examples of cases where compression subjective superiority to JPEG is not so clear.

6.5 Discussion

6.5.1 Distortion

The types of distortions that become manifest in the proposed scheme are listed below:

- *Blurring* occurs most visibly in mid-range frequencies and across weak edges. The visibility of this distortion is accentuated where it abuts textured areas. Unlike an optical system, the blurring distortion introduced by this scheme is not a function of the object distance. Thus, the blurring of objects in the same focal plane has an "unusual" appearance.

- *Smearing* occurs where image regions that don't conceptually belong to the same object become merged. The degree of smearing depends on the accuracy of the segmentation achieved by edge detection, the subsampling factor, and a "pot luck" factor determined by the location of objects with respect to the sampling grid.

- *Bleeding* occurs where gaps in contours allow intensity values to "leak out" into neighbouring regions. The subsampling factor will determine the spatial extent of bleeding. The larger the subsampling factor, the greater the extent to which bleeding may occur. Edge linking may be employed to close gaps (of one or two pixels) in contours.

- *Blocking effects* may occur around textured regions, or smooth regions if the quantization of the texture and smooth components is set too high.

6.5.2 Signal Dependency

The proposed scheme exhibits strong signal dependency. Furthermore, the benefits of the scheme are most evident when the input image is large (~640x480). The ideal class of images for the proposed scheme would seem to be large images containing large objects and/or a number of sharply defined smaller objects. This type of image is typified by the Plane, Horse, Mountain, & Boatsurf test images. The reason that the proposed scheme favours these types of images is that large subsampling factors may be used without "loosing" too much detail. Furthermore, if the DCT is used to decode the smooth component of such images, the variance in blocks containing predominantly smooth image

regions will be quite small. This enables the DCT to achieve superior compression with less distortion. One could consider the scheme to "add" a number of lower frequency basis functions, with longer impulse responses, to the DCT in this case. It simply enables the DCT to exploit correlations across a wider area of the image.

The problem of signal dependency may be alleviated to some degree by setting a lower threshold for texture inclusion. This would allow more high energy regions to be catered for by the DCT which is versatile in terms of signal dependency. The texture component cannot be highly compressed however, as reconstructed texture blocks are used by neighbouring smooth and contour blocks in their reconstruction, and errors should not be allowed to corrupt these blocks. Furthermore, *graininess* in texture blocks is particularity visible adjacent to smooth blocks. In our experiments, application of a smoothing filter to the final reconstructed image was often found to remedy this problem, albeit with a loss of clarity around strong edges. Future work could consider using the edge map to "turn-off" this smoothing filter around edges.

The restriction of the schemes' advantages to large simple images is not seen as a failing. There are numerous visually meaningful images that fall into this category, and many applications that require precisely this kind of data (e.g. low bit-rate tele-operation of mobile robots in indoor environments).

6.6 Summary & Conclusions

A novel hybrid coding scheme has been presented for low bit-rate coding of grey-level images. We have investigated the principal conceptual and architectural issues and shown the new scheme to offer significant improvements over JPEG at low bit rates. It was shown that compression is achieved by the proposed scheme by independently adjusting a number of key parameters. In this chapter we focused principally on the effects of *subsampling factor* and *smooth component compression*. A larger subsampling factor offers the potential for higher compression ratios but results in a loss of smooth texture fidelity. Furthermore, less distortion can generally be tolerated in the smooth component for larger subsampling factors. The benefits of adding texture back to the image was also demonstrated. The inclusion of texture results in an increased bit-rate however and is therefore generally only practical if the amount of texture is small (<5% of total area).

Chapter 7

CONCLUSIONS & FUTURE WORK

7.1 Introduction

In the previous chapter the results of experimental investigations into the performance of a novel hybrid coding scheme were presented. In this chapter we proceed to draw conclusions on the success of the scheme for grey-level image coding and make some recommendations for future work.

7.2 Conclusions

The novel hybrid coding scheme presented in this thesis may be considered a viable alternative to existing feature-based or transform coders for very low bit-rate coding of simple[100] images. The advantages and disadvantages of the hybrid scheme may be summarized as follows:

Advantages:

- Strong edges are rendered with good visual quality. Ringing noise and blurring are completely absent in edges reconstructed from contour data.
- Blocking effects are considerably reduced if not completely eliminated.

[100] See section 6.1.4 for a description of this term.

- The saturation threshold of BTC has been overcome in this hybrid scheme. Compression ratios of up to 120:1 may be achieved with comparatively good image quality when coding simple images.
- Reconstructed images lend themselves well to zooming and intensity adjustments.
- Computational complexity is very low in comparison to existing feature-based methods. Furthermore, subsampling reduces the number of forward transform calculations dramatically. All *required* encoder steps have simple efficient implementations.
- There is flexibility in many modules of the architecture. Different filters, edge detection methods, and contour representation methods may be used where deemed suitable.

Disadvantages:

- Small objects may be poorly rendered or lost completely .
- Line edges cannot be satisfactorily encoded without using texture. This undermines the principal philosophy of the approach, i.e. representing each component by the most appropriate method (FBC or DCT). A chain-coded contour with some line parameters (intensity, width) would be a more appropriate means of representing line edges. This is recommended as future work.
- An unpleasant blurring may be visible in some image regions, particularly where adjacent texture blocks partially restore mid-range frequencies, which then seem to "disappear" into smooth regions.

The hybrid scheme was designed for very low bit-rate coding of grey-level images and, as demonstrated in chapter six, for a certain class of images significant improvements in bit-rate and distortion may be achieved compared with JPEG. This has been accomplished while retaining a largely block based DCT approach. This research has shown that there is still "life" in the ageing transform coding approach to image compression. This may prove beneficial in the medium term while the industry continues to invest in DCT based technology for image and video coding. The scheme should not be considered dependent

on the DCT however as any waveform coding method may be substituted for the DCT within the framework of the codec architecture (DWT for example).

It should be noted that the compression ratios (up to 120:1) presented for the hybrid coder were obtained from sub-optimally configured codecs. With further research on a number of aspects of the codec, improvements over the results presented here should be expected. It is the strong belief of this author that further progress in the development of this codec architecture could lead to both reduced complexity, superior compression, and better image quality. We propose some directions for further work in the following section.

7.3 Future Work

The image coding scheme proposed in this thesis represents an approach with many unique issues. The principal issues have been investigated as part of this research, but the scope remains for a number of worthwhile investigations. A selection of suggestions for future work is presented below.

1. **Lossy contour coding**. The benefits of lossy contour coding may be investigated. For reasons of simplicity, in the extraction of the smooth component a lossless contour coding method is used. The use of a lossy contour coding method would require further analysis in the extraction of the smooth component.

2. **Edge-assisted smooth component filtering**. The considerable benefits of filtering the smooth component to remove compression artefacts following decoding were demonstrated. Unfortunately, relevant frequency content is also attenuated by this process. We propose that by using the edge map, which is of course available to the decoder, an edge-assisted method could be developed to enable better retention of "legitimate" high frequencies.

3. **Alternative Waveform Coders**. In this research, we emphasised the benefits of adopting a block based DCT approach. It would be interesting however, while retaining block processing in the codec architecture, to explore the use of alternative waveform coding methods for the smooth and texture components. In particular, the wavelet transform could be investigated for this purpose.

4. **Alternative Edge Extraction Methods**. All of the edge methods investigated in this research were gradient based methods. Edges are retained solely on the basis of the magnitude of their gradients. Our investigations have indicated however that the loss of weak contours surrounded by smooth texture represents a loss of visually salient information. This is particularly so when those contours are relatively long. Thus we believe that it would be worthwhile to consider an edge extraction method that could retain weak edges under such circumstances.

5. **Detection of isolated contours**. In the interest of codec simplicity and computational efficiency, the codec architecture was not designed to detect small closed contours contained fully within blocks. This means that visually important data may be lost completely while contour information is still recorded. Future work should address this shortcoming.

6. **Line edge extraction**. Separate line-edge extraction would seem to offer the potential for superior reconstructed image quality and better compression efficiency in the smooth component. These edges could be coded using more traditional sketch based techniques (i.e. intensity and geometry) and should be *removed* from the image prior to subsampling.

7. **Extension to Colour Images**. We have only considered the scheme for grey-level image compression. A logical progression would be extension to colour images. The principal difficulty here would seem to be generating correspondence, where appropriate, between the edge-maps of the different colour components. This problem would likely share a similar solution to suggestion 1 above.

8. **Extension to Video**. Further work on the proposed scheme should consider its extension to image sequences, i.e. video. It would be interesting to investigate how temporal variations in contours, or parts thereof, could be efficiently detected and coded.

The list above is be no means exhaustive. The novelty of the proposed scheme is such that a variety of interesting directions remain to be explored.

APPENDICES

A ORIGINAL IMAGES

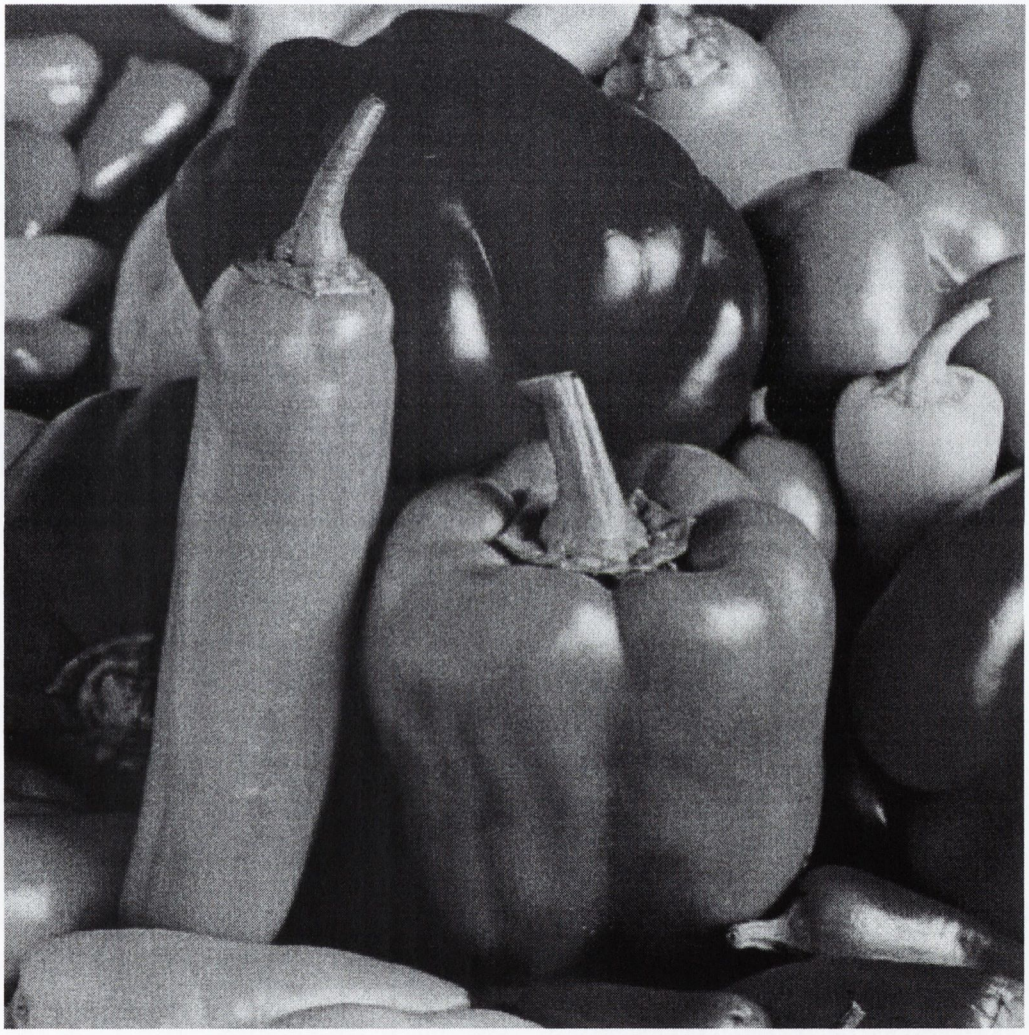
This appendix presents the original images referenced in chapter 6.

Lena 512x512

Boats 512x512

Horse 640x480

Boatsurf 640x480



Peppers 512x512

Plane 640 x480

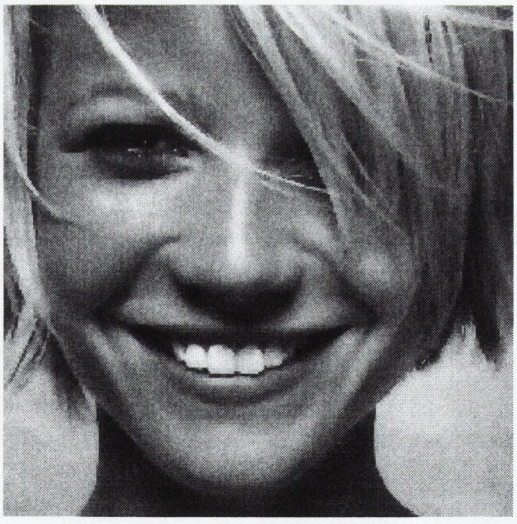
Mountain 640x480

Pig 352x512

Camman 256x256

Eye 256x256

Zelda 256x256



Gwyn 256x256

B IMAGE INTERPOLATION

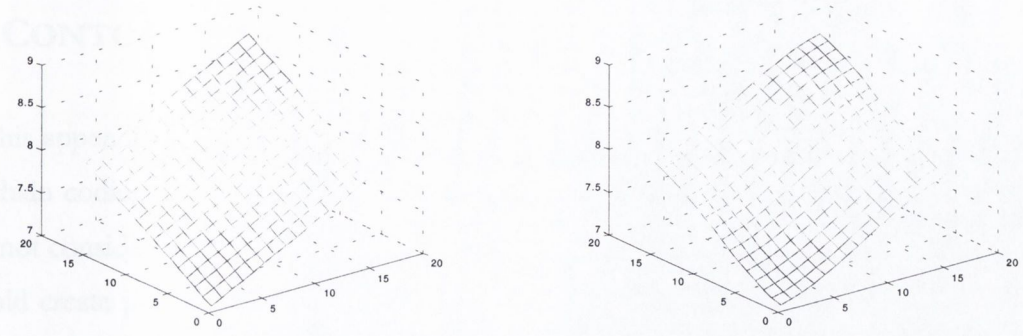
When an image is upsampled (enlarged), the "new" pixels between known sample points must be estimated. This process is known as *interpolation* or *image re-sampling*. A number of techniques exist for interpolation, the most common of which include:

1. *Nearest neighbour interpolation* (a.k.a. *zero-order interpolation*) in which an unknown pixel is assigned to the value of the closest known sample. This method produces severely blocky images for all but the most trivial cases however, and so shall not be considered further.

2. *Bilinear interpolation* (a.k.a. *first-order interpolation*) in which an unknown value is calculated as the weighted average of the four nearest sample points. The weighting of each sample point is inversely proportional to its distance from the unknown value.

3. *Bicubic interpolation* (a.k.a. *cubic convolution interpolation*) which applies a piecewise polynomial function to a 4×4 neighbourhood of known sample points. Bicubic interpolation therefore, uses the 16 nearest known samples to estimate the unknown pixel

The *area* over which an interpolation method needs to be computed, i.e. the number of known sample points required for the calculation, is referred to as its *support*. The standard interpolation methods listed above have supports of 1, 4, and 16 pixels respectively. The choice of interpolation method is generally a trade-off between better quality versus increased computation. Fig B.1 below presents examples of interpolating between 4 known points using bilinear and bicubic interpolation.



(a) (b)

Fig. B.1 Examples of bilinear (a) and bicubic (b) interpolation. In the case of bicubic interpolation, we have had to duplicate the four known pixel values to fill a 4×4 grid (i.e. 16 pixels)

Bilinear interpolation

Bilinear interpolation is computed from four known values by three linear interpolations (see Fig B.2).

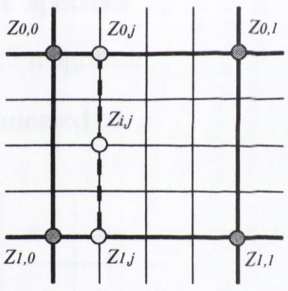


Fig. B.2 Calculation of an unknown pixel $z_{i,j}$ by bilinear interpolation

$$z_{i,j} = i \cdot z_{1,j} + (1-i) \cdot z_{0,j}$$

$$z_{0,j} = j \cdot z_{0,1} + (1-j) \cdot z_{0,0}$$

$$z_{1,j} = j \cdot z_{1,1} + (1-j) \cdot z_{1,0}$$

Bicubic interpolation.

Bicubic interpolation is a more sophisticated technique that fits a surface of the *sinc* type through a support of 16 known points. Bicubic interpolation consists of five one-dimensional cubic convolutions. Fig. B.3 shows the region of support for bicubic interpolation. One approach is to interpolate along the four rows first, and then interpolate along the column of the four interpolated points. Bicubic interpolations requires about 7 times more calculations than bilinear interpolation.

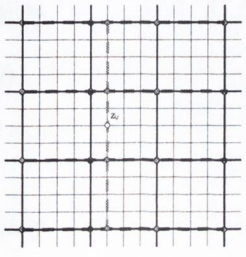


Fig. B.3 Region of support for bicubic interpolation.

C CONTOUR CODING

In this appendix we discuss a contour-oriented approach to contour/shape coding known as chain coding. Chain coding has both lossy and lossless modes of operation [85]. We did not consider employing lossy contour coding methods in this research, as this approach would create problems for the generation of the smooth-component using edge-sensitive subsampling. Accurate placement of contour pixels is needed to avoid erroneous sampling from the original image. While this issue really only arises where sample points lie close to contours, adopting a lossless approach to contour representation is more straightforward. Future work may explore the use of lossy contour coding methods.

C.1 Preliminaries

Before proceeding to discuss chain coding we need to define the concept of connectivity, that specifies how contour pixels may be connected to each other. Considering a 3×3 neighbourhood of pixels (Fig. C.1), we define which pixels the centre pixel z_5 may be connected to under three types of connectivity [15]:

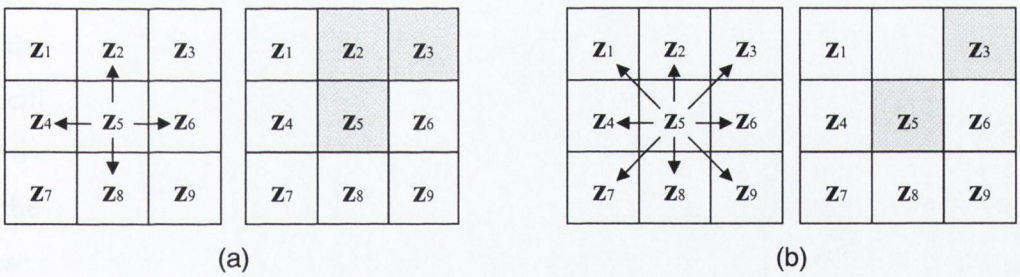


Fig. C.1 Examples of 4-connectivity (a) and 8-connectivity (b). In (a), z_5 may only be connected to its 4-neightbours, i.e. the pixels vertically or horizontally, but not diagonally. Thus diagonal links must be represented as a composition of horizontal and vertical links as shown. In (b) by contrast, z_5 may be linked directly to any of its 8-neighbours.

Connectivity

1. *4-connectivity* where z_5 may only be connected to z_2, z_6, z_8, or z_4 (North, East, South, or West). Thus, direct diagonal links are not allowed and must be made via two separate connections.

2. *8-connectivity* where z_5 may connect to any of its neighbours.

3. *m-connectivity* (mixed connectivity) where z_5 may be connected to any of its 4-neighbours, and any of its diagonal neighbours, but only if there is not already a link via 4-connectivity (see Fig C.2).

Fig. C.2 Ambiguity of 8-connectivity, addressed by m-connectivity. In (a), under 8-connectivity, z_5 may linked to z_3 by two different "routes" which causes problems when tracing contours. Under m-connectivity, the diagonal link from z_5 to z_3 is not allowed because these pixels connect to each other through 4-connectivity.

C.2 Chain Coding

In *chain coding*, or *Freeman coding* [111], a contour is represented by a sequence of *chain codes*. These codes are used to specify the direction that the contour takes from one contour pixel to the next. If 4-connectivity is specified, then the location of the next pixel may be specified as being in one of the four allowed directions from the current pixel. Similarly, for 8-connectivity, eight possible directions are permitted. A code is assigned to each one of the possible directions, and the contour specified in terms of a *start pixel* and sequence of these codes (Fig. C.3).

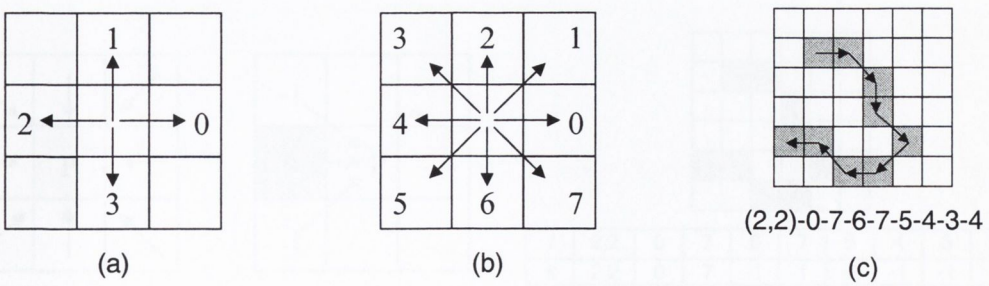


Fig. C.3 Chain codes for 4-connectivity (a) and 8-connectivity (b), with example of chain code sequence for 8-connected contour (c).

Chain coding in this manner produces a lossless representation of a contour, where each direction requires 2 or 3 bits per direction for 4 or 8-connectivity respectively. Chain coding based on 8-connectivity generally achieves superior compression to that based on

4-connectivity [113], thus the remainder of this discussion assumes 8-connectivity. Not counting the start pixel, a contour *C*, coded using chain codes, may be considered as an ordered set of *links* [113]:

$$C = \{l_i\} \quad i = 1,2,...,N$$

where l_i represents the i-th link in the chain of length N.

An adaptation of this approach, known as *derivative or differential chain coding* [85][112][113], specifies movements along a contour as *changes* in the direction of the contour with respect to the previous pixel. This approach generally achieves superior compression efficiency compared with standard chain coding [113]. For a contour coded using differential chain codes [113]

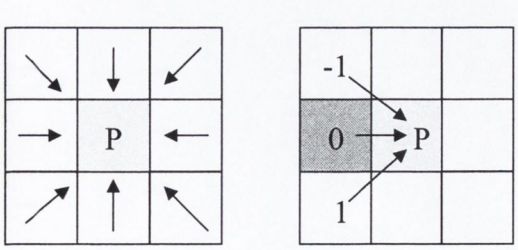
$$C = \{d_i\} \quad i = 1,2,...,N$$

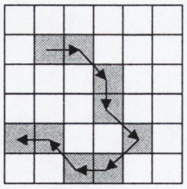
where

$$d_i = \begin{cases} k_i + 8 & if \quad k_i < -3 \\ k_i - 8 & if \quad k_i > 4 \\ k_i & otherwise \end{cases}$$

and

$$k_i = l_i - l_{i-1} \quad i = 2,3,...,N.$$





l	2,2	0	7	6	7	5	4	3	4
k	2,2	0	7	-1	1	-2	-1	-1	1
d	2,2	0	-1	-1	1	-2	-1	-1	1

(c)

Fig.C.4 Differential chain coding, a pixel P, is specified in terms of the change in direction from the current pixel. Possible directions (a) and common directions (b). Differential chain codes for contour, depicted in (c), the first direction is specified using normal chain codes.

Following differential chain coding, the codes of the representation may be efficiently coded using Huffman coding. While all directions are equally probable in standard chain coding, contours tend to exhibit repetitive directional changes that result in some direction codes being more frequent than others. Entropy coding thus yields compression benefits. In general, a bit rate of about 2 bits per link can be expected by this approach [113].

D GRADIENT OPERATORS FOR EDGE DETECTION

This appendix presents the gradient operator *kernels* for the three edge detection methods evaluated in chapter six. Edges are detected by all of the operators, by convolving an input image with two kernels, one to detect horizontal edges, and the other to detect vertical edges. These two edge-maps will consist of both positive and negative values representing rising or falling edges respectively. The *absolute* values of the "pixels" in the edge-map, will reflect the strengths of the edges they correspond to. A combined edge map may be computed by [9] :

$$E = \sqrt{(I * H_v)^2 + (I * H_h)^2}$$

where I is the original image and H_h and H_v are the kernels listed below. A binary edge-map may be generated from E, by thresholding and edge thinning.

D.1 The Sobel Operators

$$H_h = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix} \qquad H_v = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$$

D.2 The Prewitt Operators

$$H_h = \begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix} \qquad H_v = \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix}$$

D.3 The Roberts Operators

$$H_h = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \qquad H_v = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}$$

E SUMMARY OF MATRIX CONCEPTS IN LINEAR ALGEBRA

In this appendix we provide a brief summary of elementary matrix concepts in linear algebra. This is intended as a quick reminder supplement for the matrix algebra presented in chapter three.

1) The *inverse* of a matrix **A**, is the matrix that when multiplied by **A** produces the identity matrix, $AA^{-1} = A^{-1}A = I$.
   a) Only square matrices have inverses.
   b) A matrix that does not have an inverse is said to be *singular.*
   c) A matrix is *invertible* (or *non-singular*) if and only if it's determinant is non-zero.

2) The *transpose* of a matrix **A** is produced by exchanging each row in **A** with the corresponding column, row 1 becomes column 1, row 2 becomes column 2, etc. The transpose of **A** is denoted by A^T.
   a) A matrix is *symmetric* if it is equal to it's own transpose, $A = A^T$. A symmetric matrix is therefore a reflection about the main diagonal.
   b) A matrix is *orthogonal* if it is equal to the inverse of its own transpose, $A = \left(A^T\right)^{-1}$, or $A^{-1} = A^T$.
   c) If a matrix is both symmetric and orthogonal then $A = A^T = A^{-1}$

3) A matrix is *upper triangular* if all entries below the main diagonal are zero, *lower triangular* if all entries above the main diagonal are zero, and *diagonal* if all entries both above and below the main diagonal are zero.
4) The *complex conjugate* of a matrix **A**, with complex entries, is produced by replacing each entry in **A** with its conjugate. The complex conjugate of **A** is denoted $\overline{A}$.
   a) The transpose of a complex conjugate matrix, denoted A^*, is called an *adjoint* or *Hermitian conjugate* matrix. $A^* = \left(\overline{A}\right)^T$.
   b) If $A^*A = AA^*$ then A is known as a *normal* matrix.
   c) If $AA^* = I$ then A is a *unitary* matrix and $A^* = A^{-1}$.
   d) A *Hermitian matrix* is a matrix that is its own Hermitian conjugate, $A = A^*$. A Hermitian matrix is also normal.

5) The *linear combination* of the vectors $v_1, v_2, ..., v_n$ in a vector space V is defined by $\alpha_1 v_1 + \alpha_2 v_2 + ... + \alpha_n v_n$ where $\alpha_1, \alpha_2, ..., \alpha_n$ are scalars.

   a) The set of all linear combinations of $v_1, v_2, ..., v_n$ is called the *span* of $v_1, v_2, ..., v_n$.

   b) The span of $v_1, v_2, ..., v_n$ is a subspace of V.

   c) The minimal spanning set is the set of vectors in the vector space from which any other vector in the space can be generated. The set is minimal in that there are no redundant vectors in it. A minimal spanning set is called a *basis*.

   d) The vectors $v_1, v_2, ..., v_n$ in the vector space V are said to be *linearly independent* if $c_1 v_1 + c_2 v_2 + ... + c_n v_n = 0$, which implies $c_1, c_2, ..., c_n$ must equal 0.

   e) If $\{v_1, v_2, ..., v_n\}$ are a minimal spanning set of V, then they must be linearly independent.

   f) If two vectors in $\Re^2$ are *linearly dependent* then one may be written as a scalar multiple of the other (geometrically, they lie on the same straight line). If the vectors $x_1, x_2, ..., x_n$ represent the columns of a matrix $\mathbf{X}$ of dimension *n*, then they will be linearly dependent if and only if $\mathbf{X}$ is singular (i.e. the determinant is zero).

6) A mapping L from a vector space V to a vector space W is said to be a *linear transformation* or a *linear operator* if $L(\alpha v_1 + \beta v_2) = \alpha L(v_1) + \beta L(v_2)$. The mapping is denoted $L: V \rightarrow W$.

   a) For every mapping $\Re^n$ into $\Re^m$ there is an $m \times n$ matrix $\mathbf{A}$ such that $L(x) = Ax$ where $x \in \Re^n$.

   b) If $\mathbf{A}$ and $\mathbf{B}$ are $n \times n$ matrices and there exists a matrix $\mathbf{S}$ such that $\mathbf{B} = \mathbf{SAS}^{-1}$ then $\mathbf{A}$ and $\mathbf{B}$ are said to be *similar* matrices.

7) Two vectors in $\Re^n$ are said to be *orthogonal* if their *inner product*, also called the *scalar product*, is zero.

   a) If x and y are two vectors in a vector space, their inner product denoted $\langle x, y \rangle$, is a real number.

   b) The inner product of two vectors x and y in $\Re^n$ is defined as $x^T y$.

   c) If x and y are vectors in $\Re^2$ and their inner product is zero $x^T y = 0$, then x and y must be perpendicular to each other (orthogonal).

d) If $x \in \Re^n$, then the Euclidean length of x is defined by $\|x\| = \sqrt{x^T x} = \sqrt{x_1^2 + x_2^2 + ... + x_n^2}$.

e) The angle between two nonzero vectors x and y is given by $\cos\theta = \frac{x^T y}{\|x\| \cdot \|y\|}$.

f) Two orthogonal vectors x and y may be written $x \perp y$.

8) A vector space V with an inner product is called an *inner product space.*

a) If x and y are two vectors in V then the following conditions must hold true for their inner products

i) $\langle x, x \rangle \geq 0$ implies $x = 0$.

ii) $\langle x, y \rangle = \langle y, x \rangle$.

iii) $\langle \alpha \cdot x + \beta \cdot y, z \rangle = \alpha \langle x, z \rangle + \beta \langle y, z \rangle$.

b) In $C[a,b]$ the inner product may be defined $\langle f, g \rangle = \int_a^b f(x)g(x)dx$.

c) If $w(x)$ is a positive continuous function on $[a,b]$, then $\langle f, g \rangle = \int_a^b f(x)g(x)w(x)dx$ defines an inner product on $C[a,b]$ where $w(x)$ is a weighting function. Thus it is possible to generate many different inner products on $C[a,b]$.

d) If $x_1, x_2, ..., x_n$ are distinct real numbers, then the inner product of two polynomials p and q in P_n is defined by $\langle p, q \rangle = \sum_{i=1}^{n} p(x_i)q(x_i)$. If $w(x)$ is a positive function $\langle p, q \rangle = \sum_{i=1}^{n} p(x_i)q(x_i)w(x_i)$ also defines an inner product on P_n.

9) If V is vector space, it is said to be *normed linear space* is for each v in V there is associated a real number $\|v\|$ called the *norm* of v.

a) If V is a normed linear space then the following must hold true:

i) $\|v\| \geq 0$ if and only if $v = 0$.

ii) $\|\alpha v\| = |\alpha| \cdot \|v\|$.

iii) $\|v + w\| \leq \|v\| + \|w\| \quad \forall v, w \in V$.

b) If V is an inner product space, then for each v in V the norm of v is given by $\|v\| = \sqrt{\langle v, v \rangle}$.

c) If x and y are vectors in a normed linear space then the distance between x and y is given by $\|x-y\|$.

10) The *least squared solution*, denoted $\hat{x}$, of a $m\times n$ system $Ax=b$ where $m>n$ (i.e. over-determined) is the vector for which $\|r(x)\|$ is minimum where $r(x)$ is the *residual* defined by $r(x)=b-Ax$.

a) If $\mathbf{A}$ is an $m\times n$ matrix of rank n, then the *normal equations* defined by $A^TAx=A^Tb$, have a unique solution $\hat{x}=(A^TAx)^{-1}A^Tb$ and $\hat{x}$ is the least squared solution to the system $Ax=b$.

11) If the vectors $v_1,v_2,...,v_n$ belong to an inner product space V, and $\langle v_i,v_j\rangle=0$ whenever $i\neq j$ then $\{v_1,v_2,...,v_n\}$ are said to be an *orthogonal set* of vectors.

a) If $\{v_1,v_2,...,v_n\}$ are an orthogonal set of non-zero vectors in an inner product space V, then $v_1,v_2,...,v_n$ are linearly independent.

b) An *orthonormal* set of vectors is an orthogonal set of unit vectors. The set $\{u_1,u_2,...,u_n\}$ will be orthonormal if and only if $\langle u_i,u_j\rangle=\delta_{ij}$ where $\delta_{ij}=1$ for $i=j$ and zero otherwise.

c) Given any set of orthogonal non-zero vectors $\{v_1,v_2,...,v_n\}$ it is possible to form an orthonormal set $\{u_1,u_2,...,u_n\}$, by defining $u_i=\left(\frac{1}{\|v_i\|}\right)v_i$ for $i=1,2,...,n$.

d) If $B=\{u_1,u_2,...,u_k\}$ is an orthonormal set in an inner product space V, then B is an *orthonormal basis* for a subspace of V.

12) An $n\times n$ matrix $\mathbf{Q}$, is said to be an *orthogonal matrix* if the column vectors of $\mathbf{Q}$ form an orthonormal set in $\Re^n$.

a) If $\mathbf{Q}$, is an orthonormal matrix then $\mathbf{Q}$ is invertible and $\mathbf{Q}^{-1}=\mathbf{Q}^T$.

b) $\langle \mathbf{Q}x,\mathbf{Q}y\rangle=\langle x,y\rangle$

c) $\|\mathbf{Q}x\|_2=\|x\|_2$

d) $\det(\mathbf{Q})=\pm 1$

13) If $\mathbf{A}$ is an $n\times n$ matrix in the system $Ax=\lambda x$, and x is a non-zero vector in $\Re^n$, then λ is said to be an *eigenvalue* or *characteristic value* of $\mathbf{A}$, and x is said to be a *eigenvector* belonging to λ.

a) The equation $Ax = \lambda x$ can be written in the form $(A - \lambda I)x = 0$ which implies λ is a eigenvalue of **A** if and only if $(A - \lambda I)x = 0$ has a non-trivial solution.

b) The set of solutions to $(A - \lambda I)x = 0$ is $N(A - \lambda I)$ which is a subspace of $\Re^n$, called the *eigenspace* corresponding to the eigenvalue λ.

c) The equation $(A - \lambda I)x = 0$ will only have a non-trivial solution (and therefore an eigenvalue) if $A - \lambda I$ is singular, i.e. the determinant of $(A - \lambda I) = 0$.

14) Let X be the set of N vectors $x_1, x_2, ..., x_n$, in a real vector space.

a) The *mean vector* in X is defined by $m = E\{X\}$ where E is the expectation operator[†]. We can compute m by $m = \frac{1}{N}\sum_{k=1}^{N} x_k$.

b) *Covariance* is a statistical measure of the variance (the square of the standard deviation) of two random variables. It is frequently used to quantify the degree of correlation between two related variables.

c) The *covariance matrix* **C**, of X is defined by $C = E\{(X - m)\cdot(X - m)^T\}$. We can compute **C** by $C = \frac{1}{N}\sum_{k=1}^{N} x_k x_k^T - mm^T$.

d) The element c_{ii} of **C** (main diagonal entry) equals the variance of x_{ii} in X while the element c_{ij} (off the main diagonal) equals the covariance between the elements x_i and x_j in X.

e) The covariance matrix is real and symmetric, thus finding a set of N normal eigenvectors is always possible.

f) If elements x_i and x_j are uncorrelated, their covariance is zero so $c_{ij} = c_{ji} = 0$.

GLOSSARY

Terms in the thesis that contain a glossary entry are post fixed with a superscript cross symbol thus: term† .

Autoregressive Process – denoted AR(p) where p indicates the order of the process. Also known as a Markov process [45 p109]. In this thesis we deal with first-order Markov processes (AR(1)).

Circulant Matrix – A matrix in which each row is a cyclic permutation of the row above, and such that all the elements of the main diagonal are equal., illustrated for $n=4$ by

$$\begin{pmatrix} a & b & c & d \\ d & a & b & c \\ c & d & a & b \\ b & c & d & a \end{pmatrix}$$

CODEC – **Co**mpressor-**Dec**ompressor, an architecture/set of algorithms, implemented in hardware or software, that define how an input image is translated to and from a compressed representation.

Compression Ratio – the ratio of the uncompressed image size to the compressed image size.

$$\mathrm{CR} = \frac{uncompressed}{compressed}$$

Constrained Optimisation – applies to problems with restrictions on values that may be taken by certain variables or combinations of variables, with consequent restrictions on permissible values of the function itself.

Correlation matrix – a matrix representation of all correlations between pairs of variables. The entry r_{ij} is the correlation coefficient between the ith and jth variables. The matrix is symmetric with diagonal elements all unity.

Covariance - the average of the products of deviations for each data point pair. Covariance is calculated to determine the relationship between two data sets. Given two data sets X and Y, the covariance is calculated by

$$\text{cov}(X,Y) = \frac{1}{n}\sum_{i-1}^{n}(x_i - \mu_x)(y_i - \mu_y)$$

Decimation – The term decimation is somewhat of a misnomer as the term originally meant "reduction by one tenth" (particularly with respect to a population). In the field of DSP however, decimation has acquired the meaning "reduction by some integral factor".

Discrete-time – with respect to a signal, the term discrete-time implies a sampled signal. A digital image is thus a discrete-time signal, often generated by sampling a continuous time (analog) image.

Eigenvalue - the amount a particular eigenvector is scaled when transformed.

Eigenvector - a vector that when transformed by a matrix retains its direction but perhaps not length.

Expectation Operator – denoted $E(X)$ where X is a random variable, the expectation operator yields the first moment about the origin, also called the *mean value*. For a discrete random variable X taking on a finite set of values x_i with probabilities p_i the *expected value* is calculated by

$$E(X) = \sum_i p_i x_i$$

Hermitian Transpose – the Hermitian transpose of a matrix is found by first replacing each entry in the matrix with its complex conjugate, and then transposing the matrix. The Hermitian transpose is equivalent to normal transpose if the entries are real valued (i.e. not complex).

Hybrid scheme – a coding scheme employing a combination of two or more different coding techniques.

Krönecker delta function– denoted δ_{lk} the Kronecker delta function takes on the value *one* when $l = k$ and *zero* otherwise. In matrix form the Kronecker delta is simply the identity matrix of appropriate dimension.

$$\delta_{lk} = \begin{cases} 1 & l = k \\ 0 & l \neq k \end{cases}$$

Lagrange Multipliers – A means of evaluating the maxima or minima of a function $f(x_1, x_2, ..., x_n)$, subject to one or more equality constraints $g_i(x_1, x_2, ..., x_n) = 0$. The solution is found by minimizing $L = f + \lambda_1 g_1 + \lambda_2 g_2 + ...$ with respect to the x_i and λ_i, where the λ_i are Lagrange multipliers.

Markov Process – see autoregressive process.

Mean vector – the mean vector of a population is defined as $\mathbf{m}_x = E\{\mathbf{x}\}$. It is approximated for N samples by

$$\mathbf{m}_x = \frac{1}{N}\sum_{i=0}^{N-1}\mathbf{x}_i$$

Standard Deviation - a measure of how widely values are dispersed from the *mean*† value. Equal to the positive square root of the *variance*†.

$$\sigma = \sqrt{\frac{1}{N}\sum_{i=0}^{N-1}(x_i - \bar{x})^2}$$

Stationary process – a process for which the probabilistic behaviour is constant over time.

Stochastic – implying random variation, generally used to describe systems that are not deterministic. A stochastic process is a random process. In this thesis we sometimes model images as stochastic processes as successive pixels may conceivably take on any value.

Toeplitz Matrix – A square matrix with constant diagonals, illustrated for $n=4$ by

$$\begin{pmatrix} a & b & c & d \\ e & a & b & c \\ f & e & a & b \\ g & f & e & a \end{pmatrix}$$

Transcendental equation – a transcendental equation is one that is not *algebraic.* An algebraic equation is one in which variables appear as terms with coefficients that are subject only to the fundamental operations of addition, subtraction, multiplication, and division.

Variance – For a sample, the variance is the second sample moment about the *mean*† , $\sigma^2 = E\{[x-\mu]\}^2$. Equal to the square of the *standard deviation*†. For a population of N samples x_i, $i = 0,1,...,N-1$

$$\sigma_x^2 = \frac{1}{N}\sum_{i=0}^{N-1}(x_i - \bar{x})^2 = \frac{1}{N}\sum_{i=0}^{N-1}(x_i^2) - \left(\frac{1}{N}\sum_{i=0}^{N-1}x_i\right)^2$$

BIBLIOGRAPHY

[1] J.G. Proakis, D.M. Manolakis, *Digital Signal Processing: Principals, Algorithms and Applications.* New York: Macmillan Publishing Company, 2nd Edition, 1992

[2] C. Shannon, "*A mathematical theory of communication*", Bell Syst. Tech. J., vol. 27, no. 3, pp. 379-423, 1948

[3] T, Berger, *Rate Distortion Theory*, Englewood Cliffs: Prentice-Hall, 1971

[4] S.P. Lloyd, "*Least squares quantization in PCM*", IEEE Trans. Info. Theory, vol. 28, pp. 129-137, Mar 1982

[5] J. Max, "*Quantizing for minimum distortion*", IRE Trans. Info. Theory, vol. 6, pp. 7-12, Mar 1960

[6] A.C. Popat "*Scalar quantization with arithmetic coding*", M.Sc., MIT, June 1990

[7] D. Huffman, "*A Method for Construction of Minimal Redundancy Codes*", Proc. IRE, pp. 1098-1101, Sept. 1952

[8] T.A. Ramstad et al, "*Subband Compression of Images: Principles and Examples*", Advances in Image Communication, Elsevier, vol. 6, 1995.

[9] J.S. Lim, *Two-Dimensional Signal and Image Processing*, Prentice-Hall PTR, New Jersey, 1990.

[10] A.M. Tekalp, "*Digital Video Processing*", Prentice Hall Signal Processing Series, New Jersey, 1995.

[11] H-M. Hang et al, "*Handbook of Visual Communications*", AP Academic Press, 1992.

[13] A. Gersho, "*Vector Quantization and Signal Compression*", Boston: Kluwer Academic Publishers, 1992

[14] T.A. Ramstad et al, "*Block-based attractor coding: Potential and comparison to vector quantization*", Proc. Twenty-Seventh Asilomar Conference on Signals, Systems, and Computers, Nov. 1993

[15] R.C. Gonzalez et al, *Digital Image Processing*, Addison Wesley, 1993.

[16] M. Nelson, *The Data Compression Book*, M&T Books, Redwood City, 1991.

[17] S. Roman, *Introduction to Coding and Information Theory*, Springer Verlag, 1997.

[18] T.D. Lookabaugh et al, "*High-resolution quantization theory and the vector quantizer advantage*", IEEE Trans. Inform. Theory, volIT-35, pp. 1020-1033, Sept 1989.

[19] K.R. Rao, *Techniques & Standards for Image, Video & Audio Coding*, Prentice Hall PTR, 1996.

[20] K.R. Rao, P. Yip, *Discrete Cosine Transform: Algorithms, Advantages, Applications*, Academic Press, 1990.

[21] N.S. Jayant, P. Noll, *Digital Coding of Waveforms*, Englewood Cliffs, NJ: Prentice Hall, 1984.

[22] N. Ahmed, T. Natarajan and K.R. Rao, *Discrete Cosine Transform*, IEEE Trans. Computers, vol. C-23, pp. 90-93, Jan. 1974.

[23] A.K. Jain, "*A sinusoidal family of unitary transforms*", IEEE Trans. Pattern Anal. Machine Intell., vol. 1 pp. 356-365, Oct 1979.

[24] H.S. Malvar and D.H. Staelin, "*Reduction of blocking effects in image coding with a lapped orthogonal transform*", ICASSP 88, Int. Conf. On Acoustic., Speech, and Signal Process., pp. 781-784, New York, April 1988.

[25] H.S. Malvar and D.H. Staelin, "*The LOT: Transform coding without blocking effects,* " IEEE Trans. Acoust., Speech, and Signal Process., vol. ASSP-37, pp 553-559, April 1989.

[26] H.S. Malvar, "*The LOT: a link between block transform coding and multirate filter banks,* " Intl. Symp. On Circuits and Systems, pp. 835-838, Espoo. Finland June 7-9, 1988.

[27] N. Ahmed and K.R. Rao, "*Orthogonal transforms for Digital Signal Processing,* " New York: Springer-Verlag 1975.

[28] D.F. Elliot and K.R. Rao, "*Fast Transforms: Algorithms, Analyses and Applications*," New york: Academic Press, 1982.

[29] H.S. Malvar and D.H. Staelin, "*Signal Processing with Lapped Transforms*," Norwood, MA: Artech House, 1992.

[30] J.W. Woods, Ed., "*Subband Coding of Images*," Hingham, MA: Kluwer Academic, 1991.

[31] J. Shapiro, "*Embedded image coding using zerotrees of wavelet coefficients*," IEEE Trans. Signal Processing, vol. 41, pp3445-3462, Dec. 1993.

[32] K.R. Rao Ed., "*Discrete Transforms and their applications*,", NY: Van Nostrand Reinhold, 1985.

[33] K. Karhunen, "*Ueber lineare methoden in der Wahrscheinlichkeitsrechnung*," Ann. Acad. Sci. Fenn. Ser A.I. Math. Phys., vol. 37, 1947.

[34] M. Loeve, "*Probability theory*, " 2nd Ed., Princeton, N.J. Van Nostrand, pp 478, 1960.

[35] B. Nobel "*Applied Linear Algebra*", Prentice Hall Englewood Cliffs, N.J. 1969.

[36] J.B. Burl, "*Estimating the basis functions of the Karhunen-Loeve transform*," IEEE Trans. Acoust., Speech, and Signal Processing., vol. 19, pp. 99-105, Jan. 1989.

[37] P.A. Wintz, "*Transform Picture Coding*," Proc. IEEE, vol. 60, no. 7 pp.809-820, 1972.

[38] T. Ebrahimi and M. Kunt, "*Object-Based Video Coding*", Signal Processing Laboratory, Swiss Federal Institute of Technology – EPFL, CH-1015 Lausanne.
http://ltssg3.epfl.ch/~ebrahimi/ObjectBasedCoding/FinalPaper.htm

[39] D. Nelson, *The Penguin Dictionary of Mathematics*, Penguin 2nd Edition, 1998.

[40] B. Girod, R. M. Gray, J. Kovacevic, and M. Vetterli, "*Image and Video Coding*," IEEE Signal Processing Magazine, pp. 40-46, 56-57, March 1998.

[41] A.N. Skodras, C.A. Christopoulos, T. Ebrahimi, "*JPEG2000: The Upcoming Still Image Compression Standard*," Proc. 11th Portuguese Conf. On Pattern Recognition, pp. 359-366, 11th May 2000.

[41] C.S. Burrus, R.A. Gopinath, H. Guo, "*Introduction to Wavelets and Wavelet Transforms : A Primer*," Prentice Hall, 1998.

[42] P.N. Topiwala, "*Wavelet Image and Video Compression*," Kluwer Academic Publishers, 1998.

[43] M. Vetterli, J. Kovacevic , "*Wavelet and Subband Coding*," Prentice Hall-PTR, 1995.

[44] A.N. Netravali & B.G. Haskell, "*Digital Pictures*", New York: Plenum Press, 1988.

[45] A. Mertins, "*Signal Analysis: Wavelets, Filter Banks, Time-Frequency Transforms and Applications*" John Wiley & Son, 1999.

[46] W. D. Ray, R.M. Driver, "*Further decomposition of the Karhunen-Loeve series representation of a stationary random process*," IEEE Trans. Inform. Theory, vol. IT-16, pp. 663-668, Nov 1970.

[47] A.G. Tescher, R.V. Cox, "*An adaptive transform coding algorithm*," Intl. Conf. on Commun., pp. 47-23, 14 July 1976.

[48] K.N. Ngan, "*Adaptive transform coding of video signals*," IEE Proc., vol. 129, pp. 28-40, Feb. 1982.

[49] P.M. Cassereau, D.H. Staelin, and G.D. Jager, "*Encoding of images on a lapped orthogonal transform*," IEEE Trans. Commun., vol. 37, pp 189-193, Feb. 1989.

[50] JPEG Technical Specification: Revision (DRAFT), Joint Photographic Experts Group, ISO/IEC JTC1/SC2/WG8, CCITT SGVIII, August 1990.

[51] D.J. LeGall, "*MPEG: a video compression standard for multimedia applications*," Communications of the ACM, vol. 34, no. 4, pp. 46-58, April 1991.

[52] C. Fogg, D.J. LeGall, J.L. Mitchell et al., "MPEG Video Compression Standard," Chapman & Hall, New York, 1996.

[53] "*MPEG-2 Video Compression Standard. Generic Coding of Moving Pictures and Associated Audio*," ISO/IEC CD 13818-2, 1993.

[54] "*Video Codec for audiovisual services at p x 64 kbits/s,*" ITU-T Recommendation H.261, Geneva, 1990

[55] "*Video coding for narrow telecommunications channels at <64 kbits/s,*" ITU-T Recommendation H.263, 1996.

[56] Z. Wang, "*Fast algorithms for the discrete W transform and for the discrete Fourier transform,*" IEEE Trans. Acoust., Speech and Signal Process., vol. ASSP-32, pp. 803-816, Aug. 1984.

[57] P. Lee & F.Y Huang, "*Restructured recursive DCT and DST algorithms,*" IEEE Trans. Signal Process., vol. 42, pp. 1600-1609, July 1994.

[58] A. N. Skodras, "*Fast discrete cosine transform pruning,*" IEE Trans. Signal Process., vol. 42 pp. 1833-1836, July 1994.

[59] L.P. Chau & W.C. Siu, "*Efficient formulation for the realization of the discrete cosine transform using recursive structure,*" ICASSP, vol. 3, pp. 437-440, April 1994.

[60] P. Yip and K.R. Rao, "*Energy packing efficiency for the generalized discrete transforms,*" IEEE Trans., Commun., vol. COM-26, pp. 1257-1262, Aug. 1978.

[61] H.C. Reeve and J.S. Lim, "*Reduction of blocking effect in image coding,*" ICASSP 83, Intl. Conf., on Acoust., Speech, and Signal Process., pp. 1212-1215, Boston, MA, April 1983.

[62] A. Habibi, "*Hybrid coding of pictorial data,*" IEEE Trans. Commun., vol. COMM-22, pp. 614-624, May 1974.

[63] P. Pirsch, "*Design of DPCM quantizers for video signals using subjective tests,*" IEEE Trans. Commun., vol. COM-29, pp. 990-1000, July 1981.

[64] A. Habibi, "*An adaptive strategy for hybrid coding,*" IEEE Trans. Commun., vol. COM-29, pp. 1736-1740, Dec. 1981.

[65] J.W. Woods, "*Subband Coding,*" Kluwer Academic Press, 1990.

[66] M. Vetterli, "*Multi-dimensional subband coding: some theory and algorithms,*" Signal Processing, vol. 6, April 1984, pp. 97-112.

[67] J.W. Woods & S.D. O'Neil, "*Subband coding of images,*" IEEE Trans. Acoust. Speech Sig. Proc., vol. ASSP-34, Oct. 1986, pp. 1278-1288.

[68] P.P Vaidyanathan, "*Quadrature mirror filter banks, M-band extensions, and perfect reconstruction technique,*" IEEE Acoust. Speech Sig. Proc., vol. 4, pp. 4-20, 1987.

[69] M. Vetterli & D. LeGall, "*Perfect reconstruction FIR filter banks: Some properties and factorisations,*" IEEE Trans., Acoust. Speech Sig. Proc., col. 37, pp. 1057-1071, 1989.

[70] P.J. Burt & E.H. Adelson, "*The Laplacian pyramid as a compact image code,*" IEEE Trans., Commun., vol. 31, pp. 532-540, 1983.

[71] B. Girod, F. Hartung, U. Horn, "*Multiresolution coding of image and video signals,*" *Proc. European Signal Processing Conference,* Island of Rhodes, Greece, pp. 1957-1960, September 1998. *Invited paper*

[72] G. M. Schuster & A. K. Katsaggelos, "*Rate-Distortion Based Video Compression,*" Kluwer Academic Publishers, 1997.

[73] R. G. Lyons, "*Understanding Digital Signal Processing,*" Addison Wesley, 1997.

[74] J-F. Yang & C-P. Fan, "*Fast Structural two Dimensional Discrete Cosine Transform Algorithms,*" IEICE Trans. Fundamentals, Vol. E81-A, No. 6, June 1998.

[75] W.F. Schreiber, C.F. Knapp & N.D. Kay, "*Synthetic highs, an experimental TV bandwidth reduction system,*" J. SMPTE, vol. 68, pp. 525-537, Aug. 1959

[76] W.F. Schreiber, T.S. Huang, O. J. Tretiak, "*Contour Coding of Images,*" from "*Picture Bandwidth Compression,*" Ed. T.S. Huang & O. J. Tretiak, pp. 443-448. Gordon & Breach, Science Publishers, 1972.

[77] K.K. Yan & D.J. Sakrison, "*Encoding of images based on a two-component source model,*" IEEE Trans. Commun., vol. COM-25, pp.1315-1315, Nov. 1977.

[78] D.E. Troxel et al., "*A two-channel picture coding system: I – Real-time implementation,*" IEEE Trans. Commun., vol. COM-29, pp. 1841-1849, Dec. 1981.

[79] D.N. Graham, "*Image Transmission by Two-Dimensional Contour Coding,*" Proc. IEEE, vol. 55, No. 3, pp 336-346, 1967.

[80] M. Kunt, A. Ikonomopopoulos, & M. Kocher, "*Second generation image coding techniques,*" Proc. IEEE vol. 73, pp. 795-812, 1985.

[81] R. Sekuler & R. Blake, "*Perception,*", 3rd Ed., Mc Graw-Hill 1994.

[82] O.Egger, E. Reusens, T. Ebrahimi, and M.Kunt, "*Very Low Bit Rate Coding of Visual Information --- A Review ---,*" in ISCAS'95 Tutorial Book, Chapter 1.5. IEEE Press, May 1995.

[83] J.R. Casas and L. Torres, "*Strong edge features for image coding,*" in "*Mathematical Morphology and its Applications to Image and Signal Processing,*" R.W. Schafer P. Maragos and M.A. Butt, editors, pp 443--450. Kluwer Academic Publishers, May 1996.

[84] J.A. Robinson, "*Image Coding with Ridge and Valley Primitives,*" Trans. on Comm., vol. 43, no. 6, June 1995.

[85] V. Christopopoulos, "*Segmented Still Image and Video Coding*", Ph.D. Thesis, Vrije Universiteit Brussel, 1998.

[86] U.Y. Desai, M.M. Mizuki, et. al. "*Edge and Mean Based Image Compression*" A.I. Memo No. 1584, M.I.T. Nov. 1996.

[87] S. Carlsson, "*Sketch Based Coding of Grey Level Images,*" Signal Processing, vol. 15, pp. 57-83, 1988.

[88] William R. Hendee and Peter T. Wells, eds., "*Perception of Visual Information,*" Springer-Verlag, Second edition 1996.

[89] R.L. Gregory (Ed.) "*The Oxford Companion to The Mind,*" Oxford University Press, 1987.

[90] P.A. Kolers, "*Reading Pictures: Some Cognitive Aspects of Visual Perception,*" from "*Picture Bandwidth Compression,*" Ed. T.S. Huang & O. J. Tretiak, pp. 443-448. Gordon & Breach, Science Publishers, 1972.

[91] X. Ran and N. Farvardin, "*A Perceptually motivated Three-Component Image Model,*" Tech. Report TR 92-32, Systems Research Center, University of Maryland, 1992.

[92] J.M. Wolfe "*Visual search*", from "*Attention,*" H. Pashler (Ed.), London, University College London Press, 1996.

[93] B.L. Beard and A.J. Ahumada Jr., "*A Technique to Extract Relevant Image Features for Visual Tasks,*" Proc. IS&T.SPIE Electronic Imaging Symposium, pp 24-30, 1998.

[94] M. Bigger, O. Morris, & A. Constantinides, "*Segmented-image coding: Performance comparison with the discrete cosine transform,*" IEEE Proceeding 135, 2, pp. 121-132. 1988.

[95] R. Nevatia and K. Ramesh Babu, "*Linear feature extraction and description,*" Computer Graphics and Image Processing, vol. 13, pp 257-269, 1980.

[96] K. Bowyer, C. Kranenburg, and S. Dougherty, "*Edge Detector Evaluation Using Empirical ROC Curves,*" Computer Vision and Pattern Recognition '99, Fort Collins, Colorado. vol. 1, pp. 354-359. June 1999.

[97] C. Schmid, R. Mohr, and C. Bauckhage, "*Evaluation of Interest Point Detectors,*" Inter. Journal of Computer Vision, Vol. 37, No 2, pp. 151-172, 2000.

[98] M.D. Heath, S. Sarkar, T. Sanocki, and K.W. Bowyer, "*A robust visual method for assessing the relative performance of edge-detector algorithms,*" IEEE Trans. on Pattern Analysis and Machine Intelligence, vol. 19, no. 12, pp. 1338-1359, 1997.

[99] A.M. Lopez, F. Lumbreras, J. Serrat, and J.J. Villanueva, "*Evaluation of methods for ridge and valley detection,*" IEEE Trans. on Pattern Analysis and Machine Intelligence, vol. 21, no. 4, pp. 327-335, 1999.

[100] X. Ran and N. Farvardin, "*Low Bit-Rate Image Coding Using a Three-Component Image Model,*" Tech. Report TR 92-75, Systems Research Center, University of Maryland, 1992.

[101] C.A. Christopoulos, W. Philips, et. al "*Segmented Image Coding: Techniques and Experimental Results,*" Signal Processing: Image Communication, vol. 11, pp 63-80, 1997.

[102] W. Philips, C.A. Christopoulos, "*Fast Segmented Image Coding Using Weakly Separable Bases,*" Proceedings ICASSP, Vol. 5, pp 345-348, April 1994.

[103] M. M. Reid, R.J. Millar & N.D. Black, "*Second-Generation Image Coding: An Overview,*" ACM Computing Surveys, Vol. 29, No. 1, March 1997.

[104] A. Katsaggelos and N. Galatsanos (Eds.), "*Signal Recovery Techniques for Image and Video Compression and Transmission,*" Kluwer Academic Publishers, 1998.

[105] L. Atzori & F.G.B. De Natale, "*A Novel Method for the Recovery of Lost Visual Data using Correlated Edges Information,*" The 9th International Packet Video Workshop, New York, April 1999. http://www.research.att.com/~mrc/pv99/CONTENTS/PAPERS/ATZORI/PV99.HTM.

[106] Y. Wang, Q.F. Zhu, L. Shaw, "*Maximally smooth image recovery in transform coding,*" IEEE Trans. Commun., vol. 41, no. 10, pp. 1544-1551, Oct 1993.

[107] L.Torres, M. Kunt and F. Pereira, "*Second Generation Video Coding Schemes and their Role in MPEG-4,*" European Conference on Multimedia Applications, Services and Techniques, pp 799 - 824, Louvain-la-Neuve, Belgium, May 1996.

[108] P. Salembier, L. Torres, F. Meyer, C. Gu, "*Region-Based Video Coding Using Mathematical Morphology,*" Proc. of the IEEE, vol. 83, no. 6, pp 843-856, June 1995.

[109] W. Osberger, A.J. Maeder & D. McLean, "*A Computational Model of the Human Visual System for Image Quality Assessment,*" Proceedings DICTA-97, Auckland, New Zealand, pp. 337-342, December 1997.

[110] W.Osberger, A.J.Maeder and N. Bergmann, "*A Perceptually Based Quantization Technique for MPEG Encoding,*" Proceedings SPIE 3299 - Human Vision and Electronic Imaging III, San Jose, California, USA, January 1998.

[111] H. Freeman, "*On the Encoding of Arbitrary Geometric Configurations,*" IRE Trans. Electronic Computers 10, pp. 260-268, June 1961.

[112] H. Freeman, "*Computer Processing of Line Drawing Images,*" Computer Survey 6, pp. 57-98, March 1974.

[113] C. Le Buhan Jordan, S. Bhattacharjee, et al., "*Shape representation and coding of visual objects in multimedia applications: an overview*" in Annales des Telecommunications 53, No 5-6, pp. 164-178, May 1998

[114] J. Canny, "*A Computational approach to edge detection,*" IEEE Trans. on. Pattern Analysis and Machine Intelligence, vol. PAMI-8, pp. 679-698, November 1986.

[115] D. Ziou and S. Tabbone, "*Edge Detection Techniques - An Overview,*" Int. Journal of Pattern Recognition and Image Analysis, Vol. 8, No. 4, pp. 537-559, 1998.

[116] J.R. Casas, P. Salembier, and L Torres, "*Morphological interpolation for texture coding*," IEEE International Conference on Image Processing," vol. 1, pages 526-529, Washington DC, USA, October 1995.

[117] M. Gilge, "*Region-oriented transform coding (ROTC) of images,*" Proc. of ICASSP'90, pp. 2245-2248, April 1990.

[118] M. Gilge, T. Engelhardt, and R. Mehlan, "*Coding of Arbitrarily Shaped Image Segments Based on a Generalized Orthogonal Transform,*" Signal Processing: image Communication vol. 1 pp. 153-180, 1989.

[119] Z. Xiong, K. Ramchandran, et. al., "*A Comparative Study of DCT- and Wavelet-Based Image Coding,*" IEEE Trans. on Circuits and Systems for Video Technology, Vol. 9, No. 5, August 1999.