SPIHT-BASED IMAGE ARCHIVING UNDER BIT BUDGET CONSTRAINTS

BY

Yifeng He

A thesis submitted in conformity with the requirements
for the degree of Master of Applied Science,
Graduate School of Electrical Engineering and Computer Science,
in the University of Ottawa.

For my family

SPIHT-based Image Archiving Under Bit Budget Constraints

Master of Applied Science Thesis
School of Electrical Engineering and Computer Science
University of Ottawa

by Yifeng He
April 2013

Abstract

Image archiving is important in various applications. In this work, we formulate the problem of compressing image archives as an optimization problem and present an efficient method for solving this problem. Our method is based on adapting SPIHT to image archiving where the total bit budget is distributed optimally across the images. We show experimentally that this method significantly outperforms its counter-part, namely, SPIHT with equal distribution of bit budget.

Contents

Chapter 1

Introduction

Numerous images are produced daily. The scale of image data and correspondingly the size of image archives are witnessed to grow rapidly. This presents challenges for storing and communicating large sets of images or image archives, particularly in the circumstances where the communication or storage capacity is limited relative to the size of the image archive. Such scenarios include, for example, a smart phone with limited memory intending to save a large number of images, a medical image database to communicate a large set of images over a communication network. a social network administrator to back up the image data uploaded by the users, etc.

This thesis studies the problem of compressing image archives in such circumstances. There may be various formulations of such a problem and the one we adopt in this thesis is approximately as follows: give a (rather stringent) bit budget not greater than which the image archive has to be compressed into, how should the images be compressed such that they can be reconstructed to as high quality as possible?

In case when the images to compress have similar contents or are highly correlated, some previous works have approached the archiving problem by exploring the correlation across the images. One drawback of such an approach is that it does not work well when images in the archive are of diverse nature. In addition, with such an approach the decompression of a single image in the archive is usually computationally costly since reconstructing one image requires the reconstruction of (nearly) all its correlated images. This is undesirable in most applications. This latter issue also motivates us to

develop an archiving technique primarily based on the existing single-image compression algorithms. That is, we wish to adapt a single-image compression algorithm to image archive compression, so that the overall qualities of the reconstructed images are better and each image in the archive can be reconstructed separately.

An examination of various contemporary image compression algorithms suggests that SPIHT is the best for this purpose. In particular, SPIHT naturally supports the formulation of archiving problem as an optimization problem in which one attempts to distribute the bit budget to each image in the archive. To that end, we formulate the archiving problem as an integer program. We show that under a mild condition (certain concavity) the integer program can be relaxed into a linear program without changing its solution. We present a quadratic programming based algorithm to correct certain statistics of the image data so that the condition is met and then solve the archiving problem using a linear-programming algorithm.

Via extensive experiments, we show that the proposed archiving method outperforms it counterpart in which the bit budget is uniformly distributed across the images in the archive. More specifically, we show that the performance gain is over a large range of bit budget, and that for a moderate-sized archive, about 1 dB average PSNR gain is attainable.

The thesis is organized as follows.

In Chapter 2, we introduce background materials necessary for this work, which includes image transforms, transform-based image compression algorithms and the metrics for evaluating image qualities. We give particular emphasis to the SPIHT algorithm. We also introduce various forms of optimization problems that are relevant to this thesis.

In Chapter 3, we formulate the archiving problem as an optimization problem, discuss its properties and introduce our SPIHT-based archiving method.

In Chapter 4, we present the experimental set up, and experimental results.

The thesis is concluded in Chapter 5.

Chapter 2

Preliminaries

2.1 Image Transformations

In practice color images are of primary interest, but in the scope of this thesis, we consider only monochrome images. This is because of the well-known fact that upon certain decomposition, only luminal strength of a color matters the most.

A digital image of size M by N may be regarded as a real-valued function $\mathbf{p}$ on $\mathcal{S} := \{1, 2, \ldots, M\} \times \{1, 2, \ldots, N\}$. Every element $(i, j) \in \mathcal{S}$ is called a pixel, and $\mathbf{p}(i, j)$ is the grey level of the pixel (i, j). The set $\mathcal{S}$ is often referred to as the "spatial domain", and the function $\mathbf{p}$ is the spatial-domain representation of the image.

A transformation of image is a (linear) map Ω from the space of the all functions on $\mathcal{S}$ to the space of all functions on a different space, say $\mathcal{S}'$, where $\mathcal{S}'$ and $\mathcal{S}$ typically have the same size. Typically $\mathcal{S}'$, like $\mathcal{S}$, may also be regarded as $\{1, 2, \ldots, M\} \times \{1, 2, \ldots, N\}$, but each element of $\mathcal{S}'$ can no longer be interpreted as a spatial location. Well-known transformations include DFT(Discrete Fourier Transform), DCT(Discrete Cosine Transform) and DWT(Discrete Wavelet Transform). Transforming an image using these transformations allows powerful techniques for image analysis and processing.

2.1.1 DFT and DCT

Discrete Fourier Transform (DFT) is a linear map from the time or spatial domain to the frequency domain. The input of DFT, a function $f(x)$ on the time or spatial domain, is a function that only has non-zero values at N discrete x points in the domain, so we may rewrite $f(x)$ as an N dimensional vector $\mathbf{f} = \{f(0), \cdots, f(k), \cdots, f(N-1)\}$. Let $\mathbf{X} = \{\mathbf{x}_u\}$, $u = 0, 1, \cdots, N-1$ a set of N dimensional vectors. The vectors in $\mathbf{X}$ are selected to be linear independent in the sense that any of the vectors in the set cannot be represented as a non-trivial linear combination of the other vectors in the same set. Set $\mathbf{X}$ form a basis such that an N dimensional vector $\mathbf{f}$ can be represented as a linear combination of the vectors in $\mathbf{X}$ as $\mathbf{f} = \mathbf{X} \cdot \mathbf{c}$ where $\mathbf{X}$ is in the matrix notation with each column being a basic vector, $\mathbf{c}$ is the column vector containing the coefficients of the linear combination, and the product operation "$\cdot$" is the matrix-vector product. All the functions that can be represented as the linear combination of $\mathbf{X}$ form a space $V_{\mathbf{X}}$, it is spanned by $\mathbf{X}$. The N-dimensional vector $\mathbf{c}$ may be regarded as the representation of function $\mathbf{f}$ under basis $\mathbf{X}$. Let $\mathbf{x}_u = \{1, \cdots, \exp(i2k\pi u/N), \cdots, \exp((i2k\pi u(N-1))/N)\}$, the representation of $\mathbf{f}$ under basis $\mathbf{X}$ and its relationship with the spatial-domain function f can be expressed by equation (2.1). In these equations, each coefficient $c(u)$ corresponds a frequency component, and the vector $\mathbf{c}$ may be regarded as a frequent-domain representation of $\mathbf{f}$. The procedure to obtain the vector $\mathbf{c}$ as in equation (2.2) is the DFT, which provides a signal decomposition on frequency domain.

$$f(n) = \frac{1}{N} \sum_{u=0}^{u<N} c(u) \exp\left(\frac{i2k\pi u}{N}\right) \quad k = 0, 1, \cdots, N-1 \tag{2.1}$$

$$c(u) = \sum_{k=0}^{k<N} f(k) \exp\left(\frac{-i2k\pi u}{N}\right) \quad u = 0, 1, \cdots, N-1 \tag{2.2}$$

If f is a function on spatial domain $\mathcal{S} = \{1, 2, \cdots, M\} \times \{1, 2, \cdots, N\}$, or equivalently an $M \times N$ matrix, then the frequency-domain representation of f can be obtained by first applying DFT to each row of f, then applying DFT to each column of the resulting matrix. This gives rise to the notion of two-dimensional DFT. Two-dimensional DFT of an $M \times N$ matrix and its inverse are given below:

$$c(u,v) = \sum_{j=0}^{j<M} \sum_{k=0}^{k<N} f(j,k) \exp\left(\frac{-i2j\pi u}{M}\right) \exp\left(\frac{-i2k\pi v}{N}\right)$$
$$u = 0, \cdots, M-1; \quad v = 0, \cdots, N-1 \tag{2.3}$$

$$f(j,k) = \sum_{u=0}^{u<M} \sum_{v=0}^{v<N} c(u,v) \exp\left(\frac{i2j\pi u}{M}\right) \exp\left(\frac{i2k\pi v}{N}\right)$$
$$j = 0, \cdots, M-1; \quad k = 0, \cdots, N-1 \tag{2.4}$$

The order of summation in the equations above is immaterial. Clearly an algorithm to compute a one-dimensional DFT is sufficient to compute a two dimensional DFT. This approach is known as the row-column algorithm[1].

Replacing the exponential-form basis used by DFT with cosine-form basis, we have one dimensional DCT (Discrete Cosine Transform) and inverse DCT

$$c(u) = a(u) \sum_{k=0}^{k<N} f(k) \cos\left(\frac{(2k+1)\pi u}{2N}\right) \quad u = 0, 1, \cdots, N-1 \tag{2.5}$$

and

$$f(k) = \sum_{u=0}^{u<N} a(u)c(u) \cos\left(\frac{(2k+1)\pi u}{2N}\right) \quad k = 0, 1, \cdots, N-1 \tag{2.6}$$

where $a(u)$ is defined as

$$a(u) = \begin{cases} \sqrt{\frac{1}{N}} & \text{for } u = 0 \\ \sqrt{\frac{2}{N}} & \text{for } u \neq 0 \end{cases} \tag{2.7}$$

and two dimensional DCT and inverse DCT

$$c(u,v) = a(u)a(v)\sum_{j=0}^{j<M}\sum_{k=0}^{k<N} f(j,k)\cos(\frac{(2j+1)\pi u}{2M})\cos(\frac{(2k+1)\pi v}{2N}) \\ u \in \{0,1,\cdots,M-1\} \quad v \in \{0,1,\cdots,N-1\} \tag{2.8}$$

$$f(j,k) = \sum_{u=0}^{u<M}\sum_{v=0}^{v<N} a(u)a(v)c(u,v)\cos(\frac{(2j+1)\pi u}{2M})\cos(\frac{(2k+1)\pi v}{2N}) \\ j \in \{0,1,\cdots,M-1\} \quad k \in \{0,1,\cdots,N-1\} \tag{2.9}$$

DCT has some advantages for computation comparing with DFT. First, DFT is a complex transform while DCT is a real transform. As most signals in most of the applications are real signals, complex transform brings redundancy. In addition, studies have shown that DCT provides better energy compaction than DFT for most natural images[2]. Therefore, distortions at the high frequency components do not cause significant quality degradation. Furthermore, comparing with DFT, DCT-based image coding introduces less artifacts caused by the implicit periodicity of the signals under these transformation.

DCT is widely used in practice. A key factor that makes it popular is the fact that DCT can be computed efficiently using a fast Fourier transform (FFT) algorithm[3]. In addition, DCT better exploits inter-pixel redundancies and renders nearly optimal decorrelation for most natural images[4] so that all transform coefficients can be encoded independently without compromising coding efficiency.

2.1.2 DWT

Wavelets are a type of functions that satisfy certain mathematical requirements and are used in representing data or other functions. In our discussion, we are mostly interested in the wavelets that are constructed to have zero average value, finite energy and finite support so we can use their translation combination to represent the signal with excellent frequency and time (space) localization features.

Let $\phi(t)$ be such a wavelet function. By replacing t with t/a and translating the function by an amount b we have the translation form of $\phi(t)$ as

$$\phi_{a,b}(t) = \frac{1}{\sqrt{a}}\phi(\frac{t-b}{a}) \tag{2.10}$$

Specially if a and b are selected to let $a = a_0^{-m}$ and $b = kb_0a_0^{-m}$, where m and k are integers. Also let $a_0 = 2$, $b_0 = 1$, then we have a function set with discrete scaling and translating parameters:

$$\phi_{m,k}(t) = 2^{m/2}\phi(2^m t - k) \tag{2.11}$$

Fix the scaling parameter m, a large number of functions $f_m(t)$ can be represented as the linear combination of $\phi_{m,k}(t)$ as

$$f_m(t) = \sum_k c_{m,k}\phi_{m,k}(t) \tag{2.12}$$

where $c_{m,k}$ is a scalar. For all k, the functions $\phi_{m,k}(t)$ form a set $\{\phi_{m,k}(t)\}$. All the functions that can be represented as the linear combination of set $\{\phi_{m,k}(t)\}$ as shown in equation (2.12) form a space denoted as V_m and the space V_m spanned by the function set $\{\phi_{m,k}(t)\}$. Specially, when $m = 0$, we have a space V_0 and $\{\phi_{0,k}(t) = \phi(t-k)\}$.

Now if $\{V_m, m \in \mathcal{Z}\}$ is a multiresolution approximation of vector space $L^2(R)$ as defined in [5], one can always find a unique function $\phi(t)$ such that for any fixed m, the function set $\{\phi_{m,k}(t)\}$ that dilated and translated from $\phi(t)$ following (2.11) is an orthonomal basis of V_m. This function $\phi(t)$ is named as scaling function [5]. A function $f_m(t)$ that is represented as a superposition of scaling function set in (2.12) can be considered as an approximation of an $L^2(R)$ space function $f(t)$ in V_m.

As defined in (2.11), for any k, $\phi_{m,k}(t)$ is only non-zero over interval $[k/2^m, (k+1)/2^m)$. Comparing $f_m(t)$ with $f_{m-1}(t)$, the approximation of $f(t)$ in V_{m-1}, we need two times the number of coefficients (or the basis) to represent the signal of the same duration at scale level m. Thus we say that $f_m(t)$ is a higher resolution approximation of function $f(t)$ than $f_{m-1}(t)$, where m in fact indexes the resolution level and larger m represents higher resolution. Usually $f_m(t)$ is called a 2^m resolution approximation of $f(t)$. The

resolution level is often called scale level in our discussion. As an example, Figure 2.1 illustrates a function $f(t)$ and its two approximations at space V_0 and V_1, where $\phi(t)$ is Haar wavelets.

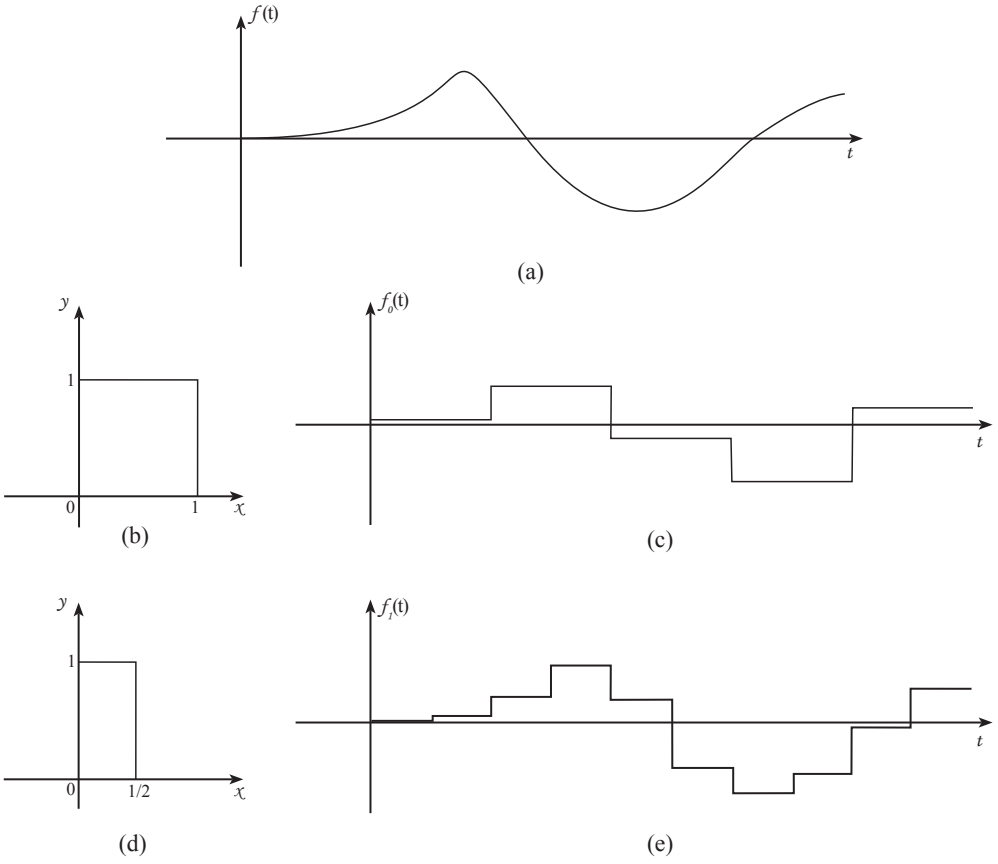

Figure 2.1: $f(t)$ and its approximation at scale level 0 and 1. (a) $f(t)$. (b) Haar scaling function $\phi(t)$. (c) $f_0(t)$. (d) $\phi(2t)$. (e) $f_1(t)$

One important property of the scaling function and its dilated function sets is that any $f_m(t)$ that can be represented by $\{\phi_{m,k}(t)\}$ can also be represented by the dilated function set $\{\phi_{m+1,k}(t)\}$ on scale level $m+1$. Therefore any base function $\phi_{m,k}(t)$ can also be represented by the set $\{\phi_{m+1,k}(t)\}$ as

$$\phi_{m,k}(t) = \sqrt{2}\sum_{l} h_{l-2k}\phi_{m+1,l}(t) \tag{2.13}$$

where $\phi_{m+1,l}(t) = 2^{(m+1)/2}\phi(2^{(m+1)}t - l)$ and $h_k = \int \phi_{m,k}(t)\phi(2^{(m+1)}t - k)^* dt$. Equation (2.13) is called the multi-resolution analysis (MRA). It illustrates the relation between the scaling functions of the adjacent scale levels.

For any $m \in \mathcal{Z}$, since $f_m(t)$ and $f_{m+1}(t)$ are two approximations of $f(t)$ at different scale levels, they are different. Let $w_m(t) = f_{m+1}(t) - f_m(t)$ denote the difference which describes the information lost when going from scale level $m+1$ to m. Represent $w_m(t)$ by the linear combination of another set of wavelet functions $\{\psi_{m,k}(t)\}$ which are dilated and translated from a wavelet $\psi(t)$. where

$$\psi_{m,k}(t) = 2^{m/2}\psi(2^m t - k) \;\; m, k \in \mathcal{Z} \tag{2.14}$$

Similarly $\{\psi_{m,k}(t)\}$ spans a space W_m. In particular, $\{\psi_{m,k}(t)\}$ is carefully constructed so that the space W_m is precisely the orthogonal complement of V_m in V_{m+1}, so $w_m(t)$ and $f_m(t)$ form a decomposition of $f_{m+1}(t)$. Which can be written as:

$$f_{m+1}(t) = f_m(t) + w_m(t) = \sum_k c_{m,k}\phi_{m,k}(t) + \sum_k w_{m,k}\psi_{m,k}(t) \tag{2.15}$$

Use $\bigoplus$ to denote the orthogonality of space V_m and W_m, the relation between V_{m+1}, V_m and W_m is usually represented as

$$V_{m+1} = V_m \bigoplus W_m \tag{2.16}$$

Equation (2.15) can be applied to all possible integer m therefore signal $f_m(t)$ in space V_m can be further decomposed into a signal $f_{m-1}(t)$ in space V_{m-1} and a signal $w_{m-1}(t)$ in W_{m-1} using the same procedure. If we start the decompose procedure from V_M with resolution level M, after D times of decompositions, the signal will be decomposed to a coarse function $f_{M-D}(t)$ in space V_{M-D} and a set of wavelet functions w_m in W_m where $m = W_{M-1}, W_{M-2}, \cdots, W_{M-D}$. i.e.,

$$f_M(t) = \sum_k c_{M-D,k}\phi_{M-D,k}(t) + \sum_{m=M-1}^{M-D}\sum_k w_{m,k}\psi_{m,k}(t) \tag{2.17}$$

In another word, the space V_M is decomposed to:

$$V_M = V_{M-D} \bigoplus W_{M-2} \bigoplus \cdots \bigoplus W_{M-D} \tag{2.18}$$

Equation (2.17) gives the fundamental format of 1-D wavelet transformation. From (2.18) we can see that space V_m and W_m are both the subspace of V_{m+1} since any function in V_m or W_m is also in space V_{m+1}. In particular, function set $\{\psi_{m,k}(t)\}$ is also able to be represented by the linear combination of scaling function $\phi_{m+1,k}(t)$ as

$$\psi_{m,k}(t) = \sqrt{2}\sum_l (-1)^l g_{l-2k}\phi_{m+1,l}(t) \tag{2.19}$$

where $\phi_{m+1,l}(t) = 2^{(m+1)/2}\phi(2^{(m+1)}t - l)$ and $g_k = h^*_{1-k}$. Equation (2.19) is the counterpart of the MRA in the space V_{m+1}. Together with Equation (2.13) they are the most important equations in wavelet analysis because they not only illustrate the relation of the basis in adjacent resolution levels in the mathematical format, but also provide a coefficients pair $\{h_k\}$ and $\{g_k\}$ that build up the link between the wavelet transformation coefficients at different scale levels.

From equation (2.13) and (2.19), it is derived in [5] that the relation between $c_{m,k}$, $w_{m,k}$ and $c_{m+1,k}$ will be

$$c_{m,k} = \sum_l h^*_{l-2k} c_{m+1,l} \tag{2.20}$$

and

$$w_{m,k} = \sum_l (-1)^l g^*_{l-2k} c_{m+1,l} \tag{2.21}$$

Implementation of wavelet transform on digital image is based on the relation described by equations (2.20) and (2.21) above. The spatial domain image samples are immediately the coefficients $\{c_{m+1,l}\}$ to represent the image as a linear combination of

$\{\psi_{m+1,l}(t)\}$. The procedure of wavelet transform as described in the equations is a down sampling decomposition from $f_{m+1}(t)$ to $f_m(t)$ and $w_m(t)$. So the transformation is from discrete input to discrete output and is naturally a Discrete Wavelet Transform (DWT). In the equations, both h_k and g_k are finite length filters to generate the down sampled coefficients $\{c_{m,k}\}$ and $\{w_{m,k}\}$. That is, if $\{c_{m+1,l}\}$ has length $2k$, both $\{c_{m,k}\}$ and $\{w_{m,k}\}$ will be with length k. Representation $f_{m+1}(t)$ on scale level $m+1$ passes h (resp, g) filter to obtain the lower resolution representation $f_m(t)$ (resp, $w_m(t)$). The output coefficients $\{c_{m,k}\}$ are the coarser representation of the image on resolution level m while the output coefficients $\{w_{m,k}\}$ are the lost details of the image from resolution level $m+1$ to m.

Similar to DCT transformation, a DWT can be performed by using two dimensional filters or applying one dimensional filter on image rows and columns separately. The second approach is the most popular one and is discussed in many previous work such as [6]. Applying wavelet filter on each row and each column once, one image with size M by N will be down sampled and decomposed to four sub-images each with size $M/2 \times N/2$. The fact that DWT is using two filters to decompose image is very close to the idea in subband coding [6] and the output components are localized at a certain frequency range on both row and column, so usually we also call the sub-image a subband of the original image. Normally we use letter L (resp, letter H) to refer to the subband that outputs from the low (resp. high) pass filter along one direction. Applying low pass filter on both rows and columns results in the LL subband, from previous analysis we know this is actually an approximate image that is only half of the resolution of the original image. Other than that, LH (resp, HL, HH) refers to the output subband that passed the low (resp. high, high) pass filter on the first direction and passed the high (resp. low, high) pass filter on the second direction. Since the number of coefficients generated from DWT is the same as the input image size and each subband contains a quarter of the number of the coefficients in the original image, usually a same $M \times N$ size matrix is used to represent the coefficients and the subbands relation after decomposition, as showed in Figure 2.2. If the filter is first applied to each column of the image, HL subband will be placed on the right above corner of the matrix, otherwise, it will be on the left bottom corner of the matrix.

Every subband can be further decomposed into four lower resolution subbands. The

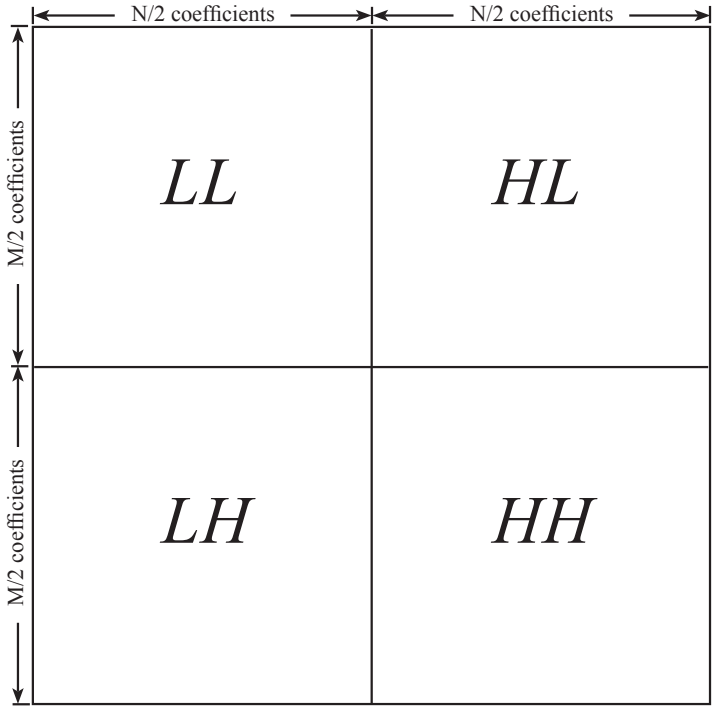

Figure 2.2: Subbands of an $M \times N$ image after one level discrete wavelet decomposition

decomposition procedure can be done many times until all desired subbands are created. Although each of the subband can be further decomposed independently, the most popular way to decompose the image is to only perform the decomposition on LL band for each iteration. This is called a hierarchical decomposition. Figure (2.3) displays a three-level decomposition. In this example, after three times of decomposition, there is only one LL band among the final decomposed subbands. We use the same terminology "scale level" as in previous section, subband LL is then said to be on the scale level 0. Correspondingly subbands HL, LH and HH are created on each scale level. All subband with the same type (HL, LH or HH) on different scale levels form a subband class. They retain the detail information of the image on each scale level. Size of each subband reflects its resolution in terms of coefficients. Apparently larger size subband has higher resolution.

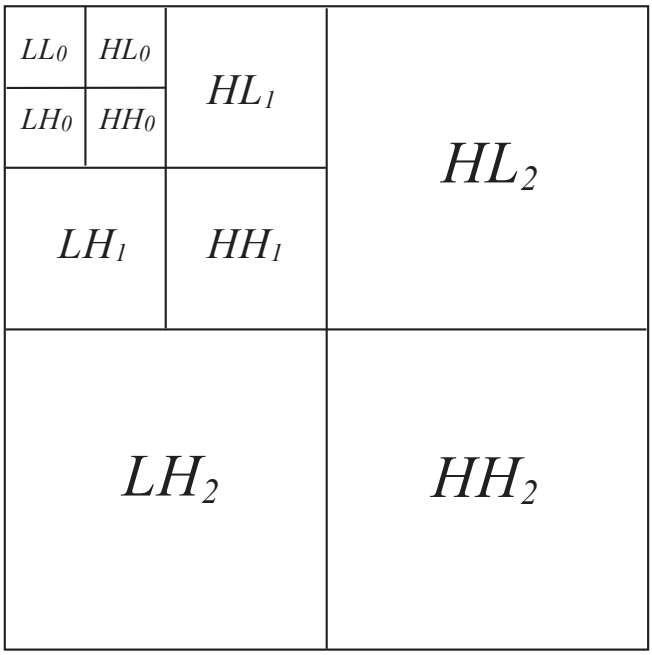

Figure 2.3: Three-level discrete wavelet decomposition in hierarchical structure

Every subband represents the image at a particular scale level. This means every coefficient on a subband corresponds a spatial area. The size of the area that is covered by one coefficient is different on different scale levels. Approximately, from the DWT setting the size of the area that is covered by a coefficient on the scale level m is 2×2 times than the size of the area covered by a coefficient on the next finer scale level $m + 1$. In another word, the spatial area corresponded by a coefficient in scale level m is equivalent to the spatial area corresponded by four coefficients in scale level $m + 1$. If we use the same spatial size to illustrate all subbands, the coefficient density in different subbands is shown as Figure 2.4.

2.2 Image Compression

The objective of image compression is to exploit the redundancies in image data and remove them as much as possible so that on average fewer number of bits are required to

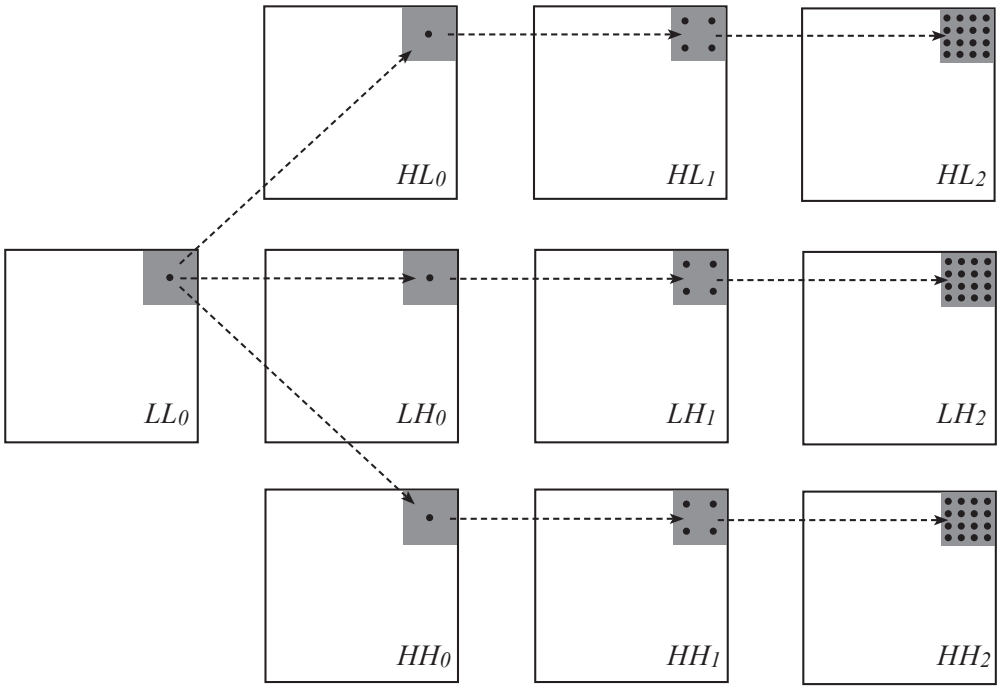

Figure 2.4: Coefficient density of different subbands and spatial relation

represent the image with acceptable fidelity. Compression rate, which is the ratio between uncompressed and compressed image data size in terms of bits, is usually used to indicate the degree of compression. Higher compression rate corresponds to a smaller number of bits with a more aggressive compression procedure to represent a given uncompressed image.

Image compression techniques are broadly classified into two categories: lossless compression and lossy compression. Lossless compression algorithms guarantee a perfect reconstruction of each pixel but the compression rate is highly limited. On the other hand, lossy techniques provide much higher compression rates by allowing distortion in the reconstructed image. Many lossy compression methods have been developed in the past decades. Among them, transform coding gained a great deal of attention. From JPEG to latest JPEG2000 standard, most of the state-of-the-art compression algorithms make use of image transformation techniques.

The purpose of image transformation, is to decompose image information formatted in spatial domain into multiple components that are more independent of each other in another domain and expose the weights of each component for compression. After transformation, variant compression configurations such as compression rates or discretization steps can be applied to different components to reflect the various importance level of the components. Normally a transformation-based image compression procedure includes three major steps:

1. Image transformation to and from different domain or representations;
2. Lossy compressing the coefficients to keep only the essential information;
3. Lossless entropy encoding to further remove the redundancies.

As described in previous section, there are a few different types of transformation. Selecting transformation type is important but not as critical on the efficiency of image compression comparing with the role of lossy compression and entropy coding [7]. However, a well selected transformation scheme is precondition of efficient information redundancy removal and compression. DCT is a very popular transformation method which is widely used in image compression such as the JPEG standard. Wavelet transformation, because of its excellent spatial and frequency localization characteristics, has also gained a lot of attention in the image compression area and has demonstrated great performance on image compression, see example [8] and [9].

2.2.1 JPEG

JPEG is one of the most popular lossy image compression standards. The basic implementation of JPEG is baseline JPEG. The value of each pixel is first shifted to a zero centered magnitude value, then the whole image is divided into blocks of size 8 by 8. After that, DCT is performed for each sample block. From previous analysis and the equation (2.8) every image block will be transformed to a new 8 by 8 frequency domain coefficient matrix in which each coefficient is the amplitude of corresponding frequency component. The second step, JPEG standard provides a pre-defined "quantization table", which is a matrix having the same size as a coefficient block. Each element of the

quantization table, denote as $\mathcal{Q}_{ij}$, where i and j denote the row and column indices of the matrix, indicates the quantization step size that is used to quantize the coefficient at the location (i, j) on each 8-by-8 coefficient block. Normally a smaller quantization step size applies to the coefficients that are closer to left upper corner of the block. It is because the coefficients on the left upper corner of a block represent magnitude of low frequency components, which are perceptually more important than high frequency components. The quantization of each coefficient can be described as

$$l_{ij} = \lfloor \frac{c_{ij}}{\mathcal{Q}_{ij}} + 0.5 \rfloor \tag{2.22}$$

where c_{ij} is the coefficient at location (i, j) with sign. operator "$\lfloor \cdot \rfloor$" is roundoff function to map the value $\frac{c_{ij}}{\mathcal{Q}_{ij}} + 0.5$ to the largest integer that is no greater than $\frac{c_{ij}}{\mathcal{Q}_{ij}} + 0.5$. We refer to l_{ij} as a label to distinguish the quantization outputs from the coefficients. Apparently l_{ij} is integer for any (i, j). Comparing with c_{ij}, clearly l_{ij} needs much fewer bits to represent. In Figure 2.5 an example is provided to show the coefficient values before quantization and the labels after quantization using the given quantization table. It is important to notice that after quantization most of the label values becomes zero and the zero values in general cover the high frequency components. If the "zigzag" scan order suggested by [10] is used to serialize the matrix, with very high probability a long sequence of zeros will appear at the end of the scan. An "End of Block" symbol is added immediately after the last non-zero value in lieu of the long run zeros. Besides, run length encoding (RLE) and differential coding [10] are also used so that the block is substantially compressed. In the end entropy coding such as Huffman codes [10] or its equivalent will be used to further compress the serialized symbol sequence.

JPEG has a very obvious "block effect", i.e., very sharp difference at the edge of the spatial domain encoding blocks. It is particularly obvious at very high compression rate when each block can only be represented by very few bits. Normally that means higher quantization step sizes are used for high frequency components thus very little detail information of each block can be retained in the quantized symbol sequence even if these components have relatively higher magnitude than that of the lower frequency components of the other blocks. The artifacts on the border of each block become very

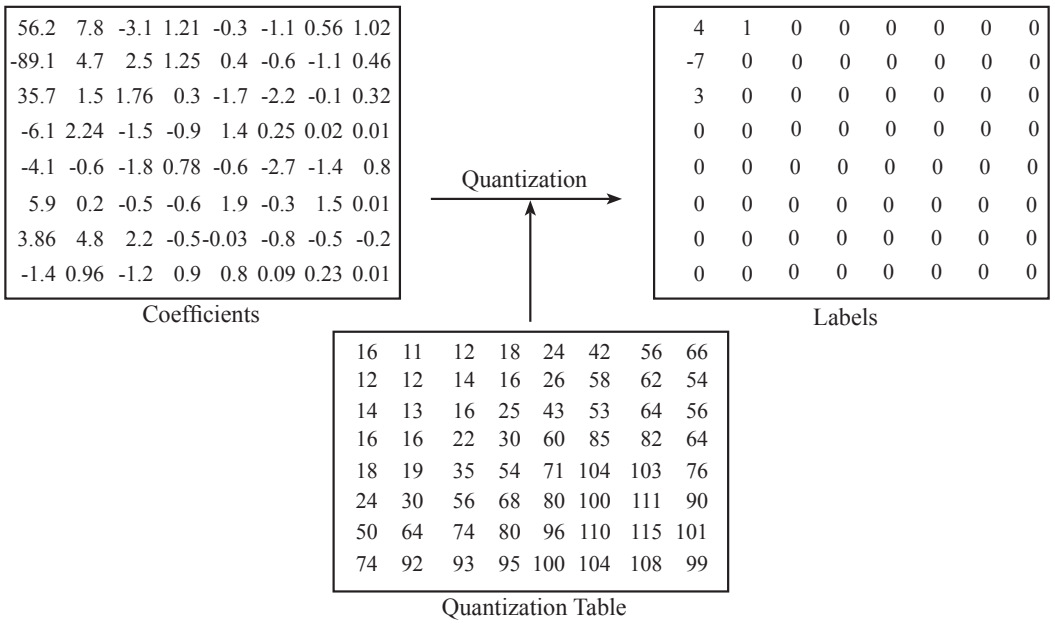

Figure 2.5: JPEG Quantization Example

large due to the missing of the high frequency information. Besides, JPEG transmits one spatial block after another. Terminating the transmission of a JPEG compressed image in the middle will result in a complete loss of information for some blocks. Reconstruction of these blocks then becomes impossible, which significantly increase the distortion.

2.2.2 Orientation Tree and Coding Layer on Wavelet Domain

From previous chapter, we see that after DWT transformation, an image is decomposed into a set of subbands each capturing the full spatial area of the original image on a particular scale level. No subband-level global statistics in any form obviously exists inside one subband. This is very different from a conventional subband coding system, in which coefficients in the same subband are usually considered having similar statistical characteristics and therefore may apply the same quantization steps to approximate the coefficients. In wavelet domain, regardless of the subband residing in, every coefficient is considered statistically independent. However, because of the spatial and scale level

relationship between the coefficients in different subbands, a new orientation of sorting the coefficients in the order of coding importance is discovered. It provides the most important contribution for the coding efficiency in wavelet domain and is common for all wavelet-based algorithms. Because of its importance, before introducing the popular wavelet-based image compression algorithms, we first explain two key notions of the orientation which are built on top of the structure of decomposed wavelet domain coefficients matrix: Orientation Tree and Coding Layer. They will be helpful for later description.

The coefficients in the LL subband provide a spatial partitions of the image, where each coefficient represents one portion of the spatial area as shown in Figure 2.4. All the coefficients in other subbands from all scale levels corresponding to the same spatial location can be organized as a tree that is rooted at the coefficient in LL subband. This we call the "**Orientation Tree**". The number of orientation trees depends on the partitioning of the coefficients in the LL subband. Except the root coefficients in the LL subband, the other coefficients belonging to an orientation tree are the descendants of a root. Coefficients in a coarser scale subband are the parents of the coefficients in the next finer scale subbands. The depth of the orientation tree equals total number of the scale levels plus one. For an image decomposed into D scale levels, each orientation tree has depth $D+1$. The three coefficients in each of these three subband classes (HL, LH and HH as introduced earlier) other than the LL subband on scale level 0 are the offsprings of the root coefficient in LL subband. Scale level $d, 1 \leq d \leq D-1$, contains four offsprings of the coefficient in the same subband class but on scale level $d-1$. One example of a four-level orientation tree is displayed in Figure 2.6.

From the property of wavelet transform, it is observed that the importance of a coefficient is positive correlated with its amplitude[6][8]. After the wavelet transform, coefficients in a lower scale level subband usually have much higher amplitudes comparing with those in a higher scale subband. That is, in a wavelet orientation tree, the coefficients in a scale level that is closer to the root usually carry more important information than the coefficients in a higher scale levels. Naturally a scan order from the root to the leaves among the different scale levels is usually chosen to reflect the order of importance. In another word, following the root-to-leaf order to scan an orientation tree, if a coefficient

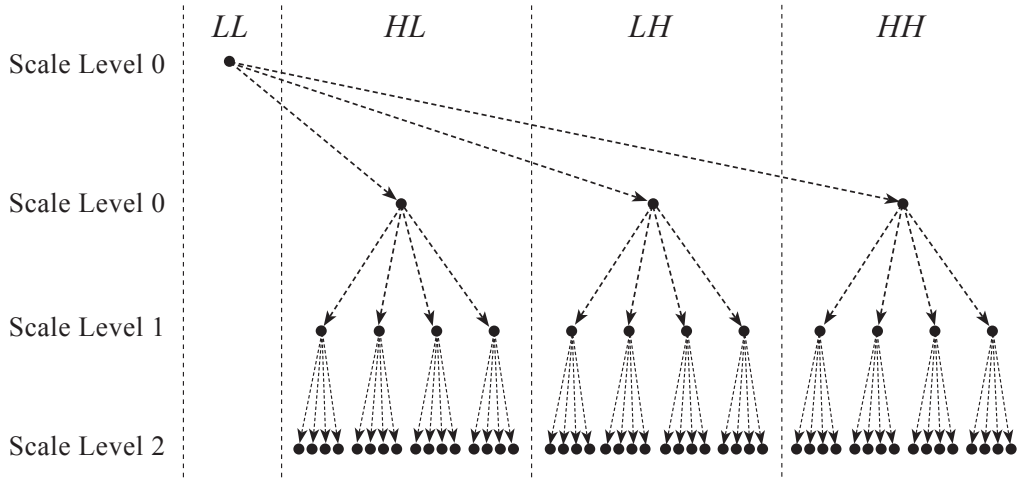

Figure 2.6: An orientation tree formed by the coefficients from three scale levels

has amplitude lower than a certain threshold, then we expect that with high probability its descendants will also have amplitude lower than this threshold. An appropriate coding order need to follow the same root-to-leaf order so that the coefficients with amplitude higher than the threshold and with scale level closer to the root of an orientation tree will be coded first.

Coding of wavelet coefficients is based on a simple comparison between a threshold and a coefficient. The comparison results in a binary value. Ignoring the sign of the coefficient, we say that a coefficient is "significant" with respect to a threshold T and associate with it a binary value (usually a "1") to indicate its significance if this coefficient has higher absolute value than the threshold. Otherwise, the coefficient is said to be insignificant and we assign to it the other binary value (usually a "0"). We call this comparison a "significance test". After comparing each coefficient in the wavelet coefficient matrix with the threshold T using the significance test, a binary "significance map" is created. Figure 2.8 shows an example. First let us ignore the sign of each coefficient and focus on the amplitude. After comparison, a few coefficients in the left up corner of the matrix are marked with "1" in the significant "map" indicating that they have higher amplitude comparing with threshold 32. If we add the sign of each significant coefficient back and convert the matrix into a symbol string following the orientation tree's scale order from

low to high and on each scale level following the same scan order as shown in the figure, it will end up with the binary string at the bottom of the Figure 2.8.

The procedure of significance test divides the coefficients into two sets: the significant set and insignificant set. Apparently a significant coefficient with respect to T has amplitude between either positive or negative T and $2T$. Then $1.5 \cdot T$ is usually chosen as the best guess of a coefficient given that the coefficient is significant with respect to T. And 0 will always be selected as the best guess of the insignificant coefficients since a coefficient is insignificant if its amplitude is between $-T$ and T and we do not know the sign of the insignificant coefficient. So the threshold T in fact serves as a three-level quantizer to map the coefficient to one of the three intervals. The three-level quantizer is shown in Figure 2.7. The string at the bottom of Figure 2.8 is in fact a quantized version of the coefficients with respect to the threshold T and the recovered matrix is shown in Figure 2.9.

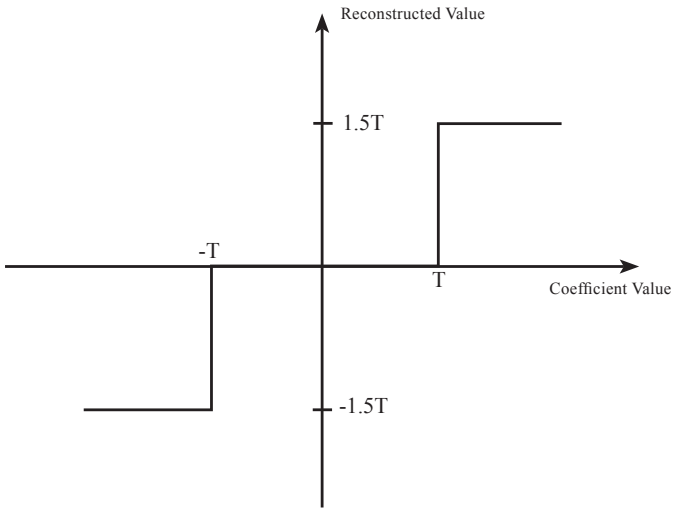

Figure 2.7: The equivalent three level quantizer for significance coding

For a coefficient in insignificant set, another significance test can be performed with a lower threshold. The significance test for the set of insignificant coefficients is usually called a "dominant" procedure. On the other hand, the coefficients in the significant set,

can be any value in the quantization interval between threshold T and $2T$. Assigning an extra bit for the significant coefficient will help to narrow down its location within the quantization interval to a $T/2$ range either between T and $1.5T$ or between $1.5T$ and $2T$. This is usually called a "refinement" procedure of a significant coefficient. Later we will see that continually narrowing down the range of the interval up to a certain precision by sending extra bits will produce a binary approximation of the significant coefficient. The significance test for each insignificant coefficient and the refinement process for each significant coefficient together complete a coding pass for the entire wavelet coefficient matrix. We call the coding pass a **Coding Layer**. Each coding layer is associated with a significance test threshold T. Normally the initial threshold T_0 will be selected so that $\|c\| < 2 \cdot T_0$ where c is the amplitude of any coefficient in the matrix. A series of thresholds $T_0,\ T_1,\ \ldots\ T_{L-1}$ are selected to associate with L coding layers. Usually they satisfy the relation:

$$T_i = T_{i-1}/2 \ i = 1, 2, \cdots, L-1 \tag{2.23}$$

Although the ratio between two adjacent thresholds could be arbitrary, choosing the ratio 1/2 provides great convenience in mapping a coefficient to a binary sequence.

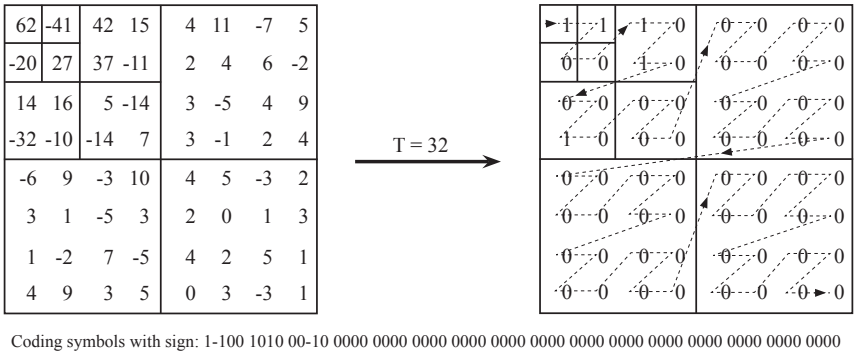

Figure 2.8: An example of significance coding and output symbol string

We use an example to explain the coding process for a particular coefficient across multiple coding layers. Assume this coefficient has amplitude 39.2 and the target is to proceed significance test for the coefficient in four coding layers with the thresholds {64, 32, 16, 8}. In the first layer value 0 will be assigned since the absolute value 39.2 is

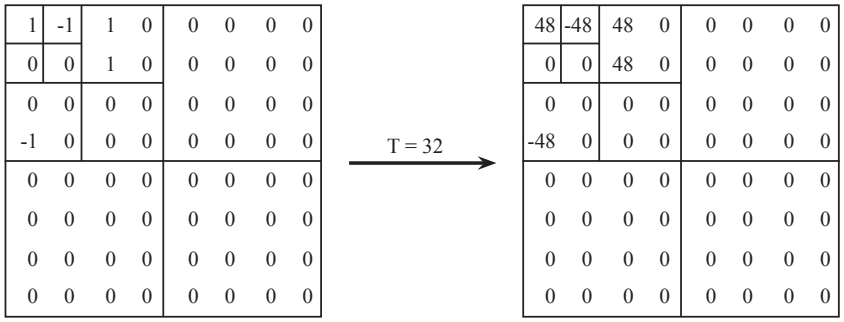

Figure 2.9: Reconstructed coefficient after the first coding pass

lower than threshold 64. After that, the threshold is reduced to 32, the coefficient is now significant, so 1 will be given to indicate its significance on the second coding layer. This value will be reconstructed as $1.5 \times 32 = 48$. On the third coding layer, the threshold is further reduced to 16. Since the coefficient has already been significant in previous layer, a bit will be assigned to indicate its finer location in the quantization interval. This in turn refines the reconstructed value of the coefficient. Since 39.2 is closer to the previous threshold 32 than 48, a "0" will be given to indicate that the coefficient is in the lower half interval of $[32, 48]$, then the reconstructed value will take the center value between 32 and the new boundary 40, which is then 36. Moving to the fourth layer,the threshold is reduced to 8, 39.2 is closer to the value 40 rather than 32, so "1" will be given for this refinement step. The reconstructed value then will be $36 + 2 = 38$. So the generated symbols for coefficient after four coding layer will be $\{0, 1, 0, 1\}$.

Orientation tree and coding layers are two most commonly used notions in the popular wavelet-domain compression algorithms. These compression algorithms essentially find a way to represent each coding layer along the orientation tree with high efficiency and appropriate importance order. However, the way of defining the coefficients and the relationship between the coefficients are algorithm dependent and impact the performance of the algorithm. We will introduce coefficient types and their relationships together with the concrete algorithms such as EZW and SPIHT.

2.2.3 Embedded Zerotree Wavelet (EZW)

Embedded Zerotree Wavelet (EZW) algorithm proposed in [8] uses four symbols to represent significance test results for one or a set of coefficients on a coding layer. Observe a coefficient on a coding layer with threshold T, if the coefficient and all its descendants have amplitude less than T, this coefficient is called a zero tree root ("zr"). From the significance test point of view, all coefficients rooted at a "zr" coefficient have never been significant in previous coding layers. The symbol "zr" is the most important symbol in EZW algorithm and it contributes to a large portion of the compression efficiency by representing the large set of insignificant coefficients on a coding layer using a single symbol. This comes from the fact that in most coding layers, the insignificant coefficients occupy the major portion of the matrix.

While reducing the bits allocated for insignificant coefficients, EZW allocates more bits on significant coefficients and describing exceptions to ensure any important information to be considered and captured at an earlier stage. In addition to the zero tree root, there are three other symbols: "positive significant" ("ps" in short) is used to describe an individual coefficient that is significant with positive sign while "negative significant" ("ns") is used to describe an individual coefficient that is significant with negative sign, "isolated zero" ("iz") is used to describe the situation where a coefficient in the orientation tree is insignificant itself but has at least one descendant that is significant in the current coding layer. If any of the symbols above is recognized, the algorithm will continue checking its descendants until all the significant coefficients are identified. It is worthy to mention that in order to use the smallest possible alphabets to represent the significant situations, both positive significant and negative significant symbols take the sign information but do not provide the information as to whether their descendants are all insignificant. So the significance test will continue for their offsprings in next scale level. On the other hand, symbols "ps" and "ns" carry "becomes significant" information which more or less are redundant. As for each coefficient, changing status from insignificant to significant occurs only once in the whole coding procedure. Coefficient will be added to a significant coefficient list to conduct refinement coding (Which will be explained later) after it becomes significant. Coding the sign of the coefficient is sufficient to indicate the location where the status change happens. Redundancies contained in

the symbols limit the performance of the algorithm.

Using these new EZW symbols to represent the coded sequence of significance map in Figure 2.8, we have: $ps, ns, iz, zr, ps, zr, ps, zr, zr, zr, ps, zr$. It is much more efficient and simpler than the previous representation.

Following the significance test process (dominant process) is the refinement process or subordinate process for all the coefficients that are added to the significant list in all previous coding layer. As mentioned before, an additional bit will be used to refine the magnitude of each coefficient to improve the precision. Naturally the order that coefficients enter the significant list will also be the order of the bits that are used to refine the coefficients. This avoids extra order information being added to the coding sequence and therefore saves bit budget. Meanwhile, once a coefficient has been added to the significant list, it will be marked as magnitude zero in the original coefficient matrix to ensure that it will not be identified as significant coefficient again during the significance test in next coding layers.

In EZW, four factors determine the importance of a bit being used in the compression algorithm: the magnitude, the wavelet scale level, the spatial location and the numeric precision of the coefficients. Magnitude is a determination factor of importance since coefficients that are becoming significant have greater magnitude than those insignificant coefficients and they will be coded first. Scale level is a determination factor since the coefficient scan follows the orientation tree direction, in another word, it follows the order from a coarser scale level to a finer scale level on each coding layer. So a bit from a coarser scale level will be coded first. Spatial location is a determination factor since the dominant and refinement scanning both follow a simple spatial order. At last, numeric precision is a determination factor of importance since the uncertainty intervals for the amplitude of all significant coefficients are refined to the same precision before the uncertainty interval for any coefficient is refined further. Therefore coarser numeric precision is always coded before the next finer numeric precision. These four factors in turn guarantee that any bit used earlier will be more important that the bit used later.

In EZW apparently the relationship of the coefficients are coding layer dependent. It abstracts the significance trend of the coefficients from coarse scale to finer scale without detailed magnitude comparison. It groups the coefficients by identifying their

significance and transmit the significant map in such an order that the decoder could duplicate without needing much redundant information.

2.2.4 Set Partition in Hierarchical Trees (SPIHT)

The SPIHT (Set Partition in Hierarchical Trees) algorithm[9], proposed by Said and Pearlman, changed the way of significant test on the orientation tree. Recall that in the EZW algorithm, coding only stop at "zr" symbol because only this symbol represents the "significance" status of a coefficient set rooted at the "zr" symbol. The other three symbols do not provide any information of the significant status of the coefficient set rooted at these symbols, and the significance test has to be done for their descendants until all significant coefficients being identified. SPIHT reorganizes the information and changes the definition of the symbols so that the significance test of a coefficient set and significance test of an individual coefficient are decoupled, which makes the significance test more efficient.

To achieve this goal, SPIHT defines significance test for both individual coefficient and coefficient subset on the orientation tree, or Spatial Oriented Tree (SOT) as is named in SPIHT. Need to point out that the subset is not arbitrary. It has to be all the descendants of a coefficient $\{i, j\}$(the so-called $\mathcal{D}$-type subset in SPIHT) or all the descendants excluding the immediately offsprings of the coefficient $\{i, j\}$ (the so-called $\mathcal{L}$-type subset in SPIHT). Both tests on an individual coefficient and a subset are indexed by a coordinate $\{i, j\}$. For a significance test of a coefficient, the coordinate represents its location. But for a subset significance test, the coordinate refers to the location of the root of the subset. So the subset is also indexed by $\{i, j\}$ with a certain type $\mathcal{D}$ or $\mathcal{L}$.

A significance test of an individual coefficient (which we refer to CST in short) is to compare the coefficient magnitude with the significance threshold. If the magnitude is greater than the threshold, the coefficient is significant in this coding layer, otherwise it is insignificant. The subset significance test (in short we call it SST) also has a boolean outcome. A subset is "significant" if at least one of the coefficients in this subset is significant, otherwise it is "insignificant". Because a subset may be of either one of the two types, there are two different decisions for the significance test results on the subset.

- For a $\mathcal{D}$-type subset $\{i, j\}$, if the result of SST is "insignificant", none of the coefficient in the subset will be tested in current coding layer. Instead, the $\mathcal{D}$-type set will be tested again in the next coding layer. If the result is "significant", all four offsprings of the root of the $\mathcal{D}$-type subset $\{i, j\}$ will perform CST immediately. Insignificant offsprings will be added to the set of individual insignificant coefficients and to perform CST on next coding layer. Significant offsprings will be added to the set of significant coefficients and perform refinement procedure on next coding layer. These four offsprings will be removed from the original coefficient subset and the original subset, which was a $\mathcal{D}$-type, is now an $\mathcal{L}$-type subset. If the $\mathcal{L}$-type set is empty, no more SST is needed. Otherwise, SST must be done in the same coding layer for the new $\mathcal{L}$-type subset $\{i, j\}$ later.

- For an $\mathcal{L}$-type subset $\{i, j\}$, if the result of SST is "significant", the subset will be partitioned into four $\mathcal{D}$-type subsets each rooted at one of the four offsprings of $\{i, j\}$. SST will be performed later on the same coding layer for each of these four new subsets. Meanwhile no more SST will be performed on subset $\{i, j\}$. But if the result of SST for subset $\{i, j\}$ is "insignificant", the $\mathcal{L}$-type subset will be retained and SST will be performed for this subset on next coding layer.

The process of separating the $\mathcal{D}$-type subset to four individual coefficients and an $\mathcal{L}$-type subset, and partitioning an $\mathcal{L}$-type subset into four $\mathcal{D}$-type subsets form the SPIHT sorting pass. It is the core "set partitioning" procedure of the SPIHT algorithm. To help understanding the set partitioning rule in SPIHT, of particular importance is understanding the decoupling of coefficient significance test and subset significance test, an example of the sorting pass is showing in Figure 2.10. The gray area in the figure refers to the coefficient set in which CST or refinement process will be performed for each coefficient individually. We call it set $\mathcal{C}$. Area inside the dash line are the roots of the subsets for which SST either with type $\mathcal{D}$ or with type $\mathcal{L}$ will be performed. This area is denoted as set $\mathcal{B}$. Before coding starts, all four coefficients belonging to the lowest scale level are initially in the set $\mathcal{C}$. Notice that the coefficient matrix in Figure 2.10 only has two scale levels. The tree structure of SPIHT, which stipulates that in LL band, the coefficient located at the top left corner of every 2 coefficient block has no descendant,

is slightly different from the EZW tree. In this example, number 62 has no descendant, therefore initially the other three coefficients are placed in the set $\mathcal{B}$ as the roots of $\mathcal{D}$-type subsets.

Coefficient in set $\mathcal{C}$ are sorted into two lists: insignificant coefficient list (LIP) and significant coefficient list (LSP). Initially in the first coding layer all the coefficients in set $\mathcal{C}$ are considered insignificant and are placed in the LIP (not illustrated in the figure). The sorting pass on each coding layer starts with CST for all insignificant coefficients in set $\mathcal{C}$. So the significance test will be performed for each of the coefficients in LIP and the result of CST is coded. Any of the coefficients that is significant will be moved to the LSP (not illustrated in the figure) and the sign of the coefficient will be coded immediately following its significance test result. Normally "1" is used to represent the positive sign and "0" to represent the negative sign. In the example in Figure 2.10, 62 and -41 are moved into significant list immediately after their significance test while -20 and 27 remain in the LIP. Bit sequence $\{1, 1, 1, 0, 0, 0\}$ is then generated to represent the results for all four coefficients. Later during the SST for set $\mathcal{B}$, more coefficients will be added to the set $\mathcal{C}$ and before they are moved into set $\mathcal{C}$, CST will be performed to decide which list the coefficient should be sorted to. In later coding layer the same CST will be performed for the coefficients in the LIP and the refinement process will be done for the LSP after the sorting pass.

After CST for set $\mathcal{C}$, the sorting pass will continue with SST for each subset rooted at set $\mathcal{B}$. Subsets in set $\mathcal{B}$ are sorted into a list named LIS. Here the coefficients -41, -20 and 27 are initially considered as the roots of insignificant subsets so they are in set $\mathcal{B}$ and LIS. To simplify the description, we call these three sets horizontal (H) set, vertical (V) set and diagonal (D) set respectively. Because in set H, 42 and 37 are significant with respect to threshold 32, H is significant. We use "1" to represent the significance of set H. All four offsprings of -41 will be added to set $\mathcal{C}$ after the SST for each of them, as described in last paragraph. As a result, 42 and 37 will be added to the LSP while 15 and -11 will be added to LIP. A binary sequence $\{1, 1, 0, 1, 1, 0\}$ is then generated after the CST on these offsprings. Similarly, set V is significant and the four offsprings of -20 will also be added to the set $\mathcal{C}$ where -32 will be in LSP and $14, 16$ and -10 will be in LIP. Bits $\{1, 0, 0, 1, 0, 0\}$ are now coded for set V. Set D is insignificant because

none of the descendants of 27 is significant, so none of the offsprings of 27 will be added to the set $\mathcal{C}$. Only the SST result 0 for set D will be coded. Sorting pass continues to test the subsets H and V since they are now $\mathcal{L}$-type sets after the $\mathcal{D}$-type partitioning. Because none of the coefficients is significant, both H and V are insignificant and are kept in set $\mathcal{B}$ but with type $\mathcal{L}$. Bits $\{0, 0\}$ are used to indicate the insignificance of these two $\mathcal{L}$-type sets at the end of the sorting pass. The coded bit sequence is illustrated at the bottom of Figure 2.10. Note that the roots of these three subsets are not involved in the significance test. Apparently the ranges of both set $\mathcal{C}$ and $\mathcal{B}$ have changed after each sorting pass. More and more coefficients will be added to set $\mathcal{C}$, and size of set $\mathcal{B}$ will fluctuate with the changing of coding layer. However, benefiting from the definition of the SST, the partitioning of the insignificant coefficient subset will be in a slower pace and does not depend on the significance of the root. Decoupling of SST and CST brings tremendous improvement in coding efficiency comparing with EZW.

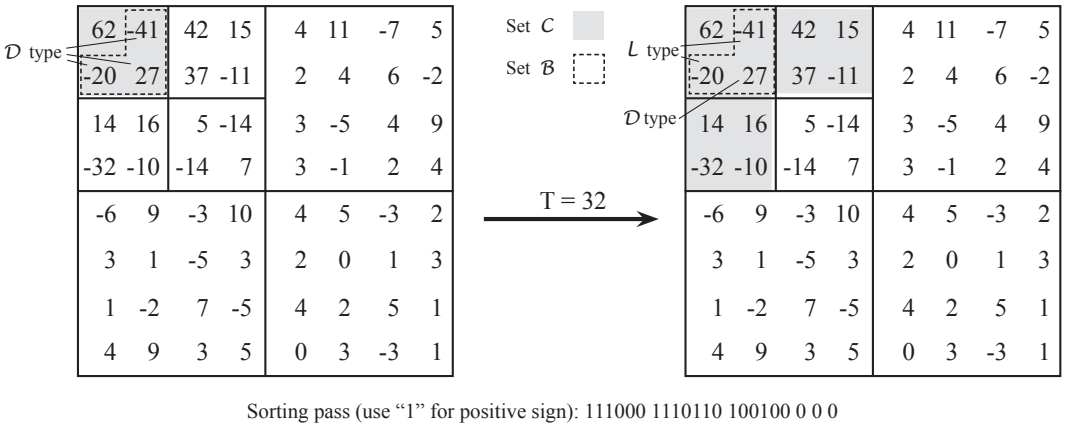

Figure 2.10: OneSPIHT sorting pass with respect to threshold $T = 32$

The refinement pass is similar to that of EZW. In SPIHT, the selection of the thresholds is identical to that of EZW. So for any coding layer k,$k = 0, 1, \cdots$, the threshold T will be 2^{n-k} where n is an integer. Correspondingly any coefficient c_{ij} without sign could be approximately represented by a binary stream. If a coefficient is significant in coding layer k, the coefficient has magnitude $2^{n-k} \leq \|c_{ij}\| \leq 2^{n-k+1}$, and the significance test result in coding layer k therefore represents the most significant bit of the binary

representation of c_{ij}. In the following coding layers, the coefficient stays in the significant set,the algorithm simply sends the $n-k-j$th bit ($j = 1, 2, \cdots$) of the binary representation of c_{ij} directly as the refinement output. Since the threshold does not change, the three level quantizer remains the same as EZW.

Three coefficient lists LIP, LSP and LIS are defined above. They are used to maintain the order of CST, SST and the refinement process. Algorithm in fact scans LIP and LIS in the sorting pass and LSP in the refinement pass on each coding layer. So the procedure can also be described as following:

1. CST for each insignificant coefficient in LIP. Move significant coefficients to LSP;
2. SST and set partition for each element in set LIS. Move significant coefficients to LSP and move insignificant coefficients that are removed from LIS to LIP;
3. Refinement of the significant coefficients in LSP.
4. Reduce quantization step size and repeat step 1 to 3.

Because the coding orders are naturally embedded in the three lists, and information of next step is contained in previous coded symbols, there is almost no extra information needed to be included in the header of the encoded file except the information of the first threshold and the number of scale levels.

We note that the following properties regarding importance order naturally exist:

- The coefficients with higher magnitude are coded earlier. Even if they are in a higher scale level, the algorithm always aggregate the coefficients of the same scale of magnitude and code them together because CST is before SST.
- In each coefficient set or list, if two coefficients have same magnitude scale, the coefficient in the lower scale level will always be treated before the one in the higher scale.
- The scale of magnitude change for coefficients in LIP is always higher than that for coefficients in the LSP in the same coding layer because the coefficients in LIP, once becomes significant, will have $1.5T$ reconstruction magnitude change while

the reconstruction magnitude change by refining the coefficients in LSP is $0.5T$. So higher reconstruction contribution coefficients always have higher priority as sorting pass is always done before refinement pass.

- The spatial zigzag scan order also applies to SPIHT. So for two coefficients in the same scale level and with the same magnitude scale, the spatial location decides their coding order.

SPIHT completely decouples the set significance test (SST) from the coefficient significance test (CST). So the coding symbols only need to deal with three binary decisions. They are:

- Coefficient is significant or insignificant.
- Subset is significant or insignificant.
- Sign of coefficient is positive or negative.

Therefore binary symbols are sufficient for SPIHT encoding. Even without further entropy coding, SPIHT is in fact already very efficient. Moreover, decoding can be terminated precisely at a given bit budget.

The SPIHT algorithm and its generated bit stream also have the following important properties:

- SPIHT is a progressive coding algorithm. Longer coding stream provides lower distortion level. Particularly, SPIHT coder produces an almost convex, non-increasing operational D-R function[11] evaluated by means of Mean Square Error (MSE). This means the most important information (in terms of reducing distortion) is transmitted first. From the relationship between MSE and PSNR, this also means that the rate-quality function is an almost concave, non-decreasing function.
- Because SPIHT is an embedded coder, it is possible to divide the entire output bit stream into multiple concatenated layers or segments. Every segment is made up of a few bits in the stream. Decoding procedure can be terminated at the end of any layer pre-defined.

- Also because SPIHT is an embedded coder, for any positive integers R_i and R_j with $R_j < R_i$, the output bit stream of length R_j is the prefix of the output bit stream of length R_i. Single encoded bit stream can be truncated at arbitrary bit rates to provide various reconstruction qualities.

2.3 Image Quality Metrics

To pursue higher compression rate, irreversible transform filter and lossy compression method are usually chosen and inevitably distortion is introduced in the reconstructed image. To evaluate the compression performance, the perceptual impact of the distortion is very important. For example, research indicates that not all frequencies in an image have the same importance perceptually[12]. Distortion of high frequency components might not be as critical as low frequency distortion. However, measure of perceptual quality is an elusive goal since the human response to the quality varies from time to time and from person to person. Generally objective measures via mathematical methods are more practical. So objective image quality metrics are highly emphasized and a lot of research and comparative studies have been done to model perceptual meaningful framework based on these metrics in the past decade such as [13], [14] and [15]. Among these proposed objective image quality metrics, Mean Square Error (MSE) and Peak Signal Noise Ratio (PSNR) are widely used.

Mean square error (MSE) refers to the average squared difference between the magnitudes of original image samples and those of the reconstructed image samples. Equation (2.24) shows how to calculate MSE for an $N \times N$ image where i and j are the coordinate of the samples on the image matrix, c_{ij} is the original sample value, and $\hat{c}_{ij}$ is the reconstructed sample value.

$$MSE = \frac{1}{N^2} \sum_{i=1}^{N} \sum_{j=1}^{N} (c_{ij} - \hat{c}_{ij})^2 \tag{2.24}$$

Peak Signal Noise Ratio (PSNR) is another popular distortion measure for images. It considers distortion as noise added to the original image and examines the strength of the

noise by comparing its power with the max possible signal power. It is calculated in the logarithmic scale since dynamic range of the noise power is usually large. Let c_{max} denote peak signal power, i.e., the max possible magnitude of a sample in the original image. For example, for a gray level image represented by 8-bits per sample, the number being used to represent the maximum possible lumina magnitude of a pixel is 255. The noise power is represented by mean square error (MSE) defined above. PSNR is formulated as equation (2.25) and higher PSNR implies better fidelity of reconstructed.

$$PSNR = 10\log_{10}(\frac{c_{max}^2}{MSE})\ dB \tag{2.25}$$

Only distortion level alone is not sufficient to evaluate the performance of a compression algorithm. Distortion (or image quality or fidelity) must be considered in conjunction with coding rate. Image fidelity is judged by the distortion level of the reconstructed image. In our description MSE and PSNR are both distortion measurements and they are practical objective measures. Coding rate often refers to total data in bits being used to represent the image. As the number of bits used to represent a coefficient (or a pixel) varies across coefficient (or pixels, resp.), bit per pixel (*bpp*), which is defined as the average number of bits used to represent a pixel, is also popular to represent the coding rate. Distortion measures on a series of coding rates form a rate-distortion function to evaluate the compression performance of a single image. Given the distortion level, the higher is the bit rate, the lower is the compression efficiency. In general we expect that with a lower coding rate, the distortion will be larger. In our work, the distortion level is represented by PSNR on a given coding rate. The rate-distortion function is therefore a rate-PSNR curve.

Distortion at a given bit rate highly depends on the statistics of the source signal, so usually it is hard to be modeled. However for non-stationary sources such as natural images, some encoding algorithms naturally display great characteristics. For example, [16] shows that rate-distortion function exhibits power-law behavior $D(R) \approx \alpha R^{-\gamma}$ at moderately high coding rates (0.03 to 3*bpp*) using wavelet based image compression algorithms such as SPIHT and JPEG2000 although value γ is still image dependent. Later we will show that these characteristics will be helpful to extend the compression algorithm to a

wider range of applications such as image archiving.

It is worthy to mention that for the SPIHT algorithm, rate-PSNR curve is non-decreasing and almost concave. In fact, for SPIHT encoded stream, the bits represent either the refinement of a coefficient, the significance of a coefficient or set, or the sign of the coefficient. So give a single bit, the magnitude change that impacts PSNR gain has three scenarios: 0 for the subset significance indicator or a coefficient indicator; 1.5 times of the threshold if this bit represents the "true" result of a significance test result (i.e., transmit the sign of the coefficient) or the magnitude of the threshold (refinement). Therefore the PSNR changing rate with respect to the bit rate is hard to be linear when all three type of symbols exist. But once all coefficients have been significant and only refinement pass takes effect, the distortion drops exponentially and rate-PSNR function is nearly linear as pointed out in [16].

2.4 Standard Optimization Problems

An optimization problem tries to maximize (or minimize) an objective function subject to certain constraints, where either the maximal (or minimal, resp.) value of the function or the maximizing (or minimizing, resp.) configuration of the function is of interest. The standard form of an optimization problem is: minimize $f(x)$ subjects to $g_i(x) \leq 0$, $i = 0, 1, \cdots, m$ and $h_j(x) = 0$, $j = 0, 1, \cdots, n$, for some finite numbers m and n. Maximization of an objective function can be transformed to the standard form above. Depending on the exact form of the objective function, the domain of the variables and the form of the constraints, optimization problems can be categorized into many types. We will highlight the following three families below, which are relevant to the thesis.

A linear program is an optimization problem with linear objective function and linear equality or inequality constraints. The feasible region of a linear program is a convex polyhedron[17] and the objective function is a real-valued affine function[17] defined on this polyhedron. A linear programming algorithm tries to find a point on the polyhedron where the objective function has the smallest value. The SIMPLEX algorithm[18], developed by George Dantzig, solves linear programs by constructing a feasible solution

at a vertex of the polyhedron and then walking along a path on the edges of the polyhedron to vertices with non-decreasing values of the objective function until a minimum is reached. In practice, the SIMPLEX algorithm is quite efficient and can guarantee finding the global optimum if certain precautions against cycling[17] are taken.

A linear integer program is a linear program in which some or all of the unknown variables are restricted to be integers. If only some of the unknown variables are required to be integers, it is also called mixed integer program. Contrast to the linear programs, which can be solved efficiently in the worst case, integer programs are in many practical situations (those with bounded variables) NP-hard. Sometimes the integer programs may be converted to a linear program by removing the integer constraints on the variables. Such a technique, known as "relaxation", sometimes preserves the solution of the integer program and allows the problem to be solved efficiently and optimally. Even in cases when relaxation changes the solution, it may still be considered as a technique to approach difficult integer programs as long as the solution is not "far" from the original integer program.

A quadratic program is an optimization problem with a quadratic objective function and linear constraints. The objective function has the form[19]:

$$f(\mathbf{x}) = \frac{1}{2}\mathbf{x}^T Q\mathbf{x} + \mathbf{c}^T\mathbf{x} \tag{2.26}$$

where $\mathbf{x}$ and $\mathbf{c}$ are length n column vectors and Q is an $n \times n$ matrix. If the matrix Q is a positive semi-definite matrix, then $f(\mathbf{x})$ is a convex function and in this case the quadratic program has a global minimizer if there exists some feasible vectors and if the function is bounded on the feasible region. If the matrix Q is positive definite and the problem has a feasible solution, then the global minimizer is unique. If Q is zero, the problem becomes a linear program.

There are many approaches for quadratic programming. Typical methods include the Trust-Region-Reflective[20], Active-Set[21] and Interior-Point[22] methods.

The basic idea of Trust-Region-Reflective is to approximate function f with a simpler function q, which reasonably reflects the behavior of function f in a low dimensional (such as two dimensional) subspace (trust region) around the point $\mathbf{x}$. Then determine

an approximate trial step $\mathbf{s}$ by resolving the trust region sub-problem. If $f(\mathbf{x}+\mathbf{s}) < f(\mathbf{x})$, then $\mathbf{x} = \mathbf{x}+\mathbf{s}$ and conduct another iteration. In a popular solution, the dominant work is to decide the subspace via certain preconditioned conjugate gradient process such as in [23],

The key of Active-Set method is to maintain an active set matrix, which is an estimation of the active constraints (the points that always on the constraint boundaries), at solution point. The matrix is updated every iteration and the search direction is guaranteed remaining on the active constraints. Active-Set solution requires an initial feasible point which if does not exist, can be found by resolving a linear problem. An estimation of the active set gives us a subset of inequalities to watch while searching the solution, which reduces the complexity of the search.

The Interior-Point method attempts to follow a path that is strictly inside the constraints. This method is targeting the convex optimization problem. It contains a presolve module to remove redundancies and to simplify the problem[24]. After that, the algorithm tries to find a point where the Karush-Kuhn-Tucker (KKT) conditions hold. Each iteration the algorithm predicts a step from the Newton-Raphson formula[23], then computes a corrector step. Searching procedure follows[25].

Chapter 3

Image Archiving Based On SPIHT

3.1 Motivation

Storage space and network bandwidth are limited resources. Archiving and transmitting large number of images such as geographical and medical image libraries are often constrained by these resource limitations. Such limitations can be represented by a given number of bits that allowed for representing images, which we refer to as the "bit budget". Under the bit budget constraints, maximizing the quality of the images is then particularly important.

Let B denote the bit budget for a set of K image to be archived or transmitted. The B bits need to be distributed to each image in the set with a certain rule of allocation. Let $k = 1, 2, ..., K$ index the images in the set. Let B_k indicate the number of bits assigned to image k. Obviously

$$B = \sum_{k=1}^{K} B_k \quad B_k \geq 0 \quad \forall\ k \tag{3.1}$$

Equation (3.1) forms an allocation. Under the constraint B, there are numerous allocation possibilities. Using PSNR as the quality metric, for a given compression algorithm, we can use function $q = D_k(r)$ to denote the rate-PSNR behavior for image k, where $D_k(\cdot)$ maps the rate r in terms of number of bits to an absolute PSNR level q. Specially $D_k(0)$ indicates the PSNR of an image k when no bit is allocated to it. Value of $D_k(0)$

is greater than zero and depends on the spatial domain average amplitude value of the image so it varies for different images. Value $D_k(B_k)$ represents the absolute PSNR of the reconstructed image k with B_k bits used. Let Q_k denote the PSNR gain achieved for image k with bit allocation $(B_1, B_2, \cdots, B_K)$. Apparently

$$Q_k = D_k(B_k) - D_k(0) \tag{3.2}$$

Let Q denote the overall image quality of this set of K images, given allocation (3.1), we have

$$\begin{aligned} Q &= \sum_{k=1}^{K} Q_k \\ &= \sum_{k=1}^{K} (D_k(B_k) - D_k(0)) \\ &= \sum_{k=1}^{K} D_k(B_k) - \sum_{k=1}^{K} D_k(0) \end{aligned} \tag{3.3}$$

From equation (3.3), when $K = 1$, bit allocation is unique, the rate-PSNR behavior of the algorithm is the only factor that can be adjusted for the image quality. Thus for single image compression, further algorithm optimization is the only way to improve compression efficiency. Once the compression algorithm is determined, the rate-PSNR behavior is known and the achievable performance for a single image is determined.

For an image set, other than the image dependent rate-PSNR function $D_k(\cdot)$, the allocation $(B_1, B_2, \cdots, B_K)$ is another factor that may impact the overall quality. However, most of conventional works focus on tuning the rate-distortion behavior by adjusting individual image compression procedure that adaptive to the image properties or adding another layer of compression procedure to further exploit the correlations between images from different aspects so that the rate-distortion behavior can be improved. The allocation $(B_1, B_2, \cdots, B_K)$ usually is the result of the improvement. The work of [26] developed an algorithm to perform a two step optimization procedure to achieve both

individual and overall optimization based on JPEG. Run time adaptive waveform transformation is used in [27] to match the image characteristics in order to reduce the overall distortion given a bit rate or bit budget. There are other image archiving approaches using predictive de-correlation method for an image set that consists of statistically similar images (See example [28]). These approaches focus on taking advantage of prior statistical correlation and dependencies of the images in the set. Some of them are only suitable for a statistically similar image set but not for general image set consisting of images with diverse characteristics while the others are hard to adapt a wide range of bit budgets. Application limitations are obvious.

Notice that if we compress image individually, each image in the set in fact is independent of the rests in the sense that increasing or decreasing the number of bits for an individual image will only change the quality of this particular image without touching others. Immediately the allocation $(B_1, B_2, \cdots, B_K)$ becomes a set of control variables for the overall image quality. In [29], efficiently transmitting the information of a vector source signal (not just image) with certain recovery quality criterion is studied. It also solved the bit allocation problem with certain convex assumption. Mean square error was used as the criterion in this work. Also it focused on correlated sources in general while the source coding class was not studied in detail. However, it pointed out a direction of resource sharing between independent components, which is attractive for the case of image archiving.

Equally allocating the bits to all the images in a set is not optimal. In Figure 3.1, two rate-PSNR functions are drawn from two 512 by 512 pixel gray level SPIHT encoded images: image 0 and image 1. Gradients of the curves, which show the PSNR growth rate, are not equal everywhere. In particular bit rate range, assigning bit to one image will provide higher gain than to the other one. For example, assume both images have been assigned the same number of bits up to the crossover point. Starting from that point, assigning any extra bit to image 1 will gain more PSNR gain than to image 0. Instead of equally assigning bits to both images, assign extra bits to image 1 will provide a higher overall PSNR gain.

This very simple example shows that there exist opportunities to achieve higher overall quality under a given bit budget by simply adjusting the bit allocation plan. The

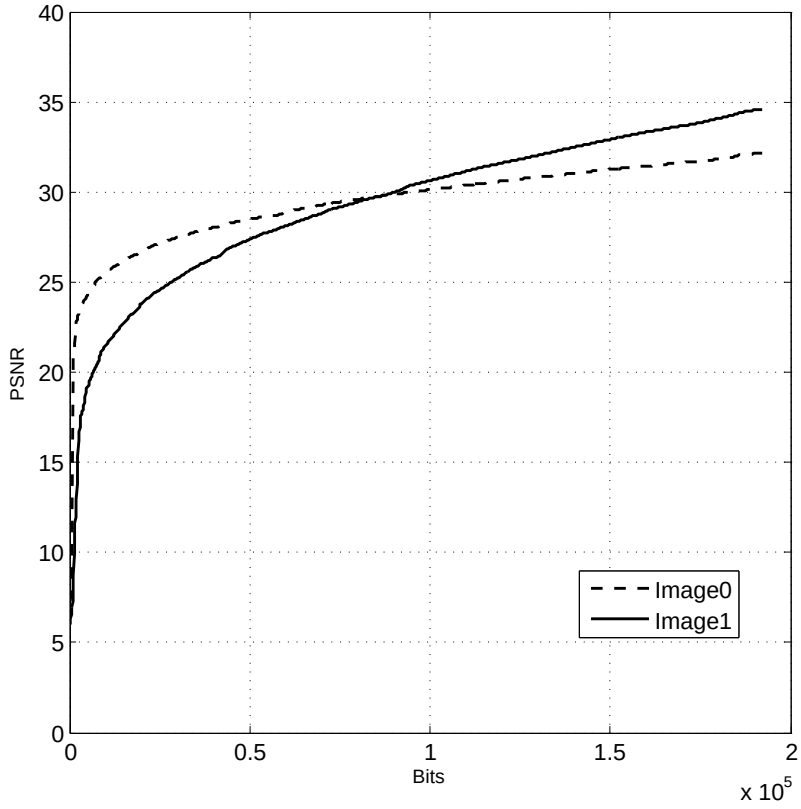

Figure 3.1: Rate-PSNR functions of two image examples

basic idea is as simple as always assigning the bits to the image for which the bits may contribute the highest quality gain.

However, in order to find the best solution, the optimization procedure has to compare the overall quality produced from numerous allocation plans. Potentially it requires image reconstruction at arbitrary bit rate.

As we introduced in previous chapter, many wavelet based image compression methods such as SPIHT are embedded. That is, encoding can be stopped at any specified bit rate and the encoded bit stream can be decoded at any truncated length, resulting in various possible reconstruction qualities [11]. Among them, SPIHT and other later improvements based on the same framework showed excellent importance order on its

coded stream. From Figure 3.1 we can see that the PSNR growth is slowing down with the increasing of bit rate, suggesting that bits are generated in decreasing order of their importance. Bits with higher importance, which contribute higher quality improvement for the image, are always located at the earlier portion of the bit stream. SPIHT also has the property that the quality recovery is relatively even in the entire space domain rather than in part of the pixel blocks. So the block effect is much less than many other algorithms such as JPEG. Therefore, given an arbitrary bit budget, PSNR achieved by SPIHT has much better visual quality. Also it is mentioned in [9] that the coding symbols that are used by SPIHT are in the form of single bits, so the rate can be precisely controlled bit by bit. The embedded characteristics, optimal rate-PSNR properties and precise rate control allow SPIHT for flexible bit allocation without losing its efficiency.

We propose a bit allocation procedure for image archiving under bit budget constraint based on SPIHT algorithm in the following section. The procedure is applied to the bit streams that have been SPIHT algorithm encoded and have the embedded rate-PSNR information measured during the SPIHT encoding. The parameters that specify bit allocation will be archived together with the image content as the overhead. However, later we will show that the impact of the overhead is negligible if the coding rate is in a normal range (for example between 0.1 and 2 *bpp*) and the size of the image to be archived is large enough (for example, normal size for a natural images for displaying and printing). On decoder side, the regular SPIHT decoding procedure will be done for the archived image set and terminated at exactly the bits decided for each image. Experimental results show that by applying the optimization process, more than $1dB$ PSNR gain per image is achievable for a not very large image set comparing with the image archiving plan which assigns equal bits. More importantly, the procedure will have gain on arbitrary image set and a wide range of bit budgets. This approach is completely different from all previous introduced solutions.

3.2 Optimization Formulation of Archiving

3.2.1 Basic Formulation

Let $\mathcal{I}$ be a set of images and be of size M by N indexed by $k, k = 1, 2, \cdots, K$. Let S_k denote the bit stream of the pre-encoded image k using SPIHT to a sufficiently high bit rate.

We are not performing allocation bit by bit because for an individual bit, its PSNR gain contribution fluctuates drastically and is difficult to model. In addition, bit-by-bit allocation entails high computational complexity. Instead, we partition the SPIHT-coded stream of each image into a number of bit blocks. We will show that appropriate partitioning of the encoded bit stream can still provide sufficient granularity in bit allocation while maintaining low computation complexity. More than that, if the bit blocks are reasonably small, the PSNR trend in each block is nearly linear which, as we will show later, allows for linear optimization techniques as a solution to allocation.

We partition S_k into L_k bit blocks. Each bit block in the image set can then be indexed by (l, k), where $l = 1, 2, \cdots, L_k$ and $k = 1, 2, \cdots, K$. Let $s_{l,k}$ denote the number of bit included in bit block (l, k). We define a set of binary variables $\mathbf{X} = \{x_{l,k}\}$, $x_{l,k} \in \{0, 1\}$, $1 \leq l \leq L_k$, $1 \leq k \leq K$ to denote the decision that whether bits are allocated for the block (l, k). That is, bits are allocated for block (l, k) if $x_{l,k} = 1$, otherwise, bit block (l, k) is skipped. First we assume that bits are allocated to each block independently. Let B_k be the total number of bits assigned to image k, then we have

$$B_k = \sum_{l=1}^{L_k} x_{l,k} s_{l,k} \tag{3.4}$$

Let $q_{l,k}$ denote PSNR gain contribution brought by block (l, k). The accumulated PSNR gain by decoding transmitted bit blocks of a single stream S_k can then be represented as

$$Q_k = \sum_{l=1}^{L_k} x_{l,k} q_{l,k} \tag{3.5}$$

where Q_k is defined in (3.2). Note that the assumption that each bit block (l,k) can be selected and transmitted independently does not stand. For any image k, block (l,k) cannot be decoded without completely decoding all its prior bit blocks $(1,k),(2,k),\cdots,(l-1,k)$. Therefore the bits in lth block have contribution on PSNR gain only if $x_{m,k}=1, \quad \forall \ 1 \leq m < l$ otherwise block (l,k) is useless because of the missing prior information. Luckily the constraint is easy to be formulated when the unknown variables $\mathbf{X}$ only take value 1 and 0. The inequality (3.6) will be sufficient to ensure that if a bit block is selected, none of the bit blocks with lower indices will be skipped in the same image.

$$0 \leq x_{l+1,k} \leq x_{l,k} \leq 1 \quad \forall \ l \in [1, L_k - 1], \quad \forall \ k \tag{3.6}$$

With the definitions and formula (3.4), (3.5) and (3.6) above, we now can formulate the allocation optimization problem into the following integer program.

Maximize:

$$Q = \sum_{k=1}^{K} \sum_{l=1}^{L_k} x_{l,k} q_{l,k} \tag{3.7}$$

Subject to:

$$B \geq \sum_{k=1}^{K} \sum_{l=1}^{L_k} x_{l,k} s_{l,k} \tag{3.8}$$

and

$$0 \leq x_{l+1,k} \leq x_{l,k} \leq 1 \ \ x_{l,k} \in \{0,1\}, \ \ \forall \ \ l, \ k \tag{3.9}$$

In the formulation above, B is exactly the bit budget constraint that has been defined in (3.1). Value of Q in the objective function (3.7), which has been defined in (3.3), is the total amount of quality gain in terms of PSNR to be maximized.

Note from equation (3.5) the PSNR gain of each image is with respect to the initial PSNR values $D_k(0)$ so in the objective function (3.7) those initial PSNR values are excluded.

3.2.2 Linear-Programming Relaxation

The bit allocation problem is formulated as integer program in previous section. There is however a problem with this formulation. In practice we may consider allocate bits partially for a bit block in order to use up the bit budget. The integer-program formulation above however insists that either the entire block is allocated bits for or no bit is allocated to the block. As a consequence, the solution to the integer program often has constraint (3.8) satisfied with strict inequality. That is, the specified bit budget may not be used up.

In addition to this problem, it is well known that integer programs are usually difficult to solve and solving them often requires significant computation complexity.

In order to allow bits to be allocated to a fractional block, consider the following linear-program relaxation of the above integer program, where each $x_{l,k}$ takes a real value in $[0, 1]$.

Maximize:

$$Q = \sum_{k=1}^{K} \sum_{l=1}^{L_k} x_{l,k} q_{l,k} \tag{3.10}$$

Subject to:

$$B \geq \sum_{k=1}^{K} \sum_{l=1}^{L_k} x_{l,k} s_{l,k} \tag{3.11}$$

and

$$0 \leq x_{l+1,k} \leq x_{l,k} \leq 1 \quad \forall \ l, \ k \tag{3.12}$$

To give more insight about this linear program, let us define $r_{l,k} = q_{l,k}/s_{l,k}$ to indicate the average PSNR gain per bit in block (l, k). Then equation (3.10) essentially makes an implicit assumption that every bit in block (l, k) contributes exactly $r_{l,k}$ amount of PSNR to the overall PSNR gain. This assumption does not hold strictly in reality but should be approximately correct when the block sizes are made reasonably small.

As noted earlier, with SPIHT, for every image, a block is useful for reconstruction

only when all previous blocks have been selected (i.e., used for reconstruction). In the integer program, this constraint is implemented by (3.9). In the linear program, however, the corresponding constraint (3.12) is not sufficient for this purpose. The solution of the linear program in fact highly depends on $\{r_{l,k}\}$.

3.2.3 Concavity of Rate-Distortion Curve

In this section we prove that if the following proposition holds, the linear optimization procedure will unconditionally guarantee that all $x_{m,k}$ $(m = 1, 2, \cdots, l-1)$ take value 1 before any of the bits in block (l, k) being selected. Interesting thing is, if the proposition 1 holds, all the constraints in (3.12) can be removed.

Proposition 1. *If concavity holds on the entire bit block partitioned rate-PSNR function, for any $k \in K$ and any l where $1 < l < L_k$, after optimization procedure, if $0 < x_{l,k} < 1$, then $x_{m,k} = 1 \quad \forall \ m,\ 0 < m < l$.*

Proof. Assume when concavity holds, after the optimization procedure, if $0 < x_{l,k} < 1$ for a particular (l, k), there exists at least one bit block (m, k), where $0 < m < l$, such that $x_{m,k} < 1$. From the rule of bit block selection described above, occurrence of this scenario implies that the average PSNR gain in (l, k) is greater than or equal to that of bit block (m, k). Or equivalently $r_{l,k} \geq r_{m,k}$. From definition of $r_{l,k}$ we have

$$\frac{q_{l,k}}{s_{l,k}} \geq \frac{q_{m,k}}{s_{m,k}} \tag{3.13}$$

For a given image k, let $(B_{l-1,k}, Q_{l-1,k})$ and $(B_{l,k}, Q_{l,k})$ denote the two points on the rate-PSNR function where $B_{l-1,k}$ and $B_{l,k}$ are the start and end point of bit block (l, k) respectively. Correspondingly $(B_{m-1,k}, Q_{m-1,k})$ and $(B_{m,k}, Q_{m,k})$ denote rate-PSNR function segment corresponding to block (m, k). Draw a line λ by connecting $(B_{m-1,k}, Q_{m-1,k})$ and $(B_{l,k}, Q_{l,k})$, we can find the gradient of λ and denote it as r_λ.

Since concavity holds on the entire rate-PSNR function, we know it must also hold between $(B_{m-1,k}, Q_{m-1,k})$ and $(B_{l,k}, Q_{l,k})$. So PSNR values between these two points will be consistently on or above the line λ. Apparently $(B_{m,k}, Q_{m,k})$ is on or above the line λ and thus $r_{m,k}$ is greater than the gradient of line λ. i.e.,

$$\frac{q_{m,k}}{s_{m,k}} \geq r_\lambda \tag{3.14}$$

Similarly we have $r_{l,k}$ is less than r_λ because $(B_{l-1,k}, Q_{l-1,k})$ is on or above the line λ. So we also have

$$r_\lambda \geq \frac{q_{l,k}}{s_{l,k}} \tag{3.15}$$

Combine (3.14) and (3.15), we have

$$\frac{q_{l,k}}{s_{l,k}} \leq \frac{q_{m,k}}{s_{m,k}} \tag{3.16}$$

This result directly conflicts with equation (3.13). So the assumption is incorrect and the proposition is true. □

3.3 Proposed Scheme

We summarize the optimal bit budget allocation procedure as following. It includes four major steps. Figure 3.2 gives the flow chart of the proposal.

1. SPIHT encoding: Compress each image in the set to a bit stream using SPIHT algorithm. Build up the rate-PSNR function for each image by measuring the reconstructed quality in terms of PSNR at a dense set of bit rates that cover the entire bit stream.

2. Bit block partitioning: Partition the SPIHT encoded bit stream into a set of bit blocks. Correspondingly the rate-PSNR function is divided into segments and characterized by the bit block length $s_{l,k}$ and PSNR gain $q_{l,k}$.

3. Concavity adjustment: Make concavity correction to the partitioned rate-PSNR function of each image. The correction ensures that bits on the same image will be selected in appropriate order.

4. Solve problem: Given a bit budget constraint, use linear-programming method such as SIMPLEX[18] or equivalent to solve the problem and find the optimal bit budget allocation plan for the image set under the bit budget.

3.3.1 Concavity Correction Using Quadratic Programming

From previous analysis we know the rate-PSNR function is not strictly concave across the entire bit rate range even with partitioning unless the size of each bit block is carefully customized for each image to avoid the non-concavity. Instead, we use quadratic program to find a concave approximation based on the partitioned curve. That is, instead of making this curve bit wise concave, the correction only applies to bit blocks. We describe the formulation of the problem as following:

Let $(B_{l,k},\ Q_{l,k})$ be the given set of rate-PSNR measurements each corresponding to the boundary between bit block (l,k) and $(l+1,k)$ on image k where $l = 1, 2, \cdots, L_k$. Let $(B_{l,k},\ P_{l,k})$, $l = 1, 2, \cdots, L_k$, be another arbitrary set of rate-PSNR points. Define function $E(P_{1,k}, P_{2,k}, \cdots, P_{L_k,k}) := \sum_{l=1}^{L_k}(Q_{l,k} - P_{l,k})^2$, namely, the square error. Then for any image k, we can formulate the concavity correction into quadratic programming problem as following

Minimize:

$$E(P_{1,k}, P_{2,k}, \cdots, P_{L_k,k}) = \sum_{l=1}^{L_k}(Q_{l,k} - P_{l,k})^2 \tag{3.17}$$

Subject to:

$$(P_{l,k} - P_{l-1,k})/(B_{l,k} - B_{l-1,k}) \geq (P_{l+1,k} - P_{l,k})/(B_{l+1,k} - B_{l,k}) \quad \forall \quad 1 < l < L_k \tag{3.18}$$

and

$$\begin{aligned} P_{1,k} &= Q_{1,k} \\ P_{L_k,k} &= Q_{L_k,k} \end{aligned} \tag{3.19}$$

The obtained set $\{P_{1,k}, P_{2,k}, \cdots, P_{L_k,k}\}$ is the adjusted new PSNR measures to be fed

to SIMPLEX linear optimization procedure. Notice that both P and Q here are measured on the boundary of bit blocks. Equantions (3.19) ensures that the correction is within an acceptable range so that the optimization bias is limited in a small range. In our implementation, constraint (3.19) is applied by removing $P_{1,k}$ and $P_{L_k,k}$ from the list of quadratic programming variables so they in fact do not increase calculation complexity.

Figure 3.3 shows an example of a rate-PSNR curve and its corresponding concave correction. True values of the adjustment on these bit blocks are listed in Table 3.1. Concave correction changes bit efficiency on multiple bit blocks. It is bit block partitioning dependent. Different bit block partitioning will introduce different correction error. The correction error may be large if the rate-PSNR curve does not follow a nearly concave trend. Using this correction method, the worst case is that the entire rate-PSNR function will be adjusted to a nearly linear curve.

Table 3.1: Example of bit efficiency change with concave correction

Bit rate (bpp)	0.5298	0.5864	0.6484	0.7168	0.7920	0.8745
Original PSNR (dB)	37.0453	37.5049	37.9421	38.3014	38.9196	39.3930
Adjusted PSNR (dB)	37.0453	37.4962	37.9155	38.3759	38.8804	39.3930

Once the non-concave limitation in rate-PSNR functions being removed, pick up two bit blocks on any rate-PSNR curve, from equation (3.16) we know the PSNR growth rate of any bit blocks with lower index is always higher than that of bit blocks with higher index. Any available bit, as long as it is assigned to image k, will always be first assigned to the earlier blocks. As the result, the constraints in equation (3.12) can be removed without any impact and the algorithm is further simplified.

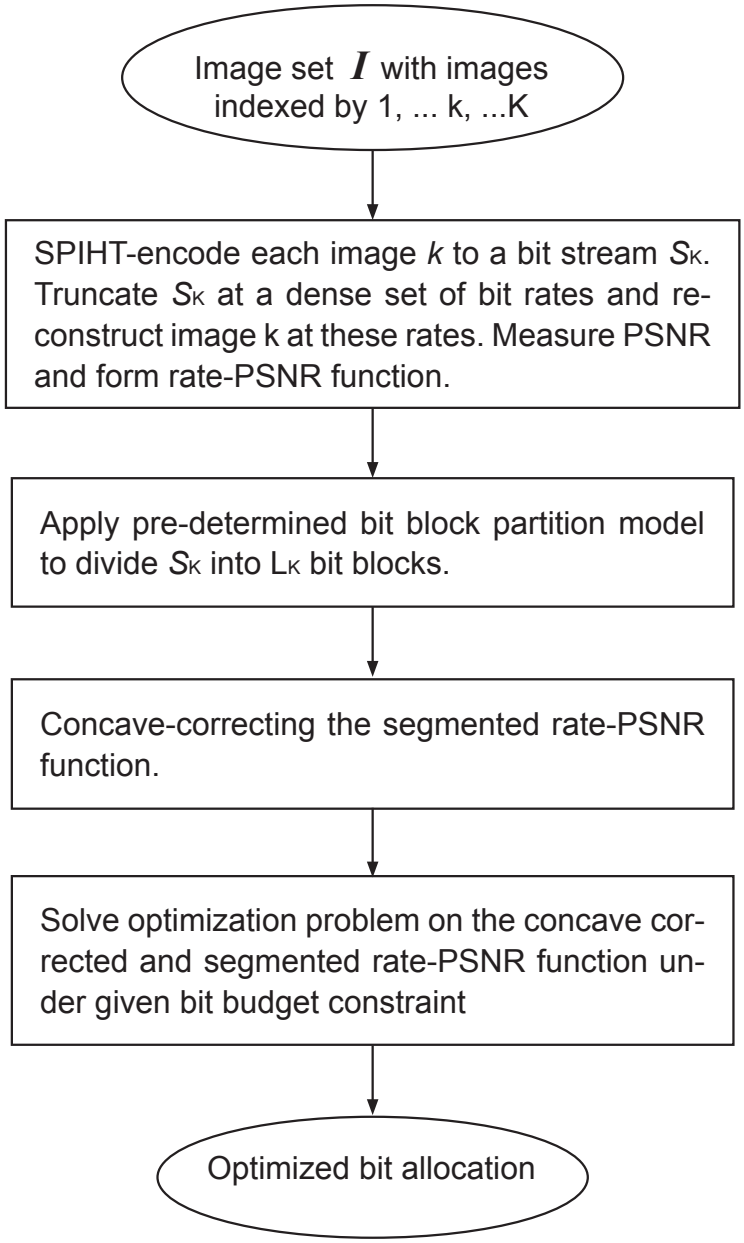

Figure 3.2: Flow chart of allocation optimization for SPIHT-based image archiving

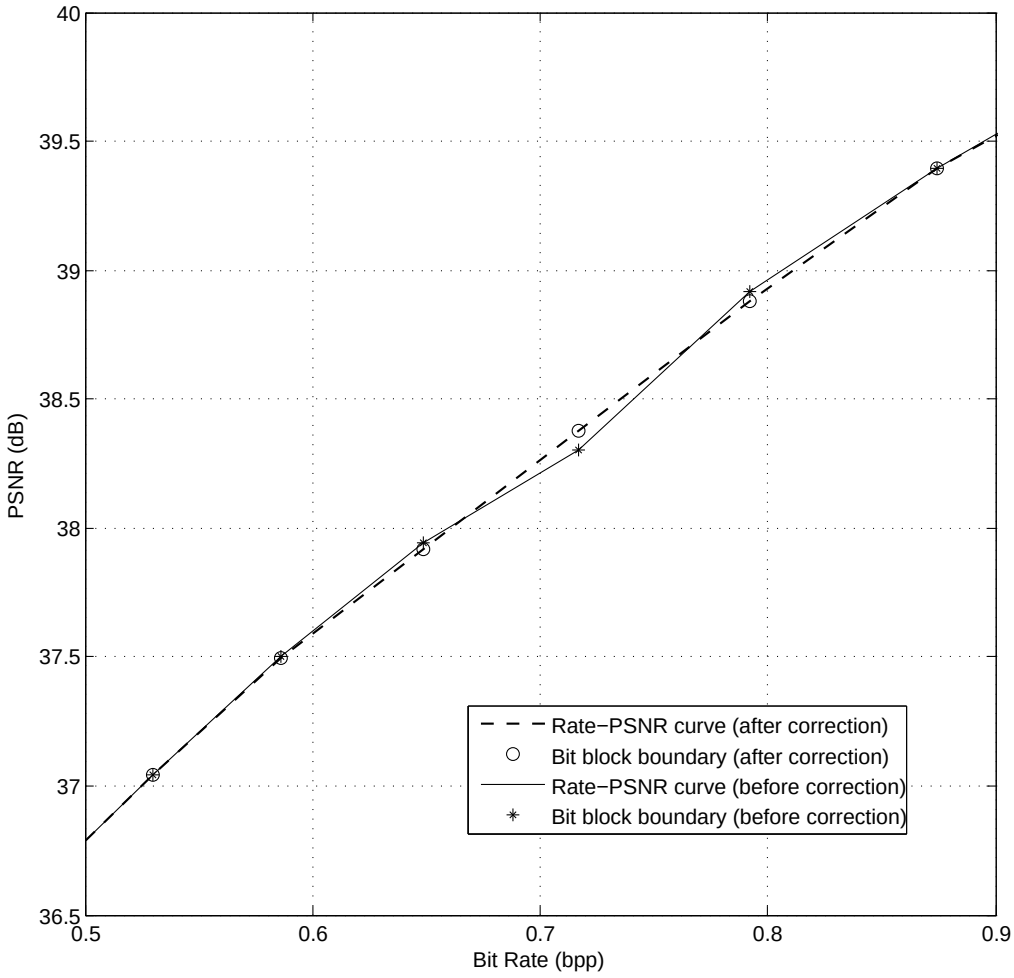

Figure 3.3: A rate-PSNR curve before and after concave correction

Chapter 4

Experimental Results

4.1 Image Data Set

Our experiments were performed on a collection of total number of 500 monochrome, 8 *bpp*, 512×512 size images which were selected from public research image databases [1]. Images are selected randomly from various categories such as biomedicine and geography etc., areas. Both natural and artificially processed images are included. Their diverse rate-PSNR characteristics are helpful for investigating universality of the method. The purpose of using a large number of images is to introduce high degree of freedom to ensure flexibly and freely bits allocation among images. Another advantage is the capability of analyzing the performance on different scale of set size. Later we will also show that a large enough size of image set ensures the performance stability.

Discussion below is mostly interested in the absolute bits applying to a single image or an image set under the bit budget constraint. But sometimes "rate" presentation such as bit per pixel (*bpp*) is more convenient and easier to analyze and display the results. Since the images selected have the same dimension and quality scale, "bit rate" and the absolute number of bits are equivalent in terms of performance evaluation. So we sometimes mix the usage of both in the discussion below.

All images were prior encoded to a desired bit rate using SPIHT algorithm. Our

[1]Including images from http://www.tecnick.com/; http://decsai.ugr.es/cvg/dbimagenes/index.php and http://sipi.usc.edu/database/

experiments focus on the impact of relative bit efficiency among images rather than absolute performance of the wavelet based image compression algorithm. Although for a given image, rate distortion curves vary by using different wavelet based compression algorithms, selection of compression algorithms will not impact the conclusion as long as the algorithm exhibits the similar rate distortion behavior as that of wavelet based algorithm. So we select the basic SPIHT algorithm II from [9] without arithmetic binary encoding and further optimization. Selection of wavelet filter is also outside the scope of the discussion in our proposal, so we simply use the 4 -level pyramids constructed with 9/7-tap filters for wavelet transform as selected by [9]. The same as in [9], bit rate for each measurement is not the result of entropy estimates but calculated from the actual number of bits being used.

We use rate-PSNR as the rate distortion metric in our experiments. We observed that normally a recovered image achieves a PSNR above 45 dB when it is encoded to a rate around two bits per pixel (2.0 *bpp*). Visually this level of rate provides quality that very close to the original image. Optimization at a quality region higher than that has no significant benefit. Besides, at a very high bit rate range (greater than 2.0 *bpp*) the rate-PSNR function is nearly linear for most of the images[16]. (See Figure 4.1, which uses the famous Lena image as an example). So we only one time encode all the images to 2.0 *bpp* in our experiments. During the SPIHT encoding, instead of being measured at each bit, PSNRs are captured every 128 bits. All measurements form a dense sampled discrete rate-PSNR function. PSNR values were obtained by comparing the amplitude of recovered image pixels in space domain with that of the original image for each pixel. Benefiting from the embedded properties of SPIHT codec, the procedure to collect the embedded rate-PSNR information does not take too much effort. The whole work is mostly done on the decoding side with only one complete encoding procedure.

The sampling of rate-PSNR curve does not harm the results. In our experiments later, we let all bit blocks start and end at the sampled bit rate therefore the bit blocks always have accurate PSNRs increment. From the description of the approach in previous chapter, we know it does not introduce too much approximation error because most of the bit blocks are either fully used or skipped except the last ones that will be taken at the last as a portion. Interpolation will be done for these partially used bit blocks.

However, the interpolation will be done in a very small 128 bit interval, at any bit rate, the PSNR dynamic in such a small range is nearly linear and interpolation result will be accurate.

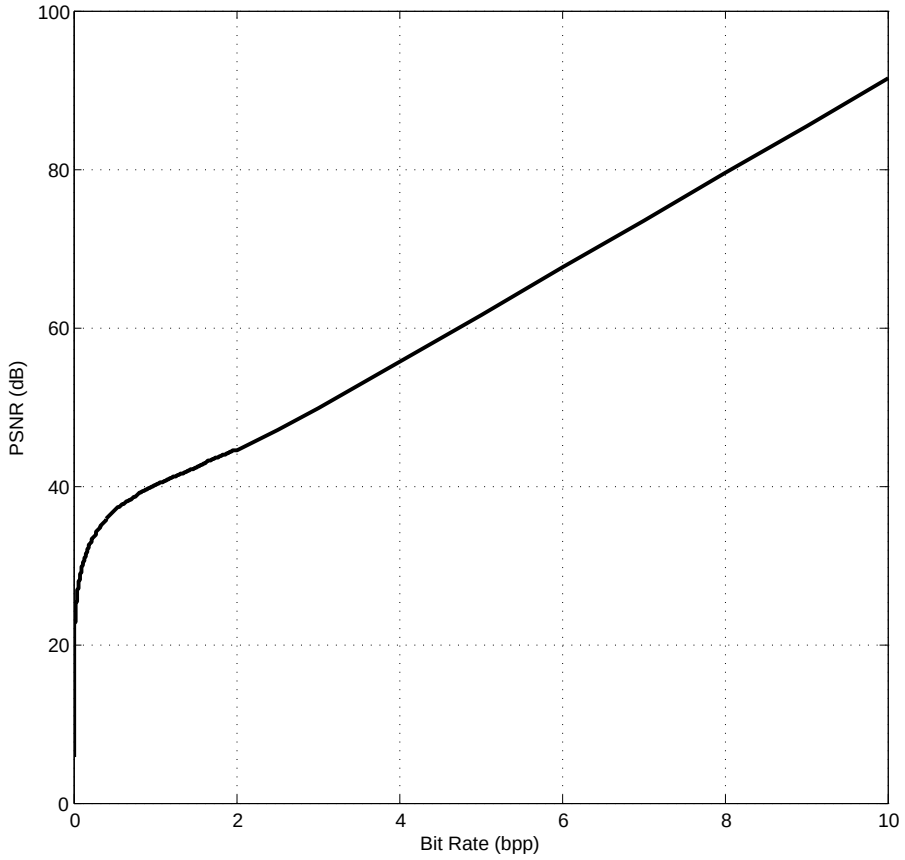

Figure 4.1: Lena rate-PSNR curve up to 10 *bpp*

4.2 Implementation Considerations

4.2.1 Bench Mark and Performance Metric

Finding a bench mark to evaluate the performance is not very straightforward. It is not easy to find the reference to compare the performance because the proposed approach studies the "relative" bit efficiency based on a fixed SPIHT like compression algorithm

and the absolute performance changes while we change the base compression algorithm. So we evaluate the performance by comparing the approach with the results from an archiving plan without allocation optimization, which refers to an equal bit allocation scenario. In later discussion, we often use "EBA" to represent the results produced from this scenario. Correspondingly, we refer to the proposed approach an unequal bit allocation or the "UBA" plan in short. Let $\mathcal{P}_e$ denote the PSNR obtained for an image by applying the EBA plan and $\mathcal{P}_s$ denote the PSNR from UBA plan for the same image under the same bit budget. Let $\mathcal{P}_\Delta$ denote the PSNR gain achieved by using UBA instead of EBA, we have

$$\mathcal{P}_\Delta = \mathcal{P}_s - \mathcal{P}_e \tag{4.1}$$

For a K image set, the achieved average quality improvement, represented by the average PSNR gain, is

$$\bar{\mathcal{P}_\Delta} = \frac{1}{K}\sum_{K}\mathcal{P}_\Delta \tag{4.2}$$

4.2.2 Bit Stream Partitioning

There are numerous ways to partition the SPIHT encoded stream S_k into bit blocks. Naturally, LIS, LIP and LSP pass defined in SPIHT procedure provide a way to partition the encoded stream S_k where each bit block maps to a particular coding pass. However, this partition does not provide a meaningful and consistent segmentation for different images. On the other side, this partition usually cannot provide enough granularities and the optimization performance is thus limited.

Let the length of first bit block $s_{1,k} = a$ and let every next bit block size increasing by a factor of r, we have

$$s_{l,k} = s_{l-1,k} \times r, \quad l > 1 \tag{4.3}$$

Equation (4.3) then provides a partioning strategy. It is simple because we only need two parameters to control the partitioning. We set $r > 1$ and let a sufficiently small so that the partition can sense the PSNR change quickly at the low bit rate and remain stable at high bit rate. Specifically when $r = 1$, all bit blocks will have the equal length a. From our experiments it makes the bit blocks unnecessarily small at higher bit rate,

so usually is not optimal.

Values of a, r and correspondingly the number of bit blocks are not necessarily have to be identical for all images. But practically it is much simpler to force the parameters to be the same for all images.

In our experiments, because the rate-PSNR curves are discrete, when r takes arbitrary value, usually bit blocks do not bound at sampled points. But because the sampling density is very high at a pace of 128 bits, interpolating the PSNR gain between two samples to estimate the PSNR gain is not difficult. However, because we know precisely the PSNR and bit rate at each sample points, in our experiments, we always ensure the partitioned bit blocks starting and ending at the real measurement samples so that the PSNR gain for each bit block is accurate. That means each bit block approximately follows the equation (4.3). Figure 4.2 shows an example of bit block partitioning and boundaries of the bit blocks on the rate-PSNR curve of Lena image. We applied different combination of a and r in our experiments following the principles above and observed how they actually impact the performance. Results will be shown in later sections.

4.2.3 Overhead

In order to decode SPIHT encoded stream, overhead has been added for each image. The overhead contains information of scale levels, image size and initial threshold. As a part of the compressed stream, they are the payload and have no impact to performance. In addition to that, to apply the "UBA" scheme to the image set, extra information has to be added to describe the bit block partitioning scheme and the bit allocation prior to the content information. For example, parameters a and r for partitioning will need to be conveyed to the decoder side prior to data transmission. If the bit block partitioning scheme is unique for all images, these parameters are negligible for a large image set. Under a bit budget, the index of the images which contain the fractional bit block must be informed. Besides, the number of bit blocks selected for each image is different. Decoder will not be able to predict the termination of the stream of each individual image. The number of bit blocks selected for each image or the order information of the bit blocks that consist in the combined bit stream therefore are necessary for decoder.

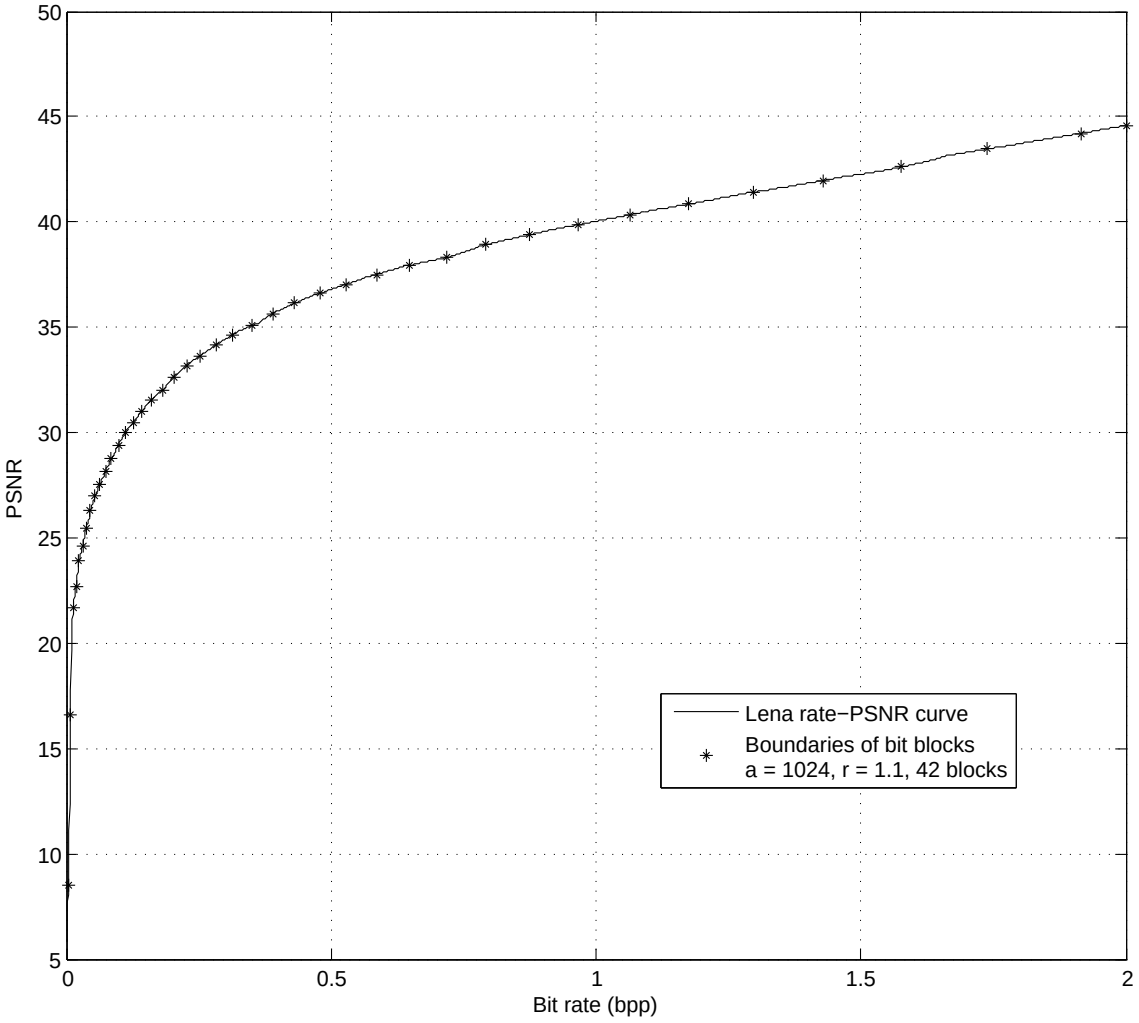

Figure 4.2: Example of bit block partitioning

However, the size of overhead highly depends on the application. The method of bit block partitioning, degree of information sharing among images and the size of image set are all impact factors. With a preliminary estimation, we anticipate that with carefully design, the overhead of "UBA" can be controlled to a very small size so that its performance impact is very minor. We therefore did not count in the overhead when comparing UBA with EBA in our experiments.

4.3 Results and Discussion

In this section we discuss the results of UBA we obtained from the 500 image set. We first show the results from a simplest two images scenario to understand how the bits were allocated between images. In second subsection we will show the best performance achieved on the image set, discuss the behavior of bit allocation and understand the performance change due to the selection of parameters a and r. In third subsection we focus on the relative low bit rate range (normally less than 1 *bpp*) that is mostly interested comparing to the high bit rate range. At last we observe the performance impact with the change of the size of image set.

4.3.1 Two Images Study

We start our experiments with a series of two images simulations. We chose famous Barbara and Gold Hill from the image set (Displays in Figure 4.3) and plot their rate-PSNR curves using both EBA and UBA in Figure 4.4. Note here the rate is no longer the single image rate, instead it represents the overall bit budget. We selected 500 bit budget points with equal interval from 128 bytes to 62.5 k bytes per image to form the PSNR curve. Parameter $a = 128$ bytes and $r = 1.1$ are used for UBA method in this group of tests.

The shapes of the curves from both methods are significantly different. The EBA method produces very smooth curve, which indicates that both images continuously improve PSNR with the growth of bit budget. On the contrary, UBA method creates clear turns on curve, which reflects the fact that the bits were only assigned to one of

Figure 4.3: Sample images: Left: image A - Gold Hill; Right: image B - Barbara

the images instead of both therefore PSNR only grows on one image at a time. The only exception is that if both images have the same bit efficiency for the next yet selected bit block, quality of both images will grow simultaneously. In Figure 4.4, image Barbara (Image B) was assigned more bits by using UBA than EBA and achieved higher PSNRs on most of the bit budget samples. At the same time, quality of image Gold Hill (image A) is degraded. For both images, at the end the PSNRs from both UBA and EBA reach the same level. This is obvious because bit streams of SPIHT compressed images stop at 2.0 *bpp* so they are fully decoded eventually and the highest PSNR values are limited by the final SPIHT compression results.

Figure 4.5 from another aspect illustrates the effect of bit allocation between two images at a serials of bit budgets. The paths emphasize the PSNR evolving for both images. Method EBA and UBA again show very different behavior. Many sharp turns appearing on the curve of UBA tell that the preference of bit allocation or the relative bit efficiency at any moment is clear. Note that both curves are plotted by connecting measurements from 500 discrete bit budget points, it is possible from last bit budget to current bit budget one bit block from one image is completely selected and the next bit

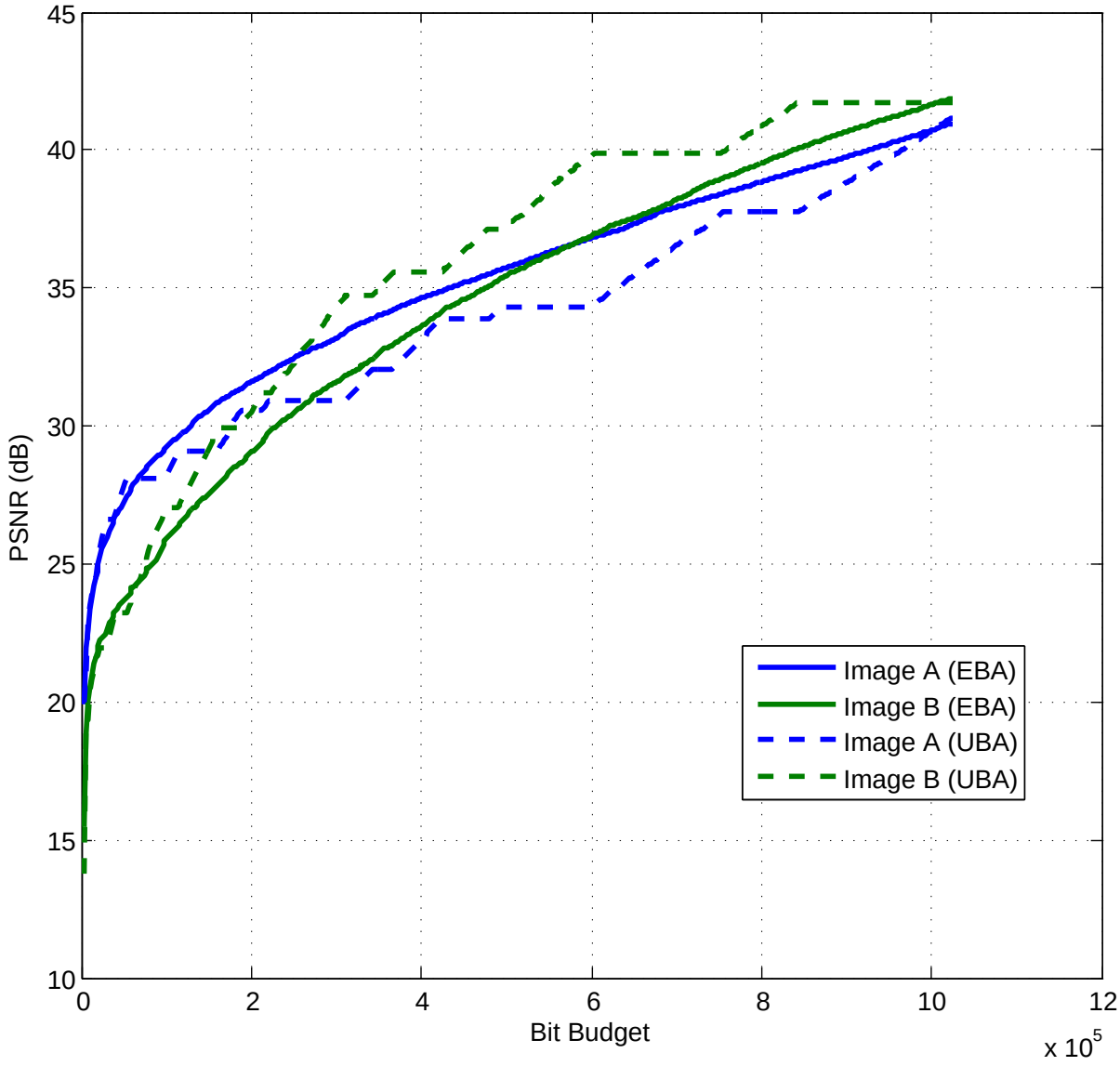

Figure 4.4: Two image bit budget - PSNR contrasts: EBA v.s. UBA

block is chosen from the other image partially, so both images gained PSNR and the UBA curve is not strictly in parallel with the axes. It happened at earlier part of the curve since the samples that form the curve are sparse. At the later part of the curve, the sample density is much higher (did not show in the figure) because PSNR increasing rate slows down and the turns on the curve are always sharp and clear.

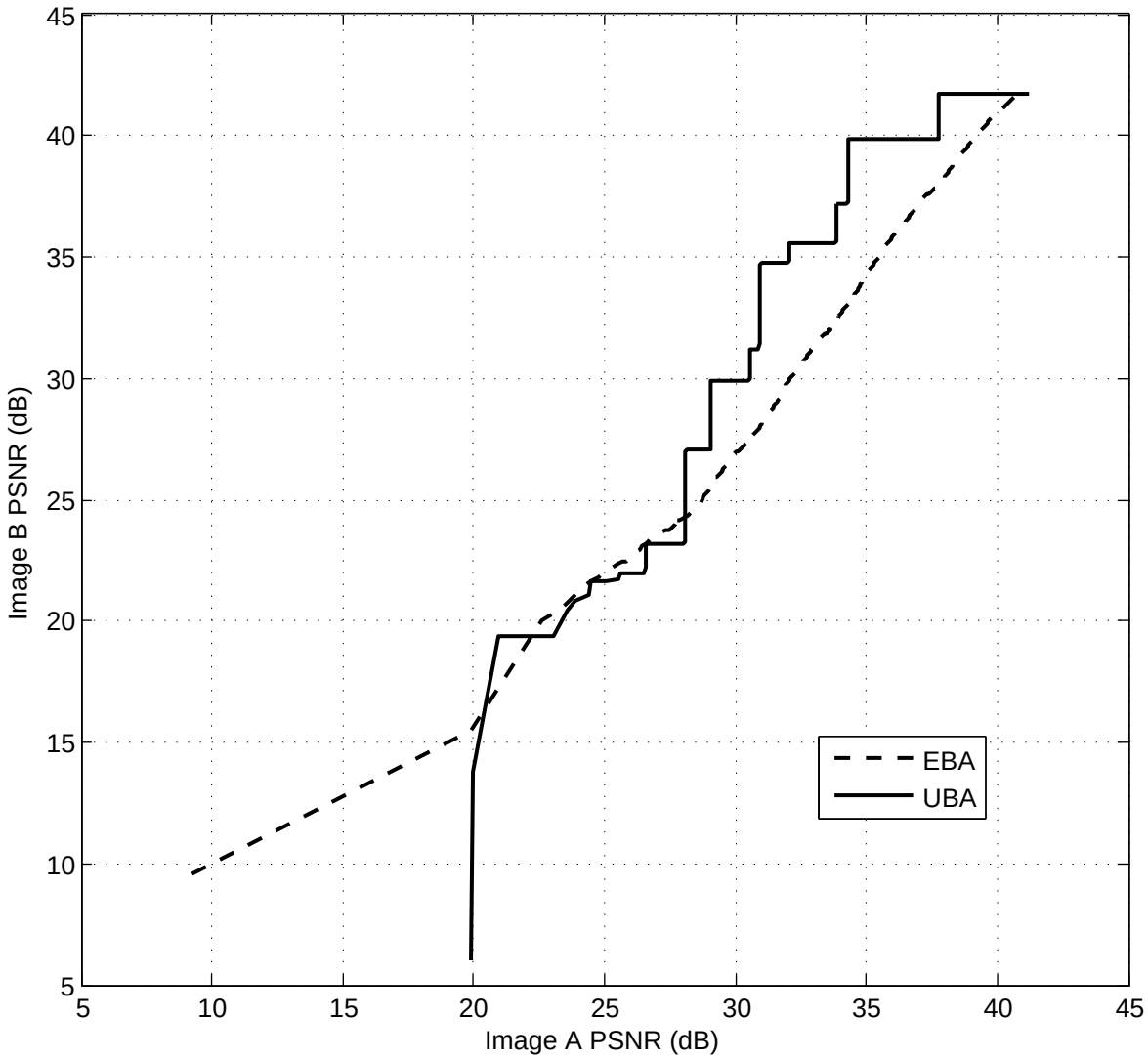

Figure 4.5: Two image PSNR contrasts: UBA v.s. EBA

4.3.2 Best Archiving Performance For An Image Set

We included all 500 images in this group of tests to show the best performance of UBA in compare with EBA for a large set of images.

First we fixed parameter $a = 128$ bytes and $r = 1.1$, each image stream is divided into 42 bit blocks. Later we will show that this group of parameters provides nearly the best result. Figure 4.6 displays the average PSNR on the 500 images archiving using both the UBA approach and the EBA approach. Similar to Figure 4.4, curves are also plotted by connecting PSNRs measured at the same 500 bit budget samples. Differently, each point on the curve indicates the average PSNR of the 500 images at each particular bit budget. In order to keep a clear scale of the curve in the figure, the first a few bit budget samples that produce lower than 20 dB average PSNR at the very low bit rate are excluded. We can see that the distance between EBA and UBA curves, which reflects the average PSNR gain $\bar{\mathcal{P}_\Delta}$ as defined in equation (4.2), changes fairly successive. At the bit budget around 1.18 bpp, or equivalently, around 148 M bytes (or $1.5 \cdot 10^8$ bits), the average PSNR gain ($\bar{\mathcal{P}_\Delta}$) achieved peak value which is more than 1 dB. Average PSNR gain decreased after the peak point and curves of both approaches eventually merged since each image is only pre-encoded to 2 bpp and the bit streams of more and more images are fully involved with the growth of bit budget. This matches the reality that the length of pre-encoded SPIHT bit stream is finite. Allocation is meaningless if the bit budget is greater than the total bits of all the SPIHT encoded bit streams. From the figure we can clearly see that UBA always provides better average PSNR when bit budget is limited except at the very low bit rate. From another direction, given a target average PSNR, UBA has significant bit budget saving. We selected a few examples of bit budget savings and listed them in Table 4.1. Among them, 39.5 dB (resp. 38.4 dB) is the average PSNR of UBA (resp. EBA) at the point where the max average PSNR gain was achieved. From the table we can see that when the average PSNR is between 30 dB and 40 dB, the bit saving of UBA approach may achieve as high as 15%.

We created another view to show the absolute value of average PSNR difference between EBA and UBA in any given bit budget range. Magnitude of the curve directly represents the average PSNR gain $\bar{\mathcal{P}_\Delta}$. Figure 4.7 shows the result of Figure 4.6 from this view. It is more straightforward to show the performance and provides better scaling of

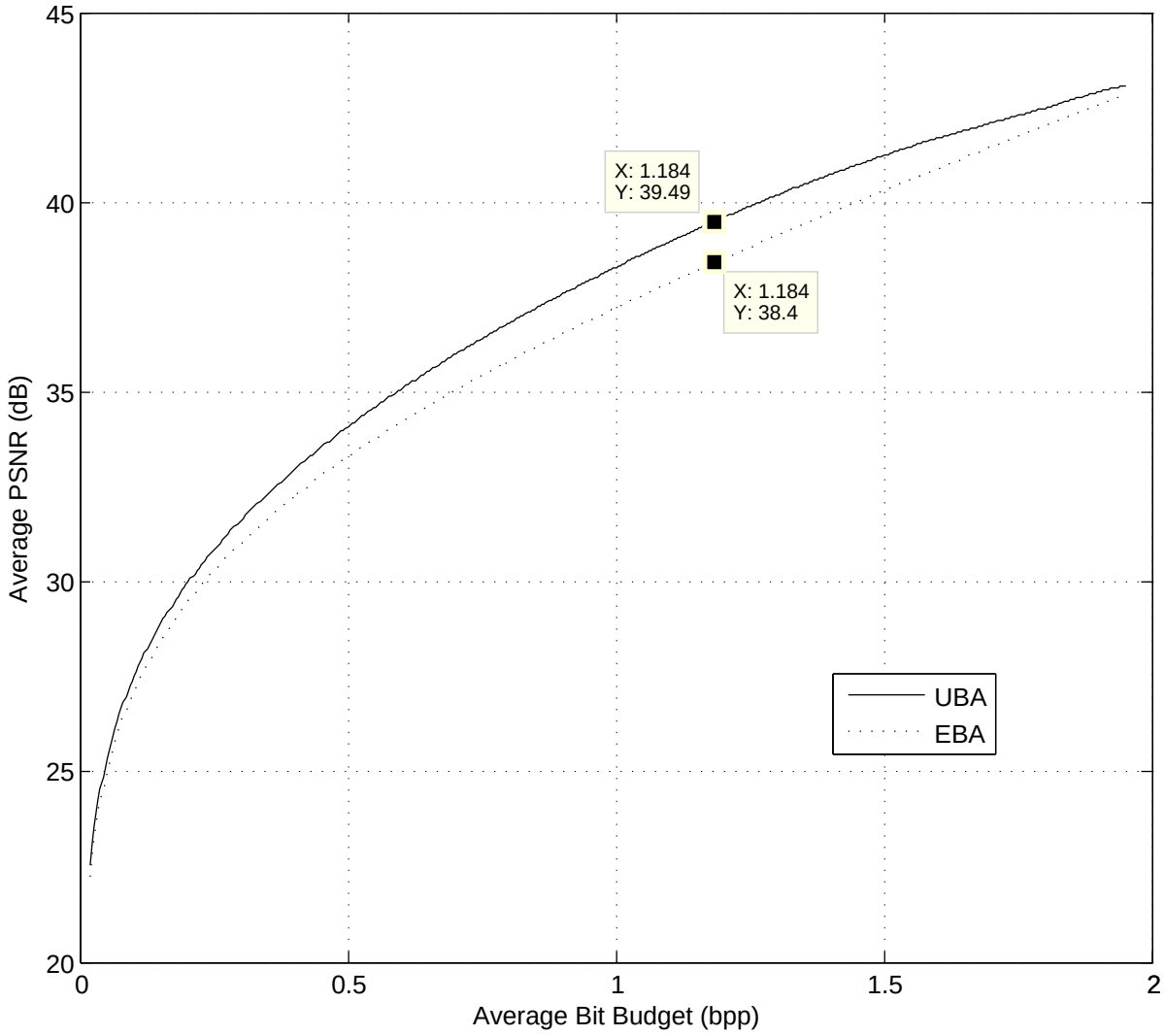

Figure 4.6: Average PSNR gain (average $\mathcal{P}_s$ and average $\mathcal{P}_e$): $a = 128$ Bytes, $r = 1.1$

Table 4.1: Bit budget saving with UBA

Average PSNR (dB)	39.5	38.4	35	30
bpp of EBA	1.363	1.184	0.695	0.231
bpp of UBA	1.184	1.016	0.590	0.203
Bit budget saving	13.1%	15.0%	15.1%	11.7%

the data. Later we will use this type of figure to analyze the peak performance under different key parameters a and r. The fact that all data analysis uses the same EBA result as references makes this type of figure the best for the performance comparison between different settings.

Approach UBA at each bit budget in fact has an independent problem setting. The PSNR increments in the extreme low bit rate region are largely variant and very unpredictable. With UBA the uncertainty results in the inconsistency between average PSNR growth rate of bit block and the local PSNR rate. In addition, UBA approach has a "taking-bit-blocks-in-full" effect. When bit budget is severely limited, many of the images are not selected. Average PSNR gain performance therefore is unpredictable and drastic fluctuates on the extremely low bit budget region. The performance of UBA sometimes is high but sometimes even worse than EBA. In one word, the performance at this region is not very reliable.

Figure 4.7 also shows that with a fixed partitioning configuration, there is only one performance peak. The average gain dropped quickly at low bit budget range.

The UBA approach is built on the idea of unequal bit allocation among images. Inevitably some images will lose certain degree of quality while others gain extra comparing with equal bit allocation. Figure 4.8 clearly shows the quality change from EBA to UBA for each individual image. It provides an atomic view of the performance of UBA approach at a provided bit budget.

Figure 4.8 is taken at the bit budget where the maximum average PSNR gain $\bar{\mathcal{P}_\Delta}$ is achieved as shown in Figure 4.6. The 45 degree line indicates equal PSNR value from both approaches. An image has no PSNR change from EBA to UBA if the sample that represents the image is located on this line. Samples above the line are the images that gain quality from the UBA approach. Meanwhile, images under the line lost quality with UBA. Vertical distance between a sample and the zero gain line shows the amplitude of quality change. In this figure, almost half of the 500 images actually have quality downgraded. Concretely, the ratio of quality gain to the quality lost is 262 : 238 in this figure. However, visually in average the quality downgraded samples have shorter distance from the line than upgraded samples. Means the average quality lost is smaller than the average quality gain. This explains why we have overall quality gain for the

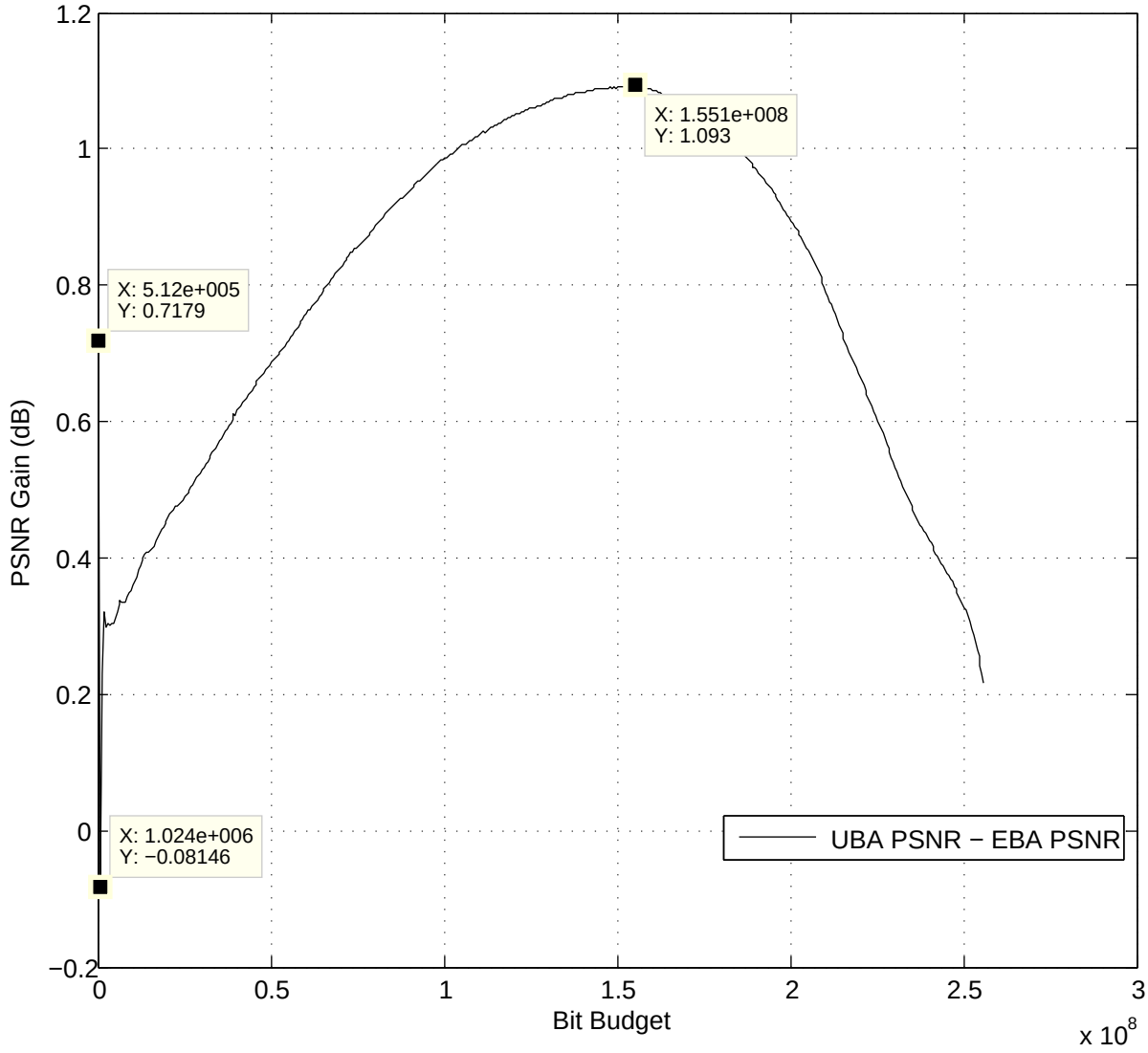

Figure 4.7: Average PSNR gain ($\bar{\mathcal{P}_\Delta}$) , $a = 128$ Bytes, $r = 1.1$

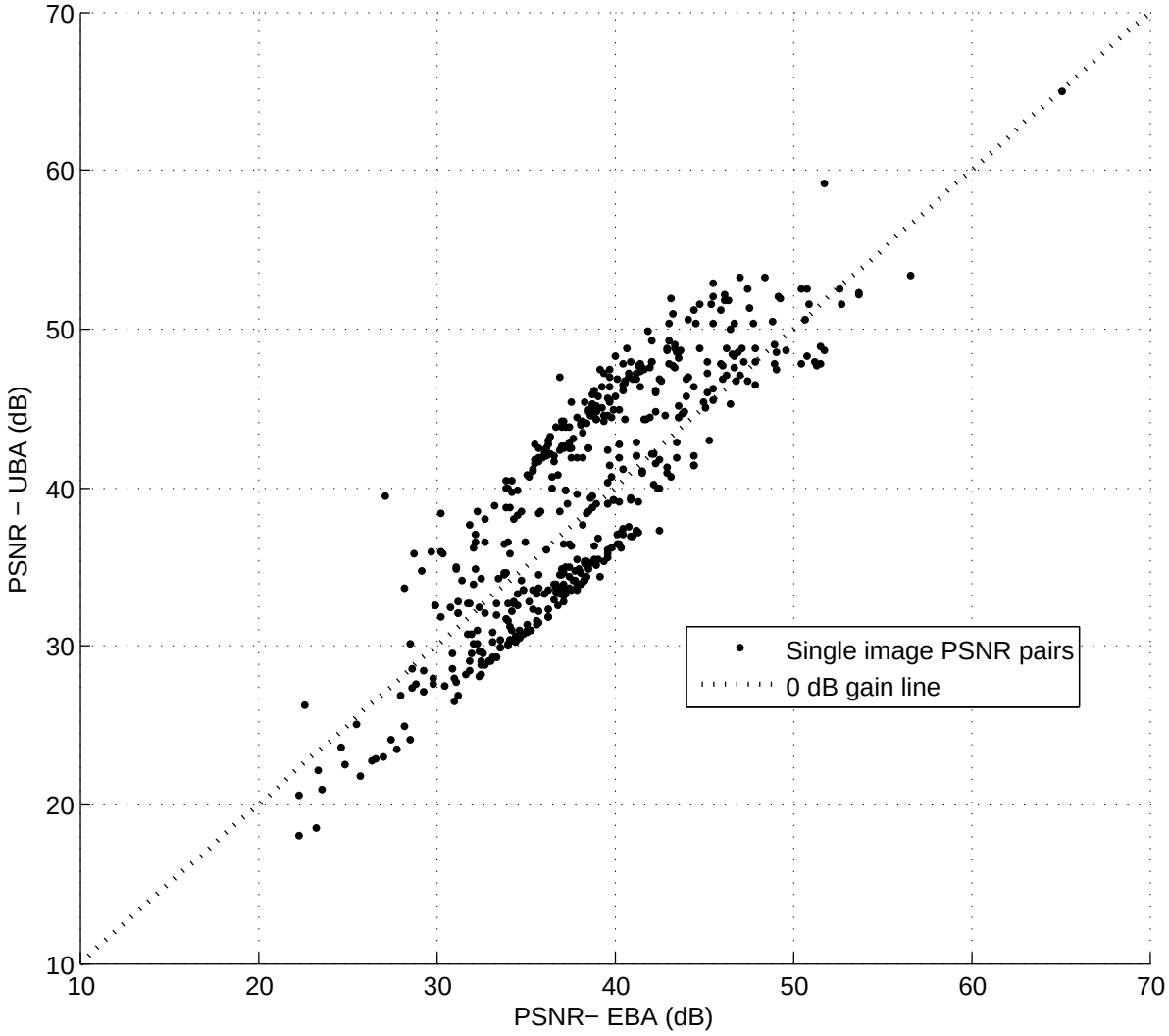

Figure 4.8: Single image UBA PSNR v.s. EBA PSNR when maximum average gain is achieved: $a = 128$ Bytes, $r = 1.1$, bit budget = 147.95 MBytes

image set.

Figure 4.8 also tells that images with lower PSNR when using EBA are losing PSNR with higher probability. It makes sense because by using the same number of bits, if an image has lower PSNR, in average each bit used on this image has less efficiency. Allocating bits to other images will usually contribute more PSNR gain. It matches the expectation that we exchange the quality of some images for higher overall quality. However, this leads to an unwanted consequence that using UBA, lower quality images are often further impaired and higher quality images gain extra unnecessary benefit from the bit re-allocation. This drawback is interpreted in Figure 4.9 in which the variance of image PSNR increased from EBA to UBA and the histogram shows bi-modal status. Variance change of the images set is useful to indicate the performance and apparently we want to have the variance change as little as possible.

Figure 4.10 shows the quality change of real image examples under the same bit budget between EBA and UBA. The samples are not selected randomly. One image has the lowest PSNR with EBA and the other one has the largest PSNR lost from EBA to UBA. These images lost some details but still visible after significant bits reduction. Advantage or disadvantage of the bit reduction and quality reduction depends on how these images need to be used for specific application.

As a comparison, Figure 4.11 shows another two real image examples under the same bit budget. Differently, they benefit from UBA. One of these two images has very complex texture, with EBA its quality is limited at very low level even with high bit budget. UBA help it visibly improved the quality significantly. The other one is the case that had maximum PSNR gain among the set.

In the second group of experiments, we selected different combination of starting bit block size a and power ratio r to verify the impact of bit stream partitioning. Results of different combinations are listed in Table 4.2. Interestingly we notice that parameter change has little impact on the max possible PSNR gain especially when the total number of bit blocks is higher than a certain amount. The measured maximum PSNR gains with the selected parameter combinations have the variance less than 0.2 dB per image which is very small. Heuristically we can see the trends are: Equal length bit block partitions, although produce high amount of bit blocks, achieve lower maximum gain than unequal

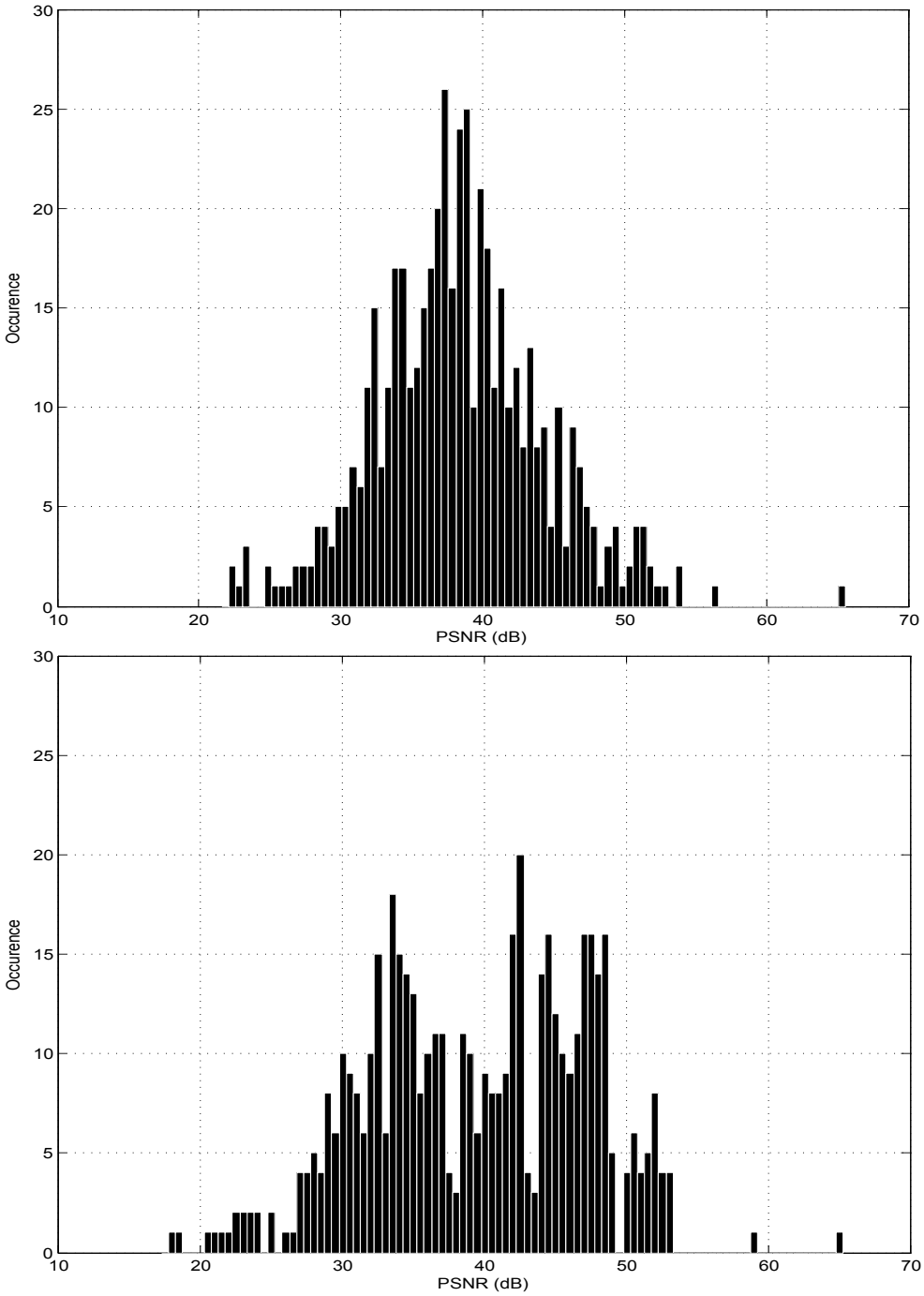

Figure 4.9: PSNR Variance: Up: EBA, $var = 34.65$; Bottom: UBA, $var = 57.84$. $a = 128$ Bytes, $r = 1.1$, bit budget = 147.95 MBytes

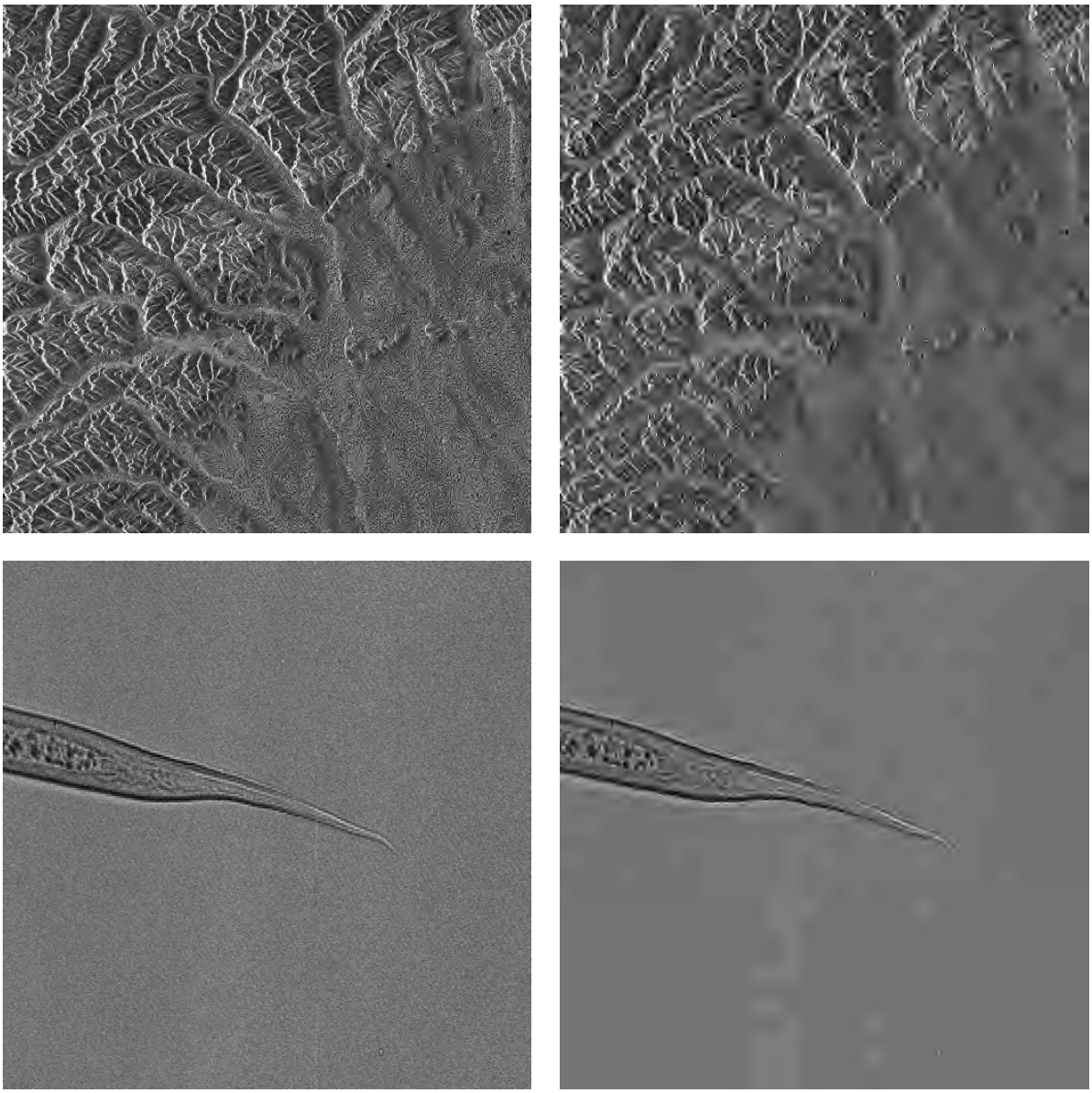

Figure 4.10: Example of PSNR comparison using EBA and UBA. Left: EBA. Up: 23.3 dB; Bottom: 31.3 dB. Right: UBA. Up: 4.76 dB lost, save bit budget 89.3%; Bottom: 4.40 dB lost, save bit budget 97.9%

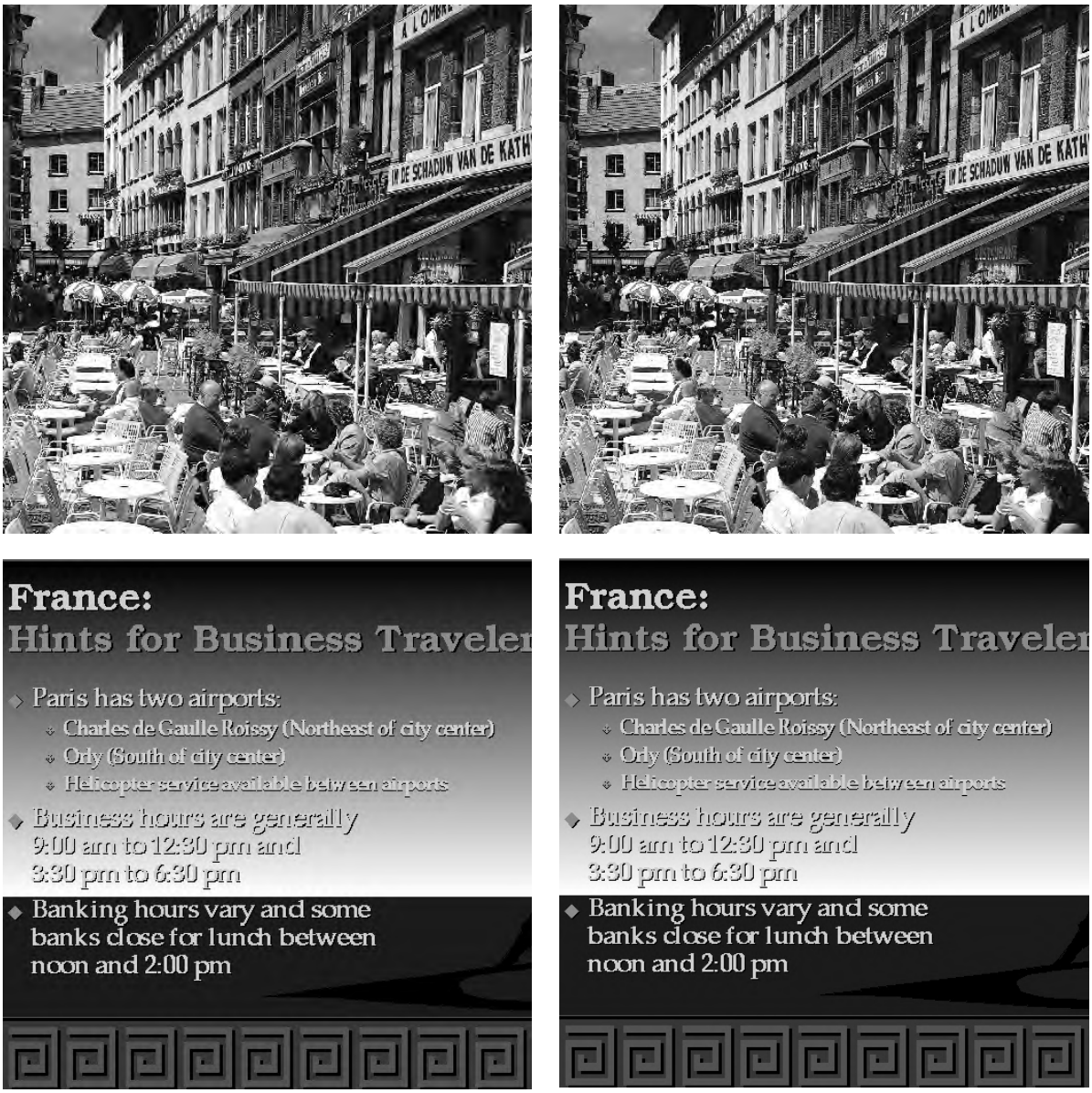

Figure 4.11: Example of PSNR comparison using EBA and UBA. Left: EBA. Up: 22.6 dB; Bottom: 36.9 dB. Right: UBA. Up: 3.65 dB gain, consume extra 46.8% bit budget ; Bottom: 9.97 dB gain, consume extra 68.9% bit budget

partitions. With unequal length bit block partitioning, given a fixed a, higher r which lead to less bit blocks has the lower performance; with fixed r, as long as a is not too large, it has very little impact on the max performance. Parameter r plays more important role on the highest performance.

Table 4.2: Best Performance Versus Parameter Change

a (byte)	r	Total bit blocks	Best average PSNR gain (dB)		a (byte)	r	Total bit blocks	Best average PSNR gain (dB)
16	1.1	64	1.091		16	1.5	19	1.068
16	2.0	10	1.024		16	4.0	7	0.934
32	1.1	56	1.090		32	1.5	18	1.068
32	2.0	12	1.025		32	4.0	7	0.970
64	1.1	49	1.091		64	1.5	16	1.068
64	2.0	11	1.024		64	3.0	7	0.954
128	1.1	42	1.093		128	1.5	14	1.068
128	2.0	10	1.025		128	3.0	7	0.999
256	1.05	54	1.081		256	1.1	35	1.091
256	1.5	12	1.065		256	2.0	9	1.026
512	1.0	129	1.026		512	1.1	28	1.089
512	1.5	12	1.067		512	2.0	8	1.026
1024	1.0	65	1.058		1024	1.1	21	1.089
1024	1.5	9	1.066		1024	2.0	7	1.026
2048	1.0	33	1.084		2048	1.1	16	1.087
2048	1.5	7	1.054		2048	2.0	6	1.022
4096	1.0	17	1.090		4096	1.1	11	1.081
8192	1.0	8	1.049		8192	1.1	7	1.039
16384	1.0	4	0.914		16384	1.1	4	0.910

Maximum PSNR gain has single peak value in the wide bit budget range, and the point that produces the best performance, from our observation, is happening at various bit budget when parameters are different. However, they usually appear around 1.5 *bpp*, i.e., a relative high bit rate. Possibly the maximum performance is only depending on the characteristics of image set and the maximum bit rate each image can achieve. In order to clearly see the impact of the parameters to the highest achievable gain, we plot Figure 4.12 and 4.13 to show how the parameters impact the peak performance and the trend

around the peak point. Figure 4.12 uses a fixed a but different r. We see that the best performance is reducing when r increases. Meanwhile, Figure 4.13 shows that when r is fixed, the best performance drops quickly when a is large. But once a is selected small enough, it has very small impact on the best performance. Interestedly, performance is not very sensitive to the parameter change in a relative wide range so selecting of parameters is not very critical.

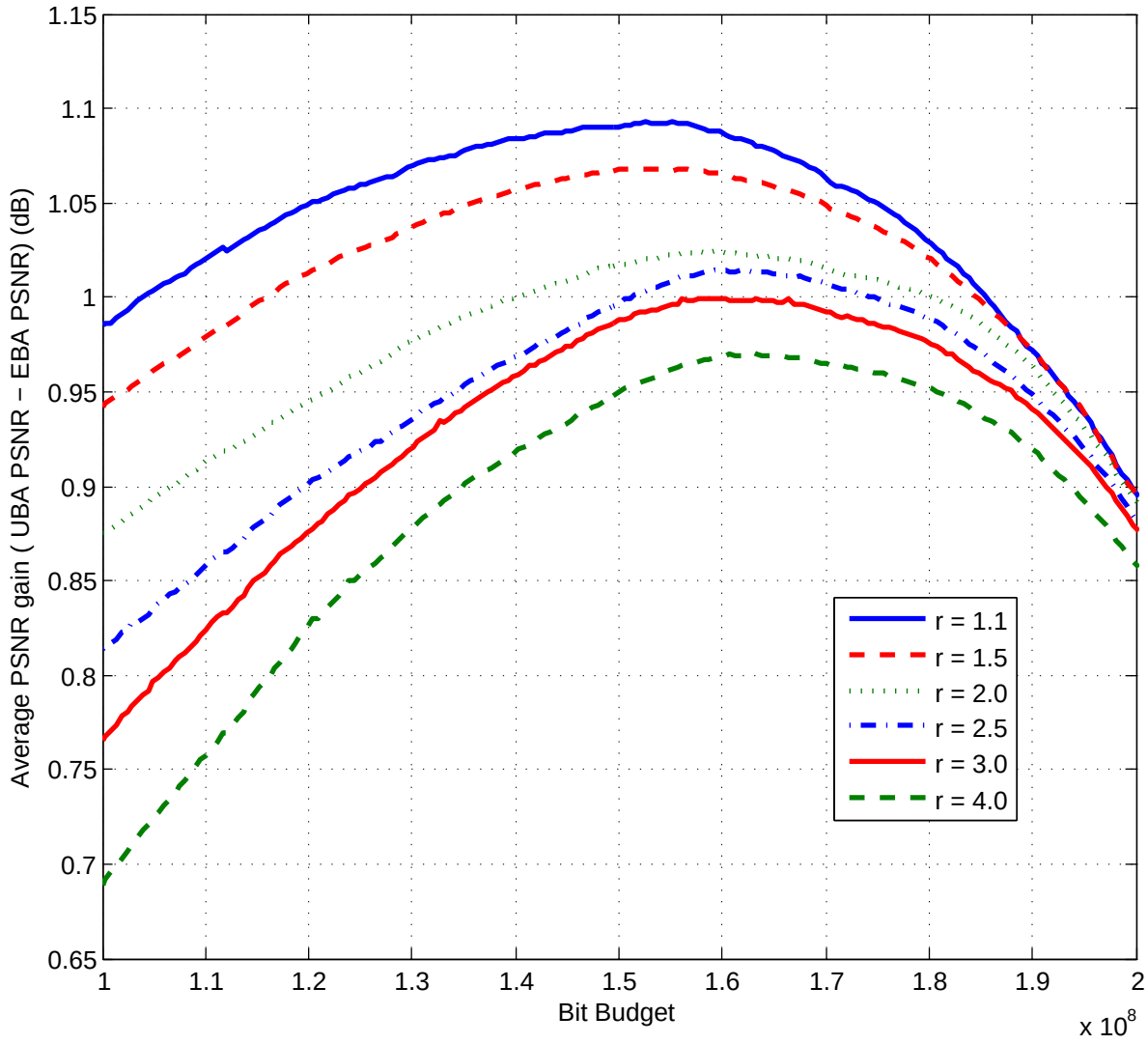

Figure 4.12: The best average PSNR gain with different r. $a = 128$ Bytes

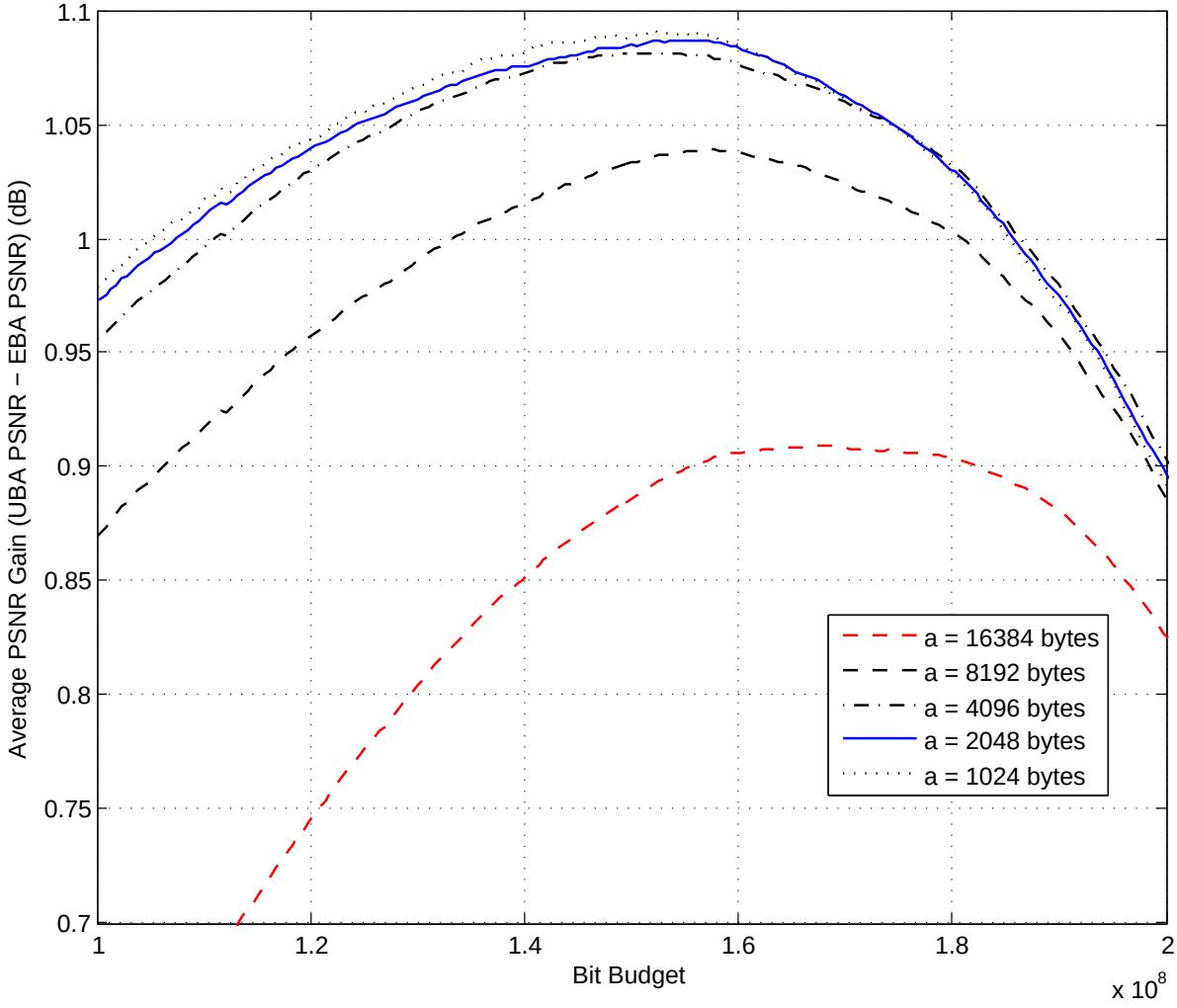

Figure 4.13: The best average PSNR gain with different a. $r = 1.1$

4.3.3 Performance At Low Bit Rate

Demand of bit efficiency and quality optimization is more essential at a relative low bit budget. In this section we take closer look at the performance of the UBA algorithm from both overall and atomic aspects at low bit budget range. We picked up eight low bit budget samples from 0.125 to 1 *bpp* for the analysis. The full list of the bit budget points can be found in Table 4.4.

Again we calculated the average and variance of PSNR gain with a set of parameter combinations at these eight bit budget points. Different from Table 4.2 which focuses on the highest achievable gain value but not the location of the occurrence, this time we obtain the gains at particular bit budgets and keep them in Table 4.3. We noticed that at lower bit budget points, impact of r is more obvious. Apparently large a also significantly reduce the resolution at low bit rate, performance drops quickly in the low bit budget range when a is large. It drops under 0 dB when average bit budget per image is lower than a, so large a must be prevented for low bit budget. Once a is small enough, it has very little impact on the performance. With suitable a, parameter r plays the most important role for low bit budget range performance. But once $r < 1.5$, average PSNR gain becomes stable and shows the characteristics of convergence. From application perspective, this is a useful attribute since algorithm does not need to carefully tune parameters for the very minor performance difference and complexity of the implementation will be reduced.

Figure 4.14 and 4.16 again show the performance curve change with respect to configuration changes at the low bit budget region. Normally, once a set of parameters produces better performance than the other sets of parameters, it has better performance in almost the entire constraint range. Global consistency of the performance simplifies the analysis. We noticed that in Table 4.3 and Figure 4.16 some measurements have precisely zero gain. This is because at the selected bit budget points, in average each image has bit budget that equals to the a value of this particular measurement. Since the first bit block of an image has higher bit efficiency than any of the second bit block of all other images with very high possibility especially when a is large, the first bit block of every image will be chosen at that bit budget, consequently the UBA is exactly the same as EBA. In Figure 4.14, we also noticed that the performance evolving is not very smooth with

Table 4.3: Performance impact at low bit budget (Average)

a (bytes)	r	Bytes per image							
		4096	8192	12288	16384	20480	24576	28672	32768
16	1.1	0.4212	0.5509	0.6814	0.7959	0.8967	0.9778	1.0331	1.0705
16	1.2	0.4126	0.5417	0.6728	0.7871	0.8890	0.9716	1.0269	1.0674
16	1.3	0.4039	0.5330	0.6621	0.7726	0.8760	0.9590	1.0148	1.0573
16	1.4	0.3938	0.5228	0.6508	0.7633	0.8658	0.9458	1.0033	1.0437
16	1.5	0.3863	0.5121	0.6414	0.7513	0.8569	0.9383	0.9982	1.0407
16	2.0	0.3384	0.4671	0.5908	0.6960	0.7892	0.8707	0.9268	0.9793
16	2.5	0.3007	0.4147	0.5386	0.6313	0.7284	0.8297	0.9087	0.9686
16	3.0	0.2558	0.3524	0.5073	0.5805	0.6504	0.7364	0.8217	0.8988
16	4.0	0.1962	0.2038	0.3985	0.5375	0.6193	0.6830	0.7480	0.8185
32	1.1	0.4214	0.5512	0.6811	0.7952	0.8979	0.9794	1.0336	1.0700
64	1.1	0.4204	0.5504	0.6800	0.7943	0.8974	0.9799	1.0339	1.0717
128	1.1	0.4181	0.5498	0.6795	0.7937	0.8970	0.9795	1.0335	1.0715
256	1.1	0.4143	0.5474	0.6766	0.7926	0.8961	0.9795	1.0329	1.0713
512	1.1	0.4070	0.5437	0.6746	0.7908	0.8938	0.9778	1.0320	1.0705
1024	1.0	0.3941	0.5349	0.6649	0.7770	0.8740	0.9512	1.0041	1.0452
1024	1.1	0.3861	0.5342	0.6684	0.7857	0.8886	0.9743	1.0303	1.0679
2048	1.0	0.3543	0.5286	0.6685	0.7864	0.8870	0.9707	1.0270	1.0634
2048	1.1	0.3451	0.5176	0.6571	0.7785	0.8812	0.9677	1.0246	1.0631
4096	1.0	0	0.4546	0.6259	0.7562	0.8723	0.9616	1.0226	1.0644
4096	1.1	0	0.4436	0.6141	0.7466	0.8633	0.9507	1.0122	1.0579
8192	1.0	-7.1443	0	0.4268	0.6245	0.7669	0.8747	0.9491	1.0051
8192	1.1	-7.1443	0	0.4160	0.6137	0.7538	0.8621	0.9364	0.9938
16384	1.0	-11.784	-7.682	-3.597	0	0.3820	0.5897	0.7188	0.8167
16384	1.1	-11.784	-7.682	-3.597	0	0.3736	0.5767	0.7096	0.8100

high r value. We decomposed an inflection point at 0.25 *dpp* and generated the Figure 4.15. The clear-cut aggregations in the figures with large r indicate that the granularity of bit allocations are impaired at low bit budgets, which is caused by the coarse bit block partitioning and the possible large bias introduced during the concave correction.

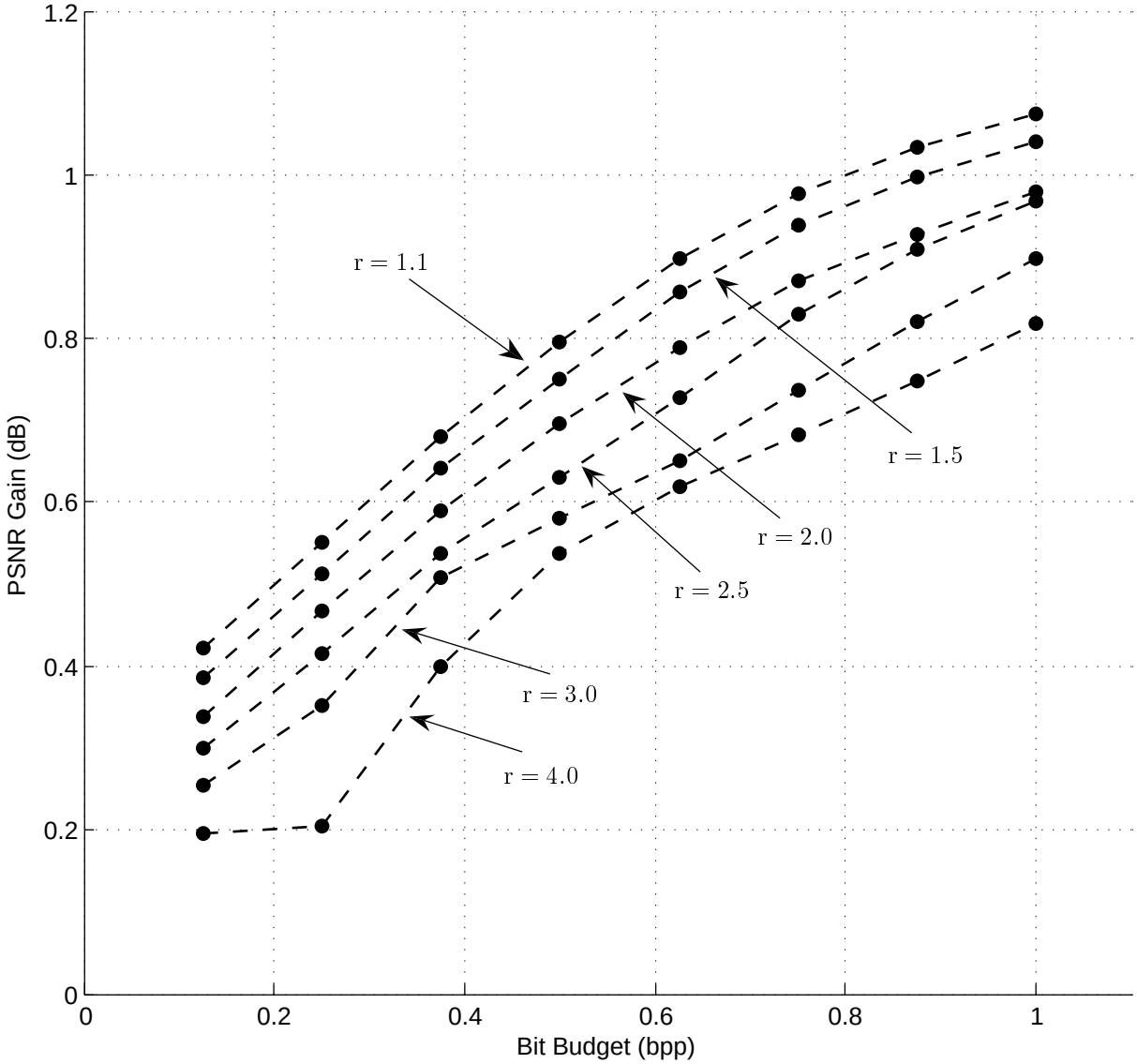

Figure 4.14: Average PSNR gain with different r at low bit budgets. $a = 128$ Bytes

From Figure 4.14 and 4.16, we noticed that similar to the highest performance, smaller a and r that slightly greater than one usually produce better performance at low bit

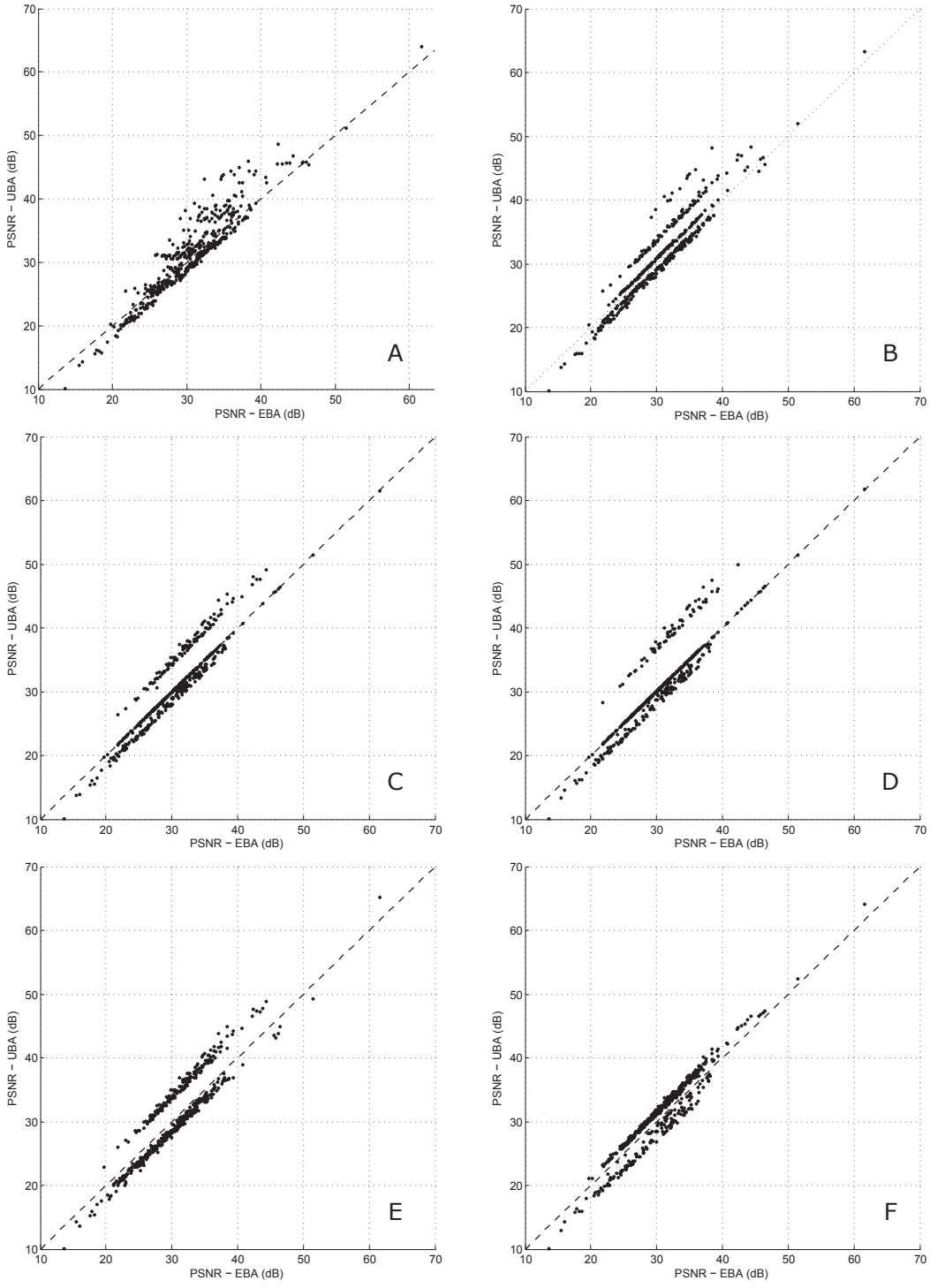

Figure 4.15: Average PSNR gain with different r at low bit budgets. $a = 128$ Bytes; average bit budget: 0.25 *bpp*. r: A: 1.1; B: 1.5; C:2.0; D: 2.5; E:3.0; F:4.0

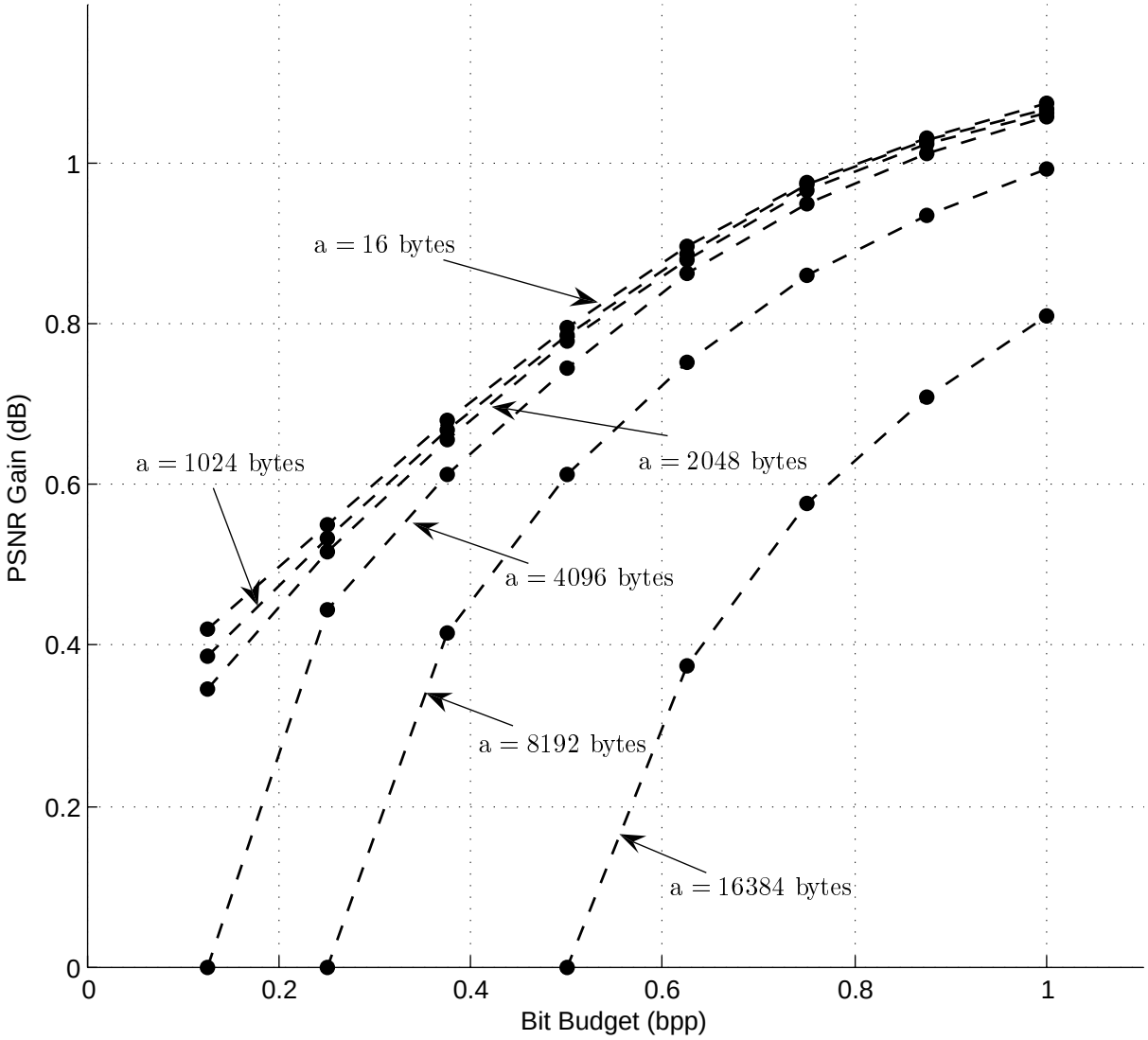

Figure 4.16: PSNR gain with different a at low bit budgets. $r = 1.1$

budget. But in general the curves of different parameters keep in parallel. Based on the results we take a and r value at 128 bytes and 1.1 respectively as reference to observe the performance evolving along with the bit budget growth. This set of parameters provided nearly the highest gain at the selected bit budget points and also achieved the highest possible PSNR average gain in the whole bit budget range. We list the average and variance of the PSNR gain at these bit budget points in Table 4.4. As expected, when the average PSNR gain increases with the growth of bit budget, the gain variance becomes greater and greater. But cases of extreme high gain fell back after mid level bit budget, the degree of imbalance becomes less significant.

Table 4.4: Performance at low bit rates, $a = 128$ bytes, $r = 1.1$

Bit budget (bpp)	Bytes per image	Average PSNR gain (dB)	Variance of gain	Max gain (dB)	Max lost (dB)
0.125	4096	0.42	3.84	13.80	2.11
0.25	8192	0.55	5.32	10.73	3.53
0.375	12288	0.68	8.97	24.41	2.68
0.5	16384	0.79	11.35	22.94	3.45
0.625	20480	0.90	14.13	20.72	3.33
0.75	24576	0.98	15.67	18.29	3.47
0.875	28672	1.03	16.57	14.93	4.19
1.000	32768	1.07	17.41	13.69	4.19

Table 4.4 shows results of set wise average and extreme cases. We further plot Figure 4.17 and 4.18 to provide single image wise quality dynamic at these eight low bit budget points. From 0.125 *bpp* to 1 *bpp*, the amplitude of gains scattered to a wider and wider range. We can see that some images have higher priority in lower bit budget but later lose priority and then stay on the same quality level on the rest of the bit budget points. Overall from the shape of the scatters on these eight bit budget points the trend of gain distribution is fairly consistent. It implies that the performance with respect to the individual image is stable.

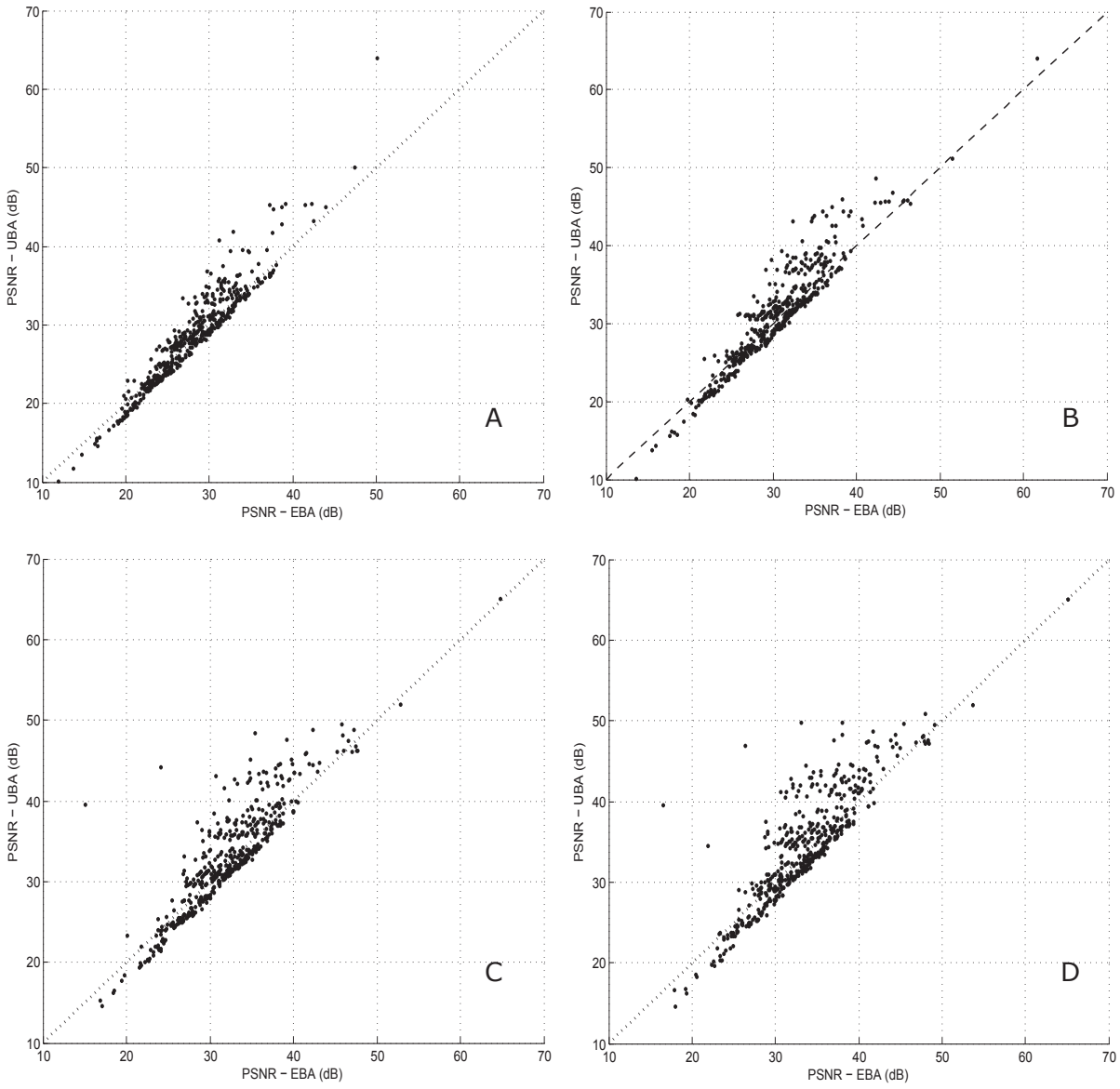

Figure 4.17: Single image PSNR contrasts at low bit budgets: A: 0.125 *bpp*; B: 0.25 *bpp*; C: 0.375 *bpp*; D: 0.5 *bpp*;

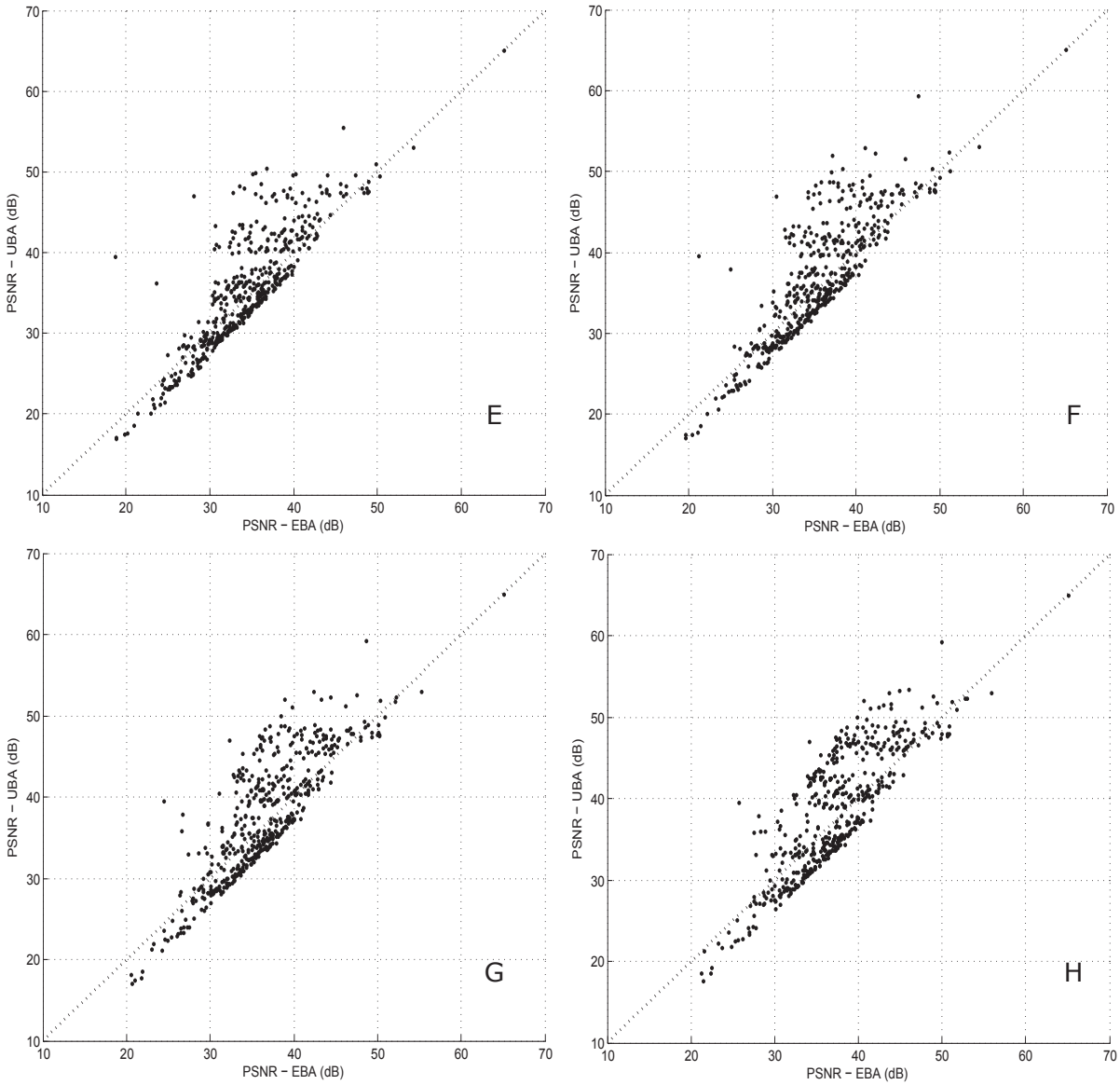

Figure 4.18: Single image PSNR contrasts at low bit budgets. E: 0.625 *bpp*; F: 0.75 *bpp*; G: 0.875 *bpp*; H: 1.0 *bpp*;

4.3.4 Image Selection and Size of Image Set

Image set size naturally is a key performance impact factor. Due to the limitation of the number of image samples, the experiments in this section cannot fairly reflect the diverse of the image set since we are taken images from the same image set. The efficiency of discussion is limited only on a preliminary guidance level.

To simulate the randomness of image selection from the given image set, we introduced a variable p to indicate the probability that an image to be selected. Independently applying the same probability p to each one of the 500 images will result in a subset of the image set. By controlling the value of p, we can control the average size of the image set. We measured the maximum achievable PSNR gain with different value of p. For each given p, we repeated the image selection procedure and performed a sufficient number of experiments to ensure the confidence level is high enough to reflect the randomness. The results of 50 samples are shown in Table 4.5. Clearly when p is small, the maximum average PSNR gain, even though the images are chosen from the same image set, has very high variance. But in average, the highest gain remains on a relative stable level.

Table 4.5: Maximum PSNR gain with different p

Probability of choosing an image (p)	Highest achievable average gain	Lowest average gain	Mean of average gain	Variance of gain
0.02	2.7432	0.2473	1.0281	0.1284
0.10	1.3597	0.6743	1.0623	0.0216
0.20	1.3271	0.8692	1.1031	0.0122
0.40	1.3280	0.9576	1.0979	0.0062
0.60	1.1698	1.0176	1.0914	0.0017
0.80	1.1417	1.0570	1.0870	0.0005

We applied the similar simulation to the lower bit budgets. Similar behavior was observed as shown in Figure 4.19. Each spot in the figure represents a test experiment. With the growth of p, average PSNR gain has little change but the variance of samples are smaller and smaller and eventually centered at the average value. We listed the average PSNR gain over samples in Table 4.6. Clearly higher p has results approaching the full image set performance as we have shown in Table 4.4. This in fact tells us the image set must not be too small if we expect a relative stable quality gain.

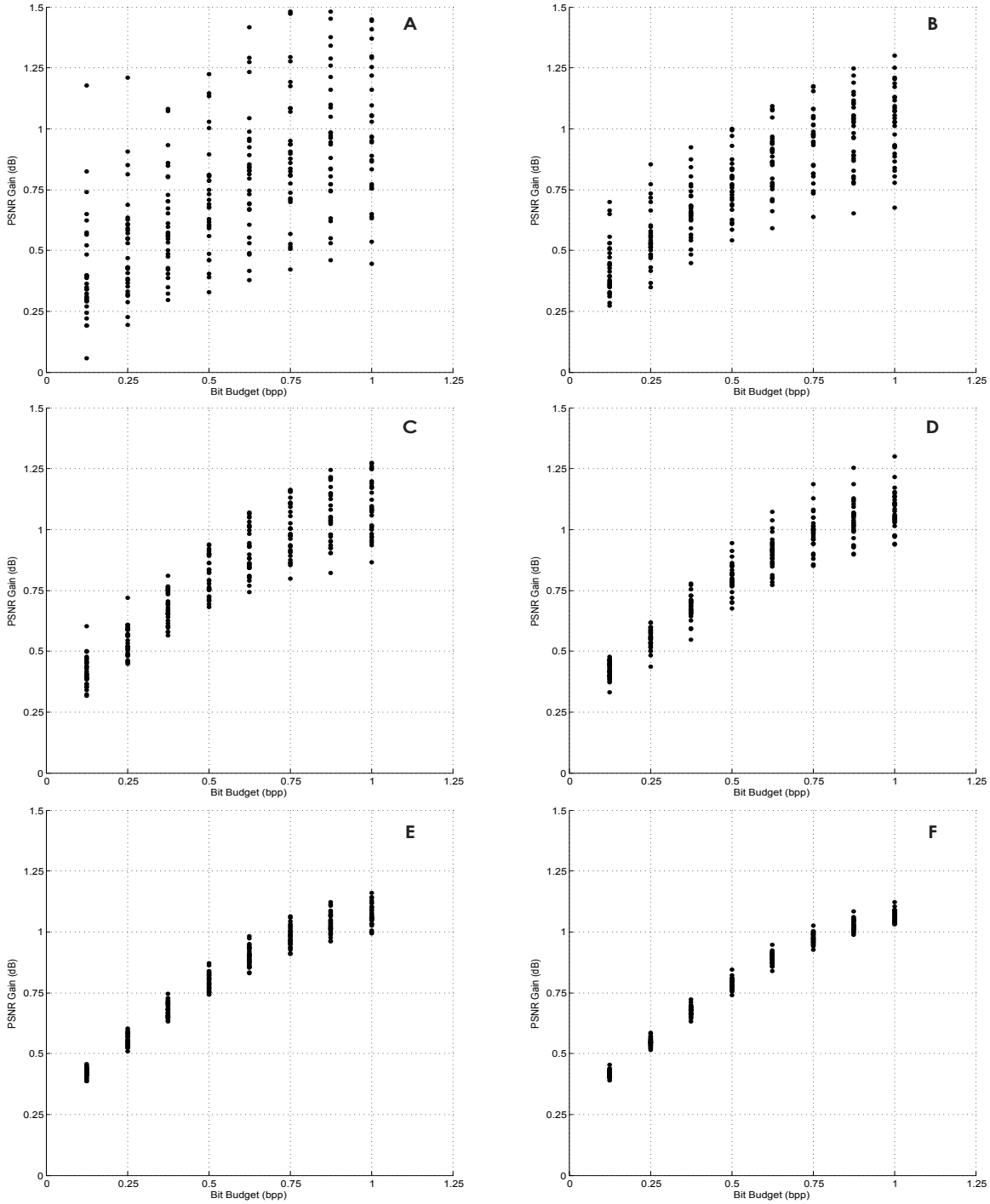

Figure 4.19: Variance of average PSNR gain with different size of image set. A: $p = 0.02$; B: $p = 0.1$; C: $p = 0.2$; D: $p = 0.4$; E: $p = 0.6$; F: $p = 0.8$. Number of samples: 30

Table 4.6: Average PSNR gain with different p at low bit budgets

Probability of p choosing an image (p)	Average budget per image (bytes)							
	4096	8192	12288	16384	20480	24576	28672	32768
0.02	0.3736	0.5043	0.6167	0.7207	0.8227	0.9043	0.9582	0.9680
0.10	0.4314	0.5543	0.6679	0.7704	0.8650	0.9374	0.9867	1.0059
0.20	0.4149	0.5439	0.6800	0.8058	0.9126	0.9981	1.0502	1.0966
0.40	0.4221	0.5523	0.6825	0.8010	0.9023	0.9833	1.0373	1.0826
0.60	0.4228	0.5536	0.6813	0.7948	0.8981	0.9797	1.0348	1.0769
0.80	0.4186	0.5476	0.6749	0.7886	0.8918	0.9736	1.0285	1.0736

It also proves from another aspect that image selection is one of the most important factors that control the possible performance gain for an image set. In another word, once image set is determined, the converged performance is determined. It provides another possibility of understanding the characteristics of the image set.

4.4 Summary

In this chapter we have applied the proposed SPIHT based bit allocation optimization plan (UBA) to a 500 image set and analyzed the performance. In order to simplify the simulation, the rate-PSNR curve of each image was in fact represented by a dense sampled measurement set. We partition the bit stream of each image into bit blocks following the model in (4.3) and enforce all images using the same model.

After a simple two image study to understand the optimization behavior, three groups of simulations were performed and the results of UBA outperform equal bit allocation (EBA) plan on a few aspects. We first performed the exhaustive searching through medium to high bit budget range with different parameter sets and found that the achievable average PSNR gain is over 1 dB. At the most interested low bit budget, we found the new method has the capability of adapting various constraints while still providing steady improvement with appropriate parameter set. In fact the performance is not very sensitive to the parameter change as long as the number of bit blocks is sufficiently high. In third group of simulation, we applied the optimization procedure to image sets with various sizes and combinations. We noticed that the size of image set highly impacts the

stability at low bit budget range and the content of images caps the maximum achievable performance. Although the performance is bit budget and image dependent, algorithm itself is easy to adapt a wide bit budget range and is insensitive to the selection of images. Overall the proposed algorithm exhibits good performance and attractive characteristics.

Meanwhile we analyzed and discussed the side effect of the proposed UBA method. One observation is the unequal allocation increases the variance of PSNR. In a negative situation, some simple images (the images that have less visual details and are easy to achieve high compression efficiency) unnecessarily gain extra quality while some complex images (the images with rich details) further lose essential information. This drawback is built in and inevitably.

Chapter 5

Conclusion

In the thesis, we proposed a new method to maximize overall quality of an image archive under the given stringent bit budget constraints. Different from regular attempts for image archiving compression, which usually focus on correlation of the image set under determined resource limitations, the new solution intends to take advantage of the characteristics of wavelet transformation based compression algorithms such as SPIHT to optimize the bit allocation among images.

In the new method, the optimization problem is solved as linear program with concave correction on the rate-PSNR curve. We partition each compressed bit stream into bit blocks and characterize rate-PSNR behavior on each bit block by its average PSNR gain per bit. The optimization procedure allocates the bits to the images that provide the highest PSNR contribution to the images set so that the overall quality is maximized under the bit budget limitation. Quadratic-programming is used to ensure that the concave condition is met for every image.

Experimental results showed that the new method outperforms the equal bit allocation in different perspectives. We illustrated that in average a more than 1 dB quality gain in terms of PSNR is achieved for a not so large image set. Accordingly, the new method can save as much as 15% bit budget to achieve the same overall quality comparing to its counter-part. In a wide range of bit budget constraints, the algorithm provides steady performance with appropriate selected partitioning parameters. Moreover, the

new method is very simple and suitable for a large set of wavelet transform based compression algorithms as well as is adaptive to arbitrary image and arbitrary constraints.

Implementation of the new method in reality has to deal with certain built in side effects. The variance of PSNR becomes larger and shows bi-modal like distribution which may downgrade the quality of some images to an intolerable level. Correspondingly minimal quality constraints need to be placed in the real application scenario. Note that in our experiments, bit block partitioning is uniform for all images. In reality, each image may have its own partitioning that takes the region of interest or quality level into account. Or alternatively, bit blocks may use pre-defined priority levels to mix with absolute PSNR contribution rate to balance the individual quality and overall quality.

The implementation detail was merely discussed on a very limited level. For example, the overhead information included was mentioned but not formulated into concrete form. Further analysis is required to evaluate the impact to the overall performance. In addition, both concave correction and linear programming introduce bias. The bias more or less reduced the efficiency of the algorithm. Future work need to be done to analyze the level of bias and compensate accordingly.

PSNR is simple, has clear physical meanings, and is convenient in the context of optimization. But it is not very well matched to perceived visual quality. Alternatively, structural similarity based quality metrics [30] is more attractive to adapt to human visual system. Creating objective function based on the new category of quality metrics will be another interesting direction in the future.

The complexity of the algorithm due to the high dimension of the optimization program was not studied in depth. The number of optimization constraints increase significantly with finer bit block partitioning and larger number of images included so the time cost of optimization procedure increases accordingly. It is not severe problem if optimization is a pre-processing procedure. However, usually the optimization need to adapt the bit budget during the run time and the time cost of optimization procedure is no longer negligible. For real time application, this must be carefully considered.

Bibliography

[1] R. M. Mersereau and T. C. Speake, "A unified treatment of Cooley-Tukey algorithms for the evaluation of the multidimensional DFT," *IEEE Transactions On Acoustics, Speech, and Signal Processing*, vol. 29, no. 5, pp. 1011–1018, 1981.

[2] N. Jayant and P.Noll, *Digital Coding of Waveforms*. Prentice Hall, 1984.

[3] N. Ahmed, T. Natarajan, and K. R. Rao, "Discrete cosine transform," *IEEE Transactions On Computers*, pp. 90–93, 1974.

[4] R. J. Clarke, *Transform coding of images*. Academic Press, 1985.

[5] S. G. Mallat, "A theory for multiresolution signal decomposition: The wavelet representation," *IEEE Transactions On Pattern Analysis And Machine Intelligence*, vol. 11, no. 7, pp. 674–693, 1989.

[6] M. Antonini, M. Barlaud, P. Mathieu, and I. Daubechies, "Image coding using wavelet transform," *IEEE Transactions On Signal Processing*, vol. 1, no. 2, pp. 205–220, 1992.

[7] Z. Xiong, I. Ramchandran, M. T. Orchard, and Y.-Q. Zhang, "A comparative study of DCT and wavelet-based image coding," *IEEE Transactions On Circuits And Systems For Video Technology*, vol. 9, no. 5, pp. 692–695, 1999.

[8] J. M. Shapiro, "Embedded image coding using zerotrees of wavelet coefficients," *IEEE Transactions On Signal Processing*, vol. 41, no. 12, pp. 3445–3462, 1993.

[9] A. Said and W. A. Pearlman, "A new, fast, and efficient image codec based on set partitioning in hierarchcal tree," *IEEE Transactions On Circuits and Systems For Video Technology*, vol. 6, no. 3, pp. 243–250, 1996.

[10] W.-H. Chen and W. K. Pratt, "Scene adaptive coder," *IEEE Transactions On Communications*, vol. 32, no. 3, pp. 225–232, 1984.

[11] R. Hamzaoui, V. Stankovic, and Z. Xiong, "Optimized error protection of scalable image bit streams," *IEEE Signal Processing Magazine*, pp. 91–107, 2005.

[12] A. Ortega and K. Ramchandran, "Rate-distortion methods for image and video compression," *IEEE Signal Processing Magzine*, pp. 23–50, 1998.

[13] A. M. Eskicioglu and P. S. . Fisher, "Image quality measures and their performance," *IEEE Transactions On Communications*, vol. 43, no. 12, pp. 2959–2965, 1995.

[14] Z. Wang and A. C. Bovik, "A universal image quality index," *IEEE Signal Processing Letters*, vol. 9, no. 3, pp. 81–84, 2002.

[15] K. Kipli, S. Krishnan, N. Zamhari, M. S. Muhammad, S. W. Masra, K. L. Chin, and K. Lias, "Full reference image quality metrics and their performance," *IEEE 7th International Colloquium on Signal Processing and its Applications*, pp. 33–38, 2011.

[16] N. Sarshar and X. Wu, "On rate-distortion models for natural images and wavelet coding performance," *IEEE Transactions On Image Processing*, vol. 16, no. 5, pp. 1383–1394, 2007.

[17] G. B. Dantzig and M. N. Thapa, *Linear Programming 1: Introduction*. Springer-Verlag, 1997.

[18] G. B. Dantzig, *Linear Programming and Extensions*. Princeton Umiversity Press, 1963.

[19] J. Nocedal and S. J. Wright, *Numerical Optimization*. Springer-Verlag, 2006.

[20] J. More and D. Sorensen, "Computing a trust region step," *SIAM Journal on Scientific and Statistical Computing*, vol. 3, pp. 553–572, 1983.

[21] P. E. Gill, W. Murray, and M. H. Wright, *Practical Optimization*. Emerald Group, 1982.

[22] G. B. Dantzig and M. N. Thapa, *Linear Programming 2: Theory and Extensions*. Springer-Verlag, 2003.

[23] T. F. Coleman and Y. Li, "A reflective newton method for minimizing a quadratic function subject to bounds on some of the variables," *SIAM Journal on Optimization*, vol. 6, no. 4, pp. 1040–1058, 1996.

[24] N. Gould and P. L. Toint, "Preprocessing for quadratic programming," *Mathematical Programming*, vol. 100, no. 1, pp. 95–132, 2004.

[25] S. Mehrotra, "On the implementation of a primal-dual interior point method," *SIAM Journal on Optimization*, vol. 2, no. 4, pp. 575–601, 1992.

[26] M. Livny and V. Ratnakar, "Quality-controlled compression of sets of images," *Proceedings of International Workshop on Multimedia Database Management Systems*, vol. 16, no. 5, pp. 109–114, 1996.

[27] V. Kustov, P. Srinivasan, S. Mitra, S. Shishkin, and D. Mehri, "Adaptive wavelet technique for effective storage and fast internet transmission of medical images," *13th IEEE Symposium on Computer-Based Medical Systems*, pp. 221–226, 2000.

[28] J.-D. Lee, S.-Y. Wan, C.-M. Ma, and R.-F. Wu, "Compressing sets of similar images using hybrid compression model," *IEEE International Conference on Multimedia and Expo Proceedings.*, vol. 1, pp. 617–620, 2002.

[29] A. Segall, "Bit allocation and encoding for vector sources," *IEEE Transactions On Information Theory*, vol. IT-22, no. 2, pp. 162–169, 1976.

[30] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, "Image quality assessment: From error visibility to structural similarity," *IEEE Transactions On Image Processing*, vol. 13, no. 4, pp. 600–612, 1976.