Nonuniform Weighted Subsampling for Digital Image Compression

Hani Sorial

A Thesis

in

The Department

of

Electrical and Computer Engineering

Presented in Partial Fulfillment of the Requirements

for the Degree of Master of Applied Science at

Concordia University

Montréal, Québec, Canada

ISBN 0-612-10897-X

ABSTRACT

Nonuniform Weighted Subsampling for Digital Image Compression

Hani Sorial

This thesis presents Weighted Subsampling (WS), a new nonuniform image subsampling method suitable for "region-of-interest" applications. The proposed technique uses a weighting function to change the sampling pattern so that more samples are taken in important regions of the image. The first order autoregressive model (AR(1)) is used to model the input image. The signal is weighted and the eigenvectors of the new autocorrelation matrix are computed. A sampling pattern is then deduced and applied to the image. The performance of the proposed technique is compared to uniform subsampling. Both techniques use the same number of samples. Subjective and objective measures show that the presented scheme is superior to uniform subsampling at the region of interest in the image.

In WS, the simulated annealing is used to derive the nonuniform sampling pattern. The large complexity of the algorithm required that this optimization be done offline. In this thesis, Fast Weighted Subsampling (FWS), a fast method to deduce the sampling pattern with a considerable lower complexity, is also proposed.

The compression achieved by the subsampling operation is extended using predictive lossless coding. In the latter, adaptive linear prediction based on the density of the subsampling pattern is used. Suitable predictors and Huffman codes are designed and simulation verifies the efficacy of the presented scheme.

Keywords: Image compression, subsampling, linear prediction, lossless coding.

To my mother and father

ACKNOWLEDGEMENTS

I would like to thank my supervisor Dr. William E. Lynch for his expert guidance throughout this work. I have been extremely fortunate to have had the chance to work with Dr. Lynch who provided me with the excellent technical assistance, and the encouragement to continue doing research. Because of this I have grown both in research ability and in self-confidence. I would also like to thank all the professors with whom I have interacted over the course of my studies at Concordia University.

This work has been supported by the Concordia Faculty Research Development Program, Grant number RO-20-I887 and NSERC Research, Grant number OGP0155217.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF SYMBOLS

T	Sampling period
t_n	Sampling instants
$x_a(t_1, t_2)$	A 2-D continuous signal
$x(n_1, n_2)$	A 2-D discrete signal
Ω	Frequency of a continuous signal
$X_a(\Omega_1, \Omega_2)$	Fourier transform of a 2-D continuous signal
ω	Frequency of a discrete signal
$X(\omega_1, \omega_2)$	Fourier transform of a 2-D discrete signal
h_j	Prediction filter coefficients
$e(n)$	Prediction error in a DPCM system
$E\{x\}$	Expected value of x
${\sigma_e}^2$	Prediction error variance
R_{av}	Average number of bits per source symbol
L_k	Code-word length assigned to the kth luminance level
P_k	Probability of occurrence of the kth luminance level
H	Entropy of the source
$\mathcal{R}^N$	Space of all vectors of N real components
R_{xx}	Autocorrelation matrix of the signal x
ρ	Intersample correlation coefficient
x_p	Projection of the signal x
A^T	Transpose of the matrix A
u	Weighted signal
$W(i)$	Weighting function
$diag[W]$	Diagonal matrix of the vector W
R_{uu}	Autocorrelation matrix of the weighted signal
$\mathcal{C}$	A configuration in simulated annealing algorithm

D	Determinant value
T_c	SA control parameter (temperature)
α_{T_c}	SA temperature reduction coefficient
l_k	The kth eigenvector
λ_k	The kth eigenvalue
y	KLT coefficient vector
$\hat{u}$	Reconstruction of the weighted signal u
$\mathcal{I}$	The set of all indices
w	Window used in reconstruction filter
$\mathcal{M}(n)$	The nth Interpolation filter multiplier
$\hat{x}$	Reconstruction of the signal x

LIST OF ACRONYMS

pel	Picture element
TC	Transform Coding
VQ	Vector Quantization
SBC	Subband Coding
DWT	Discrete Wavelet Transform
LPC	Linear Predictive Coding
HVS	Human Visual System
KLT	Karhunen-Loève Transform
AR(1)	First Order Autoregressive model
DCT	Discrete Cosine Transform
WHT	Walsh-Hadamard Transform
DFT	Discrete Fourier Transform
JPEG	Joint Photographic Experts Group
MPEG	Moving Pictures Experts Group
PSNR	Peak Signal-to-Noise Ratio
MATLAB	Matrix Laboratory Software
DPCM	Differential Pulse-Code Modulation
WSS	Wide-Sense Stationary
VLC	Variable Length Coding
JPEG	Joint Photographic Experts Group
MSE	Mean Square Error
WS	Weighted Subsampling
KLT	Karhunen-Loève Transform
SA	Simulated Annealing
CPU	Central Processing Unit
WSP	Weighted Subsampling using Projection

LS	Least Squares
FWS	Fast Weighted Subsampling

Chapter 1

Introduction to Image Compression

1.1 Introduction

Human beings perceive most of the information about the world around them through their visual sense. The amount of data associated with visual information is so huge that its storage would require enormous storage capacity. Further, transmission of such data require large bandwidth, which could be very expensive. However, this data contains considerable redundancy and therefore, compression is possible. Image compression techniques are concerned with reduction of the amount of data required to store or transmit an image while maintaining an acceptable image quality, and consequently, save transmission and storage resources. Image compression plays a crucial role in many important applications including broadcast television, teleconferencing, multimedia computing, facsimile transmission, remote sensing via satellite, medical images, etc.

1.2 Problem Statement

Subsampling is a simple, low complexity image compression method where only a subset of the pixels are retained, that is, the digital image is represented on a new sampling lattice with a lower sampling density than the original lattice. The sampled image can then be processed, stored, or coded for transmission. At the receiver, an interpolation is carried out to reconstruct an approximation of the original image. The general goal of a sampling system is to give the best possible reproduction of the original image by a proper choice of preprocessing, sampling strategies, and interpolation [1].

Uniform subsampling saves regularly spaced sample locations. This is suitable when all areas in the image are equally important. In some applications however, where certain regions are more important than others, it is desirable to represent the image by a nonuniform sampling scheme that assigns more samples to the important areas. Consider for example an image of a human face superimposed on a simple background. The background carries little information and can be represented by coarse sampling. The face however carries more important information. Therefore, the remaining samples can be assigned to the face region to enhance its quality in reconstruction.

This thesis is concerned with digital image compression using a nonuniform subsampling scheme suitable for "region-of-interest" applications. The class of images under consideration, whose information varies with spatial coordinates, is represented by samples distributed nonuniformly with higher sampling density in the region of interest and a lower density in other areas.

In this work, it is assumed that the part of interest is known *a priori* to be located in a particular region of the image. This is an essential assumption for Weighted Subsampling (WS) scheme presented in Chapter 3 where the sampling pattern is derived using the simulated annealing optimization method, as the complexity of the algorithm required that this optimization be done offline. However, in Fast Weighted Subsampling (FWS) heuristic presented in Chapter 4, this a priori requirement is of less vital importance as no optimization is needed in deriving the sampling pattern.

As in subsampling we deal with real images which are usually full band signals, the image has to be restricted to a certain space prior to subsampling. In the case of uniform subsampling, and from the classical sampling theorem [2], the images are restricted to the space of *bandlimited* signals. A *space-invariant* lowpass filter is used for proper image conditioning (bandlimiting) prior to uniform subsampling in order to prevent *aliasing* effects. For the nonuniform subsampling scheme; the class of images (which are not bandlimited) are said to belong to the space of *locally bandlimited* signals [3, 4]. Image conditioning prior to nonuniform subsampling is achieved using a *space-varying* lowpass filter. Thus, images in the space of locally bandlimited signals are characterized as having a space-varying bandwidth [3, 4].

The bandlimiting process reduces the bandwidth of the original image and therefore results in some degradation. Thus, the subsampled image will no longer represent the original but rather the bandlimited image. Hence, for a given number of samples (i.e. for a given compression ratio), assigning a higher sampling density to the region of interest in the image, then locally bandlimiting the signal, prior to subsampling, using a space-varying lowpass filter, result in less degradation in this region than if, for the same number of samples, uniform subsampling is used. This is illustrated in the following example.

Figure 1.1 shows the original Lena image of size 256×256 and 8 bits/pixel. Figures 1.2 and 1.3 show locally bandlimited and bandlimited Lena images prior to nonuniform and uniform subsampling respectively. In the former, a space-varying lowpass filter is used. Here, image conditioning (bandlimiting) for both cases are done in order to retain a subset of size 64×64 of the pixels (and therefore, to achieve a compression ratio of 16 for each case). In Figure 1.2, the region of interest (the rectangular area) is assigned more samples than the other areas (details on this will be given in Chapter 3). Note the difference in the quality of this region for both image conditioning cases. It can be seen that the region of interest in Figure 1.2 has more high frequency details than in Figure 1.3. Consequently, the quality of the corresponding reconstructed region using nonuniform subsampling will be superior to the same region reconstructed using uniform subsampling. This idea will be developed throughout the thesis.

The theory in this thesis will be developed in terms of one-dimensional sequences and systems (unless otherwise stated). The generalization to two-dimensions will be achieved in a separable fashion, i.e., the two-dimensional operation will be performed as two successive applications of one-dimensional operation: first performed on each row, and then on each column of the image.

Figure 1.1: Original Lena image (256 × 256; 8 bits/pixel)

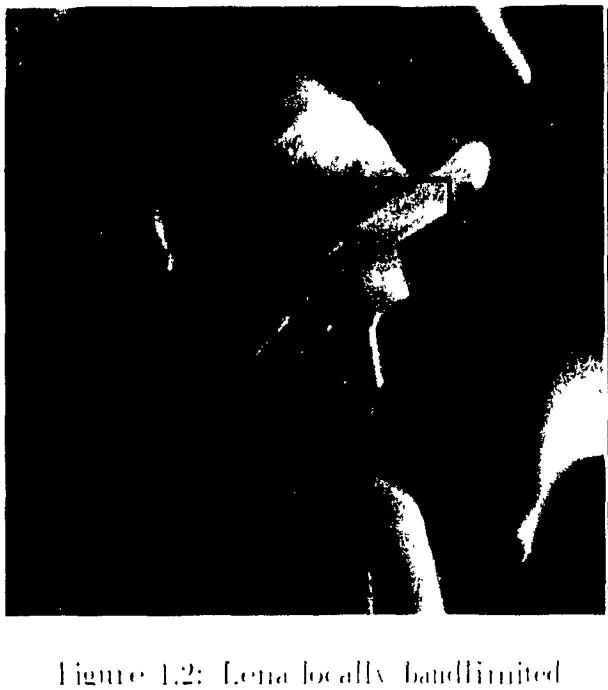


Figure 1.2: Lena locally bandlimited (256 × 256; 8 bits/pixel) using a space-varying lowpass filter. Region of interest is the rectangular area.

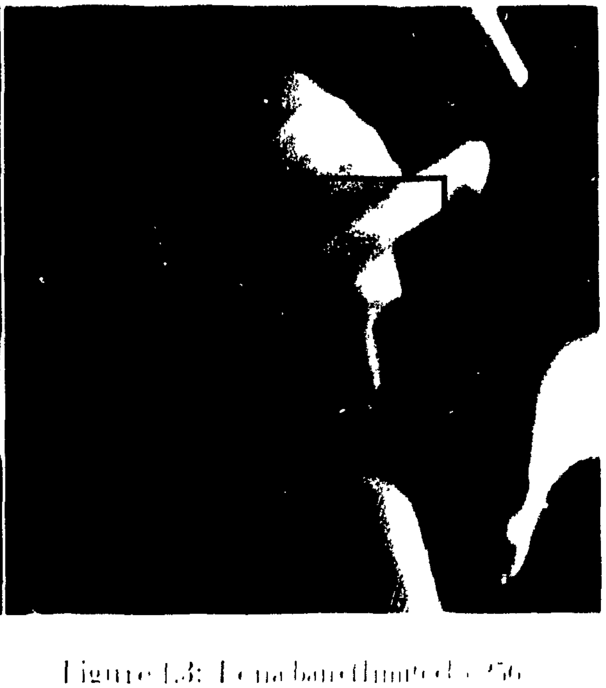


Figure 1.3: Lena bandlimited (256 × 256; 8 bits/pixel) using a space-invariant lowpass filter. Region of interest is the rectangular area.

1.3 Digital Image Representation

Visual information is expressed mathematically by numerically representing the distribution of light energy and wavelengths of an image field. For storage within a digital media, processing by a digital computer, or transmission via a digital communication channel, an image field is digitized both in spatial coordinates and in light intensity (brightness), that is, the image is *sampled* on a discrete grid and each sample is *quantized* using a finite number of bits. In this thesis, only monochrome images will be considered.

A digital image is represented by a two-dimensional array of numbers. The elements of such a digital array are called *picture elements, pixels,* or *pels.* In standard images, each sample is uniformly quantized with 256 quantization levels (grey levels). Hence, the luminance is represented by integer numbers between 0 and 255. A value of zero corresponds to black, and 255 corresponds to white. A monochrome digital image can be considered as a matrix whose row and column indices give the position of the pixel, and the corresponding value represents the grey level at that position. An example illustrating the axis convention used throughout this thesis is shown in Figure 1.4. Note that the notations (n_1, n_2) and (i, j) will be used alternatively to denote the axis of a digital image. In the next section, some popular digital image compression techniques are presented in brief.

1.4 Popular Image Compression Techniques

Many Image compression techniques have been developed and studied extensively. Some popular techniques include Transform Coding (TC), Vector Quantization (VQ), Subband Coding (SBC), Discrete Wavelet Transform (DWT), Linear Predictive Coding (LPC) and some Hybrid combinations of these. A brief review of the basics of each of the above techniques follows.

Transform coding has been studied extensively and has shown a relatively good

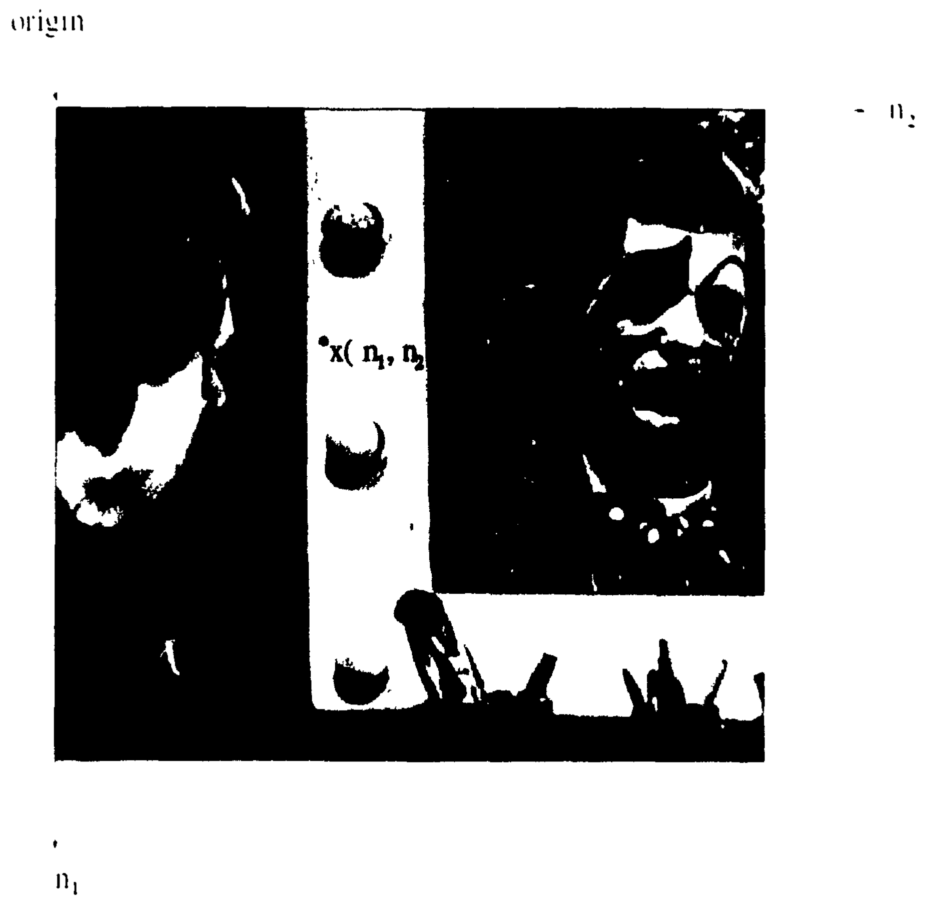


Figure 1.1: Axis convention used throughout the thesis

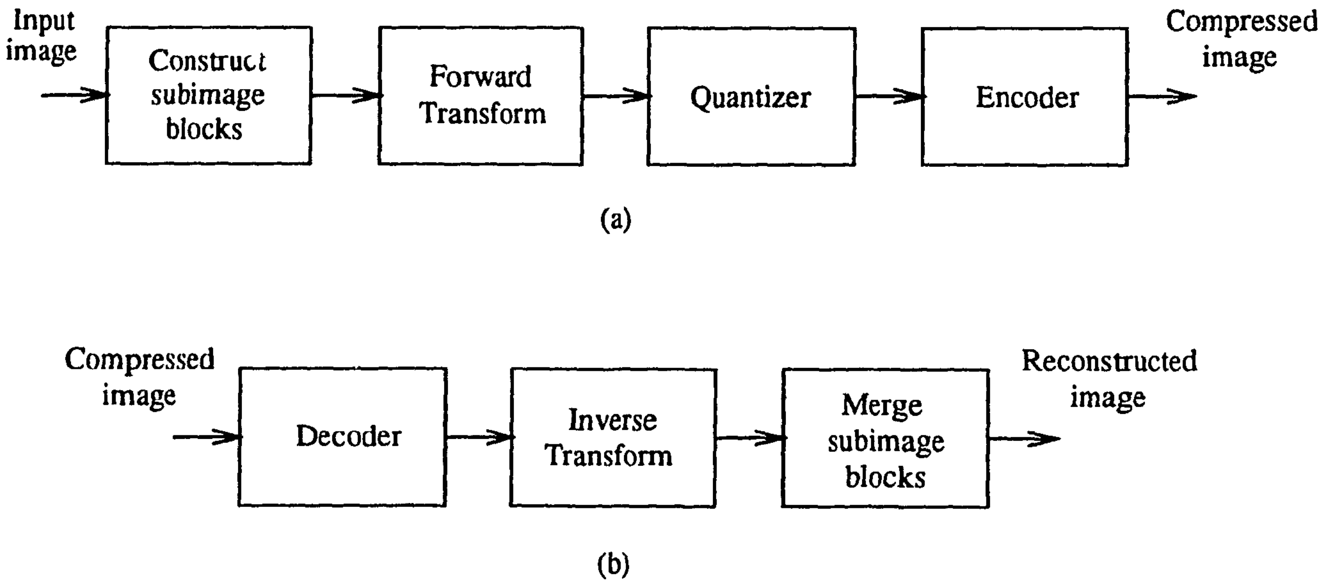


Figure 1.5: A transform coding system: (a) encoder, (b) decoder.

capability for bit rate reduction [5]. Figure 1.5 shows a typical transform coding system. In this technique, a block of dependent data is transformed into a set of less correlated coefficients. Often, the transform is linear and orthogonal. A large fraction of the total energy of the data is packed into relatively few coefficients. These coefficients are generally related to the spatial frequencies in the image. Since the human visual system (HVS) is less sensitive to errors in the higher spatial frequencies than to errors in the lower frequencies, the high-frequency coefficients are quantized more coarsely than the low-frequency ones to reduce the bit rate. The quantized coefficients are then coded (usually using variable-length coding) and transmitted. The decoder implements the inverse sequence (except the quantization) of the encoder to reconstruct the image. The optimum transform in a mean-square sense, is one that minimizes the mean square reconstruction error for a given number of total bits, resulting in the Karhunen-Loève Transform [6, 7]. However, the KLT is data dependent and of relatively high computational complexity. A solution to data dependency is to use statistical image models (e.g. first order Autoregressive model AR(1) [6]). Data independent transforms such as Discrete Cosine Transform (DCT),

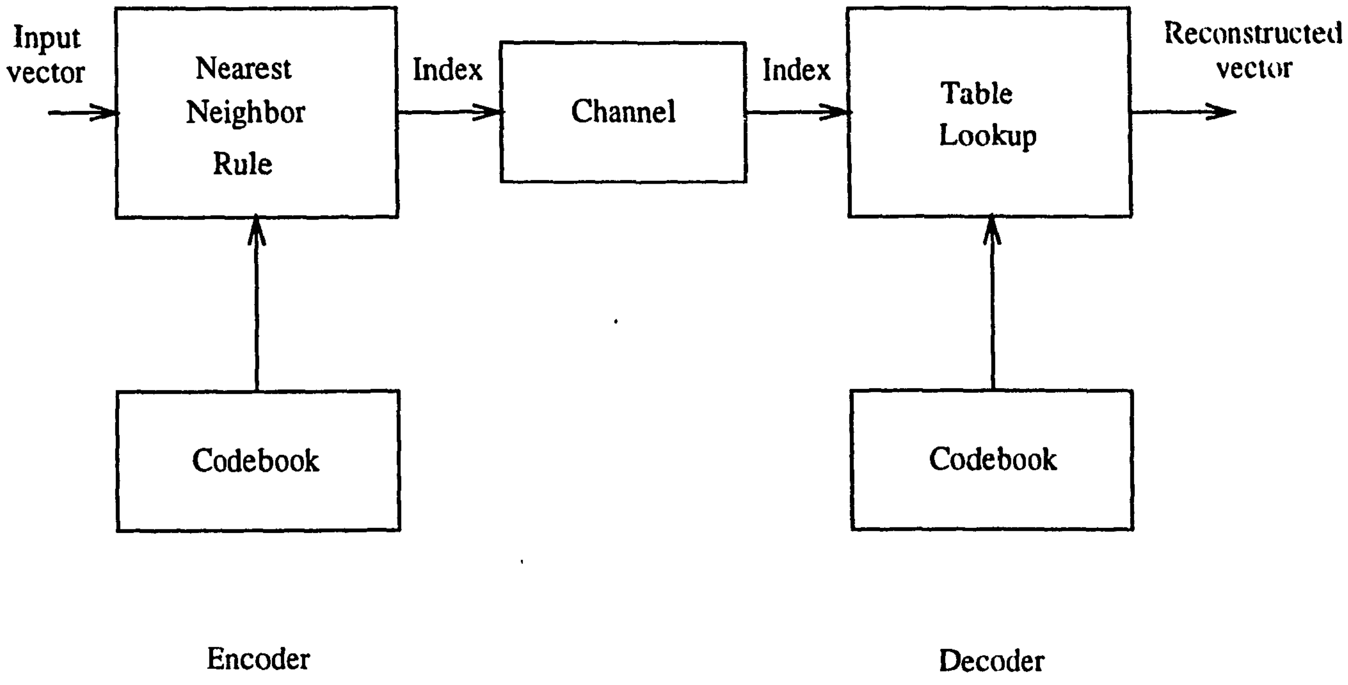


Figure 1.6: Vector Quantization.

Walsh-Hadamard Transform (WHT), and the Discrete Fourier Transform (DFT) are simple and practical to use. The DCT has proved to be of such practical value that it has become the international standard for transform coding systems [5]. The Joint Photographic Experts Group (JPEG) [8] and the Moving Pictures Experts Group (MPEG) [9] have produced their popular DCT-based compression standards for still images and video sequences, respectively. The DCT provides a good compromise between the ability of packing the data into few coefficients and the computational complexity.

Vector quantization is another popular image compression technique. Two excellent reviews on VQ can be found in [10, 11]. A survey that examines a number of variations on the VQ design algorithms to allow for the incorporation of image processing into the compression system is given in [12]. A block diagram of vector quantization is shown in Figure 1.6. In a vector quantizer, the input vector is compared to the entries of a codebook containing representative (also called reproduction) vectors. The encoder determines the closest match (the code vector) according to some distortion criteria. The most commonly used distortion measure is the Mean Square Error (MSE). The index of the code vector is then transmitted. At the receiver, the decoder also has a copy of the codebook and operates as a simple table lookup. The index is used to extract the code vector which represents an approximation of the original input vector. The codebook in VQ is usually based on a training set of typical data. The main advantage of vector quantization is the simple decoder structure. The disadvantage is coder complexity, that is, the large effort required to search the whole codebook in order to find the closest match. This complexity grows exponentially with vector dimension. Further, the optimization of the codebook itself involves lot of computations. Another disadvantage is the fact that images dissimilar to those in the training set may not be well represented by the reproduction vectors in the codebook.

Subband coding is a multi-frequency decomposition scheme which has been shown to be an efficient technique for image coding [13]. A good review of the basic principles of subband coding can be found in [6]. Figure 1.7 shows a block diagram of a subband coding system. The general idea of subband coding is to decompose the frequency band of a signal into a number of subbands using a bank of bandpass filters. Each subband is then subsampled (decimated or downsampled) and encoded appropriately. At the receiver, the encoded subbands are interpolated (upsampled) and then passed through reconstruction filters. The output signals from these filters are summed to give a close replica of the original signal. Special care should be taken in designing the bank filters for signal splitting and reconstruction so that aliasing errors are explicitly canceled out in the reconstruction stage [14]. A main advantage of subband coding is that by appropriately allocating the number of bits in different subbands, the overall reconstruction error spectrum can be shaped as a function of frequency. Further, the quantization error in encoding a subband is

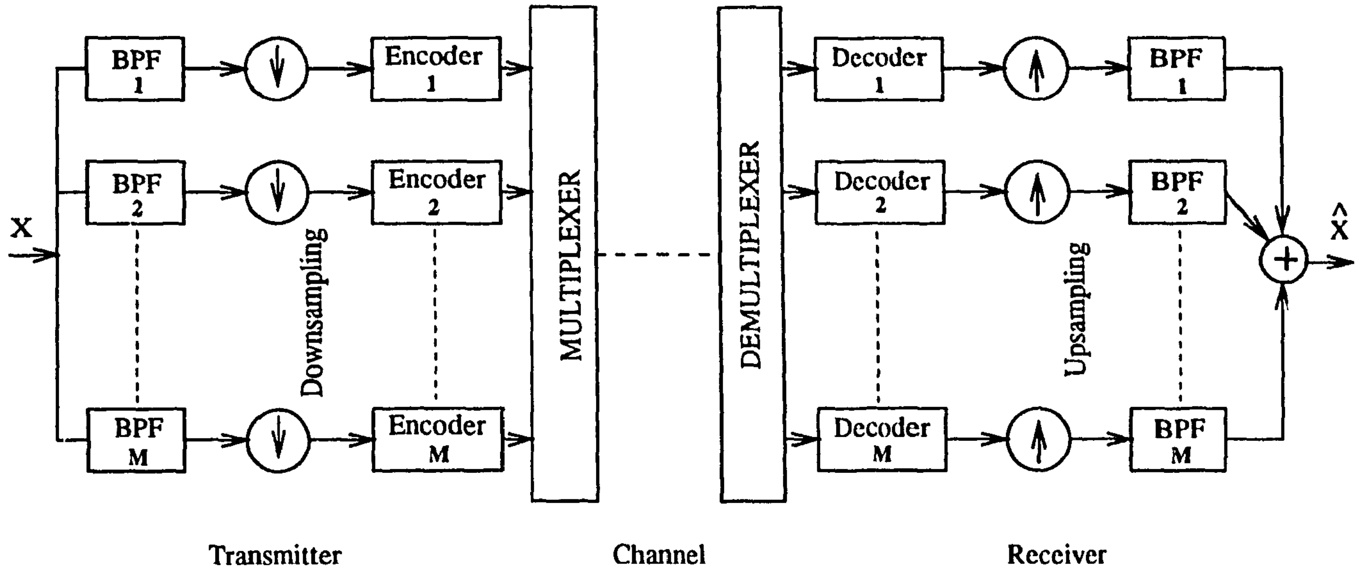


Figure 1.7: Block diagram of Subband coding.

contained within the subband, and hence doesn't mask a weak signal in another subband. This advantage offers perceptual improvement of the coding scheme.

The discrete wavelet transform (DWT) is a powerful technique for decomposing images into multi-resolution approximations [15, 16]. Multi-resolution decomposition has shown to be very effective for high-quality image coding at low bit-rate. The basic idea of the discrete wavelet transform is that of successive approximation, together with that of "added detail". At each stage, the input signal is decomposed into a lowpass approximation and an "added detail" signal (which can be considered as a highpass version of the input). Hence, the discrete wavelet transform decomposes the signal into a set of frequency subbands. Like subband coding, the DWT has a perceptual advantage. By appropriately allocating the number of bits in each band, the overall reconstruction error spectrum can be controlled so as to achieve a perceptual improvement in the reconstructed image.

Linear Predictive Coding (LPC) is a method that exploits the statistical redundancy expressed in the inter-sample correlation property of images. LPC is used in almost all lossless compression techniques. In predictive coding schemes, the current pixel value is predicted from some N previous pixels ($N = 1, 2, ..$) and the difference between the actual and the predicted pixel value is coded and transmitted. As LPC is used in this thesis to achieve lossless compression, Chapter 2 will have more details on its theory. Next section presents two fidelity measures used for evaluation of image quality.

1.5 Image Fidelity Measures

Image fidelity measures are useful for evaluating image quality and for rating the performance of an image compression scheme. There are two types of fidelity measures: subjective and objective. The subjective measures use rating scales such as goodness scales and impairment scales [17]. Subjective evaluations are the most reliable image fidelity measures as the end user is usually a human observer. In this thesis, images resulting from different compression schemes are presented and are left to the reader for a subjective comparison.

As yet there are no agreed upon objective measures of subjective goodness, the peak signal-to-noise ratio (PSNR) [5] is used in this work as an objective measure of image fidelity. This is given by

$$PSNR = 10\log_{10}\frac{255^2}{\frac{1}{N^2}\sum_{i=1}^{N}\sum_{j=1}^{N}\left[x(i,j) - \hat{x}(i,j)\right]^2} \tag{1.1}$$

where N^2 denotes the number of pixels in the original image, and $\hat{x}(i,j)$ represents the reconstructed value of the original $x(i,j)$.

1.6 Organization of the Thesis

This thesis is organized as follow:

Chapter 2 covers background material related to the work presented in this thesis. The chapter includes two parts: image sampling and reconstruction, and predictive coding. In the first part, the sampling and reconstruction of two-dimensional continuous signals is considered. Next, the theory of subsampling (decimation) and interpolation of two-dimensional discrete signals with application to digital images is presented. This part also deals with aliasing effects that result when the sampling theorem is violated.

In the second part, the theory of linear prediction and its application to images is discussed. The two-dimensional prediction of image models with isotropic and separable autocorrelation functions is also considered. Next, the theory of variable length coding (entropy coding) and in particular Huffman coding is reviewed. Many illustrative examples are given throughout this chapter.

Chapters 3 and 4 represent the contributions of this thesis. Chapter 3 is divided into two parts. First, nonuniform Weighted Subsampling (WS) for digital image compression [18], the topic of this thesis, is presented. The idea of WS, including the algorithm used to derive the sampling pattern, is first described. Next, Weighted Subsampling using Projection (WSP) is presented. Here, the subsampled signal is the projection of the weighted input signal. Numerical examples show how the projection can be exactly reconstructed from its samples. An illustrative example using test images is also given. Next, signal projection is approximated in WS using a space varying linear low pass filter. Image reconstruction using interpolation is also treated. Finally, experimental results including subjective and objective evaluations illustrate WS. A comparison between WSP (using projection) and WS (using filtering) is also considered.

The second part of this chapter extends the compression achieved by WS using lossless predictive coding. In this method, suitable predictors (based on the density of the subsampling pattern) and Huffman codes are designed. Lossless compression ratios of 1.4 are achieved yielding an overall compression ratio (including the subsampling) of 22 (0.36 bits/pixel).

Chapter 4 presents Fast Weighted Subsampling (FWS), a fast heuristic to derive the sampling pattern of WS (presented in Chapter 3). In the latter, simulated annealing was used to deduce the sampling pattern. The large complexity of the algorithm required that this optimization be done offline. The heuristic presented in this chapter has much lower complexity than the simulated annealing. To deduce a sampling pattern having M samples for an N-dimensional vector ($M < N$), FWS requires NM multiplications and additions, and $2N$ comparisons. Many illustrative examples are given throughout this chapter using different weighting functions. Finally, subjective and objective evaluations compare the results obtained in WS (using simulated annealing) to those obtained using FWS.

Chapter 5 concludes the thesis by summarizing the proposed nonuniform subsampling scheme and discussing its performance and contributions. This chapter also includes future work.

The simulation programs are given in Appendix A. All simulations were done using MATLAB, Version 4.2c., on a SPARC 5 Sun station.

Chapter 2

Background

2.1 Introduction

The material covered in this chapter includes two parts. First, two-dimensional sampling of continuous and discrete signals with application to images is examined. Section 2.2.1 reviews the one-dimensional Shannon sampling theorem which addresses the problem of uniform sampling and reconstruction of *bandlimited* signals. A direct extension of the one-dimensional sampling to the two-dimensional case is uniform sampling on a rectangular grid discussed in Sections 2.2.2 and 2.2.3. Section 2.2.4 deals with aliasing effects which result when a signal is undersampled, i.e., sampled below its *Nyquist* rate (the Nyquist rate is equal to twice the highest frequency contained in the signal). As subsampling is the topic of this thesis, it is essential to review the theory of decimation (subsampling) and interpolation of two-dimensional discrete signals. This is covered in Sections 2.2.5 and 2.2.6. Aliasing effects in image subsampling are also considered in Section 2.2.7. Next, an example of uniform subsampling of images is presented in Section 2.2.8.

The second part in this chapter introduces predictive coding and its application to images. Predictive coding is used in this work to extend the compression achieved by the nonuniform subsampling method presented in Chapter 3. Section 2.3 reviews the theory of linear prediction. An example of two-dimensional prediction for image models with isotropic and separable autocorrelation functions is presented at the end of this section. Finally, Section 2.3.3 reviews briefly the theory of variable length coding and in particular Huffman coding. Many examples illustrate the material presented throughout this chapter.

2.2 Image Sampling and Reconstruction

For suitable storage, processing or transmission, a continuous image field $x_a(t_1, t_2)$ has to be digitized both spatially and in amplitude. Digitization of the spatial coordinates (t_1, t_2) is known as sampling. The set of sample points might be a rectangular array [2] or other form of spatial sampling (e.g. hexagonal sampling [19]). In some applications, the sample locations are nonuniformly chosen to lie within a certain "region-of-interest" according to some given criterion [4]. In all cases, after processing or transmission, these samples are used to reconstruct a continuous image for viewing. A simple method of image compression is to further sample the digital image, i.e. to retain only a subset of the samples, and then use the retained samples for reconstruction at the receiver. Sampling and reconstruction of digital images are known as subsampling and interpolation respectively. In the following, both continuous and digital images are considered for the analysis of image sampling and reconstruction methods. Analysis is restricted to rectangular sampling lattices.

2.2.1 The Shannon Sampling Theorem

The Shannon [20] sampling theorem (also known as the Whittaker-Kotel'nikov-Shannon (WKS) sampling theorem) addresses the problem of uniform sampling and reconstruction of bandlimited signals. The theorem shows that for a signal $x(t)$ bandlimited to a frequency $f \leq f_0$ and sampled at the uniform points $t_n = nT$, $(n = 0, \pm 1, \pm 2, \ldots)$, and T is the sampling period given by

$$T = \frac{1}{2f_0} \tag{2.1}$$

then $x(t)$ is completely determined by its uniform samples $x(t_n)$. The reconstruction of the signal is

$$x(t) = \sum_{n=-\infty}^{\infty} x(t_n) S(t, t_n) \tag{2.2}$$

where $S(t, t_n)$ is the composing (or sampling) function given by

$$S(t, t_n) = \frac{\sin[2\pi f_0 (t - t_n)]}{2\pi f_0 (t - t_n)} \tag{2.3}$$

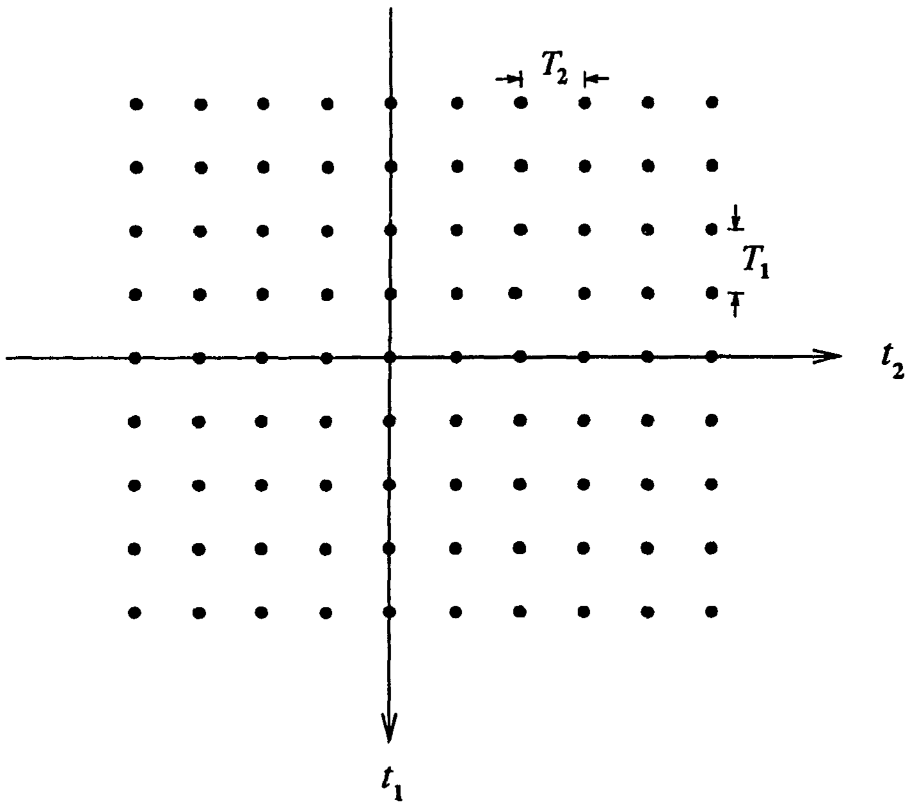


Figure 2.1: Sampling locations in the (t_1, t_2)-plane for rectangular sampling

2.2.2 Image Sampling

A straightforward generalization of the one-dimensional sampling [21] to the two-dimensional case is uniform sampling in rectangular coordinates, which is simply called *rectangular sampling* [2]. Image sampling on a rectangular grid can be achieved in a separable fashion, i.e., by first sampling each row, then each column.

Let $x_a(t_1, t_2)$ denotes a continuous image field. The discrete signal $x(n_1, n_2)$ obtained from a rectangular sampling of the continuous image is given by:

$$x(n_1, n_2) = x_a(n_1T_1, n_2T_2) \tag{2.4}$$

where T_1 and T_2 are the vertical and horizontal sampling intervals. The sample locations in the continuous (t_1, t_2)-plane are shown in Figure 2.1. In order to find a relation between the Fourier transforms of the continuous and discrete signals, we start with the two-dimensional Fourier transform relations for continuous signals

$$X_a(\Omega_1,\Omega_2) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} x_a(t_1,t_2)\exp(-j\Omega_1 t_1 - j\Omega_2 t_2)dt_1 dt_2 \tag{2.5}$$

$$x_a(t_1,t_2) = \frac{1}{4\pi^2}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} X_a(\Omega_1,\Omega_2)\exp(j\Omega_1 t_1 + j\Omega_2 t_2)d\Omega_1 d\Omega_2 \tag{2.6}$$

Where Ω_1 and Ω_2 are the frequency axis of the continuous signal in the vertical and horizontal directions respectively. Combining Equations 2.4 and 2.6, we can write

$$x(n_1,n_2) = \frac{1}{4\pi^2}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} X_a(\Omega_1,\Omega_2)\exp(j\Omega_1 n_1 T_1 + j\Omega_2 n_2 T_2)d\Omega_1 d\Omega_2 \tag{2.7}$$

The frequency plane (ω_1,ω_2) of the discrete signal is related to that of the continuous signal by

$$\omega_1 = \Omega_1 T_1 \qquad \text{and} \qquad \omega_2 = \Omega_2 T_2 \tag{2.8}$$

and from Equations 2.6 and 2.8,

$$x(n_1,n_2) = \frac{1}{4\pi^2}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} \frac{1}{T_1 T_2} X_a\left(\frac{\omega_1}{T_1},\frac{\omega_2}{T_2}\right)\exp(j\omega_1 n_1 + j\omega_2 n_2)d\omega_1 d\omega_2 \tag{2.9}$$

The double integral over the entire (ω_1,ω_2)-plane can be broken into an infinite series of integrals, each one is over a square of area $4\pi^2$. If $\mathcal{A}(k_1,k_2)$ represents the square $\{-\pi+2\pi k_1 \le \omega_1 < \pi+2\pi k_1; -\pi+2\pi k_2 \le \omega_2 < \pi+2\pi k_2\}$, then Equation 2.9 can be written as

$$x(n_1,n_2) = \frac{1}{4\pi^2}\sum_{k_1}\sum_{k_2}\iint_{\mathcal{A}} \frac{1}{T_1 T_2} X_a\left(\frac{\omega_1}{T_1},\frac{\omega_2}{T_2}\right)\exp(j\omega_1 n_1 + j\omega_2 n_2)d\omega_1 d\omega_2 \tag{2.10}$$

The dependence of the limits of integration on k_1 and k_2 can be removed by Replacing ω_1 by $\omega_1 - 2\pi k_1$ and ω_2 by $\omega_2 - 2\pi k_2$. This yields

$$\begin{aligned} x(n_1,n_2) &= \frac{1}{4\pi^2}\int_{-\pi}^{\pi}\int_{-\pi}^{\pi}\left[\frac{1}{T_1 T_2}\sum_{k_1}\sum_{k_2} X_a\left(\frac{\omega_1}{T_1}-\frac{2\pi k_1}{T_1},\frac{\omega_2}{T_2}-\frac{2\pi k_2}{T_2}\right)\right] \\ &\cdot\exp(j\omega_1 n_1 + j\omega_2 n_2)\exp(-j2\pi k_1 n_1 - j2\pi k_2 n_2)d\omega_1 d\omega_2 \end{aligned} \tag{2.11}$$

The term $\exp(-j2\pi k_1 n_1 - j2\pi k_2 n_2)$ is equal to one for all the integer variables n_1, k_1, n_2 and k_2. Comparing the form of Equation 2.11 to that of Equation 2.6 it can be seen that the former is the inverse Fourier transform of the sampled signal. Thus we conclude that the Fourier transform of the sampled signal is

$$X(\omega_1,\omega_2) = \frac{1}{T_1T_2}\sum_{k_1}\sum_{k_2} X_a\left(\frac{\omega_1}{T_1} - \frac{2\pi k_1}{T_1}, \frac{\omega_2}{T_2} - \frac{2\pi k_2}{T_2}\right) \tag{2.12}$$

Alternatively, we can write

$$X(\Omega_1 T_1,\Omega_2 T_2) = \frac{1}{T_1T_2}\sum_{k_1}\sum_{k_2} X_a\left(\Omega_1 - \frac{2\pi k_1}{T_1}, \Omega_2 - \frac{2\pi k_2}{T_2}\right) \tag{2.13}$$

Equation 2.13 gives the relation between the spectrum of the sampled signal and that of the continuous signal. It can be seen that the spectrum of the sampled signal consists of the spectrum of the continuous signal repeated over a frequency grid of resolution $\frac{2\pi}{T_1}$ and $\frac{2\pi}{T_2}$ in the Ω_1 and Ω_2 directions. The terms $\frac{2\pi}{T_1}$ and $\frac{2\pi}{T_2}$ represent the sampling frequencies. If the signal $x_a(t_1, t_2)$ is a bandlimited signal, that is, if the Fourier transform $X_a(\Omega_1, \Omega_2)$ is confined to a rectangular region of finite extent in the (Ω_1, Ω_2)-plane such that

$$X_a(\Omega_1,\Omega_2) = 0 \qquad \text{for } |\Omega_1| \geq \frac{\pi}{T_1}, \quad |\Omega_2| \geq \frac{\pi}{T_2} \tag{2.14}$$

There will be no spectrum overlap in the sampled image. This condition is equivalent to the one-dimensional sampling theorem discussed in Section 2.2.1.

Figure 2.2 shows the spectrum of a continuous two-dimensional signal (for simplicity, a symmetric diamond-shaped spectrum is used). The signal is sampled at four times the highest frequency components (twice the Nyquist rate) in the horizontal and vertical directions. The spectrum of the sampled signal is shown in Figure 2.3. Note that for the sampling rates used, The highest frequency components in the ω_1 and ω_2 directions are each equal to $\pi/2$. Thus, the spectrum can be enclosed in a square of area π^2. It will be shown in Section 2.2.5 that this sampled (discrete) signal can be subsampled by a factor of 2 in both horizontal and vertical directions and still represents the original continuous signal.

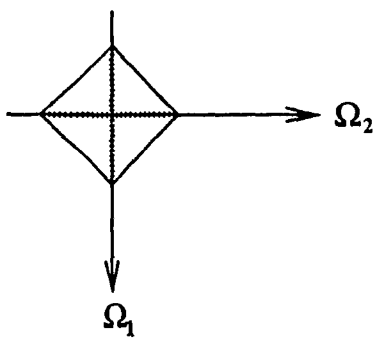


Figure 2.2: Original spectrum of a two-dimensional continuous signal (top view).

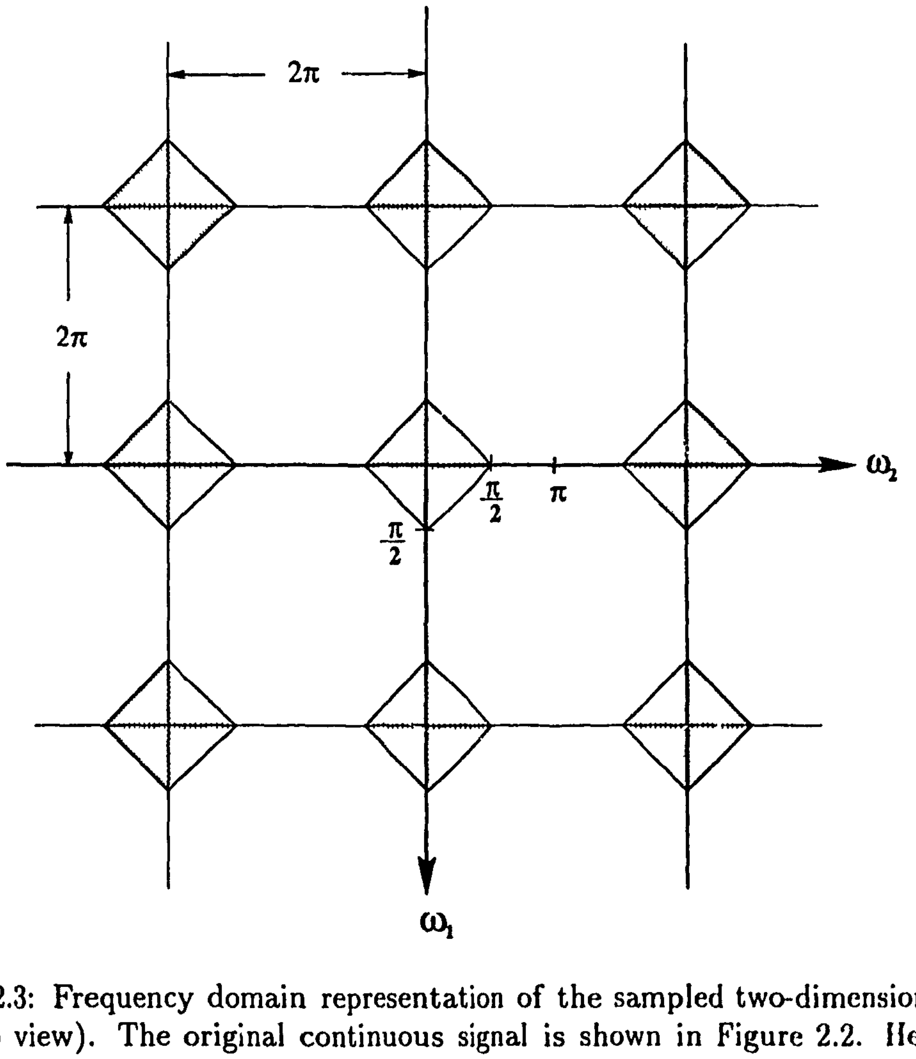


Figure 2.3: Frequency domain representation of the sampled two-dimensional signal (top view). The original continuous signal is shown in Figure 2.2. Here, the continuous signal is sampled at twice the Nyquist rate.

2.2.3 Image Reconstruction

If Equation 2.14 is satisfied, then $X_a(\Omega_1, \Omega_2)$ can be recovered from $X(\Omega_1 T_1, \Omega_2 T_2)$ using Equation 2.13 to get

$$X_a(\Omega_1,\Omega_2) = \begin{cases} T_1 T_2 X(\Omega_1 T_1. \Omega_2 T_2) & |\Omega_1| < \frac{\pi}{T_1}, \quad |\Omega_2| < \frac{\pi}{T_2} \\ 0 & \text{otherwise} \end{cases} \tag{2.15}$$

Co, ,equently, it is possible to reconstruct the continuous signal $x_a(t_1, t_2)$ from the sampled signal by using a rectangular lowpass filter $H(\Omega_1, \Omega_2)$ given by

$$H(\Omega_1,\Omega_2) = \begin{cases} T_1 T_2 & |\Omega_1| \le \frac{\pi}{T_1}, \quad |\Omega_2| \le \frac{\pi}{T_2} \\ 0 & \text{otherwise} \end{cases} \tag{2.16}$$

To derive the reconstruction formula in the (t_1, t_2)-domain, we express $x_a(t_1, t_2)$ in terms of its inverse Fourier transform

$$\begin{aligned} x_a(t_1,t_2) &= \frac{1}{4\pi^2}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} X_a(\Omega_1,\Omega_2)\exp(j\Omega_1 t_1 + j\Omega_2 t_2)d\Omega_1 d\Omega_2 \\ &= \frac{1}{4\pi^2}\int_{-\frac{\pi}{T_1}}^{\frac{\pi}{T_1}}\int_{-\frac{\pi}{T_2}}^{\frac{\pi}{T_2}} T_1 T_2 X(\Omega_1 T_1, \Omega_2 T_2) \\ &\qquad \cdot \exp(j\Omega_1 t_1 + j\Omega_2 t_2)d\Omega_1 d\Omega_2 \end{aligned} \tag{2.17}$$

Expressing $X(\Omega_1 T_1, \Omega_2 T_2)$ in terms of $x(n_1, n_2)$ we can write

$$\begin{aligned} x_a(t_1,t_2) &= \frac{1}{4\pi^2}\int_{-\frac{\pi}{T_1}}^{\frac{\pi}{T_1}}\int_{-\frac{\pi}{T_2}}^{\frac{\pi}{T_2}} T_1 T_2 \left[\sum_{n_1}\sum_{n_2} x(n_1,n_2)\exp(-j\Omega_1 T_1 n_1 - j\Omega_2 T_2 n_2)\right] \\ &\qquad \cdot \exp(j\Omega_1 t_1 + j\Omega_2 t_2)d\Omega_1 d\Omega_2 \end{aligned} \tag{2.18}$$

By interchanging the integrations and the summations, Equation 2.18 can be written as

$$\begin{aligned} x_a(t_1,t_2) &= \frac{T_1 T_2}{4\pi^2}\sum_{n_1}\sum_{n_2} x(n_1,n_2)\int_{-\frac{\pi}{T_1}}^{\frac{\pi}{T_1}}\int_{-\frac{\pi}{T_2}}^{\frac{\pi}{T_2}} \exp[j\Omega_1(t_1 - n_1 T_1) \\ &\qquad + j\Omega_2(t_2 - n_2 T_2)]d\Omega_1 d\Omega_2 \end{aligned} \tag{2.19}$$

Therefore,

$$x_a(t_1,t_2) = \sum_{n_1}\sum_{n_2} x_a(n_1 T_1, n_2 T_2)\frac{\sin[\pi(t_1 - n_1 T_1)/T_1]}{\pi(t_1 - n_1 T_1)/T_1}\frac{\sin[\pi(t_2 - n_2 T_2)/T_2]}{\pi(t_2 - n_2 T_2)/T_2} \tag{2.20}$$

Equation 2.20 gives the reconstruction of the continuous signal $x_a(t_1, t_2)$ from its sample values using the two-dimensional *sinc*[1] function. This function falls off rapidly from its peak at the origin. This means that the most significant terms are those for which (n_1T_1, n_2T_2) are in close proximity of (t_1, t_2). Equations 2.4, 2.14, 2.15 and 2.20 form the basis of the two-dimensional sampling theorem which states that a bandlimited continuous signal may be recovered completely from its sample values if the condition in Equation 2.14 is satisfied.

2.2.4 Aliasing Errors and Moiré-Effect

Aliasing errors or spectral foldover occur when a signal is sampled at a rate less than the Nyquist rate. Figure 2.4 shows aliasing in one-dimension where a sinusoidal signal of frequency 15 Hz. is sampled at $T = \frac{1}{f_s} = 0.05$ seconds. It can be seen that there exists another sinusoidal signal with the same set of samples as the original but with a lower frequency. This ambiguity is known as *aliasing*. In the frequency domain, the violation of the sampling theorem results in a spectral foldover. Any frequency above $\frac{f_s}{2}$ (folding frequency) is folded and becomes indistinguishable from its mirror in the foldover region. This is shown in Figure 2.5 where the 15 Hz. component is folded to its 5Hz. mirror spectrum.

If an image is undersampled, spectral overlap (foldover) occurs in two-dimensions. This introduces artificial low spatial frequency components in reconstruction [22]. The artifacts generated by this two-dimensional aliasing are sometimes called *Moiré* patterns. Undersampling also results in a loss of some high spatial frequency components of the image, causing a loss of resolution. Figure 2.6 shows an undersampled two-dimensional function with sampling frequencies $\frac{2\pi}{T_1}$ and $\frac{2\pi}{T_2}$ in the Ω_1 and Ω_2 directions respectively. It can be seen that artificial low spatial frequencies have been introduced (shaded area).

[1]A *sinc* function is defined as $sinc(\nu) = \frac{sin(\nu)}{\nu}$

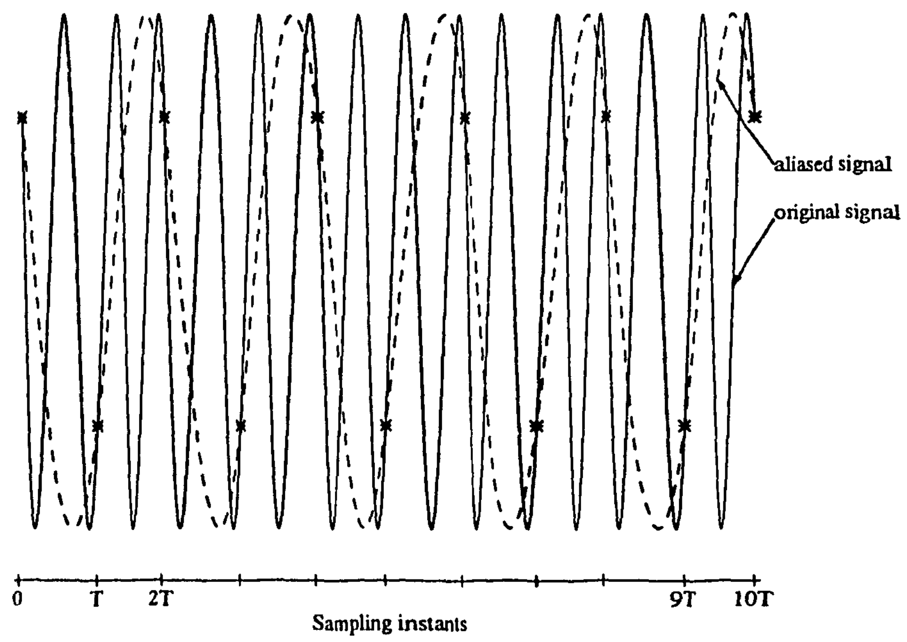


Figure 2.4: Aliasing effects resulting from undersampling the original signal.

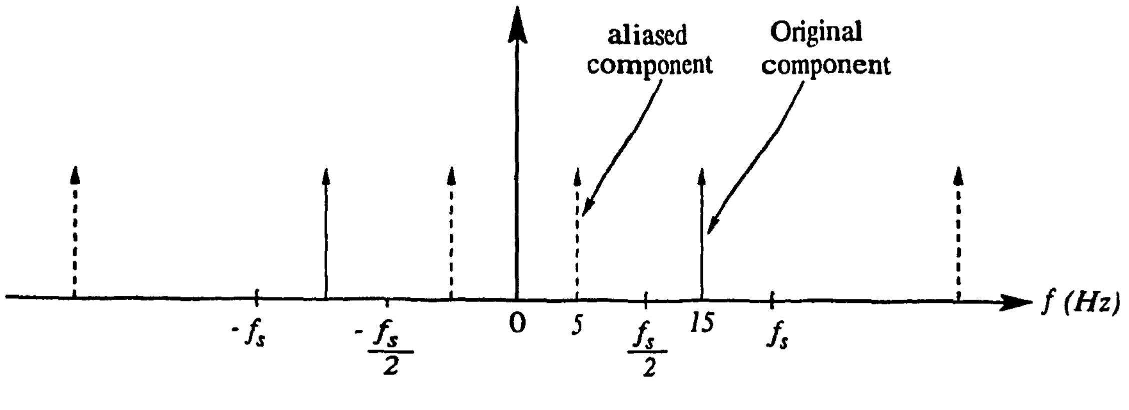


Figure 2.5: Spectral foldover resulting from undersampling the sinusoidal signal

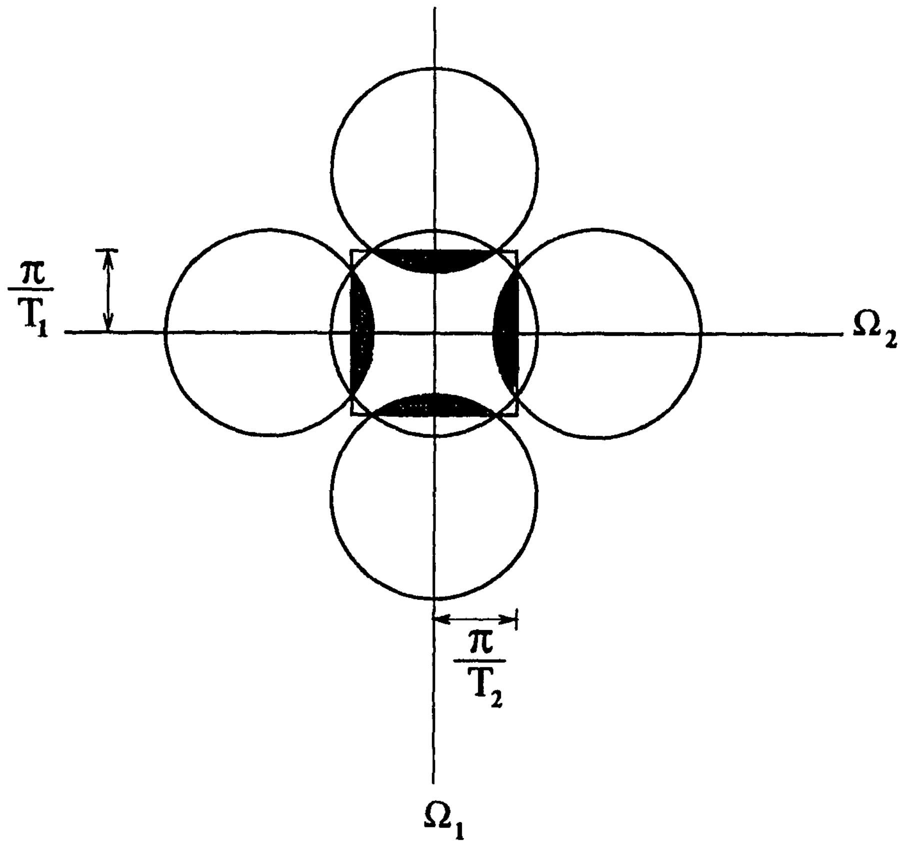


Figure 2.6: Spectral overlap for an undersampled two-dimensional function (top view).

2.2.5 Decimation by Integer Factors (M_1, M_2):Subsampling

In Sections 2.2.2 and 2.2.3, the sampling and reconstruction of two-dimensional continuous signals has been considered. This section deals with the sampling of two-dimensional discrete signals (i e., subsampling) on a rectangular grid. An excellent textbook that covers the one-dimensional decimation case is given in [23].

Consider the process of reducing the sampling rate (known as subsampling, downsampling or decimation) of $x(n_1, n_2)$ by the integer factors (M_1, M_2). The new sequence can be written as

$$x_{ss}(n_1, n_2) = x(n_1 M_1, n_2 M_2) = x_a(n_1 T_1 M_1, n_2 T_2 M_2) \tag{2.21}$$

It can be seen that the subsampled signal $x_{ss}(n_1, n_2)$ can be obtained directly from the $x_a(t_1, t_2)$ by sampling with intervals $T_1{}' = T_1 M_1$ and $T_2{}' = T_2 M_2$. The relation between the Fourier transform of the subsampled signal and that of the continuous signal can be directly derived from Equation 2.12 by replacing T_1 by $T_1{}' = T_1 M_1$, and T_2 by $T_2{}' = T_2 M_2$.

$$X_{ss}(\omega_1, \omega_2) = \frac{1}{T_1 T_2 M_1 M_2} \sum_{r_1} \sum_{r_2} X_a \left(\frac{\omega_1 - 2\pi r_1}{T_1 M_1}, \frac{\omega_2 - 2\pi r_2}{T_2 M_2} \right) \tag{2.22}$$

The summation indices r_1, and r_2 can be expressed as

$$r_1 = m_1 + k_1 M_1 \qquad \text{and} \qquad r_2 = m_2 + k_2 M_2 \tag{2.23}$$

Where m_1 and m_2 are integers such that $0 \leq m_1 \leq M_1 - 1$ and $0 \leq m_2 \leq M_2 - 1$. Now Equation 2.22 can be expressed as

$$\begin{aligned} X_{ss}(\omega_1, \omega_2) &= \frac{1}{M_1 M_2} \sum_{m_1=0}^{M_1-1} \sum_{m_2=0}^{M_2-1} \left[\frac{1}{T_1 T_2} \sum_{k_1} \sum_{k_2} X_a \left(\frac{\omega_1}{T_1 M_1} - \frac{2\pi k_1}{T_1} - \frac{2\pi m_1}{T_1 M_1}, \right. \right. \\ &\qquad \left. \left. \frac{\omega_2}{T_2 M_2} - \frac{2\pi k_2}{T_2} - \frac{2\pi m_2}{T_2 M_2} \right) \right] \end{aligned} \tag{2.24}$$

The term inside the brackets in Equation 2.24 can be recognized from Equation 2.12 as

$$X\left(\frac{\omega_1 - 2\pi m_1}{M_1}, \frac{\omega_2 - 2\pi m_2}{M_2}\right) = \frac{1}{T_1T_2}\sum_{k_1}\sum_{k_2} X_a\left(\frac{\omega_1 - 2\pi m_1}{T_1M_1} - \frac{2\pi k_1}{T_1},\right.$$
$$\left.\frac{\omega_2 - 2\pi m_2}{T_2} - \frac{2\pi k_2}{T_2}\right) \quad (2.25)$$

Therefore, Equation 2.24 can be expressed as

$$X_{ss}(\omega_1,\omega_2) = \frac{1}{M_1M_2}\sum_{m_1=0}^{M_1-1}\sum_{m_2=0}^{M_2-1} X\left(\frac{\omega_1 - 2\pi m_1}{M_1}, \frac{\omega_2 - 2\pi m_2}{M_2}\right) \quad (2.26)$$

Equation 2.26 expresses the Fourier transform of the subsampled signal $x_{ss}(n_1,n_2)$ (sampling intervals (M_1, M_2)) in terms of the Fourier transform of the discrete signal $x(n_1,n_2)$. Thus $X_{ss}(\omega_1,\omega_2)$ can be thought of as being a frequency scaled version (scaling factors (M_1, M_2)) of the periodic Fourier transform $X(\omega_1,\omega_2)$, shifted by integer multiples of $\frac{2\pi}{M_1}$ and $\frac{2\pi}{M_2}$ in the ω_1 and ω_2 directions respectively.

Aliasing in subsampling can be avoided by ensuring that $X(\omega_1,\omega_2)$ is bandlimited, i.e.,

$$X(\omega_1,\omega_2) = 0 \qquad \text{for } \frac{\pi}{M_1} \le |\omega_1| \le \pi, \quad \frac{\pi}{M_2} \le |\omega_2| \le \pi \quad (2.27)$$

Equation 2.27 can be realized by using a rectangular *digital* low pass filter that approximates the ideal characteristics. The frequency response of the filter is given by

$$H(\omega_1,\omega_2) = \begin{cases} 1 & |\omega_1| \le \frac{\pi}{M_1}, \ |\omega_2| \le \frac{\pi}{M_2} \\ 0 & \text{otherwise} \end{cases} \quad (2.28)$$

Note that if the original sampling rates used to produce the discrete signal $x(n_1,n_2)$ from $x_a(t_1,t_2)$ was at least (M_1, M_2) times twice the highest frequency components of $x_a(t_1,t_2)$, then subsampling $x(n_1,n_2)$ by factors (M_1, M_2) results in a signal that still represents the original continuous signal $x_a(t_1,t_2)$. In practice however, $x(n_1,n_2)$ may be a full band signal, that is, its spectrum may be nonzero for all frequencies in the ranges $(-\pi < \omega_1 < \pi; -\pi < \omega_2 < \pi)$. Thus, in this case the bandwidth of $x(n_1, n_2)$ must be first reduced by factors of (M_1, M_2) before subsampling. The subsampled signal $x_{ss}(n_1, n_2)$ will no longer represent the original continuous signal $x_a(t_1, t_2)$.

One advantage of subsampling on a rectangular grid is that all operations can be done in a separable fashion. Thus, the rectangular filter $H(\omega_1, \omega_2)$ in Equation 2.28 can be implemented as two lowpass filters in cascade $H_1(\omega_1)$ and $H_2(\omega_2)$. The filter $H_1(\omega_1)$ is given by

$$H_1(\omega_1) = \begin{cases} 1 & |\omega_1| \leq \frac{\pi}{M_1} \\ 0 & \text{otherwise} \end{cases} \tag{2.29}$$

Likewise, $H_2(\omega_2)$ is given by

$$H_2(\omega_2) = \begin{cases} 1 & |\omega_2| \leq \frac{\pi}{M_2} \\ 0 & \text{otherwise} \end{cases} \tag{2.30}$$

To illustrate the subsampling process, the two-dimensional discrete signal $x(n_1, n_2)$ whose spectrum is shown in Figure 2.3 is subsampled with factors $M_1 = M_2 = 2$ and $M_1 = M_2 = 3$. Figure 2.7 shows the spectrum of the subsampled signal $x_{ss}(n_1, n_2)$ when using the factors $M_1 = M_2 = 2$. As the spectrum of $x(n_1, n_2)$ is zero in the ranges $(\frac{\pi}{2} \leq |\omega_1| \leq \pi; \frac{\pi}{2} \leq |\omega_2| \leq \pi)$, it is possible to subsample $x(n_1, n_2)$ by the factors $M_1 = M_2 = 2$ and having a signal $x_{ss}(n_1, n_2)$ that completely describes $x(n_1, n_2)$. However, subsampling $x(n_1, n_2)$ with factors $M_1 = M_2 = 3$ requires bandlimiting the signal with a rectangular lowpass digital filter in order to avoid aliasing. This filter is specified by the frequency response (assuming ideal characteristics)

$$H(\omega_1, \omega_2) = \begin{cases} 1 & |\omega_1| \leq \frac{\pi}{3}, \ \ \omega_2| \leq \frac{\pi}{3} \\ 0 & \text{otherwise} \end{cases} \tag{2.31}$$

Figure 2.8 shows the spectrum of the lowpass filter. Note that the subsampled signal $\tilde{x}_{ss}(n_1, n_2)$ no longer represents the signal $x(n_1, n_2)$, but rather a signal $\tilde{x}(n_1, n_2)$ obtained by lowpass filtering $x(n_1, n_2)$ with the digital filter given in Equation 2.31.

Figure 2.9 shows the spectrum of the bandlimited signal $\tilde{x}(n_1, n_2)$. The spectrum of the subsampled signal $\tilde{x}_{ss}(n_1, n_2)$ is shown in Figure 2.10.

A block diagram of a separable image subsampling system is shown in Figure 2.11. The filter $H_2(\omega_2)$ operates on the rows. The output is then fed to the filter $H_1(\omega_1)$ which in turn operates on the columns. Next, the prefiltered image $\tilde{x}(n_1, n_2)$ is subsampled, first the rows by a factor of M_2, then the columns by a factor of M_1. Note that the subscripts 1 and 2 in $H_1(\omega_1)$, $H_2(\omega_2)$, M_1, and M_2 are only used to indicate the axis direction (and not the order) of the operations.

2.2.6 Interpolation by Integer factors (L_1, L_2): Upsampling

The process of increasing the sampling rate (upsampling or interpolation) of a discrete signal $x(n_1, n_2)$ involves operations analogous to that of digital-to-analog conversion. If the sampling rates are increased by integer factors (L_1, L_2), then the new sampling intervals can be written as

$$T_1' = \frac{T_1}{L_1} \quad \text{and} \quad T_2' = \frac{T_2}{L_2} \tag{2.32}$$

The new sequence $x_e(n_1, n_2)$ resulting from upsampling can be written in terms of $x(n_1, n_2)$ as

$$x_e(n_1, n_2) = \begin{cases} x(\frac{n_1}{L_1}, \frac{n_2}{L_2}), & n_1 = 0, \pm L_1, \pm 2L_1, \ldots \text{ and } n_2 = 0, \pm L_2, \pm 2L_2, \ldots \\ 0 & \text{otherwise} \end{cases} \tag{2.33}$$

Equation 2.33 can be expressed in the form

$$x_e(n_1, n_2) = \sum_{m_1=-\infty}^{\infty} \sum_{m_2=-\infty}^{\infty} x(m_1, m_2)\delta[n_1 - m_1 L_1, n_2 - m_2 L_2] \tag{2.34}$$

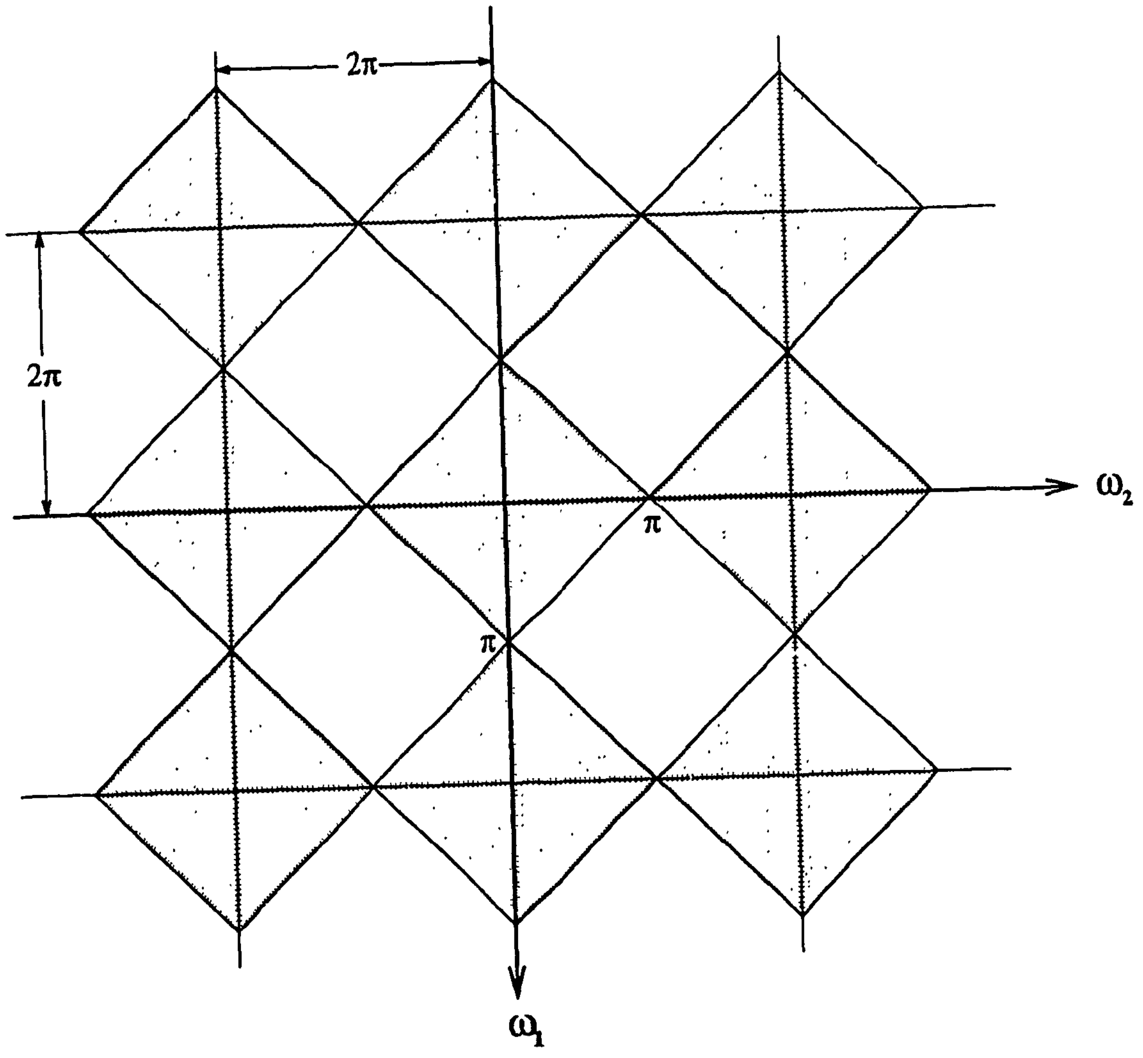


Figure 2.7: Spectrum of the subsampled signal $x_{ss}(n_1, n_2)$ when $M_1 = M_2 = 2$ (top view). The spectrum of the original discrete signal is shown in Figure 2.3.

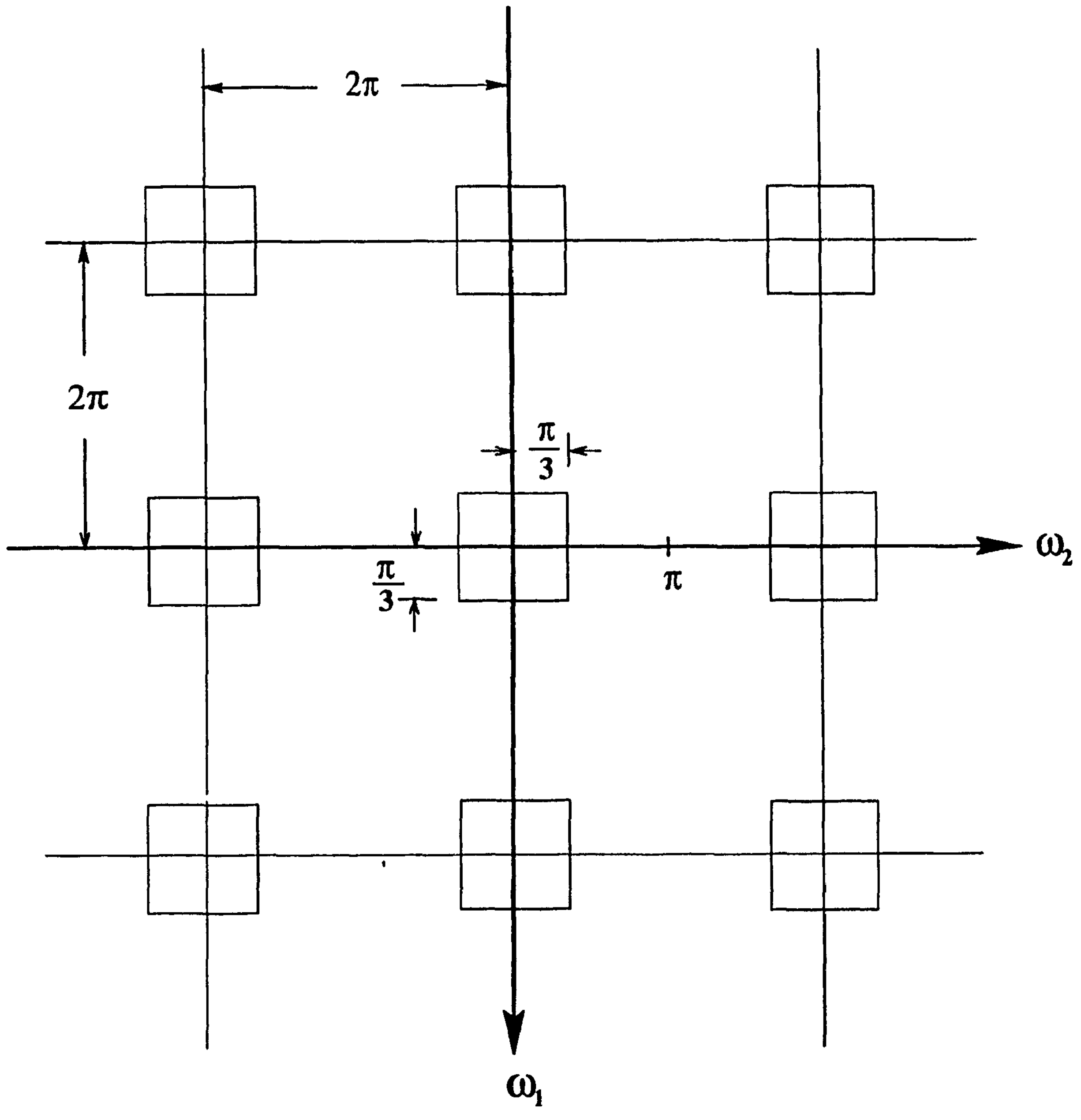


Figure 2.8: Spectrum of the rectangular digital anti-aliasing filter (top view) given by Equation 2.31 (assuming ideal characteristics). The filter is used to bandlimit the discrete signal $x(n_1, n_2)$ whose spectrum is shown in Figure 2.3 before subsampling by factors $M_1 = M_2 = 3$.

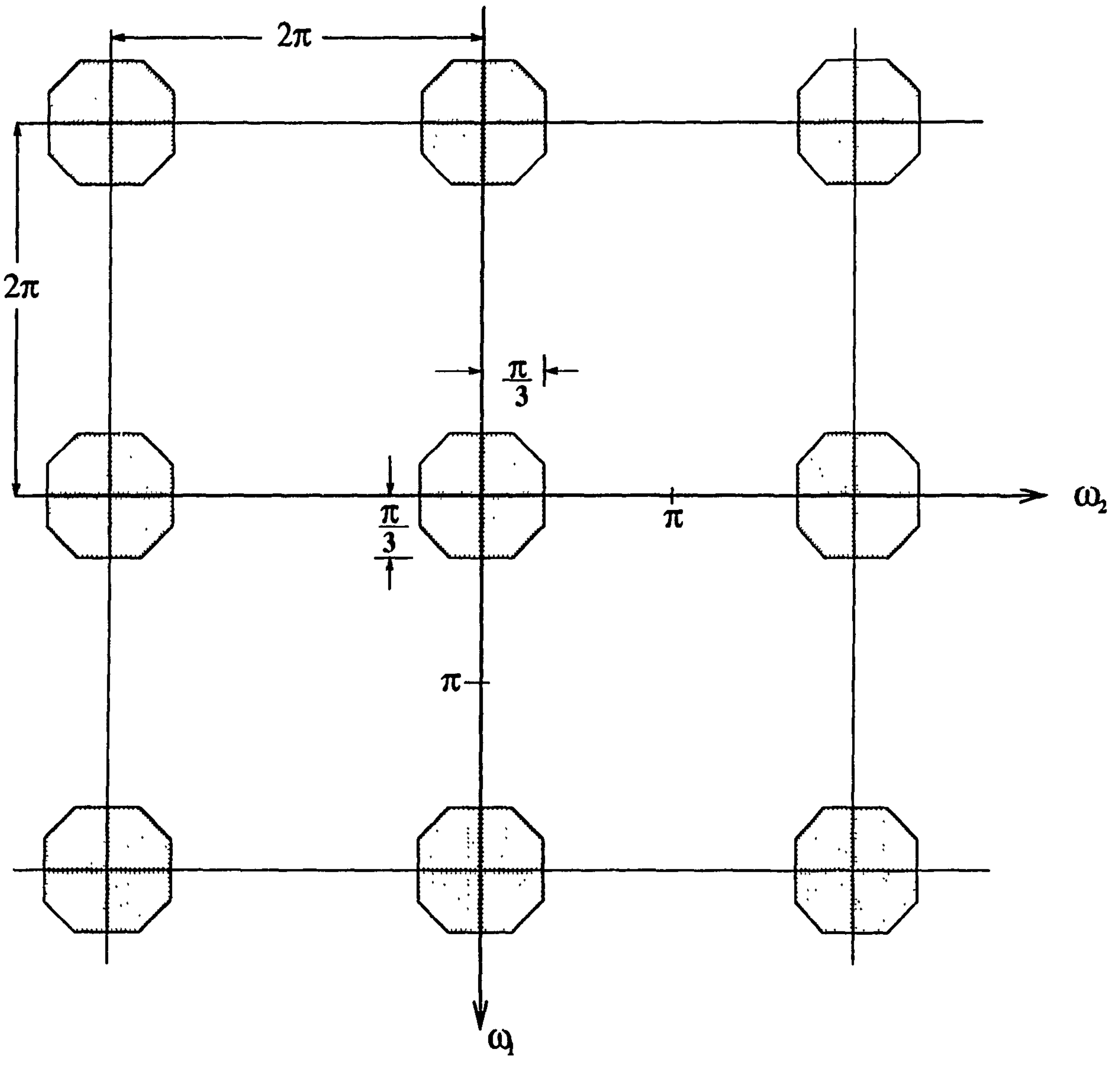


Figure 2.9: Spectrum of the bandlimited signal $\tilde{x}(n_1, n_2)$ after applying the digital filter shown in Figure 2.8 (top view). The spectrum of the original discrete signal is shown in Figure 2.3.

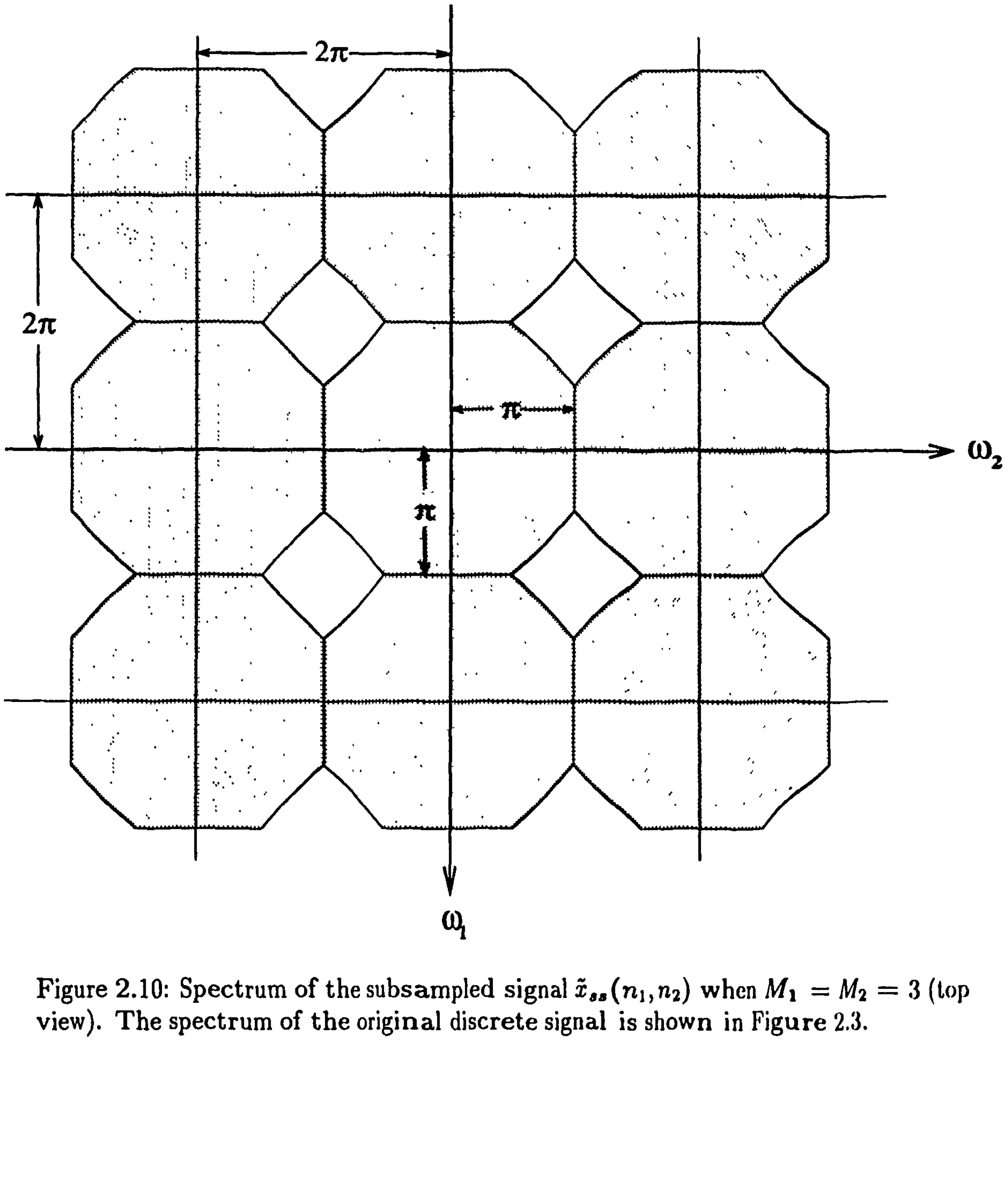


Figure 2.10: Spectrum of the subsampled signal $\tilde{x}_{ss}(n_1, n_2)$ when $M_1 = M_2 = 3$ (top view). The spectrum of the original discrete signal is shown in Figure 2.3.

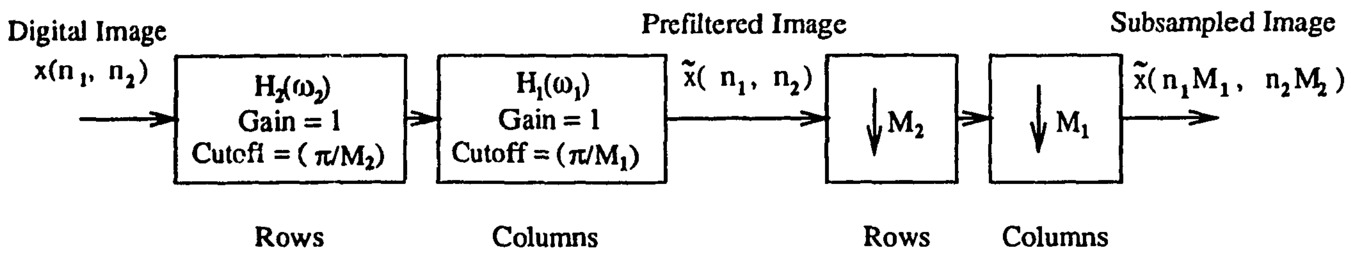


Figure 2.11: Image subsampling system (separable). The operations are done first on the rows, then on the columns.

The Fourier transform of $x_e(n_1,n_2)$ can then be written as

$$\begin{aligned} X_e(\omega_1,\omega_2) &= \sum_{n_1=-\infty}^{\infty}\sum_{n_2=-\infty}^{\infty}\left(\sum_{m_1=-\infty}^{\infty}\sum_{m_2=-\infty}^{\infty} x(m_1,m_2)\delta[n_1-m_1L_1,n_2-m_2L_2]\right) \\ &\qquad \cdot\exp(-j\omega_1 n_1 - j\omega_2 n_2) \\ &= \sum_{m_1=-\infty}^{\infty}\sum_{m_2=-\infty}^{\infty} x(m_1,m_2)\exp(-j\omega_1 L_1 m_1 - j\omega_2 L_2 m_2) \\ &= X(\omega_1 L_1,\omega_2 L_2) \end{aligned} \tag{2.35}$$

Therefore, the Fourier transform of $x_e(n_1,n_2)$ is a frequency scaled version of the Fourier transform of $x(n_1,n_2)$. To recover the baseband of interest, i.e., The spectrum within the rectangular region $(-\frac{\pi}{L_1} < \omega_1 < \frac{\pi}{L_1}; -\frac{\pi}{L_2} < \omega_2 < \frac{\pi}{L_2})$, it is necessary to filter the signal $x_e(n_1,n_2)$ with a digital lowpass (anti-imaging) filter which approximates the ideal characteristics. The frequency response of the filter is given by

$$H(\omega_1,\omega_2) = \begin{cases} L_1L_2 & |\omega_1| \le \frac{\pi}{L_1},\ \omega_2| \le \frac{\pi}{L_2} \\ 0 & \text{otherwise} \end{cases} \tag{2.36}$$

Analogous to the continuous case, The reconstruction equation that yields $\tilde{x}(n_1,n_2)$ can be expressed as

$$\tilde{x}(n_1,n_2) = \sum_{m_1}\sum_{m_2} x(m_1,m_2)\frac{\sin[\pi(n_1-m_1L_1)/L_1]}{\pi(n_1-m_1L_1)/L_1}\frac{\sin[\pi(n_2-m_2L_2)/L_2]}{\pi(n_2-m_2L_2)/L_2} \tag{2.37}$$

Unlike the continuous case (Equation 2.20) where all continuous values of the signal $x_a(t_1,t_2)$ are interpolated from the sequence $x(n_1,n_2)$; in the above equation only specific values need to be determined.

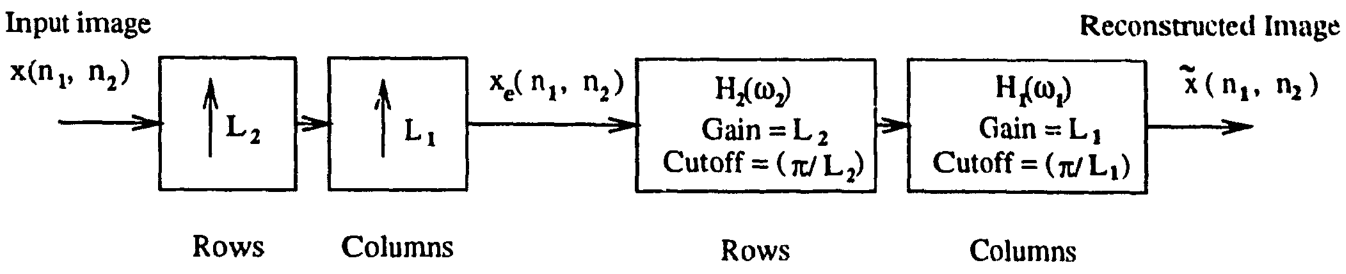


Figure 2.12: Image interpolation system (separable). The operations are done first on the rows, then on the columns.

Figure 2.12 shows a block diagram of a separable image interpolation system. The rows of the input image $x(n_1, n_2)$ are first stuffed with $L_2 - 1$ zero-valued samples between each pair of samples. The same procedure is then applied to the columns of the resulting signal by filling in $L_1 - 1$ zeros between each pair of samples. After increasing the sampling rate, the signal is passed through two filters in cascade. $H_2(\omega_2)$ operates on the rows and has a cutoff frequency of π/L_2 and a gain equal to L_2. The output of $H_2(\omega_2)$ is fed into $H_1(\omega_1)$ which in turn operates on the columns and has a cutoff frequency of π/L_1 and a gain equal to L_1.

2.2.7 Aliasing in Subsampling

If a digital image is to be subsampled by factors (M_1, M_2), then prefiltering is necessary prior to subsampling to bandlimit the image to $(\frac{\pi}{M_1}, \frac{\pi}{M_2})$ so that aliasing effects are eliminated. Figure 2.13 shows the original Cameraman test image. The reconstructed image using prefiltering prior to the subsampling operation is shown in Figure 2.14. Here $M_1 = M_2 = 2$. Figure 2.15 shows aliasing effects, in the reconstructed image, when no prefiltering is used.

To show how prefiltering affects the original signal, the intensity variation of line 200 of the original and the bandlimited Cameraman image for $M_1 = M_2 = 4$ is plotted in Figure 2.16. It can be seen that prefiltering removes excess picture details.

2.2.8 Uniform Subsampling of Images: An Example

This section illustrates practical applications of uniform subsampling for digital image compression. Uniform subsampling represents an image on a new sampling lattice with equally spaced samples and a lower sampling density than the original lattice. Figure 2.17 (a) shows the original lattice of an image. The new sampling lattices using uniform subsampling with factors $M_1 = M_2 = 2$, and $M_1 = M_2 = 4$ are shown in Figure 2.17 (b) and (c) respectively. The solid dots shows the pixels that are saved and transmitted.

An example using the Peppers test image illustrates uniform subsampling. The block diagram of the subsampling and interpolation systems used in this example are shown (previously) in Figures 2.11 and 2.12 respectively. Figure 2.18 (a) shows the original image of size 512×512 and 8 bits/pixels. Figures 2.18 (b), (c) and (d) show the subsampled images for the different decimating factors $M_1 = M_2 = 4, 8$ and 16 respectively. The FIR digital filters used to bandlimit the images prior to subsampling are denoted by h_a, h_b and h_c respectively. The coefficients of these filters are shown in Table 2.1. Here the length l of a filter is related to the decimating factor $M = M_1 = M_2$ by the equation $l = 1 + 2(M + 1)$. This has shown a better performance than using a fixed length for all filters. Figure 2.19 shows the magnitude response of the FIR filters given in Table 2.1. The cutoff frequency of h_a, h_b and h_c are chosen to be $\frac{\pi}{4}$, $\frac{\pi}{8}$ and $\frac{\pi}{16}$ respectively. Figures 2.20 (a), (b) and (c) show the reconstruction from the subsampled images of Figures 2.18 (b), (c) and (d) respectively. In reconstruction, the same filters used to bandlimit the signals are also used for interpolation. The multipliers 4, 8 and 16 are used with h_a, h_b and h_c respectively to adjust the gain of the interpolation filters.

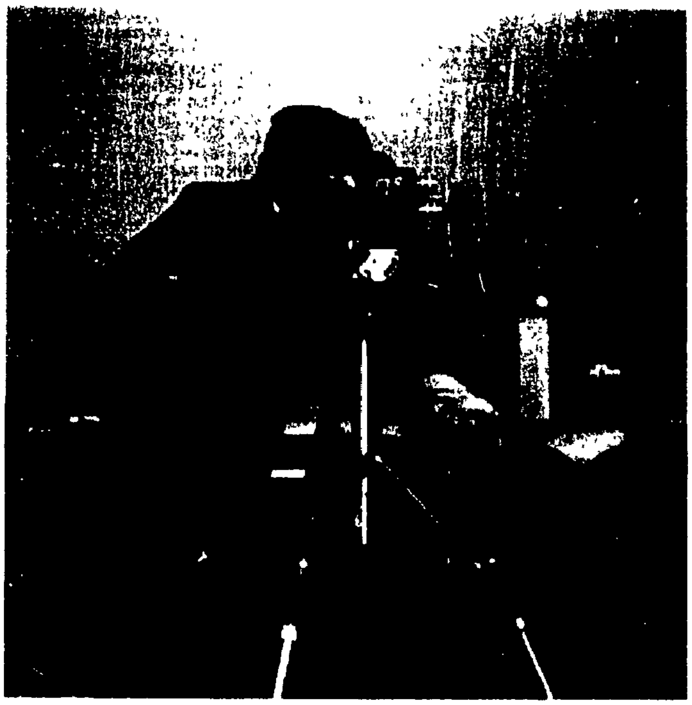


Figure 2.13: Original Cameraman image (256 × 256, 8 bit/pixel)

Table 2.1: Coefficients of the anti-aliasing FIR filters h_4, h_8 and h_{16} used for the decimating factors 4, 8 and 16 respectively.

h_4	0.0039	0.0000	0.0321	0.1167	0.2207	0.2687	0.2906
	0.1167	0.0321	0.0000	0.0039			
	0.0012	0.0000	0.0037	0.0130	0.0303	0.0552	0.0841
h_8	0.1123	0.1325	0.1398	0.1325	0.1123	0.0841	0.0552
	0.0303	0.0130	0.0037	0.0000	0.0012		
	0.0003	0.0000	0.0005	0.0015	0.0031	0.0057	0.0093
	0.0140	0.0198	0.0266	0.0341	0.0419	0.0496	0.0568
h_{16}	0.0630	0.0677	0.0707	0.0718	0.0707	0.0677	0.0630
	0.0568	0.0497	0.0419	0.0341	0.0266	0.0198	0.0140
	0.0093	0.0057	0.0031	0.0015	0.0005	0.0000	0.0003

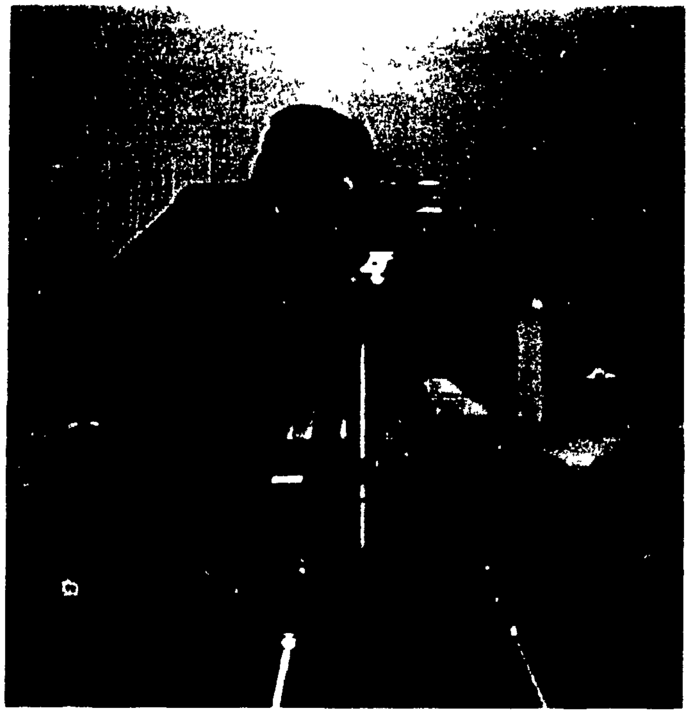


Figure 2.14. Reconstructed Cameraman (no aliasing effects)

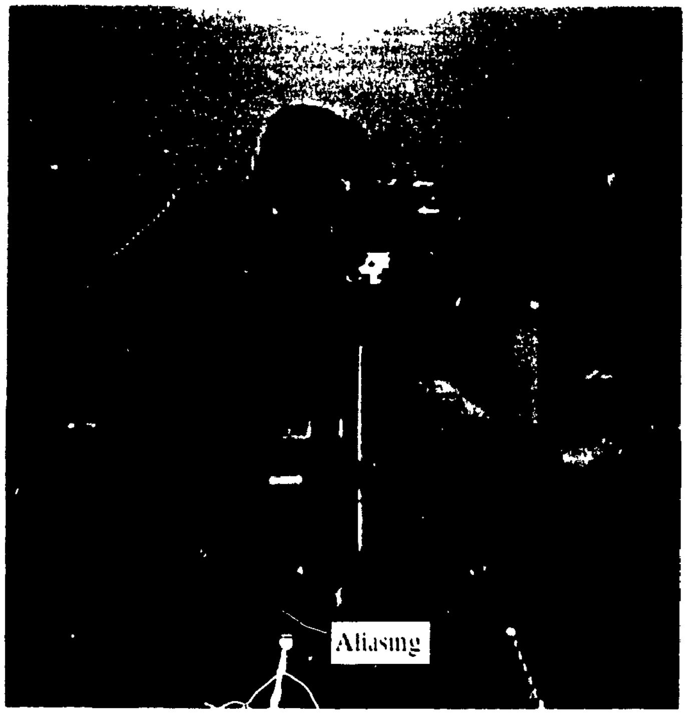


Figure 2.15. Reconstructed Cameraman. The arrows show aliasing effects which result when the image is not bandlimited prior to subsampling.

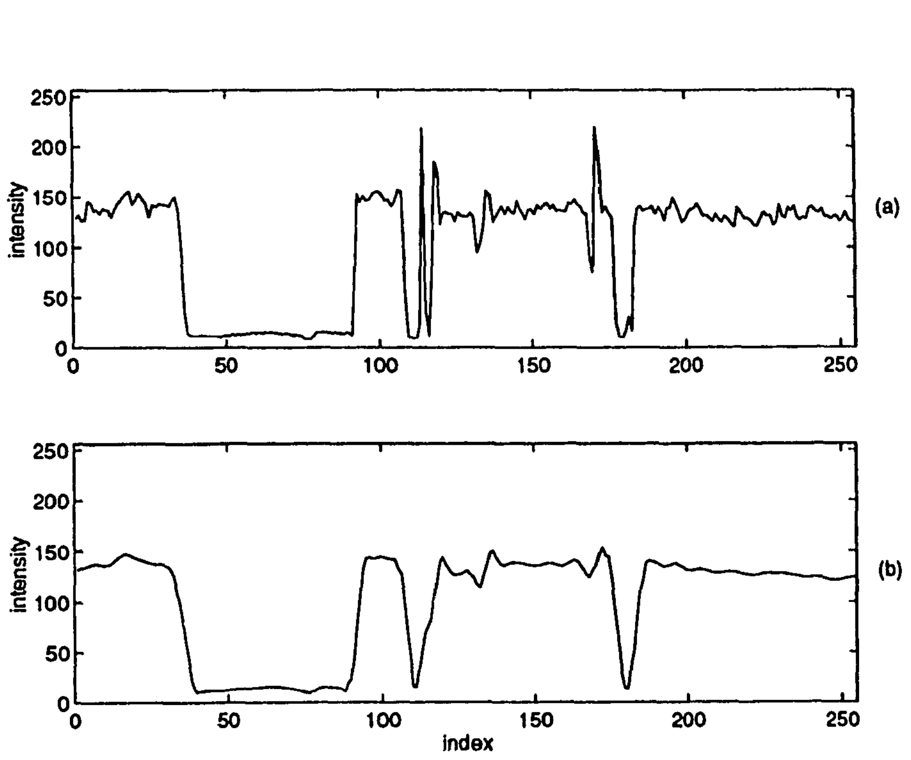


Figure 2.16: Prefiltering prior to subsampling is necessary to avoid aliasing. (a) line 200 of original Cameraman image (256 × 256). (b) line 200 of the prefiltered image ($M_1 = M_2 = 4$).

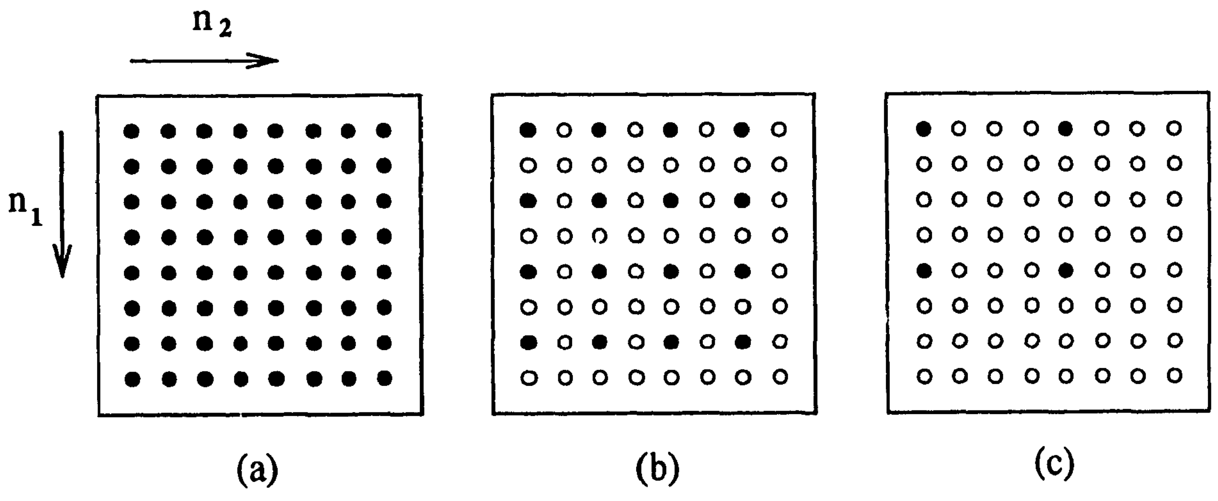


Figure 2.17: Uniform subsampling of an image. (a) Original lattice. (b) Subsampling by a factor of 2 in both horizontal and vertical directions ($M_1 = M_2 = 2$). (c) Subsampling by a factor of 4 in both horizontal and vertical directions ($M_1 = M_2 = 4$). The solid dots are the pixels that are saved and transmitted.

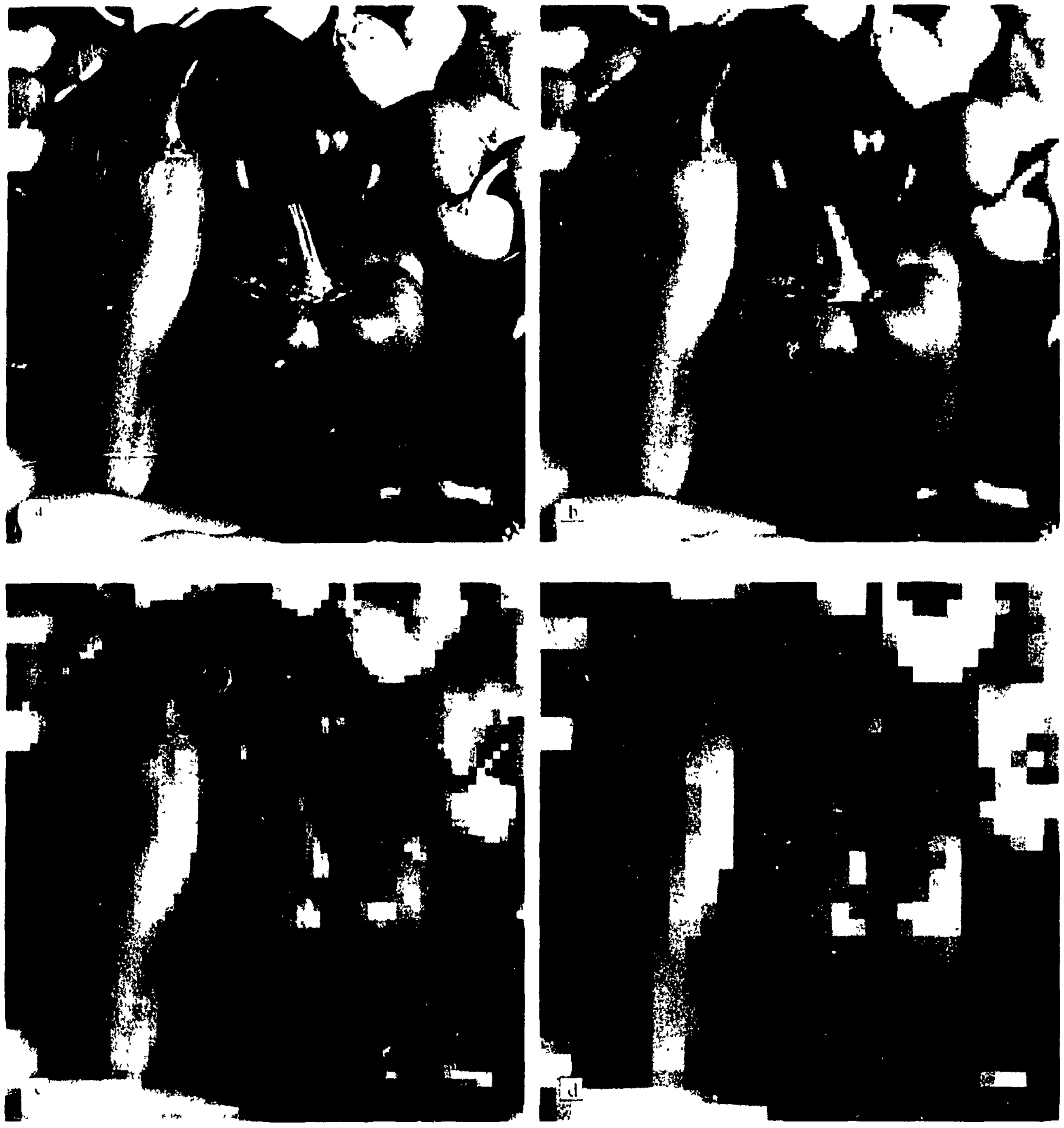


Figure 2.18: Uniform subsampling of the test image Peppers. a) Original Peppers (512 × 512, 8 bits/pixels). Subsampled Peppers: b) 128 × 128, 8 bits/pixels, c) 64 × 64, 8 bits/pixels, d) 32 × 32, 8 bits/pixels.

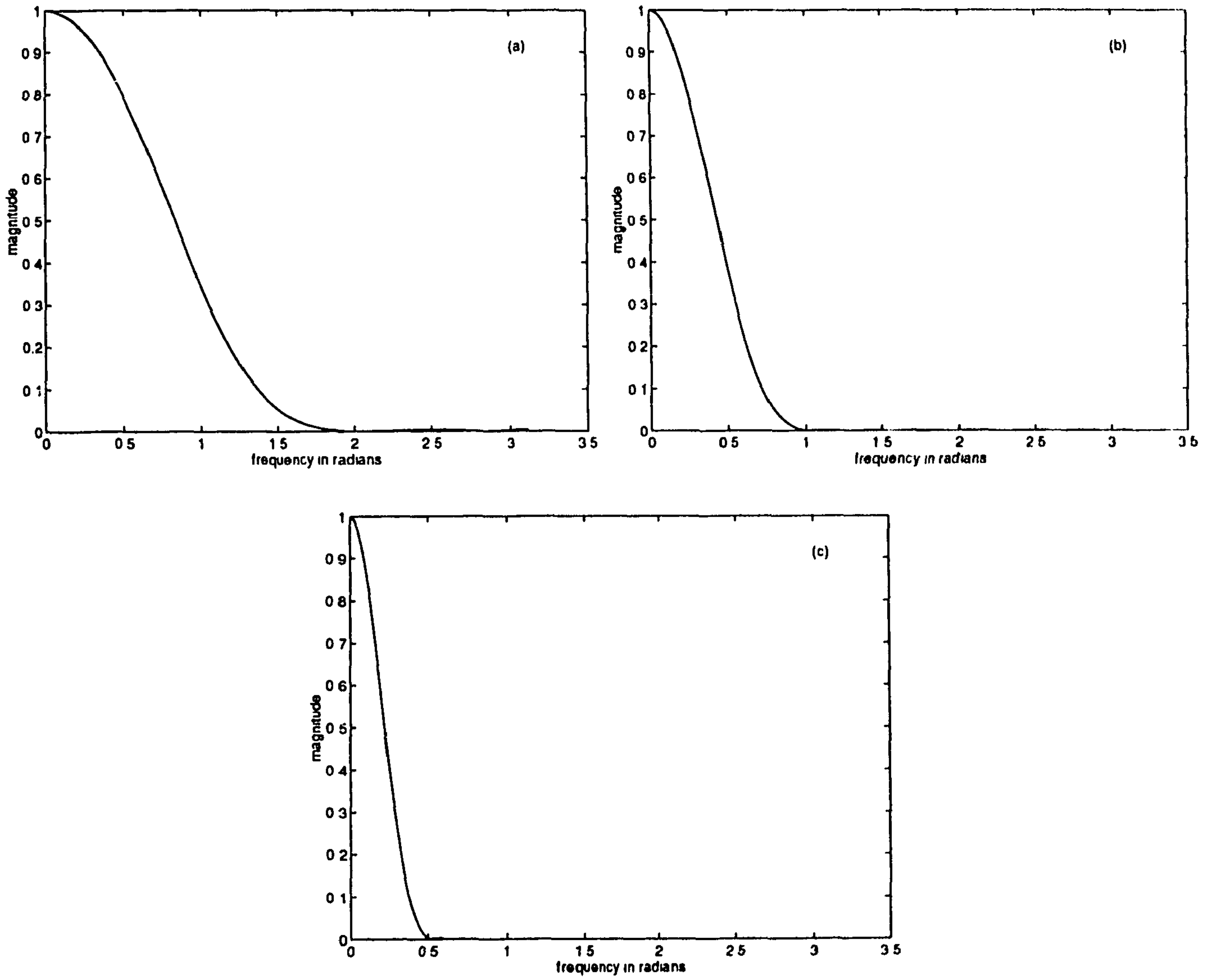


Figure 2.19: Magnitude response of the FIR anti-aliasing filters used in the subsampling process. a) Filter h_a: cutoff = $\frac{\pi}{4}$ ($l = 11$ and $M = 4$). b) Filter h_b: cutoff = $\frac{\pi}{8}$ ($l = 19$ and $M = 8$). c) Filter h_c: cutoff = $\frac{\pi}{16}$ ($l = 35$ and $M = 16$).

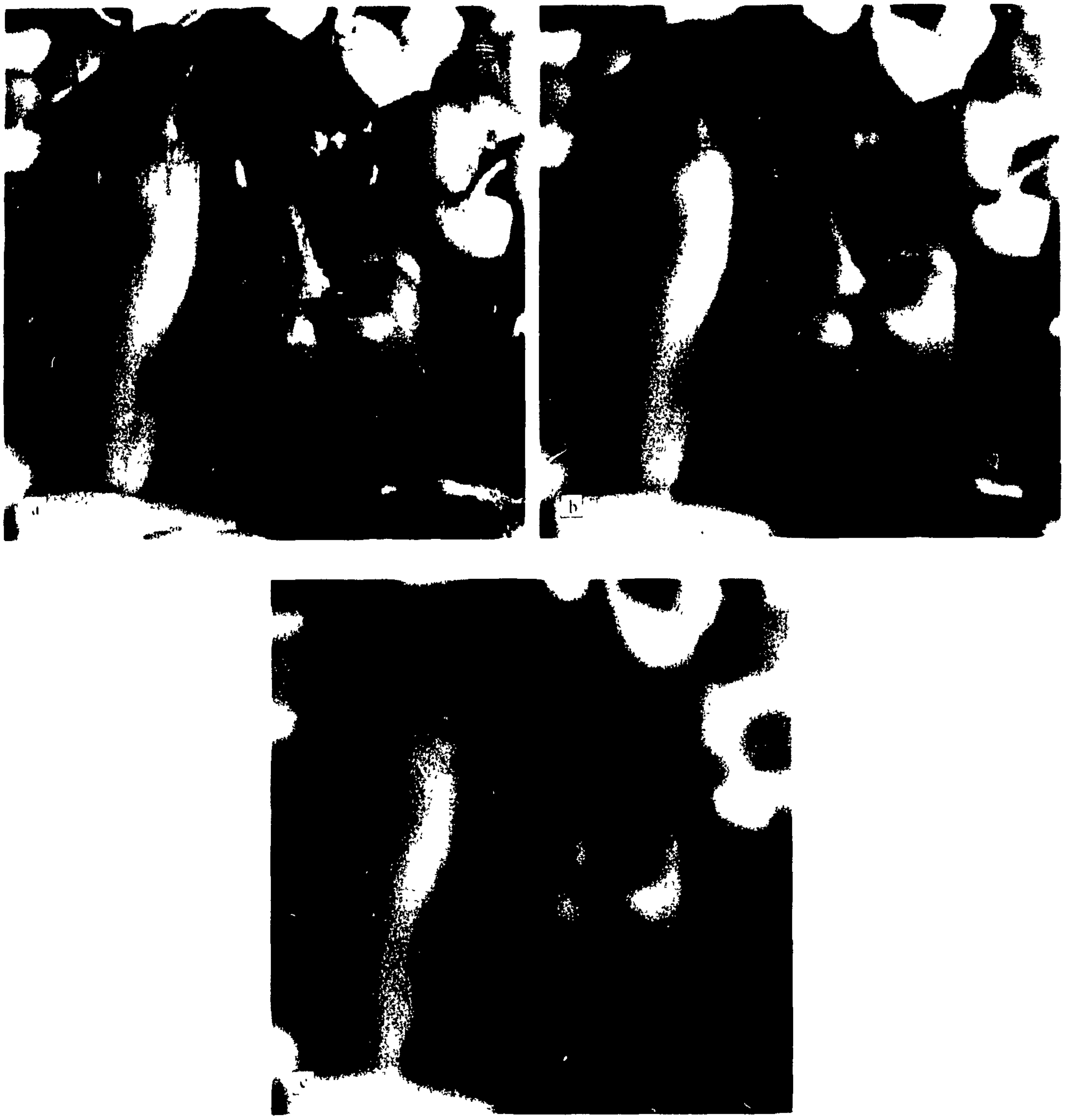


Figure 2.20 Reconstructed images from subsampling. (a) Peppers compressed to 0.5 bits/pixel. (b) Peppers compressed to 0.125 bits/pixel. (c) Peppers compressed to 0.0312 bits/pixel. Images in (a), (b), and (c) are reconstructed from the subsampled images of Figure 2.18 (b), (c), and (d) respectively.

2.3 Predictive Coding

Predictive coding [6, 24, 25] also known as differential pulse code modulation (DPCM) is a standard lossless compression method [8] that exploits the statistical redundancy of waveforms to realize straightforward reductions in bit rate. The basic idea in predictive coding is to generate a prediction $\hat{x}(n)$ of the input data $x(n)$, from N previous samples, i.e.

$$\hat{x}(n) = f\left(x(n-1), x(n-2), \ldots, x(n-N)\right) \tag{2.38}$$

If we restrict Equation 2.38 to linear prediction, then $\hat{x}(n)$ can be expressed as

$$\hat{x}(n) = \sum_{j=1}^{N} h_j x(n-j) \tag{2.39}$$

Where h_j are the prediction coefficients. Figure 2.21 shows a block diagram of a linear predictive coding scheme. Both transmitter and receiver use the same predictor type. The prediction error sequence $e(n) = x(n) - \hat{x}(n)$ is then quantized, entropy coded and transmitted. At the receiver, the quantized error $e_q(n)$ of the decoder output is added to the predicted value $\hat{x}(n)$ to reproduce $\tilde{x}(n)$ which, with error free transmission, differs from $x(n)$ only by the quantization error. The best linear predictor in a mean square sense, is the set of coefficients h_j which, on the average, minimizes $e^2(n)$.

A general equation for optimum linear predictor of order N is derived in Section 2.3.1. For analytical simplicity, prediction based on past unquantized samples rather than on quantized samples is assumed. An important application of linear prediction is the two-dimensional image predictor discussed in Section 2.3.2. Two special cases of interest, corresponding to image models with isotropic and separable autocorrelation functions, are discussed. An example illustrates the two-dimensional second and third order prediction of images and shows the prediction error for both isotropic and separable models.

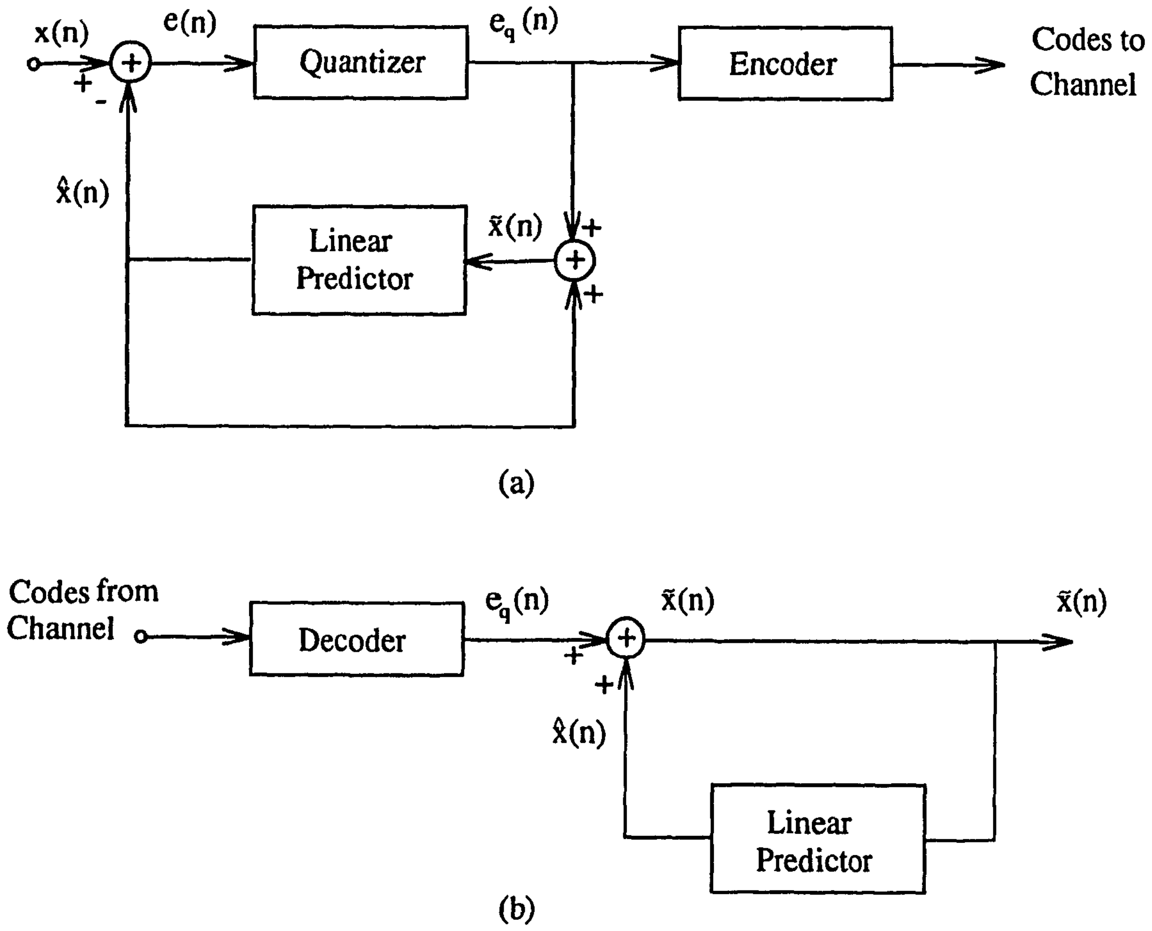


Figure 2.21: Linear Predictive Coding. (a) Transmitter. (b) Receiver.

Lossless Predictive coding is used to extend the compression achieved by the nonuniform subsampling method (the topic of this thesis) presented in Chapter 3.

2.3.1 Optimum Prediction

Consider the case where it is desired to estimate $x(n)$, the current sample value, from the set of N past input samples represented by the random sequence $\{x(n-1), x(n-2), \ldots, x(n-N)\}$. A linear prediction of $x(n)$ can be written as

$$\hat{x}(n) = \begin{bmatrix} x(n-1) & x(n-2) & x(n-3) & \ldots & x(n-N) \end{bmatrix} \begin{bmatrix} h_1 \\ h_2 \\ h_3 \\ \vdots \\ h_N \end{bmatrix} \tag{2.40}$$

in vector notation

$$\hat{x} = \mathbf{x}^T \mathbf{h} \tag{2.41}$$

where $\hat{x}$ (or $\hat{x}(n)$) is the predicted value of $x(n)$, T denotes the transpose, $\mathbf{x}$ is a column vector containing the past sample values and $\mathbf{h}$ is the vector of filter coefficients appropriately chosen to minimize the mean-square error given by

$$\sigma_e^2 = E\{(x(n) - \hat{x}(n))^2\} \tag{2.42}$$

Where E denotes the statistical expectation. Here it is assumed a zero-mean *wide sense stationary* (WSS) process. The vector $\mathbf{h}$ minimizes the mean-square error if its coefficients are chosen such that

$$E\{e(n)x(n-i)\} = 0 \qquad i = 1, 2, \ldots, N \tag{2.43}$$

That is, the minimum error $x(n) - \hat{x}(n)$ is statistically *orthogonal* to all the data used in the prediction. In a vector form, this can be written as

$$E\left\{\mathbf{x}\left(x(n) - \mathbf{x}^T\mathbf{h}\right)\right\} = 0 \tag{2.44}$$

Resulting in

$$\begin{bmatrix} R_{xx}(1) \\ R_{xx}(2) \\ R_{xx}(3) \\ \vdots \\ R_{xx}(N) \end{bmatrix} = \begin{bmatrix} R_{xx}(0) & R_{xx}(1) & R_{xx}(2) & \cdot & R_{xx}(N-1) \\ R_{xx}(1) & R_{xx}(0) & R_{xx}(1) & \cdot & R_{xx}(N-2) \\ R_{xx}(2) & R_{xx}(1) & R_{xx}(0) & \cdot & R_{xx}(N-3) \\ \vdots & \vdots & & & \vdots \\ R_{xx}(N-1) & R_{xx}(N-2) & R_{xx}(N-3) & \cdot & R_{xx}(0) \end{bmatrix} \begin{bmatrix} h_1 \\ h_2 \\ h_3 \\ \vdots \\ h_N \end{bmatrix} \quad (2.45)$$

Equation 2.45 can be written in matrix notation as

$$\mathbf{r}_{xx} = \mathbf{R}_{xx}\mathbf{h} \quad (2.46)$$

where

$$\mathbf{r}_{xx}{}^T = \{R_{xx}(i)\}\,; \quad \text{and} \quad \mathbf{R}_{xx} = \{R_{xx}(|i-j|)\}\,; \qquad i,j = 1,2,\ldots,N \quad (2.47)$$

The correlation matrix $\mathbf{R}_{xx}$ is a symmetric *Toeplitz* matrix. If the values of the correlation function are known, then Equation 2.46 can be solved with matrix inversion to get the set of coefficients for the optimum predictor,

$$\mathbf{h} = \mathbf{R}_{xx}^{-1}\mathbf{r}_{xx} \quad (2.48)$$

Note that the values of the correlation function may be calculated using typical statistical models (e.g. first order Autoregressive model (AR(1)) [6]), or empirically measured from real data. Equations 2.46, 2.47 and 2.48 are called *normal equations, Yule-Walker prediction equations* or *Wiener-Hopf equations* [6].

2.3.2 Two-Dimensional Image Predictor

A typical two-dimensional image predictor is designed to estimate the current pixel $x(i,j)$ from its causal neighboring pixels $x(i,j-1)$, $x(i-1,j)$, and $x(i-1,j-1)$ as illustrated in Figure 2.22. Two special cases of interest are the *isotropic* and *separable* autocorrelation models. In fact, according to experimental data, the autocorrelation functions of a large variety of images follow one of these two models [6, 22].

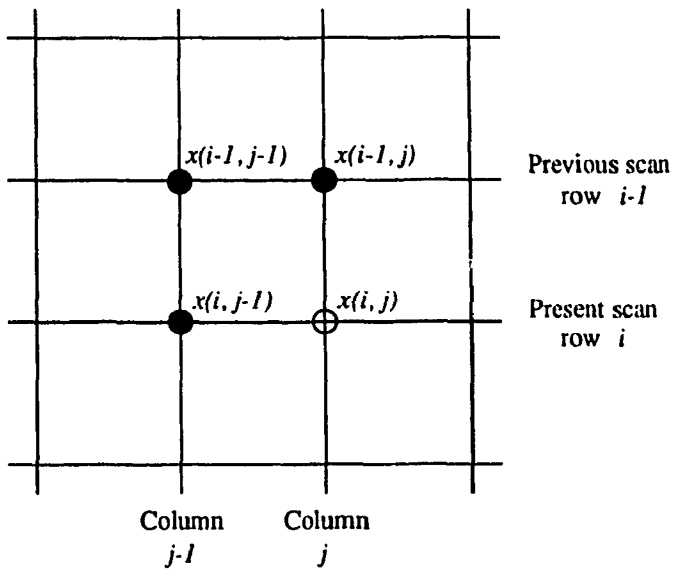


Figure 2.22: Neighboring pixels used in prediction of current pixel $x(i,j)$.

In Sections 2.3.2 and 2.3.2 both models are considered for two-dimensional image prediction.

Let ρ denotes the one-lag normalized autocorrelation value $R_{xx}(1)/R_{xx}(0)$. Assuming wide-sense stationarity, the correlation factor between adjacent samples in Figure 2.22 is equal to ρ. Further, samples on the diagonal have a correlation factor equal to ρ^m where $m = \sqrt{2}$ for an image model with an *isotropic* autocorrelation function and $m = 2$ for a *separable* image model [6].

Two-Dimensional Second Order Prediction

Consider the prediction of pixel intensity $x(i,j)$ from the previous samples $x(i,j-1)$ on the same line, and $x(i-1,j)$ on the previous neighbor line (see Figure 2.22). This can be expressed in the form

$$\hat{x}(i,j) = h_1 x(i,j-1) + h_2 x(i-1,j) \tag{2.49}$$

where h_1 and h_2 are the predictor coefficients. The optimal coefficients of the second order prediction filter [6] are given by

$$h_1 = h_2 = \frac{\rho}{1 + \rho^m} \tag{2.50}$$

where $m = \sqrt{2}$ for an image model with an isotropic autocorrelation function, and $m = 2$ for a model with a separable autocorrelation function. For a value $\rho = 0.95$ the optimal coefficients for the separable and isotropic models are approximately $h_1 = h_2 = 0.5$ which can be interpreted as an averaging predictor.

Two-Dimensional Third order Prediction

Consider the prediction of pixel $x(i,j)$ from the three neighbors $x(i,j-1)$, $x(i-1,j)$ and $x(i-1,j-1)$ shown in Figure 2.22. This can be expressed in the form

$$\hat{x}(i,j) = h_1 x(i,j-1) + h_2 x(i-1,j) + h_3 x(i-1,j-1) \tag{2.51}$$

The optimal coefficients of the third order predictor, for an image model with an isotropic autocorrelation function [6], are

$$h_1 = h_2 = \frac{\rho(1 - \rho^{\sqrt{2}})}{1 + \rho^{\sqrt{2}} - 2\rho^2} \tag{2.52}$$

and

$$h_3 = \rho^{\sqrt{2}} - \frac{2\rho^2(1 - \rho^{\sqrt{2}})}{1 + \rho^{\sqrt{2}} - 2\rho^2} \tag{2.53}$$

Likewise, the optimal coefficients for a model with a separable autocorrelation function are given by

$$h_1 = h_2 = \rho \qquad \text{and} \qquad h_3 = -\rho^2 \tag{2.54}$$

For a value $\rho = 0.95$, the predictor coefficients for an isotropic model are $h_1 = h_2 = 0.53$ and $h_3 = -0.08$. For a separable model, the coefficients are $h_1 = h_2 = 0.95$ and $h_3 =$ -0.9. Note that an approximation to the separable predictor is the easily implemented planar predictor [6] defined by

$$h_1 = h_2 = 1; \qquad \text{and} \qquad h_3 = -1 \tag{2.55}$$

Table 2.2: MSE for a 2-D second and third order prediction of Bridge, Airplane and Boy test images.

Prediction order	model	MSE		
		Bridge (256 × 256)	Airplane (512 × 512)	Boy (256 × 256)
2	isotropic	279.48	459.07	71.30
2	separable	277.94	425.47	66.92
3	isotropic	269.21	420.31	65.51
3	separable	361.77	243.02	41.77

Two Dimensional Prediction: An Example

An example using test images illustrates second and third order prediction of images discussed in Sections 2.3.2 and 2.3.2.

Table 2.2 shows the mean square error for a 2-D second and third order prediction of originals Bridge, Airplane and Boy test images using separable and isotropic models ($\rho = 0.95$). The original images are shown in Figure 2.24 (a), Figure 2.26 (a) and Figure 2.28 (a) respectively. The mean square prediction error (MSE) was evaluated using a deterministic equation [22] given by

$$MSE = \frac{1}{N^2}\sum_{i=1}^{N}\sum_{j=1}^{N}[x(i,j) - \hat{x}(i,j)]^2 \qquad (2.56)$$

where N^2 is the number of pixels in the original $N \times N$ image, and $\hat{x}(i,j)$ is the predicted value of the original $x(i,j)$. It can be seen from Table 2.2 that the third order prediction (in general) do better than second order prediction. Further, a comparisor between Bridge and Boy images (as they both have the same size) shows that the prediction error is smaller for images having large areas of slowly varying luminance and simple details.

The prediction error is usually coded using *variable-length coding* (entropy coding). This method is discussed briefly in the next Section. Huffman coding, an efficient variable length coding technique, is discussed in Section 2.3.3. This technique is used in Chapter 3 to code the prediction error of the subsampled images.

2.3.3 Variable Length Coding

Consider a monochrome image quantized to B bits/pixel. A reduction in the average number of bits per word can be achieved by assigning shorter code-words to luminance levels having high probability of occurrence, whereas longer code-words are assigned to levels having lower probability. This method is called *Variable Length Coding* (VLC) [5]. The average code-word length for the image will be

$$B_{av} = \sum_k L_k P_k \tag{2.57}$$

where L_k denotes the code-word length assigned to the kth luminance level, and P_k is its probability of occurrence. A fundamental result due to Shannon [26] establishes the entropy H of the source as a lower bound for the average number of bits per source symbol needed to code a discrete source, i.e.,

$$B_{av} \geq H \tag{2.58}$$

and

$$H = -\sum_k P_k log_2 P_k \tag{2.59}$$

as $\{P_k\}$ becomes highly concentrated, the entropy becomes smaller, and a variable length coding technique is more advantageous.

Huffman Coding

Huffman code [27, 28] is a statistical variable-length coding that satisfies the *prefix rule*, which states that no code word forms the prefix of any other, that is, the code is uniquely decodable once the starting point of the symbol sequences is known. The code achieves the entropy rate when all symbol probabilities are integral powers of 1/2. The Huffman coding scheme is as follow:

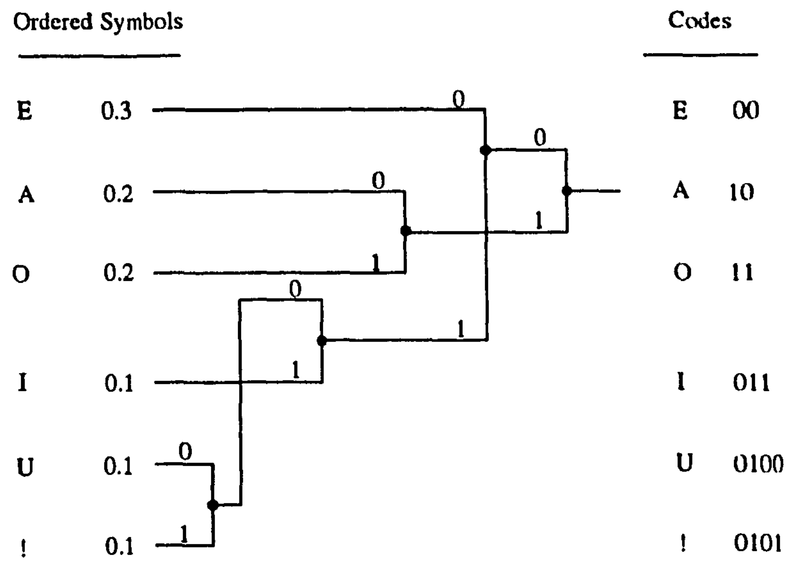


Figure 2.23: An example of Huffman Coding.

- List all possible set of symbols with their probabilities, in decreasing probability order.
- Locate the two symbols with the smallest probabilities and aggregate them into a single new node whose probability is the sum of their individual probabilities.
- Repeat until the entire symbol set is represented by a single node.

The result is a Huffman tree with all symbols as leaf nodes. A code can be generated for each symbol by assigning a binary digit to each branch, then by following the path from the top node to the symbol leaf node. Figure 2.23 illustrates the Huffman coding for a message having the symbols $\{A, E, I, O, U, !\}$ with probability $\{0.2, 0.3, 0.1, 0.2, 0.1, 0.1\}$ respectively.

Huffman code has an average word-length that lies in the range

$$H \leq B_{av} \leq H + 1 \tag{2.60}$$

However, as symbols in a Huffman coding scheme translate into integral number of bits; the average word-length is also constrained by

$$R_{av} \geq 1 \;\; bit/symbol \tag{2.61}$$

regardless of how small the entropy is. Note that for a given probability distribution, there are many possible Huffman codes having the same word-length. Furthermore, Huffman coding requires that one or more sets of Huffman code tables be specified by the application, that is, the same tables used for coding are also needed for decoding. Other variable length coding techniques that can approach the theoretical entropy bound to compression efficiency are available (e.g. arithmetic coding [29, 30, 31]), but simple methods like Huffman coding are good enough to be used in this work.

Variable Length Coding: An Example

The advantage of using a variable length coding technique to code the prediction error of images is illustrated in this example.

Figure 2.25 (a), Figure 2.27 (a) and Figure 2.29 (a) show the histograms of original (Program A.1) Bridge, Airplane and Boy images respectively. The predictors used in this example are different from those discussed previously. Here a predictor is selected from one of the seven types used by the Joint Photographic Experts Group (JPEG) [8] and shown in Table 2.3. These predictors have the advantage that they are easy to implement in practice. Note that selection 7 corresponds to the two-dimensional second order prediction for both isotropic and separable models discussed in Section 2.3.2 with $\rho = 1$, while selection 4 corresponds to the two-dimensional third order prediction of a separable model with $\rho = 1$. Selections 1 to 3 are one-dimensional predictors, while 4 to 7 form two-dimensional predictors. The predictor can switch between any of the seven selections. The one which yields the minimum mean square prediction error defined by Equation 2.56 is chosen.

The prediction error images of Bridge, Airplane and Boy are shown in Figure 2.24 (b), Figure 2.26 (b) and Figure 2.28 (b). respectively. These images are formed by a linear mapping of the difference (which can be both positive or negative) between the original and the prediction images onto the range $0 - 255$ (8 bits/pixel) for display. It can be seen that linear prediction removes much of the redundant

Table 2.3: $1-D$ and $2-D$ Practical predictors

Selection value	Prediction
1	$x(i,j-1)$
2	$x(i-1,j)$
3	$x(i-1,j-1)$
4	$x(i,j-1)+x(i-1,j)-x(i-1,j-1)$
5	$x(i,j-1)+[(x(i-1,j)-x(i-1,j-1))/2]$
6	$x(i-1,j)+[(x(i,j-1)-x(i-1,j-1))/2]$
7	$[x(i,j-1)+x(i-1,j)]/2$

information from the image, leaving only information about the edges. The prediction error histograms (Program A.2) are shown in Figure 2.25 (b), Figure 2.27 (b) and Figure 2.29 (b) for Bridge, Airplane and Boy images respectively. Because a great deal of inter-pixel redundancy is removed by the prediction, the histogram of the prediction error is, in general, highly peaked about zero.

The entropy values of the original images and of their corresponding prediction error are shown in Table 2.4. Note the reduction of H as the visual appearance of images becomes simpler, and larger areas of uniform or slowly varying luminance are present. Moreover, the entropy of the prediction error is significantly smaller than the entropy of the corresponding original images. It can be seen that a variable length coding leads to a much smaller average word-length in coding the prediction error than in coding the original image. In fact, the amount of compression achieved in lossless predictive coding is related directly to the entropy reduction that results from mapping the input image into the prediction error sequence.

Next chapter presents the topic of this thesis, a nonuniform subsampling method suitable for "region-of-interest" applications.

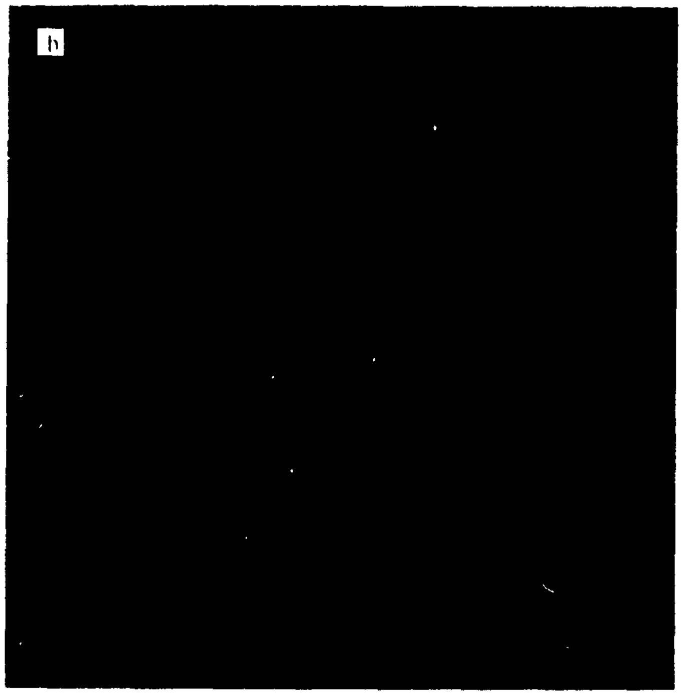


Figure 2.24 (a) Original Bridge image (256 × 256), (b) Error image

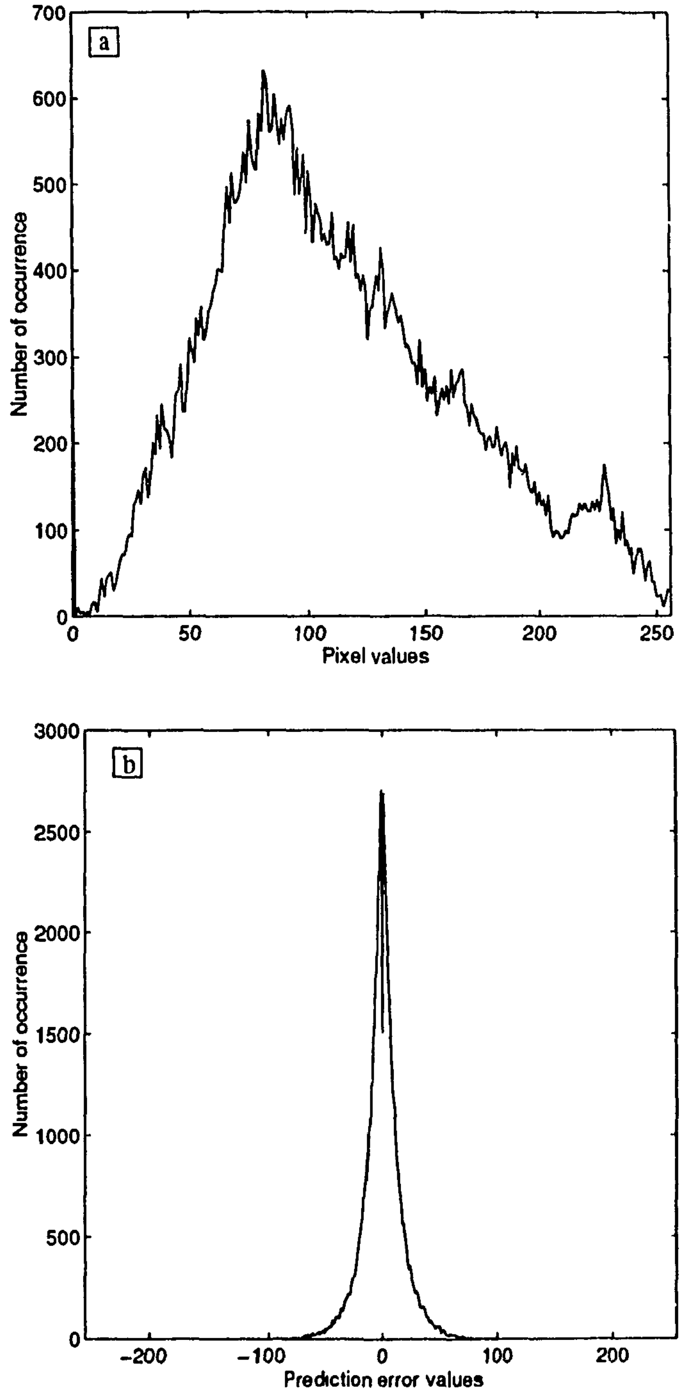


Figure 2.25: a) Histogram of the original Bridge image, b) Prediction error histogram (prediction using selection 7 in Table 2.3).

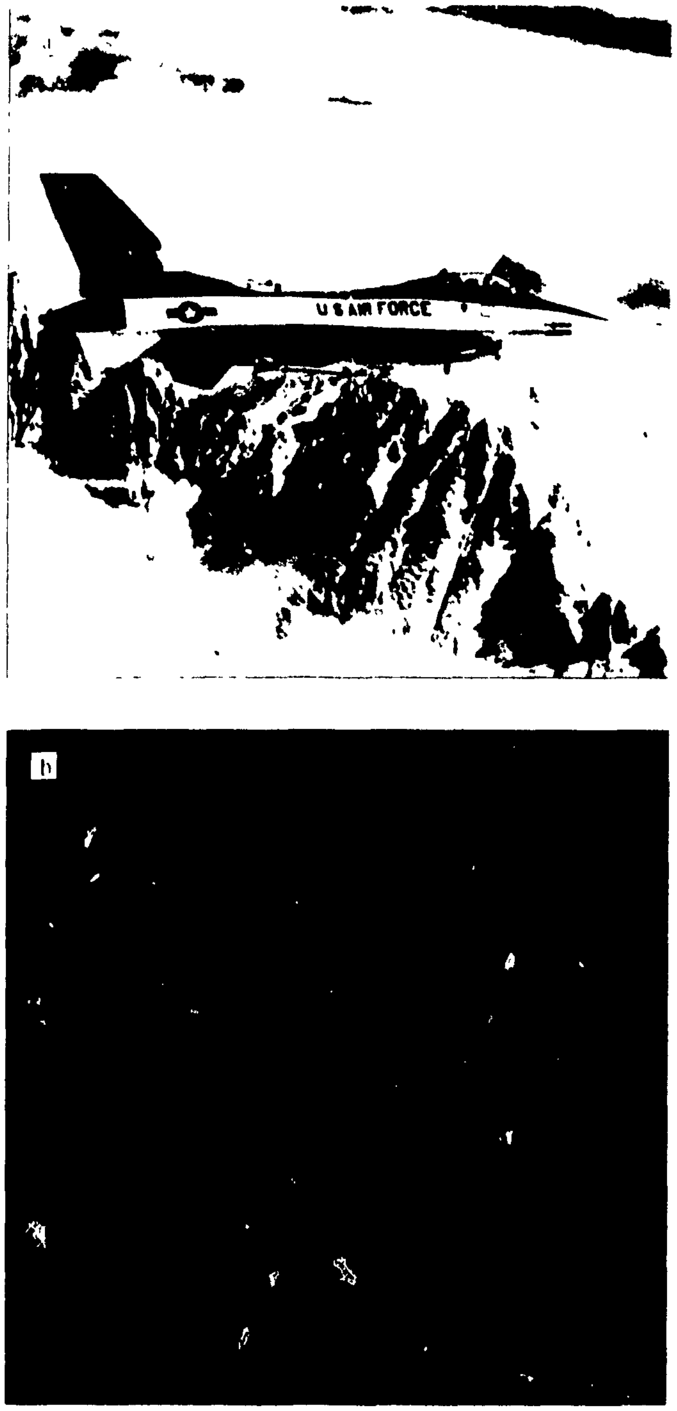
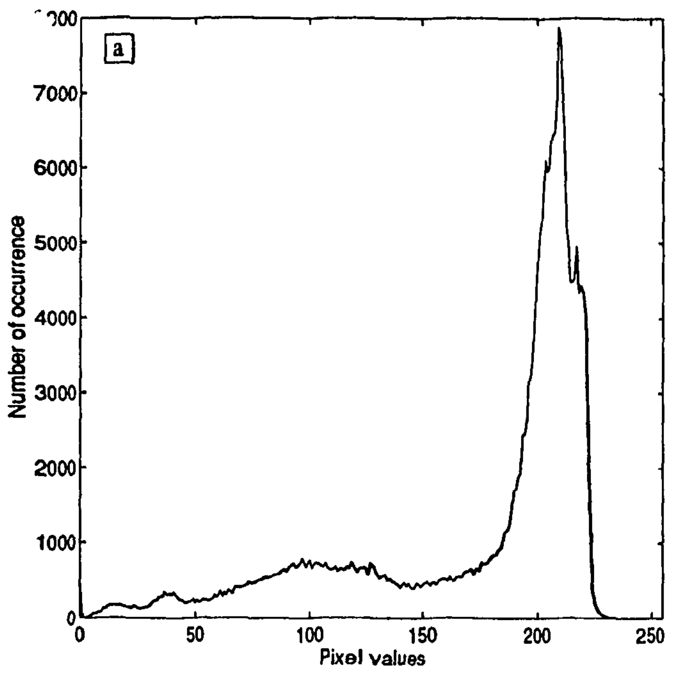


Figure 2.26: a) Original Airplane image (512 × 512), b) Error image.



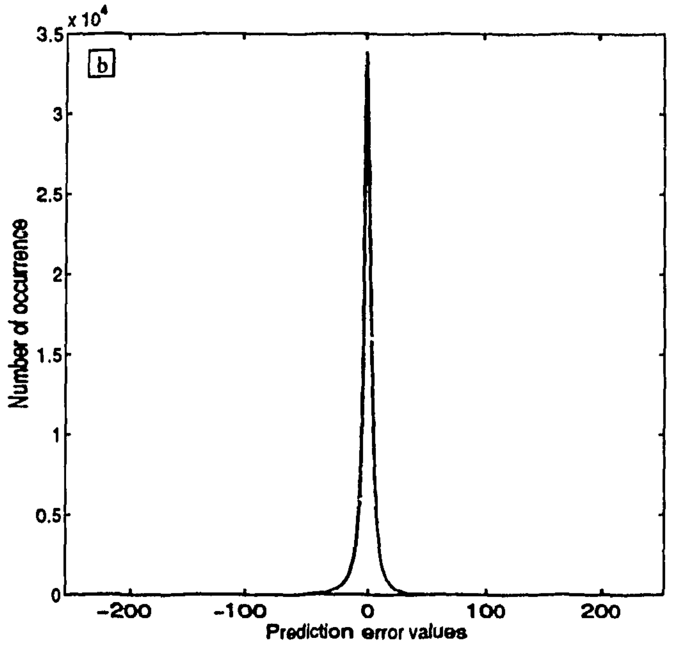


Figure 2.27: a) Histogram of the original Airplane image, b) Prediction error histogram (prediction using selection 4 in Table 2.3).

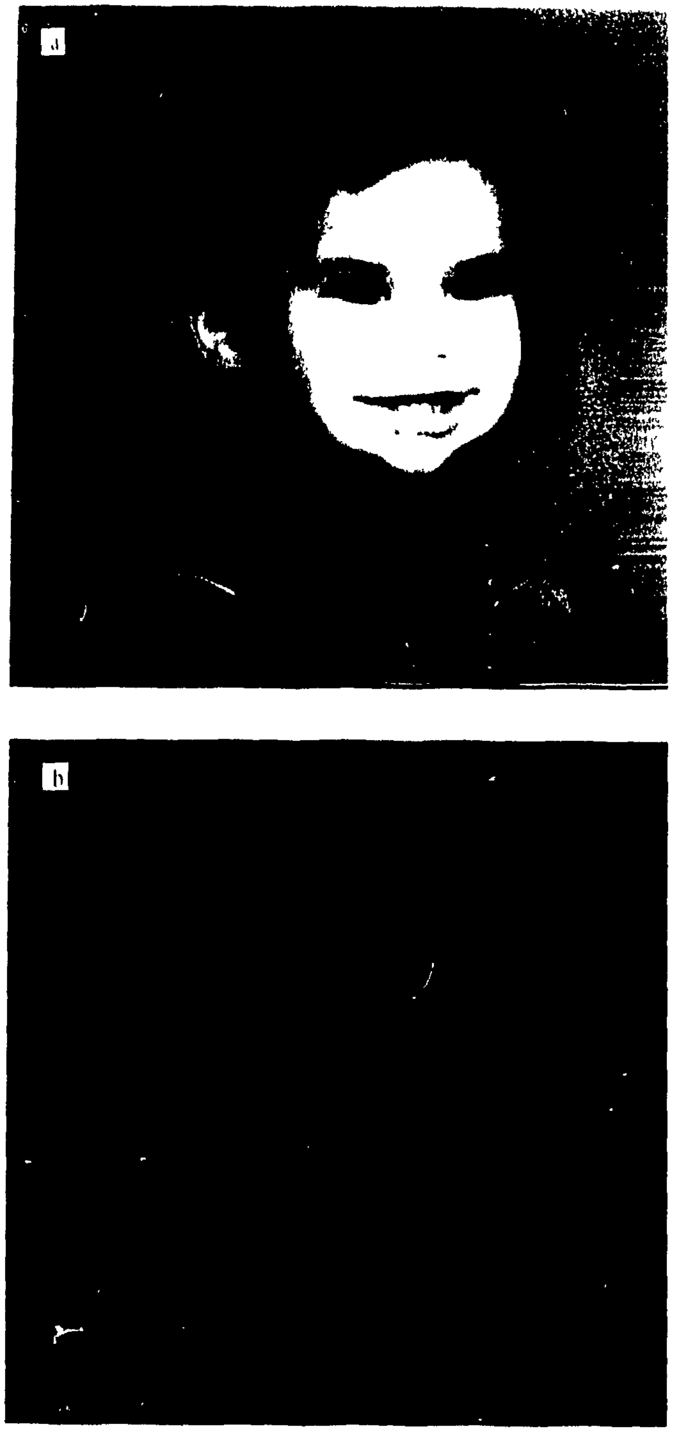
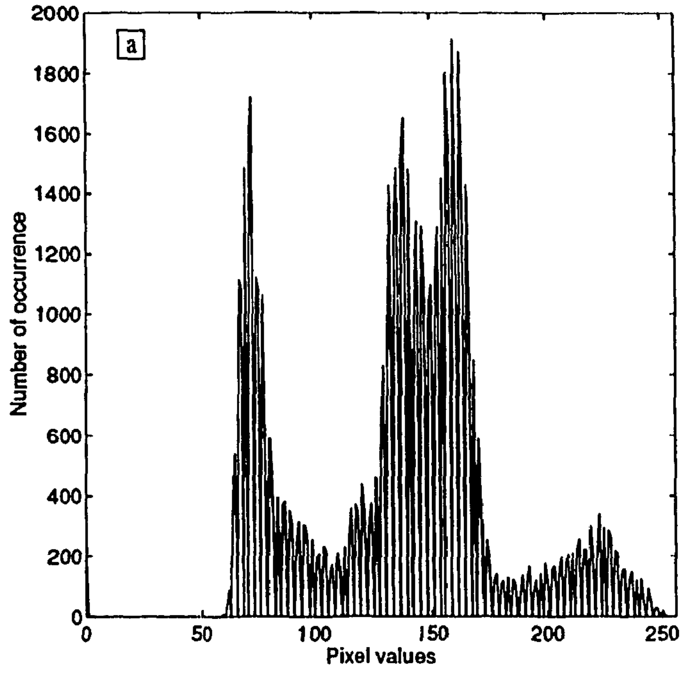


Figure 2.28: a) Original Boy image (256 × 256), b) Error image.



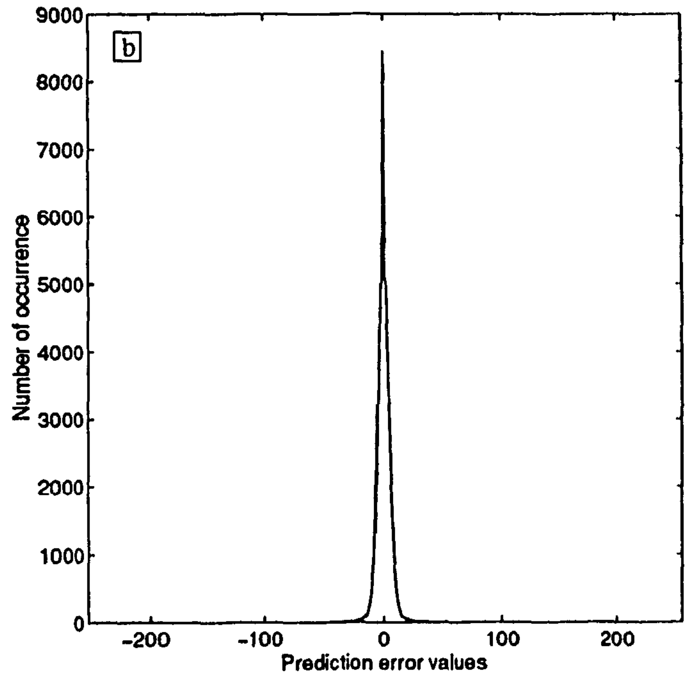


Figure 2.29: a) Histogram of the original Boy image, b) Prediction error histogram (prediction using selection 5 in Table 2.3).

Table 2.4: Entropy values (bits per pixel) of original Bridge, Airplane and Boy images and of their corresponding prediction error.

Image	H (original image)	H (prediction error)
Bridge	7.6686	6.0117
Airplane	6.7990	4.5868
Boy	6.3264	4.3911

Chapter 3

Nonuniform Weighted Subsampling for Digital Image Compression

3.1 Introduction

Subsampling is a simple image compression method where only a subset of the pixels are retained. Uniform subsampling saves regularly spaced pixel locations. This method is suitable when all regions in the image are equally importan.. However, in applications where a prior knowledge of a certain region-of-interest is available, a nonuniform subsampling scheme is more effective. Nonuniform subsampling can be used to assign more samples to important regions of the image, enhancing their quality in reconstruction.

Many studies in the literature have applied the subsampling method in image compression. In [32], uniform subsampling was used in chrominance decimation of color images. In [33], adaptive subsampling was presented. Here the image is divided into square blocks and each is assigned a subsampling mode depending on the rate-distortion of that block. In highly detailed Blocks a dense sampling lattice is used, and in blocks with little details or slowly varying luminance, only a few pixels are retained. This results in discontinuities in the sampling grid. The time-axis transform (TAT) was presented in [34]. The TAT compresses the bandwidth of picture signals by reducing the number of transmitted pixels based on a hybrid system of fixed and variable sampling. In [4], a nonuniform image representation scheme suitable for "area-of-interest" imaging was presented. The scheme considers the class of images that can be sampled according to a nonuniform sampling grid with emphasis on the case of sampling density that is monotonically decreasing as a function of the centre of the area of interest. A generalized pyramidal approach with application to images sampled nonuniformly both in Cartesian and polar coordinate systems was also presented.

In this chapter, Weighted Subsampling (WS), a new nonuniform image subsampling scheme is presented [18]. The method uses a weighting function to change the sampling pattern so that more samples are taken in important regions of the image. The first order autoregressive model (AR(1)) [6, 35] is used to model the input image. This model has received much attention because of its analytical and computational tractability as well as its good performance in representing correlation matrices of real images.

In the presented scheme, it is assumed that some portions of the image are known *a priori* to be more important than others. WS is derived so that there are more samples in these portions than in other areas. Classes of images where this sort of *a priori* knowledge may be available are data bases of human faces where the (visually important) face is centered, or in industrial inspection photographs where the part of interest is known to always be located in a particular region of the image. Here we use WS to code images of human faces. Chapter 5 will present future work including an adaptive WS able to locate human faces. Various applications result from combining an automatic human face location system to the proposed WS method. Some of these applications are:

- Automatic human face recognition systems.
- Identification of criminals.
- Teleconference.
- Verification of identity at security sites.
- Surveillance systems.

The material in this chapter includes two parts. The first part presents the nonuniform weighted subsampling scheme. In Section 3.2.1 an approach to uniform subsampling starting with an AR(1) model is presented. The idea of WS is then described in Section 3.2.2. The simulated annealing algorithm used to deduce the nonuniform sampling pattern is presented in Section 3.3. This optimization is done offline.

Weighted Subsampling using Projection (WSP) is presented in Section 3.4. Here, the signal which is subsampled and transmitted is the projection of the weighted input signal. This projection is in effect the KL transform of the signal. In Section 3.4.4, an example using test images illustrates WSP.

In WS, signal projection is approximated using a space varying linear low filter. This is described in Section 3.5. Image reconstruction from its nonuniform samples is treated in Section 3.6 where a space varying interpolation filter with an adjustable gain is used to reconstruct the image. In Section 3.7, experimental results including subjective and objective evaluations demonstrate the effectiveness of WS. A comparison between WSP (using projection) and WS (using filtering) is also presented.

The second part in this chapter extends the compression achieved by WS using lossless predictive coding. A switched prediction based on the density of the sampling pattern is used. In Section 3.8.1 a suitable prediction filter for each type of subsampling pattern is evaluated using the least-square method. The prediction error is then computed and coded using a Huffman code. Section 3.8.2 shows the results obtained by combining WS with lossless compression.

3.2 Weighted Subsampling

The input signals are assumed to be finite length digital signals and are denoted $x \in \mathcal{R}^N$. The subsampling problem is to select a subset of the components of x which represents the entire signal. In the remainder of this work analysis is done on one dimensional signals and the results are extended to two dimensions by first sampling horizontally, then vertically (i.e. separable model).

3.2.1 Uniform Subsampling

One of the ways that uniform subsampling can be derived for digital images is to start with the AR(1) model. The autocorrelation matrix (R_{xx}) is $N \times N$ and the element in its ith row and jth column is $\rho^{|i-j|}$; i.e.

$$R_{xx} = \sigma_x^2 \begin{bmatrix} 1 & \rho & \rho^2 & \cdot & \cdot & \rho^{N-1} \\ \rho & 1 & \rho & \cdot & \cdot & \cdot \\ \rho^2 & \rho & 1 & \rho & & \cdot \\ \vdots & \vdots & \rho & 1 & & \cdot \\ \cdot & & & & \cdot & \rho \\ \rho^{N-1} & \cdot & \cdot & & \cdot & 1 \end{bmatrix} \tag{3.1}$$

where σ_x^2 is the variance of the ensemble of samples and ρ is the model parameter (intersample correlation coefficient). The eigenvectors of this matrix yield the Karhunen-Loève Transform (KLT) [6, 36] for this model (since the matrix is symmetric they will be orthogonal). Since R_{xx} is a symmetric positive definite matrix, its eigenvalues are real and positive. The eigenvalue associated with each eigenvector gives a measure of how much of the signal energy lies along the direction of that eigenvector. The higher the eigenvalue the "more important" the corresponding eigenvector [6, 7]. Selecting the ($M < N$) eigenvectors with the highest eigenvalues defines the most important subspace of dimension M. Putting these eigenvectors as columns into the $N \times M$ matrix A the projection of the input signal to this subspace is:

$$x_p = Ay = AA^T x \tag{3.2}$$

Here y contains the M most important KLT coefficients. Note that the eigenvectors in the matrix A are normalized to length one. Figure 3.1 shows the eight eigenvectors with the largest eigen values for a 256×256 autocorrelation matrix with $\rho = 0.95$. The corresponding eigen values are given in Table 3.1. A plot of all eigenvalues for this autocorrelation matrix is shown in Figure 3.2. it can be seen from the plot that the KLT has the capability of compacting the energy in few coefficients.

If A contains M orthogonal rows then x_p can be represented by the corresponding M components of x_p. (this is "essentially" Kramer's sampling theorem [20, 37].)

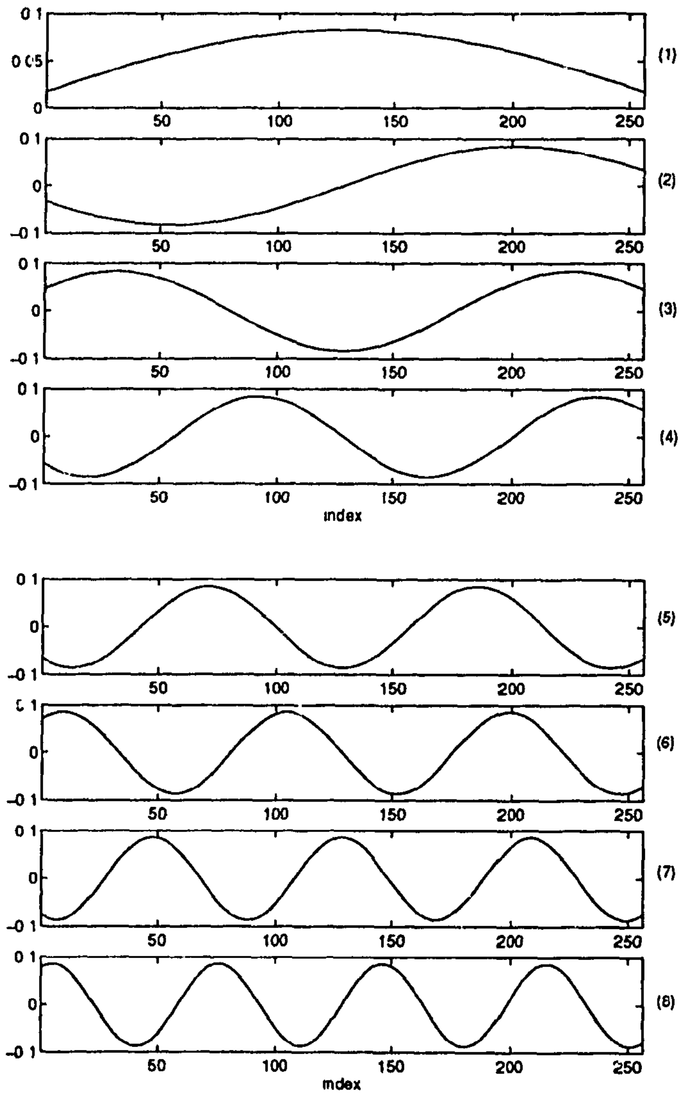


Figure 3.1: The eight eigenvectors corresponding to the largest eigenvalues for a 256×256 autocorrelation matrix of an AR(1) model ($\rho = 0.95$).

Table 3.1: Eigenvalues associated with the eight eigenvectors of Figure 3.1.

Eigenvector *No.*	Eigenvalues
1	37.3829
2	33.1968
3	27.8738
4	22.6520
5	18.1630
6	14.5557
7	11.7450
8	9.5766

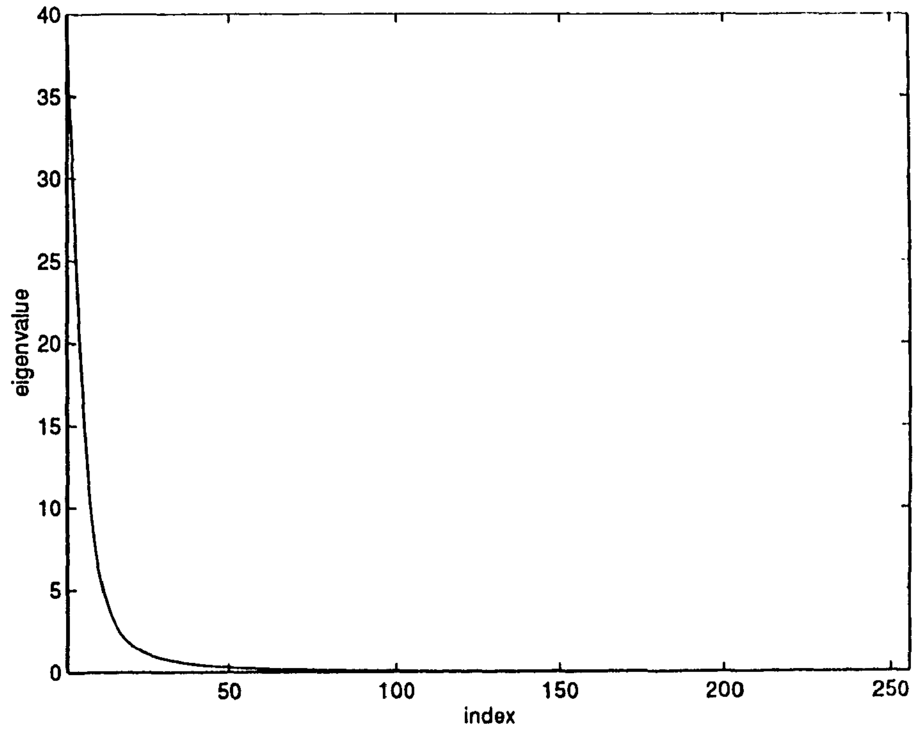


Figure 3.2: Plot of all eigenvalues for a 256×256 autocorrelation matrix of an AR(1) model ($\rho = 0.95$).

Table 3.2: Eigenvectors of an 8×8 autocorrelation matrix R_{xx} and their corresponding eigenvalues ($\rho = 0.95$).

Eigenvectors								Eigenvalues
0.3383	0.3512	0.3599	0.3642	0.3642	0.3599	0.3512	0.3383	7.0303
-0.4809	-0.4204	-0.2860	-0.1013	0.1013	0.2860	0.4204	0.4809	0.5751
0.4665	0.2065	-0.1789	-0.4557	-0.4557	-0.1789	0.2065	0.4665	0.1683
0.4226	-0.0854	-0.4865	-0.2783	0.2783	0.4865	0.0854	-0.4226	0.0818
0.3602	-0.3468	-0.3558	0.3513	0.3513	-0.3558	-0.3468	0.3602	0.0509
0.2833	-0.4882	0.0942	0.4154	-0.4154	-0.0942	0.4882	-0.2833	0.0370
-0.1952	0.4623	-0.4603	0.1904	0.1904	-0.4603	0.4623	-0.1952	0.0300
0.0996	-0.2786	0.4156	-0.4896	0.4896	-0.4156	0.2786	-0.0996	0.0266

If the parameter ρ in the standard AR(1) model is close to one, the eigenvectors approach sinusoids, evenly spaced in frequency, with higher eigenvalues being associated with the lower frequencies (see Figure 3.1 and Table 3.1). In this case, selecting the M eigenvectors with the largest eigenvalues and projecting the signal onto them is approximately equivalent to bandlimiting the signal. If M is a factor of N, evenly spaced rows of A will be orthogonal, thus yielding uniform subsampling.

Uniform Subsampling: An example

A numerical example illustrates the approach to uniform subsampling discussed in this section. The eigenvectors of an 8×8 autocorrelation matrix ($\rho = 0.95$) and their corresponding eigenvalues are given in Table 3.2. Selecting four eigenvectors corresponding to the largest eigenvalues, and putting them as columns in a matrix A, the task is to find four rows of this matrix which are "most" orthogonal. This was measured by normalizing each row to be of length one and then checking the determinant of the different 4×4 submatrices, looking for the largest one. Normalizing the rows of A results in the matrix

$$A_{\mathcal{N}} = \begin{bmatrix} 0.4906 & 0.5416 & -0.5583 & 0.3928 \\ -0.1444 & 0.3491 & -0.7106 & 0.5936 \\ -0.7022 & -0.2583 & -0.4128 & 0.5195 \\ -0.4254 & -0.6965 & -0.1548 & 0.5567 \\ 0.4254 & -0.6965 & 0.1548 & 0.5567 \\ 0.7022 & -0.2583 & 0.4128 & 0.5195 \\ 0.1444 & 0.3491 & 0.7106 & 0.5936 \\ -0.4906 & 0.5416 & 0.5583 & 0.3928 \end{bmatrix} \qquad (3.3)$$

where the suffix in $A_{\mathcal{N}}$ denotes the normalization of the rows of A. A full search algorithm is used for this (small scale) problem to check the determinant of all combinations of 4×4 submatrices looking for the largest one. The highest determinant value in this problem is 0.9320 and corresponds to the rows number: $2, 4, 6$, and 8. Note that rows: $1, 3, 5$, and 7 have also the same determinant value. It can be seen that the result agree with the approach presented in this section, i.e., the indices of these rows yield uniform subsampling.

3.2.2 Nonuniform Weighted Subsampling (WS)

In the previous section all spatial locations are given the same weight. In this section more importance is placed on certain pixel locations using a weighting function. A typical weighting function is shown in Fig. 3.3. Here pixels in the centre are more important than pixels on the sides. The weighted signal is:

$$u = diag[W(i)]x = \mathcal{D}_W x \qquad (3.4)$$

Where $W(i)$ is the weighting function. Weighting the signal can be thought as weighting the norm of the error. Thus, errors in the regions where the weighting function is large have a more dramatic effect on the norm of the error. Usually, the

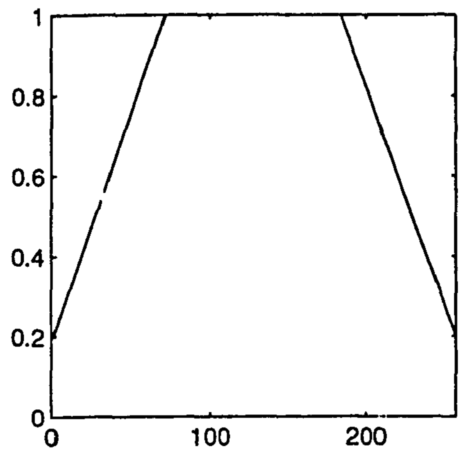


Figure 3.3: Weighting function.

values of $W(i)$ are chosen such that

$$0 < W(i) \leq 1 \tag{3.5}$$

If the part of interest in the image is not centered, then two different weighting functions could be used for the horizontal and vertical directions. The autocorrelation matrix of the weighted signal u is:

$$R_{uu} = E[uu^T] = E[\mathcal{D}_W xx^T \mathcal{D}_W] = \mathcal{D}_W R_{xx} \mathcal{D}_W \tag{3.6}$$

Figure 3.4 shows the 8 eigenvectors of R_{uu} (256×256) corresponding to the largest eigenvalues ($\rho = 0.95$). Here the weighting function given in Figure 3.3 is used. Selecting the M eigenvectors of R_{uu} with the largest eigenvalues and putting them as columns in matrix A_W the task is to find M rows of this matrix which are "most" orthogonal. This is measured, as previous, by normalizing each row to be of length one and then checking the determinant of the different $M \times M$ submatrices, looking for the largest one. Simulated Annealing [38] is used to optimize over the N choose M possibilities. This optimization is done offline once the weighting function is set and is not required for each image. Fig. 3.5 shows selected pixel locations for the given weighting function with $N = 256$, $M = 64$ and $\rho = 0.95$. These represent the

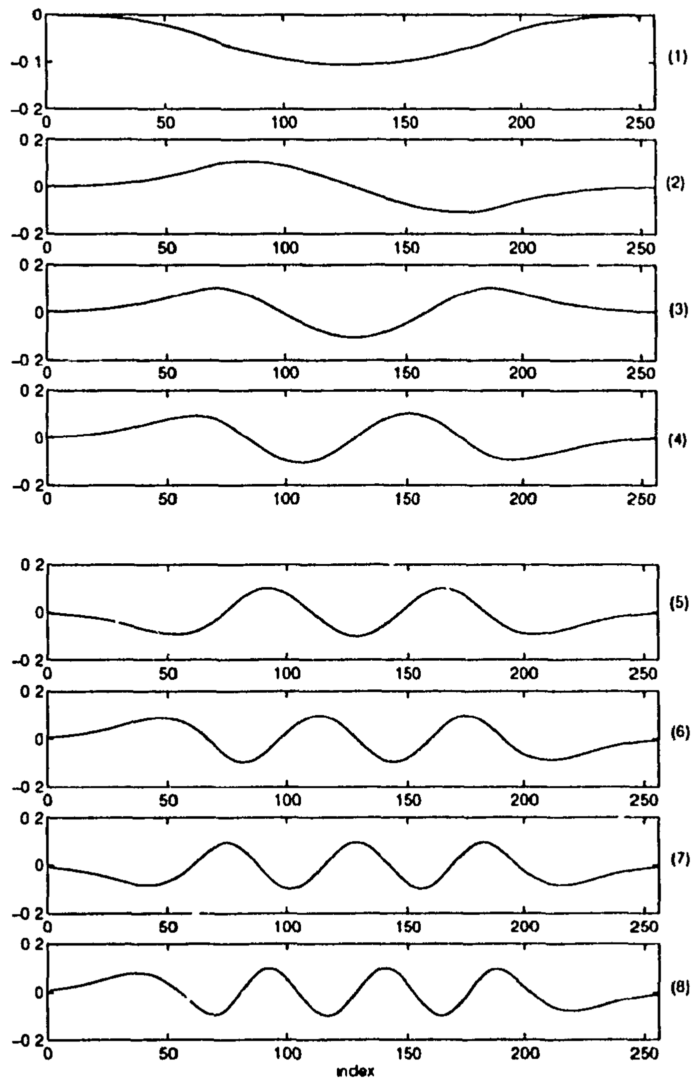


Figure 3.4: The eight eigenvectors corresponding to the largest eigenvalues for a 256×256 autocorrelation matrix of the weighted signal(Equation 3.6). Here $\rho = 0.95$.

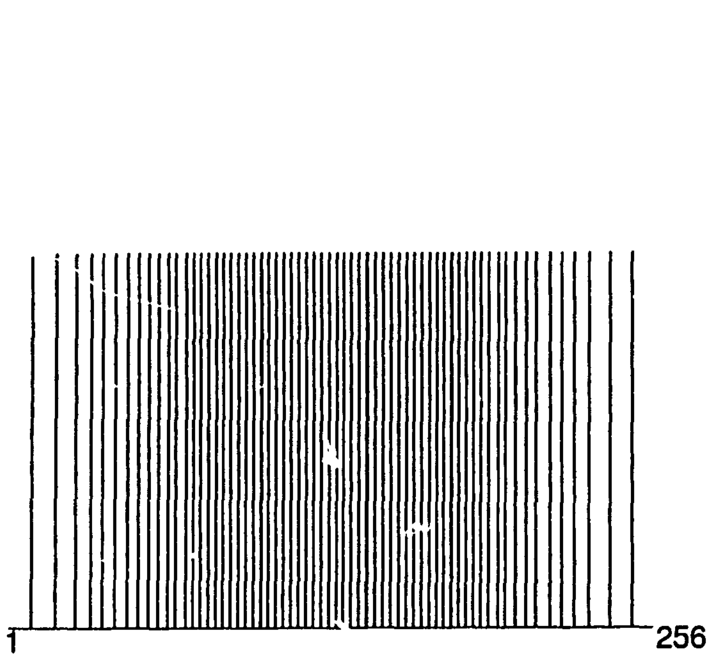


Figure 3.5: Selected pixel locations representing the nonuniform subsampling pattern.

nonuniform sampling pattern applied to each row and column. The main program of WS used to derive the sampling pattern, together with the simulated annealing algorithm are given in Programs A.6 and A.7 respectively.

Nonuniform Weighted Subsampling: An Example

A numerical (small scale) example illustrates the proposed WS scheme. The autocorrelation R_{uu} of the weighted signal given by equation 3.6 is computed using an 8×8 autocorrelation matrix R_{xx} with $\rho = 0.95$ and a weighting function

$$W = \begin{bmatrix} 0.1 & 0.1 & 1 & 1 & 1 & 1 & 0.1 & 0.1 \end{bmatrix} \tag{3.7}$$

Table 3.3 shows the eigenvectors of R_{uu} and their corresponding eigenvalues. The four eigenvectors corresponding to the largest eigenvalues are selected. Putting these eigenvectors as columns in a matrix A_W and normalizing each row of this matrix to

Table 3.3: Eigenvectors of the 8 × 8 autocorrelation matrix R_{uv} and their corresponding eigenvalues ($\rho = 0.95$).

Eigenvectors								Eigenvalues
0.0444	0.0467	0.4917	0.5040	0.5040	0.4917	0.0467	0.0444	3.788235226122
-0.0594	-0.0622	-0.6471	-0.2718	0.2718	0.6471	0.0622	0.0594	0.165059658751
0.0477	0.0492	0.4994	-0.4959	-0.4959	0.4994	0.0492	0.0477	0.051071122171
0.0268	0.0273	0.2691	-0.6528	0.6528	-0.2691	-0.0273	-0.0268	0.030054325999
-0.5946	-0.3717	0.0913	-0.0022	-0.0022	0.0913	-0.3717	-0.5946	0.002400791066
-0.5946	-0.3717	0.0913	-0.0023	0.0023	-0.0913	0.3717	0.5946	0.002400699938
-0.3762	0.5961	-0.0228	0.0001	-0.0001	0.0229	-0.5993	0.3781	0.000389132974
-0.3781	0.5993	-0.0229	0.0001	0.0001	-0.0228	0.5961	-0.3762	0.000389132975

a length one yields the matrix A_{W_N} given by:

$$A_{W_N} = \begin{bmatrix} 0.2906 & 0.5175 & -0.6447 & 0.4819 \\ 0.2841 & 0.5129 & -0.6476 & 0.4868 \\ 0.2715 & 0.5038 & -0.6529 & 0.4961 \\ -0.6528 & -0.4959 & -0.2718 & 0.5040 \\ 0.6528 & -0.4959 & 0.2718 & 0.5040 \\ -0.2715 & 0.5038 & 0.6529 & 0.4961 \\ -0.2841 & 0.5129 & 0.6476 & 0.4868 \\ -0.2906 & 0.5175 & 0.6447 & 0.4819 \end{bmatrix} \tag{3.8}$$

A full search is used (Program A.5), as in previous example of the uniform case, to find the set of four most orthogonal rows of A_{W_N}. Here, the largest determinant value is 1 and corresponds to the rows number 3,4,5, and 6. Note that for a determinant value of one (optimal value), the corresponding rows are orthogonal. The above result was expected as the weighting function places more importance on the central components of the signal.

In WS, the sampling pattern for a certain weighting function is derived offline using the simulated annealing optimization method. Next section presents the simulated annealing algorithm used for this particular problem.

3.3 The Simulated Annealing Algorithm

The method of Simulated Annealing (SA) [38, 39] has attracted many researchers as it is suitable for combinatorial optimization problems of large scale. The algorithm can be seen as an iterative improvement (or local search) that focuses on finding minimum or maximum values of a function of independent variables. This function is usually called the *cost function* or *objective function.* A solution to the optimization problem is often referred to as a "configuration". The moves from one configuration to another is randomly chosen. An annealing schedule is used to control the moves between configurations.

For our particular problem, the objective function is the determinant value D of the matrix A_{W_N} (of normalized rows), and its maximization is the goal of the algorithm. A configuration C is defined as the set of M rows of A_{W_N}. In each move to a new configuration, the resulting difference ΔD between determinants value of the new and old configurations is computed.

$$\Delta D = D_{new} - D_{old} \tag{3.9}$$

The new configuration C_{new} is accepted if $\Delta D \geq 0$ and accepted with probability $\exp(\Delta D/T_c)$ if $\Delta D \leq 0$ where T_c is a control parameter (usually called temperature). The starting value of the parameter T_c is chosen considerably larger than the largest ΔD such that all proposed reconfigurations are accepted. Here a value of $T_c = 10$ is chosen as initial temperature. A cooling schedule decides how the value of T_c is updated (lowered) as well as the number of iterations for each temperature. The algorithm is stopped when no further improvement is expected. This condition is known as "thermal equilibrium". The simulated annealing optimization method used for this particular problem is as follows:

1. Choose an initial configuration C_0 selected at random.

   Choose an initial temperature T_{c_0} and a temperature reduction coefficient α_{T_c}.

   Set $C_{old} = C_0$ and $D_{old} = D_0$ (D_0 is the determinant value of C_0).

2. Generate a new configuration C_{new}.
   If $D_{new} \geq D_{old}$ then accept the new configuration,
   set $C_{old} = C_{new}$ and $D_{old} = D_{new}$;
   else:
   compute the acceptance probability $r = \exp\left(\frac{D_{new} - D_{old}}{T_c}\right)$
   If $random\ (0, 1) \ \leq\ r$ then set $C_{old} = C_{new}$ and $D_{old} = D_{new}$.

3. Update T_c after a number J reconfigurations or K successful reconfigurations ($J > K$) is reached, whichever comes first. An updating rule proposed in [39] is used. This is given by:

$$T_{c_u} = \alpha_{T_c} T_c$$

   where T_{c_u} and T_c are the new and current temperature parameter values respectively and $0 < \alpha_{T_c} < 1$. Here, $\alpha_{T_c} = 0.9$ is used.

4. If thermal equilibrium is not reached, then goto step 2.
   else, $C_{opt} = C_{new}$ and $D_{opt} = D_{new}$.
   Output best configuration C_{opt}, its corresponding row indices, and stop.

Figure 3.6 shows the search algorithm looking for the largest normalized determinant. The search was used to derive the sampling pattern of Figure 3.5. The computation time was evaluated on a SPARC 5 station.

3.4 Weighted Subsampling Using Projection (WSP)

In Section 3.2.2 the signal which is subsampled, is the projection of the weighted input signal onto the selected subspace of M eigenvectors. Only M samples of the projection are saved. Signal reconstruction can be achieved using straightforward matrix manipulation and an inverse weighting. Weighted Subsampling using Projection (WSP) is equivalent to a KLT coding of the weighted signal, with the difference

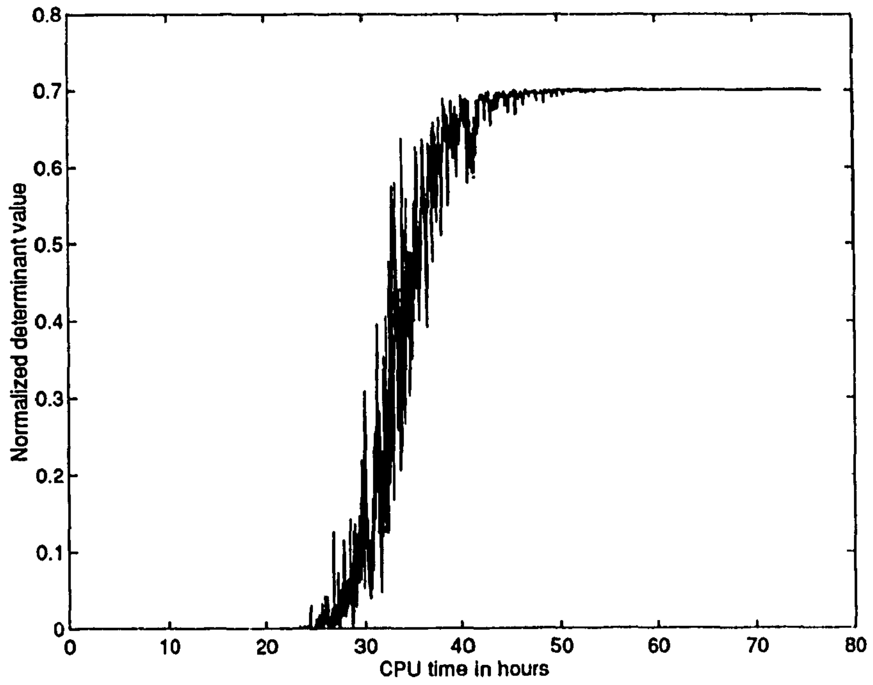


Figure 3.6: Simulated Annealing search over 256 choose 64 possibilities.

in that, the subsampled projection is transmitted instead of the KLT coefficients. Next, WS:? is discussed in details, including subjective and objective evaluations.

3.4.1 KL Transform : An Optimal Sampling Approach

Projecting the input signal onto the eigenvectors of its autocorrelation matrix is equivalent to performing the KL transform (a forward transform followed by an inverse transform) of the signal. This section reviews the basics of the KL transform with an approach to optimal sampling.

One-Dimensional Signals

Let an $N \times 1$ vector x represe..ts the input signal. The basis vectors of the KL transform are given by the orthonormalized eigenvectors of the signal autocorrelation matrix R_{xx}. The set of eigenvectors I_k and their corresponding eigenvalues λ_k

associated with R_{xx} are defined by:

$$R_{xx}I_k = \lambda_k I_k \qquad k = 1, 2, \ldots, N \tag{3.10}$$

The KL transform of x is defined as

$$y = A^T x \tag{3.11}$$

where y is the vector of transform coefficients. The columns of A are the eigenvectors of the autocorrelation matrix R_{xx}. Equation 3.11 can be written as

$$y(k) = \sum_{n=1}^{N} x(n)\phi(k, n) \qquad k = 1, 2, \ldots, N \tag{3.12}$$

where $\phi(k, n)$ is a forward transformation kernel and represents the element of A^T at row k and column n. The inverse transform that recovers the input sequence is given by

$$x = Ay \tag{3.13}$$

This can be written as

$$x(n) = \sum_{k=1}^{N} y(k)\theta(n, k) \qquad n = 1, 2, \ldots, N \tag{3.14}$$

where $\theta(n, k)$ is the inverse transformation kernel and represents the element of A at row n and column k. Ordering the eigenvalues according to their magnitude, maximum energy is compacted in the M coefficients corresponding to the eigenvectors of the M largest eigenvalues [6]. If only these M coefficients are sent, then the reconstructed signal is

$$\hat{x}(n) = \sum_{k=1}^{M} y(k)\theta(n, k) \qquad n = 1, 2, \ldots, N \tag{3.15}$$

At the receiver, the reconstruction error is given by

$$\sum_{m=M+1}^{N} \lambda_m \tag{3.16}$$

This is usually small, since only the smallest eigenvalues are included in the above summation.

Equation 3.15 can be considered as a generalized form of sampling, with a sampling function $\theta(n,k)$, where the reconstructed signal $\hat{x}$ converges to the original in the mean square sense [17], and for a given number of M terms, the mean square error in the reconstructed image is minimum among all possible sampling functions. The main difficulty in using this result for optimal sampling of images is in generating the coefficients $y(k)$ as compared to the conventional sampling of bandlimited signal discussed in Chapter 2 where the sampling function is the sinc function and the coefficients are simply the values of the signal at the sampling instants which are easy and simple to obtained.

Two-dimensional Signals

The KL transform of an $N \times N$ image x using a separable autocorrelation model is given by

$$y = A^T x A \tag{3.17}$$

where A is an $N \times N$ transform matrix and its columns are the eigenvectors of the autocorrelation matrix R_{xx} of size $N \times N$. The advantage of modeling an image autocorrelation function by a separable model [17] is that instead of solving the $N^2 \times N^2$ matrix eigenvalue problem, only two $N \times N$ eigenvalue problems need to be solved. Since an $N \times N$ matrix eigenvalue problem requires $O(N^3)$ computations, the reduction in complexity achieved using a separable model is $O(N^6)/O(N^3) = O(N^3)$ which is very significant. Image reconstruction is achieved using the inverse KL transform given by

$$x = AyA^T \tag{3.18}$$

Note that Equations 3.17 and 3.18 assume a zero mean image. Here the mean is found, substructed from the original prior to transformation, then sent with the final encoded image.

3.4.2 Uniform Subsampling

In Section 3.2.1 an approach to uniform subsampling starting with an AR(1) model was presented. The M eigenvectors (corresponding to the largest eigenvalues) of the autocorrelation matrix of this model were used to form the matrix A, defining the most important subspace of eigenvectors. It was shown that the M "most" orthogonal rows of A (these rows are also independent) yielded uniform subsampling.

Here, the projection x_p (Equation 3.2) is subsampled at M locations corresponding to the set of M most orthogonal rows of A, yielding the signal $x_{p_{ss}}$. This signal is then transmitted. At the receiver, in order to reconstruct x_p, a matrix A_{ss} is formed from the M rows of A (which is available at the receiver) corresponding to the subsampled components of x_p. A vector C of coefficients is computed from

$$C = A_{ss}^{-1} x_{p_{ss}} \tag{3.19}$$

The reconstructed signal $\hat{x}$ is given by

$$\hat{x} = AC \tag{3.20}$$

In the next example it is shown that the signal $\hat{x}$ (assuming a zero mean signal) is exactly the projection signal x_p. that is, x_p is completely determined from its M components corresponding to the set of M most orthogonal rows of A.

Uniform Subsampling: An Example

Consider the same example of an 8×8 autocorrelation matrix ($\rho = 0.95$) given in Section 3.2.1. Selecting four eigenvectors corresponding to the largest eigenvalues, and arranging them as columns in a matrix A, this matrix is computed as

$$A = \begin{bmatrix} 0.4226 & 0.4665 & -0.4809 & 0.3383 \\ -0.0854 & 0.2065 & -0.4204 & 0.3512 \\ 0.4865 & -0.1789 & -0.2860 & 0.3599 \\ -0.2783 & -0.4557 & -0.1013 & 0.3642 \\ 0.2783 & -0.4557 & 0.1013 & 0.3642 \\ 0.4865 & -0.1789 & 0.2860 & 0.3599 \\ 0.0854 & 0.2065 & 0.4204 & 0.3512 \\ -0.4226 & 0.4665 & 0.4809 & 0.3383 \end{bmatrix} \tag{3.21}$$

Consider a signal x of length $N = 8$ given by the vector

$$x = \begin{bmatrix} 5 & 5 & 6 & 5 & 4 & 5 & 6 & 7 \end{bmatrix}^T \tag{3.22}$$

Subtracting the mean value (here 5.375) from x yields

$$\overline{x} = \begin{bmatrix} -0.375 & -0.375 & 0.625 & -0.375 & -1.375 & -0.375 & 0.625 & 1.625 \end{bmatrix}^T \tag{3.23}$$

From Equation 3.2, projecting $\overline{x}$ onto the matrix A results in the signal

$$x_p = \begin{bmatrix} -0.4878 & -0.0143 & 0.1956 & -0.3221 & -0.9693 & -0.7186 & 0.5617 & 1.7576 \end{bmatrix}^T \tag{3.24}$$

Previous results in Section 3.2.1 show that the rows number: 1, 3, 5, and 7 (as well as rows number: 2, 4, 6, and 8) are the most orthogonal rows of the matrix A. Subsampling the signal x_p at the indices 1, 3, 5, and 7 results in

$$x_{p_{ss}} = \begin{bmatrix} -0.4878 & 0.1956 & -0.9693 & 0.5617 \end{bmatrix}^T \tag{3.25}$$

If $x_{p_{ss}}$ is sent, then at the receiver x_p can be completely recovered from $x_{p_{ss}}$ as follows:

A matrix A_{ss} is formed from rows number 1, 3, 5, and 7 of the matrix A. Thus,

$$A_{ss} = \begin{bmatrix} 0.4226 & 0.4665 & -0.4809 & 0.3383 \\ 0.4865 & -0.1789 & -0.2860 & 0.3599 \\ 0.2783 & -0.4557 & 0.1013 & 0.3642 \\ 0.0854 & 0.2065 & 0.4204 & 0.3512 \end{bmatrix} \tag{3.26}$$

The vector C of coefficients is computed from Equation 3.19 as

$$C = \begin{bmatrix} -1.5245 \\ 1.3876 \\ 0.9950 \\ -0.0367 \end{bmatrix} \tag{3.27}$$

Therefore, using Equation 3.20 the reconstructed signal $\tilde{x}$ (before adding the mean) is evaluated as

$$\tilde{x} = \begin{bmatrix} -0.4878 & -0.0143 & 0.1956 & -0.3221 & -0.9693 & -0.7186 & 0.5617 & 1.7576 \end{bmatrix}^T \tag{3.28}$$

Thus, adding the mean yields

$$\hat{x} = \begin{bmatrix} 4.8872 & 5.3607 & 5.5706 & 5.0529 & 4.4057 & 4.6564 & 5.9367 & 7.1326 \end{bmatrix}^T \tag{3.29}$$

Note that the same result is obtained if rows number: $2, 4, 6$, and 8 are chosen. From the above, it can be seen that the reconstructed signal $\tilde{x}$ is exactly the projection x_p.

3.4.3 Nonuniform Weighted Subsampling Using Projection (WSP)

In Section 3.2.2, more importance is placed on certain pixel locations by weighting the input signal (Equation 3.4). The autocorrelation matrix R_{uu} of the weighted signal is computed (Equation 3.6) and a matrix A_W is formed having as columns the M eigenvectors of R_{uu} corresponding to the largest eigenvalues. The projection of the weighted input signal onto the matrix A_W is given by

$$u_p = A_W {A_W}^T u \tag{3.30}$$

The sampling pattern is deduced by searching for a set of M most orthogonal rows of A_W. The signal u_p is subsampled at these corresponding M locations and then transmitted. At the receiver, a matrix $A_{W_{ss}}$ is formed using the M rows of A_W

corresponding to the M subsampled components of u_p. A vector of coefficients C_W is evaluated from

$$C_W = A_{W_{ss}}^{-1} u_{p_{ss}} \tag{3.31}$$

where $u_{p_{ss}}$ is the subsampled version of the signal u_p. The reconstruction of the weighted signal is given by

$$\hat{u} = A_W C_W \tag{3.32}$$

The signal $\hat{u}$ is exactly the projection signal u_p. An inverse weighting is performed to reconstruct an approximation of the original input signal. This is written as

$$\hat{x} = D_W{}^{-1} \hat{u} \tag{3.33}$$

The next example illustrates nonuniform weighted subsampling using projection.

Nonuniform Weighted Subsampling Using Projection: An example

Consider the signal x, of the previous example (Equation 3.22), given by the vector $x = [5,\ 5,\ 6,\ 5,\ 4,\ 5,\ 6,\ 7]^T$. Removing the mean of x leads to the signal $\overline{x}$ (see Equation 3.23). Using the weighting function given in Equation 3.7, then by weighting $\overline{x}$ (Equation 3.4) we have

$$u = \begin{bmatrix} -0.0375 & -0.0375 & 0.6250 & -0.3750 & -1.3750 & -0.3750 & 0.0625 & 0.1625 \end{bmatrix}^T \tag{3.34}$$

As in the example of Section 3.2.2, The autocorrelation matrix R_{uu} of the weighted signal (Equation 3.6) is computed using an 8×8 autocorrelation matrix R_{xx} with $\rho = 0.95$ and the weighting function of Equation 3.7. Selecting four eigenvectors of R_{uu} corresponding to the largest eigenvalues and putting them as columns in a matrix A_W, this matrix is evaluated as

$$A_W = \begin{bmatrix} 0.0268 & 0.0477 & -0.0594 & 0.0444 \\ 0.0273 & 0.0492 & -0.0622 & 0.0467 \\ 0.2691 & 0.4994 & -0.6471 & 0.4917 \\ -0.6528 & -0.4959 & -0.2718 & 0.5040 \\ 0.6528 & -0.4959 & 0.2718 & 0.5040 \\ -0.2691 & 0.4994 & 0.6471 & 0.4917 \\ -0.0273 & 0.0492 & 0.0622 & 0.0467 \\ -0.0268 & 0.0477 & 0.0594 & 0.0444 \end{bmatrix} \tag{3.35}$$

and from Equation 3.30, projecting the signal u onto the matrix A_W yields

$$u_p = \begin{bmatrix} 0.0573 & 0.0594 & 0.6068 & -0.3746 & -1.3757 & -0.3481 & -0.0312 & -0.0288 \end{bmatrix}^T \tag{3.36}$$

From the pr vious example in Section 3.2.2, it has been shown that the rows number 3, 4, 5, and 6 of the matrix A_W are orthogonal. Subsampling the signal u_p at corresponding components results in a signal

$$u_{p_{ss}} = \begin{bmatrix} 0.6068 & -0.3746 & -1.3757 & -0.3481 \end{bmatrix}^T \tag{3.37}$$

At the receiver, u_p can be recovered from the subsampled signal $u_{p_{ss}}$ as follows:

The matrix $A_{W_{ss}}$ is formed from rows number 3, 4, 5, and 6 of A_W. Thus,

$$A_{W_{ss}} = \begin{bmatrix} 0.2691 & 0.4994 & -0.6471 & 0.4917 \\ -0.6528 & -0.4959 & -0.2718 & 0.5040 \\ 0.6528 & -0.4959 & 0.2718 & 0.5040 \\ -0.2691 & 0.4994 & 0.6471 & 0.4917 \end{bmatrix} \tag{3.38}$$

Using Equation 3.31, the vector of coefficients C_W is evaluated as

$$C_W = \begin{bmatrix} -0.3918 \\ 0.9999 \\ -0.9008 \\ -0.7524 \end{bmatrix} \tag{3.39}$$

Table 3.4: Original signal (Equation 3.22) and reconstructed signals using WSP and uniform subsampling.

Original signal	5	5	6	5	4	5	6	7
WSP	5.9480	5.9690	5.9818	5.0004	3.9993	5.0269	5.0625	5.0874
Uniform subsampling	4.8872	5.3607	5.5706	5.0529	4.4057	4.6564	5.9367	7.13261

and from Equation 3.32, the reconstruction of the weighted signal is

$$\hat{u} = \begin{bmatrix} 0.0573 & 0.0594 & 0.6068 & -0.3746 & -1.3757 & -0.3481 & -0.0312 & -0.0288 \end{bmatrix}^T \quad (3.40)$$

Note that $\hat{u} = u_p$. That is, the projection u_p is exactly recovered from its subsampled version $u_{p_{ss}}$ given in Equation 3.37. The signal x is reconstructed by inverse weighting the signal $\hat{u}$ and then by adding the mean. This yields

$$\hat{x} = \begin{bmatrix} 5.9480 & 5.9690 & 5.9818 & 5.0004 & 3.9993 & 5.0269 & 5.0625 & 5.0874 \end{bmatrix}^T \quad (3.41)$$

For an easy comparison, Table 3.4 shows the original signal (Equation 3.22), with both reconstructed signals using WSP (Equation 3.41) and uniform subsampling (Equation 3.28). It can be seen in WSP that, as the weighting function places more importance on the elements number $3, 4, 5$, and 6, these elements have their values much closer to their corresponding components in the original signal than in the case of uniform subsampling. Furthermore, larger errors result in elements number $1, 2, 7$, and 8. For these elements, uniform subsampling has a lower reconstruction error than WSP.

3.4.4 Illustration of WSP: A Practical Example

An example using Lena and Girl test images illustrates WSP. Here, $N = 256$, $M = 64$, and $\rho = 0.95$. The original Lena and Girl images are shown in Figures 3.7 and 3.8 respectively. The reconstructed images using WSP and uniform subsampling (using projection) are shown in Figures 3.9 and 3.10 respectively for the Lena image, and in Figures 3.11 and 3.12 respectively for the Girl image. Both

Table 3.5: SNR for different regions of the reconstructed image for both WSP and uniform subsampling

Reconstructed image	Image region	SNR (dB)	
		WSP	Uniform subsampling
Lena	central region	24.4	22.8
	entire image	24.2	24.8
Girl	central region	29.3	26.6
	entire image	26.4	27.8

WSP and uniform subsampling use the same number of samples. It can be seen that WSP is subjectively superior to uniform subsampling at the centre of the image (the region-of-interest). As an objective measure, the peak signal-to-noise ratio (PSNR) was computed from Equation 1.1. WSP has a lower reconstruction error than uniform subsampling at the centre. Table 3.5 shows the PSNR taken over the central region, as well as over the entire reconstructed images.

3.5 Projection onto the Selected Subspace or Bandlimiting

In WSP, the projection onto the Selected Subspace of M eingenvectors (Equations 3.2 and 3.30) involves $N \times M$ operations at compression time. Since subsampling is usually thought of as a low complexity operation, this is excessive. To ease this concern a space varying linear lowpass filter (Program A.4) is used to "bandlimit" the signal. This is thought to be sufficiently close to projection to ease aliasing concerns. One of the main reasons that this is valid is that the M eigenvectors corresponding to the largest eigenvalues are those vectors having the lowest frequencies (this is generally true for R_{uu} eigenvectors as well). Thus projecting the signal onto this subspace is approximately equivalent to passing it through a low pass filter. The case of using filtering instead of projection is simply referred to as WS. Here, the complexity is reduced to $N \times l$ operations, where l is the filter

Figure 3.7: Original Lena image (256 × 256; 8 bits/pixel)

Figure 3.8: Original Girl image (256 × 256; 8 bits/pixel)

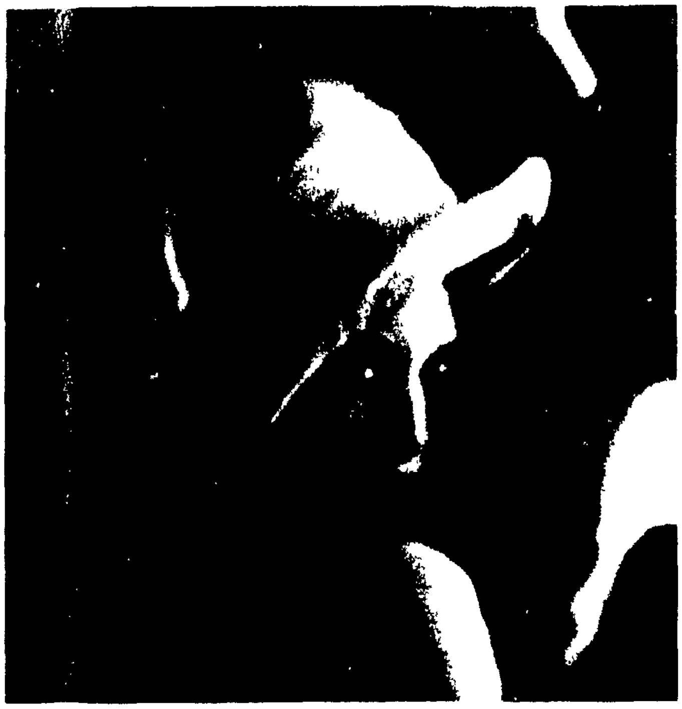


Figure 3.9: Lena compressed to 0.5 bits/pixel using WSP

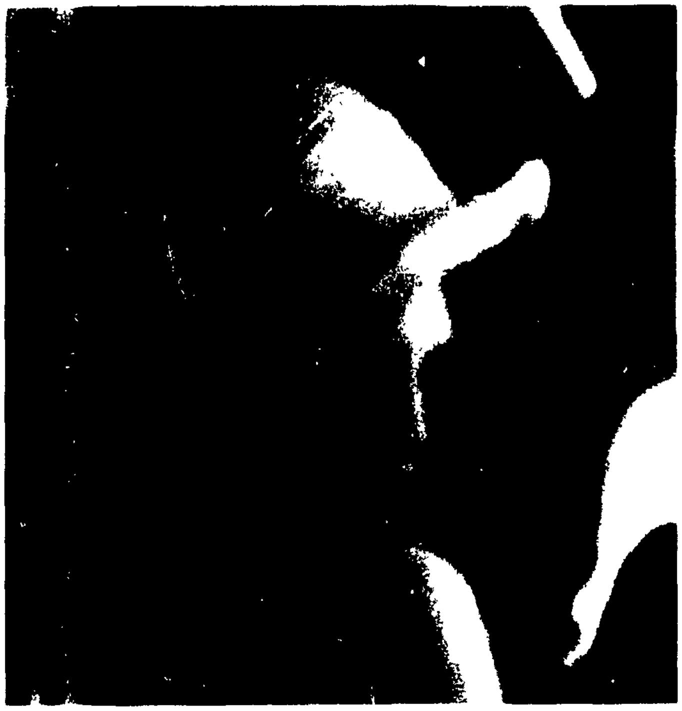


Figure 3.10: Lena compressed to 0.5 bits/pixel using uniform subsampling

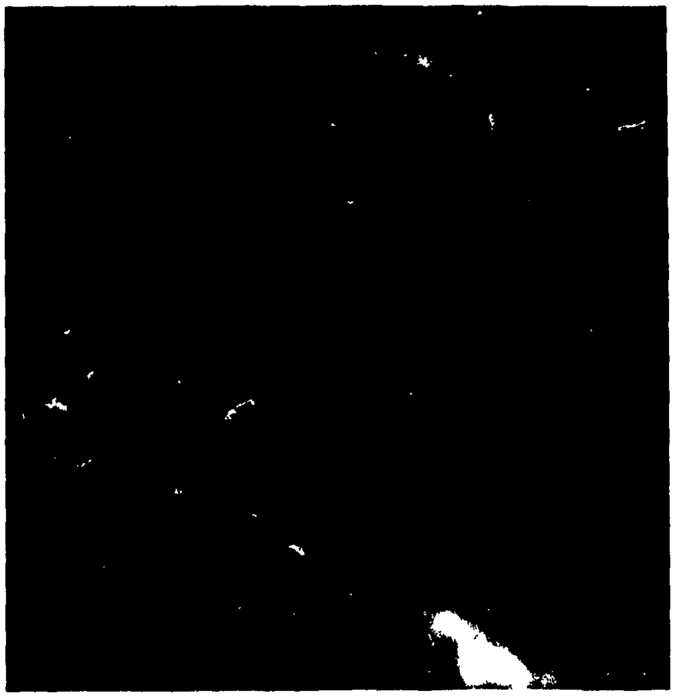


Figure 3.11: Girl compressed to 0.5 bits/pixel using WSP

Figure 3.12: Girl compressed to 0.5 bits/pixel using uniform [illegible]

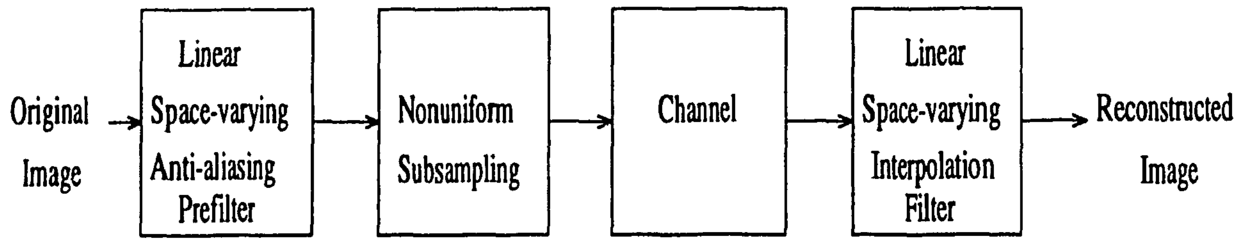

Original Image
Linear Space-varying Anti-aliasing Prefilter
Nonuniform Subsampling
Channel
Linear Space-varying Interpolation Filter
Reconstructed Image

Figure 3.13: Nonuniform Subsampling

length $(l < M)$.

A block diagram of WS is shown in Figure 3.13. It consists of a linear space varying anti-aliasing filter to bandlimit the input signal, followed by the nonuniform subsampling scheme. The subsampled image is saved, processed, or transmitted. At the receiver, an appropriate zero stuffing followed by a linear space varying interpolation filter is used for reconstruction. Next section deals with image reconstruction from its nonuniform samples.

3.6 Image Reconstruction from its Nonuniform Samples

Image reconstruction is accomplished by appropriate zero stuffing followed by space varying filtering (Program A.8). This filter uses an adjustable gain to ensure that a *dc* signal is recovered without magnitude degradation. Consider a bandlimited signal $x \in \mathcal{R}^N$ subsampled at nonuniform locations $n_s \in \mathcal{I}$, where $\mathcal{I}$ is defined as the set of all indices of the samples of x. The space varying filter uses a window w of fixed length which scans all the $\mathcal{I}$-space. For each n, a suitable cutoff frequency f_n is used depending on the largest number of consecutive zeros within the window. An adjustable gain is achieved using multipliers given by

$$\mathcal{M}(n) = \left[\sum_{q \in w} g(n_q) \frac{\sin 2\pi f_n (n - n_q)}{2\pi f_n (n - n_q)} \right]^{-1} \tag{3.42}$$

where $g(n_q)$ is a subsampled unit dc signal at locations $n_q \in n_s$. Note that the locations n_q are those of the nonuniform sampling pattern n_s within the window w. The reconstruction of the signal x is given by

$$\hat{x}(n) = \mathcal{M}(n) \sum_{q \in w} x(n_q) \frac{\sin 2\pi f_n(n - n_q)}{2\pi f_n(n - n_q)} \tag{3.43}$$

where $\hat{x}(n)$ is the reconstructed sample value and $x(n_q)$ is the subsampled value of the signal x that lie within the window. Here, the window in Equations 3.42 and 3.43 is always centered on the nth reconstruction sample.

3.7 Illustration of WS: A Practical Example

An example using the original Lena (Figure 3.7) and Girl (Figure 3.8) images illustrates the proposed WS technique. Here, $N = 256$, $M = 64$, and $\rho = 0.95$ (same values as in Section 3.4.4). The nonuniformly and uniformly subsampled Lena and Girl images are shown in Figures 3.14 and 3.15 respectively. It can be seen that the nonuniform case has more pixels taken from the centre than from the sides of the original image. The reconstructed images using WS and uniform subsampling are shown in Figures 3.16 and 3.17 respectively for the Lena image, and Figures 3.16 and 3.17 respectively for the Girl image. Both techniques use the same number of samples. It can be seen that WS is subjectively superior to the uniform subsampling at the centre of the image. In terms of objective measure, the peak signal-to-noise ratio (PSNR) was computed from Equation 1.1. WS has lower reconstruction error than uniform subsampling in the central region. Table 3.6 shows the PSNR taken over this region and over the entire image.

Comparing the results of Table 3.5 to that of Table 3.6 it can be seen that WSP is objectively superior to WS for the same compression ratio. Subjectively, images from both WS and WSP methods are quite similar. Note that WS uses filtering which, for an image with autocorrelation function modeled by an AR(1)

Table 3.6: SNR for different regions of the reconstructed image for both WS and uniform subsampling.

Compressed image	Image region	SNR (dB)	
		WS	Uniform subsampling
Lena	central region	24.2	22
	entire image	23.4	23.9
Girl	central region	29.2	25.3
	entire image	25.4	26.5

process, is a sub-optimal method compared to WSP. However, the projection in the latter involves $N \times M$ operations at subsampling time, while in WS, the filtering requires $N \times l$ operations where l is the length of the filter ($l < M$).

In the next section, lossless predictive coding is used to extend the compression achieved by WS. Suitable predictors and Huffman codes are designed and experimental results illustrate the lossless compression scheme.

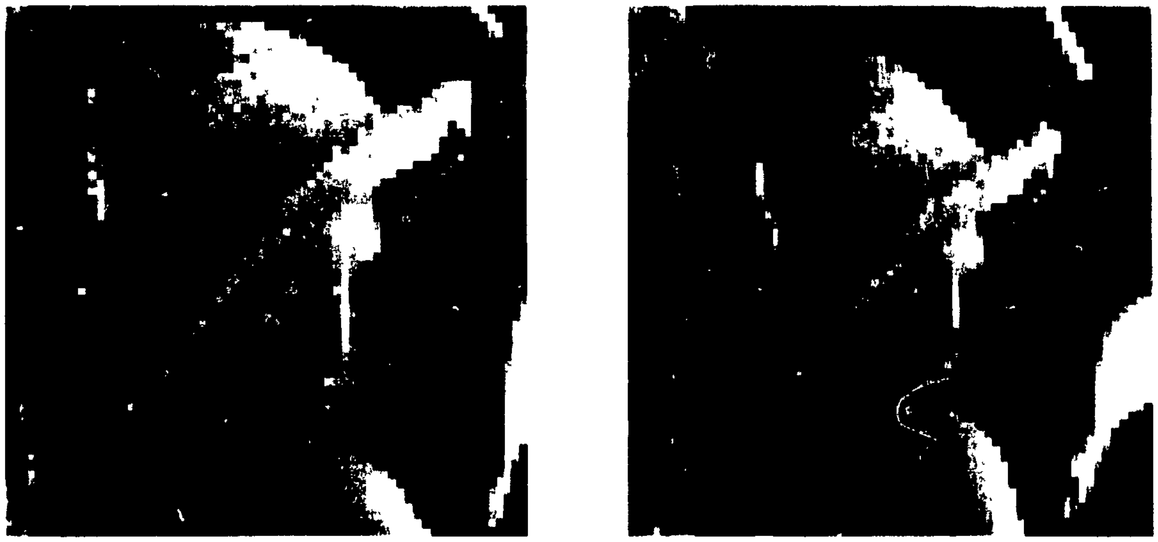


Figure 3.14: Nonuniformly and uniformly subsampled Lena respectively (64 × 64).

Figure 3.15: Nonuniformly and uniformly subsampled Girl respectively (64 × 64).

Figure 3.16: Lena compressed to 0.5 bits/pixel using WS.

Figure 3.17: Lena compressed to 0.5 bits/pixel using uniform subsampling.

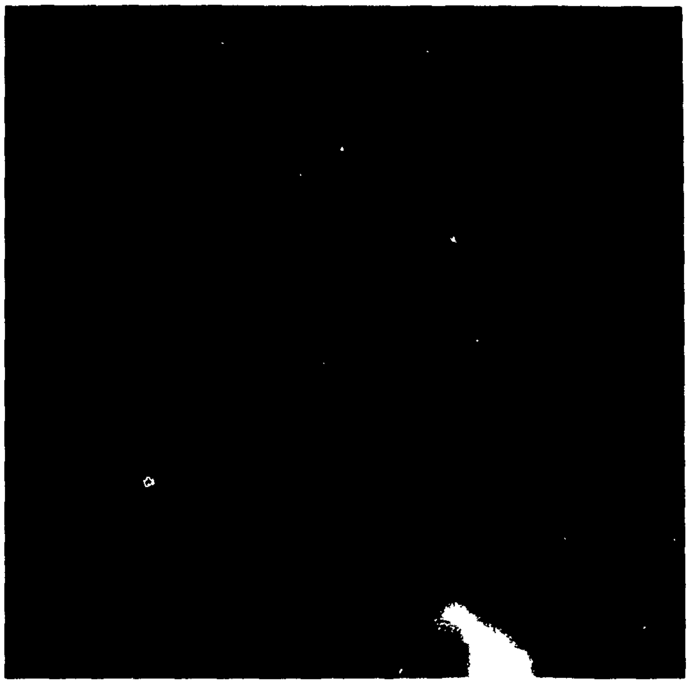


Figure 3.18: Girl compressed to 0.5 bits/pixel using WS

Figure 3.19: Girl compressed to 0.5 bits/pixel using uniform [illegible]

3.8 Lossless Compression

In the previous sections, nonuniform Weighted Subsampling (WS) was presented. In WS more important areas of the image (based on the weighting function) are sampled more finely. This section extends the compression achieved by subsampling using predictive lossless compression. Here switched prediction based on the density of the subsampling pattern is used. The prediction error sequence is coded using a suitable huffman code [27, 28] designed from a set of typical facial images. This eliminates the cost of sending the Huffman codebook. Lossless compression ratios of 1.4 are achieved yielding an overall compression ratio (including the subsampling) of 22 (0.36 bits/pixel). The lossless scheme is presented in Section 3.8.1 followed by experimental results in Section 3.8.2.

3.8.1 Lossless Predictive Coding

Most lossless image compression techniques use predictive coding to exploit the statistical redundancy expressed in the correlation property of neighboring pixels. Here we use adaptive linear prediction method [6]. Lossless techniques mainly involve two stages: a prediction stage in which pixel values are predicted, and a coding stage in which the difference between the actual pixel value and the predicted value is coded [8].

Let $x(i,j)$ be the value of the grey-scale image at row i and column j. Using a left to right top to bottom scanning strategy, the causal linear filter that predicts pixel $x(i,j)$ is:

$$x(i,j) = h_1 x(i,j-1) + h_2 x(i-1,j) + h_3 x(i-1,j-1) \tag{3.44}$$

where h_1, h_2, and h_3, are the filter coefficients. Figure 2.22 in the previous chapter shows the three neighboring pixels used in prediction of pixel $x(i,j)$. The weighting function used in WS here emphasizes the central portion of the image. This is thought to be an appropriate strategy for the coding of face portraits. This results

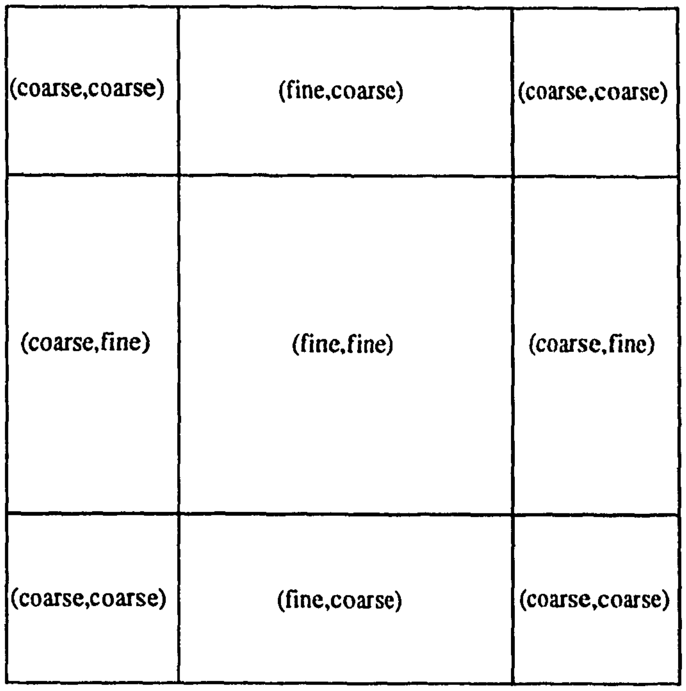


Figure 3.20: The subsampling patterns used in WS. (.,.) defines the sampling pattern in the horizontal and vertical directions respectively.

in four different types of subsampling patterns (See Figure 3.20). The lossless scheme will use a different predictive filter in each of these four areas. To determine the best filter for a particular area $\mathcal{A}$ we use the Least Squares (LS) method [40]. Specifically we form $M >> 3$ equations each of the form:

$$x(i,j) = h_1 x(i,j-1) + h_2 x(i-1,j) + h_3 x(i-1,j-1); \qquad \forall (i,j) \in \mathcal{A} \quad (3.45)$$

Putting this in matrix form:

$$\mathbf{x} = A_x \mathbf{h} \quad (3.46)$$

where $\mathbf{x}$ is a column vector containing the left hand side of Equation 3.45, A_x contains the image values used for prediction and $\mathbf{h}$ contains the predictive filter coefficients. The problem is to choose the best $\mathbf{h}$ that minimizes the error $[\mathbf{x} - A_x\mathbf{h}]$ in the least squares sense. The solution satisfies the *orthogonality principles*:

$$A_x^T(\mathbf{x} - A_x\mathbf{h}) = 0 \quad (3.47)$$

Table 3.7: Prediction filters for the subsampled Lena image (64 × 64).

Sampling Pattern	h_1	h_2	h_3
(coarse,coarse)	0.3988	0.8800	-0.2980
(coarse,fine)	0.3230	1.0089	-0.3427
(fine,coarse)	0.5365	0.5562	-0.0835
(fine,fine)	0.5169	0.8249	-0.3603

Table 3.8: Prediction filters for the subsampled Girl image (64 × 64).

Sampling Pattern	h_1	h_2	h_3
(coarse,coarse)	0.7248	0.2587	0.0194
(coarse,fine)	0.6597	0.9027	-0.5660
(fine,coarse)	0.8888	0.4324	-0.3158
(fine,fine)	0.7970	0.6603	-0.4572

The best **h** is:

$$\mathbf{h} = (A_x^T A_x)^{-1} A_x^T \mathbf{x} \tag{3.48}$$

The prediction filters are computed for each image, then sent to the receiver with the coded image. Tables 3.7 and 3.8 give the prediction filters (Program A.9) for each of the sampling patterns shown in Figure 3.20 for the subsampled Lena and Girl images respectively.

Using these filters a prediction image is constructed, it is subtracted from the original and the difference or residue is compressed using a Huffman code. Program A.10 is used to generate the prediction error sequence.

3.8.2 Experimental Results

The test images used in this section are Lena and Girl shown previously in figures 3.7 and 3.8 respectively. The prediction error images of the nonuniformly subsampled Lena and Girl using adaptive prediction are shown in Figure 3.22. These images are formed by mapping the difference between the original and the prediction images onto the range 0 – 255 so that the prediction error sequence could be displayed as an image. The prediction error is losslessly compressed using a Huffman

Table 3.9: Average bits/pixel of the compressed (256×256) image, after losslessly encoding the nonuniform subsampled images.

Coded image	average bits/pixel using a typical preset Huffman codebook
Lena	0.3632
Girl	0.3436

code. Here, a preset typical Huffman code is designed using several facial images. Thus the Huffman codebook need not be sent. Figure 3.21 shows the relation between the code length and the prediction error value. It can be seen that the code has a minimum length about zero, as the prediction error is highly peaked at that region. This shows the advantage of a variable length coding technique in assigning shorter code-words to prediction error values having high probability of occurrence so that a reduction in the average number of bits per word is achieved. Table 3.9 shows the bit rate achieved by combining WS with lossless compression. Note that the Lena image was outside the test set used to design the Huffman code. Girl was in the test set. Here the coefficients of the prediction filters have been quantized to 8 bits and their cost has been added in.

In WS, the simulated annealing is used to deduce the sampling pattern. The optimization algorithm is of very high complexity (however, it is done offline). Next chapter presents a fast heuristic, to derive the sampling pattern, which has significantly lower complexity.

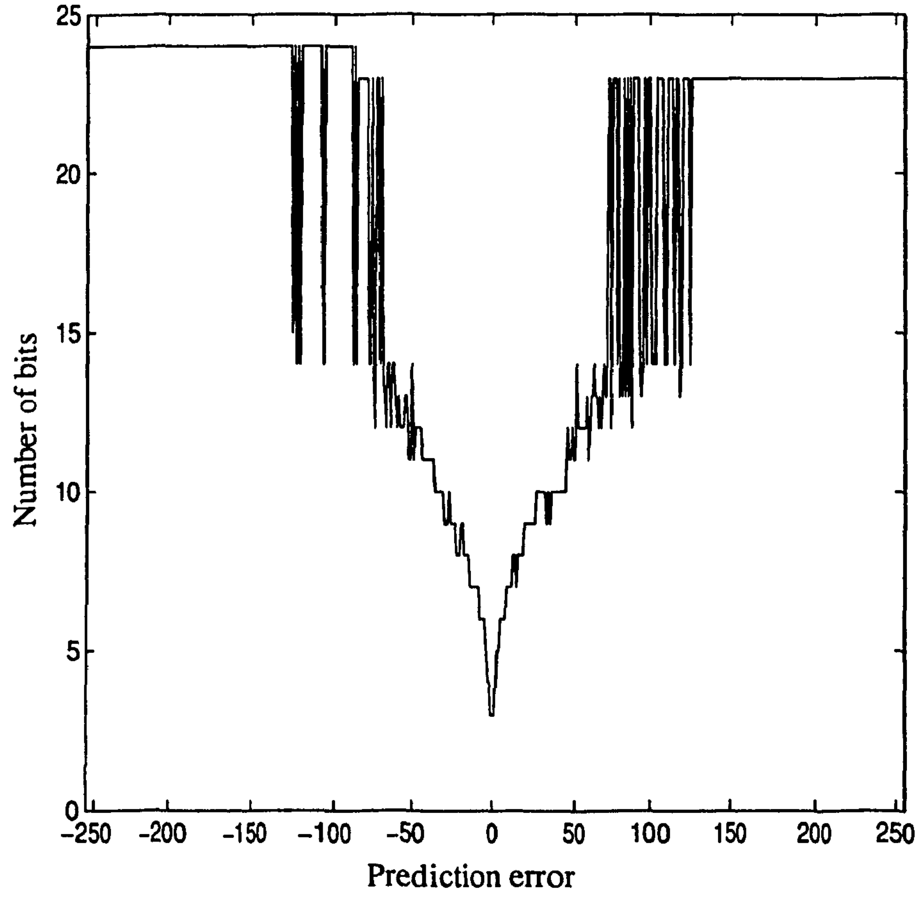


Figure 3.21: Huffman code length versus prediction error

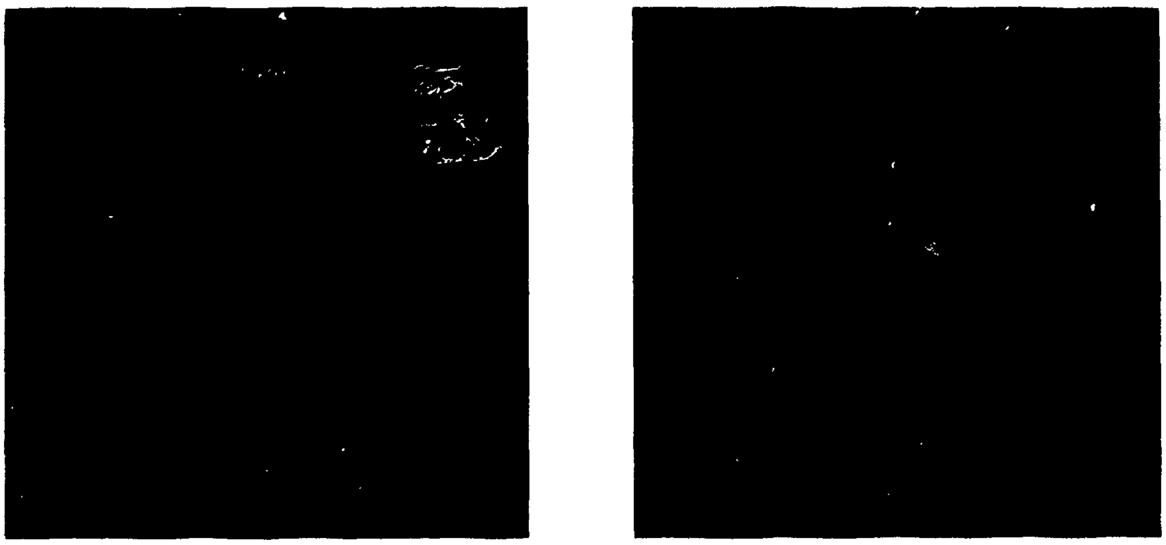


Figure 3.22: Prediction error images (64 × 64) of the nonuniformly subsampled Lena and Girl respectively.

Chapter 4

A Fast Method for Nonuniform Weighted Subsampling of Digital Images

4.1 Introduction

In Chapter 3, nonuniform Weighted Subsampling (WS) was presented [18]. In WS more important areas of the image are indicated by a weighting function and are sampled more finely. Simulated Annealing (SA) is used to derive the subsampling pattern. The SA process is done offline after the weighting function is set. However, the convergence of the algorithm takes a considerable amount of computational time. As an example, it takes approximately 60 hours CPU-time on a SPARC 5 Sun station to optimize over 256 choose 64 in order to derive the sampling pattern shown in Figure 3.5. Thus, if WS is made adaptive (which is beyond the scope of this thesis), i.e., if the system is able to locate the face and hence to assign a suitable weighting function, the simulated annealing will represent a major constraint due to its computational complexity.

This chapter presents Fast Weighted Subsampling (FWS), a fast heuristic that takes 0.45 second for the above problem. Section 4.2.1 presents the idea of the heuristic with an approach to uniform subsampling. FWS is then presented in Section 4.2.2 with various illustrative examples. Finally, subjective and objective measurements in Section 4.4 show that the proposed method gives results similar to those obtained using the SA algorithm.

4.2 Fast Weighted Subsampling (FWS)

In WS, a matrix A_w of M eigenvectors (corresponding to the M largest eigenvalues) of the autocorrelation matrix of the weighted signal was used to derive the sampling pattern. This was achieved by searching for the M "most" orthogonal rows of A_w (see Section 3.2.2). Here, a heuristic to derive the sampling pattern, by selecting the subset of M rows that reflect most of the energy of the signal, is presented.

4.2.1 Uniform Subsampling

Consider the case where M samples of x are to be retained. The autocorrelation matrix R_{xx} has its parameter ρ close to one. The matrix A is $N \times M$ and contains the M eigenvectors of R_{xx} with the largest eigenvalues and normalized to length one (see Section 3.2.1).

$$A = \begin{bmatrix} a_{1,1} & a_{1,2} & \cdot & a_{1,M} \\ a_{2,1} & a_{2,2} & \cdot & \cdot \\ \vdots & \vdots & & \cdot \\ a_{N,1} & a_{N,2} & \cdot & a_{N,M} \end{bmatrix}$$

The energy in each row is evaluated as

$$e(i) = \sum_{j=1}^{M} a_{i,j}^2 \tag{4.1}$$

Putting this in a vector $\mathcal{E} = [e(1)\, e(2)\, \ldots\, e(N)]$, this can be thought as a summation of M squared sinusoid vectors (which could be approximated by the DCT basis vectors (as $\rho \to 1$)) uniformly distributed in frequency in $(0, \pi)$ [6, 41]. In all our examples, the resulting $\mathcal{E}$-curve has M maxima. These represent the points of energy-maxima along the subset of M eigenvectors. The location of these maxima gives the required uniform sampling pattern.

4.2.2 Nonuniform Subsampling

The same heuristic is applied to the nonuniform subsampling case. The matrix A is replaced by A_W which contains the M eigenvector (with the largest eigenvalues) of R_{uu} (the autocorrelation matrix of the weighted signal). Often the $\mathcal{E}$-curve has M maxima. Their location represents the nonuniform sampling pattern. The case where the resulting $\mathcal{E}$-curve has not the required M maxima will be addressed in Section 4.3.

In terms of complexity, Equation 4.1 involves NM multiplications and additions, while finding maxima of $\mathcal{E}$ needs $2N$ operations. For a 256 choose 64 sample points, this method takes 0.45 second on a SPARC 5 Sun station compared with SA which takes approximately 60 hours to derive the sampling pattern shown in Figure 3.5. Note that the computational speed in evaluating Equation 4.1 can be improved by using a parallel structure.

We follow with some examples to illustrate the proposed method. Figure 4.1 shows the weighting functions used in this example. The plot of $\mathcal{E}$ for different M values ($M = 1,\ 4,\ 8,$ and 12) for the weighted and non-weighted cases is shown in Figure 4.2. It can be seen that the sampling patterns follow the corresponding weighting functions, in a sense that, samples are denser where the weighting function is larger. Figure 4.3 shows a plot of $\mathcal{E}$ for the weighting functions shown in Figure 4.1 with $M = 64$, $N = 256$ and $\rho = 0.95$. Each $\mathcal{E}$-curve has 64 maxima corresponding to the 64 nonuniform sample points. The sampling pattern obtained for each of the above weighting functions is shown in Figure 4.4

4.3 Loss of Spatial Resolution: A Solution

This section deals with the case where the resulting $\mathcal{E}$-curve has not the desired M maxima. We refer to this case by the term "loss of spatial resolution". This is illustrated by the following example.

Choose the weighting function shown in Figure 4.5. This function is similar to the one in Figure 4.1 (b), the only difference is that its lower values equal to 0.1 instead of 0.3. Here it is required to retain 64 out of 256 samples. The weighting function is given as

$$W(i) = \begin{cases} 0.1 & 1 \leq i \leq 75 \\ 1 & 76 \leq i \leq 181 \\ 0.1 & 182 \leq i \leq 256 \end{cases} \tag{4.2}$$

The $\mathcal{E}$-curve corresponding to this weighting function is shown in Figure 4.6 (here $\rho = 0.95$). In this case, the resulting $\mathcal{E}$-curve has only 55 maxima. The locations of

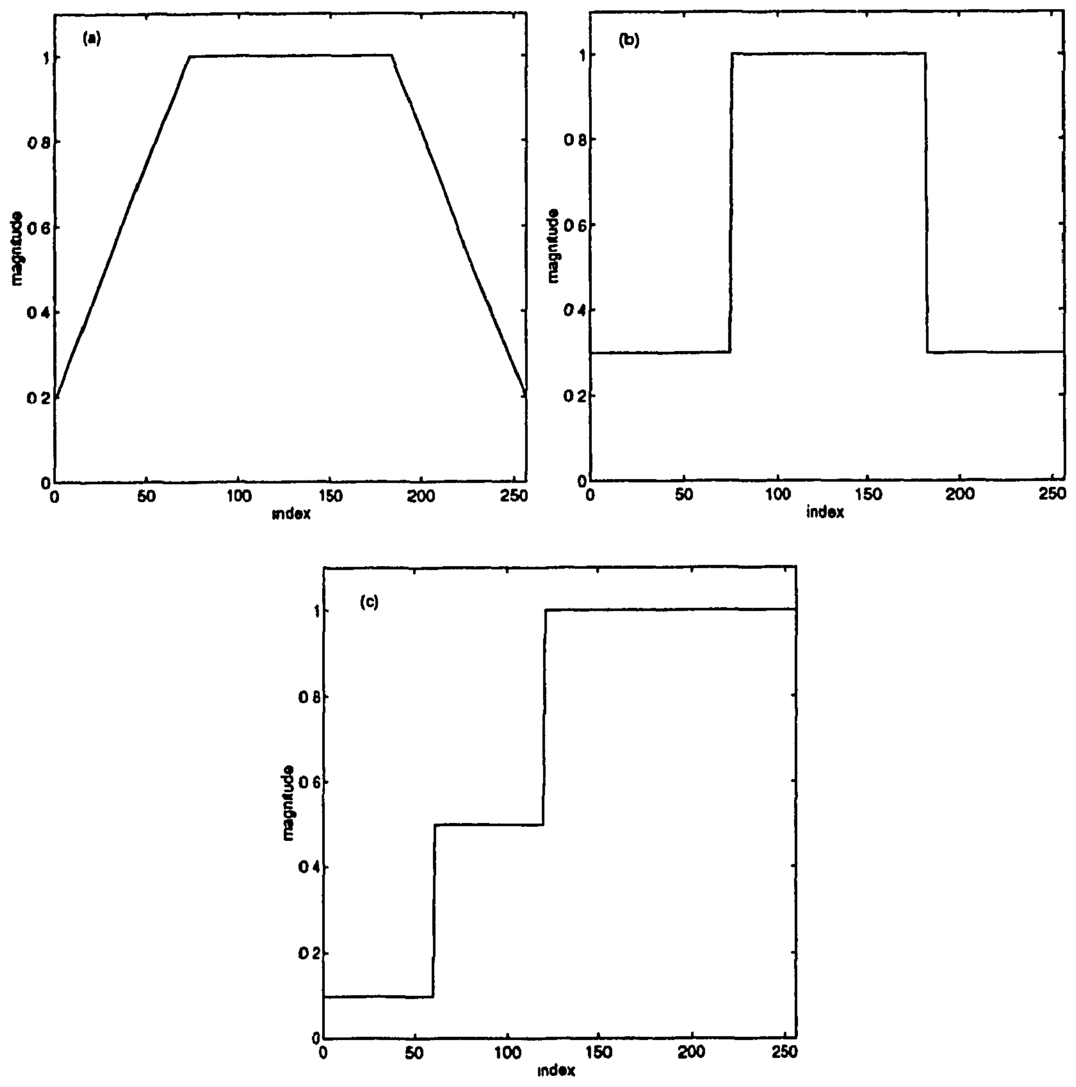


Figure 4.1: Weighting functions used to illustrate FWS.

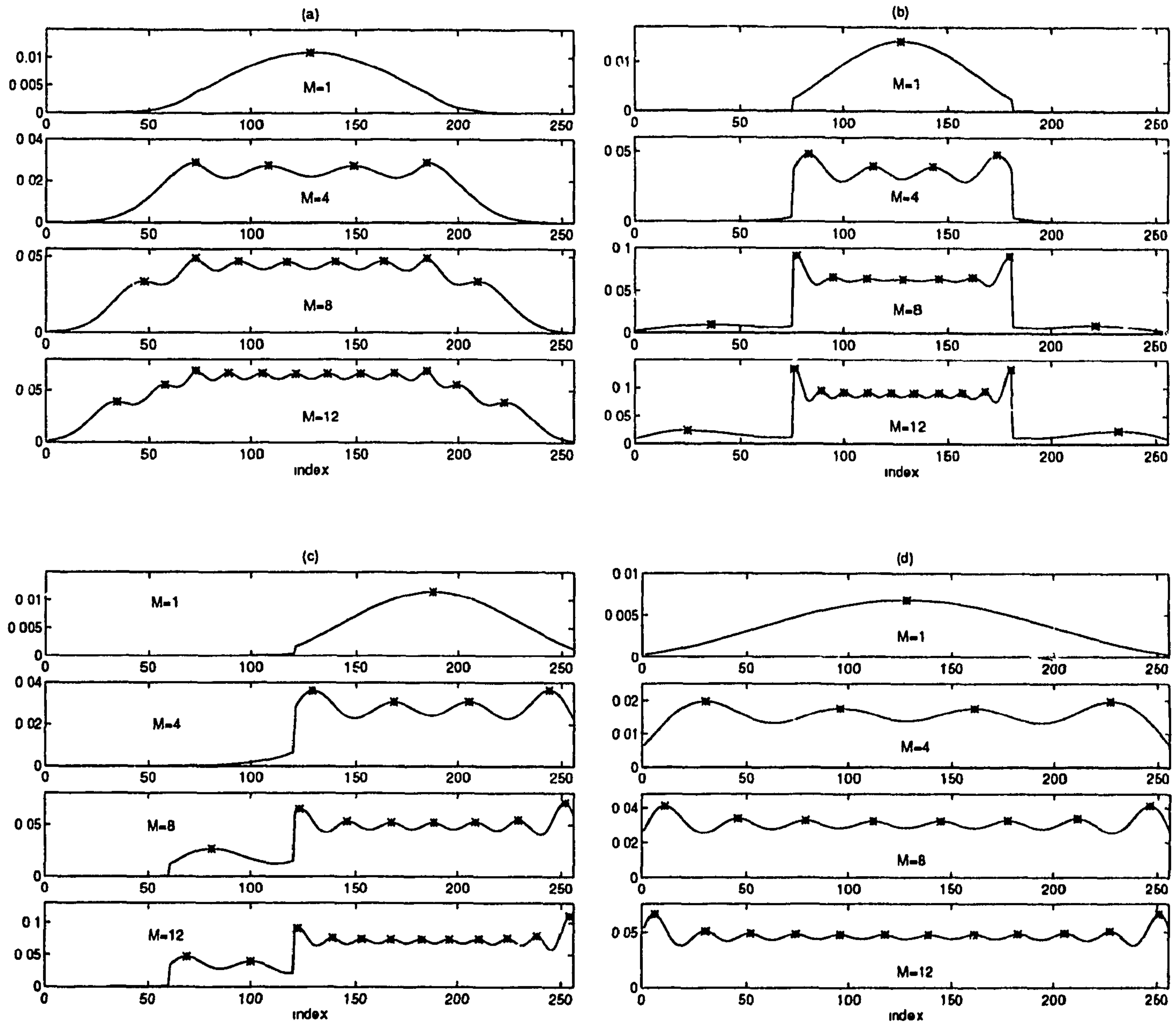


Figure 4.2: Plot of $\mathcal{E}$ for $M = 1$, 4, 8 and 12 for the weighted cases (Figure 4.2 (a), (b), and (c)) and for the non-weighted case (Figure 4.2 (d)). For the weighted cases, the corresponding weighting functions shown in Figure 4.1 (a), (b), and (c) are respectively used. The stars on the curves denote the sample points. Here $\rho = 0.95$. Note that the sample points are denser where the value of the weighting function is larger.

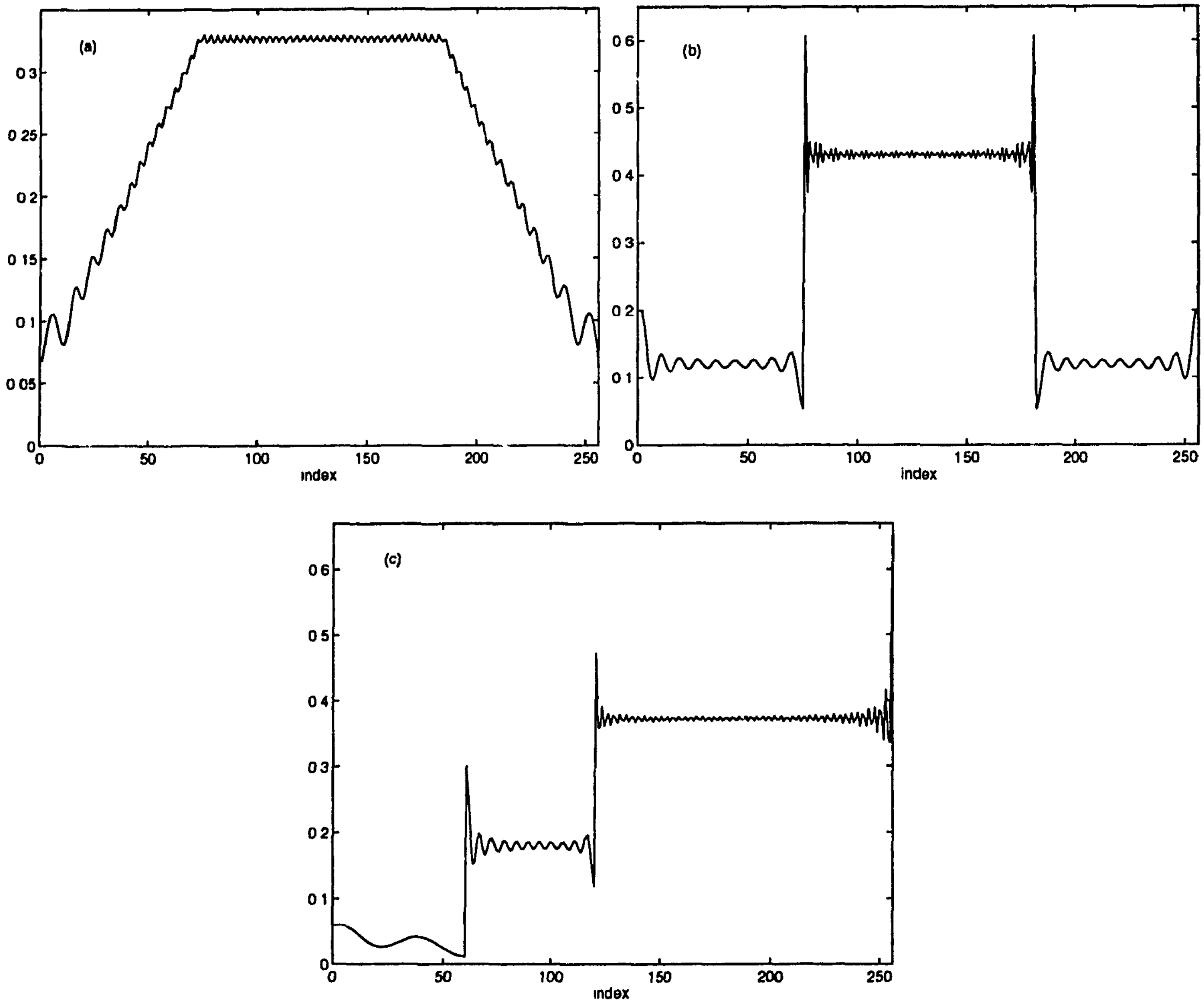


Figure 4.3: Plot of different $\mathcal{E}$-curves (M=64, N=256, ρ = 0.95). Figures 4.3 (a), (b) and (c) correspond respectively to the weighting functions shown in Figures 4.1 (a), (b) and (c). Here each curve has 64 maxima corresponding to the 64 nonuniform sample points.

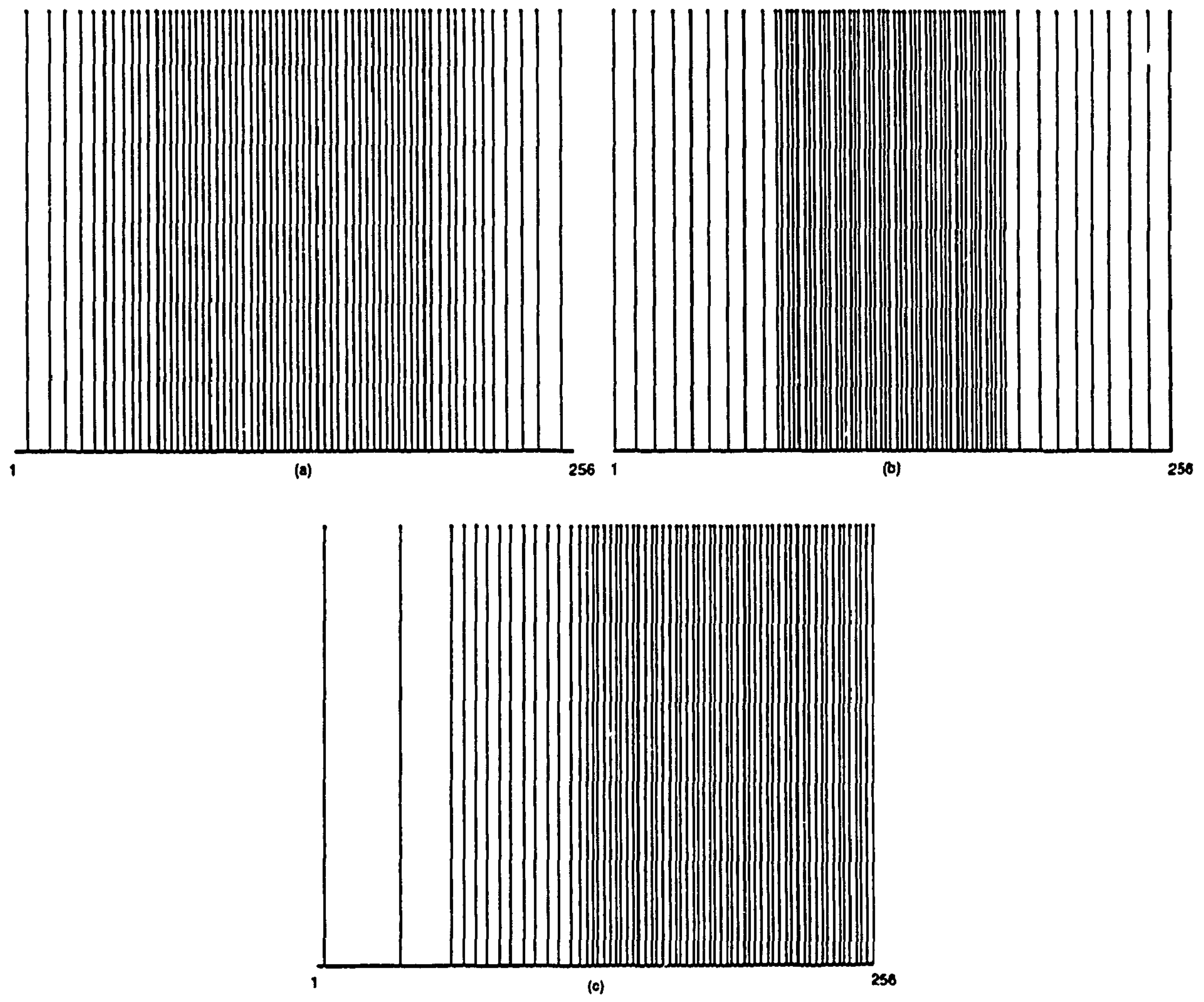


Figure 4.4: Sampling patterns deduced from the $\mathcal{E}$-curves of Figure 4.3. Here, Figures 4.4 (a), (b) and (c) correspond respectively to the weighting functions shown in Figures 4.1 (a), (b) and (c).

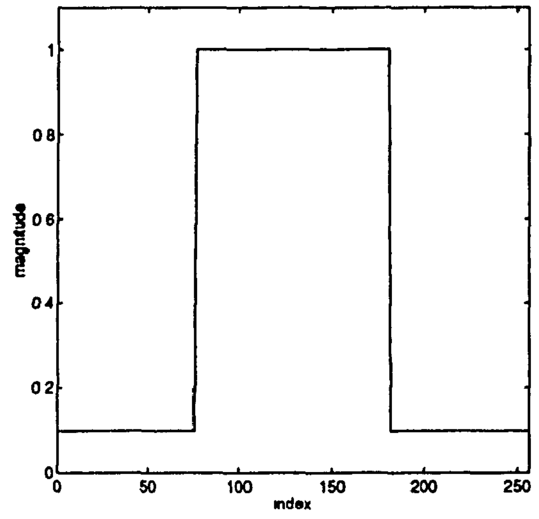


Figure 4.5: Weighting function given by Equation 4.2 and used for an $N = 256, M = 64$ problem.

these maxima are given in Table 4.1. It can be seen that, due to the lower values of the weighting function in the intervals $1 \leq i \leq 75$ and $182 \leq i \leq 256$, only few samples (here, three samples) are assigned to each of these intervals. In other words, such a weighting function assigns most of the samples to the interval $76 \leq i \leq 181$ which has a length of 106. In this interval the $\mathcal{E}$-curve can only have a number of maxima not exceeding 53. Hence, the overall $\mathcal{E}$-curve will have a number of maxima which is less than the required M samples. Comparing the resulting $\mathcal{E}$-curve to the one obtained using the weighting function of Figure 4.1 (b), it can be seen that in the latter curve (i.e., Figure 4.3 (b)) the intervals $1 \leq i \leq 75$ and $182 \leq i \leq 256$ (where the value of the weighting function is 0.3) both have 9 maxima, and therefore allowing to have the remaining maxima in the interval $76 \leq i \leq 181$. A solution to the loss of spatial resolution follows.

The weighting function in Equation 4.2 is linearly interpolated to increase its

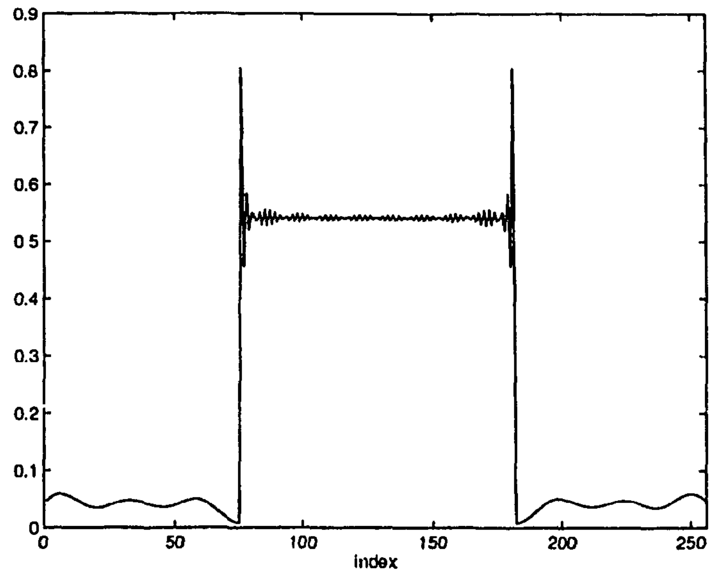


Figure 4.6: $\mathcal{E}$-curve corresponding to the weighting function given by Equation 4.2 ($\rho = 0.95$). The curve has 55 maxima.

samples from 256 to 1024 elements. The new weighting function is then given by

$$W(i) = \begin{cases} 0.1 & 1 \leq i \leq 300 \\ 1 & 301 \leq i \leq 724 \\ 0.1 & 725 \leq i \leq 1024 \end{cases} \tag{4.3}$$

Using this weighting function, a 1024×1024 autocorrelation matrix R_{uu} is computed from Equation 3.6, with a 1024×1024 matrix R_{xx} having $\rho = 0.95^{1/4}$. The problem of finding 64 out of 256 sample points is then converted to that of finding 64 out of 1024 samples. The resulting $\mathcal{E}$-curve has 64 maxima. This is shown in Figure 4.7. The locations of the maxima, which span the range $1-1024$ are then linearly mapped to the range $1-256$. A rounding to the nearest integer can be used when the mapped value is not an integer. Note that the problem of loss of spatial resolution can be solved adaptively.

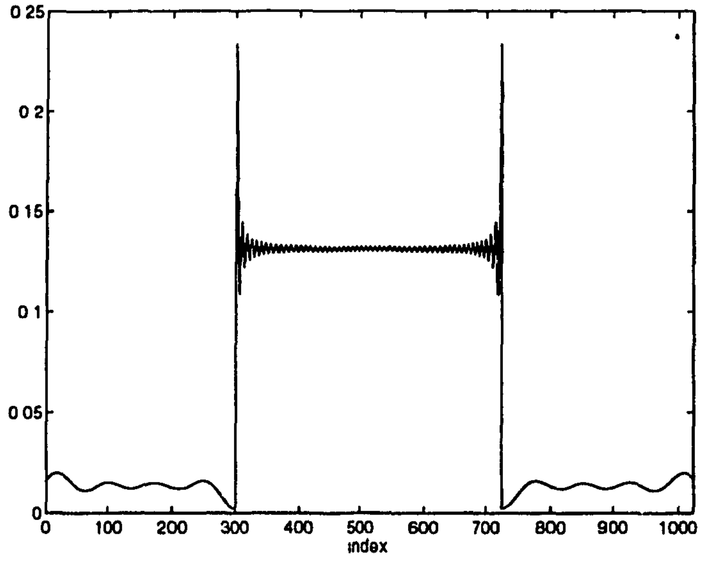


Figure 4.7: $\mathcal{E}$-curve corresponding to the weighting function given by Equation 4.3 ($\rho = 0.95^{1/4}$). The curve has 64 maxima.

Table 4.1: Locations of maxima corresponding to the $\mathcal{E}$-curve of Figure 4.6 (No. of maxima = 55).

7	33	58	76	78	80	83	85	87	89	91	94
96	98	100	102	104	107	109	111	113	115	118	120
122	124	126	128	131	133	135	137	139	142	144	146
148	150	153	155	157	159	161	163	166	168	170	172
174	177	179	181	199	224	250					

Table 4.2: Locations of maxima corresponding to the $\mathcal{E}$-curve of Figure 4.7 (No. of maxima = 64).

4	25	44	62	75	78	80	81	83	85	87	89
91	93	95	97	99	101	103	104	106	108	110	112
114	116	118	120	122	123	125	127	129	131	133	135
137	139	141	143	144	146	148	150	152	154	156	158
160	162	164	165	167	169	171	173	175	177	179	181
194	213	232	252								

Table 4.3: PSNR for different regions of the reconstructed image for WS, FWS and uniform subsampling.

Image region	PSNR (dB)		
	FWS	WS	Uniform subsampling
central region	24	24.2	22
entire image	23.3	23.4	23.9

4.4 Experimental results

An example using the Lena test image and the weighting function of Figure 4.1 (a) illustrates the proposed technique. Here the weighting function assigns more samples to the central region of the image. The sampling patterns obtained using FWS and WS are shown in Figures 4.8 and 4.9. It can be seen that both sampling patterns are similar. The reconstructed images using FWS, WS and uniform subsampling are shown in Figures 4.10, 4.11 and 4.12 respectively. All techniques use the same number of samples. It can be seen that both FWS and WS techniques are subjectively superior to uniform subsampling at the centre of the image. Further, these two techniques give similar results. In terms of objective measure, the peak signal-to-noise ratio (PSNR) [5] was calculated from equation 1.1. Both FWS and WS methods have lower reconstruction error than uniform subsampling at the centre of the image. The two methods also have very close signal to noise ratios. Table 4.3 shows the PSNR taken over different regions in the image.

4.5 Summary

In conclusion, subjective and objective measures show that the FWS heuristic gives similar results to those obtained using WS. A major advantage of this heuristic over WS is its very low complexity. Note that still an eigenvectors calculation needs to be done with complexity $O(N^3)$ for an $N \times N$ autocorrelation matrix. However in WS, the complexity of the simulated annealing dominates.

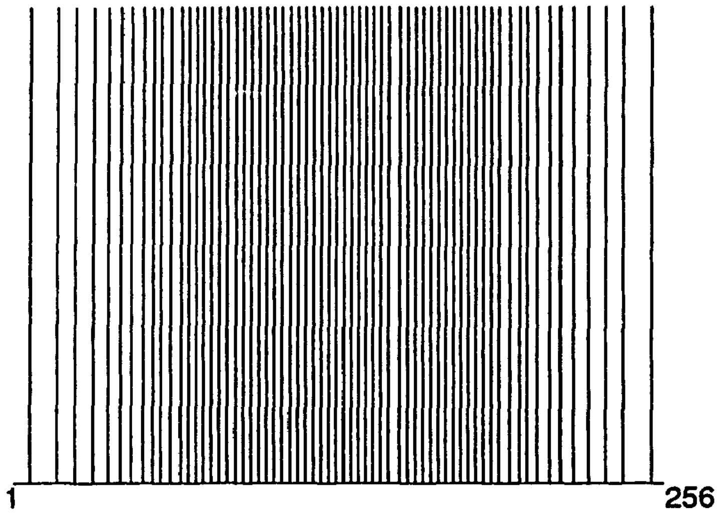


Figure 4.8: Sampling pattern obtained using FWS and corresponds to the weighting function of Figure 4.1 (a).

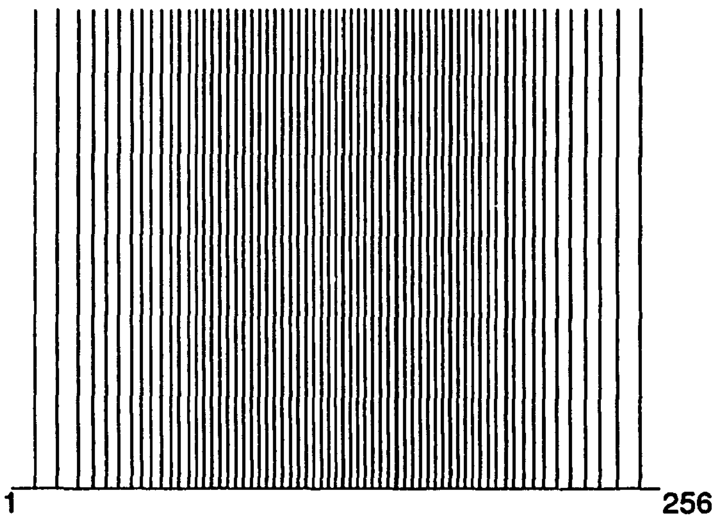


Figure 4.9: Sampling pattern obtained using WS and corresponds to the weighting function of Figure 4.1 (a).

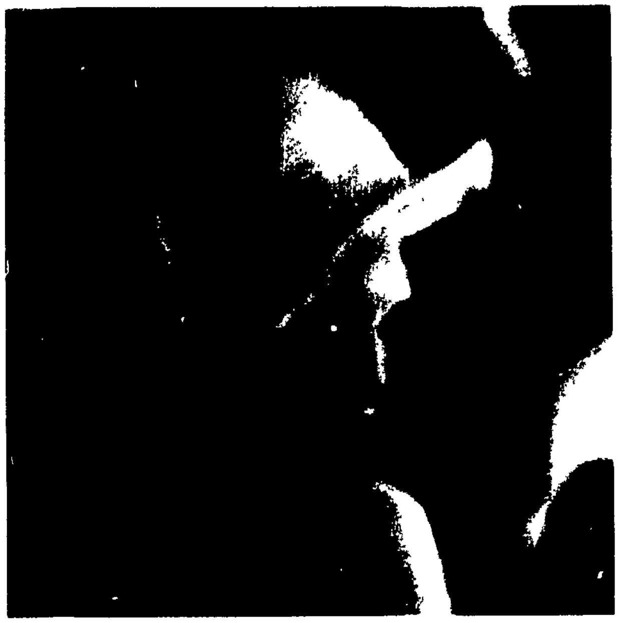


Figure 4.10: Lena (256 × 256) compressed to 0.5 bits/pixel using FWS.

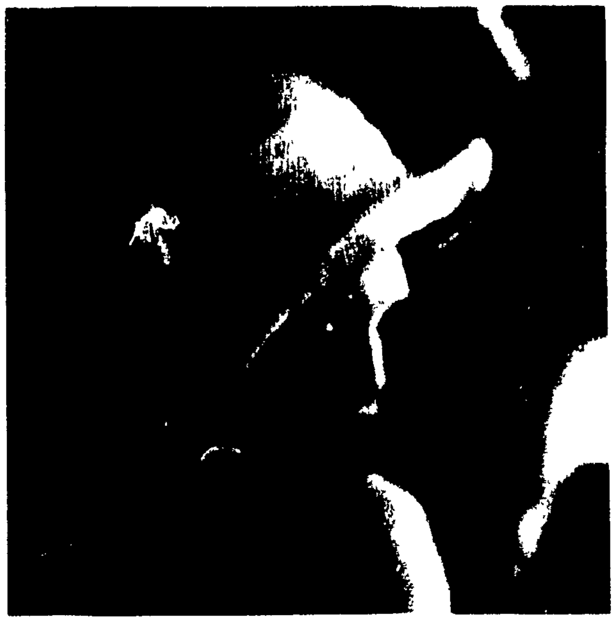


Figure 4.11: Lena (256 × 256) compressed to 0.5 bits/pixel using WS.

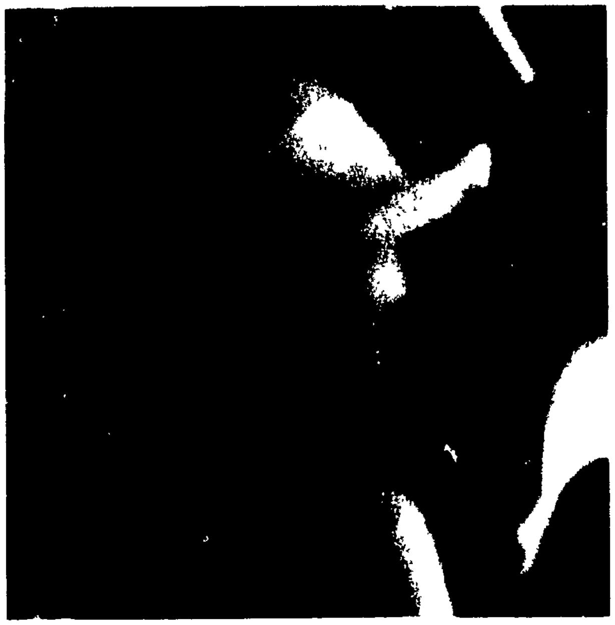


Figure 4.12: Lena (256 × 256) compressed to 0.5 bits/pixel using uniform subsampling.

Next chapter concludes the thesis by discussing the contributions of this work, and finally presents future work.

Chapter 5

Conclusion

5.1 Contributions

In Chapter 3, Weighted Subsampling (WS), a new nonuniform subsampling method suitable for "region-of-interest" applications was presented. The main idea of WS is to assign more samples (using a weighting function) to important regions of the image to enhance their quality in reconstruction. Here, WS was used to compress images of human faces. The performance of this method was evaluated and compared to uniform subsampling. Experimental results using Lena and Girl test images have shown that WS is subjectively superior to uniform subsampling at the region of interest in the image. In terms of objective measures, the peak signal-to-noise ratio was computed in this region for WS and uniform subsampling. An increase of 2.2 dB for Lena and 3.9 dB for Girl images has been achieved using WS.

Weighted Subsampling using projection (WSP) was also presented in this chapter. This technique is different from WS in that the subsampled signal is the projection of the weighted input signal, while in WS the projection is approximated using filtering. It was shown that reconstruction can be achieved using matrix manipulation and inverse weighting. WSP is superior to WS for an image with an autocorrelation function modeled by an AR(1) process. However, WSP method has a higher complexity.

In Section 3.8 lossless predictive coding was used to extend the compression achieved by WS. The prediction error of the subsampled image was obtained using different predictors based on the density of the sampling pattern. A preset typical Huffman codebook has been designed using several facial images. This codebook was used to code the prediction error sequence. A lossless compression ratio of 1.4 has been achieved yielding an overall compression ratio (including the subsampling) of 22 (0.36 bits/pixel).

In WS, the simulated annealing (SA) was used to derive the subsampling pattern. The optimization algorithm was the part of highest complexity (however, this was done offline). As an example, to optimize over 256 choose 64 has taken approximately 60 hours CPU-time on a SPARC 5 Sun station. Chapter 4 has presented Fast Weighted Subsampling (FWS), a fast heuristic which has taken 0.45 seconds for the above problem. It has been shown that FWS gives similar results to those obtained using WS method.

5.2 Future Work

Future work related to this thesis includes an adaptive nonuniform weighted subsampling scheme able to locate the human face and consequently uses a suitable weighting function. Further, improving the computational speed of the above method using parallel implementation is also another concern for real-time applications. These two are briefly discussed next.

5.2.1 Face Location

The face location problem represents an important task for the nonuniform subsampling scheme presented in this thesis. The result of combining an automatic human face location system to FWS heuristic presented in Chapter 4, is very advantageous. Such a system will throw out the a priori knowledge (about the location of the area of interest) that was needed to set the weighting function. Thus, a face location system together with a parallel implementation structure makes it possible for real-time applications. This results in various applications (see Section 3.1).

Face location represents a challenging task because of the wide variation in the appearance of a particular face due to imaging conditions such as lighting, changing in facial expression, and different viewing directions (different pose) [42]. A facial feature finder can locate the major facial features such as the eyes, nose, mouth and face outline and uses these location to geometrically mark the boundary between the face region and the background.

Many studies have addressed the problem of detecting and locating human faces. One popular approach is *template matching* or deformable templates [43, 44, 45, 46]. The general idea in template matching is the construction of an artificial template to match a corresponding feature of the face. The template consists of a collection of parametrized curves which describe the shape of the feature (such as eyes, mouth, nose, etc.) to be detected in the image. An energy function which links edges, peaks and valleys in the image intensity to corresponding properties of the template is defined. The template can adjust itself to the image by altering its parameter values to minimize the energy function. Thus the problem of fitting the template to the image is reduced to the problem of minimizing the energy function.

Another method of face location is detecting and extracting face contours and facial features using edge information [47, 48]. Lines and curves of facial features are detected using the Hough transform [49]. A major limitation of this method is the dependence on the quality of edge detection. Poor lighting conditions and low image quality result in a loss of edge information and hence may cause errors in capturing the correct face location.

An approach for detecting and locating human faces using *eigenfaces* is given in [50]. In this method, face images are projected onto a feature space called "face space". The face space is defined by the "eigenfaces", which are the eigenvectors of the covariance matrix of a training set of face images. The image is projected onto the face space and the distance between the image and its projection can be used to detect the presence of a face in a scene. The result of calculating the distance from face space at every point in the image is a "face map". A minimum in the face map gives the location of the face in the image. This technique was extended in [51] to include a facial feature extraction method using feature eigentemplates.

In summary, a good human face location system should accomplish the following tasks:

- The system should be able to handle different lighting conditions, and to distinguish faces from complex backgrounds.

- The system should have a good precision in locating facial features. It should handle different head orientations and sizes, different facial expressions and occluded faces.
- The system should handle faces with glasses or beards.
- The face location algorithm should be fast enough to make it possible for real-time applications.

The speed of the algorithm may be improved by using a parallel structure.

5.2.2 Implementation

In the Fast Weighted Subsampling (FWS) heuristic presented in Chapter 4, signal bandlimiting and interpolation can be achieved using a parallel filtering structure, that is, the operations can be done separately (in parallel) for each row, then for each column of the image. Further, Equation 4.1, which computes the energy in each of the rows of matrix A (or A_W), can be also parallelized.

Future work will also include investigation on solving the eigenvalue problem using a parallel structure. This may also allow the use of a nonseparable autocorrelation image model, and hence, a nonseparable sampling pattern.

In summary, It can be seen that an automatic face location system and a parallel implementation structure are two of major importance in realizing an adaptive nonuniform subsampling scheme suitable for real-time applications. These are left for future investigation.

Bibliography

[1] E. Dubois, "The Sampling and Reconstruction of Time-Varying Imagery with Application in Video Systems," *Proceedings of the IEEE*, vol. 73, pp. 502–522, April 1985.

[2] D. E. Dudgeon and R. M. Mersereau, *Multidimensional Digital Signal Processing.* Prentice-Hall, Inc., 1984.

[3] Y. Y. Zeevi and E. Shlomot, "Nonuniform Sampling and Antialiasing in Image Representation," *IEEE Transactions on Signal Processing*, vol. 41, pp. 1223–1236, March 1993.

[4] N. Peterfreund and Y. Y. Zeevi, "Nonuniform Image Representation in Area-of-Interest Systems," *IEEE Transactions on Image Processing*, vol. 4, pp. 1202–1212, September 1995.

[5] A. N. Netravali and B. G. Haskell, *Digital Pictures Representation Compression, and Standards.* Plenum Press, 1995.

[6] N. S. Jayant and P. Noll, *Digital Coding of Waveforms: Principles and Applications to Speech and Video.* Prentice-Hall, 1984.

[7] L. Torres-Urgell and R. L. Kirlin, "Adaptive Image Compression using Karhunen-Loeve Transform," *Signal Processing*, vol. 21, pp. 303–313, December 1990.

[8] G. K. Wallace, "The JPEG Still Picture Compression Standard," *Communications of the ACM*, vol. 34, pp. 30–44, April 1991.

[9] D. J. Legall, "MPEG: A Video Compression Standard for Multimedia Application," *Communications of the ACM*, vol. 34, pp. 47–58, April 1991.

[10] R. M. Gray, "Vector Quantization," *IEEE ASSP Magazine*, pp. 4–29, April 1984.

[11] N. M. Nasrabadi and R. A. King, "Image coding Using Vector Quantization: A Review," *IEEE Transactions on Communications*, vol. 36, pp. 957–971, August 1988.

[12] P. C. Cosman, K. L. Oehler, E. A. Riskin, and R. M. Gray, "Using Vector Quantization for Image Processin," *Proceedings of the IEEE*, vol. 81, pp. 1326-1341, September 1993.

[13] J. W. Woods and S. D. O'Neil, "Subband Coding of Images.," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. ASSP-34, pp. 1278-1288, October 1986.

[14] P. H. Westerink, J. Biemond, and D. E. Boekee, "Scalar Quantization Error Analysis for Image Subband Coding Using QMF's," *IEEE Transactions on Signal Processing*, vol. 40, pp. 421–428, February 1992.

[15] P. Sriram and M. W. Marcellin, "Image Coding Using Wavelet Transforms and Entropy-Constrained Trellis-Coded Quantization," *IEEE Transactions on Image Processing*, vol. 4, pp. 725–733, June 1995.

[16] O. Rioul and M. Vetterli, "Wavelets and Signal Processing," *IEEE Signal Processing Magazine*, vol. 8, pp. 14–38, October 1991.

[17] A. K. Jain, *Fundamentals of Digital Image Processing.* Prentice-Hall, 1989.

[18] H. Z. Sorial and W. E. Lynch, "Nonuniform Weighted Subsampling for Digital Image Compression," *Proceedings Canadian Conference on Electrical and Computer Engineering*, vol. 1, pp. 571–574, September 1995.

[19] R. M. Mersereau, "The Processing of Hexagonally Sampled Two-Dimensional Signals," *Proceedings of the IEEE*, vol. 67, pp. 930–949, June 1979.

[20] A. J. Jerri, "The Shannon Sampling Theorem-Its Various Extensions and Applications : A Tutorial Review," *Proccedings of the IEEE*, vol. 65, pp. 1565–1596, November 1977.

[21] A. V. Oppenheim and R. W. Schafer, *Discrete-Time Signal Processing.* Prentice-Hall, Englewood Cliffs, N.J., 1989.

[22] W. K. Pratt, *Digital Image Processing.* Wiley, New York, 1991.

[23] R. E. Crochiere and L. R. Rabiner, *Multirate Digital Signal Processing.* Prentice-Hall, Inc., 1983.

[24] A. Habibi, "Comparison of nth Order DPCM Encoder with Linear Transformation and Block Coding," *IEEE Transactions on Communications*, vol. 19, no. 6, pp. 948-956, 1971.

[25] J. Makhoul, "Linear Prediction: A Tutorial Review," *Proceedings of the IEEE*, vol. 63, pp. 561–580, April 1975.

[26] C. E. Shannon, "A Mathematical Theory of Communication," *Bell Systems Technical Journal*, vol. 27, pp. 379–423 and 623–656, July 1948.

[27] D. A. Huffman, "A Method for the Construction of Minimum Redundancy Codes," *Proceedings IRE*, vol. 40(9), pp. 1098–1101, September 1952.

[28] J. G. Proakis, *Digital Communications.* McGraw-Hill, 1995.

[29] G. G. Langdon, *Tutorial on Arithmetic Coding.* Research Report RJ3128, IBM Research Laboratory, San Jose, CA, 1981.

[30] J. H. Witten, R. M. Neal, and J. G. Cleary, "Arithmetic Coding for Data Compression," *Communications of the ACM*, vol. 30, pp. 520–540, June 1987.

[31] T. C. Bell, J. G. Cleary, and I. H. Witten, *Text Compression.* Prentice Hall, Inc., 1990.

[32] B. Girod and W. Geuen, "Vertical sampling rate decimation and line-offset decimation of colour difference signals," *Signal Processing*, vol. 16, pp. 109–127, February 1989.

[33] R. A. Belfor, M. P. Hesp, R. L. Lagendijk, and J. Biemond, "Spatially Adaptive Subsampling of Image Sequences," *IEEE Transactions on Image Processing*, vol. 3, pp. 492–499, September 1994.

[34] M. Tanimoto, N. Chiba, H. Yasui, and M. Murakami, "TAT (Time-Axis Transform) Bandwidth Compression System of Picture Signals," *IEEE Transactions on Communications*, vol. 36, pp. 347–354, March 1988.

[35] A. K. Jain, "Advances in Mathematical Models for Image Processing," *Proceedings of the IEEE*, vol. 69, pp. 502–528, May 1981.

[36] P. A. Wintz, "Transform Picture Coding," *Proceedings of the IEEE*, vol. 60, pp. 809–820, July 1972.

[37] H. P. Kramer, "A Generalized Sampling Theorem," *Journal of Mathematics and Physics*, vol. 38, pp. 68–72, 1959.

[38] W. H. Press, B. P. Flannery, S. A. Teukolsky, and W. T. Vetterling, *Numerical Recipes in C.* Cambridge University Press, 1988.

[39] S. Kirkpatrick, C. Gelatt, and M.Vecchi, "Optimization by Simulated Annealing," *Science*, vol. 220, pp. 671-680, 1983.

[40] G. Strang, *Linear Algebra and its Applications.* Harcourt Brace Jovanovich, Inc., 1988.

[41] A. K. Jain, "Image Data Compression: A Review," *Proceedings of the IEEE*, vol. 69, pp. 349–389, March 1981.

[42] D. J. Beymer, *Face Recognition Under Varying Pose.* Technical Report, Massachusetts Institute of Technology, Artificial Intelligence Laboratory, AI Memo No. 1461, December 1993.

[43] A. L. Yuille, D. S. Cohen, and P. W. Hallinan, "Feature Extraction from Faces Using Deformable Templates," *Proceedings IEEE Conference on Computer Vision and Pattern Recognition*, pp. 104–109, June 1989.

[44] M. A. Shackleton and W. J. Welsh, "Classification of Facial Features for Recognition," *Proceedings IEEE Conference on Computer Vision and Pattern Recognition*, pp. 573–579, June 1991.

[45] G. Chow and X. Li, "Towards a System for Automatic Facial Feature Detection," *Pattern Recognition*, vol. 26, no. 12, pp. 1739–1755, 1993.

[46] C. L. Huang and C. W. Chen, "Human Facial Feature Extraction for Face Interpretation and Recognition," *Pattern Recognition*, vol. 25, no. 12, pp. 1435-1444, 1992.

[47] X. Li and N. Roeder, "Face Contour Extraction from Front-View Images," *Pattern Recognition*, vol. 28, no. 8, pp. 1167–1179, 1995.

[48] G. Yang and T. S. Huang, "Human Face Detection in a Complex Background," *Pattern Recognition*, vol. 27, no. 1, pp. 53–63, 1994.

[49] J. Illingworth and J. Kittler, "A Survey of the Hough Transform," *Computer Vision, Graphics, and Image Processing*, vol. 44, pp. 87-116, October 1988.

[50] M. A. Turk and A. P. Pentland, "Face Recognition Using Eigenfaces," *Proceedings IEEE Conference on Computer Vision and Pattern Recognition*, pp. 586-591, June 1991.

[51] A. Pentland, B. Moghaddam, and T. Starner, *View-Based and Modular Eigenspaces for Face Recognition.* Technical Report, Massachusetts Institute of Technology, Media Laboratory, Technical Report No. 245, 1993.

Appendix A

Programs

A.1 Original Image Histogram

```
function [occ,pelval]=histogram(x,Qlevels);

% histogram  Returns the frequency of ocuurence at each grey
%            level.
%
%            [occ,pelval]=histogram(x,Qlevels) returns in vectors
%            occ and pelval the number of occurrence at each grey
%            level and the corresponding grey level respectively.
%            Input image and number of grey levels are x and
%            Qlevels respectively.
%

x=round(x);
range=Qlevels-1;
[k,l]=size(x);
%
% Evaluating for each row the frequency of occurence at each grey level
%
for i=1:k,
   for r=0:range,
        m=find(x(i,:)==r);
        num_perline(i,r+1)=length(m);
   end
end
%
% Outputs the frequency of occurence and grey level vectors
%
occ=sum(num_perline);
pelval=0:range;
%
%
```

A.2 Prediction Error Histogram

```
function [occ,diffval]=diffhist(x,Qlevels);


% diffhist    Returns the frequency of ocuurence at each
%             prediction error value.
%
%             [occ,diffval]=diffhist(x,Qlevels) returns in vectors
%             occ and diffval the number of occurrence at each
%             prediction error value and the corresponding value
%             respectively. The input prediction error sequence
%             and the number of grey levels of the original
%             image are x and Qlevels respectively.
%

x=round(x);
MAXrange=Qlevels-1;
MINrange=-Qlevels+1;
[k,l]=size(x);
%
% Evaluating for each row the frequency of occurence at each
% prediction error value
%
for i=1:k,
   for r=MINrange:MAXrange,
     m=find(x(i,:)==r);
     num_perline(i,r+abs(MINrange)+1)=length(m);
   end
end
%
% Output frequency of occurence and corresponding prediction
% error vectors
%
occ=sum(num_perline);
diffval=MINrange:MAXrange;
%
%
```

A.3 Switched Predictor

```
function [imp,imd,error,type]=imagepredict(x,row,col,Q)

% imagepredict  Returns the prediction image and its corresponding
%               prediction error sequence that yield a minimum
%               square error.
%
%               [imp,imd,error,type]=imagepredict(x,row,col,Q) returns
%               the prediction image and its corresponding prediction
%               error sequence in matrix imp and imd respectively.
%               The vector error is of length 7 and contains the mean
%               square prediction error of the seven types of predictors.
%               The predictor that yields minimum square error is
%               given in type.
%               Input image, number of rows, of columns, and number of
%               grey levels are  x, row, col, and Q respectively.
%
%
%


im_predict(1,1)=.5*R;
im_predict(1,2:col)=x(1,1:col-1);
for i=2:row,
   im_predict(i,1)=x(i-1,1);
end
%
  for i=2:row,
      im_predict(i,2:col)=x(i,1:col-1);
  end
imd1=x-im_predict;
error(1)=(1/(row*col))*sum(sum((abs(imd1)).^2));
imp1=im_predict;
%
  for i=2:row,
      im_predict(i,2:col)=x(i-1,2:col);
  end
imd2=x-im_predict;
error(2)=(1/(row*col))*sum(sum((abs(imd2)).^2));
imp2=im_predict;
%
  for i=2:row,
      im_predict(i,2:col)=x(i-1,1:col-1);
  end
imd3=x-im_predict;
error(3)=(1/(row*col))*sum(sum((abs(imd3)).^2));
imp3=im_predict;
%
```

```
   for i=2:row,
       im_predict(i,2:col)=x(i,1:col-1)+x(i-1,2:col)-x(i-1,1:col-1);
   end
imd4=x-im_predict;
error(4)=(1/(row*col))*sum(sum((abs(imd4)).^2));
imp4=im_predict;
%
   for i=2:row,
       im_predict(i,2:col)=x(i,1:col-1)+((x(i-1,2:col)-x(i-1,1:col-1))./2);
   end
imd5=x-im_predict;
error(5)=(1/(row*col))*sum(sum((abs(imd5)).^2));
imp5=im_predict;
%
   for i=2:row,
       im_predict(i,2:col)=x(i-1,2:col)+((x(i,1:col-1)-x(i-1,1:col-1))./2);
   end
imd6=x-im_predict;
error(6)=(1/(row*col))*sum(sum((abs(imd6)).^2));
imp6=im_predict;
%
   for i=2:row,
       im_predict(i,2:col)=(x(i,1:col-1)+x(i-1,2:col))./2;
   end
imd7=x-im_predict;
error(7)=(1/(row*col))*sum(sum((abs(imd7)).^2));
imp7=im_predict;
%
opt=min(error);
type=find(error==opt);
%
if type==1,imp=imp1;imd=imd1;end
if type==2,imp=imp2;imd=imd2;end
if type==3,imp=imp3;imd=imd3;end
if type==4,imp=imp4;imd=imd4;end
if type==5,imp=imp5;imd=imd5;end
if type==6,imp=imp6;imd=imd6;end
if type==7,imp=imp7;imd=imd7;end
%
%
```

A.4 Space Varying Antialiasing Prefilter

```
function y=bandlimit(x,index,row,col)

% bandlimit     Space varying antialiasing prefilter.
%
%               y=bandlimit(x,index,row,col), y is the output of
%               the space varying linear low pass filter. The input
%               image x is of size (row, col). The nonuniform sample
%               locations are given in vector index.
%

I=[0 index col+1];
len=length(I);
for i=2:len;
  M(i)=abs(I(i)-I(i-1)-1);
end
range=max(M)+2;
z=zeros(1,range);
A=ones(1,col);
dc=ones(1,col);
a=zeros(1,col);
a(1,index)=dc(1,index);
%
AA=[z A z];
aa=[z a z];
Z=zeros(row,range);
XZ=[Z x Z];
%
for i=1:col
%
%
    if i>range & i<col-range+1
       m=find(index >(i-range-1)&index<(i+range+1));
        t=[a(1,i-range:i+range) 1];
       count=0;
       s=[];
       for k=1:2*range+2,
           if t(k)==0
              count=count+1;
           else
              s(k)=count;
              count=0;
           end
       end
       Wc=1/(max(s+1));
       B=fir1(2*range,Wc);
       x_prefilt(:,i)=x(:,i-range:i+range)*B';
    end
```

```
%
%
    if (i>0 & i<range+1)|(i>col-range & i<col+1)
        m=find(index>(i-range-1)&index<(i+range+1));
       tt=[aa(1,i:i+2*range) 1];
       count=0;
       s=[];
       for k=1:2*range+2,
           if tt(k)==0
              count=count+1;
           else
              s(k)=count;
              count=0;
           end
       end
       Wc=1/(max(s+1));
       B=fir1(2*range,Wc);
       mult=AA(1,i:i+2*range)*B';
       x_prefilt(:,i)=XZ(:,i:i+2*range)*B'/mult;
    end
end
%
y=x_prefilt;
```

A.5 Full Search of Example 3.2.2

```
% Rxx of an AR(1) process of size 8x8
%
ro=.95;
for k=1:8,
    for l=1:8,
      rxx(k,l)=ro^abs(k-l);
    end
end
%
% Weighting the signal and computing eigenvectors
% of the new autocorrelation matrix
%
f=[0.1 0.1 1 1 1 1 0.1 0.1];
k=diag(f);
ruu=k*rxx*k;
[vecruu,valruu]=eig(ruu);
%
% taking 4 eigenvectors with largest eigenvalues
%
auu(:,1:4)=vecruu(:,5:8);
%
% Normalization of rows of auu
%
for i=1:8,
    auunorm(i,:)=auu(i,:)./norm(auu(i,:));
end
%
% Search for the most orthogonal 4 rows of auu
%
s=8; n=4; t=s-n+1;
temp=0; count=0; time=0;
%
% temp shows how orthogonal is the set of rows.
% count gives the No. of determinant computations
% time gives the cputime.
%
time = cputime; for i1=1:t,
for i2=(i1+1):(t+1),
 for i3=(i2+1):(t+2),
  for i4=(i3+1):(t+3),
    m=[auunorm(i1,:);auunorm(i2,:);auunorm(i3,:);auunorm(i4,:)];
    d=det(m);
    count = count+1;
    if abs(d) > temp
       temp = abs(d);
       mostorth=[auu(i1,:); auu(i2,:); auu(i3,:); auu(i4,:)];
       rowauu=[i1;i2;i3;i4];
    end
   end
  end
 end
end; time = cputime-time, count, temp, rowauu,
```

A.6 Main Program of WS

```
%Rxx of an AR(1) process of size 256x256.
%
row=256; col=256; rho=.95; M=64;
for k=1:row,
    for l=1:col,
        rxx(k,l)=rho^abs(k-l);
    end
end
%
% Weighting the signal and computing eigenvectors
% of the new autocorrelation matrix
%
load Weight_func
k=diag(Weight_func); ruu=k*rxx*k; [vecruu,valruu]=eig(ruu);
%
%Taking M eigenvectors with the largest eigenvalues
%
diagval=diag(valruu);
if diagval(1) < diagval(col)
   auu(:,1:M)=vecruu(:,1:M);
else
   auu(:,1:M)=vecruu(:,col-M+1);
end
%
% Random permutation of rows of auu
%
rand('seed',sum(100*clock))
rnd = randperm(row);
for j=1:row,
    auu_random(j,:) = auu(rnd(j),:) ;
end
index=rnd(1:M);
%
%Normalization of rows of auu_random
%
for i=1:row,
    auunorm_random(i,:)=auu_random(i,:)./norm(auu_random(i,:));
    auunorm_init(i,:)=auu(i,:)./norm(auu(i,:));
end
%
%Search for M most orthogonal rows of auu_random
%
T=10; alpha=.9; v=auunorm_random(1:M,:); vuu=auu_random(1:M,:);
dold = abs(det(v)); count_success=0; count_reconfig=0;
dsuccess_reconfig=0; store_high=0; store_orthog=0; store_index=0;
m=v; time = cputime; SA_64_256;  dval=dd; store_time=interval_time;
%
```

```
% Stopping criterion
%
while  store_high < .9999
       SA_64_256;
       dval=[dval dd]; store_time=[store_time interval_time];
       save main_64_256 dval store_index store_high
end
time = cputime - time, temp = store_high
mostorth=store_orthog, row_index = sort(store_index),
%
%
```

A.7 Simulated Annealing Algorithm

```
% SA_64_256
%
%
dd=0; interval_time=0;
for k=1:M,
    l=randperm(N);
 for j= 1:N,
  if (l(j)==index(1))|(l(j)==index(2))|(l(j)==index(3))|(l(j)==index(4))|...
  (l(j)==index(5))|(l(j)==index(6))|(l(j)==index(7))|(l(j)==index(8))|...
  (l(j)==index(9))|(l(j)==index(10))|(l(j)==index(11))|(l(j)==index(12))|...
  (l(j)==index(13))|(l(j)==index(14))|(l(j)==index(15))|(l(j)==index(16))|...
  (l(j)==index(17))|(l(j)==index(18))|(l(j)==index(19))|(l(j)==index(20))|...
  (l(j)==index(21))|(l(j)==index(22))|(l(j)==index(23))|(l(j)==index(24))|...
  (l(j)==index(25))|(l(j)==index(26))|(l(j)==index(27))|(l(j)==index(28))|...
  (l(j)==index(29))|(l(j)==index(30))|(l(j)==index(31))|(l(j)==index(32))|...
  (l(j)==index(33))|(l(j)==index(34))|(l(j)==index(35))|(l(j)==index(36))|...
  (l(j)==index(37))|(l(j)==index(38))|(l(j)==index(39))|(l(j)==index(40))|...
  (l(j)==index(41))|(l(j)==index(42))|(l(j)==index(43))|(l(j)==index(44))|...
  (l(j)==index(45))|(l(j)==index(46))|(l(j)==index(47))|(l(j)==index(48))|...
  (l(j)==index(49))|(l(j)==index(50))|(l(j)==index(51))|(l(j)==index(52))|...
  (l(j)==index(53))|(l(j)==index(54))|(l(j)==index(55))|(l(j)==index(56))|...
  (l(j)==index(57))|(l(j)==index(58))|(l(j)==index(59))|(l(j)==index(60))|...
  (l(j)==index(61))|(l(j)==index(62))|(l(j)==index(63))|(l(j)==index(64));
   m=v;
 else
    m(k,:)=auunorm_init(l(j),:);
    dnew= abs(det(m));
    if dnew > dold
       v=m; dold=dnew; vuu(k,:)=auu(l(j),:);
       index(k)=l(j);
    else
       y=exp((dnew - dold)/((dnew + dold)*T));
       r=rand(1);
       if r < y
          v=m; dold=dnew; vuu(k,:)=auu(l(j),:);
          index(k)=l(j);
       else
          m=v;
       end
    end
%
%counting reconfigurations
%
          count_reconfig = count_reconfig + 1;
%
%
%counting successfull reconfigurations
```

```
%
    if dold > dsuccess_reconfig
       count_success = count_success + 1;
       dsuccess_reconfig = dold;
    end
%
%
% Holding T constant for 6400 reconfigurations or for
% 640 successful reconfigurations, whichever comes first.
%
    if (count_success == 640) | (count_reconfig == 6400)
       T = alpha*T;
       count_success = 0; count_reconfig = 0;
    end
%
  end;
%
% Store always the highest value of determinant
% (dold) and its corresponding matrix.
%
    if store_high < dold
       store_high = dold; store_orthog = vuu;
       store_index = index;
    end
%
 end; dd(k)=dold; interval_time(k)=cputime-time;
%
end;
%
%
```

A.8 Space Varying Interpolation Filter

```
function y=reconstruct(x,index,row,col)

% reconstruct   Space varying interpolation filter.
%
%               y=reconstruct(x,index,row,col), y is the reconstructed
%               image . The input x is the subsampled image stuffed with
%               zeros and  of size (row x col). The sample locations are
%               given in vector index.
%

I=[0 index col+1];
len=length(I);
for i=2:len;
  M(i)=abs(I(i)-I(i-1)-1);
end
range=max(M)+2;
%
%
% Subsample a Unit DC
%
%
dc=ones(1,col);
A=zeros(1,col);
A(1,index)=dc(1,index);
%
%
z=zeros(1,range);
AA=[z A z];
Z=zeros(row,range);
XZ=[Z x Z];
%
% Find suitable cutoff frequency and do interpolation
%
for i=1:col
    if i>range & i<col-range+1
       m=find(index >(i-range-1)&index<(i+range+1));
       t=[A(1,i-range:i+range) 1];
       count=0;
       s=[];
       for k=1:2*range+2,
           if t(k)==0
              count=count+1;
           else
              s(k)=count;
              count=0;
           end
       end
       Wc=1/(max(s+1));
%       active_pel=length(m);
%       Wc=1/((2*range +1)/active_pel);
       B=fir1(2*range,Wc);
       mult=A(1,i-range:i+range)*B';
       x_recov(:,i)=x(:,i-range:i+range)*B'/mult;
    end
%
%
%
    if (i>0 & i<range+1)|(i>col-range & i<col+1)
        m=find(index>(i-range-1)&index<(i+range+1));
       tt=[AA(1,i:i+2*range) 1];
       count=0;
       s=[];
       for k=1:2*range+2,
           if tt(k)==0
              count=count+1;
           else
              s(k)=count;
              count=0;
           end
       end
       Wc=1/(max(s+1));
%        active_pel=length(m);
%        Wc=1/((2*range +1)/active_pel);
        B=fir1(2*range,Wc);
        mult=AA(1,i:i+2*range)*B';
        x_recov(:,i)=XZ(:,i:i+2*range)*B'/mult;
    end
end
%
y=x_recov;
```

A.9 Prediction Filters using LS Method

```
function [coeff1,coeff2,coeff3,coeff4]=least_square(x,row,col,r1,r2,c1,c2)

%least_square  Output the prediction filters for different areas in the
%              image.
%
%              [H1,H2,H3,H4]=least_square(x,row,col,r1,r2,c1,c2) output
%              filter coefficients for different areas determined by
%              the rows and columns r1, r2, c1 and c2. The input image
%              is x of size (row x col). H1, H2, H3, and H4 are vectors
%              each of length 3 and containing the filter coefficients.
%

% Find filter coefficients for the sampling
% pattern (coarse,coarse)
%
temp2=[];
tempx2=[];
for i=r2:row,
    for j=c2:col,
temp=[x(i-1,j-1) x(i-1,j) x(i,j-1)];
tempx=x(i,j);
temp2=[temp2;temp];
tempx2=[tempx2;tempx];
end
end
y1=temp2;xx1=tempx2;
m1=inv(y1'*y1);
n1=y1'*xx1;
coeff1=m1*n1;
%
% Find filter coefficients for the sampling
% pattern (coarse,fine)
%
temp2=[];
tempx2=[];
for i=r1:r2,
    for j=c2:col,
temp=[x(i-1,j-1) x(i-1,j) x(i,j-1)];
tempx=x(i,j);
temp2=[temp2;temp];
tempx2=[tempx2;tempx];
end
end
y2=temp2;xx2=tempx2;
m2=inv(y2'*y2);
n2=y2'*xx2;
```

```
coeff2=m2*n2;
%
% Find filter coefficients for the sampling
% pattern (fine,coarse)
%
temp2=[];
tempx2=[];
for i=r2:row,
    for j=c1:c2,
temp=[x(i-1,j-1) x(i-1,j) x(i,j-1)];
tempx=x(i,j);
temp2=[temp2;temp];
tempx2=[tempx2;tempx];
end
end
y3=temp2;xx3=tempx2;
m3=inv(y3'*y3);
n3=y3'*xx3;
coeff3=m3*n3;
%
% Find filter coefficients for the sampling
% pattern (fine,fine)
%
temp2=[];
tempx2=[];
for i=r1:r2,
    for j=c1:c2,
temp=[x(i-1,j-1) x(i-1,j) x(i,j-1)];
tempx=x(i,j);
temp2=[temp2;temp];
tempx2=[tempx2;tempx];
end
end
y4=temp2;xx4=tempx2;
m4=inv(y4'*y4);
n4=y4'*xx4;
coeff4=m4*n4;
%
%
```

A.10 Prediction Error Sequence using LS Method

```
function [imp,imd,MSE]=lspredict(x,row,col,Q,H1,H2,H3,H4,r1,r2,c1,c2)

% lspredict Returns the prediction error sequence using the suitable
%           predictor for each region (determined using LS method).
%
%           [imp,imd,MSE]=lspredict(x,row,col,Q,H1,H2,H3,H4,r1,r2,c1,c2)
%           output predicted image and prediction error sequence in imp
%           and imd respectively.
%           The mean square error of the prediction is given in MSE.
%           The input image is x of size (row,col). Q is the number of
%           grey levels. H1, H2, H3, and H4 are the filters
%           obtained using the Least Square method. Areas of different
%           sampling pattern are determined by r1, r2, c1, and c2.

im_predict(1,1)=.5*Q;
im_predict(1,2:col)=x(1,1:col-1);
for i=2:row,
    im_predict(i,1)=x(i-1,1);
end
%
for i=2:r1,
  for j=2:c1,
      im_predict(i,j)=H1(3)*x(i,j-1)+H1(2)*x(i-1,j)...
                     +H1(1)*x(i-1,j-1);
  end
end
%
for i=2:r1,
  for j=c1+1:c2,
      im_predict(i,j)=H3(3)*x(i,j-1)+H3(2)*x(i-1,j)...
                     +H3(1)*x(i-1,j-1);
  end
end
%
for i=2:r1,
  for j=c2+1:col,
      im_predict(i,j)=H1(3)*x(i,j-1)+H1(2)*x(i-1,j)...
                     +H1(1)*x(i-1,j-1);
  end
end
%
for i=r1+1:r2,
  for j=2:c1,
      im_predict(i,j)=H2(3)*x(i,j-1)+H2(2)*x(i-1,j)...
                     +H2(1)*x(i-1,j-1);
  end
end
%
for i=r1+1:r2,
  for j=c1+1:c2,
      im_predict(i,j)=H4(3)*x(i,j-1)+H4(2)*x(i-1,j)...
                        +H4(1)*x(i-1,j-1);
  end
end
%
for i=r1+1:r2,
  for j=c2+1:col,
      im_predict(i,j)=H2(3)*x(i,j-1)+H2(2)*x(i-1,j)...
                        +H2(1)*x(i-1,j-1);
  end
end
%
for i=r2+1:row,
  for j=2:c1,
      im_predict(i,j)=H1(3)*x(i,j-1)+H1(2)*x(i-1,j)...
                        +H1(1)*x(i-1,j-1);
  end
end
%
for i=r2+1:row,
  for j=c1+1:c2,
      im_predict(i,j)=H3(3)*x(i,j-1)+H3(2)*x(i-1,j)...
                        +H3(1)*x(i-1,j-1);
  end
end
%
for i=r2+1:row,
  for j=c2+1:col,
      im_predict(i,j)=H1(3)*x(i,j-1)+H1(2)*x(i-1,j)...
                        +H1(1)*x(i-1,j-1);
  end
end
%
imp=im_predict;
imd=x-im_predict;
MSE=(1/(row*col))*sum(sum((abs(imd)).^2));
%
%
```