PROGRESSIVE SOURCE CODING BY MATCHING PURSUIT

PROGRESSIVE SOURCE CODING BY MATCHING PURSUIT: APPLICATION IN IMAGE AND GAUSSIAN DATA COMPRESSION

By

ALIREZA SHOA, M.A.Sc., 2004 (ECE Dept.)
McMaster University
Hamilton, Canada

A Thesis
Submitted to the School of Graduate Studies
in Partial Fulfillment of the Requirements
for the Degree
Doctor of Philosophy

McMaster University

DOCTOR OF PHILOSOPHY (2009) MCMASTER UNIVERSITY

(Electrical and Computer Engineering) Hamilton, Ontario

TITLE: **Progressive Source Coding by Matching Pursuit: Application in Image and Gaussian Data Compression**

AUTHOR: Alireza Shoa

M.A.Sc., 2004 (ECE Dept.)

McMaster University

Hamilton, Canada

SUPERVISOR: Prof. S. Shirani

NUMBER OF PAGES: xx, 146

Abstract

Conventional image compression algorithms use transform coding to achieve a compact representation of the image. Most transforms used in image compression algorithms map image data to a complete set of transform basis functions which can decorrelate image information and represent data in a more compact form. This technique has proven to be very efficient and is used in most state of the art compression algorithms. However, if an over-complete set of basis functions is available, the image information can be captured by fewer basis functions. This results in a more compact image representation and can potentially yield a better compression performance. In this thesis, we study the use of over-complete image representation as an alternative to transform coding techniques used in image compression. The matching pursuit (MP) algorithm is used to map the image to an over-complete dictionary. We develop new quantization and encoding algorithms for matching pursuit image coding and compare the proposed MP image encoder with state of the art image codecs that use transform coding techniques. Additionally, the iterative nature of the matching pursuit algorithm can be used to design progressive encoders. We also study progressive coding by matching pursuit and design new progressive MP encoders and show how they outperform existing solutions.

We start by study of progressive coding by matching pursuit and design a progressive encoder for i.i.d. Gaussian sources. The choice of Gaussian sources is motivated by the fact that theoretical bounds on progressive coding of Gaussian sources are

known and therefore can be used to determine the efficiency of matching pursuit in progressive coding. Our proposed MP progressive encoder outperforms all existing progressive encoders designed for Gaussian sources. However, redundancies in the MP algorithm prevents us from closing the gap that exists between progressive and non-progressive Gaussian source coding. Therefore, we design another progressive encoder based on lattice quantization and address some of the issues associated with our proposed MP encoder.

In the second part of this thesis we study the application of matching pursuit in image compression. We start our study by developing a new adaptive quantization technique that can outperform existing quantization techniques designed for matching pursuit image coding. We continue our study by designing an optimal encoding algorithm for encoding MP coefficients and atom positions. The proposed encoding algorithm results in significant rate distortion improvement over existing encoding techniques. The use of our proposed encoding technique enables comparison of matching pursuit image coding with state of the art compression algorithms that use transform coding such as JPEG2000. Our proposed MP image encoder outperforms JPEG2000 at low bit rates and results in better visual quality at moderate bit rates. We show that the flexibility offered by the over-complete dictionary can result in superior performance compared to image compression using transform coding techniques.

Acknowledgements

I would like to express my sincere gratitude and appreciation to my supervisor Prof. Shahram Shirani for his guidance, supervision and support throughout my graduate studies at McMaster University. I would also like to thank the members of my PhD supervisory committee Prof. Xiaolin Wu and Prof. Tim Davidson for their invaluable suggestions and comments.

I would like to acknowledge the VXP group at Sigma Designs Technology Canada and Gennum Corporation, especially my managers David Vrhovnik, Mark McGrath and David Lynch for giving me the opportunity to finish my studies while working in the video algorithm development group.

Finally, I would like to thank my wife Atosa for her understanding and encouragements. I am also deeply grateful to my parents Zahra and Parviz, my sister Tina and my grandparents for always supporting me and believing in me.

List of Abbreviations

AEP	Asymptotic Equipartition Property
AVC	Advanced Video Coding
DC	Direct Current
DCT	Discrete Cosine Transform
DVD	Digital Video Disc
DWT	Discrete Wavelet Transform
Fig.	Figure
HH	High High
HL	High Low
i.i.d.	Independent and Identically Distributed
IEEE	Institute of Electrical and Electronics Engineers
JPEG	Joint Photographic Experts Group
LBG	Linde Buzo Gray
LH	Low High
LL	Low Low
MPEG	Moving Picture Experts Group
MS-TCQ	Multi Stage Trellis Coded Quantization
GB	Giga Bytes
KLT	Karhunen Leove Transform

MP	Matching Pursuit
NP	Non Polynomial
SRLQ	Successively Refinable Lattice Quantizer
dB	Decibel
PDF	Probability Distribution Function
PSNR	Peak Signal to Noise Ratio
R/D	Rate Distortion
SNR	Signal to Noise Ratio
SR-TCVQ	Successively Refinable Trellis Coded Vector Quantization
TB-SVQ	Trellis Based Scalar Vector Quantizer
TS-TCQ	Tree Structured Trellis Coded Quantizer
TCQ	Trellis Coded Quantization
TV	Television
VC	Video Codec
VLC	Variable Length Coding
VQ	Vector Quantization
ZIP	Compressed file
1-D	One Dimension
2-D	Two Dimensions

List of Symbols

$I(x)$	The amount of information in event x
$p(x)$	Probability of event x
$h(x)$	Entropy of random variable x
H_0	Low pass filter
H_1	High pass filter
R	Rate
D	Distortion
δ	The quantization step size
q	Number of quantization levels
$G(\Lambda)$	Normalized second moment of lattice Λ
Π	The Voronoi region
$V(\Lambda)$	The volume of the Voronoi region of lattice Λ
N	Dimension
$f_X(X)$	Probability Distribution Function of random variable X
ϵ	Arbitrary small number
$\|x\|$	Norm of x
Γ	The Gamma function
σ	variance
$E(x)$	Expectation of x
N-sphere	Sphere in $\mathbb{R}^N$

$var(x)$	Variance of x
D	Set of dictionary elements
g_γ	Dictionary element
k	Number of MP stages
c_i	Inner product coefficient at stage i
f_k	k stage MP representation of f
M	Number of MP dictionary elements
$\mathbf{H}$	Hilbert Space
$\mathbb{R}^N$	N dimensional space of real numbers
$R^i f$	MP residual vector at stage i
$\|x\|$	Absolute value of x
λ	Lagrangian Multiplier
∞	Infinity
$g(t)$	Gaussian function
$g_\alpha(i)$	One dimensional Gabor function
ϕ	phase shift
ζ	Modulation frequency
s	Scaling factor
$G_{\alpha,\beta}(i,j)$	Two dimensional Gabor function
$g(x,y)$	Two dimensional generating function of the MP dictionary
π	3.14159
e	2.71828
$\overrightarrow{a}$	Anisotropic scaling vector
(a_1, a_2)	Anisotropic scaling parameters
$\overrightarrow{b}$	Translation vector
(b_1, b_2)	Translation parameters
θ	Angle

f	Input vector
$\vec{r_i}$	Residual vector for the normalized input
r_i	Norm of $\vec{r_i}$
D_{MP}	Matching pursuit distortion
Π	Product
Σ	Summation
A_N	Surface area of unit N-sphere
S	Surface area of each Voronoi region
V_{N-1}	Volume of an $N-1$ dimensional sphere
$\mathcal{R}$	Radius of sphere with volume = surface area of the Voronoi region
$\mathbf{r_i}$	Random variable corresponding to r_i
$F_{\mathbf{r}_i}(r_i)$	Cumulative distribution function
$f_{\mathbf{r}_i}(r_i)$	Probability distribution function
$\mathbf{r}'_i$	$\mathbf{r_i}$ divided by $\mathcal{R}$
$J(x,y)$	The Jacobian matrix
$\mathbf{z}_k$	Product of $\mathbf{r}'_1$ to $\mathbf{r}'_k$
$R^i f_q$	Residual vector of quantized matching pursuit
c_{qi}	Quantized inner product coefficient
e_{qi}	Quantization error
r_{qi}	Residual vector of normalized quantized input vector
D_{MPQ}	Distortion of quantized matching pursuit
m_{c_i}	Mean of inner product coefficients
σ_{c_i}	Variance of inner product coefficients
$C(i)$	The upper bound of distortion for quantized matching pursuit
Ω_N	Surface are of N dimensional unit sphere
$\phi_i(x)$	Mapping function

W_Λ	Wrapped spherical codebook with respect to lattice Λ
$\bigcup$	Union
X_s	Shape component
X_g	Gain component
$Q[.]$	Quantization operator
X_{in}	The i^{th} shape residual vector
X_{in_q}	Quantized value of X_{in}
C	The range of vector X_{1n}
L	Intersection of C and the plane $z_N = 0$
X_L	The closest point to X_{in} located on L
X'	Mapping from N-D to (N-1)-D space by setting the N^{th} coordinate to zero
m	Mapping function
X_{inq}	Quantized value of X_{in}
R_i	Rate associated with the i^{th} refinement stage
R_{s_i}	Shape rate associated with the i^{th} refinement stage
R_{g_i}	Gain rate associated with the i^{th} refinement stage
Θ	Thickness of a lattice
R_c	Covering radius of a lattice
D_s	The shape distortion
D_g	The gain distortion
δ_i	Expectation of r_{qi}
$J(\lambda)$	The Lagrangian cost function
q_i	Number of quantization levels at stage i
e_{oi}	Overload error at stage i
$I_{i,j,k}$	Range of inner product coefficient at stage k in block (i,j)
$q_{i,j,k}$	Number of quantization levels at stage k in block (i,j)
B	Block size

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The evolution of digital electronic systems in recent decades has had an inevitable impact on people's daily lives. Electronic devices have made their ways into human's lives and many people cannot imagine finishing the day without getting online on their computers or receiving a phone call on their cell phones. Digital systems have changed the way humans can communicate with each other through development of a variety of digital media available today. Cell phones, digital TV, digital cameras, video games, internet, DVDs and many other types of digital media have provided variety of resources for people to communicate, interact with each other and send and receive information.

In order to use digital data, digital media content must be delivered to the end user through a transmission medium. This transmission medium can be an internet link, a wireless channel or a cable. Additionally in many applications, after creation of digital media, the digital media data must be stored on a storage device for future use or transmission. Storage devices can be a DVD or Blu-ray disc, a memory card, or any other storage device that can store digital content.

The storage and transmission of digital content can only be achieved if the storage device or the transmission channel have enough capacity to store or transmit

data. However, the size of data in many digital contents can exceed the available storage resources or the capacity of the communication channel. In order to facilitate data storage and transmission, digital data is usually compressed before storage or transmission. Data compression reduces the size of the data and therefore allows more data to be stored on limited storage resources or transmitted through a communication channel with limited bandwidth. As an example, if video data is not compressed, a DVD can only store about 2 minutes of uncompressed video while an MPEG2 compression codec allows the whole two hour movie to be stored on a 4 GB DVD disc [1]. Additionally, for most communication channels, bandwidth is an expensive and limited resource and therefore it is important to compress data as much as possible before sending it through the channel. A 300 Kbps internet connection allows for transmitting uncompressed video with only a resolution of 40 pixels by 40 pixels at 15 frames per second. For comparison, *YouTube* video sequences have a resolution of 240×320 pixels (i.e. 48 times higher resolution than a 40×40 video) and are transmitted at 30 frames per second (twice the frame rate of the uncompressed video) [2]. This is possible by the use of an H.263 compression codec in storage and transmission of YouTube sequences.

While the storage and bandwidth requirements necessitate the use of compression algorithms in transmitting digital media, it is worth mentioning that compression codecs are relatively cheap compared to their benefit. Any computer can easily decode MPEG video streams or compress data into a ZIP file [3]. JPEG codecs [4] are widely used in digital cameras. Even the old fax machines that are still in use compress data before transmission. The relatively low cost of compression codecs is an important factor in the fast adoption of compression techniques in industrial applications.

1.1 Overview of data compression

Data compression is achieved by using the structure in the data in order to represent it in a compact form. A data compression system consists of a compression encoder and a compression decoder. The encoder compresses data and generates a bitstream that contains fewer bits than the original data size. This smaller bitstream can be stored in a storage device or sent through a communication channel. At the receiver side, the compression decoder, receives the compression bit stream, decompresses the data and generates the original data from the encoded bitstream. The decoder must be aware of the compression algorithm used at the encoder in order to be able to decode data.

In many applications, a perfect reconstruction is achieved after compression and decompression and no data is lost during the compression process. This type of compression is referred to as lossless compression. A well known example is the ZIP file format that is widely used in computers for compressing and archiving computer files. After compression and decompression, the exact original file is reconstructed from the compressed ZIP file and no data is lost in the process.

Lossless compression cannot achieve the storage and bandwidth requirements of many applications. Moreover, in many applications, some loss of data is tolerable and humans cannot notice small differences to the original data. This is the case for image, video, audio and speech data. When some loss of information occurs during the compression process the compression algorithm is referred to as lossy compression. Most lossy compression algorithms take advantage of human perceptual system. For example, the human eye is less sensitive to loss in high frequency image information. Thus most compression algorithms take advantage of this property and allocate fewer bits for high frequency information in the image.

1.2 Overview of thesis

1.2.1 Motivations and objectives

Data compression has evolved considerably in the past few decades. JPEG and JPEG2000 [53] are both very efficient in image compression and are widely used in computers, digital cameras and digital cinema applications. Video compression standards like MPEG4 [5], H.264/AVC [6] and VC.1 [7] can compress video data to the bandwidth required for most of today's commercial applications. Although many compression techniques have been developed that can achieve performances that are very close to theoretically achievable performance bounds, in many cases comparison between theoretical bounds and practical compression qualities show a relatively large gap between what can be achieved theoretically and practical results. This is especially the case for image and video data in which the theoretical bounds of compression are not exactly known. At the core of most image and video compression algorithms is a transform coding which can be in the form of wavelet, DCT or integer transforms. All these transforms are complete transforms and they map the original image data to a complete set of transform basis functions. In this thesis we focus on an alternative technique to complete transforms and study the use of over-complete representations in compression of data. We use matching pursuit algorithm to map the data to an over-complete dictionary of basis functions. We try to find the best coding performance achievable by over-complete transforms and compare our results with the conventional compression techniques that use complete transforms.

While it is important to be able to compress data as much as possible with minimum loss, there are other properties that may be required for some applications that can constrain the compression algorithm. For example, in many applications it is important to be able to decode only part of the bitstream and achieve a relatively meaningful reconstruction of the original data before decoding the entire stream. As

an example, when a user is visiting a web page that contains an image, it is desired that a coarse reconstruction of the full image is available first and as more data is downloaded the quality of image improves. This can be achieved by progressive coding. The compression encoder ensures the compressed data is formatted in such a way that more data adds to the quality of reconstruction. This is in contrast with sequential coding in which the original data is encoded sequentially at its full compression quality. Another application of progressive coding is in browsing applications. In browsing, the viewer is only interested in a coarse representation of the image or video content and does not need to wait for the full resolution image or video to be decoded. Additionally in many applications, the same data should be transmitted through several channels with different channel capacities. The transmitter should be able to transmit as much data as each channel can transmit using a single progressive stream. An application of this is watching video over the internet. Users may have different internet speeds and if a video is encoded using a progressive or scalable coding technique, a same stream can be used by different users with different internet connection speeds. In this dissertation, we also focus on progressive coding of data and try to develop progressive encoding algorithms that are as close as possible to the theoretical bounds known for progressive coding. For this study, we focus on Gaussian sources due to the existence of theoretical limits on compression for progressive coding. However, we also design a progressive image encoder based on over-complete representations and compare it with existing image compression standards.

1.2.2 Outline

This dissertation is organized as follows: We began with this chapter which presents an introduction to our work. Chapter 2 provides the necessary background on data compression. In chapter 3 we propose a new progressive coding technique for Gaussian sources based on matching pursuit. In chapter 4 an alternative progressive

coding algorithm based on lattice quantization is designed for Gaussian sources. In chapter 5 we study image coding by over-complete representations and propose a new quantization algorithm for matching pursuit image coding. In chapter 6 we find an optimal encoding algorithm for matching pursuit image coding and compare our results with the conventional image compression algorithms. Chapter 7 includes our concluding remarks and future directions.

1.2.3 Contributions and publications

The results of the research presented in this thesis have been published in various journals and conference proceedings. Different parts of this thesis have been published in 5 conference papers [12–16] and 2 journal papers [8, 9]. Another journal paper is accepted for publication and will be published in March 2009 [10]. One journal paper has been accepted with mandatory minor revisions and is currently under revisions [11].

The research results presented in chapter 3 of this thesis is published in the IEEE Transactions on Signal Processing [8]. The algorithm proposed in chapter 5 of this thesis is published in the IEEE Transactions on Image Processing [9]. The work in chapter 4 of this thesis will be published in the March 2009 issue of the IEEE Transactions on Signal Processing and a summary of chapter 6 is submitted to the IEEE Transactions on Image Processing. This paper has been accepted with mandatory minor revisions and is currently under revisions.

Chapter 2

Background

Data compression was first studied and formulated by Claude Shannon in his classic paper [17] in the 1940's. In his paper Shannon founded a new branch in mathematics called information theory. Information theory addresses two fundamental questions:

1. How can we define and measure information?
2. What is the maximum information that can be sent through a communication channel?

Shannon showed that the amount of information in an outcome of a source depends on the probability of that outcome and can be measured by:

$$I(x) = -\log_2 p(x) \tag{2.1}$$

where $p(x)$ is the probability of event x. When the base of the log function is 2, information is measured in bits. Based on the above definition, the average information or *entropy* of a source can be found by:

$$h(x) = -\sum_x p(x) \log_2(p(x)) \tag{2.2}$$

Shannon showed that entropy of a source is the minimum bit rate which can be achieved by any lossless compression algorithm. In other words, in order to be able to transmit a source through a channel, the bandwidth of the channel must be at least equal to the entropy of the source.

In lossless compression the goal is to compress data as much as possible with no loss of information. According to information theory the minimum bit rate achieved by any compression algorithm is the entropy. Entropy of a source only depends on the probability distribution of the source. Therefore, the compression process can be summarized in two steps:

1. Find the probability distribution of the source

2. Find an encoding method that matches the probability distribution

The second step is often referred to as *entropy coding*. We start by assuming the probability distribution of the source is known and discuss different entropy coding techniques. Then we explain how probability distributions are estimated.

2.1 Entropy coding

Once the probability distribution function is known an entropy coding algorithm should be designed to match the PDF function of the input source. In other words, unique decodable binary strings should be assigned to each symbol such that the average number of bits required for encoding the input symbols is minimized. A common approach is to use variable length coding (VLC). In variable length coding the length of each codeword is selected based on the probability of the symbol corresponding to that codeword. Many VLC techniques are developed and used in data compression algorithms. Here we only describe two simple examples to clarify how entropy coding algorithms can compress data.

Symbol	Probability	Codeword
B	0.75	0
R	0.125	10
Y	0.0625	110
G	0.0625	111

Figure 2.1: (a) Huffman code for 4 symbols

2.1.1 Huffman coding

Huffman algorithm is one of the most widely used algorithms for designing variable length codes. Huffman algorithm finds the optimum code length for each symbol based on the probability of the symbol and generates a uniquely decodable binary string with a length found based on the probabilities. An example is shown in Fig. 2.1. The input alphabet contains 4 symbols and the probability of each symbol is shown in Fig. 2.1. Note that the entropy of this source is 1.1836 and the average rate obtained by this Huffman code is 1.375.

2.1.2 Extended Huffman coding

Although Huffman codes find the best code lengths for each symbol, the gap between the entropy and the rate of the Huffman code in the example in Fig. 2.1 shows that Huffman coding is not optimum in terms of compression. One way to achieve better rates than Huffman coding is to block multiple symbols together and encode them by Huffman codes. This method is called extended Huffman coding. An example is shown in Fig. 2.2. In this example two symbols from the example in Fig. 2.1 are blocked together and the corresponding Huffman code is computed. Note that the average rate for this code is 1.2031 which is much closer to the entropy than the Huffman code for a single symbol. It can be shown that extended Huffman codes can achieve rates equal to entropy if an infinite number of symbols are blocked together

Symbol	Probability	Codeword
BB	0.5625	0
BR	0.09375	100
RB	0.09375	1010
BY	0.046875	1011
BG	0.046875	1100
YB	0.046875	1101
GB	0.046875	1110
RR	0.015625	111100
RY	0.0078125	1111010
RG	0.0078125	1111011
YR	0.0078125	1111100
GR	0.0078125	1111101
YY	0.00390625	11111100
YG	0.00390625	11111101
GY	0.00390625	11111110
GG	0.00390625	11111111

Figure 2.2: (a) Extended Huffman code.

and encoded. However, the computational and storage cost grows exponentially which limits its use in practical applications. In order to solve this problem arithmetic coding has been developed. In arithmetic coding large number of symbols can be blocked together and encoded based on the probability of the group of symbols without having to compute all possible codes. This approach is used in many advanced image and video coding standards like JPEG2000.

2.2 Finding probability models

Finding probability distributions is required if the probability distribution of the source is not explicitly known. This is the case for many sources like image and video data. For image and video data the probability distributions are usually estimated based on the information available from the previously encoded data. The more information we use, the less uncertainty will remain in the source data. In other words

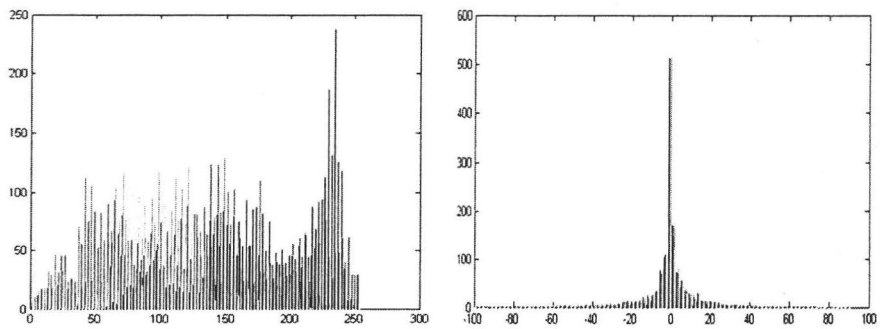
(a) (b)

Figure 2.3: (a) The original image (b) Prediction error image, prediction is found from the top pixel.

the variance of the approximated probability distribution will be smaller. In general the entropy depends on the variance of the PDF of the input source [27]. Probability distributions with small variances generally result in lower entropy and thus allow for more compression. Therefore, in order to achieve maximum compression, it is desired to find a probability distribution for the source that has the least amount of uncertainty or in other words results in the lowest possible variance and entropy. Many different approaches are used to achieve this goal. Since, in this dissertation we focus on image compression, we only describe the techniques used in image coding.

2.2.1 Prediction

Prediction is used in many image and video coding algorithms as a means to find probability distributions with low entropy. It is known that neighboring image pixels are highly correlated. This correlation can be extracted by spatial prediction techniques. An example is shown in Fig. 2.3. In this example each pixel is predicted from the pixel above. The prediction errors are shown in Fig. 2.3 (b). If the first line in the image and this prediction error image are encoded and transmitted, the decoder can reconstruct the original image from the prediction errors. As mentioned



(a) (b)

Figure 2.4: Histogram of the (a) original image (b) prediction errors

previously, the compression ratios that can be achieved depends on the probability distribution of the data that is being compressed. The histograms of the original and the prediction error data are shown in Fig. 2.4. As shown in this figure, the PDF of prediction errors has a much smaller variance than the PDF of the original image pixels. This means that better compression efficiency can be achieved if the compression is performed on the prediction error data rather than the original pixel data. Moreover, the shape of the PDF of prediction errors is very similar to a Laplacian distribution. This is the property of most natural images and can be used to design efficient entropy coding in order to compress prediction error data. More complex prediction techniques can result in prediction errors with lower variance and hence higher compression efficiency.

Successive video frames are also highly correlated and this correlation can be extracted by temporal prediction often referred to as motion compensation.

2.2.2 Transforms

Image data in its original pixel domain exhibits a probability distribution with high variations as shown in Fig. 2.4. Prediction was used to alleviate this problem and find new symbols that have a probability distribution function with a smaller variance. However, if the image data can be transformed to another domain in which the probability distribution is known and shows a small variance, we can encode and transmit the transform coefficients instead of original pixels and perform the inverse transform at the decoder in order to retrieve the original data. If the probability distributions of the transform coefficients are known and have small variances high compression ratios can be achieved by this approach. This technique is called transform coding. The next step is to find a transform that can result in probability distributions with the smallest variances.

2.2.2.1 Karhunen-Loeve Transform

It is shown in [48] that if the rows of the transform consist of the eigenvectors of the autocorrelation matrix of the input data, the resulting transform will minimize the geometric mean of the variance of the transform coefficients. This transform is called the Karhunen-Loeve Transform (KLT) and provides the largest transform coding gain among all transforms. However, if the source data is non-stationary which is the case for image data, the autocorrelation matrix is not fixed and the transform must be recomputed and sent to the decoder. This will limit the efficiency of the KLT transform in practical applications in which a fixed transform is desired.

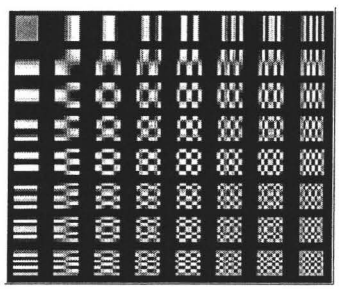


Figure 2.5: DCT basis functions

2.2.2.2 Discrete Cosine Transform

It is well known that image data exhibit similar characteristics in the frequency domain. Most of the information in natural images are in the low frequency coefficients in the frequency domain. This results in probability distributions with small variances for high frequency coefficients which in turn results in low entropy and high compression ratios. One of the most efficient transforms used in image compression is the discrete cosine transform (DCT). The DCT basis functions are shown in Fig. 2.5. The high frequency transform coefficients are generally smaller than the low frequency coefficients. An example is shown in Fig. 2.6. An 8×8 DCT transform is applied to the image in Fig. 2.6 (a) and the resulting transform coefficients are shown in Fig. 2.6 (b). High frequency coefficients are usually zero or close to zero (small variance) and most of the information are captured in a few low frequency coefficients. Moreover, the DC coefficients of neighboring blocks are also very close to each other. This means a simple prediction mechanism can be used to reduce the variance of PDF of DC coefficients and achieve better compression. DC prediction is used in most block DCT based codecs (e.g. JPEG [4]).

(a) (b)

Figure 2.6: (a) The original image (b) Transformed image

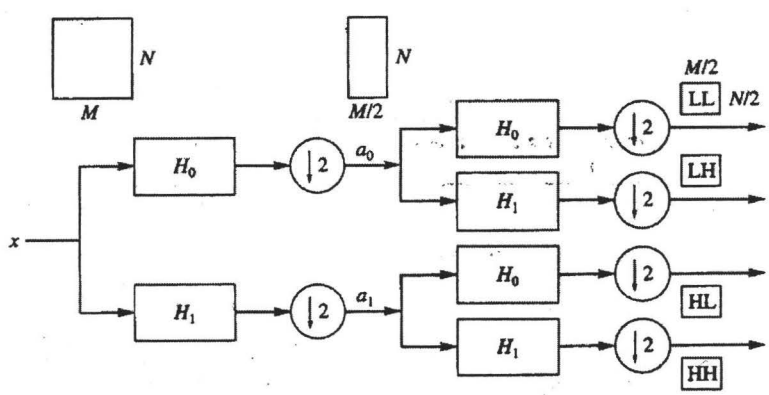


Figure 2.7: Implementation of wavelet transform by low pass and high pass filtering and downsampling. H_0 and H_1 are the low pass and high pass filters respectively.

2.2.2.3 Discrete wavelet transform

Wavelet transform is usually realized by applying low pass and high pass filters and downsampling the output. This is shown in Fig. 2.7. If the filters satisfy certain conditions [27], perfect reconstruction is possible and the original data can be recovered by the inverse wavelet transform. In terms of compression performance, for most natural images the majority of the information will be captured by the low frequency subband. An example is shown in Fig. 2.8 (b). Most of the high frequency coefficients are zero or close to zero (small variance) and therefore high compression ratios can be achieved in the high frequency subbands. For image coding applications, usually the low frequency subband is transformed further by the wavelet transform. An example is shown in Fig. 2.8 (c). Most of the image information is in the low frequency subband and high compression ratios can be achieved in the high frequency bands since most coefficients are zero and close to zero in these subbands. Another advantage of the wavelet transform is that it preserves the correlations between neighboring pixels. This property can be used to for better prediction of wavelet coefficients and is used in some wavelet based image compression techniques to achieve better compression (e.g. JPEG2000 [53]).

2.2.2.4 Over-complete transform

The transforms that we discussed in the previous sections are all complete transforms, i.e. the input signal is mapped to N independent transform basis functions where N is the dimension of the signal space. However, it is possible to have more than N basis functions in the transform dictionary. If there are more than N basis functions, the transform is called an over-complete transform. Since there are more basis functions than needed in the dictionary, the input signal can be represented with fewer basis functions and therefore, more energy compactness can be achieved. This

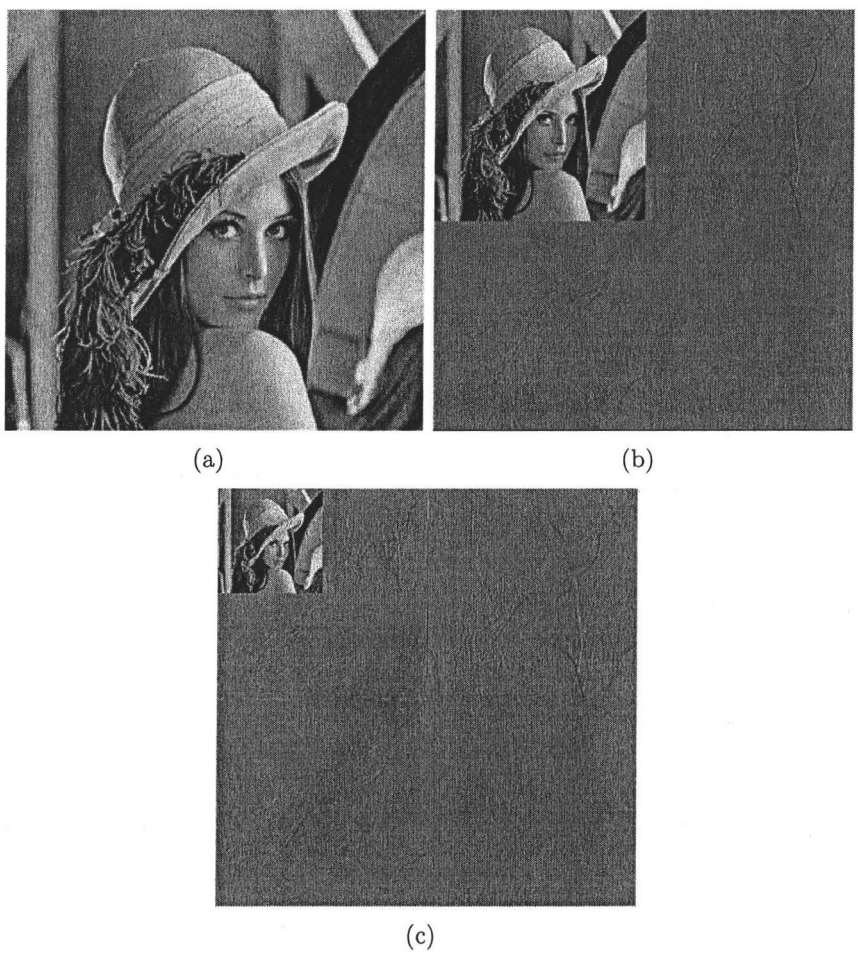


Figure 2.8: (a) The original image (b) Transform coefficients after 1 level of wavelet transform, (c) Transform coefficients after 2 levels of wavelet transform.

can result in better compression than complete transforms. In this thesis we study the application of over-complete signal representations in image coding. Therefore, we provide a more detailed review of over-complete representations in section 2.5.

2.2.3 Probability estimation

After prediction or transform, the input symbols must be compressed by an entropy coding technique. If the PDF has a small variance, the entropy coding technique can compress data efficiently and achieve high compression ratios. However, in order to design an entropy coding technique that matches the probability distribution of data, the exact PDF function must be known. The PDF function depends on the input image and should be estimated before an entropy coding algorithm can be used. In some applications fixed probability distributions that can closely approximate the actual PDF function are used. For example as shown in Fig. 2.4 the PDF of prediction errors is very similar to the PDF of a Laplacian source. In some compression algorithms a fixed Laplacian distribution is assumed for the prediction errors and the entropy coding is designed based on this distribution. In order to find a more accurate estimation a large set of images can be used and an average distribution can be estimated. This method results in better compression since the PDF matches the data more accurately.

For non-stationary sources the PDF function is not fixed and can change over time. For these applications the PDF function can be estimated adaptively from the already encoded data. This method is called context modeling and is used in most advanced image and video coding standards like JPEG2000 and H.264/AVC.

2.3 Lossy compression

So far we showed how data can be compressed to bit rates as close as possible to its entropy. However, in many cases more compression is required and compression to a bit rate equal to the entropy may not satisfy the storage or transmission requirements of the application. According to the information theory, more compression cannot be achieved without any loss of information. However, in many applications slight loss of information is tolerable and it is possible to achieve more compression at the cost of losing some information. This is especially the case for image and video data, since the viewer usually cannot notice small loss of data or in some cases is willing to watch a lower quality image or video to satisfy the bandwidth costs.

2.3.1 Rate distortion theory

Shannon's rate distortion theory provides the theoretical bounds for lossy compression. The theory states that, for every source there is a rate distortion function $R(D)$ that specifies the minimum bit rate required for any encoder to compress data such that the distortion of the reconstructed data is not larger than D. Based on the above definition the rate distortion function depends on:

1. the probability distribution of the source
2. the distortion metric used to find D

An example of a rate distortion function is shown in Fig. 2.9. According to the rate distortion theory, no encoder exists that operates outside the gray area. The rate distortion function is known for some sources (e.g. Gaussian source with mean squared error distortion). However for many sources the R/D function is not explicitly known due to the complexity of computing the rate distortion function and the unavailability of the probability distribution or a distortion metric that represents true distortion.

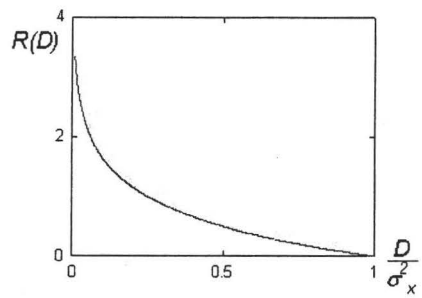


Figure 2.9: Rate distortion function of a Gaussian source with variance σ_x^2.

According to the rate distortion theory, in order to design a lossy compression algorithm, the PDF of the input source and a distortion metric must be used and the lossy compression algorithm must ensure the rate distortion performance is as close as possible to the rate distortion function.

2.3.2 Distortion metrics

The choice of distortion metric depends on the type of data that is being compressed. For image data the best metric is the one that can model how image is perceived by human visual system. Many different techniques based on modeling the human visual system are proposed and used in lossy compression. However, a complete model of the human visual system is not known and the ones designed in the literature are mostly very computationally complex. Therefore, in practical applications simpler distortion metrics are often used. Sum of squared errors is one of the most widely used metrics. However, for many practical applications sum of absolute errors is preferred in order to avoid the high computational complexity of finding squared errors. In order to model the human visual system simple techniques are often used. For example it is known that human eye is less sensitive to errors

in high frequency components in the image. This property is used in many image compression standards and the low frequency components are usually compressed at less loss than the high frequency components. Other techniques like color subsampling or region of interest coding are also used in order to approximate how images are perceived by the human visual system.

2.3.3 Quantization

In lossy compression, the amount of information in the source is reduced in order to be able to encode the information at the desired bit rate. To achieve optimum performance the encoded information must result in minimum distortion. The loss of information is usually achieved by quantization of data. Quantization is the process of representing a large or infinite set of values with a much smaller set [27]. The input range is divided to several regions and each region is represented by a codeword. All input values that lie within the same region are represented by the same codeword. The decoder receives the codeword and assigns a reconstruction value for each codeword. The design of a quantizer includes defining all regions, codewords and reconstruction values such that the output rate distortion performance is as close as possible to the rate distortion curve of the source. For example, if the bit rate is fixed and we use fixed length coding for encoding the codewords, the optimum quantizer is the one that minimizes distortion for a given number of quantization regions .

Quantization techniques vary in terms of performance and complexity. In general more complex quantization techniques can reduce the amount of information while achieving lower distortions. In this section we review some of the more common quantization techniques as well as the ones used in this thesis.

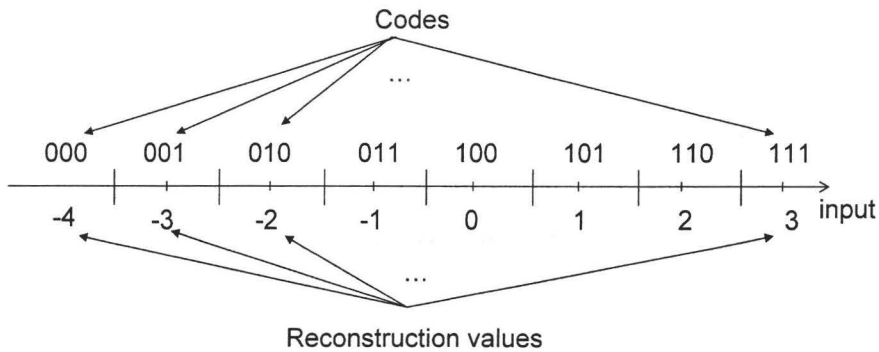


Figure 2.10: A 3-bit uniform quantizer.

2.3.3.1 Scalar quantization

In this technique quantization of each sample is performed independently of other samples. The simplest form is uniform scalar quantization. An example is shown in Fig. 2.10. In this quantizer all quantization regions are the same size except the two outer intervals. The reconstruction values are the midpoint of each interval. In scalar quantization the size of the quantization interval is called the quantization step size. For example the quantizer in Fig. 2.10 has a step size of $\delta = 1$. Since zero is one of the reconstruction values this special uniform scalar quantizer is called the midtread quantizer.

Suppose we want to design a q level uniform scalar quantizer for an input uniformly distributed in $[-X_{max}, X_{max}]$. In the uniform scalar quantizer, the range of input must be divided to equally sized intervals. Therefore, each quantization step size will be $\delta = \frac{2X_{max}}{q}$. Since data is uniformly distributed in each quantization interval and the midpoint of the interval is used as the reconstruction value, the mean square distortion can be computed by:

$$D = \frac{1}{\delta}\int_{-\delta/2}^{\delta/2} x^2 dx = \frac{\delta^2}{12} = \frac{(2X_{max})^2}{12q^2} \tag{2.3}$$

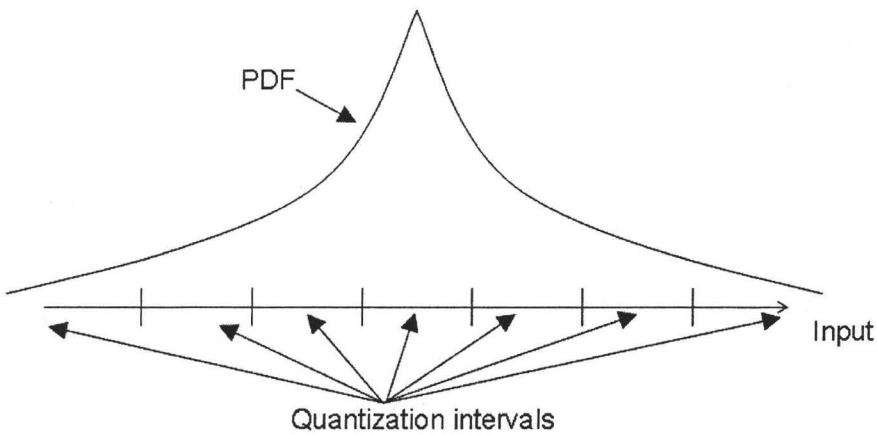


Figure 2.11: A uniform quantizer for a Laplacian PDF.

A uniform quantizer can be used to quantize nonuniform sources as well. An example is shown in Fig. 2.11. In this example a Laplacian source is quantized by a uniform quantizer. In this case since the data is unbounded, the quantization errors are unbounded as well. For uniform quantizers the errors in the inner intervals are bounded. This type of error is called granular error. However, for the outer intervals the error can be infinite. This error is referred to as overload error. Both overload errors and granular errors must be computed to find the distortion for this type of quantizer.

A better approach for quantizing nonuniform sources is to use nonuniform quantizers. In nonuniform quantizers, the quantization intervals can be different and the reconstruction values are not evenly distributed in the input range. An example is shown in Fig. 2.12 for a Laplacian source. The quantization step size is smaller in the intervals close to the origin since the data is more likely to be in this range. Therefore, for most cases a smaller distortion is obtained than a uniform scalar quantizer.

One of the most commonly used nonuniform quantizers is the pdf-optimized quantizer. In this quantizer, the quantization intervals and the reconstruction values are

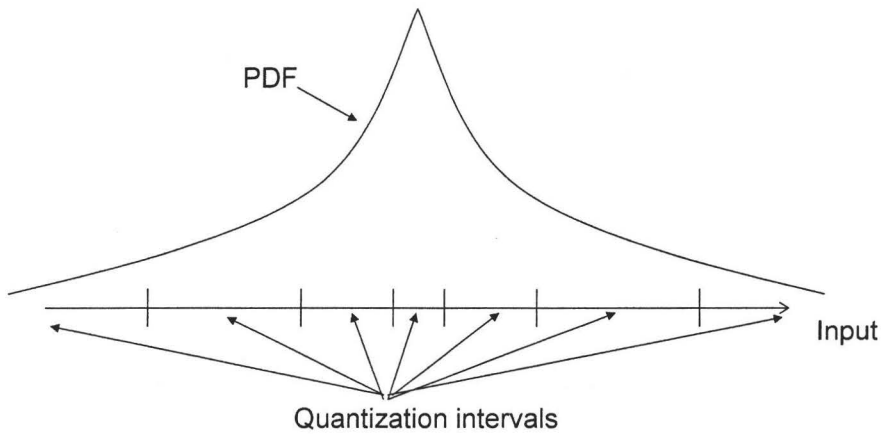


Figure 2.12: A non-uniform quantizer for a Laplacian PDF.

obtained based on the probability distribution of the source such that the average output distortion is minimized. However, in order to minimize the distortion in terms of quantization intervals, the values of the reconstruction levels are required and to minimize in terms of reconstruction levels the quantization intervals must be known. Lloyd and Max designed a technique for solving this problem iteratively [55]. This quantizer is called Lloyd-Max quantizer and is widely used for nonuniform quantization of nonuniform sources.

2.3.3.2 Vector quantization

In vector quantizers instead of encoding each input value individually, a group of input values are grouped together to form a vector and the vector is quantized. By quantizing symbols together it is possible to exploit the structure in the source data and achieve better quantization performance, i.e. better distortion at a given rate or a better rate at a given distortion. Even when the source is uniformly distributed better rate distortion performance can be achieved when a vector of input values are quantized compared to scalar quantization. We will discuss this in more detail

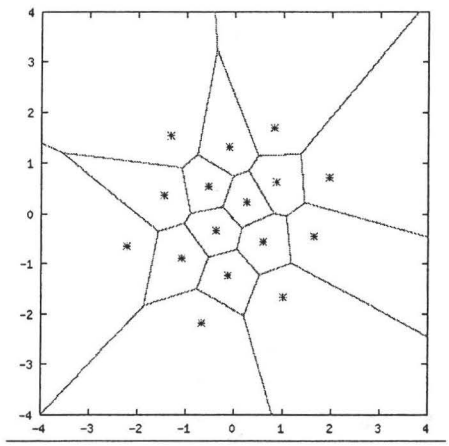


Figure 2.13: Vector quantization example. The stars are the reconstruction values and each cell is the Voronoi region.

in the section 2.3.3.3. However, when there is some structure in the source data, the structure is easier to extract when a large number of input values are available. Therefore a vector quantizer can achieve better performance than a scalar quantizer. The design of vector quantizers involve designing quantization regions, quantization codewords and reconstruction values. The quantization region is often referred to as the Voronoi region. An example is shown in Fig. 2.13. In this example the stars are the reconstruction values and each cell is the Voronoi region. All values inside the Voronoi region are quantized to the same reconstruction value. In order to exploit the structure in the source, the Voronoi regions in the areas where the probability of input source is high must be smaller. In other words more reconstruction values and Voronoi regions must be placed in the areas where data is more likely to be located. The most common approach to achieve this is to use the Linde-Buzo-Gray algorithm (LBG) [29]. LBG algorithm is based on a clustering procedure known as the k-means algorithm. In the LBG algorithm, we assume a training set containing a large number of samples from the input source is available. We start with an initial

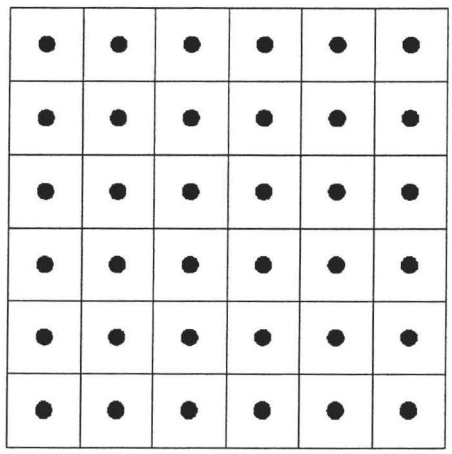


Figure 2.14: The cubic lattice in 2-D

set of reconstruction values. Then for each reconstruction value we find the points that are closer to this reconstruction value than any other reconstruction value. Then, each reconstruction value is updated by the average value of the points that are closer to this reconstruction value. This process continues until a convergence condition is satisfied.

2.3.3.3 Lattice quantization

A lattice is a regular arrangements of points in the space. In Lattice quantization a subset of lattice points is used as the reconstruction values of a vector quantizer. The simplest lattice is the cubic lattice shown in Fig. 2.14. This lattice is in fact similar to performing scalar quantization on each of the elements in the input vector. A better lattice in two dimensions is the hexagonal lattice quantizer shown in Fig. 2.15. It can be shown that if the size of the Voronoi regions are similar, the granular distortion of the hexagonal lattice is smaller than the granular distortion of a cubic lattice. In fact as the shape of the Voronoi region approaches the shape of a sphere

(circle in 2-D), the resulting distortion will be lower. In our 2 dimensional example, since hexagon is closer to a circle than a square, the granular distortion of a hexagonal lattice quantizer is smaller than the granular distortion of a cubic lattice quantizer. It is proven than the best lattice quantizer in 2 dimensions is the hexagonal lattice quantizer [28].

Many different factors are used to compare lattices. In general as the shape of the Voronoi region approaches the shape of a sphere the lattice results in lower granular distortion. The normalized average mean squared error per dimension can be used to compare lattices designed at different dimensions and with different Voronoi region size. It is computed as:

$$G(\Lambda) = \frac{\frac{1}{N-1}\int_{\Pi} \|t\|^2 dt}{V(\Lambda)^{1+\frac{2}{N-1}}} \tag{2.4}$$

where Π is the Voronoi region of the lattice Λ, $V(\Lambda)$ is the volume of the Voronoi region and N is the dimension of the lattice. $G(\Lambda)$ is often called the normalized second moment of the lattice. Its value is 0.083333 for the cubic lattice and 0.080188 for the hexagonal lattice [31]. Other important properties are packing radius and density. The packing radius is the largest radius of the non-overlapping spheres that are centered at the lattice points and the density is the proportion of space that is occupied by these spheres. If the density is 1, the entire space is covered by nonoverlapping spheres which results in the best sphere packing and therefore the best lattice quantizer. The density of a hexagonal lattice is 0.9069 while for a cubic lattice it is 0.785. The packing radiuses of the cubic lattice and the hexagonal lattice are both 0.5. Other related properties are the covering radius and the thickness. The covering radius of a lattice is the radius of the smallest spheres centered at the lattice points that cover the entire space and the thickness of the lattice is the ratio of the volume of this sphere to the volume of the Voronoi region of the lattice. If the thickness is 1, the Voronoi region is a sphere and therefore the best lattice quantizer is achieved. For example the thickness of a cubic lattice is 1.5708 while for a hexagonal

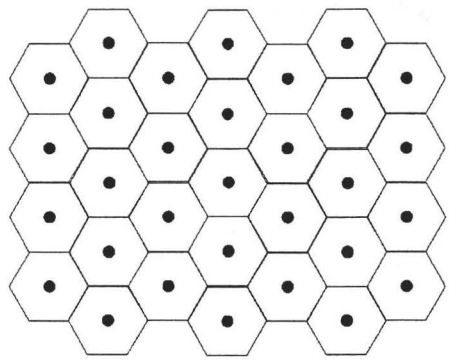


Figure 2.15: The hexagonal lattice in 2-D.

lattice the thickness is equal to 1.2092. The covering radii of the cubic lattice and the hexagonal lattice are 0.7071 and 0.5774 respectively.

Since a lattice is a structured arrangement of points, this structure can be used to design fast quantization algorithms. In fact in a lattice quantizer it is not necessary to find all lattice points and their corresponding codewords and save them in memory. For each input point the closest lattice point can be computed without finding all the lattice points and their distances to the input point. This significantly reduces the number of computations and the size of memory. However, lattices can only be effective if the data is uniformly distributed in the range of the lattice. Additionally, since lattices minimize the granular distortion, they are more effective if the granular distortion is the dominant source of distortion compared to overload errors. This is not the case at low bit rates and therefore lattice quantization is more effective in higher resolution and bit rates.

In this thesis, we use the 24-dimensional Leech lattice in chapters 3 and 4. This lattice is the best known lattice and results in the smallest granular distortion. The normalized second moment of this lattice is equal to 0.065771 compared to 0.08333 for the cubic lattice and 0.080188 for the hexagonal lattice. Its packing and covering radii

are 1 and 1.4142 respectively. Its thickness is 7.9035 and its density is 0.001930 [31].

2.3.4 Rate control

Most lossy compression algorithms include a rate control block that ensures the encoder operates as close as possible to the rate distortion function. Rate control usually adjusts the number of quantization regions used for quantizing different image components such that the resulting rate is smaller than the desired output rate while the overall distortion is minimized. Lagrangian optimization [30] is usually used in order to allocate bits to different components in the image in a rate distortion optimal manner. In Lagrangian optimization it is proven that optimum performance is achieved when the slopes of the R/D curves of all components are the same. Therefore, the rate distortion function associated with each image component must be computed. This can be done by performing the actual encoding and calculating the resulting bit rates and distortions for several different encoder parameters. In some applications it is possible to compute the bit rate and distortion from the probability distribution function of the input source. In any case, the Lagrangian optimization approach ensures a point on the R/D curve with similar slope for all image components are selected in order to ensure the overall performance is rate distortion optimal. This approach is used throughout this thesis to allocate bit rates to different data components in a rate distortion optimal manner.

2.4 Progressive source coding

In many applications it is desired to be able to decode an approximation of the input signal at a lower bit rate before decoding the entire stream at full bit rate. The user can see a low quality image first and if needed can decode the rest of the bit stream in order to achieve higher qualities. This is important in browsing applications

where the viewer does not need a high quality representation of every image, instead wants to be able to get the high quality image of the selected images. Progressive coding refers to first approximating a signal using a few bits and then successively refining the approximation by sending more information about signal. This process is also called successive refinement. Equitz and Cover studied the problem of successive refinement in [18]. They defined a source as successively refinable if it is possible to achieve rate distortion optimal descriptions of the source at each refinement stage. In other words, a source is successively refinable if it is possible to achieve all points on the rate distortion curve using a single scalable stream. They showed that successive refinement can only be achieved if and only if the descriptions at each refinement stage can be written as the Markov Chain [18]. Therefore not every source is successively refinable. However, they showed that i.i.d. Gaussian sources with squared error distortion, Laplacian sources with absolute error distortion and arbitrary discrete sources with Hamming distortion are successively refinable. They showed that Gaussian sources are successively refinable if an infinite number of samples from the Gaussian source are blocked together and quantized. They also described a general strategy for designing successively refinable quantizers for Gaussian sources. However, in practice the dimension of the input vector cannot be arbitrarily large and therefore successive refinement may not be possible. In this dissertation, we propose two different algorithms for successive refinement of Gaussian sources. Since it is proven in [18] that successive refinement theoretically does not increase the bit rate required for encoding Gaussian sources, our goal is to achieve rate distortion functions that are obtained by other non-progressive encoders designed for Gaussian sources.

2.4.1 Gaussian source coding

Before we study progressive coding of Gaussian sources we need to explain how Gaussian sources are quantized and encoded. Let X be an N dimensional vector consisting of i.i.d. random variables x with pdf $f_x(x)$ and entropy $h(x)$. The asymptotic equipartition property (AEP) states that for sufficiently large N and arbitrary small ϵ, we can write [27]:

$$\left|\frac{\log f_X(X)}{N} + h(x)\right| < \epsilon \tag{2.5}$$

The above equality means that as N approaches infinity, almost all vectors lie close to a contour of constant probability:

$$\left|\frac{\log f_X(X)}{N}\right| = -h(x) \tag{2.6}$$

In the case of having a Gaussian distribution, these contours are in fact hyper-spheres. The result of the above statement is that the optimum way to encode Gaussian sources at rate R is to distribute 2^{RN} points uniformly on the N-dimensional sphere [35].

If elements of X are selected from a memoryless Gaussian source with zero mean and variance σ^2, the properties of random variable $\|X\|$ can be summarized by [37]:

$$f_{\|X\|}(r) = \frac{2r^{N-1}\exp(\frac{-r^2}{2\sigma^2})}{\Gamma(N/2)(2\sigma^2)^{N/2}} \tag{2.7}$$

$$E(\|X\|) = \frac{\sqrt{2\sigma^2}\Gamma(\frac{N+1}{2})}{\Gamma(\frac{N}{2})} \tag{2.8}$$

$$E(\|X\|^2) = N\sigma^2 \tag{2.9}$$

$$\text{var}(\|X\|) = N\sigma^2 - 2\sigma^2\left(\frac{\Gamma(\frac{N+1}{2})}{\Gamma(\frac{N}{2})}\right)^2 \tag{2.10}$$

where $\Gamma(k)$ is the Gamma function defined as:

$$\Gamma(k) = \int_0^\infty l^{k-1}e^{-l}dl \tag{2.11}$$

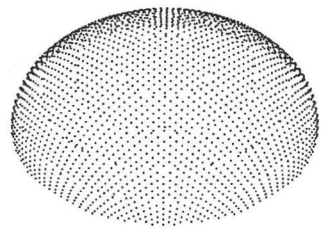


Figure 2.16: Codebook generated by wrapped spherical codes

It is easy to verify that for asymptotically large N, the random vector $X/\sqrt{N\sigma^2}$ is uniformly distributed on the surface of an N-dimensional unit sphere. Many quantizers designed for Gaussian sources encode the Gaussian source using vectors uniformly distributed on the surface of a unit N-dimensional sphere [35, 37, 46]. In this thesis we use the wrapped spherical codes designed in [36] to distribute vectors uniformly on the sphere. Wrapped spherical codes refer to a method of distributing points uniformly on the surface of an N dimensional sphere [36]. In wrapped spherical codes, a lattice in $N-1$ dimensions is mapped to the surface of an N dimensional sphere. The surface of the sphere is divided into several annuli and each annulus is partitioned by an $N-1$ dimensional lattice. A sample dictionary designed based on wrapped spherical codes is shown in Fig. 2.16. The wrapped spherical codes have been used in [37] to design a quantizer for Gaussian sources. This quantizer is one of the best quantizers designed for Gaussian sources and operates within 1 dB of the rate distortion function for bit rates higher than 1 bit per sample. In this thesis, we extend this quantizer to provide successively refinable output streams. Two different method are used and compared with other progressive quantizers in chapters 3 and 4.

2.5 Over-complete representations

In most traditional image compression algorithms, first a transform is used to capture image information in few transform coefficients. In the transform domain, the lower frequency components contain most of the image energy, while there is little information in the high frequency components. Therefore, most of the image information can be captured by encoding a few transform coefficients which results in a compressed representation. Most transforms used in image compression are complete transforms, i.e. the image is mapped to a complete set of basis functions (e.g. wavelet basis functions in JPEG 2000 or DCT basis functions for JPEG). This means that if the space of the input signal is of dimension N, the input signal is mapped to a set of N independent basis functions (the DCT basis functions are shown in Fig. 2.5). Now let $D = \{g_\gamma\}_{0 \leq \gamma \leq M}$ be an over-complete (or redundant) dictionary of basis functions. Therefore D includes at least N linearly independent basis functions (i.e. $M > N$). For any $k \geq 1$ an approximation f_k of f can be calculated with a linear combination of k elements in D [38]:

$$f_k = \sum_{i=0}^{k-1} c_i g_{\gamma_i} \tag{2.12}$$

Since an over-complete set of basis functions is available, the input data can be more efficiently represented and the information can be captured by fewer basis functions than in complete transforms. This potentially can result in better compression.

Obviously, there is no unique solution for the above representation and many combinations of basis functions can be formed to represent f_k. For general over-complete dictionaries finding k basis functions that minimize $\|f - f_k\|$ is an NP hard problem [38,39]. Therefore, pursuit algorithms have been developed to reduce the computational complexity of finding sparse signal representations. Many pursuit algorithms have been proposed for efficient mapping of signals to over-complete dictionaries. Matching pursuit [25], orthogonal matching pursuit [57], tree based pursuit [58] and

basis pursuit [56] are among the most efficient pursuit algorithms designed to map input signals to redundant dictionaries. Among these methods matching pursuit (MP) is perhaps one of the most commonly used due to its relatively low computational complexity and good performance and is the one used in this thesis to solve the problem of mapping signals to over-complete dictionaries.

2.5.1 Matching Pursuit

Matching pursuit is a greedy algorithm that decomposes a signal $f \in \mathbf{H}$ into an over-complete dictionary of bases ($g_\gamma \in \mathbf{H}$) [25] where $\mathbf{H}$ is a Hilbert space. At each stage of matching pursuit the dictionary element g_γ that results in the maximum inner-product with the residual signal $R^i f$ is found. Then, the residual signal is projected on g_γ. This can be written as:

$$R^i f = \langle R^i f, g_\gamma \rangle g_\gamma + R^{i+1} f \tag{2.13}$$

where $R^{i+1} f$ is the new residual signal. At the first stage $R^0 f$ is replaced by the input signal f. The matching pursuit iterations continue based on equation (2.13) for k stages in order to achieve a k stage MP decomposition. Therefore, the MP signal representation can be written as:

$$f = \sum_{i=0}^{k-1} \langle R^i f, g_{\gamma_i} \rangle g_{\gamma_i} + R^k f \tag{2.14}$$

The inner product coefficients and the dictionary indexes of all stages form the description of the input signal.

2.5.1.1 Convergence of matching pursuit

Since g_{γ_i} and $R^{i+1} f$ are orthogonal and the dictionary elements are normalized, it can be shown that:

$$\|R^i f\|^2 = |\langle R^i f, g_{\gamma_i} \rangle|^2 + \|R^{i+1} f\|^2 \tag{2.15}$$

Equations (2.15) and (2.14) result in signal energy decomposition that can be written as:

$$\|f\|^2 = \sum_{i=0}^{k-1} |\langle R^i f, g_{\gamma_i} \rangle|^2 + \|R^k f\|^2 \tag{2.16}$$

$\|R^k f\|^2$ is the square norm of the error in the k-stage MP representation. Mallat proved in [38] (theorem 9.10) that the residual norm is upper-bounded and the upper-bound is an exponentially decreasing function of the iteration stage i. This can be written as:

$$\|R^i f\| \leq 2^{-i.\lambda} \|f\| \tag{2.17}$$

where λ is a constant that depends on the dictionary. Equation (2.17) shows that as $i \rightarrow \infty$, $\|R^i f\| \rightarrow 0$ and therefore error in MP representation converges to 0. Moreover, according to equation (2.15):

$$|\langle R^i f, g_{\gamma_i} \rangle|^2 \leq \|R^i f\|^2 \tag{2.18}$$

Substituting equation (2.18) into equation (2.17) results in:

$$|\langle R^i f, g_{\gamma_i} \rangle| \leq 2^{-i.\lambda} \|f\| \tag{2.19}$$

Thus, the absolute values of the inner product coefficients are also upper-bounded by an exponentially decreasing function of the iteration number. This upper-bound is often used for efficient quantization of inner product coefficients. Equations (2.17) and (2.19) show that, although the sparsity of the MP representation is not guaranteed, the MP representation always converges and since the value of λ depends on the dictionary, the rate of convergence depends on the dictionary that is used.

2.5.2 MP image coding

Matching pursuit was first used for encoding motion compensated residual video frames by Neff et al. in [22]. MP was used instead of DCT transform and significant

PSNR improvements was achieved at very low bit rate video coding. In a more recent work in [24], MP is applied to image compression and it is shown that matching pursuit image coding can improve image compression quality at very low bit rates while providing additional features like fine grained PSNR and bit rate scalability. Moreover, the subjective quality of MP encoded images is significantly better than those compressed by traditional compression algorithms like JPEG 2000.

When MP is used for image coding applications, the input image is successively mapped to a redundant dictionary of bases. MP representation consists of both the dictionary indexes and the quantized inner product coefficients at each stage. Therefore, efficient dictionary design and efficient encoding techniques heavily affect the bit rate of the MP encoder.

2.5.2.1 MP dictionary design

When MP is used for image coding, the dictionary usually consists of a set of functions, that are scaled, rotated, and modulated versions of a generating function. Then each dictionary element is translated to all pixel locations in the image to generate the over-complete dictionary. The dictionary elements are often called atoms. Therefore, the matching pursuit representation consists of atom parameters (e.g. scaling, frequency modulation or rotation parameter), atom positions and inner product coefficients.

The most commonly used dictionary is the separable Gabor dictionary which consists of a collection of 2 dimensional Gabor functions. The Gabor functions are found by scaling and modulating a Gaussian window defined as:

$$g(t) = \sqrt[4]{2}e^{-\pi t^2} \tag{2.20}$$

The 1-D Gabor functions are found by scaling and modulating the above Gaussian

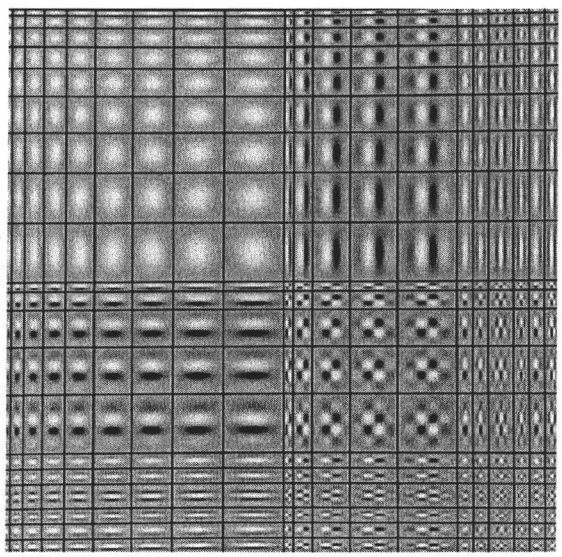


Figure 2.17: The 2-D separable Gabor dictionary.

window:

$$g_\alpha(i) = K_\alpha . g(\frac{i - N/2 + 1}{s}) . \cos(\frac{2\pi\zeta(i - N/2 + 1)}{8} + \phi) \tag{2.21}$$

$i \in 0, 1, ..., N-1$. K_α is a normalizing factor that ensures the function is of unit norm. s is the scale, ζ is the modulation frequency and ϕ is a phase shift. The 2 dimensional separable Gabor functions are found by:

$$G_{\alpha,\beta}(i,j) = g_\alpha(i) g_\beta(j) \tag{2.22}$$

In practice a set of discrete scales, frequency modulation and phase shifts are selected to form the dictionary. Then this dictionary is translated to all pixel positions in the image to form the over-complete dictionary. For example, in [22] a set of 20 triples of scales, modulation frequency and phase shifts was selected to form a dictionary of size 400 that is translated to all pixel locations. This dictionary is shown in Fig. 2.17. The use of this dictionary in low rate video coding applications results in significant improvements over DCT based codecs.

Although the above dictionary is very efficient in representing motion compensated residual frames, it is not very efficient for image coding applications. A better dictionary was designed in [24] and it was shown that this dictionary can result in better coding performance than wavelet based JPEG2000 codec at very low bit rates. This dictionary is also built based on changing parameters of a generating function. Instead of a Gaussian window, a different generating function was designed with the goal of being able to efficiently approximate contours in 2 dimensions. To achieve this goal, the generating function is a Gaussian function in one direction and the second derivative of a Gaussian function in the other direction:

$$g(x,y) = \frac{2}{3\pi}(4x^2 - 2)e^{-(x^2+y^2)} \tag{2.23}$$

The choice of Gaussian function in one direction is motivated by the optimal localization of Gaussian functions in time and frequency domain and the second derivative of Gaussian functions in the other direction is selected for efficient representation of strong edges in the image. This generating function is scaled by anisotropic scaling factor $\overrightarrow{a} = (a_1, a_2)$ to adapt to contour smoothness and is rotated by angles θ to adapt to contour direction. Similar to other dictionaries used in image coding, all atoms are translated to every pixel position by a translation vector $\overrightarrow{b} = (b_1, b_2)$ to find the exact location of contours in the image. The resulting basis functions can be written as:

$$g_\gamma(x,y) = \frac{2}{\sqrt{3\pi}}(4g_1^2 - 2)e^{-(g_1^2+g_2^2)} \tag{2.24}$$

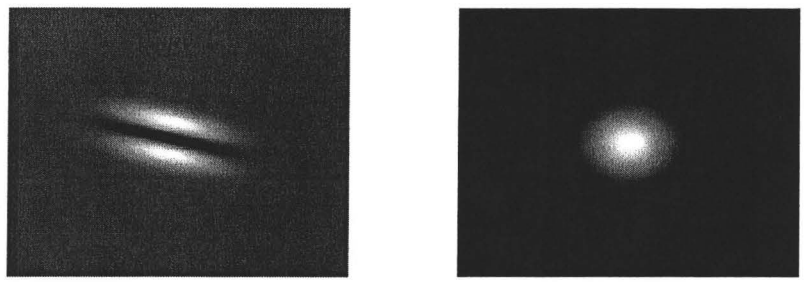
where

$$g_1 = \frac{\cos(\theta)(x - b_1) + \sin(\theta)(y - b_2)}{a_1} \tag{2.25}$$

and

$$g_2 = \frac{\cos(\theta)(y - b_2) - \sin(\theta)(x - b_1)}{a_2} \tag{2.26}$$

The above generating function is not able to efficiently capture low frequency information in the image. Therefore, a second generating function is included in



(a) (b)

Figure 2.18: Example of dictionary elements used in our experiments (a) anisotropic atom, (b) Gaussian atoms.

the dictionary for efficient representation of low frequency information in the image. The second generating function is a Gaussian function in both directions and can be written as:

$$g(x,y) = \frac{1}{\sqrt{\pi}} e^{-(x^2+y^2)} \tag{2.27}$$

The above generating function is scaled by isotropic scaling and no rotation is applied since the generating function is symmetric around origin. Two dictionary elements belonging to this dictionary are shown in Fig. 2.18.

2.5.2.2 MP atom and coefficient coding

As mentioned before, the MP representation consists of quantized inner product coefficients, atom positions and atom indexes which can consist of different scale, modulation and rotation parameters. While it is possible to encode each parameter independently, a better approach would be to use the correlations that exist between atom parameters and different atoms in order to achieve higher compression efficiency. There have been a number of works addressing the problem of efficient encoding of MP coefficients and indexes. These coding methods can be classified into two major

categories. The more common method is to find all atoms and encode them in their position order. The position of atoms are differentially encoded which results in fewer bits needed to encode the position information of dictionary indexes [22, 23, 54]. In the second class of methods, atoms are encoded in the order of the magnitude of coefficients. In this method more bits are assigned to more important atoms which results in fewer bits required to encode the inner product coefficients [9, 41]. Both methods are shown to result in almost similar performances for image coding applications [41]. However, the second class allows for fine grained PSNR and bit rate scalability which is an important property in image compression and is the method used in [24] for compressing still images by matching pursuit.

In chapter 5 of this dissertation, we propose an encoding algorithm that belongs to the second category. In our proposed algorithm atoms are encoded in their importance order. We use the correlations between successive atoms in order to achieve lower bit rates than the ones obtained in [41]. We show that our proposed encoding algorithm outperforms the one in [41] for all images and bit rates.

Additionally, we propose another encoding algorithm that finds the optimum tradeoff between encoding in position order and encoding in importance order. Significant improvements are achieved by this algorithm and our results are better than JPEG2000 at low bit rates. This algorithm is discussed in detail in chapter 6.

2.5.3 Progressive coding by matching pursuit

The iterative nature of matching pursuit algorithm can be used for progressive coding. In MP at each stage the closest dictionary element to the input signal is found and encoded along with its inner product coefficient. Therefore, each stage results in an increment in signal quality and therefore MP is an immediate candidate for progressive coding. In fact this property is used in [24] and they showed how MP can be used in progressive coding of images. They showed MP can provide

fine-grained PSNR and bit rate scalability beyond what could be achieved by JPEG 2000. In this thesis we also study the application of matching pursuit in progressive coding. From a practical point of view we study the application of MP in progressive coding of images and design new algorithms that can outperform existing techniques designed for progressive coding of image sources. Moreover, we study the application of MP in progressive coding from a theoretical point of view and try to answer if MP can provide *optimum* progressive coding. For this study we chose Gaussian sources since the theoretical rate-distortion limits for progressive coding of Gaussian sources are known. A progressive encoder based on matching pursuit is designed in chapter 3 and we compare it with other progressive encoders designed for Gaussian sources. We also design another successively refinable quantizer based on lattice quantization in chapter 4 and try to achieve better performance than the one designed in chapter 3.

Chapter 3

Progressive Coding of i.i.d. Gaussian Sources Using Matching Pursuit

As mentioned in chapter 2 a source is said to be successively refinable if the output distortion at each rate is minimal, i.e. no decrease in the rate-distortion performance of the encoder occurs due to successive refinement. Equitz et al. proved in [18] that i.i.d. Gaussian sources are successively refinable if infinite number of samples from the Gaussian source are blocked together and quantized. However, in practice the dimension of the input vector cannot be too large and thus successive refinement may not be possible. In fact, in all the existing progressive quantizers for Gaussian sources [19–21, 49], rate-distortion performance is traded for successive refinability and no practical method exists that preserves the performance of a Gaussian source quantizer while providing successive refinement.

In this chapter, we study the application of matching pursuit algorithm in successive refinement of i.i.d. Gaussian sources. The rate-distortion performance of the MP encoder is analyzed in terms of dictionary size and number of quantization levels

and optimum parameters are calculated based on a probabilistic model for Matching pursuit residual vectors.

As mentioned in chapter 2, the optimum way to encode a Gaussian source at rate R is to distribute 2^{RN} points uniformly on the N-dimensional sphere. Our approach in this chapter is to use this property and apply matching pursuit to successively map the signal to a dictionary with vectors uniformly distributed on the surface of a sphere. This approach results in an embedded bitstream for the Gaussian source. We compare the performance of the MP encoder with other quantizers designed for successive refinement of Gaussian sources and show that significant improvements can be achieved by our MP encoder.

3.1 MP for Spherically Uniform Signals

In this section the average distortion of matching pursuit is calculated in terms of MP encoder parameters for signals uniformly distributed on the surface of a unit-sphere. As mentioned in chapter 2, matching pursuit decomposes a signal $f \in \mathbf{H}$ into an over-complete dictionary of bases ($g_\gamma \in \mathbf{H}$) [25] where $\mathbf{H}$ is a Hilbert space (in this chapter $\mathbf{H} = \mathbb{R}^N$). At each stage of matching pursuit the dictionary vector g_γ that results in the maximum inner-product with the residual signal $R^i f$ is found. Then, the residual signal is projected on g_γ. This can be written as:

$$R^i f = \langle R^i f, g_\gamma \rangle g_\gamma + R^{i+1} f \tag{3.1}$$

where $R^{i+1} f$ is the new residual signal. At the first stage $R^0 f$ is replaced by the input signal f. The matching pursuit iterations continue based on equation (3.1) for k stages in order to achieve a k stage MP decomposition. Therefore, the MP signal representation can be written as:

$$f = \sum_{i=0}^{k-1} \langle R^i f, g_{\gamma_i} \rangle g_{\gamma_i} + R^k f \tag{3.2}$$

The inner product coefficients and the dictionary indexes of all stages form the description of the input signal. Since g_{γ_i} and $R^{i+1}f$ are orthogonal and the dictionary vectors are normalized, it can be shown that:

$$\|R^i f\|^2 = |\langle R^i f, g_{\gamma_i}\rangle|^2 + \|R^{i+1} f\|^2 \tag{3.3}$$

Equations (3.3) and (3.2) result in signal energy decomposition that can be written as:

$$\|f\|^2 = \sum_{i=0}^{k-1} |\langle R^i f, g_{\gamma_i}\rangle|^2 + \|R^k f\|^2 \tag{3.4}$$

$\|R^k f\|^2$ is the square norm of the error in the k-stage MP representation. Let us divide both sides of equation (3.1) by $\|R^i f\|$:

$$\frac{R^i f}{\|R^i f\|} = \langle \frac{R^i f}{\|R^i f\|}, g_\gamma \rangle g_\gamma + \frac{R^{i+1} f}{\|R^i f\|} \tag{3.5}$$

and let:

$$\vec{r_i} \triangleq \frac{R^i f}{\|R^{i-1} f\|} \quad \text{and} \quad r_i \triangleq \|\vec{r_i}\| = \frac{\|R^i f\|}{\|R^{i-1} f\|} \tag{3.6}$$

Equation (3.5) is the MP equation for the normalized signal $\frac{R^i f}{\|R^i f\|}$ and therefore, r_i is the norm of the resulting residual signal when matching pursuit is applied to the normalized signal. Substituting $\|R^0 f\|$ by $\|f\|$ and using equation (3.6) repeatedly, we can write:

$$\|R^k f\| = \|f\| r_1 r_2 ... r_k \tag{3.7}$$

The square of the norm of the residual signal at the k^{th} stage is equal to the total distortion of k-stage matching pursuit. Since the input signal is on the surface of a unit N-sphere, $\|f\| = 1$, Therefore:

$$D_{\text{MP}} = \|R^k f\|^2 = (r_1 r_2 ... r_k)^2 \tag{3.8}$$

where D_{MP} is the distortion of matching pursuit. In order to find the distortion of matching pursuit, the behavior of the residual norm r_i must be studied. In what

follows, we first approximate the Voronoi regions in which the vectors $\vec{r}_i$ are located. Then, we find the probability distribution of r_i and their product $(\prod_{i=1}^{k} r_i)$. Once this probability distribution is available, the expectation of D_{MP} can be calculated.

Since f is an N-dimensional vector, $\frac{R^i f}{\|R^i f\|}$ is a vector on the surface of a unit N-sphere where N-sphere is an N-dimensional hyper-sphere. Moreover, since the dictionary elements are normalized N dimensional vectors, they are also on the surface of the unit N-sphere. Now suppose the negative of each dictionary vector is included in the dictionary. Therefore, finding the maximum inner-product is equivalent to finding the minimum Euclidean distance (which is equal to the mean squared error distortion). If the size of the original dictionary is M, the size of the new dictionary will be $2M$. Note that this assumption is only for our modeling convenience and is not implemented in practice. In fact, the sign bit of the inner product coefficient is now used to encode the dictionary index for a dictionary of size $2M$.

Our analysis is for signals uniformly distributed on the surface of the unit N-sphere. Since the signal is uniform, for the best rate-distortion performance, the dictionary vectors must be uniformly placed on the surface of the N-sphere. This means that the Voronoi regions for each dictionary vector must be identical and therefore have the same area. Moreover, to maximize uniformity, the Voronoi regions must approach the shape of a spherical cap. The latter argument is justified by the similarity of matching pursuit dictionary design to sphere packing problem in lattice quantization [31]. In sphere packing problem, it is argued that the best covering of space is a dense packing of spheres with minimal overlap. Therefore, in lattice quantizer design the best Voronoi regions are the ones that best approximate a sphere [32]. The difference between matching pursuit and lattice quantizer dictionary design is that in matching pursuit the space is the surface of a unit N-sphere rather than $\mathbb{R}^N$. Therefore, N-spheres are analogous to N-dimensional spherical caps (sphere caps of a hyper-sphere in $\mathbb{R}^N$). So the best covering of the surface of a unit N-sphere is a

dense packing of N-dimensional spherical caps with minimal overlap.

If the total surface area of the unit N-sphere is A_N, since there are $2M$ Voronoi regions and the Voronoi regions are identical, the surface area of each of the Voronoi regions will be $S = \frac{A_N}{2M}$. For the optimum dictionary these shapes must be as close as possible to the shape of a spherical cap and their surface areas must be as close as possible to S. Therefore, when the number of elements in the dictionary is high, it is reasonable to approximate the Voronoi regions by $N-1$ dimensional spheres with the same volume as the surface area of the Voronoi regions (S). This is shown in Fig. 3.1 (a). Thus, using the equations for the volume and surface area of hyper-spheres [33], the volume of these $N-1$ dimensional spheres can be written as (the radius of the unit N-sphere is 1):

$$V_{N-1} = S = \frac{A_N}{2M} \quad \Rightarrow \quad \frac{\pi^{\frac{N-1}{2}}}{(\frac{N-1}{2})!}\mathcal{R}^{N-1} = \frac{N\pi^{\frac{N}{2}}}{(\frac{N}{2})!2M} \tag{3.9}$$

where $\mathcal{R}$ is the radius of the $N-1$ dimensional sphere. Therefore, $\mathcal{R}$ can be found by:

$$\mathcal{R} = \left(\frac{\sqrt{\pi}N(\frac{N-1}{2})!}{2(\frac{N}{2})!}\right)^{\frac{1}{N-1}} M^{-\frac{1}{N-1}} = t(N)M^{-\frac{1}{N-1}} \tag{3.10}$$

where $t(N)$ is defined by:

$$t(N) \triangleq \left(\frac{\sqrt{\pi}N(\frac{N-1}{2})!}{2(\frac{N}{2})!}\right)^{\frac{1}{N-1}} \tag{3.11}$$

Now that the Voronoi regions are modelled for the uniform dictionary, the distribution of the residual norms (r_i) can be found. According to (3.5) and (3.6):

$$\frac{R^{i-1}f}{\|R^{i-1}f\|} = \langle \frac{R^{i-1}f}{\|R^{i-1}f\|}, g_\gamma \rangle g_\gamma + \vec{r_i} \tag{3.12}$$

Therefore, $\vec{r_i}$ is the residual vector for the normalized signal $\frac{R^{i-1}f}{\|R^{i-1}\|}$. Hence, $\vec{r_i}$ points from the approximation of the normalized vector to the normalized vector which is located in the Voronoi region as shown in Fig. 3.1.(b). The Voronoi regions are

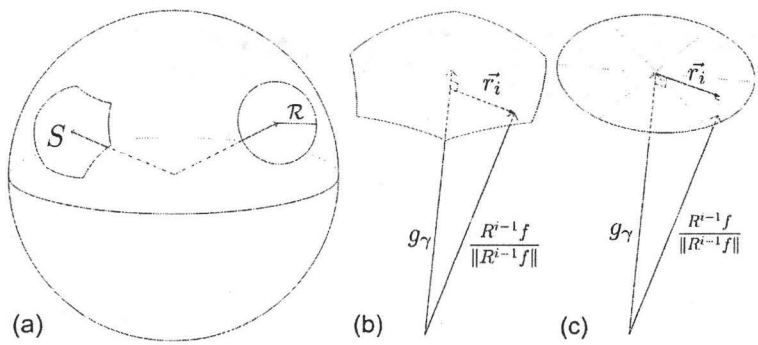


Figure 3.1: (a) 3-dimensional Voronoi region is approximated by a two dimensional circle with the same volume as the surface area of the Voronoi region (in two dimensions, volume=$\pi\mathcal{R}^2$, surface area=$2\pi\mathcal{R}$) (b) Geometric interpretation of the residue $\vec{r_i}$. (c) $\vec{r_i}$ is uniformly distributed in the volume of the 2-dimensional sphere.

approximated by an $N-1$ dimensional sphere with a radius found in equation (3.10). Since the input signal is uniformly distributed on the surface area of a unit N-sphere, the residual vector $\vec{r_i}$ is uniformly distributed in the volume of the $(N-1)$-sphere with radius $\mathcal{R}$ as shown in Fig. 3.1.(c). $r_i = \|\vec{r_i}\|$ will be the distance of the point corresponding to vector $\vec{r_i}$ to the center of the sphere. Let $\mathbf{r_i}$ be the random variable corresponding to this distance. Since the residual vector is uniformly distributed in the volume of the $(N-1)$-sphere, the probability distribution of $\mathbf{r_i}$ can be expressed as:

$$F_{\mathbf{r}_i}(r_i) = P(\mathbf{r}_i \leq r_i) = \frac{\frac{\pi^{\frac{N-1}{2}}}{(\frac{N-1}{2})!} r_i^{N-1}}{\frac{\pi^{\frac{N-1}{2}}}{(\frac{N-1}{2})!} \mathcal{R}^{N-1}} = \frac{r_i^{N-1}}{\mathcal{R}^{N-1}} \tag{3.13}$$

Therefore:

$$f_{\mathbf{r}_i}(r_i) = \frac{dF_{\mathbf{r}_i}(r_i)}{dr_i} = \frac{(N-1)r_i^{N-2}}{\mathcal{R}^{N-1}} \quad 0 \leq r_i \leq \mathcal{R} \tag{3.14}$$

Now in order to compute the right hand side of equation (3.8), the probability distribution function of a new random variable corresponding to $\mathbf{r}_1\mathbf{r}_2...\mathbf{r}_k$ must be found.

First, Let us define a new random variable $\mathbf{r}'_i$ as:

$$\mathbf{r}'_i \triangleq \frac{\mathbf{r}_i}{\mathcal{R}} \tag{3.15}$$

Therefore, the PDF of $\mathbf{r}'_i$ can be computed as [34]:

$$f_{\mathbf{r}'_i}(r'_i) = (N-1)r_i'^{N-2} \quad 0 \leq r'_i \leq 1 \tag{3.16}$$

From equations (3.8) and (3.15), the distortion of k-stage MP is:

$$D_{\text{MP}} = \mathcal{R}^{2k}(r'_1 r'_2 ... r'_k)^2 \tag{3.17}$$

Thus, in order to find D_{MP}, the PDF of $\mathbf{r}'_1\mathbf{r}'_2...\mathbf{r}'_k$ must be calculated. We use the following lemma to find this probability distribution.

Lemma 1 *Suppose* $\mathbf{x}$ *and* $\mathbf{y}$ *are independent random variables with PDF's:*

$$f_{\mathbf{y}}(y) = \eta y^{\beta} \quad 0 \leq y \leq 1,\ f_{\mathbf{x}}(x) = ax^{\beta}(\ln x)^{\rho-1} \quad 0 \leq x \leq 1 \tag{3.18}$$

Let $\mathbf{z} = \mathbf{xy}$. *Then:*

$$f_{\mathbf{z}}(z) = -\frac{\eta a}{\rho} z^{\beta}(\ln z)^{\rho} \quad 0 \leq z \leq 1 \tag{3.19}$$

where η, β, a *and* ρ *are constants.*

Proof: Let $\mathbf{w}$ be an auxiliary random variable defined by $\mathbf{w} = \mathbf{x}$. The determinant of the Jacobian matrix is $J(x, y) = -w$. Therefore the PDF of $\mathbf{z}$ can be written as [34]:

$$f_{\mathbf{z}}(z) = \int f_{\mathbf{zw}}(z, w)dw = \int \frac{1}{|w|} f_{\mathbf{xy}}\left(w, \frac{z}{w}\right) dw \tag{3.20}$$

For nonzero probability distribution functions, x and y must be between 0 and 1. Therefore, $0 \leq w \leq 1$ and $0 \leq \frac{z}{w} \leq 1$. This yields the boundaries of the integral equation as $z \leq w \leq 1$. The independence of the two random variables implies that:

$$f_{\mathbf{z}}(z) = \int_z^1 \frac{1}{w} f_{\mathbf{x}}(w) f_{\mathbf{y}}\left(\frac{z}{w}\right) dw \tag{3.21}$$

Substituting the PDF's of $\mathbf{x}$ and $\mathbf{y}$ from (3.18) into (3.21):

$$f_{\mathbf{z}}(z) = \int_z^1 \frac{a}{w} w^{\beta} (\ln w)^{\rho-1} \eta \frac{z^{\beta}}{w^{\beta}} dw = a\eta z^{\beta} \int_z^1 (\ln w)^{\rho-1} \frac{dw}{w} \tag{3.22}$$

Therefore, the solution of the integral equation is:

$$f_{\mathbf{z}}(z) = -\frac{\eta a}{\rho} z^{\beta} (\ln z)^{\rho} \quad 0 \leq z \leq 1 \quad \blacksquare \tag{3.23}$$

From equation (3.16) the PDF's of $\mathbf{r}'_1$ and $\mathbf{r}'_2$ have the same form as equation (3.18) with $\eta = a = N-1$, $\beta = N-2$ and $\rho = 1$. In order to use Lemma 1 we must prove that $\mathbf{r}'_1$ and $\mathbf{r}'_2$ are independent. According to (3.5) and (3.6), $\mathbf{r}_{i+1}$ is the norm of the residual vector at the $(i+1)^{th}$ stage if matching pursuit is applied to the normalized residual vector at the i^{th} stage (i.e. $\frac{R^i f}{\|R^i f\|}$). Therefore, $\mathbf{r}_{i+1}$ is determined only by $\frac{R^i f}{\|R^i f\|}$. But

$$\frac{\vec{r}_i}{\|\vec{r}_i\|} = \frac{\frac{R^i f}{\|R^{i-1} f\|}}{\|\frac{R^i f}{\|R^{i-1} f\|}\|} = \frac{R^i f}{\|R^i f\|} \tag{3.24}$$

Thus, $\mathbf{r}_{i+1}$ depends only on $\frac{\vec{r}_i}{\|\vec{r}_i\|}$. Since $\|\vec{r}_i\|$ and $\frac{\vec{r}_i}{\|\vec{r}_i\|}$ are independent (the norm and the direction of a vector are independent) and $\mathbf{r}_{i+1}$ only depends on $\frac{\vec{r}_i}{\|\vec{r}_i\|}$, we can conclude that $\mathbf{r}_{i+1}$ is independent of $\|\vec{r}_i\| = \mathbf{r}_i$. Therefore their scaled versions $\mathbf{r}'_{i+1}$ and $\mathbf{r}'_i$ are also independent.

Using Lemma 1, the PDF of $\mathbf{z}_2 \triangleq \mathbf{r}'_1 \mathbf{r}'_2$ can be written as:

$$f_{\mathbf{z}_2}(z_2) = -(N-1)^2 z_2^{N-2} (\ln z_2) \quad 0 \leq z_2 \leq 1 \tag{3.25}$$

Equation (3.25) has the same form as the PDF of x in Lemma 1 with $a = -(N-1)^2$, $\beta = N-2$ and $\rho = 2$. The PDF of $\mathbf{r}'_3$ (equation (3.16)) has also the same form as the PDF of y in Lemma 1 with $\eta = N-1$ and $\beta = N-2$ and $\mathbf{r}'_3$ and $\mathbf{z}_2 = \mathbf{r}'_1 \mathbf{r}'_2$ are independent. Thus, the PDF of $\mathbf{z}_3 \triangleq \mathbf{z}_2 \mathbf{r}'_3 = \mathbf{r}'_1 \mathbf{r}'_2 \mathbf{r}'_3$ can be found using Lemma 1:

$$f_{\mathbf{z}_3}(z_3) = \frac{(N-1)^3}{2} z_3^{N-2} (\ln z_3)^2 \quad 0 \leq z_3 \leq 1 \tag{3.26}$$

Continuing this for k times, the PDF of $\mathbf{z}_k \triangleq \mathbf{r}'_1\mathbf{r}'_2...\mathbf{r}'_k$ will be:

$$f_{\mathbf{z}_k}(z_k) = \frac{(N-1)^k(-1)^{k-1}}{(k-1)!} z_k^{N-2}(\ln z_k)^{k-1} \quad 0 \leq z_k \leq 1 \tag{3.27}$$

Now $E(z_k^n) = \int_0^1 z_k^n f_{\mathbf{z}_k}(z_k)dz_k$ can be computed by:

$$E(\mathbf{z}_k^n) = \int_0^1 z_k^n \frac{(N-1)^k(-1)^{k-1}}{(k-1)!} z_k^{N-2}(\ln z_k)^{k-1}dz_k \tag{3.28}$$

Let $l \triangleq -(n+N-1)\ln z_k$. Therefore, $z_k = e^{-\frac{l}{n+N-1}}$, $dz_k = -\frac{e^{-\frac{l}{n+N-1}}}{n+N-1}dl$ and $0 \leq l \leq \infty$. Thus:

$$E(\mathbf{z}_k^n) = \frac{(N-1)^k \int_0^\infty l^{k-1}e^{-l}dl}{(k-1)!(n+N-1)^k} = \frac{(N-1)^k\,\Gamma(k)}{(k-1)!(n+N-1)^k} \tag{3.29}$$

where $\Gamma(k)$ is the Gamma function defined in equation (2.11). Since $\Gamma(k) = (k-1)!$ for integer k, the expectation of $\mathbf{z}_k^n$ is:

$$E(\mathbf{z}_k^n) = \left(\frac{N-1}{n+N-1}\right)^k \tag{3.30}$$

Substituting (3.30) and (3.10) into (3.17), the average distortion of k-stage matching pursuit for signals and dictionaries uniformly distributed on a hyper-sphere can be found by:

$$D_{\text{MP}} = t(N)^{2k}M^{\frac{-2k}{N-1}}\left(\frac{N-1}{N+1}\right)^k \tag{3.31}$$

3.2 Quantized Matching Pursuit

In the previous section we found the distortion of matching pursuit assuming the values of inner product coefficients are known to the decoder. However, since inner product coefficients can take on any real values, they have to be quantized before they can be encoded and sent to the decoder. Therefore, the distortion caused by quantization must be considered in the total matching pursuit distortion. In practical matching pursuit encoders, the quantized inner product coefficient is used in the

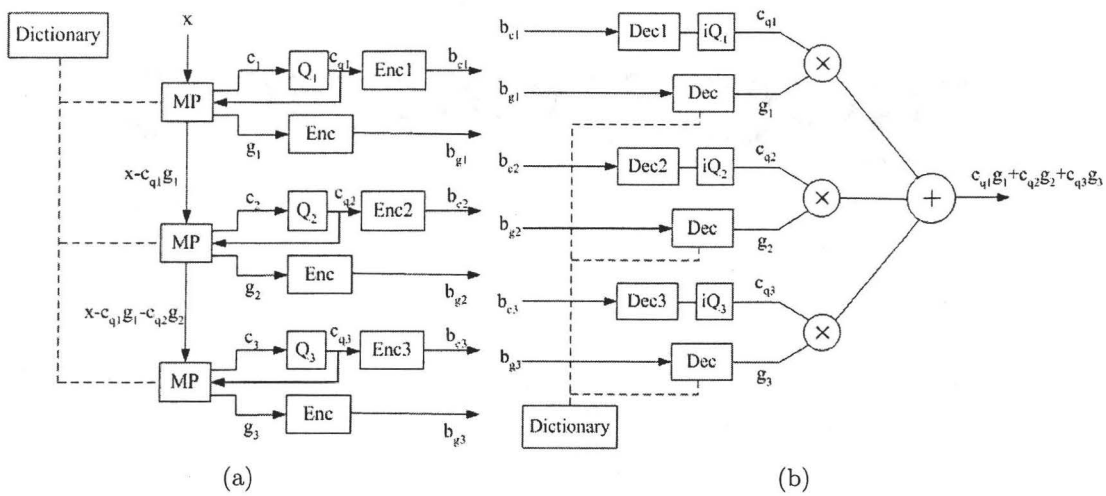


Figure 3.2: The block diagrams of (a) MP encoder, (b) MP decoder

computation of the residual vector in order to prevent accumulation of quantization error (in-loop quantization). This results in partial correction of the quantization error in the subsequent stages. Therefore direct addition of the quantization error to the MP distortion will not accurately model the total distortion. In this section we find a reasonably accurate approximation for the distortion in quantized matching pursuit. Our approach is based on the results of previous section with modifications to facilitate computation for quantized matching pursuit.

The block diagrams of quantized matching pursuit encoder and decoder are shown in Fig. 3.2. At the encoder side, the input vector goes through the first stage of matching pursuit and the dictionary vector with maximum inner product coefficient is found along with the inner product coefficient. The inner product coefficient is quantized and encoded and is sent to the decoder together with the encoded index of the dictionary vector. Then, the quantized inner product coefficient is used to find the residual vector which goes through the second matching pursuit stage. This process

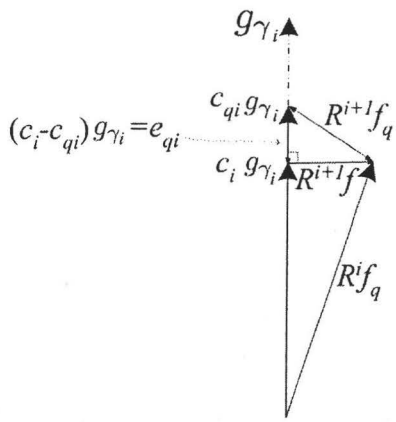


Figure 3.3: Geometric interpretation of quantized matching pursuit

continues for all MP stages until the full k stage decomposition is constructed. The decoder decodes the inner product coefficients and dictionary indexes and reconstructs the input vector according to equation (3.2). Since the quantized inner product coefficient is used to find the input to next stages, the MP equation at each stage (equation (3.1)) can be modified to:

$$R^i f_q = \langle R^i f_q, g_{\gamma_i} \rangle g_{\gamma_i} + R^{i+1} f \tag{3.32}$$

where $R^i f_q$ is the residual vector which is computed using the quantized value of inner product coefficients at previous stages.

Let c_i and c_{qi} be the unquantized and quantized values of the inner product coefficient at stage i respectively. If the inner product coefficients are not quantized, the total distortion at stage i will be $\|R^{i+1} f\|^2$. However, at each stage the inner product coefficient is quantized. This introduces an additional quantization error $\|e_{qi}\|^2$, where:

$$e_{qi} = (c_i - c_{qi}) g_{\gamma_i} \tag{3.33}$$

Part of this error will be corrected in the subsequent stages if we plug

$$R^{i+1}f_q = R^{i+1}f + e_{qi} = R^i f_q - c_{qi}g_{\gamma_i} \tag{3.34}$$

into the next stage instead of $R^{i+1}f$ (see Fig. 3.2). The distortion at stage i is equal to the square norm of the total residual vector at stage i (i.e. $\|R^{i+1}f_q\|^2$). Since $R^{i+1}f$ and e_{qi} are orthogonal (Fig. 3.3):

$$\|R^{i+1}f_q\|^2 = \|R^{i+1}f + e_{qi}\|^2 = \|R^{i+1}f\|^2 + \|e_{qi}\|^2 \tag{3.35}$$

Equation (3.35) can be written as:

$$\|R^{i+1}f_q\|^2 = \|R^i f_q\|^2 r_{q_{i+1}}^2 + \|e_{qi}\|^2 \tag{3.36}$$

where r_{qi} is defined as:

$$r_{qi} \triangleq \frac{\|R^i f\|}{\|R^{i-1}f_q\|} \tag{3.37}$$

According to this definition and equation (3.32), $r_{q_{i+1}}$ is the residual vector when matching pursuit is applied to the normalized input vector ($\frac{R^i f_q}{\|R^i f_q\|}$). This is exactly similar to the definition of r_{i+1} in the previous section except that in quantized MP, the input vector is $\frac{R^i f_q}{\|R^i f_q\|}$ while in unquantized MP the input vector was $\frac{R^i f}{\|R^i f\|}$. Since in section 3.1, the only assumption on the input vector $\frac{R^i f}{\|R^i f\|}$ was that it is uniformly distributed on the surface of the unit sphere and this assumption holds for $\frac{R^i f_q}{\|R^i f_q\|}$ as well, we can conclude that r_{qi} has the same characteristics as r_i defined in equation (3.6). More specifically, the PDF of r_{qi} is the same as the PDF of r_i computed in equation (3.14).

The average distortion at stage i can be written as:

$$D_{MPQi} = E\{\|R^{i+1}f_q\|^2\} = E\{\|R^i f_q\|^2 r_{q_{i+1}}^2\} + E\{\|e_{qi}\|^2\} \tag{3.38}$$

Now in the above equation, $r_{q_{i+1}}$ only depends on $\frac{R^i f_q}{\|R^i f_q\|}$ (i.e. the direction of vector $R^i f_q$). Since norm and direction of the residual vectors are independent, $\|R^i f_q\|$ and

$r_{q_{i+1}}$ are independent and we can write:

$$D_{MPQi} = E\{\|R^{i+1}fq\|^2\} = E\{\|R^i f_q\|^2\}E\{r^2_{q_{i+1}}\} + E\{\|e_{qi}\|^2\} \tag{3.39}$$

$E\{r^2_{q_{i+1}}\}$ can be found from the PDF of $r_{q_{i+1}}$ (which is found in equation (3.14)) and equation (3.30) for $k = 1$ and $n = 2$:

$$E\{r^2_{q_{i+1}}\} = t(N)^2 M^{\frac{-2}{N-1}} \left(\frac{N-1}{N+1}\right) \tag{3.40}$$

Now, let us assume that the values of $E\{\|e_{q_i}\|^2\}$ are known. If $E\{\|e_{qi}\|^2\}$ is known for every i, starting from the first stage, we can write:

$$\|R^1 f_q\|^2 = \|R^1 f\|^2 + \|e_{q0}\|^2 = \|f\|^2 r^2_{q1} + \|e_{q0}\|^2 \tag{3.41}$$

or

$$D_{MPQ0} = E\{\|R^1 f_q\|^2\} = E\{\|f\|^2\}E\{r^2_{q1}\} + E\{\|e_{q0}\|^2\} \tag{3.42}$$

$D_{MPQ1} = E\{\|R^2 f_q\|^2\}$ can also be found from equations (3.40), (3.42) and (3.39). This process can be continued to find $E\{\|R^{i+1} f_q\|^2\}$ for every i using the recursive formula in equation (3.39) and the expectations found in the previous stages.

Now, it only remains to compute the values of $E\{\|e_{qi}\|^2\}$ at each stage. The quantization error $E\{\|e_{qi}\|^2\}$ depends on the quantizer that is used at each stage and the PDF of inner product coefficients at that stage. In order to find the quantization distortion at each stage we need to find the PDF of the inner product coefficients. Unfortunately, unlike the previous section, it is not easy to compute the probability distributions when quantization is performed within the encoding cycle. However, if quantization is done outside the encoding cycle, we can approximately find the PDF of inner product coefficients. If quantization error is not the dominant source of distortion (relatively high resolution quantization), we can approximately assume that the PDF of inner product coefficients does not change significantly by quantization.

According to equation (3.3):

$$\|R^i f\|^2 = c_i^2 + \|R^{i+1} f\|^2 \tag{3.43}$$

Thus, using equation (3.6) we can write:

$$c_i^2 = \|R^i f\|^2 (1 - r_{i+1}^2) \tag{3.44}$$

For large dictionary sizes ($M > 2^{2N}$), $\mathcal{R}$ is smaller than 0.25 and we can assume $r_{i+1}^2 \ll 1$ (since $r_{i+1} < \mathcal{R}$). Therefore:

$$c_i \approx \|R^i f\| = \mathcal{R}^i z_i \tag{3.45}$$

where $z_i = r_1' r_2' ... r_i'$. Since the PDF of z_i is known, we can quantize c_i using PDF optimized quantization techniques. Assuming c_i is quantized by q levels, the quantization distortion at stage i can be written as:

$$E\{\|e_{qi}\|^2\} = \frac{\alpha_i \sigma_{c_i}^2}{q^2} \tag{3.46}$$

where α_i is a factor that depends on the PDF of c_i and the quantizer [27] and σ_{c_i} is the variance of c_i which can be found by:

$$\sigma_{c_i}^2 = \text{var}\left(\mathcal{R}^i z_i\right) = \mathcal{R}^{2(i)} \left(E(z_i^2) - E(z_i)^2\right) \tag{3.47}$$

$$\sigma_{c_i}^2 = t(N)^{2(i)} M^{\frac{-2(i)}{N-1}} \left(\left(\frac{N-1}{N+1} \right)^i - \left(\frac{N-1}{N} \right)^{2(i)} \right) \tag{3.48}$$

Equation (3.46) is the quantization distortion at each stage and can be used in equation (3.39) to find the total matching pursuit distortion at every stage.

As discussed in section 2.4.1, for asymptotically large N, the N dimensional vector of the form $\frac{X}{\sqrt{N\sigma^2}}$, with elements of vector X selected from a memoryless Gaussian source, lies on the surface of an N dimensional unit-sphere. However, in practical applications N cannot be arbitrarily large and the input vector is not localized on

the surface of a sphere. Therefore, the norm of the input vector has to be quantized and encoded before the MP encoder can be used. The input signal is normalized by this gain and the resulting normalized signal will be located on the surface of a unit sphere and can be quantized using the MP encoder. Quantization of the gain component causes additional distortion which can be written as:

$$D_{Q_g} = \frac{\alpha_g \sigma_g^2}{q_g^2} \tag{3.49}$$

where σ_g is the variance of the norm found by equation (2.10), q_g is the number of quantization levels and α_q is a parameter depending on the type of the quantizer and the distribution of the gain component found by (2.7).

We can further improve the rate-distortion performance of the MP encoder by combining the quantizers for the gain component and the inner product of the first stage. In fact, instead of sending two codewords for the gain component and the inner product coefficient of the first stage, we can find the inner product coefficient of the dictionary vector and the original input vector (instead of the normalized vector) and quantize the resulting inner product coefficient. Therefore, a single quantizer is required at the first stage. Moreover, since for not very large N and large dictionary sizes, the gain component is more dominant than the inner product coefficient of the first stage, we can approximate the quantization distortion at the first stage by equation (3.49). Therefore the total quantization distortion at the first stage can be written as:

$$E\{\|e_{q0}\|^2\} = \frac{\alpha_g \sigma_g^2}{q_g^2} \tag{3.50}$$

Now that $E\{\|e_{qi}\|^2\}$ is found for each stage, the total matching pursuit distortion can be found from the recursive formula in equation (3.39).

In order to achieve the best quantization strategy for inner product coefficients, a set of PDF optimized quantizers must be designed for quantizing the inner product coefficients at each stage. However, the distribution of the inner product coefficients

obtained by equation (3.27) is an approximate estimation of the actual PDF and its accuracy depends on the quality of the lattice used in the wrapped spherical codes (this will be shown in the simulation results presented in the next section). Therefore, in this chapter, we use a simple uniform quantizer for quantizing the inner product coefficients. In order to apply the characteristics of the distribution of coefficients which vary significantly for different stages, the uniform quantizer for each stage is centered around the mean of the inner product coefficients and its granular region is set proportional to standard deviation of coefficients of that stage. In other words, the granular region of the uniform quantizer is set to $(m_{c_i} - \alpha'_i \sigma_{c_i}, m_{c_i} + \alpha'_i \sigma_{c_i})$, where α'_i is a constant, m_{c_i} is the mean of the inner product coefficients at stage i and is computed by:

$$m_{c_i} = t(N)^i M^{\frac{-i}{N-1}} \left(\frac{N-1}{N}\right)^i \tag{3.51}$$

and σ_{c_i} is the standard deviation of the inner product coefficients at stage i which is derived in equation (3.48). In case of high resolution (q a large value), the high resolution analysis can be used and the values of α_i in equation (3.46) can be computed as $\alpha_i = \alpha'^2_i/3$ [55]. The quantization method described above is used to quantize the inner product coefficients throughout this chapter. Note that although uniform quantization is used, the range and center of the quantizers are different for different stages and are adapted to the distribution of the coefficients. Therefore, our quantization algorithm matches the distribution of the inner product coefficients at each stage and as will be shown in the next section our quantization scheme results in very efficient quantization of the inner product coefficients.

3.2.1 Convergence of quantized MP

As mentioned in section 2.5.1.1, the distortion of MP is guaranteed to converge to zero if an infinite number of stages are applied. Moreover there exists an exponentially

decreasing upper bound for the distortion at each stage. However, this is only true when quantization distortion is ignored. When a new quantization scheme is proposed for MP, its convergence must be studied in order to ensure a generally decreasing distortion. In this section we analyze the convergence of our proposed quantization scheme and prove that the MP distortion is upper-bounded by a decreasing function that is always larger than the exponentially decreasing upper bound for unquantized MP. According to equation (3.36), MP distortion at stage $i+1$ is equal to:

$$\|R^{i+1}f_q\|^2 = \|R^i f_q\|^2 r_{q(i+1)}^2 + \|e_{qi}\|^2 \tag{3.52}$$

where $r_{q(i+1)}$ is defined in equation (3.37). Based on the discussions following equation (3.37), the PDF of $r_{q(i+1)}$ is defined by equation (3.14) and therefore $r_{q(i+1)} \leq \mathcal{R}$. Hence:

$$\|R^{i+1}f_q\|^2 \leq \|R^i f_q\|^2 \mathcal{R}^2 + \|e_{qi}\|^2 \tag{3.53}$$

Therefore, in order to find an upper-bound for $\|R^{i+1}f_q\|^2$, we need to find the maximum value of the quantization distortion $\|e_{qi}\|^2$. According to our discussions in the previous section, our quantizer is a uniform quantizer with q quantization steps that partition the range between $(m_{c_i} - \alpha_i'\sigma_{c_i}, m_{c_i} + \alpha_i'\sigma_{c_i})$. Therefore, the maximum quantization error caused by our quantization scheme depends on the value of c_i.

1. If c_i is within the granular region of the quantizer, it can be easily verified that:

$$e_{qi} \leq \frac{\alpha_i'\sigma_{c_i}}{q} \quad c_{qi} \in (m_{c_i} - \alpha_i'\sigma_{c_i}, m_{c_i} + \alpha_i'\sigma_{c_i}) \tag{3.54}$$

2. If $c_i \leq m_{c_i} - \alpha_i'\sigma_{c_i}$, the quantized coefficient will be $m_{c_i} - \alpha_i'\sigma_{c_i}\left(1 - \frac{1}{q}\right)$ if midpoint reconstruction is used. Therefore:

$$e_{qi} = m_{c_i} - \alpha_i'\sigma_{c_i}\left(1 - \frac{1}{q}\right) - c_i \leq m_{c_i} - \alpha_i'\sigma_{c_i}\left(1 - \frac{1}{q}\right) \quad c_i \leq m_{c_i} - \alpha_i'\sigma_{c_i} \tag{3.55}$$

3. When $c_i \geq m_{c_i} + \alpha_i' \sigma_{c_i}$, the quantized coefficient is $m_{c_i} + \alpha_i' \sigma_{c_i} \left(1 - \frac{1}{q}\right)$ and therefore the quantization error can be computed as:

$$e_{qi} = c_i - m_{c_i} - \alpha_i' \sigma_{c_i} \left(1 - \frac{1}{q}\right) \leq \|R^i f_q\| - m_{c_i} - \alpha_i' \sigma_{c_i} \left(1 - \frac{1}{q}\right) \quad c_i \geq m_{c_i} + \alpha_i' \sigma_{c_i} \tag{3.56}$$

where the last inequality is based on the definition of c_i and the fact that dictionary vectors are normalized ($c_i = |\langle R^i f_q, g_{\gamma i} \rangle| \leq \|R^i f_q\|$).

Therefore, the maximum distortion at stage i can be written as:

$$\|R^{i+1} f_q\|_{\max}^2 = \max \begin{cases} \left(\frac{\alpha_i' \sigma_{c_i}}{q}\right)^2 + \|R^i f_q\|_{\max}^2 \mathcal{R}^2 \\ \left(m_{c_i} - \alpha_i' \sigma_{c_i} \left(1 - \frac{1}{q}\right)\right)^2 + \|R^i f_q\|_{\max}^2 \mathcal{R}^2 \\ \left(\|R^i f_q\|_{\max} - m_{c_i} - \alpha_i' \sigma_{c_i} \left(1 - \frac{1}{q}\right)\right)^2 + \|R^i f_q\|_{\max}^2 \mathcal{R}^2 \end{cases} \tag{3.57}$$

where $\|R^i f\|_{\max}$ is the maximum of $\|R^i f\|$. The above recursive equation can be used to find the upper-bound at each stage, if the upper bound at the first stage and the values of α_i' and q are known. In what follows, we show that for any function $C(i)$ (where i is the stage number), we can find α_i' and q such that $\|R^i f_q\|^2 \leq C(i)$ provided that $C(i+1) > C(i)\mathcal{R}^2$.

Lemma 2 *Suppose $C(i)$ is a function that satisfies $C(i+1) > C(i)\mathcal{R}^2$. If $\|R^i f_q\|_{max}^2 \leq C(i)$, there exists α_i' and q such that $\|R^{i+1} f_q\|^2 \leq C(i+1)$.*

Proof: Equation (3.57) can be used to find the conditions on α_i' and q that satisfy $\|R^{i+1} f_q\|^2 \leq C(i+1)$. Using the first term in equation (3.57), one can show that:

$$\frac{\alpha_i'}{q} \leq \frac{\sqrt{C(i+1) - \|R^i f_q\|_{\max}^2 \mathcal{R}^2}}{\sigma_{c_i}} \tag{3.58}$$

The second term in equation (3.57) implies that:

$$\alpha_i' \left(1 - \frac{1}{q}\right) \geq \frac{m_{c_i} - \sqrt{C(i+1) - \|R^i f_q\|_{\max}^2 \mathcal{R}^2}}{\sigma_{c_i}} \tag{3.59}$$

Finally, using the third term in equation (3.57), we can write:

$$\alpha_i' \left(1 - \frac{1}{q}\right) \geq \frac{\|R^i f_q\|_{\max} - m_{c_i} - \sqrt{C(i+1) - \|R^i f_q\|_{\max}^2 \mathcal{R}^2}}{\sigma_{c_i}} \tag{3.60}$$

The above inequalities are satisfied for any values of α_i' and q such that:

$$\alpha_i' \geq \max\left(\frac{m_{c_i}}{\sigma_{c_i}}, \frac{\|R^i f_q\|_{\max} - m_{c_i}}{\sigma_{c_i}}\right) \tag{3.61}$$

and

$$q \geq \frac{\alpha_i' \sigma_{c_i}}{\sqrt{C(i+1) - \|R^i f_q\|_{\max}^2 \mathcal{R}^2}} \tag{3.62}$$

Since $C(i+1) > C(i)\mathcal{R}^2$ and $\|R^i f_q\|_{\max}^2 \leq C(i)$, therefore $C(i+1) > \|R^i f_q\|_{\max}^2 \mathcal{R}^2$. Hence, according to equations (3.61) and (3.62), there exists finite real values for α_i' and q that guarantee $\|R^{i+1} f_q\|^2 \leq C(i+1)$. ■

Now suppose the input vector square norm is $\|f\|^2 \leq C(0) < \infty$. For any function $C(i)$ that satisfies $C(i+1) > C(i)\mathcal{R}^2$, lemma 2 can be used repeatedly to find α_i' and q for any stage (starting from $i = 0$) such that $\|R^i f_q\|_{\max}^2 \leq C(i)$. Note that $C(i+1) > C(i)\mathcal{R}^2$ implies that $C(i) > \mathcal{R}^{2i}\|f\|^2$ for all values of i. But $\mathcal{R}^{2i}\|f\|^2$ is the exponentially decreasing upper bound for MP distortion when quantization error is not considered (this can be easily verified by ignoring the quantization distortion terms in equation (3.57) and substituting $\|R^0 f_q\|_{\max}$ by $\|f\|$ and the fact that $\mathcal{R} < 1$ for all dictionaries used in this chapter). Therefore, it is expected that when inner product coefficients are quantized, only functions that are larger than the upper bound for unquantized MP distortion at every stage can provide an upper bound for quantized MP distortion.

3.2.2 Optimum dictionary size and quantization levels

The matching pursuit representation of signals consists of the dictionary indexes and inner product coefficients at all stages. Therefore, if the dictionary size is M, the

dimension of the input vector is N and the inner product coefficients are quantized by q levels, the rate of a k-stage MP encoder can be found by:

$$Rate = \frac{k}{N}(\log_2 q + \log_2 M) \tag{3.63}$$

and can be found for any value of N, k, M and q. Now, in order to find the best dictionary size M and number of quantization levels q, the following optimization problem has to be solved:

$$\begin{aligned} &\min D \\ \text{s.t.} \quad &\frac{k}{N}(\log_2 q + \log_2 M) = Rate \end{aligned} \tag{3.64}$$

where:

$$D = E\{\|R^k f_q\|^2\} \tag{3.65}$$

In this chapter, we use a fixed value for N which is chosen by the lattice used in the dictionary or the application requirement. The number of stages (k) determines the number of successive refinements performed on the input vector and is selected by the application requirements. When k is fixed, the best dictionary size and number of quantization levels must be found such that the resulting distortion is minimized. The possible choices for both q and M are integer values. Therefore the optimization problem can be solved by an exhaustive search over a limited number of integer values. Moreover, the rate budget constraint imposes a relationship between q and M. Thus, choosing a value for each of these two parameters yields a value for the other one according to equation (3.63). Since the range of possible values for q is smaller than the range for M, we search over the possible choices for q in order to find the optimal parameters that result in minimum distortion.

Now that we proposed an algorithm for finding the optimum values for q and M for a fixed number of MP stages, we can investigate the case when there is no requirement on the number of MP stages and k can be optimized such that the

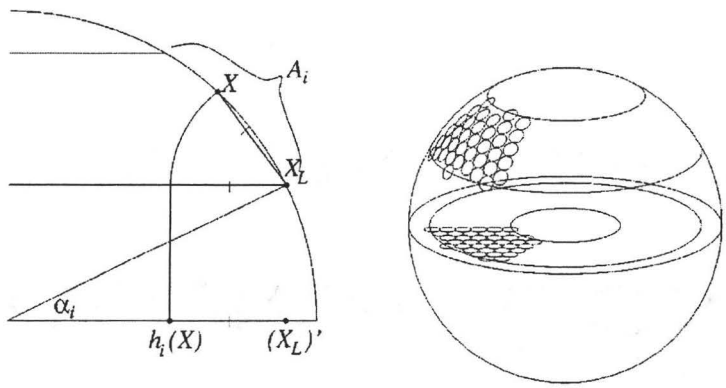
total distortion is minimum. Again, we employ an exhaustive search strategy and minimize the distortion for a range of possible values of k (the optimization algorithm described for a fixed k must be performed for all values of k). We performed the above optimization algorithm for different bit rates and input dimensions and for all our experiments, the optimum number of stages was found to be $k = 1$ (for one such example see section 3.3.2, Fig. 3.9). This means that single stage matching pursuit which is equivalent to gain-shape vector quantization is optimum for Gaussian signals. However, successive refinement is only achieved if $k > 1$ and for applications that need progressive coding, the rate-distortion performance must be sacrificed in order to achieve the desired structure in the output bitstream.

3.3 Simulation Results

In this section, first the dictionary used in our experiments is introduced. Then, the accuracy of our analytical study of MP encoder is verified by comparison to practical matching pursuit encoders. In the end, the application of our proposed MP encoder in progressively quantizing Gaussian signals is discussed and our MP encoder is compared with other quantizers designed for Gaussian sources.

3.3.1 Dictionary Design

As discussed in section 2.4.1, a vector that consists of samples selected from an independent Gaussian source is uniformly distributed on the surface of a sphere. Therefore, in order to efficiently represent this vector with an over-complete dictionary, the dictionary vectors must be uniformly distributed on the surface of a unit sphere. In this chapter, we use the wrapped spherical codes proposed in [36] to generate such dictionary. In wrapped spherical codes, a lattice in $N-1$ dimensions is mapped to the surface of an N dimensional sphere. In the following, we briefly review



(a) (b)

Figure 3.4: (a) Geometric interpretation of the wrapped spherical codes (b) Example in 3 dimensions (figures are taken from [37])

the wrapped spherical codes (for more detailed discussion see [36]).

Let Λ be a lattice in $\mathbb{R}^{N-1}$ and let the latitude of a point $X = (x_1, x_2, ..., x_N)$ be defined as $\sin^{-1}(x_N)$. Suppose $n+1$ latitudes are selected in the range $(-\pi/2, \pi/2)$:

$$-\pi/2 = \theta_0 < ... < \theta_n = \pi/2 \tag{3.66}$$

The i^{th} annulus is defined as the set of points satisfying:

$$A_i = \{(x_1, ..., x_N) \in \Omega_N : \theta_i \leq \sin^{-1} x_N < \theta_{i+1}\} \tag{3.67}$$

where Ω_N is the surface of an N dimensional sphere. For each $X \in A_i$, the point X_L which is the closest point to X and is located on the border between the i^{th} and $i-1^{th}$ annuli can be found by:

$$X_L = \arg\min_{Z}\{\|X - Z\| : Z = (z_1, ..., z_{N-1}, \sin\theta_i) \in \Omega_N\} \tag{3.68}$$

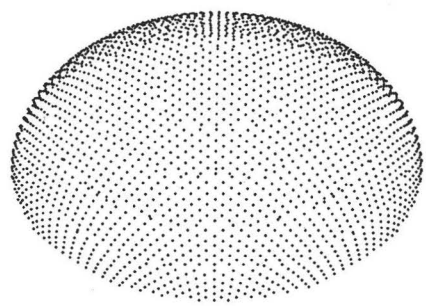


Figure 3.5: Dictionary generated by wrapped spherical codes

For each i, a mapping ϕ_i is defined from A_i to a subset of $\mathbb{R}^{N-1}$ by:

$$\phi_i(X) = \frac{X'}{\|X'\|}.(\|(X_L)'\| - \|X_L - X\|)_+ \tag{3.69}$$

where $(x)_+ \equiv \max(0, x)$ and prime notation denotes mapping from $\mathbb{R}^N$ to $\mathbb{R}^{N-1}$ by deleting the last coordinate. According to [36] the wrapped spherical codebook W_Λ with respect to lattice Λ is defined as:

$$W_\Lambda \equiv \bigcup_i \phi_i^{-1}(\Lambda \backslash \{0\}) \tag{3.70}$$

A pictorial representation of the above definitions is shown in Fig. 3.4 (for more details see [36,37]). To summarize the algorithm, in order to quantize a point $X = (x_1, ..., x_N)$ on the surface of the N dimensional unit sphere, first, the annulus number i must be found satisfying $\theta_i \leq sin^{-1}x_N < \theta_{i+1}$. The second step is to calculate the $N-1$ dimensional vector $\phi_i(X)$ according to (3.69). Then, $\phi_i(X)$ is quantized to $\hat{\phi}_i(X)$ by the $N-1$ dimensional lattice Λ and finally the codevector $\phi_i^{-1}(\hat{\phi}_i(X))$ is computed.

A sample dictionary generated by wrapped spherical codes is shown in Fig. 3.5. Note that the efficiency of the dictionary in uniformly partitioning the surface of a sphere, directly depends on the choice of the $N-1$ dimensional lattice. This is because the surface of the unit sphere is partitioned by this lattice and the input

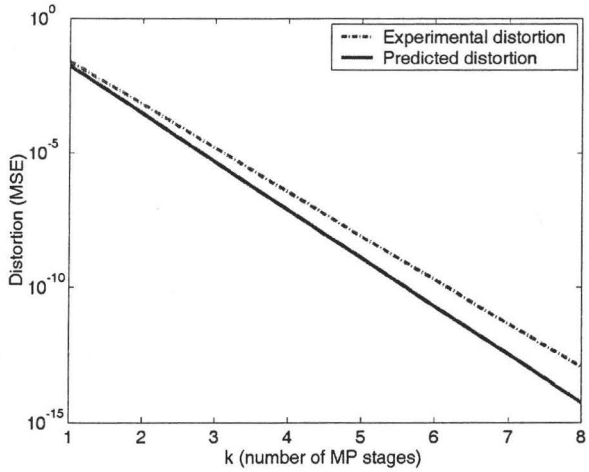


Figure 3.6: Comparison of distortions computed by equation (3.31) and the distortion obtained through experiment for $N = 25$ and $M = 2^{72}$. The cubic lattice is used in the wrapped spherical code

vectors will be quantized by this lattice (or more accurately by the mapping of the $N-1$ dimensional lattice on the surface of the sphere). Therefore, in order to achieve the best performance with the wrapped spherical codes, a high quality lattice must be used to partition the surface of the unit sphere.

3.3.2 Verification

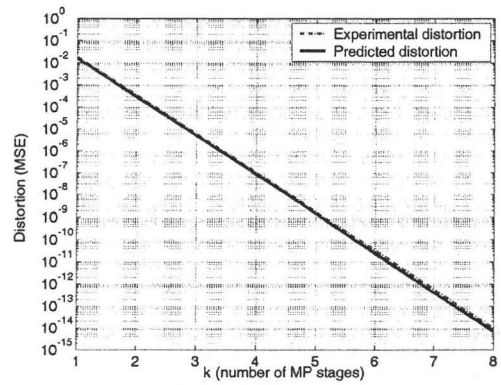
In this section, our mathematical analysis is verified by comparing the rate-distortion performance predicted by our equations and what is obtained by experimental data. The dictionary used in all our experiments is designed based on the algorithm described in section 3.3.1. The input signal is generated by blocking i.i.d Gaussian random samples into N dimensional vectors and then normalizing the vectors. 1000 N-dimensional vectors generated as described above are encoded by the MP encoder and the experimental data is computed by averaging over the results for

all 1000 vectors.

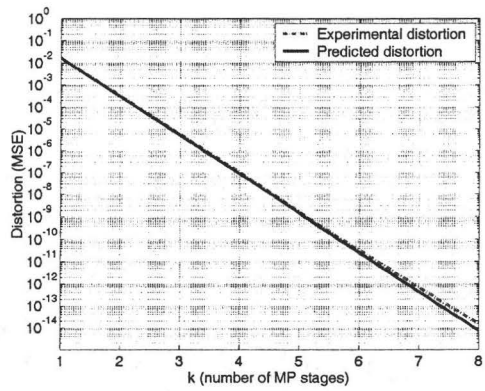
Fig. 3.6 shows the distortion of the output of an MP encoder for different number of matching pursuit stages. The dimension of the input vector is 25 and the dictionary size is $M = 2^{72}$. The quantization error is not considered in this experiment and will be discussed later in this section. The 24 dimensional cubic lattice ($\mathbb{Z}^{24}$) is used in the wrapped spherical code to quantize the 24 dimensional vector $\phi_i(X)$. As can be seen in the figure, the distortion calculated by equation (3.31) approximately predicts the distortion computed using experimental data. The main reason for discrepancies is the approximation of the Voronoi regions by spheres in our analysis. The Voronoi regions in the $\mathbb{Z}^{24}$ lattice are cubical. However, in our equations they are modeled by spheres and the distortion is calculated for the spherical Voronoi regions. Since this error exists in all stages, as the number of stages increases, the error accumulates and causes more error in our prediction. Therefore, we can conclude that using a lattice whose Voronoi regions are closer to the shape of a sphere will cause our experimental data to be closer to the predicted distortion. This is done in Fig. 3.7.(a) where a 24-dimensional Leech lattice is used instead of the cubic lattice in the wrapped spherical code. The Leech lattice provides the best known packing of space in 24 dimensions. Thus, the shapes of its Voronoi regions are closer to the shape of a sphere. Therefore, our experimental results should be closer to our analysis results when the Leech lattice is used instead of the cubic lattice. As can be seen in Fig. 3.7.(a), the predicted distortion is a perfect approximation of the experimental distortion which proves the accuracy of our assumptions in section 3.1.

Comparison of Fig. 3.6 and Fig. 3.7.(a) shows that the choice of lattice in the wrapped spherical codes has a great impact on the rate-distortion performance of the MP encoder. Therefore in this chapter we use the Leech lattice in wrapped spherical codes to achieve the best partitioning of space in 24 dimensions.

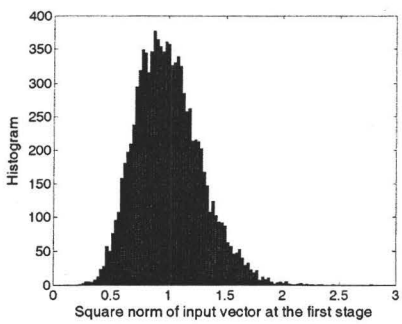
Fig. 3.7.(b) performs the same experiment on quantized matching pursuit. The



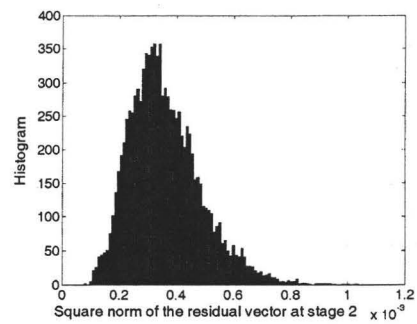
(a)



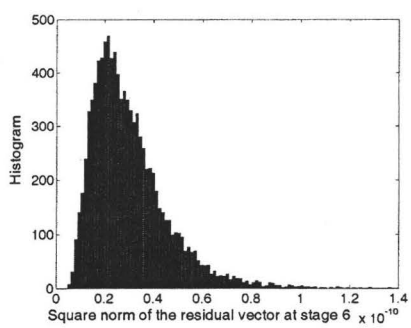
(b)



(c)



(d)



(e)

Figure 3.7: (a) and (b) Comparison of distortions obtained by experiments and the distortions computed by (a) equation (3.31) and (b) equation (3.39). (c) The histogram of square norm of the input vector. (d) and (e) The histogram of the square norm of residual vectors at stages 2 and 6 respectively. In all figures $N = 25$, $M = 2^{72}$, $q = 16$ and the Leech lattice is used in the wrapped spherical code.

inner product coefficients are quantized by the method described in the previous section using 4 bits for each stage. The corresponding quantization errors are considered in both the experimental and analytical graphs. As can be seen in the figure, the distortion predicted by our approximate equations is very close to the distortion obtained in practice which proves the accuracy of our approximation for quantization error. Also note that the distortions obtained in quantized matching pursuit are very close to the distortions when the inner product coefficients are not quantized. The quantization error results in only a slight increase in the distortion in both the theoretical and experimental distortions. This shows that our assumption in section 3.2 that quantization does not significantly change the inner product coefficients is valid. In section 3.2, we assumed inner product coefficients are quantized at a very high resolution and therefore quantization does not significantly change the characteristics of inner product coefficients. However, in this experiment we only use 4 bits for quantization for all stages and the reason for accurate quantization is mostly the fact that our quantization scheme is very efficient and matches the distribution of coefficients at every stage. In fact although the number of quantization levels is fixed and is not too high for all quantizers at every stage, the quantizer at each stage is different from other stages since both the granular region and the step size are adjusted according to the mean and variance of the inner product coefficients at that particular stage. The histogram (PDF) of the square norm of the input vector is shown in Fig. 3.7.(c) for 10000 input vectors. This figure shows how the norm of the input vector is distributed around its mean (see equation (2.7)). The histogram (PDF) of the square norm of the residual vectors (which is equivalent to MP distortion) at stages 2 and 6 are shown in Fig. 3.7.(d) and (e) (histograms are obtained using 10000 input vectors). These figures show how the distortions are distributed around their means (plotted in Fig. 3.7(b)) and converge to zero as the number of stages increase.

Fig. 3.8 shows how the quantizers at each stage are adapted to the distribution

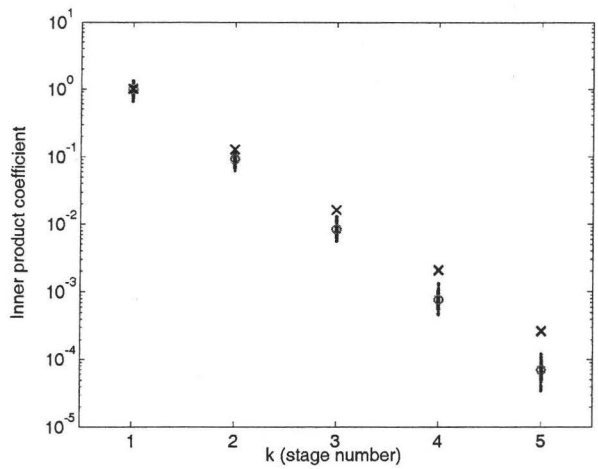


Figure 3.8: The dots show the absolute value of inner product coefficients for 100 tests performed on 25 dimensional vectors with a dictionary with 2^{84} vectors. The circles show the means obtained by equation (3.51) and the crosses show the exponentially decreasing upper bound for the inner product coefficients.

of inner product coefficients. In this figure, the dots represent the actual values of inner product coefficients at each stage obtained by 100 different tests performed on 25 dimensional vectors consisting of Gaussian samples (Each dot represents the value obtained in one test). The circles show the mean of the inner product coefficients for each stage obtained in equation (3.51). Fig. 3.8 shows that equation (3.51) is a very accurate approximation of the mean of inner product coefficients. Moreover, since the inner product coefficients have much higher probability of occurrence in the areas around the average values, centering the quantizers around these means will result in efficient quantization of inner product coefficients.

As shown in Fig. 3.8, the mean values of inner product coefficients exponentially decrease as the number of stages increases (This can also be seen in equation (3.51)). This result is in agreement with the fact that the upper bound of inner product coefficients exponentially decreases with the number of MP stages for unquantized MP.

Also shown in the figure is the well known exponentially decreasing upper-bound of the inner product coefficients derived in equation (2.19) (the cross points). The value of this upper bound is found using the algorithm described in [40] for the dictionary used in this experiment. This upper-bound is found in [38] and is often used in matching pursuit encoders to provide efficient quantization [23, 40, 41, 54]. As an example, in [41] the range between zero and this upper bound is quantized using a variable number of quantization levels at each stage. As shown in Fig. 3.8 our quantization scheme is much more efficient than the ones that only use the exponentially decreasing upper-bound since the range of our quantizers only covers the areas in which inner product coefficient have higher probability of occurrence. Therefore, our quantization scheme results in finer quantization and better performance for Gaussian signals. Moreover, since we explicitly find the distortion in terms of encoder parameters such as dictionary size, signal dimension and number of quantization levels, our optimization method is more accurate than the ones that only use the upper bound for the residual vectors to find the distortion of matching pursuit (e.g. [40, 41]).

Fig. 3.9 compares 4 different MP encoders with different number of stages. All encoders operate at the rate of 7 bits per sample and the dimension of the input vector is $N = 25$. The Leech lattice is used in the wrapped spherical code. The vertical axis is the distortion and the horizontal axis is the number of MP stages used in each encoder. The number of MP stages are different for each encoder and the dictionary sizes are adjusted such that all encoders operate at the rate of 7 bits per sample. For a k stage encoder operating at 7 bits per sample the dictionary size can be approximately found by:

$$M = 2^{\frac{7(N-1)}{k}} \tag{3.71}$$

where $N = 25$ in this experiment. The above equation and equation (3.63) mean that $\frac{24}{25}$ of the bit rate is assigned to the dictionary (which partitions the surface of a 25 dimensional sphere) and $\frac{1}{25}$ of the bit rate is allocated to the quantization of inner

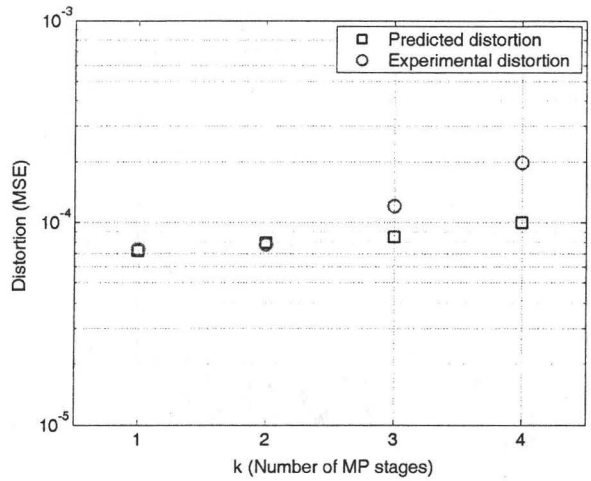


Figure 3.9: The distortion of different MP encoders with different number of stages operating at the rate of 7 bits per sample

product coefficients. The dictionary sizes obtained by our optimization algorithm and the ones found to be optimum in practice are both very close to the dictionary sizes found using the above equation. Therefore, the dictionary sizes of the 4 encoders used in this experiment can be found from the above equation. This experiment proves that in order to achieve the best performance by matching pursuit encoder, the input vector has to be quantized in only one stage and as the number of stages increases, the output distortion increases. Both experimental and analytical graphs prove this statement. However, the rate of distortion increase of the analytical results is smaller than the experimental results. The reason for this is that for large number of MP stages, the dictionary size has to be relatively small. The wrapped spherical codes are only efficient when relatively large dictionary sizes are chosen and as the dictionary size decreases the efficiency of the wrapped spherical codes in uniformly partitioning the surface of a sphere decreases. This makes some of our assumptions less accurate (e.g. approximation of the Voronoi regions by spheres). Therefore,

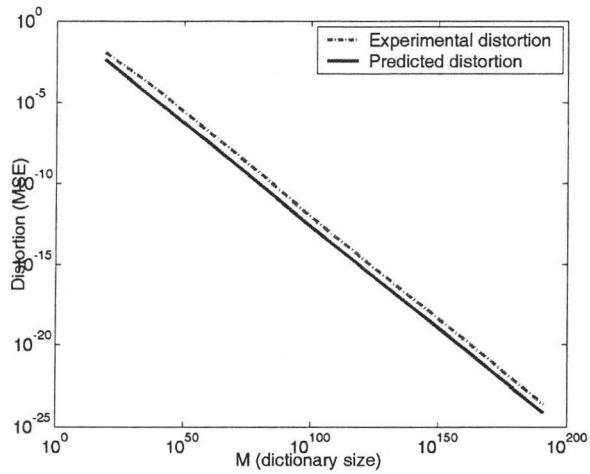


Figure 3.10: Comparison of distortions computed by equation (3.31) and the experimental distortion for $N = 64$ and $k = 4$

for small dictionary sizes our prediction of distortion is less accurate. Nevertheless, the analytical graph shows that the reduction in performance, although small, is inevitable when multi stage matching pursuit is applied. This is the price for having a progressive coding of the input.

Finally, Fig. 3.10 compares the distortion obtained by experimental data and the one calculated by equation (3.31) for different dictionary sizes. In this experiment the dimension of the input vector is $N = 64$, the cubic lattice is used in the wrapped spherical code and the number of MP stages is $k = 4$. The computed distortion is a good approximation for the distortion obtained by experimental data which proves the accuracy of our analysis. In this experiment, the discrepancies are due to the use of the cubic lattice and the approximation of the Voronoi regions.

3.3.3 Application in Gaussian Source Coding

As mentioned in section 2.4.1, a vector consisting of samples of an i.i.d memoryless Gaussian source is localized on the surface of a sphere, if the dimension of the vector is sufficiently large. Considering the structure of our MP dictionary, the MP encoder is an immediate candidate for progressively encoding the Gaussian source. The implementation details of our Gaussian source encoder is as follows:

25 samples of an i.i.d memoryless Gaussian source are blocked into 25-dimensional vectors and this vector forms the input to the MP encoder. The wrapped spherical codes are used as the matching pursuit dictionary as discussed in section 3.3.1. The 24-dimensional Leech lattice is applied in the wrapped spherical codes. The quantization method is based on discussions in section 3.2 and the number of quantization levels and dictionary size are obtained by the method described in section 3.2.2. The values of α_i' are selected such that they provide a reasonable balance between low average distortion and rapid convergence rate. This is done by finding the best α_i' values for each stage experimentally such that minimum average distortion is obtained at each stage. If the number of training inputs are large, this approach results in a good tradeoff between minimum average distortion and rapid convergence rate. We used 25000 Gaussian samples at each stage in order to find optimum values of α_i'. This generally results in larger values for α_i' for higher stages which is mainly the result of consideration for convergence. Our simulation results are shown in Table 3.1. In order to compute the signal to noise ratios, the SNR for 1000 25-dimensional vectors is calculated. The SNR computation is performed 10 times and the average of the resulting SNR values is reported in Table 3.1. Our MP encoder is compared to some of the best known quantizers designed for quantizing Gaussian sources and the results are shown in Table 3.1.

The first row in Table 3.1 shows the results when a 3-stage matching pursuit encoder quantizes the source at 6 bits per sample. Each stage contributes 2 bits per

Method Rate:	1	2	3	4	5	6	7
MP (2 bits per stage)		11.06		22.01		32.87	
MP (3 bits per stage)			17.35			34.21	
MP (4 and 2 bits for stages 1 and 2)				23.36		33.75	
MP (3, 2 and 2 bits for stages 1, 2 and 3)			17.35		28.28		38.32
Shape-gain VQ using Leech lattice [37]	2.44	11.02	17.36	23.33	29.29	35.27	41.33
TB-SVQ (4 state) [42]	5.39	11.18	16.92				
TB-SVQ (32 state) [42]	5.49	11.28	17.05				
Wilson (128 state) [43]	5.47	10.87	16.78				
TCQ (256 state) [44]	5.56	11.04	16.64				
TCQ (2D, 16 state) [44]	5.29	10.84	16.62	22.63			
Entropy coded scalar quantizer [37,49]	4.64	10.55	16.56	22.55	28.57	34.59	40.61
Z^{16} lattice [46]		10.07	15.52	21.00	26.16	32.07	37.68
Unrestricted polar quantizer [47]	4.40	9.63					
Lloyd-Max Scalar [37,49]	4.40	9.30	14.62	20.22	26.02	31.89	37.81
Uniform scalar [48]	4.40	9.25	14.27	19.38	24.57	29.83	35.13

Table 3.1: Comparison of different quantizers designed for Gaussian source with MP encoder proposed in this chapter. Only the uniform scalar and the Lloyd-Max scalar quantizers are scalable (The information in this table is obtained from [37]).

sample. Thus, the resulting embedded bitstream can be used to decode data at the rates of 2, 4 and 6 bits per sample. The signal to noise ratios obtained at each of these bit rates are also shown in the same row of the table. In the second example, the MP encoder is configured so that each stage contributes 3 bits per sample. Therefore the 6 bit per sample bitstream can be used to decode data at the rates of 3 and 6 bits per sample. The signal to noise ratios at 3 and 6 bits are shown in the second row of the table. The third and forth rows of the table show the case when different rates are chosen for different stages. This case is especially important when we need very good quality at the base rate and would like to have the capability of increasing the quality whenever extra rates are available. In the third row, the first stage contributes 4 bits and the second stage contributes 2 bits per sample. As can be seen in the table, in this configuration, we achieve very good performance at 4 bits per sample and are still capable of using the same bitstream and add the next stage to achieve the 6 bits

per sample bit stream. Note that we could achieve an embedded bitstream which could be decoded at the rates of 4 and 6 bits per sample using the configuration in the first row. However, since this configuration has an extra matching pursuit stage for decoding at 2 bits per sample, the performance is worse than the configuration in the third row. The configuration in the fourth row produces a bitstream that can be decoded at 3, 5 and 7 bits per sample. The SNR values are comparable to other quantizers at lower bit rates and the results are always better than the quantizers that can produce embedded bitstreams (scalar and Lloyd Max scalar quantizers).

Comparing the MP encoder with other quantizers designed for Gaussian sources, one can see that, the performance of MP encoder is comparable to the best known quantizers for small number of stages $k < 3$. In fact, for a single stage matching pursuit, our results are very close to the gain shape VQ using Leech lattice [37]. This is not surprising since in this case both quantizers are essentially the same except that in matching pursuit, the inner product of the input vector and the dictionary element is encoded along with the dictionary index while in [37], the norm of the input vector is quantized as the gain component and is encoded along with the dictionary index. Our experiments showed that encoding the inner product coefficient rather than the input norm results in a slight improvement. However, since this improvement is not very significant, we can conclude that the MP encoder is the same as gain shape vector quantizer when the number of matching pursuit stages is one. As can be seen in the table, when the number of MP stages is one (results for gain-shape VQ using Leech lattice), the performance is comparable to all existing quantizers. As the number of MP stages increases, the performance of the MP encoder decreases in comparison to other quantizers including the gain-shape VQ. However, this is the price we pay to achieve an embedded bitstream and as discussed in section 3.3.2 the reduction in performance is the direct consequence of applying multi-stage matching

pursuit. We can always set the number of MP stages to one to achieve the best rate-distortion performance. In this case the resulting bitstream would not be embedded anymore. As for progressive coding, none of the quantizers shown in Table 3.1 produce embedded bitstreams except for the Lloyd Max and uniform scaler quantizers. Table 3.2 compares our proposed MP encoder with some of the quantizers designed for progressively quantizing Gaussian sources [19, 20, 49]. Also shown in the table is the rate distortion function of Gaussian sources ($D(R)$). As can be seen in the table, the progressive code generated by the MP encoder has higher SNR values than the SNR values for other quantizers that generate embedded bitstreams. The disadvantage of MP encoder is that due to poor performance of wrapped spherical codes at low bit rates, successive refinement by less than 2 bits per sample is not very efficient using our MP encoder whereas other quantizers listed in Table 3.2 can be configured to operate at lower bit rates as well. Nevertheless, the performance improvement achieved by MP is significant in the relatively high bit rates used for successive refinement.

		MP		TS-TCQ [19]		SR-TCVQ [19]		MS-TCQ [20]		Lloyd-Max [49]		D(R)	
R1	R2	D1	D2	D1	D2	D1	D2	D1	D2	D1	D2	D1	D2
2	2	11.06	22.01	10.55	21.55	10.54	21.69	10.24	19.70	9.30	20.22	12.04	24.08
3	2	17.35	28.28	-	-	-	-	15.89	25.51	14.62	26.02	18.06	30.10

Table 3.2: Comparison of different successively refinable quantizers designed for Gaussian source with MP encoder proposed in this chapter. The table shows the SNR values for different bit rates whenever such data is available. R1, R2, D1 and D2 are the rates and distortions at stages 1 and 2 respectively.

3.4 Conclusion

In this chapter, we studied the application of matching pursuit for progressively encoding samples from memoryless i.i.d Gaussian sources. The accuracy of our analytical study is verified by experimental data and we showed how our proposed matching

pursuit encoder can be applied to generate embedded bitstreams for Gaussian inputs. Our theoretical analysis shows that progressive encoding of the input signal using MP results in an increase in output distortion compared to non-embedded encoders operating at the same rate. This is the price for generating embedded bitstreams. Our experimental results are in agreement with this theoretical result. Nevertheless, our MP encoder outperforms existing quantizers that can produce embedded bitstreams for Gaussian sources.

Chapter 4

Design and Analysis of a Successively Refinable Lattice Quantizer for i.i.d. Gaussian Sources

As shown in Fig. 3.9 in the previous chapter, multiple stage matching pursuit is not able to find a rate distortion optimal progressive code for Gaussian sources. The main reason is that the same dictionary is used to quantize all residual vectors and the dictionary vectors have the same dimension as the input vector. However, as discussed in the previous chapters, the residual vector is orthogonal to the dictionary vector and therefore its dimensionality is one dimension less than the original vector. This redundancy in the dictionary is the main reason for the reduced performance of MP compared to non-progressive coding. In this chapter we solve this problem by quantizing the residual vectors in the space orthogonal to the dictionary vector and

eliminate the redundancy in the dictionary. We compare our results with other successively refinable quantizers designed for Gaussian sources and show that significant improvements are achieved by the proposed algorithm. We also compare the results with the MP encoder and discuss the advantages and disadvantages of each method.

4.1 The proposed successively refinable quantizer

In this chapter, first we quantize the N dimensional input vector X using the shape-gain quantizer designed in [37]. The wrapped spherical codes [37] are used to generate a codebook with vectors uniformly distributed on the surface of the unit sphere. In practical cases, N cannot be arbitrarily large and the input vector X is not localized on the surface of a sphere. Therefore, in [37] a shape-gain quantizer is designed and the normalized shape and the norm of the input vector are quantized independently[1]. In this chapter, in order to provide successive refinement, instead of encoding the norm of the input vector, we quantize and encode the inner product of the input vector and the codevector. Thus, the distance between the reconstructed vector and the input vector is minimized. Let g be the shape vector obtained from a spherical codebook operating at rate R_{s1} i.e. g is the vector in the codebook that has the maximum inner product coefficient with the input vector X. Therefore:

$$X = \langle X, g \rangle g + X_1 \tag{4.1}$$

where X_1 is the residual vector at stage 1. We define the gain component as $X_g = \langle X, g \rangle$ and the shape component is defined as $X_s = \frac{X}{\langle X,g \rangle}$ (note that $X = X_g.X_s$). In order to provide successive refinement, we refine the gain and shape components at

[1]The quantizer in [37] is rather different from conventional gain shape vector quantizers [52] in a sense that it normalizes the input vector before finding the shape vector and the computation of shape vector and gain component are done independently. The main reason for this is to ensure the input vector is positioned on the surface of the sphere and therefore the lattice quantizer can use the nearest neighbor methods in order to find the closest lattice point to the input vector.

each refinement stage. The gain component is refined using a scalar quantizer and at each stage the quantization step size is decreased. The resulting gain component at stage i is denoted as $Q_i[\langle X, g\rangle]$. At the first stage the shape component is quantized by the wrapped spherical codes resulting in the reconstruction vector g. In order to provide refinements for the shape component the residual shape vector $X_{1n} = X_s - g = \frac{X}{\langle X,g\rangle} - g$ is quantized further by a lattice quantizer operating at rate R_{s2}[2]. The resulting quantized vector is denoted as X_{1n_q}. At the next refinement stage the normalized residual vector is $X_{2n} = X_{1n} - X_{1n_q}$ and is quantized by another lattice operating at rate R_{s3}. Therefore, after j refinement stages the reconstructed vector X_q can be found by:

$$X_q = Q_j[\langle X, g\rangle](g + X_{1n_q} + X_{2n_q} + ... + X_{(j-1)n_q}) \tag{4.2}$$

In order to quantize X_{1n} using a lattice quantizer, the range of X_{1n} must be found and lattice points must be distributed in this range. As mentioned before, $X_{1n} = X_s - g = \frac{X_1}{\langle X,g\rangle}$. Since g is normalized $\langle X_1, g\rangle = 0$ and therefore $\langle X_{1n}, g\rangle = 0$. Thus X_{1n} is located on the $N-1$ dimensional hyperplane which is orthogonal to vector g and is tangent to the unit sphere at vector g (see Fig. 4.1). Thus, the range of X_{1n} is the mapping of the Voronoi cell of vector g to the hyper-plane that is orthogonal to g and is tangent to the unit sphere at g (see Fig. 4.1). For high resolution dictionaries, this range will be very similar to the Voronoi cell of vector g. X_{1n} is quantized by a lattice that partitions this range. The resulting quantized vector is X_{1n_q}. The remaining residual shape vector $X_{2n} = X_{1n} - X_{1n_q}$ is located in the Voronoi cell of X_{1n_q} and is on the same $N-1$ dimensional hyperplane as X_{1n}.

[2]Note that after the shape component and the inner product (gain component) are calculated by the wrapped spherical codes, the input vector is divided by the gain component. Although division is usually not desired for computational complexity reasons, the cost of division is negligible if a high complexity lattice (like Leech lattice) is used in the wrapped spherical codes or in the following stages. Moreover, the implementation of wrapped spherical codes requires a number of divisions and thus an extra division for computing the shape component does not heavily affect the computational complexity of the algorithm.

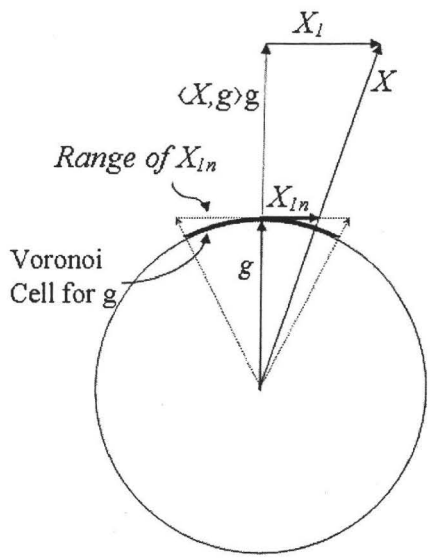


Figure 4.1: Geometric interpretation of vectors and regions defined in this chapter for $N = 2$

Therefore this region can be further partitioned using lattice Λ_2 in order to provide a refinement. This process continues for as many refinement stages as needed.

In order to use a single lattice defined in a single $N-1$ dimensional hyperplane for all vectors g, the hyperplane orthogonal to vector g which includes vectors X_{in}, must be mapped to the $N-1$ dimensional hyperplane in which the lattice is defined. In this chapter we defined the lattice in the hyperplane $C_R = \{X|X = (x_1, ..., x_N), x_N = 0\}$. Therefore all vectors (i.e. X_{in}) must be mapped to this plane before being quantized by the lattice. The quantized value must be mapped back to the original hyperplane which is orthogonal to vector g. Let

$$C = \{Y|\langle Y, g\rangle = 0, \|Y\| \leq \|Y + g - g_i\|, \forall g_i \in D\} \tag{4.3}$$

where D is the codebook. Hence, C is the range of vector X_{1n}. Let

$$L = \{Z | Z = (z_1, ..., z_N) \in C, z_N = 0\} \tag{4.4}$$

We find X_L which is the closest point to X_{in} and is located on L (see Fig. 4.2).

$$X_L = \text{argmin}_Z\{\|X_{in} - Z\| : Z \in L\} \tag{4.5}$$

Let the prime notation denote mapping from the N dimensional space to the $N-1$ dimensional space by setting the N^{th} coordinate to zero. Thus, X_L can be found by:

$$X_L = X'_{in} - \langle X'_{in}, \frac{g'}{\|g'\|}\rangle \frac{g'}{\|g'\|} \tag{4.6}$$

Now the one to one mapping m from C to $C_r \subset C_R$ is defined by:

$$m(X) = X_L + \frac{X'_{in} - X'_L}{\|X'_{in} - X'_L\|}\|X_{in} - X_L\| \tag{4.7}$$

The inverse mapping can be found by finding X_{L_q} from m_q (the quantized value of m).

$$X_{Lq} = m'_q - \langle m'_q, \frac{g'}{\|g'\|}\rangle \frac{g'}{\|g'\|} \tag{4.8}$$

The vector $m_{qC} \in C$ is found such that:

$$m_{qC} = m_q - \langle m_q, g\rangle g \tag{4.9}$$

The quantized value of X_{in} can now be found by:

$$X_{in_q} = X_{Lq} + \frac{m_{qC} - X_{Lq}}{\|m_{qC} - X_{Lq}\|}\|m_q - X_{Lq}\| \tag{4.10}$$

A geometric interpretation of the above method is shown in Fig. 4.2. This mapping is used at each stage to quantize the N dimensional vector X_{in} $(i > 0)$ by a lattice on the $N-1$ dimensional hyperplane $x_N = 0$.

Once the N dimensional vector is mapped to the $N-1$ dimensional hyperplane $x_N = 0$, it must be quantized by a lattice that partitions the range of the vector. As

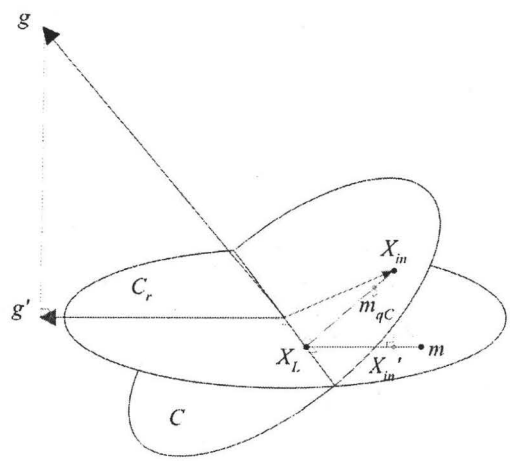


Figure 4.2: Geometric interpretation of the mapping from C to C_r and its inverse mapping

mentioned before the range of the vector X_{1n} is the mapping of the Voronoi cell of vector g to the hyperplane orthogonal to g. The Voronoi cell of g is the mapping of the Voronoi region of the lattice used in the wrapped spherical codes to the surface of the sphere. Consequently, the range of X_{1n} is a function of the Voronoi cell of the lattice used in the wrapped spherical codes (in the case of high resolution this range is almost equal to the Voronoi cell of g). Since X_{1n} is quantized by a lattice, the range of the residual vector X_{2n} is the Voronoi cell of the lattice used to quantize X_{1n} and this region must be partitioned by another lattice in order to provide a refinement. But the Voronoi regions of most lattices have irregular shapes and there is no simple way for efficiently partitioning this region using another lattice. In fact as shown in [51], even for regular lattices in lower dimensions, efficient subdividing is not possible except for the cubic lattice (which has the worst performance). Additionally, since the Voronoi region is partitioned by another lattice to provide successive refinement, no overload error can be tolerated and all errors (i.e. residual vectors) must be within the Voronoi region in order to be able to provide a refinement in the subsequent stages.

Therefore, in order to solve the above problems, we only use lattice points that are inside a sphere. In other words, the lattice which is partitioning the Voronoi region is now defined in a spherical volume. In order to minimize the overload error we need to ensure the entire Voronoi region is covered by the lattice partitioning the Voronoi region. This can be achieved by locating a lattice in the volume of a sphere with a radius that is equal to the covering radius of the lattice used in the previous stage. If this area is partitioned by a lattice all residual vectors will be inside this volume and therefore no overload error will occur. Thus, the irregularly shaped ranges of the residual vectors are efficiently partitioned while no overload error will occur due to partitioning the entire volume of the Voronoi region using a sphere with a radius equal to the covering radius of the lattice.

4.2 High resolution analysis

In this section a high resolution analysis is presented for our successively refinable lattice quantizer and we assume all the refinement rates are asymptotically large $(R_1, ..., R_j \rightarrow \infty)$. At each stage i, R_{s_i} bits are assigned to the shape component and R_{g_i} bits are assigned to the gain component. At each stage $R_i = R_{s_i} + R_{g_i}$ and all rates are asymptotically large $(R_{s_i} \rightarrow \infty, R_{g_i} \rightarrow \infty \text{ for } 1 \leq i \leq j)$.

The distortion of the quantizer designed in [37] has been calculated in the case of high resolution $(R_{s_1} \rightarrow \infty, R_{g_1} \rightarrow \infty)$ when the wrapped spherical codes are used to design the codebook. In this section we compute the rate distortion function achieved by our proposed successive refinement algorithm. We compare this rate distortion function with that of the quantizer of [37] and compute the distortion increase caused by successive refinement. In [37] it is proven that the total distortion of the shape-gain quantizer is the summation of the distortion caused by quantizing the gain component (gain distortion) and the distortion caused by quantizing the shape component (shape

distortion). Since for the case of high resolution $\langle X, g\rangle \approx \|X\|$, our proposed quantizer will be equivalent to the one designed in [37] with the exception that our quantizer provides scalability. Therefore, the distortion of our proposed quantizer is also equal to the summation of the gain and shape distortions. Note that in our quantizer the gain distortion per dimension is $D_g = \frac{1}{N}E\left[(\langle X, g\rangle - Q_j[\langle X, g\rangle])^2\right]$ and the shape distortion per dimension is $D_s = \sigma^2 E\left[\|\frac{X}{\langle X,g\rangle} - (g + X_{1n_q} + X_{2n_q} + ... + X_{(j-1)jn_q})\|^2\right]$ and therefore the total distortion per dimension is $D = D_s + D_g$ (for more details see [37]). According to [37], the shape distortion of the wrapped spherical code is:

$$D_{s_1} = (N-1)\sigma^2 G(\Lambda) V_1(\Lambda)^{\frac{2}{N-1}} \tag{4.11}$$

where $V_1(\Lambda)$ is the volume of the Voronoi region of the lattice used in the wrapped spherical code, $G(\Lambda)$ is the normalized second moment of lattice Λ and is given for the best known lattices in [31] and N is the dimension of the input vector. $G(\Lambda)$ is defined as:

$$G(\Lambda) = \frac{\frac{1}{N-1}\int_{\Pi_1} \|t\|^2 dt}{V_1(\Lambda)^{1+\frac{2}{N-1}}} \tag{4.12}$$

where Π_1 is the Voronoi region of the lattice. If the bit rate allocated to the shape component at the first stage is R_{s_1}, $V_1(\Lambda)$ can be found by:

$$V_1(\Lambda) = \frac{S_N}{2^{NR_{s_1}}} \tag{4.13}$$

where S_N is the surface area of an N dimensional sphere. Therefore, the shape distortion at the first stage can be written as:

$$D_{s_1} = (N-1)\sigma^2 G(\Lambda) S_N^{\frac{2}{N-1}} 2^{-\frac{2N}{N-1}R_{s_1}} \tag{4.14}$$

The normalized residual vector X_{1n} is located in the mapping of the volume of the Voronoi region of lattice Λ (i.e Π_1) on the $N-1$ dimensional hyperplane that is orthogonal to vector g. However, for high resolution dictionaries the Voronoi region of Λ is almost on the $N-1$ dimensional hyperplane and therefore the range of X_{1n} is

almost identical to the Voronoi region of lattice Λ. In order to provide a refinement, we partition the sphere on the $N-1$ dimensional hyperplane that is tangent to the unit sphere at point g by the scaled version of the lattice Λ at rate NR_{s_2}. The center of this sphere is at g and its radius is equal to the covering radius of the lattice which is denoted as R_{c_1}. R_{c_1} can be found by [31]:

$$\Theta = \frac{v_{N-1}R_{c_1}^{N-1}}{V_1(\Lambda)} \tag{4.15}$$

In equation (4.15), Θ is the thickness of the lattice Λ, $V_1(\Lambda)$ is the volume of the Voronoi region and v_{N-1} is the volume of the $N-1$-dimensional unit sphere which can be found by:

$$v_{N-1} = \frac{\pi^{\frac{N-1}{2}}}{(\frac{N-1}{2})!} \tag{4.16}$$

The $N-1$ dimensional sphere with radius R_{c_1} is partitioned at the rate NR_{s_2}. Thus, if the Voronoi region of the resulting lattice is denoted by Π_2, the volume of Π_2 (denoted as $V_2(\Lambda)$) can be computed by:

$$V_2(\Lambda) = \frac{v_{N-1}R_{c_1}^{N-1}}{2^{NR_{s_2}}} = \frac{V_1(\Lambda)\Theta}{2^{NR_{s_2}}} = S_N\Theta 2^{-N(R_{s_1}+R_{s_2})} \tag{4.17}$$

Π_2 defines a new lattice that is used to quantize the normalized residual vector (i.e. X_{1n}). Since the input vector X is uniformly distributed in S_N, the residual shape vector X_{1n} is uniformly distributed in the volume of the Voronoi region of Λ (i.e. Π_1). The quantization error of the scaled version of lattice Λ used to quantize X_{1n} is equal to the total shape distortion at the second stage. It can be computed by:

$$D_{s_2} = \frac{\sigma^2}{V_2(\Lambda)}\int_{\Pi_2}\|t\|^2 dt \tag{4.18}$$

Thus

$$D_{s_2} = (N-1)\sigma^2 G(\Lambda)V_2(\Lambda)^{\frac{2}{N-1}} \tag{4.19}$$

or

$$D_{s_2} = (N-1)\sigma^2 G(\Lambda)(\Theta S_N)^{\frac{2}{N-1}}2^{-\frac{2N}{N-1}(R_{s_1}+R_{s_2})} \tag{4.20}$$

Continuing this process for j times:

$$D_{s_j} = (N-1)\sigma^2 G(\Lambda) S_N^{\frac{2}{N-1}} \Theta^{\frac{2(j-1)}{N-1}} 2^{-\frac{2N}{N-1}(R_{s_1}+R_{s_2}+...+R_{s_j})} \tag{4.21}$$

Or the distortion after j refinements can be written as:

$$D_{s_j} = C_s(j) 2^{-\frac{2N}{N-1}(R_{s_1}+R_{s_2}+...+R_{s_j})} \tag{4.22}$$

where:

$$C_s(j) = (N-1)\sigma^2 G(\Lambda) S_N^{\frac{2}{N-1}} \Theta^{\frac{2(j-1)}{N-1}} \tag{4.23}$$

As discussed previously, the gain component must also be successively quantized and encoded along with the shape component. Suppose at each refinement stage i, R_{g_i} bits are assigned to the gain component. Since a successively refinable scalar quantizer is used (at each refinement stage, the quantization step size is quantized by another scalar quantizer), for high resolution quantizers ($R_{g_i} \rightarrow \infty$) the gain distortion at stage i can be written as:

$$D_{g_i} = C_g 2^{-2N(R_{g_1}+...+R_{g_i})} \tag{4.24}$$

where C_g is a constant [50]. Therefore the total distortion can be found by:

$$D = C_s(j) 2^{-\frac{2N}{N-1}(R_{s_1}+...+R_{s_j})} + C_g 2^{-2N(R_{g_1}+...+R_{g_j})} \tag{4.25}$$

In order to find the optimum values for the gain rate and the shape rate at each refinement stage, the distortion function must be minimized subject to the rate constraint. In fact, at each stage we need to allocate R_{s_i} bits to the shape component and R_{g_i} bits to the gain component such that the total bit rate is equal to a predefined value ($R_{s_i} + R_{g_i} = R_i$) at that particular refinement stage. The optimum value for each of the shape component rates can be found by solving $\frac{\partial D}{\partial R_{s_i}} = 0$. This yields:

$$R_{s_1} + ... + R_{s_j} = \frac{N-1}{2N^2} \log_2 \frac{C_s(j)}{(N-1)C_g} + \frac{N-1}{N}(R_1 + ... + R_j) \tag{4.26}$$

The distortion must be minimum at each stage. When no successive refinement is provided (i.e. $j = 1$), the optimum bit allocation can be found using the results in [37]:

$$R_s = \frac{N-1}{2N^2}\log_2\frac{C_s(1)}{(N-1)C_g} + \frac{N-1}{N}R \tag{4.27}$$

Now let us use the following allocation of bit rates to the gain and shape components:

$$R_{s_1} = \frac{N-1}{2N^2}\log_2\frac{C_s(1)}{(N-1)C_g} + \frac{N-1}{N}R_1 \tag{4.28}$$

$$R_{s_i} = \frac{N-1}{2N^2}\log_2\Theta^{\frac{2}{N-1}} + \frac{N-1}{N}R_i \qquad i > 1 \tag{4.29}$$

Using equation (4.23), it is easy to verify that the above allocation satisfies equation (4.26) for all values of j and therefore results in minimum distortion for all bit rates.

Note that in the asymptotic case when $N \to \infty$, $R_{g_i} \to 0$, $R_{s_i} \to R_i$, $\Theta \to 1$ and $G(\Lambda) \to \frac{1}{2\pi e}$ ([31]). Therefore $D_j = \sigma^2 2^{-2R}$ (where $R = R_1 + ... + R_j$) which is equal to the distortion rate function of Gaussian sources. Thus, successive refinement of the output does not affect the performance of the quantizer and minimum distortion can be achieved at all rates regardless of the value of j. This result shows that in the asymptotic case, our quantizer preserves successive refinability of Gaussian sources.

4.2.1 Cost of successive refinement

In practice, the lattice dimension cannot be very large and successive refinement causes some increase in distortion compared to the distortion obtained when no successive refinement is provided. This increase is mainly due to the redundancy caused by defining lattices in spherical volumes rather than the actual Voronoi regions. Using equations (4.26) and (4.25) the distortion of our successively refinable lattice quantizer (SRLQ) can be computed and compared for different values of j. In fact, the distortion obtained by our proposed successively refinable quantizer is approximately equal to the distortion obtained by shape gain quantizer designed in [37] multiplied by

$\Theta^{\frac{2(j-1)}{N-1}}$. This shows that in the case of high resolution, for every refinement stage our successive refinement approach increases the output distortion by a factor of $\Theta^{\frac{2}{N-1}}$. Thus, when the 24 dimensional Leech lattice is used ($N = 25$ and $\Theta = 7.9035$ [31]) every refinement stage reduces the value of SNR by approximately 0.74 dB compared to the signal to noise ratio obtained without successive refinement.

4.3 Simulation results

We block an i.i.d memoryless Gaussian source into 25-dimensional vectors and this vector forms the input to the quantizer. Thus, the 24 dimensional Leech lattice which provides the best packing in 24 dimensions can be used in the wrapped spherical code [37]. A PDF optimized scalar quantizer is used to successively quantize the value of the gain. The bit allocation for the gain and the shape components are found based on the method described in section 4.2. One important rate allocation case in successive refinement is when the refinement rate is exactly 1 bit per sample ($R_i = 1$ for $i > 1$). Based on the results of section 4.2, the optimal gain and shape bit allocation is to allocate approximately $\frac{1}{25}$ bits to refinement of gain and $\frac{24}{25}$ bits to refinement of the shape component (the first term in equation (4.29) is equal to 0.0048 for the SRLQ based on the 24-dimensional Leech lattice and therefore is negligible). This means that in order to operate at 1 bit per sample, the sphere containing the lattice points in the first refinement stage must have 2^{24} points. However, the structure of the Leech lattice does not allow 2^{24} points in any sphere. In fact, the 4th shell of the Leech lattice contains exactly 2^{24} points while the first 3 shells contain $2^{17.6}$ points. Thus, the only way to set the refinement bit rate to 1 is to use only the 4th shell. Therefore, we modified the decoding algorithm to ensure the lattice point is selected only from the 4th shell when the refinement rate is 1 bit per sample.

The simulation results are shown in table 4.3. The signal to noise ratio is calculated

		SRLQ		TS-TCQ [19]		SR-TCVQ [19]		MS-TCQ [20]	
R1	R2	D1	D2	D1	D2	D1	D2	D1	D2
2	1	11.02	16.46	10.55	15.96	10.52	16.21	-	-
2	2	11.02	22.09	10.55	21.55	10.54	21.69	10.24	19.70
3	1	17.32	22.63	16.21	21.76	16.18	21.99	15.89	19.73
3	2	17.32	28.10	-	-	-	-	15.89	25.51

		MS-TCQ [21]		Lloyd-Max [49]		MP [8]	
R1	R2	D1	D2	D1	D2	D1	D2
2	1	10.76	15.49	9.30	14.62	-	-
2	2	-	-	9.30	20.22	11.06	22.01
3	1	-	-	14.62	20.22	-	-
3	2	-	-	14.62	26.02	17.36	28.28

Table 4.3: Comparison of different successively refinable quantizers designed for Gaussian sources with the successively refinable lattice quantizer (SRLQ) proposed in this chapter. The SNR values for different bit rates are shown whenever such data is available. D1 and D2 denote the SNR values in db at rates R1 and R2 respectively. Note that using the quantizer in [37] (no successive refinement) we achieve SNR values of 11.02, 17.36, 23.33 and 29.29 db for rates 2, 3, 4 and 5 bits per sample respectively.

by quantizing 25000 samples from a Gaussian source and finding their resulting signal and noise powers. The performance of our successively refinable quantizer is compared to successively refinable quantizers designed for quantizing Gaussian sources in [19–21, 49]. As can be seen in the table, our results are always better than the results achieved by other successively refinable quantizers. Comparing the SRLQ with the MP encoder designed in chapter 3, we can see that the performance of SRLQ and the MP encoder are almost the same. This means that the redundancy of having one extra dimension in the codebook is almost equal to the redundancy caused by partitioning spheres that cover the Voronoi regions instead of the Voronoi regions. However the MP encoder cannot be used for bit rates less than 2 bits per sample due to the poor performance of the wrapped spherical codes at low bit rates. As mentioned in section 4.2 in the case of high resolution our proposed successive refinement algorithm results in 0.74 dB decrease in SNR for each stage. Comparing our successively refinable quantizer with the one designed in [37] (no successive refinement), we can see that successive refinement results in about 0.7 to 1.3 dB decrease in SNR values for low

bit rates which is relatively close to our high resolution analysis result.

4.4 Conclusion

A new successively refinable quantizer is proposed for quantizing Gaussian sources. We presented a high resolution analysis and found the optimal bit allocation for our proposed successively refinable quantizer. Moreover, we calculated the penalty caused by successive refinement and showed that 0.74 dB decrease in SNR will occur for high resolution coding of Gaussian sources using the 24-dimensional Leech lattice. Our simulation results show that in the practical case of low resolution, successive refinement causes about 0.7 to 1.3 dB decrease in the signal to noise ratio values. Our proposed quantizer outperforms the best known quantizers that provide successive refinability and its performance is almost similar to the MP encoder designed in chapter 3 with the added advantage of being able to provide refinements at 1 bit per sample. The MP encoder in chapter 3 and the SRLQ designed in this chapter have been able to reduce the gap between the theoretical bound on the rate distortion function of progressive encoders and the practical results. However, they were not able to close the gap completely due to a number of practical considerations.

Chapter 5

Adaptive Rate-Distortion Optimal In-loop Quantization for Matching Pursuit Image Coding

As discussed in chapter 2, traditional image compression algorithms usually include prediction, transform, quantization and entropy coding. The transform used in most conventional image codecs are complete transforms like DCT, wavelet or integer transform. We stated that over-complete transforms can provide a more compact image representation and consequently achieve higher compression. In this chapter and also in the next chapter, we study image compression using over-complete dictionaries. To map signals to the over-complete dictionary we use the matching pursuit algorithm due to its relatively low computational complexity compared to other algorithms and its good performance.

In chapter 3, we showed how matching pursuit can be used for progressive encoding of Gaussian sources and we showed that our proposed encoder results in better rate distortion performance than other quantizers designed for successive refinement of Gaussian sources. In this chapter we also study the application of matching pursuit

in progressive encoding of image data and propose an MP image encoder that can outperform existing MP encoders designed for image coding.

Matching pursuit has been shown to outperform DCT transform coding of residual frames in very low bit rate coding of video sequences [22]. In a more recent work in [24], MP is applied to image compression and it is shown that matching pursuit image coding can improve image compression quality at very low bit rates while providing additional features like fine grained PSNR and bit rate scalability. Moreover, based on the results reported in [24], the subjective quality of MP encoded images at low bit rates is significantly better than those compressed by traditional compression algorithms like JPEG2000 [53].

As mentioned in chapter 2, in matching pursuit, the input image or the motion-compensated residual frame are encoded by successively mapping them to a redundant dictionary of bases. MP representation consists of both the dictionary indexes and the quantized inner product coefficients at each stage. Therefore, efficient encoding of dictionary indexes and quantized inner product coefficients heavily affect the bit rate of the MP encoder. There have been a number of works addressing the problem of encoding MP inner product coefficients and dictionary indexes. In chapter 2, we classified these coding methods into two major categories. The more common method is to find all atoms and encode them in their position order. The position of atoms are differentially encoded which results in fewer bits needed to encode the position information of dictionary indexes [22, 23, 54]. In the second class of methods atoms are encoded in the order of the magnitude of coefficients (i.e. atoms with larger inner product coefficients are encoded first). In this method more bits are assigned to more important atoms and the redundancy between inner product coefficients is exploited [41]. Both methods are shown to result in almost similar performances for image coding applications [41]. However, the second class allows for fine grained PSNR and bit rate scalability which is an important property in image compression

and is the method used in [24] for compressing still images by matching pursuit. In this chapter we propose an adaptive quantization algorithm that uses the properties of inner product coefficients and show how it can outperform the quantization method proposed in [41]. Our method belongs to the second category, i.e. atoms are encoded in their importance order. Therefore, encoding of the atom positions is more expensive than methods proposed in [22] and [54]. However the inner product coefficients are quantized by fewer bits than the above methods and are more efficiently quantized than the one proposed in [41]. Moreover, in [41] a posteriori quantization is used and quantization is performed after all atoms are found. However, a posteriori quantization results in accumulation of quantization errors and is known to perform worse than in-loop quantization. In in-loop quantization, quantization error is added to residual vectors at each stage and therefore can be corrected by the atoms found in the following stages. In this chapter we consider in-loop quantization in order to maximize the efficiency of our quantization scheme while providing PSNR and bit rate scalability. We adopt our analysis in chapter 3 for in-loop quantization of MP coefficients, apply it to image data and design an adaptive in-loop quantization algorithm for quantizing MP coefficients[1].

Although the use of in-loop quantization does not allow finding the rate distortion optimal quantization parameters for all bit rates without recomputing the atoms (as opposed to a posteriori quantization), the resulting stream is still scalable and can be decoded at any bit rate. This is because atoms are encoded in their importance order which consequently results in a scalable bitstream. This scalable stream is only rate-distortion optimal if it is decoded at the same rate as the encoding rate (the bit rate that is used for finding optimal quantization parameters). However,

[1]In-loop quantization has been studied in [23] for quantizing inner product coefficients for the application of video coding. For that application atoms are encoded in their position order and therefore, the quantization method studied in [23] belongs to the first class of MP coding methods. In this chapter we study in-loop quantization when atoms are encoded in their importance order and find the optimal quantizers for each stage based on a rate distortion optimization.

our simulation results show that decoding this stream at a different bit rate than the encoding rate results in images with higher PSNR than the ones decoded from a stream generated by a posteriori quantization algorithm in [41] optimized for the decoding bit rate. This shows that the efficiency of our proposed adaptive in-loop quantization algorithm compensates for the sub-optimality caused by not being able to find optimal quantization parameters for all bit rates without recomputing the atoms.

5.1 Proposed Quantization Scheme

The inner product coefficients found at each MP stage have to be quantized before they can be encoded. Some MP encoders quantize inner product coefficients using a fixed uniform quantizer [22]. However, the properties of inner product coefficients can be used for more efficient quantization [23, 41]. The inner product coefficients of successive stages are generally close to each other and their magnitude decreases as we go to higher stages. Moreover, if we plot the probability distribution of inner product coefficients of each stage, they tend to have a high probability around a certain average value and as we move away from the average, the probability of having an inner product coefficient reduces (see Fig. 5.1 and Fig. 3.8). This result motivates us to design the quantizer for each stage such that the quantizer is centered at the mean value of the coefficients at that stage (this is similar to the quantizer used in chapter 3). As discussed in chapter 3, this quantization scheme clearly results in more accurate quantization of inner product coefficients than a scheme that quantizes the coefficients in the entire range. However, in order to use the above technique and design a PDF-optimized quantization scheme, the PDF of coefficients must be known at each stage. For Gaussian signals we were able to find the PDF, mean and variance of the inner product coefficients in section 3.1. For image data the

inner product coefficients depend on the PDF of the input image which is generally unknown. Therefore, the probability distributions of the coefficients of each stage should be found by training the MP encoder over a large number of training inputs. Similar to the MP encoder in chapter 3, in order to reduce the complexity, we only use the mean and variance of the coefficients at each stage (which are denoted as m_{c_i} and v_{c_i} respectively) and use a uniform quantizer to quantize the coefficients in the range $(m_{c_i} - \alpha\sigma_{c_i}, m_{c_i} + \alpha\sigma_{c_i})$ where α is a fixed constant. In section 5.2, we assume the mean and variance of coefficients are known and find the optimal quantization levels and number of MP stages. In section 5.3 we show how the means and variances can be found from the already quantized coefficients that are available at both the encoder and the decoder.

The second property of inner product coefficients that is considered in our proposed quantization scheme is the fact that the inner product coefficients of different stages do not have similar impact on the total MP distortion. This is because the inner product coefficients of atoms found in the early stages of matching pursuit generally have higher variances than the ones found at the higher stages. In other words, the dynamic range of coefficients in early stages is larger than the dynamic range of the coefficients found in the higher stages (e.g. see Fig. 5.1). Thus, if a fixed number of quantization levels is used for all stages, the quantization step size must be larger for quantization of coefficients in the early stages. Therefore, errors in quantization of early stage inner product coefficients can cause more distortion than errors in quantization of higher stage coefficients. On the other hand, part of the quantization error of the early stages will be corrected in the subsequent stages if in-loop quantization is applied. Hence, it is important to consider these two contrasting effects in bit allocation for quantization at each stage.

In order to consider this unequal importance of quantization error at different stages, in this chapter we allow variable number of quantization levels for different

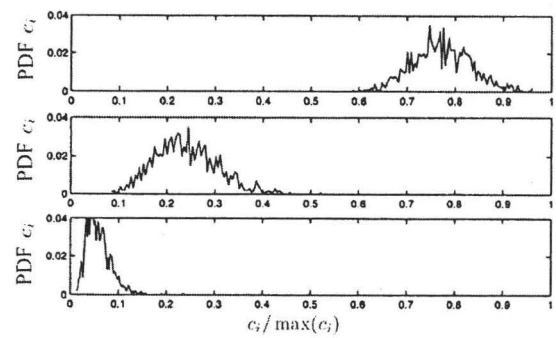


Figure 5.1: Distribution of inner product coefficients for i={0,4,9} for the MP expansion of ten-sample random signals over a dictionary of 128 atoms (figure is taken from [41]).

MP stages. Thus, an optimum number of bits can be assigned to quantization of inner product coefficients at each stage based on the impact of the quantization error at that stage on the total MP distortion. In the next section we model the matching pursuit distortion considering in-loop quantization and find the optimum quantization levels for each stage using a rate distortion optimization. Our analysis is similar to our analysis in chapter 3 for Gaussian distribution except that for image data the MP distortion cannot be explicitly computed. For the sake of completeness, we repeat the analysis in chapter 3 for computing the distortion for in-loop quantization and apply our analysis to image data.

5.2 Rate Distortion Optimization for Quantized MP

The block diagrams of matching pursuit encoder and decoder are shown in Fig. 5.2. At the encoder side, the input vector f goes through the first stage of matching pursuit and the dictionary element g_{γ_i} with maximum inner product coefficient is

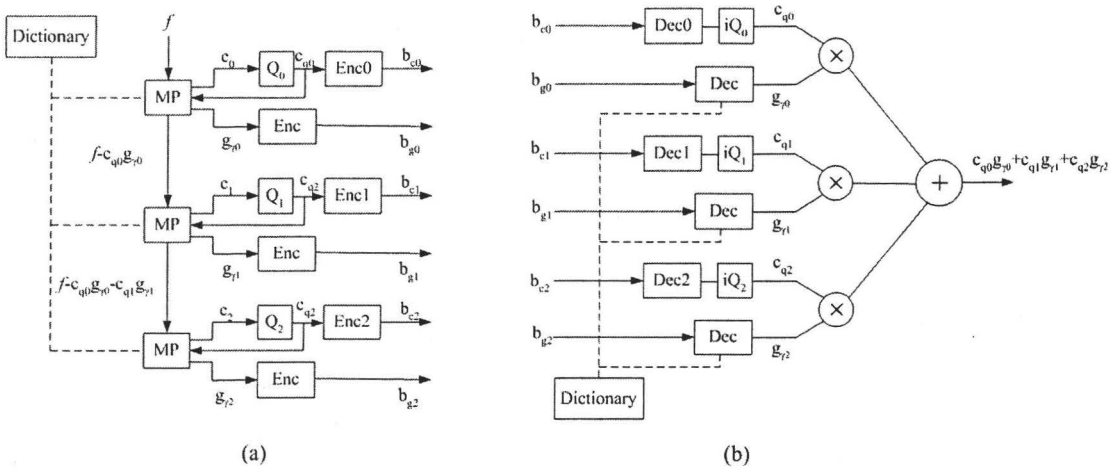


Figure 5.2: The block diagrams of (a) MP encoder, (b) MP decoder

found along with the inner product coefficient c_i. The inner product coefficient is quantized and encoded and is sent to the decoder together with the encoded index of the dictionary element. Then, the quantized inner product coefficient c_{qi} is used to find the residual vector which goes through the second matching pursuit stage. This process continues for all MP stages until the full k stage decomposition is constructed. The decoder decodes the inner product coefficients and dictionary indexes and reconstructs the input vector according to equation (2.14). Since the quantized inner product coefficient is used to find the input to next stages, the MP equation at each stage (equation (2.13)) can be modified to:

$$R^i f_q = \langle R^i f_q, g_{\gamma_i} \rangle g_{\gamma_i} + R^{i+1} f \tag{5.1}$$

where $R^i f_q$ is the residual vector which is computed using quantized value of inner product coefficients at previous stages. Thus $R^i f_q$ can be written as:

$$R^i f_q = f - \sum_{j=0}^{i-1} c_{qj} g_{\gamma_j} \tag{5.2}$$

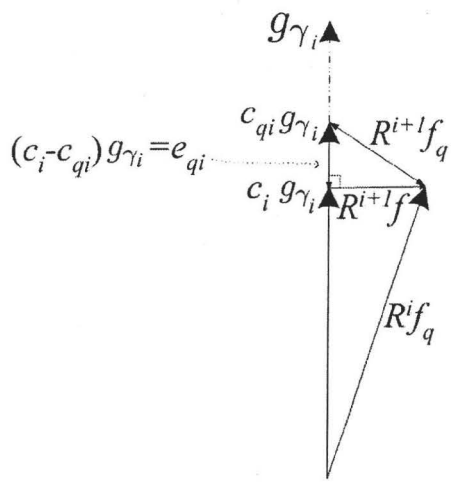


Figure 5.3: Geometric interpretation of quantized matching pursuit

where c_{qj} is the quantized value of inner product coefficient at stage j (i.e. c_j). If the inner product coefficients are not quantized, the total distortion at stage i will be $\|R^{i+1}f\|^2$. However, at each stage the inner product coefficient is quantized. This introduces an additional quantization error $\|e_{qi}\|^2$, where e_{qi} is the quantization error vector and can be written as (see Fig. 5.3):

$$e_{qi} = (c_i - c_{qi})g_{\gamma_i} \tag{5.3}$$

Part of this error will be corrected in the subsequent stages if we plug

$$R^{i+1}f_q = R^{i+1}f + e_{qi} = R^i f_q - c_{qi}g_{\gamma_i} \tag{5.4}$$

into the next stage instead of $R^{i+1}f$ (see Fig. 5.3 for the above equalities). Now, the distortion at stage i is equal to the square norm of the total residual vector at stage i (i.e. $\|R^{i+1}f_q\|^2$). As shown in Fig. 5.3, the MP distortion $R^{i+1}f$ is orthogonal to g_{γ_i}, at every iteration, thus quantization error e_{qi} is also orthogonal to matching pursuit distortion $R^{i+1}f$, therefore:

$$\|R^{i+1}f_q\|^2 = \|R^{i+1}f + e_{qi}\|^2 = \|R^{i+1}f\|^2 + \|e_{qi}\|^2 \tag{5.5}$$

Equation (5.5) can be written as:

$$\|R^{i+1}f_q\|^2 = \|R^i f_q\|^2 r_{q_i}^2 + \|e_{qi}\|^2 \tag{5.6}$$

where r_{qi} is defined as:

$$r_{qi} \triangleq \frac{\|R^{i+1}f\|}{\|R^i f_q\|} \tag{5.7}$$

Now, the average total distortion at stage i can be written as:

$$D_{MPQi} = E\{\|R^{i+1}f_q\|^2\} = E\{\|R^i f_q\|^2 r_{qi}^2\} + E\{\|e_{qi}\|^2\} \tag{5.8}$$

If both sides of equation (5.1) are divided by $\|R^i f_q\|$, we will have:

$$\frac{R^i f_q}{\|R^i f_q\|} = \langle \frac{R^i f_q}{\|R^i f_q\|}, g_{\gamma_i} \rangle g_{\gamma_i} + \vec{r}_{qi} \tag{5.9}$$

where

$$\vec{r}_{qi} \triangleq \frac{R^{i+1}f}{\|R^i f_q\|} \tag{5.10}$$

Using equations (5.7) and (5.10) we can write:

$$r_{qi} = \|\vec{r}_{qi}\| \tag{5.11}$$

Now according to equation (5.9), r_{qi} is the norm of the residual vector when matching pursuit is applied to the normalized input vector ($\frac{R^i f_q}{\|R^i f_q\|}$). Therefore, r_{qi} only depends on $\frac{R^i f_q}{\|R^i f_q\|}$ (i.e. the direction of vector $R^i f_q$). Since norm and direction of the residual vectors are independent, $\|R^i f_q\|$ and r_{qi} are independent and equation (5.8) can be written as:

$$D_{MPQi} = E\{\|R^{i+1}fq\|^2\} = E\{\|R^i f_q\|^2\} E\{r_{qi}^2\} + E\{\|e_{qi}\|^2\} \tag{5.12}$$

If c_i is quantized by q_i levels, the quantization distortion at stage i can be written as:

$$E\{\|e_{qi}\|^2\} = \frac{\alpha_i \sigma_{c_i}^2}{q_i^2} \tag{5.13}$$

where α_i is a factor that depends on the PDF of c_i and the quantizer [27] and $\sigma_{c_i}^2$ is the variance of c_i. Let us define[2]:

$$\delta_i^2 = E\{r_{qi}^2\} \tag{5.14}$$

Using equations (5.12) and (5.13) repeatedly, the distortion of k stage matching pursuit can be written as:

$$D_{MPQ} = \left(... \left(\left(\left(\|f\|^2 \delta_0^2 + \frac{\alpha_0 \sigma_{c_0}^2}{q_0^2} \right) \delta_1^2 + \frac{\alpha_1 \sigma_{c_1}^2}{q_1^2} \right) \delta_2^2 + \frac{\alpha_2 \sigma_{c_2}^2}{q_2^2} \right) \delta_3^2 + ... \right) \delta_{k-1}^2 + \frac{\alpha_{k-1} \sigma_{c_{k-1}}^2}{q_{k-1}^2} \tag{5.15}$$

Equation (5.15) can be summarized as:

$$D_{MPQ} = \|f\|^2 \delta_0^2 \delta_1^2 ... \delta_{k-1}^2 + \sum_{i=0}^{k-1} \delta_{i+1}^2 ... \delta_{k-1}^2 \frac{\alpha_i^2 \sigma_{c_i}^2}{q_i^2} \tag{5.16}$$

The above equation formulates MP distortion based on number of quantization levels at each stage (q_i), number of MP stages (k), values of δ_i and σ_{c_i} which mainly depend on the dictionary and source distribution and α_i which depends on quantization method. In this section we assume δ_i and σ_{c_i} are known (they can be found by training over a large set of training set). In the next section we show how these values can be found adaptively based on already decoded information. As mentioned above, α_i depends on the quantization scheme. The most efficient quantization scheme is the one that is designed based on the probability distribution of coefficients. In order to design such quantizer, we need to know the PDF of the coefficients. The PDF's heavily depend on the source characteristics and MP dictionary and is different for different sources and dictionaries. However, the PDF graphs tend to have a large peak close to a mean value and the probability decreases as the values get farther from the mean. As mentioned before, in this chapter we use a uniform quantizer with

[2]Note that δ_i was explicitly computed in chapter 3 equation 3.40 for Gaussian sources. However, for image data δ_i depends on the PDF of the image and cannot be computed if the PDF of the image is unknown.

a granular region that is equal to $(m_{c_i} - \alpha\sigma_{c_i}, m_{c_i} + \alpha\sigma_{c_i})$ where m_{c_i} and σ_{c_i} are the mean and standard deviation of the inner product coefficient at stage i respectively and are assumed to be known at this point. Based on our experimental results, the majority of coefficient values lie within the range $(m_{c_i} - 3\sigma_{c_i}, m_{c_i} + 3\sigma_{c_i})$. Therefore, in this chapter we set $\alpha = 3$ in all our experiments. The quantization distortion for this particular quantizer can be easily computed if the probability distribution of the inner product coefficients is known. However, if the number of quantization levels is high, the high resolution approximation can be used and the quantization error can be approximated by $\frac{\Delta^2}{12}$ where Δ is the quantization step size [55]. If the coefficients are quantized by q_i levels in the range $(m_{c_i} - \alpha\sigma_{c_i}, m_{c_i} + \alpha\sigma_{c_i})$, the quantization step size will be $\Delta = \frac{2\alpha\sigma_{c_i}}{q_i}$ and according to the above assumptions the quantization distortion at stage i can be written as:

$$D_{\mathrm{Qi}} \approx \frac{\alpha^2\sigma_{c_i}^2}{3q_i^2} \tag{5.17}$$

To have a complete description of MP expansion, the quantized values of the inner product coefficients at each stage and the index of the dictionary element with the maximum inner product must be encoded. Therefore, the bit rate of a k-stage MP encoder can be found by:

$$Rate = \mu_q \sum_{i=0}^{k-1} \log_2 q_i + \mu_M k \log_2 M \tag{5.18}$$

where M is the total number of dictionary elements and $\mu_q \leq 1$ and $\mu_M \leq 1$ are constants that model the effect of entropy coding on the bit rate. When no entropy coding is applied $\mu_q = \mu_M = 1$.

Now that the rate and distortion have been formulated based on the MP encoder parameters (q_i, k and M) and source distribution, an optimum tradeoff can be found between the number of matching pursuit stages (k) and the number of quantization levels (q_i). For a fixed bit rate, if the number of bits assigned to quantization levels is low (i.e. q_i is small), more MP stages can be encoded (since there are more bits

to send for the dictionary indexes) resulting in a decrease in MP distortion. However small number of quantization levels increases the quantization distortion. Therefore the optimum values for q_i and k must be found such that the distortion is minimized for a given rate. The Lagrangian multiplier method is used to solve this optimization problem:

$$J(\lambda) = D_{\text{MPQ}} + \lambda Rate \tag{5.19}$$

where λ is the Lagrangian multiplier. Substituting the equations for MP distortion and rate from (5.16) and (5.18) into (5.19), the Lagrangian cost function can be formulated as:

$$J(\lambda) = \|f\|^2 \delta_0^2 \delta_1^2 ... \delta_{k-1}^2 + \sum_{i=0}^{k-1} \delta_{i+1}^2 ... \delta_{k-1}^2 \frac{\alpha^2 \sigma_{c_i}^2}{3q_i^2} + \lambda(\mu_q \sum_{i=0}^{k-1} \log_2 q_i + \mu_M k \log_2 M) \tag{5.20}$$

Taking the partial derivative in terms of q_i yields the optimum values for quantization levels. It is easy to show that:

$$\frac{\partial J(\lambda)}{\partial q_i} = 0 \quad \Rightarrow \quad q_i = \sigma_{c_i} \delta_{i+1} ... \delta_{k-1} \alpha \sqrt{\frac{2 \ln 2}{3 \lambda \mu_q}} \tag{5.21}$$

Using equation (5.21), one can show that:

$$\frac{q_i}{q_{i-1}} = \frac{\sigma_{c_i}}{\sigma_{c_{i-1}}} \times \frac{1}{\delta_i} \tag{5.22}$$

Note that combining equation (5.21) and (5.16) results in:

$$D_{MPQ} = \|f\|^2 \delta_0^2 \delta_1^2 ... \delta_{k-1}^2 + \frac{k \lambda \mu_q}{2 \ln 2} \tag{5.23}$$

Substituting the value of q_i from equation (5.21) into (5.20) and taking the partial derivative, the optimum λ can be computed as a function of k, σ_{c_i} and δ_i:

$$\lambda = f(\sigma_{c_i}, \delta_i, k) \tag{5.24}$$

The function $f(\sigma_i, \delta_i, k)$ is computed in Appendix A. Now that the optimum values for λ and q_i have been found, the next step is to find the optimal number of MP stages

k that satisfies the bit budget constraint and minimizes equation (5.20). Assuming the bit budget is fixed and is denoted by R_{budget}:

$$R_{\text{budget}} = \mu_q \sum_{i=0}^{k-1} \log_2 q_i + \mu_M k \log_2 M \tag{5.25}$$

Since both λ and q_i are functions of k, equation (5.25) can be solved numerically to find k for any value of R_{budget}. Additionally, since k can only take on integer values, the computation of optimum k does not impose high complexity and the search is performed over a small number of integer values.

5.3 Adaptive Quantization

The optimization method proposed in section 5.2 has a number of limitations that makes it difficult to be used in practical applications. First, the values of δ_0 to δ_{k-1} must be known which requires knowledge of the rate distortion curve of matching pursuit coding of the source. Moreover, the distribution of the inner product coefficients at each stage must be known. In particular the mean and the variance of the inner product coefficients at each stage must be known. Although these values can be obtained from experiments for a large set of training inputs, it is more desirable to be able to find those information dynamically during encoding. Additionally, for image coding applications, the rate distortion curve of MP encoding of an image can vary significantly among different images. Thus, adapting the rate distortion curve to the rate distortion curve of the input source results in more efficient quantization than using an average rate distortion curve. Therefore, we propose an adaptive quantizer that finds the required parameters at the encoder. This algorithm is a modification of the adaptive algorithm proposed in [41] and finds the necessary parameters for the quantization algorithm proposed in section 5.2 using a similar approach as the one used in [41]. Since the encoder only uses the information available from the previously

encoded stages to find the parameters, the same parameters can be computed at the decoder and no side information is needed to be sent to the decoder.

The parameters that must be known at stage i at the encoder and decoder are:

1. q_i: number of quantization levels at stage i
2. m_{c_i}: mean of the inner product coefficients at stage i
3. σ_{c_i}: variance of the inner product coefficients at stage i

Once these parameters are known the inner product coefficient at stage i (i.e. c_i) can be quantized by q_i levels in the range $(m_{c_i} - \alpha\sigma_{c_i}, m_{c_i} + \alpha\sigma_{c_i})$. According to equation (5.1) we have:

$$\|R^i f_q\|^2 = c_i^2 + \|R^{i+1} f\|^2 \tag{5.26}$$

Thus, using equation (5.7) we can write:

$$c_i^2 = \|R^i f_q\|^2 (1 - r_{qi}^2) \tag{5.27}$$

Now, if the two consecutive inner product coefficients are divided together and equation (5.6) is used, we can write:

$$\frac{c_i^2}{c_{i-1}^2} = \frac{\|R^i f_q\|^2 (1 - r_{qi}^2)}{\|R^{i-1} f_q\|^2 (1 - r_{q(i-1)}^2)} = \frac{\|R^{i-1} f_q\|^2 r_{q(i-1)}^2 + e_{q(i-1)}^2}{\|R^{i-1} f_q\|^2} \frac{1 - r_{qi}^2}{1 - r_{q(i-1)}^2} \tag{5.28}$$

The values of r_{qi} for two consecutive stages are very close. Also, quantization distortion e_{qi} is usually much smaller than the norm of input vector at stage i ($\|R^i f_q\|$). Therefore equation (5.28) can be approximately written as:

$$\left|\frac{c_i}{c_{i-1}}\right| \approx r_{q(i-1)} \tag{5.29}$$

Equation (5.29) can be used to find the mean of the absolute value of inner product coefficient from the inner product coefficient at the previous stage if $r_{q(i-1)}$ is known:

$$m_{c_i} \approx r_{q(i-1)} |c_{i-1}| \tag{5.30}$$

Equation (5.29) is also used to estimate r_{qi} at stage i:

$$r_{qi} \approx r_{q(i-1)} = |\frac{c_i}{c_{i-1}}| \tag{5.31}$$

Therefore since $r_{q(i-1)}$ is available at the end of stage $i-1$, the value of m_{c_i} can be computed from equation (5.30) at stage i. Now the values of σ_{c_i} must be estimated. If we assume the variances of coefficients are the same for the past L stages ($L = 4$ in our experiments), σ_{c_i} can be estimated by:

$$\sigma_{c_i}^2 = \frac{1}{L}\sum_{j=1}^{L}(|c_{i-j}| - m_{c_{i-j}})^2 \tag{5.32}$$

The next step is to find the value of q_i. From equation (5.22) we can write:

$$q_i \approx q_{i-1}\frac{\sigma_{c_{i-1}}}{\sigma_{c_{i-2}}}\frac{1}{r_{q(i-1)}} \tag{5.33}$$

Note that in our adaptive quantization scheme the values of r_{qi} are used instead of the average values δ_{qi}. Consequently, the optimization is performed based on the operating rate distortion function rather than the average rate distortion function. Therefore, the adaptive quantization scheme can adapt the quantization scheme based on the operating rate distortion function and find more efficient quantizers than the method described in section 5.2.

Since the values of σ_{c_i}, m_{c_i}, q_i and r_{qi} must be calculated at both the encoder and the decoder, at the encoder the quantized values of the inner product coefficients are used to find the required parameters. This ensures both the encoder and decoder use the same quantization parameters. The adaptive algorithm can be described by the following pseudo code (in the following, $Q[c_i]$ is the quantized value of inner product coefficient c_i):

1. Initialize σ_{c_0}, m_{c_0}, q_0, σ_{c_1}, m_{c_1}, q_1.

2. $r_{q0} = r_{q1} = 1$

3. For $i = 0$ to $k-1$ do

   if $i > 1$ Then

   $q_i = q_{i-1} \frac{\sigma_{c_{i-1}}}{\sigma_{c_{i-2}}} \frac{1}{r_{q(i-1)}}$

   $\sigma_{c_i}^2 = \frac{1}{L} \sum_{j=1}^{L} (Q[c_{i-j}] - m_{c_{i-j}})^2$

   $m_{c_i} = r_{q(i-1)} Q[c_{i-1}]$

   end if

   $Q[c_i]$:= Quantize $|c_i|$ in range $(m_{c_i} - \alpha\sigma_{c_i}, m_{c_i} + \alpha\sigma_{c_i})$ by q_i levels

   $r_{qi} = \frac{Q[c_i]}{Q[c_{i-1}]}$

4. end for

The parameters σ_{c_0}, m_{c_0}, σ_{c_1} and m_{c_1} are approximated by finding the two stage MP description of a set of input signals. The values of q_0 and q_1 must be selected based on the bit rate. The optimal choice is to find these values using equation (5.21). However, this requires knowledge of rate distortion curve of the input which is not available at the beginning of coding. Therefore we found the best values for q_0 and q_1 by experiments for a number of bit rates and save them in both the encoder and decoder. The encoder selects q_0 and q_1 based on the required bit rate and sends that information to decoder. The rest of the quantization levels are found by the above algorithm.

The proposed algorithm only uses the parameters from the previous stages and the quantized inner product coefficients at each stage to estimate the required parameters for quantization. Since this information is available at the decoder side, no side information is required and both encoder and decoder can calculate the same parameters from the parameters of the previous stage and quantized inner product coefficients. The only side information is the quantization levels for the first two stages which must be sent to the decoder. The improvement in the bit rate comes

from the fact that in general fewer bits are assigned to the last MP stages while more bits are assigned to the early and more important stages.

In the quantization scheme proposed in [41] the inner products can be encoded with zero bits, i.e. no bits are sent for the inner product coefficient and only the dictionary index is sent. The inner product coefficient is mapped on the rate distortion function. However, sending atoms without inner product coefficients can cause our algorithm to converge to a nonzero distortion. This is because in our algorithm, the granular regions of the quantizers do not cover the entire range of inner product coefficients as opposed to the quantizer designed in [41]. Thus, when coefficient magnitudes are not in the range $(m_{c_i} - \alpha\sigma_{c_i}, m_{c_i} + \alpha\sigma_{c_i})$, overload error occurs which is added to the MP distortion. In some cases overload error can be significantly larger than the granular error and the MP error (e.g. compare the range $(m_{c_i} - \alpha\sigma_{c_i}, m_{c_i} + \alpha\sigma_{c_i})$ and the full range of inner product coefficients $(0, \max(c_i))$ in Fig. 5.1). Additionally, the quantization range at the beginning of encoding is defined by the choice of q_0 and q_1. If q_0 and q_1 do not match the rate distortion curve of the source that is being encoded, large overload errors can occur. As described in the following paragraphs, these cases can potentially cause our proposed quantizer to converge to a nonzero distortion. Let $e_{oi} = e_{qi}$ denote the quantization distortion vector which is caused by the overload error. According to equation (5.4) and since e_{oi} is in the same direction as g_{γ_i}, when $\|e_{oi}\| \gg \|R^{i+1}f\|$ the residual vector at stage i can be written as:

$$R^{i+1}f_q = R^{i+1}f + \|e_{oi}\|g_{\gamma_i} \approx \|e_{oi}\|g_{\gamma_i} \tag{5.34}$$

Now, since $R^{i+1}f_q \approx \|e_{oi}\|g_{\gamma_i}$, the dictionary element g_{γ_i} will have the largest inner product with residual vector $R^{i+1}f_q$ and therefore g_{γ_i} will also be selected in the next stage (i.e. $g_{\gamma_{i+1}} = g_{\gamma_i}$). Thus, according to equation (5.4), the residual vector at stage

$i+1$ can be written as:

$$R^{i+2}f_q = R^{i+1}f_q - c_{q_{i+1}}g_{\gamma_{i+1}} \approx (\|e_{oi}\| - c_{q_{i+1}})g_{\gamma_i} \tag{5.35}$$

The above equation states that the overload error will be corrected only if $c_{q_{i+1}}$ is comparable with $\|e_{oi}\|$. If no bits are sent for quantized inner product coefficients, the coefficients are quantized to their average value (i.e. $c_{q_{i+1}} = m_{c_{i+1}}$). Thus:

$$R^{i+2}f_q \approx (\|e_o\| - m_{c_{i+1}})g_{\gamma_i} \tag{5.36}$$

$m_{c_{i+1}}$ is the average of c_{i+1} and when overload error occurs it is possible that $m_{c_{i+1}} \ll \|e_{oi}\|$. In this case, the residual vector at stage $i+1$ can be written as: $R^{i+2}f_q \approx \|e_o\|g_{\gamma_i} = R^{i+1}f_q$. Therefore, the error at stage $i+1$ will be equal to the error at stage i and is equal to the overload error. Since m_{c_i} is a decreasing function of i, the overload error will not be corrected in the next stages either and the same dictionary element will be selected at the following stages. Thus, the distortion does not reduce significantly by performing more MP iterations. In order to ensure this condition will never happen in our adaptive quantization scheme, we assign at least one bit to the absolute value of the quantized inner product coefficients. This guarantees that, the values of m_{c_i} and σ_{c_i} are updated according to c_{q_i} and c_{q_i} becomes large enough in the following stages to compensate for the overload error according to equation (5.35). This feature is another advantage of our proposed adaptive quantization scheme compared to a non-adaptive scheme. As shown in chapter 3 (section 3.2.1), in a non-adaptive scheme, in order to guarantee that distortion always converges to zero, the range of the quantizers must be large to ensure overload errors are negligible (i.e. α must be large). This decreases the efficiency of the quantizers in fine quantization of inner product coefficients. However, the convergence problem does not exist in our proposed adaptive quantization scheme and it can exploit all advantages of our proposed quantization (in-loop quantization and fine quantization of inner product

coefficients). Note that, large overload errors caused by inappropriate values of q_0 and q_1 or by the small quantization range can cause some sub-optimality in our proposed quantization algorithm. This sub-optimality does not exist in a posteriori quantization since the best initial quantization ranges can be found after all atoms are computed and the quantization range is always large enough to prevent overload errors. However, the advantage of in-loop quantization compensates for this sub-optimality and as our simulation results show our proposed adaptive quantization scheme always outperforms a posteriori quantization.

5.4 Experimental Results

5.4.1 Random Signals

In this section we compare our proposed adaptive quantization algorithm with the quantization scheme proposed in [41]. Note that the main difference between our proposed method and the method introduced in [41] is that in our quantization scheme the quantization levels are placed around the mean of the inner product coefficients and the granular region of the quantizer is proportional to the standard deviation of the inner product coefficients. The second difference is the use of in-loop quantization in our quantization method.

Fig. 5.4. shows the rate distortion curves for our proposed algorithm and the algorithm proposed in [41]. The input signal is a 10 dimensional random signal. Each of the arguments in the 10 dimensional input vector is uniformly distributed between 0.5 and -0.5. The dictionary elements are also random and the size of the dictionary is 50. Our proposed quantization algorithm clearly outperforms the quantizer proposed in [41]. As can be seen in the figure, at low bit rates both quantizers have the same performance. This is because the number of MP stages is low at low bit rates. The

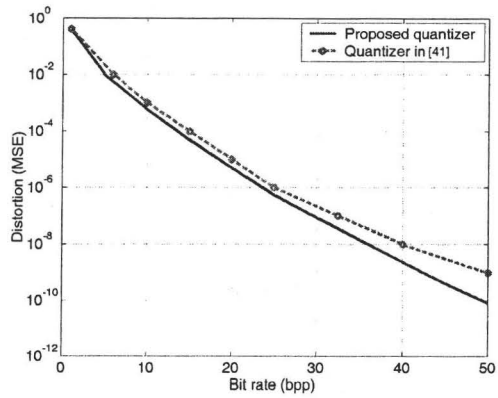


Figure 5.4: Comparison of R-D curves for MP coding of 10-dimensional random signals over a dictionary of 50 random vectors for our proposed quantization scheme and a posteriori quantization proposed in [41].

small improvement at low bit rates is caused by more accurate quantization of inner product coefficients by our proposed algorithm since in [41] the quantization range is between 0 and $\max(c_i)$ while in our method the range is $(m_i - \alpha\sigma_{c_i}, m_i + \alpha\sigma_{c_i})$. At high bit rates the improvement is more significant. This is because the inner product coefficients of the final stages are still accurately quantized while in [41] the accuracy is degraded by assigning fewer bits without adjusting the range. Additionally, since in-loop quantization is used in our proposed quantization scheme, quantization error does not accumulate which results in better rate distortion performance.

5.4.2 Image Coding

In this section the application of our method is evaluated in the more practical case of matching pursuit image coding. The dictionary used in our MP encoder is based on the one proposed in [24]. We applied the quantizer proposed in section 5.3 as well as the one proposed in [41] in order to quantize the inner product coefficients.

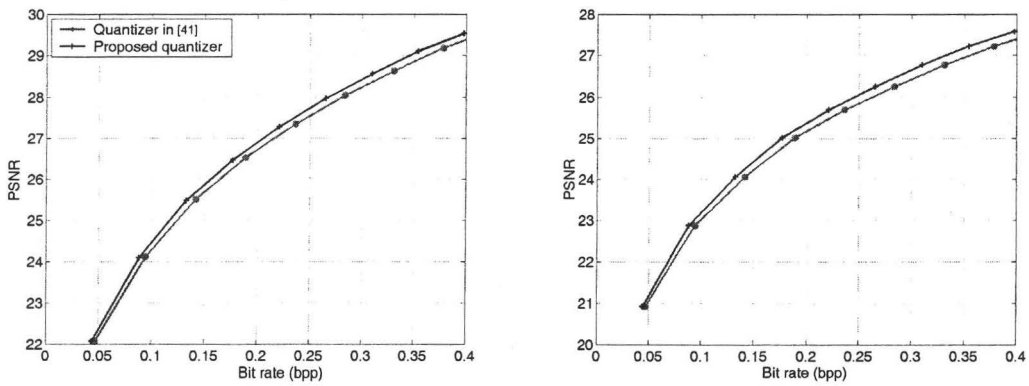


Figure 5.5: Comparison of PSNR versus bit rate curves for the proposed quantizer and the quantizer designed in [41]. (a) *Lena*, 256×256 (b) *Camera man* 256 × 256

The same dictionary is used for both quantizers. The dictionary indexes and the position information are encoded without entropy coding (the addition of entropy coding can improve the resulting PSNR graphs for both methods). The PSNR versus bit rate curves are shown in Fig. 5.5 for both our proposed quantizer and the quantizer designed in [41]. As shown in the figure, our proposed quantization scheme outperforms the quantizer proposed in [41] for both images used in this experiment.

5.5 Conclusion

In this chapter we proposed an adaptive in-loop quantization scheme for matching pursuit coding. Our quantization scheme allows different quantization levels for different stages. Additionally, the granular range of the quantizers are variable and depend on the distribution of the inner product coefficients at different stages. We found the MP distortion when in-loop quantization is used and calculated the optimum quantization levels and optimum number of MP stages using rate distortion optimization.

We also proposed an adaptive quantizer that finds the parameters required for quantization at each stage from the parameters and quantized inner product coefficients of the previous stages which are available at both the encoder and the decoder. Our experimental results show that our proposed quantization scheme outperforms the quantizer proposed in [41] which is used for matching pursuit image coding in [24].

Chapter 6

Optimized Atom Position and Coefficient Coding for Matching Pursuit Based Image Compression

In chapter 5, we designed an MP image encoder and used the correlations between inner product coefficients to achieve a better performance than the existing MP image encoders. However, the PSNR values achieved by the MP encoder in chapter 5 is still below what could be achieved by conventional image coders that use complete transforms such as JPEG2000. Although the wavelet transform used in JPEG2000 provides a very compact representation of the image, a significant portion of compression efficiency is achieved by the advanced entropy coding algorithm used in JPEG2000 [53]. The entropy coding algorithm in JPEG2000 uses all correlations between transform coefficients and therefore is able to achieve a very high compression ratio. Thus, in order to compare compression efficiency of over-complete image representations with complete transforms, more advanced entropy coding techniques must be designed to fully exploit the redundancy between the encoded data (i.e. atom indexes and inner product coefficients).

As mentioned in the previous chapters, MP encoding algorithm can be classified to encoding in the order of atom positions or in the order of inner product coefficients. Although both methods are shown to be efficient for certain applications, none of them are optimal. Differential coding of atom positions discards the correlation between inner product coefficients while encoding in importance order results in very expensive atom position coding. In this chapter, we find an optimum trade-off between the two methods and take advantage of the benefits of both encoding in atom position order and importance order. We show that our algorithm results in significant improvements over the methods proposed in [41] and the algorithm in chapter 5 and it results in better rate distortion performance than JPEG2000 in low bit rates.

6.1 Proposed Atom Position and Coefficient Coding

In order to encode atom positions and coefficients first we find all atoms and their corresponding inner product coefficients. When coding in position order, image is scanned in a raster order and atom positions are differentially encoded. When coding in atom importance order, atoms are encoded in the order of their inner product coefficients and the coefficients are differentially coded. In this method there is no correlation between atom positions. Therefore for an $N \times N$ image, $2\log_2 N$ bits must be sent for each atom in order to encode atom positions. In this chapter after atoms are found, the image is blocked into $B \times B$ blocks. For each block, the number of atoms that are located inside the block is sent. Then these atoms are encoded in importance order. Therefore, for each atom $2\log_2 B$ bits must be sent in order to encode atom positions. Then the atom coefficients are encoded differentially. Blocks

are encoded in the raster scan order. Therefore, fewer bits are needed to encode atom positions while there is still some correlations between atom coefficients which can be used for efficient coefficient encoding. However, by limiting ourselves to blocks of size $B \times B$ the amount of correlation that exists between successive atom coefficients reduces and therefore, more bits may be needed to encode these coefficients than the case when all atoms are encoded in their importance order. However, since fewer bits are needed to encode atom positions the overall bit rate can be reduced if the block size is selected in an optimum manner. In this section we solve this optimization problem and find the optimum block size and quantization levels for different bit rates.

Note that when $B = N$ our proposed algorithm reduces to one that encodes atoms in the order of inner product coefficients while when $B = 1$ our algorithm is equivalent to one that encodes atoms in their position order. In this chapter we try to find the best tradeoff between the two methods which in effect is equivalent to finding the best block size B.

As mentioned in chapter 2 section 2.5.1, the square of the norm of the residual signal at the K^{th} stage is equal to the total distortion of K-stage matching pursuit. Therefore:

$$D_{\text{MP}} = \|R^K f\|^2 \tag{6.1}$$

where D_{MP} is the distortion of matching pursuit.

Each inner product coefficient is quantized by a uniform scaler quantizer with range I and q quantization levels. The quantization distortion for this particular quantizer can be easily computed if the probability distribution of the inner product coefficients is known. However, if the number of quantization levels is high, the high resolution approximation can be used and the quantization error can be approximated by $\frac{\Delta^2}{12}$ where Δ is the quantization step size [55]. If the coefficients are quantized by q levels in the range I, the quantization step size will be $\Delta = \frac{I}{q}$ and according to the

above assumptions the quantization distortion can be written as:

$$D_{\mathrm{Q}} = \frac{I^2}{12q^2} \tag{6.2}$$

The range of inner product coefficients is different among atom coefficients and depends on when the atom is found. In order to quantize coefficients efficiently we assign different ranges to different coefficients and the range is optimized based on the probability distribution of the atom coefficient. Since the quantization ranges are variable, the number of quantization levels should be adjusted accordingly to have the minimum overall quantization distortion. Therefore, different quantization levels are assigned to each atom and the optimum number of quantization levels are found by a rate distortion optimization. Since all atoms are found first and then based on the required bit rate the coefficients are quantized, in-loop quantization cannot be used and the quantization errors are not added to residual frames. Therefore, the total quantization error will be the summation of quantization errors for each individual atom coefficient. Now, suppose there are $k_{i,j}$ atoms in block $B_{i,j}$. The range of the k^{th} atom coefficient in block $B_{i,j}$ is denoted by $I_{i,j,k}$ and the number of quantization levels assigned to it is shown by $q_{i,j,k}$. The total quantization distortion can be written as:

$$D_{\mathrm{Q}} = \sum_{i=0}^{\frac{N}{B}-1} \sum_{j=0}^{\frac{N}{B}-1} \sum_{k=0}^{k_{i,j}-1} \frac{I_{i,j,k}^2}{12q_{i,j,k}^2} \tag{6.3}$$

Now suppose there are M elements in the over-complete dictionary and these elements are translated to each pixel position in the image (i.e. the effective dictionary size is $M \times N^2$ with M main elements that are translated to $N \times N$ positions). For each block, first the number of atoms inside the block (i.e. $k_{i,j}$) must be encoded. Let us suppose we need to send $Rate(k_{i,j})$ bits to encode $k_{i,j}$. Then for each atom, $\log_2 M$ bits must be sent for each dictionary index, $2\log_2 B$ bits must be sent to identify the atom position inside the block and $1 + \log_2 q_{i,j,k}$ bits must be sent for the sign and

value of the quantized coefficient. Therefore, the bit rate can be computed as:

$$R = \sum_{i=0}^{\frac{N}{B}-1} \sum_{j=0}^{\frac{N}{B}-1} \left(Rate(k_{i,j}) + \sum_{k=0}^{k_{i,j}-1} (\log_2 q_{i,j,k} + \log_2 M + 1 + 2\log_2 B) \right) \tag{6.4}$$

In order to compute $Rate(k_{i,j})$ the probability distribution of $k_{i,j}$ must be found. Assuming atoms are randomly distributed in the image, the probability of an atom being located in a $B \times B$ block can be found by

$$p = \frac{B^2}{N^2} \tag{6.5}$$

where N^2 is the image resolution. Since the total number of atoms and the probability of each atom being in block i, j are known the probability of having $k_{i,j}$ atoms in block i, j will have a binomial distribution and can be found by

$$p(k_{i,j}) = \binom{K}{k_{i,j}} \left(\frac{B^2}{N^2}\right)^{k_{i,j}} \left(1 - \frac{B^2}{N^2}\right)^{K-k_{i,j}} \tag{6.6}$$

where K is the total number of atoms. Note that since in each stage of MP coding an atom is generated, the total number of atoms is the same as the number of stages of MP. Thus, $p(k_{i,j})$ has binomial distribution with mean $K\frac{B^2}{N^2}$ and variance $K\frac{B^2}{N^2}(1 - \frac{B^2}{N^2})$. For large values of K, binomial distribution can be approximated by normal distribution with the same mean and variance.

$$p(k_{i,j}) \simeq N\left(K\frac{B^2}{N^2}, K\frac{B^2}{N^2}\left(1 - \frac{B^2}{N^2}\right)\right) \tag{6.7}$$

Since $k_{i,j}$ has a normal distribution its entropy can be found by:

$$Ent(k_{i,j}) = \ln(\sigma\sqrt{2\pi e}) = \ln\sqrt{2\pi e K\frac{B^2}{N^2}\left(1 - \frac{B^2}{N^2}\right)} \tag{6.8}$$

Thus, the best rate that can be achieved by entropy coding $k_{i,j}$ is found as:

$$Rate(k_{i,j}) = \ln\sqrt{2\pi e K\frac{B^2}{N^2}\left(1 - \frac{B^2}{N^2}\right)} \tag{6.9}$$

Now that the rate and the distortion are found in terms of block size B and number of quantization levels $q_{i,j,k}$, the optimal values of B and $q_{i,j,k}$ can be found using Lagrangian optimization. The Lagrangian cost function can be written as:

$$J(\lambda) = D + \lambda R \tag{6.10}$$

where D and R are the distortion and rate respectively and λ is the Lagrangian multiplier. Substituting the values of rate and distortion into the Lagrangian cost function we can write:

$$J(\lambda) = \|R^K f\|^2 + \sum_{i=0}^{\frac{N}{B}-1} \sum_{j=0}^{\frac{N}{B}-1} \sum_{k=0}^{k_{i,j}-1} \frac{I_{i,j,k}^2}{12 q_{i,j,k}^2}$$

$$+\lambda \left(\sum_{i=0}^{\frac{N}{B}-1} \sum_{j=0}^{\frac{N}{B}-1} \left(\ln \sqrt{2\pi e K \frac{B^2}{N^2} \left(1 - \frac{B^2}{N^2}\right)} + \sum_{k=0}^{k_{i,j}-1} (\log_2 q_{i,j,k} + \log_2 M + 1 + 2\log_2 B) \right) \right) \tag{6.11}$$

The optimum value of number of quantization levels can be found by taking the partial derivative of $J(\lambda)$ with respect to $q_{i,j,k}$:

$$\frac{\partial J(\lambda)}{\partial q_{i,j,k}} = 0 \tag{6.12}$$

The optimum value of $q_{i,j,k}$ can be computed as:

$$q_{i,j,k} = \sqrt{\frac{I_{i,j,k}^2 \ln 2}{6\lambda}} \tag{6.13}$$

Note that the above equation results in:

$$\frac{I_{i,j,k}}{q_{i,j,k}} = \frac{I_{i',j',k'}}{q_{i',j',k'}} \tag{6.14}$$

The above equation shows the optimum quantization is achieved when the same quantization step size is used for all atom coefficients.

The next step is to find the optimum value of λ

$$\frac{\partial J(\lambda)}{\partial \lambda} = 0 \tag{6.15}$$

As shown in Appendix B, λ can be found as a function of K, B and $I_{i,j,k}$ where K is the total number of matching pursuit stages.

$$\lambda = f(K, B, I_{i,j,k}) \tag{6.16}$$

In this chapter we find the optimum MP parameters for a given number of MP stages which indirectly relates to the final bit rate. In order to optimize for a certain bit rate, different values of K must be selected until the desired bit rate is achieved.

Now the optimum value of block size B must be computed. Since, $q_{i,j,k}$ and λ are found in terms of B, the optimum value of B can be found by an exhaustive search over all values of B and selecting the one that minimizes $J(\lambda)$. Since B can only take on integer values this exhaustive search can be completed with only a few iterations. Moreover, since the atom positions within the blocks must be encoded, we restrict ourselves to block sizes of powers of two so that atom positions can be efficiently encoded. This further reduces the number of options for B and thus, the number of iterations required to find optimal value of B.

6.1.1 Practical implementation

In computation of optimum block size and number of quantization levels we assume the values of MP distortion $\|R^K f\|^2$ and the range of quantizers $I_{i,j,k}$ are known. The MP distortion can be found by training over several images and finding the average value. Also the quantization range $I_{i,j,k}$ can be found by finding the average histogram of inner product coefficients for each stage at each block size. However, the optimum value of $I_{i,j,k}$ depends on the number of quantization levels $q_{i,j,k}$ which in return depends on $I_{i,j,k}$. While an iterative algorithm can be designed to find the optimum $I_{i,j,k}$ and $q_{i,j,k}$ it is easier to find the ranges adaptively during encoding. Additionally the value of D_{MP} can be found during encoding. This enables the parameters to be optimized based on the operating rate distortion function, which in

general results in better rate distortion performance than optimizing based on an average rate distortion curve. In order to realize this adaptation we perform the optimization after all atoms are found. Thus, first atoms are found and are reordered based on the magnitude of inner product coefficients. Once atoms are found the unquantized MP distortion D_{MP} can be easily computed by computing the square norm of the residual image. Since in each block atoms are encoded in the order of the magnitude of inner product coefficients, the range of each coefficient is between zero and the value of inner product coefficient of the previous stage. For the first stage in each block, the range of coefficients $I_{i,j,1}$ is between zero and the maximum inner product coefficient. This value is sent to the decoder using 8 bits. In order to find the optimum block size and quantization levels the Lagrangian cost function is computed for all values of block sizes between 1 and N that are powers of two. The block size B and its corresponding $\lambda = f(K, B, I_{i,j,k})$ that minimize the Lagrangian cost function are selected and used for decoding. B is sent to the decoder using $\lceil \log_2 log_2 N \rceil$ bits. Instead of sending λ to the decoder, the value of $q_{i,j,1} = 2^{\text{round}(\log_2 \sqrt{I_{i,j,1}^2 \ln 2/6\lambda})}$ is computed and sent to the decoder using 8 bits. Note that $q_{i,j,1}$ can only take on integer values that are powers of two. This is to ensure each coefficient can be efficiently encoded using binary strings of length $\log_2 q_{i,j,1}$. Then blocks are scanned in the raster scan order and atoms are encoded. For each block first the number of atoms inside that block (i.e. $k_{i,j}$) must be encoded. Since the probability distribution of $k_{i,j}$ is known, Huffman coding can be used to encode $k_{i,j}$ at a rate close to its entropy. However, if the entropy of $k_{i,j}$ is less than 1, Huffman coding is not very efficient and alternative coding algorithms like arithmetic coding, extended Huffman coding or run-length coding must be used [27]. In all our experiments the optimum block size found by our optimization results in a larger than 1 entropy for $k_{i,j}$. Therefore Huffman coding can be used to achieve bit rates close to entropy. The probability distributions used to find the Huffman codes are the ones found by equation (6.7).

Once $k_{i,j}$ is encoded, the atoms inside each block must be encoded in the order of their inner product coefficient. As mentioned before, $I_{i,j,1}$ and $q_{i,j,1}$ are fixed for all blocks and are sent to the decoder. The first atom coefficient $c_{i,j,1}$ is quantized by $q_{i,j,1}$ levels in the range $I_{i,j,1}$. Let us denote the quantized value by $c_{q_{i,j,1}}$. Since atoms are reordered in the order of the magnitude of their inner product coefficients, the range of the next coefficient is equal to the previous quantized coefficient. Therefore, $I_{i,j,k+1} = c_{q_{i,j,k}}$ for $k \geq 1$ and $q_{i,j,k+1} = 2^{\text{round}(\log_2 \frac{I_{i,j,k+1}}{I_{i,j,k}} q_{i,j,k})}$. The latter equation is derived from equation (6.14) and the values of $q_{i,j,k}$ are restricted to integer values that are powers of two. Once the atom coefficient is quantized, $\log_2 q_{i,j,k}$ bits are sent for the quantized absolute inner product coefficient and one bit is sent for the sign bit of coefficient. The atom position inside the block can be encoded by $2\log_2 B$ bits and the dictionary index can be encoded by $\log_2 M$ bits. Note that all quantization levels and ranges can be computed at the decoder side and there is no need to send any extra side information. Our proposed algorithm can be summarized as follows:

1. Find all atoms by MP and reorder them based on their inner product coefficients.
2. Compute distortion $D_{\text{MP}} = \|R^K f\|^2$
3. For $\log_2 B = 0$ to $\log_2 N$
   - Set $I_{i,j,1}$ = maximum inner product coefficient
   - Set $I_{i,j,k} = c_{i,j,k-1}$ for $k > 1$
   - Find $\lambda = f(K, B, I_{i,j,k})$
   - Find $J(\lambda, B)$
4. Choose B and λ that result in minimum $J(\lambda, B)$
5. Set $I_{i,j,1}$ = maximum coefficient quantized by 8 bits and transmit to the decoder

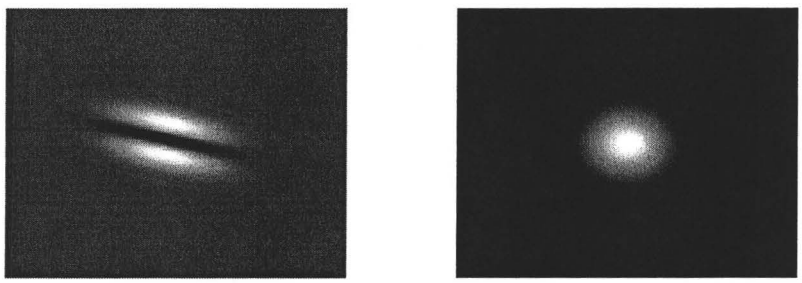
6. Set $q_{i,j,1} = 2^{\text{round}(\log_2 \sqrt{I_{i,j,1}^2 \ln 2/6\lambda})}$ and transmit to decoder using 8 bits

7. For $i = 1$ to $\log_2 \frac{N}{B}$

   - For $j = 1$ to $\log_2 \frac{N}{B}$
     - Find the Huffman code for $k_{i,j}$ (the number of atoms in block i, j) and transmit the code
     - For $k = 1$ to $k_{i,j}$
       * $c_{q_{i,j,k}}$ = Quantize $c_{i,j,k}$ by $q_{i,j,k}$ levels in the range $I_{i,j,k}$, transmit $c_{q_{i,j,k}}$ by $\log_2 q_{i,j,k}$ bits.
       * transmit the atom position by $2\log_2 B$ bits, dictionary index by $\log_2 M$ bits and sign of the coefficient by 1 bit.
       * $I_{i,j,k+1} = c_{q_{i,j,k}}$
       * $q_{i,j,k+1} = 2^{\text{round}(\log_2 \frac{I_{i,j,k+1}}{I_{i,j,k}} q_{i,j,k})}$

6.2 Simulation results

In this section our proposed encoding algorithm is compared with the ones designed in [41] and [9]. Note that [9] is the algorithm described in chapter 5. We also compare our results with rate distortion curves achieved by the JPEG2000 codec. The JPEG2000 rate distortion curves are obtained using the KaKadu software [61]. The dictionary used in our simulation results is based on the dictionary proposed in [24]. This dictionary is built based on anisotropic scaling, rotation and translation of a function that is optimized for efficient approximation of contours in two dimensions and therefore is very efficient in representing contours and edges in images [24]. However, in order to efficiently represent low frequency components in the image a Gaussian function with a number of isotropic scales is also included in this dictionary.



(a) (b)

Figure 6.1: Example of dictionary elements used in our experiments (a) anisotropic atom, (b) Gaussian atoms.

Two atoms from this dictionary are shown in Fig. 6.1 (see [24] for more information on the design of this dictionary). We applied the proposed algorithm and the ones designed in [41] and [9] to quantize and encode MP coefficients and indexes. The above dictionary is used in all our experiments. Our simulation results are shown in Fig. 6.2, 6.3 and 6.4. As can be seen in the figures, our proposed algorithm outperforms the encoding algorithms designed in [41] and [9] in all bit rates. The improvements are more significant in higher bit rates since in high bit rates it is very inefficient to encode atoms in their importance order. This is because for high bit rates there are many atoms that need to be encoded and therefore, encoding their position information becomes very expensive in algorithms designed in [41] and [9]. Nevertheless, our results show that using the correlations between the atom positions and inner product coefficients simultaneously yields significant improvements in all bit rates compared to previous MP encoding techniques.

Additionally our proposed algorithm outperforms JPEG2000 at low bit rates. The reason for the improved performance is that MP dictionary is highly flexible and includes atoms in various positions, scales, directions and frequencies. This allows

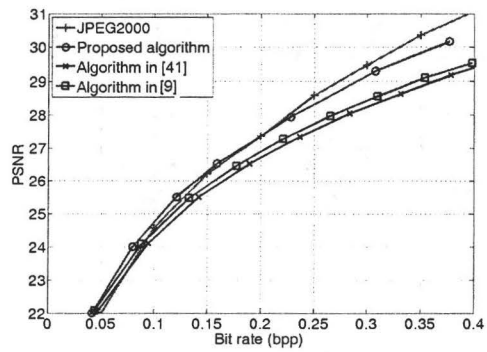


Figure 6.2: Comparison of PSNR versus bit rate curves for the proposed MP coding algorithm and the algorithms designed in [41], [9] and JPEG2000 for *Lena* image

MP algorithm to find the best match for the dominant features of the image in the early stages. However, wavelet or DCT transforms can only use the set of wavelet or DCT bases to represent data which are much more limited sets than the over-complete dictionary. Therefore, MP can extract important features and contours in the image using one of the atoms in the dictionary while wavelet or DCT transforms may need many transform coefficients to extract the same information. The flexibility of choosing atoms offered by the over-complete dictionary is the main reason why matching pursuit based encoder outperforms JPEG2000 at low bit rates.

The improvements are more significant in the case of *Barbara* image and our algorithm outperforms JPEG2000 for bit rates less than 0.23 bpp. This is mainly because the *Barbara* image contains many directional edges which cannot be efficiently represented by the wavelet transform. The dictionary used in our experiments includes atoms that are rotated to different angles and therefore can efficiently represent directionality in images.

The dictionary used in our experiments is not optimized to capture texture and details in the image. Therefore, once the dominant contours and features of the

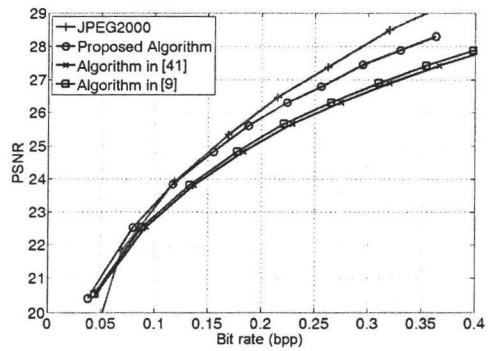


Figure 6.3: Comparison of PSNR versus bit rate curves for the proposed MP coding algorithm and the algorithms designed in [41], [9] and JPEG2000 for *Cammera man* image

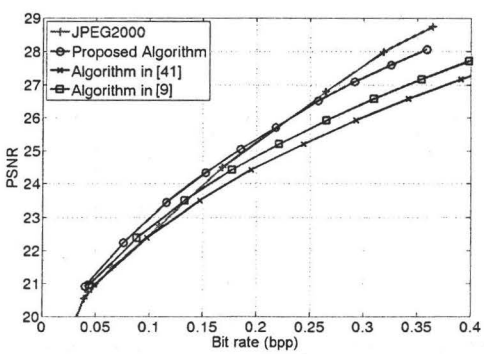


Figure 6.4: Comparison of PSNR versus bit rate curves for the proposed MP coding algorithm and the algorithms designed in [41], [9] and JPEG2000 for *Barbara* image

(a) (b)

Figure 6.5: *Lena* image encoded at 0.30 bpp , (a) JPEG2000 PSNR 29.47 (b) MP PSNR 29.15

image are extracted by the MP algorithm, the correlation between the residual image and the dictionary atoms decreases and thus, it becomes more difficult to find good matches for the patterns and features that exist in the residual image. This is the main reason why MP is not very efficient in higher bit rates. Perhaps a different dictionary must be used at higher bit rates to adapt to the changing properties of the residual image.

Fig. 6.5 shows the *Lena* image encoded by MP and JPEG2000 at the rate 0.3 bpp. While PSNR of JPEG2000 is higher than MP, the MP encoded image is visually more pleasant than the one encoded by JPEG2000. This is particularly evident around and on the hat in the *Lena* image. The main reason for the improved visual quality is the flexibility offered by matching pursuit dictionary. Wavelets are only efficient for capturing vertical and horizontal correlations in the image. However, the dictionary used in our experiment is capable of approximating contours and correlations in any direction. This is achieved by rotating the basis functions in a number of angles which results in efficient capturing of directional correlations in the image. This can

be seen by comparing the non-vertical and non-horizontal edges in the image shown in Fig. 6.5. While these edges are efficiently represented by the MP dictionary, wavelet transform used in JPEG2000 was unable to use the directional correlations that exist in these edges.

6.3 Conclusion

In this chapter we proposed an algorithm for optimal encoding of atom positions and inner product coefficients in matching pursuit image coding. We showed that our algorithm outperforms other existing encoding algorithms proposed for MP image coding. We also showed that MP image coding outperforms JPEG2000 in low bit rates. This shows that over-complete signal expansion can potentially improve image compression performance when efficient entropy coding algorithms are used to exploit the existing correlations among image components in an over-complete expansion.

Chapter 7

Concluding Remarks and Future Direction

7.1 Conclusion

In this dissertation we studied the progressive coding of image and Gaussian data using the matching pursuit algorithm. This study is motivated by the rapid growth in the applications of compression algorithms and the necessity of better understanding of alternative compression techniques to the ones used in conventional image compression algorithms. We also focused on reducing the gaps between theoretical bounds and practical compression performances and developed new compression techniques that outperform existing algorithms. We started with an introduction in chapter 1 where we outlined our motivations and summarized our contributions. We provided the necessary background in chapter 2 and summarized some of the most widely used compression techniques and discussed the techniques used in this thesis. More specifically, we focused on over-complete signal representations using matching pursuit and discussed how it is used for compressing still images. Additionally, we explained the applications of progressive coding and discussed the theoretical bounds that exist for

progressive coding of certain sources.

In chapter 3, we studied the application of matching pursuit in progressive coding of i.i.d Gaussian sources. We designed a novel progressive MP encoder and optimized the encoder parameters using Lagrangian optimization. In order to do the optimization we found the relationship between the MP distortion and the encoder parameters and optimized the encoder to achieve the best rate distortion performance. In-loop quantization was used and a novel quantization algorithm was proposed based on the properties of MP inner product coefficients. Additionally, we studied the convergence of our quantization algorithm and found the conditions that guarantee rapid convergence of the quantization algorithm. The progressive MP encoder was used to encode i.i.d Gaussian sources and our simulation results showed that our new MP encoder outperforms existing quantizers designed for progressive coding of Gaussian sources.

Although the MP encoder designed in chapter 3 outperforms all the existing progressive encoders designed for Gaussian sources, its rate distortion performance is not comparable to non-progressive quantizers designed for Gaussian sources. In fact the progressive structure is achieved at the cost of reduced compression quality. However, Gaussian sources are known to be successively refinable and thus progressive coding theoretically should not cause any decrease in the rate distortion performance. Although we were able to outperform all progressive Gaussian source coders with the MP encoder designed in chapter 3, we have not been able to close the gap between the rate distortion function of a practical progressive encoder and the theoretically achievable rate distortion function. The main reason for the reduced performance was the use of the same dictionary for all refinement stages. We showed that due to orthogonality of the MP residual vector and the dictionary vector selected at the first stage, the range of the residual vector is one dimension less than the dimension of the MP dictionary. Therefore, using the same dictionary in all stages results in some redundancy which reduces the rate distortion performance.

In chapter 4 we addressed this problem and designed a successively refinable quantizer for Gaussian sources and quantized the residual vector at the correct space to remove the redundancy in the dictionary. However, practical limitations caused additional sources of redundancy in the new algorithm and our simulation results showed that the new progressive encoder has very similar performance to the MP encoder designed in chapter 3. A high resolution analysis was presented for the quantizer designed in chapter 4 and we explicitly derived the distortion increase caused by successive refinement. Our simulation results showed that the distortion increase caused by successive refinement in the practical case of low resolution quantization is very close to our high resolution results. This is mainly due to the high performance of the Leech lattice and the wrapped spherical codes used in our quantizer.

In the next two chapters we focused on the more practical case of image coding and studied the application of matching pursuits in image compression. Most conventional image codecs use transforms to represent image data in a compact form and achieve compression. In our study we used over-complete image expansion using the matching pursuit algorithm to achieve a compact representation of the image. Our objective in this study was to find out how much compression can be achieved by using over-complete image representations instead of transform coding. We used the best dictionary designed for image coding applications and focused on finding efficient quantization and entropy coding techniques in order to achieve the best rate distortion performance.

In chapter 5 we proposed a new adaptive in-loop quantization technique for quantization of MP inner product coefficients and used it in an image coding application. Our adaptive algorithm is designed mainly based on the findings in chapter 3 and the analysis of the in-loop quantizer in chapter 3 is used in modelling the rate distortion performance. We optimized our quantizer based on the operational rate distortion function and showed that our new quantizer is more efficient than the quantizers used

for quantization of MP coefficients in image coding applications.

Although the quantizer designed in chapter 5 improved the rate distortion performance of MP image coding, our results were still not better than what can be achieved by common transform coding techniques (e.g. JPEG2000). Therefore, we tried to improve the performance of MP image coding by designing better MP encoding techniques and proposed an optimal MP image coder in chapter 6. In the new MP image encoder, we used all the correlations that exist between atom positions and inner product coefficients and designed an encoding algorithm that uses all these correlations to achieve optimum rate distortion performance. We showed that significant improvements can be achieved by using optimum coding of atom positions and coefficients over existing MP image coders and the one designed in chapter 5. We compared our results with JPEG2000 encoder which uses the wavelet transform. Our comparison showed that MP image coding results in improved rate distortion performance at low bit rates while it can provide better visual quality at moderate bit rates. The reason for the improved performance is the flexibility offered by the overcomplete dictionary used by the MP algorithm. The dictionary used by our encoder includes atoms that can efficiently approximate two dimensional contours, edges and other features in any scale and direction in the image. This results in a very compact representation of the predominant features in the image which are extracted at the early stages. Our proposed optimum atom position and coefficient coding finds the minimum number of bits required for encoding this information and therefore results in very efficient compression qualities at lower bit rates. Conventional transform coding techniques only use the limited set of transform bases to represent image features and therefore, they cannot represent image features as efficiently as the MP algorithm. Additionally most transforms including the wavelet transform used in JPEG2000 only use the correlations in vertical and horizontal directions. However, due to the high level of flexibility offered by the MP dictionary, the atoms in our MP dictionary can

approximate contours and edges in any direction. Thus, our proposed MP encoder results in visually pleasing decoded images that contain less ringing artifacts along edges than the images encoded by JPEG2000. This results in improved visual quality for the MP encoded images at moderate bit rates even though the PSNR values are in favor of JPEG2000 at these bit rates. However, since the dictionary used in our MP image codec is not designed to capture texture and detail in the image or patterns in the residual image, the high bit rate performance of our MP coder is not comparable with JPEG2000. In summary our study of MP image coding showed that over-complete image representation is very promising for image compression and can overcome some of the limitations of current transform coding techniques by the flexibility offered by the over-complete dictionary.

7.2 Future direction

Our study of progressive coding of Gaussian sources showed that although the new progressive coders designed in chapters 3 and 4 outperform the existing progressive coders for Gaussian sources, they are still not able to achieve the rate distortion performance predicted by the theoretical results. Although achieving the rate distortion function of the Gaussian source may not be possible with limited computational complexity, achieving the rate distortion function of the best non-progressive Gaussian quantizers by a progressive quantizer may not need unlimited computational complexity. The performances of the quantizers designed in chapters 3 and 4 while much better that the existing progressive solutions are not comparable with non-progressive rate distortion results and better algorithms can still be designed to close the gap between progressive coding and non-progressive quantization results.

Our study of MP image coding showed that matching pursuit can potentially result in improved compression performance. However, the existing dictionaries designed

for image coding applications do not include atoms optimized for capturing texture and detail in images. Additionally, they are not very efficient in extracting patterns and correlations that remain in the residual images. Therefore MP image coding is inefficient in high bit rates. Dictionaries optimized for high bit rate coding should be designed if MP image coding is to be used at higher bit rates. Perhaps, the dictionary elements should be used adaptively to improve rate distortion performance. Atoms optimized for extracting contours, edges and low frequency components in the image should be used first and once all the dominant features are extracted from the image, different atoms optimized for approximating details, texture and patterns in the residual image should be used.

Although the dictionary used by the MP algorithm has a huge impact on the rate distortion performance of the MP image encoder, we showed in chapter 6 that significant compression gain can be achieved by exploiting redundancies between atom positions and coefficients. However, in the algorithm designed in chapter 6, we only considered the correlations between atom positions and inner product coefficients. The dictionary indexes or more specifically the scaling, rotation and modulation parameters are highly correlated and their dependencies can be used to design better entropy coding techniques and achieve a better rate distortion performance.

Another possible approach for improving the compression performance of MP image coding is to design dictionaries that are optimized for better entropy coding. In transform coding, it is known to the encoder that most of the image information is contained in the low frequency transform coefficients. Therefore, although the image representation in the transform domain is less compact than the MP image representation, significant bit rate saving is achieved by entropy coding, since the behavior of transform coefficients is more or less known to the entropy coder. In contrast, the atoms selected by the MP algorithm are usually very random and therefore, entropy

coding cannot be as efficient as it is in transform coding applications. Perhaps dictionaries can be designed that show more predictable behavior which can result in better compression achieved by entropy coding techniques.

Another factor that limits the use of MP in image coding applications is the high computational complexity required at the encoder. The MP algorithm finds the best dictionary element by an exhaustive search over all dictionary elements which can be prohibitive in certain applications. Many algorithms have been proposed to reduce the computational complexity of the search in the MP algorithm [16, 22, 25, 58–60]. However, more work can be done in this area since the current solutions mostly come at the price of reduced performance or the reduction in computational complexity is not enough for many applications. Perhaps a small size adaptive dictionary optimized for efficient entropy coding is ideal for solving the computational complexity problems while it can improve the rate distortion performance at high bit rates.

Appendix A

Optimum Lagrangian multiplier for MP encoder in chapter 5

Substituting the optimum values of q_i from equation (5.21) into equation (5.20), the Lagrangian cost function can be written as:

$$J(\lambda) = \|f\|^2 \delta_0^2 \delta_2^1 ... \delta_{k-1}^2 + \frac{k\lambda\mu_q}{2\ln 2} + \lambda \left(\mu_q \sum_{i=0}^{k-1} \log_2 \left(\sigma_{c_i} \delta_{i+1} ... \delta_{k-1} \alpha \sqrt{\frac{2\ln 2}{3\lambda\mu_q}} \right) + \mu_M \log_2 M \right) \tag{A.1}$$

or

$$J(\lambda) = \|f\|^2 \delta_0^2 \delta_1^2 ... \delta_{k-1}^2 + \frac{k\lambda\mu_q}{2\ln 2} +$$

$$\lambda\mu_q \sum_{i=0}^{k-1} \log_2 \left(\sigma_{c_i} \delta_{i+1} ... \delta_{k-1} \alpha \sqrt{\frac{2\ln 2}{3\mu_q}} \right) + \lambda k \mu_q \log_2 \lambda^{-\frac{1}{2}} + \lambda k \mu_M \log_2 M \tag{A.2}$$

The partial derivative in terms of λ can be computed as:

$$\frac{\partial J(\lambda)}{\partial \lambda} = \frac{k\mu_q}{2\ln 2} + \mu_q \sum_{i=0}^{k-1} \log_2 \left(\sigma_{c_i} \delta_{i+1} ... \delta_{k-1} \alpha \sqrt{\frac{2\ln 2}{3\mu_q}} \right) -$$

$$\frac{k\mu_q}{2} \log_2 \lambda - \frac{k\mu_q}{2\ln 2} + k\mu_M \log_2 M = 0 \tag{A.3}$$

Therefore:

$$\log_2 \lambda = \frac{2}{k} \sum_{i=0}^{k-1} \log_2 \left(\sigma_{c_i} \delta_{i+1} ... \delta_{k-1} \alpha \sqrt{\frac{2 \ln 2}{3 \mu_q}} \right) + 2 \frac{\mu_M}{\mu_q} \log_2 M \tag{A.4}$$

Thus, λ can be expressed as a function of k, σ_{c_i} and δ_i:

$$\lambda = f(\sigma_{c_i}, \delta_i, k) \tag{A.5}$$

Appendix B

Optimum Lagrangian multiplier for MP encoder in chapter 6

Substituting the optimum values of $q_{i,j,k}$ from equation (6.13) into equation (6.11), the Lagrangian cost function can be written as:

$$J(\lambda) = \|R^K f\|^2 + \frac{K\lambda}{2\ln 2} +$$

$$\lambda\left(\sum_{i=0}^{\frac{N}{B}-1}\sum_{j=0}^{\frac{N}{B}-1}\left(Rate(k_{i,j}) + \sum_{k=0}^{k_{i,j}-1}(-\frac{1}{2}\log_2\lambda + \log_2\sqrt{\frac{I_{i,j,k}^2\ln 2}{6}} + \log_2 M + 1 + 2\log_2 B)\right)\right) \tag{B.1}$$

The partial derivative in terms of λ can be computed as:

$$\frac{\partial J(\lambda)}{\partial\lambda} = \frac{K}{2\ln 2} + \sum_{i=0}^{\frac{N}{B}-1}\sum_{j=0}^{\frac{N}{B}-1}\left(Rate(k_{i,j}) + \sum_{k=0}^{k_{i,j}-1}(\log_2\sqrt{\frac{I_{i,j,k}^2\ln 2}{6}} + \log_2 M + 1 + 2\log_2 B)\right)$$

$$+(-\frac{K}{2}\log_2\lambda) + (-\frac{K}{2\ln 2}) = 0 \tag{B.2}$$

Therefore:

$$\log_2\lambda =$$

$$\frac{2}{K}\left(\sum_{i=0}^{\frac{N}{B}-1}\sum_{j=0}^{\frac{N}{B}-1}\left(\ln\sqrt{2\pi eK\frac{B^2}{N^2}\left(1-\frac{B^2}{N^2}\right)}+\sum_{k=0}^{k_{i,j}-1}(\log_2\sqrt{\frac{I_{i,j,k}^2\ln 2}{6}}+\log_2 M+1+2\log_2 B)\right)\right) \tag{B.3}$$

Thus, λ can be expressed as a function of K, B, and $I_{i,j,k}$:

$$\lambda = f(K, B, I_{i,j,k}) \tag{B.4}$$

Bibliography

[1] ISO/IEC JTC1/SC29/WG11 13818-1, "Coding of moving pictures and associated audio," Nov. 1994.

[2] www.youtube.com

[3] J. Ziv and A. Lempel, "A universal algorithm for sequential data compression," *IEEE Trans.Inf. Theory,* vol. IT-23, no. 3, pp. 33743, Mar. 1977.

[4] W. B. Pennebaker and J. L. Mitchell, "JPEG Still Image Compression Standard" New York: Van Nostrand Reinhold, 1992.

[5] ISO/IEC 14496-2 Information technologyCoding of audio-visual objects: Visual ISO/IEC 14496-2 committee drafts, , Nov. 1997.

[6] ITU-T Recommendation H.264 and ISO/IEC 14496-10, Advanced Video Coding for Generic Audiovisual Services, May 2003.

[7] SMPTE 421M, "VC-1 Compressed Video Bitstream Format and Decoding Process," 2006.

[8] A. Shoa, S. Shirani, "Progressive Coding of a Gaussian Source Using Matching Pursuit," *IEEE Transactions on Signal Processing,* vol. 56, 2008, pp. 636-649.

[9] A. Shoa, S. Shirani, "Adaptive Rate-Distortion Optimal In-loop Quantization for Matching Pursuit," *IEEE Transactions on Image Processing,* vol. 17, No 9, Sep. 2008, pp. 1616-1623.

[10] A. Shoa, S. Shirani, "Design and Analysis of a Successively Refinable Lattice Quantizer for i.i.d. Gaussian Sources," *IEEE Transactions on Signal Processing,* vol. 57, no. 3, March 2009, to be published.

[11] A. Shoa, S. Shirani, "Optimal Atom Position and Coefficient Coding for Matching Pursuit Based Image Compression," *Submitted to IEEE Transactions on Image Processing,* under minor revisions.

[12] A. Shoa, S. Shirani, "A Successively Refinable Gain Shape Vector Quantizer for Gaussian Sources." *accepted in the Information Theory Workshop,* 2008

[13] , Shoa, S. Shirani, "Adaptive quantization for matching pursuit," *Proceedings of IEEE International Workshop on Multimedia Signal Processing,* 2006, pp. 71-64.

[14] A. Shoa, S. Shirani, "Optimization of matching pursuit encoder based on analytical approximation of matching pursuit distortion," *Proceedings of IEEE International Conference on Multimedia & Expo*, 2006, pp. 749-752.

[15] A. Shoa, S. Shirani, "Matching pursuit distortion: modeling and optimization," *Proceedings of Data Compression Conference*, 2006, pp. 408.

[16] A. Shoa, S. Shirani, "Tree Structure search for matching pursuit," *Proceedings of IEEE International Conference on Image Processing* 2005, pp. 908-911

[17] C. E. Shannon, "A Mathematical Theory of Communication", *Bell System Technical Journal,* Vol. 27, 1948, pp. 379-423, 623-656.

[18] W.H.R. Equitz, T.M. Cover, "Successive refinement of information," *IEEE Trans. Inf. Theory*, Vol. 37, Issue 2, 1991, pp. 269-275.

[19] H. Jafarkhani, V. Tarokh, "Design of successively refinable trellis-coded quantizers," *IEEE Trans. Inf. Theroy*, Vol. 45, No. 5, July 1999, pp. 1490-1497.

[20] A. Aksu, M. Salehi, "Multistage trellis coded quantization (MS-TCQ) design and performance," *IEE Proc. Commun.*, Vol. 144. No. 2, April 1997, pp. 61-64.

[21] A. Aksu, M. Salehi, "Design, performance and complexity analysis of residual trellis-coded vector quantization," *IEEE Trans. Commun.*, Vol. 46, No. 8, August 1998, pp. 1020-1026.

[22] R. Neff, A. Zakhor, "Very low bit rate video coding based on matching pursuit," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 7, 1997, pp. 158-171.

[23] C. De Vleeschouwer, A. Zakhor, "In-loop atom modulus quantization for matching pursuit and its application to video coding," *IEEE Trans. Image Process.*, Vol. 12, Issue 10, Oct. 2003 pp. 1226-1242.

[24] R.M. Figueras i Ventura, P. Vandergheynst, P. Frossard, "Low-rate and flexible image coding with redundant representations," *IEEE Trans. Image Process.*, Vol. 15, Issue 3, March 2006, pp. 726-739.

[25] S.G. Mallat, Z. Zhang, "Matching pursuits with time-frequency dictionaries " *IEEE Tran. Signal Process.*, vol. 41, 1993, pp. 3397-3415.

[26] P. Vogel, "Analytical coding of Gaussian sources," *IEEE Trans. Inf. Theory*, vol. 40, Sept. 1994, pp. 1639-1645.

[27] K. Sayood, "Introduction to Data Compression," *Morgan Kaufmann*, Second Edition, 2000.

[28] D.J. Newman. "The hexagon theorem," *IEEE Trans. on Inform. Theory*, vol. 28, March 1982, pp. 137-139.

[29] Y. Linde, A. Buzo, and R.M. Gray, "An algorithm for vector quantization design," *IEEE Trans. on Commun.*, vol. 28, Jan. 1980, pp. 84-95.

[30] M.M. Den, "Optimization by Variational Methods", *McGraw-Hill*, 1969.

[31] J. H. Conway, N. J. A. Sloane, "Sphere Packings, Lattices and Groups," Third ed. *New York: Springer-Verlag*, 1998.

[32] J. D. Gibson, K. Sayood, "Lattice quantization," *Advances in Electronics and Electron Physics*, New York: Academic, 1988, pp. 259-332.

[33] K. T. McDonald, "Volume and surface area of an N-sphere," *http://www.hep.princeton.edu/~mcdonald/examples/nsphere.pdf.*

[34] A. Papoulis, "Probability, Random Variables, and Stochastic Processes", *McGraw-Hill*, 2nd ed., 1984.

[35] D.J. Sakrison, "A geometric treatment of the source encoding of a Gaussian random variable," *IEEE Trans. Inf. Theory*, May 1968, pp. 481-486.

[36] J. Hamkins, K. Zeger, "Asymptotically efficient spherical codes Part I: Wrapped spherical codes," *IEEE Trans. Inf. Theory*, vol. 43, Nov. 1997, pp. 1774-1785.

[37] J. Hamkins, K. Zeger, "Gaussian source coding with spherical codes," *IEEE Trans. Inf. Theory*, vol. 48 no. 11, Nov. 2002, pp. 2980-2989.

[38] S. Mallat, "A Wavelet Tour of Signal Processing" *Academic Press*, 1999.

[39] G.M. Davis, S. Mallat and M. Avelanedo. "Greedy adaptive approximations," *J. of Constr. Approx.*, vol. 13, 1997, pp. 57-98.

[40] Q. Liu, Q. Wang, L. Wu, "Size of the dictionary in matching pursuit algorithm," *IEEE Trans. Signal Process.*, Vol. 52, Issue 12, Dec. 2004, pp. 3403-3408.

[41] P. Frossard, P. Vandergheynst, R.M. Figueras i Ventura, M. Kunt, "A posteriori quantization of progressive matching pursuit streams" *IEEE Trans. Signal Process.*, vol. 52, 2004, pp. 525-535.

[42] R. Laroia, N. Farvardin, "Trellis-based Scalar-vector Quantizer for Memoryless Sources," *IEEE Trans. Inf. Theory*, vol. 40, May 1994, pp. 860-870.

[43] S. G. Wilson, D. W. Lytle, "Trellis encoding of continuous-amplitude memoryless sources," *IEEE Trans. Inf. Theory*, vol. IT-28, May 1982, pp. 211-226.

[44] M.W. Marcellin, T. Fischer, "Trellis coded quantization of memoryless and Gauss-Markov sources," *IEEE Trans. Commun.*, vol. COM-38, Jan. 1990, pp. 82-93.

[45] H. S. Wang, N. Moayeri, "Trellis coded vector quantization," *IEEE Trans. Commun.*, vol. 40, Aug. 1992, pp. 1273-1276.

[46] D. G. Jeong, J. D. Gibson, "Uniform and piecewise uniform lattice vector quantization for memoryless Gaussian and Laplacian sources," *IEEE Trans. Inf. Theory*, vol. 39, May 1993, pp. 786-803.

[47] S. G. Wilson, "Magnitude/phase quantization of independent Gaussian variates," *IEEE Trans. Commun.*, vol. COM-28, Nov. 1980, pp. 1924-1929.

[48] N. S. Jayant, P. Noll, "Digital Coding of Waveforms," *Englewood Cliffs, NJ: Prentice-Hall*, 1984.

[49] P. Noll, R. Zelinski, "Bounds on quantizer performance in the low bit-rate region," *IEEE Trans. Commun.*, vol. COM-26, Feb. 1978, pp. 300-304.

[50] A. Gersho and R.M. Gray, "Vector Quantization and Signal Compression," Boston, MA: Kluwer Academic, 1993.

[51] D. Mukherjee and S. K. Mitra, "Successive refinement lattice vector quantization," IEEE Trans. Image Proc., Vol. 11, No. 12, Dec 2002, pp. 1337-1348.

[52] M.J. Sabin and R.M. Gray, "Product code vector quantizers for waveform and voice coding," IEEE Trans. Acoust., Speech, Signal, Process., ASSP-32, June 1984, pp. 148-155.

[53] D. S. Taubman, M. W. Marcellin, "JPEG 2000: Image Compression Fundamentals, Standards and Practice" *Kluwer Academic Publishers,* Norwell, MA, 2001.

[54] R. Neff, A. Zakhor, "Modulus quantization for matching-pursuit video coding," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 10, pp. 895-912, 2000.

[55] R. M. Gray, D. L. Neuhoff "Quantization," *IEEE Trans. Inform. Theory* vol. 44, No. 6, Oct. 1998, pp. 2325-2384

[56] S. S. Chen, D. L. Donoho, and M. A. Saunders, "Atomic decomposition by basis pursuit," *SIAM Rev.*, vol. 43, no. 1, pp. 12959, 2001.

[57] Y. C. Pati, R. Rezaiifar, and P. S. Krishnaprasad, "Orthogonal matching pursuit: Recursive function approximation with applications to wavelet decomposition," *Proc. 27th Annu. Asilomar Conf. Signals, Syst., Comput.,* 1993.

[58] P. Jost, P. Vandergheynst, P. Frossard, "Tree-Based Pursuit: Algorithm and Properties," *IEEE Trans. Signal Process.* vol. 54, Issue 12, Dec. 2006 pp. 4685-4697.

[59] C. De Vleeschouwer and B. Macq, "Subband dictionaries for low-cost matching pursuits of video residues," *IEEE Transactions on Circuits and Systems for Video Technology*, vol 9, 1999, pp. 984-993.

[60] Jeon Byeungwoo and Oh Seokbyung, "Fast matching pursuit with vector norm comparison," *IEEE Transactions on Circuits and Systems for Video Technology*, vol 13, 2003, pp. 338-342.

[61] http://www.kakadusoftware.com/