QUALITY-CONTROLLED LOSSY IMAGE COMPRESSION

By

Viresh Ratnakar

A dissertation submitted in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

(Computer Science)

at the

UNIVERSITY OF WISCONSIN – MADISON

1997

Abstract

Digital images are occupying a rapidly increasing amount of storage and bandwidth in today's computing and communication environments. This thesis presents the QCLIC (Quality-Controlled Lossy Image Compression) framework as a fundamental way of using lossy image compression to meet application needs while satisfying storage and bandwidth constraints.

With lossy compression of images, there is a tradeoff between the amount of compression and the quality of the resulting image. This tradeoff depends upon the image being compressed, the compression method, and the choice of certain "tuning parameters" used by the compression method. Naive choices of tuning parameters can result in poor compression-quality tradeoffs. Moreover, choosing tuning parameters to exactly achieve a compression or quality target is often a non-trivial problem. The QCLIC framework enables applications to use a nearly optimal compression-quality tradeoff profile of an image as a fundamental interface for accessing the image. Mapping an application's own needs and constraints to compression/quality targets and constraints is a natural way to access images, as opposed to the lower-level, compression technique-specific interface provided by the tuning parameters themselves. To bridge the gap between the compression-quality profile interface and the tuning parameters, compression technique-specific enabling technologies are needed.

Measuring the quality of a compressed image is an important issue. We survey several compression techniques and quality metrics, and suggest strategies for designing enabling technologies to allow their use in the QCLIC framework. A commonly used quality metric, PSNR, is extended to make it more uniform across images. For JPEG,

the most widely used lossy compression standard at present, there was no efficient enabling technology available, and the RD-OPT algorithm presented in this thesis fills that void. We use the QCLIC framework to efficiently exploit compression-quality tradeoffs across images, when sets of images are compressed together. We also describe some test-bed applications that implement and use the QCLIC framework, to meet the needs of real systems.

Quantization forms the key loss-controlling step in many image compression techniques. The tuning parameters in a compression technique are typically just the parameters used by a quantization strategy. For mid-tread uniform scalar quantization with reconstruction at mid-points of bins (a very commonly used strategy), we analyze and optimize the performance gains achieved by using adaptive zeroing thresholds, when the source has Laplacian density. We show that the performance gains are about the same as those achieved by using source-specific reconstruction levels, without the overheads and added complexity. The RD-OPT algorithm for JPEG optimizes zeroing thresholds too, and the RD-OPT results confirm the utility of this quantization strategy, as predicted by the analysis.

Acknowledgements

Miron Livny showed faith in me when I needed it, gave just the right nudges, and conceded precisely the vast amount of freedom of exploration that I had dreamed of. His vision lies in spotting interlocking patterns in apparently unrelated pieces, in (literally!) knocking on just the right doors that would put a half-muttered idea to excellent use. The whole world *is* a tree rooted at every man's personal Condor, and I am eager to go out in the real world, QCLIC'ing all and sundry. For this and everything else, I am deeply grateful to my advisor.

Prasoon Tiwari, now at IBM T.J. Watson Research Center, got me initiated into "proper" research, initially by sheer dint of accosting me in the department elevator every other week and demanding what I was up to. At just about the time when my romance with theoretical computer science was ebbing, he led me into image compression (via fractal theory). He too showed faith in me when I needed it, and I consider myself lucky to have always found the perfect helping hands.

Another bunch of able helping hands belonged to the infectiously enthusiastic researchers at Xerox PARC, where I spent a summer working on rumor-mongering protocols for achieving consistency in replicated databases. Many thanks to Carl Hauser, Dan Greene, and Al Demers, for giving me a "systems" perspective, which has proved extremely useful.

During my summer internship at T.J. Watson, I acquired the much-needed background in image compression, thanks to frequent, stimulating, and spontaneous discussions with Ephraim Feig, Eric Viscito, Heidi Peterson, and Diego Garrido. In the

Computer Sciences department, here at UW, all kinds of wonderful insights and revelations were provided by Anne Condon, Yannis Ioannidis, Raghu Ramakrishnan, Tom Reps, Debbie Joseph, and every other faculty member I came in contact with. Fellow graduate students and research staff, Jussi Myllymaki, Vishy Poosala, Michael Cheng, Kevin Beyer, Tian Zhang, Kent Wenger, and countless others provided insights into the faculty insights.

I had the best possible trio of roommates ever conceived of. Many thanks to Kambainatham Harinarayan, the often-uncomplaining chai-maker, carom-foe sans compare, and generally a froody gloop altogether. To Prakash Raman, for being there to give me a swift ride at 7:57 AM, in time to rescue my grade-critical talk at 8:04 AM; for countless "no-problem"s and much-appreciated simple "Oh"s. And bada thanks to Ratnakar Sonthi, for the intense, ineffable, and yet completely relaxed exchange of ideas, ranging from the meaninglessly abstruse to the life-alteringly meaningful; for simply understanding. Without these three to counter the often trying slumps rampant in a graduate student's life, my Ph.D. path would have been ploddingly painful.

My family has always given me the strongest possible motivation. Sonali, my wife, has been in Madison with me for the last two years, and I am forever indebted to her for her patience, support, and love, and for simply being a fun-loving and wonderful partner. My brother, Vishal, with whom I have yet another intense and ineffable relationship, will undoubtedly drown himself and the rest of Kharkov in cognac for my Ph.D. I owe a world of thanks to my parents. For the Rubik's cubes and the Manjaris, for their sleepless nights and unspoken dreams, and for their belief in me. Dear Vasudha and Balwant, we made it!

Contents

List of Tables

List of Figures

Chapter 1

Introduction

This thesis presents the concept of *quality-controlled lossy image compression* (QCLIC) as a fundamental framework for efficient management of large amounts of image data. Digital images and video are essential components of today's computing and communication environments. With rapid increases in storage capacity, computational capability, network bandwidth, and the rapid expansion of the World Wide Web, the amount of image-data is only likely to grow. Compared to "traditional" data, digital images use up exorbitantly large amounts of storage space: a single 600×600 color image uses up roughly one megabyte of space, comparable to the space needed for the entire text of a five hundred-page book. The amount of "information" present in this image megabyte is clearly much lesser than that in the book; the image could merely be the cover page of the book in a multimedia document. The disproportionately low "information/bytes" ratio for images (compared to other, traditional components of multimedia) makes storing, moving, and manipulating these bulky data items a tremendous challenge for applications.

Unlike traditional data, it is usually hard to quantify the amount of information in a digital image. Consider a large image that is uniformly gray everywhere. For this image, the information can easily be described in just a few bytes that indicate that "the image has width W, height H, and is uniformly gray at level G everywhere." Now, consider a uniformly gray image with pixel values G everywhere, except at a small number of isolated pixels which have the value $G+2$. For all practical purposes, this image is identical to the previous one, without any additional information: the anomalous pixels are probably just noise associated with the image capture device. On the other hand, if the "anomalous" pixels have some structure, such as being present along the edges of a rectangle, then they probably constitute information.

Lossy compression techniques try to modify and organize the image data to unravel the structure of the information present in the image. Discarding the "noise" part of this structure usually makes the remaining "information" much more amenable to compression. The amount of information retained, or the *quality* of the compressed image, decreases as the image is compressed more and more. Traditionally, the focus of lossy compression has been on compressing images as far as possible while letting the quality of the compressed image remain visually indistinguishable from the original. However, lossy compression offers much more than that. The structured reorganization of image data inherent in lossy compression can be used as a *summarizing tool* for images. An image can be compressed down to an extremely small size, by discarding all but the lowest level of information (such as the average pixel values in 8×8 blocks), and this low-level summary can be used by resource-constrained applications to decide their "interest" in the image. The situation is similar to summaries of books. A summary will give only a rough idea of the contents of a book (sometimes it can even give a *false* idea). But it is an essential tool for readers who want to choose to read only a certain kind of books, among the available multitude. For example, in a vast multimedia image library, remote users will select and retrieve images based on several criteria that could include quickly browsing through low-quality summaries of the images.

Unlike books, images can be summarized to *any* desired level of quality with lossy compression. This enables applications to *choose* the level of image information based on their needs and resource constraints (such as diskspace). Quality can be used as a tuning knob to best satisfy the needs under the constraints. This observation is the driving force for our research. For efficiently using quality as a tuning knob, good *quality metrics* are needed to characterize the information-content of lossy-compressed images. Further, the compression-quality tradeoff needs to be characterized accurately. Finally, efficient *quality-control* techniques are needed to compress images accurately to any desired quality.

1.1 Rate-Quality Tradeoff

The tradeoff between quality and amount of compression is seen from the *rate-quality tradeoff curve* (Figure 1). *Rate* is a convenient, uniform way of expressing the amount

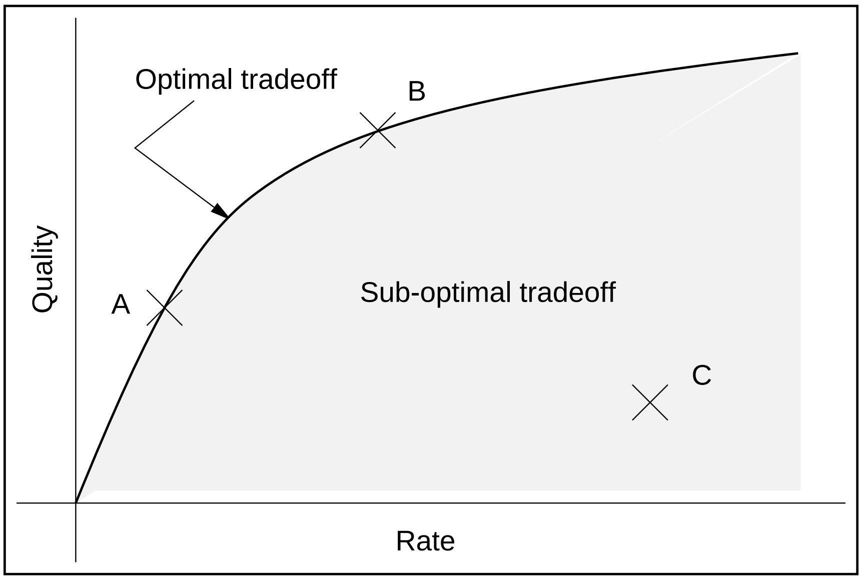


Figure 1: Rate-quality tradeoff associated with lossy image compression.

of compression. Rate is simply the bits used per pixel (bpp) in the compressed image. Thus, an uncompressed color image with each color plane (red, green, and blue) stored with 1-byte precision has a rate of 24 bpp.

The notion of quality of a compressed image with respect to the original is intuitively obvious. In practice, there is no universal metric that can adequately capture this intuitively obvious concept. Several useful approximations exist, and we shall discuss them in some detail. For the purpose of illustrating the rate-quality tradeoff, we will refer to *quality* without specifying an actual metric, and assume it to be a suitable measure of the information-content of a lossy-compressed image.

The rate-quality tradeoff is determined by four factors: compression method, image characteristics, quality metric, and tuning parameters. For a given image, compression method, and quality metric, the rate and quality of compression depend on certain parameters passed to the compression algorithm, which we refer to as the "tuning parameters." For example, for JPEG compression [JPG], these tuning parameters are the quantization tables that determine how coarsely each spatial frequency is stored. The rate-quality tradeoff resulting from a particular choice of parameters may not be the best possible. Figure 1 shows how the rate-quality space consists of three areas. The dark curve shows the optimal rate-quality tradeoff points. For a given rate, the corresponding point on this curve is the best quality that can be achieved at that rate, by varying the tuning parameters. The area above the optimal curve cannot be realized by any choice of tuning parameters. The shaded area under the optimal curve is the sub-optimal tradeoff area. A poor choice of tuning parameters results in a rate-quality point in the sub-optimal area. The rate-quality points A, B, and C shown in Figure 1 are illustrated in Figure 2, for JPEG compression. Point C corresponds to a very poor choice of the tuning parameters, and illustrates the need to achieve tradeoffs close to the optimal curve.

Some general trends are observed in the shape of typical rate-quality curves. At extremely low rates, the quality is very poor. The quality increases sharply for a while as the rate is increased, and then levels off to a slower rise. After a certain rate the compressed image gradually becomes virtually indistinguishable from the original, to the human eye.

From an information theory perspective, the sharp increase in quality lasts while information about the rough, overall structure of the image is being encoded (which has low entropy), and the leveling-off occurs when the compressor starts encoding the fine detail and random noise associated with pixel intensities (which has high entropy).

1.1.1 Quality-Control

Applications that discard the original (uncompressed) image because of storage constraints try to compress images as much as possible, without making the quality-loss visually detectable. Typically, this point occurs around the transition between the sharp-rise and slow-rise sections of the rate-quality curve. This motivates the usefulness of the following paradigm for image compression: when a user wishes to compress an image, she is presented with the rate-quality curve. The user specifies a compression target by choosing a point on the curve. The compressor then presents the image exactly meeting the specified target, using the appropriate tuning parameters. The user can further refine her selection by interacting with this rate-quality curve-based presentation of the image. This simple and fundamental *quality-control* operation on images is typically missing in image compression systems, and forms the basic idea of

A
Compression with nearly optimal
rate-quality tradeoff
Rate = 0.22 bpp (Size: 7.2 Kbytes)

B
Compression with nearly optimal
rate-quality tradeoff
Rate = 1.0 bpp (Size: 32.8 Kbytes)

C
Compression with extremely
sub-optimal rate-quality tradeoff
Rate = 3.8 bpp (Size: 124.5 Kbytes)

Figure 2: Optimal and sub-optimal rate-quality tradeoff illustrated.

our thesis.

Quality-control does not necessarily imply choosing an image quality that is visually indistinguishable from the original. The idea is general—the rate/quality targets specified can be arbitrary. Consider an image-browser application. Suppose the browser can decode images compressed using method $\mathcal{C}$, and uses a quality metric $\mathcal{Q}$. Such a browser will make requests of the following nature, to a remote image server:

> Send me the image "foo" using compression method $\mathcal{C}$ such that the quality, using metric $\mathcal{Q}$, is exactly Q_B.

Initially, the user will set a low value for the browse quality Q_B, and then for images of interest, she will repeat the call with higher Q_B. The image server may have pre-compressed the image "foo" to a number of quality levels (with progressive compression techniques a single compressed image can be pruned to achieve this), or may compress-on-demand. Exactly how the server implements this basic QCLIC functionality would depend upon factors such as available computational and storage resources, response-time constraints, network speed, and compression method.

Digital images may also be intended for sophisticated analyses done by computers, apart from and along with human viewing. Often, the errors in the analysis can be bounded in terms of the quality (using a convenient, application-specific metric) of the lossy compressed image. Consider, for a simplified example, classification software that classifies multispectral image pixels into three classes, A, B, and C. Suppose the task is to find those images out of a large set that have at least 5% of class C pixels, and it can be shown that for lossy compressed images with quality $\geq Q$, the classification error in each class is at most 1%. Then the classifier can be used on compressed images retrieved at quality Q, and only the images with $\geq 4\%$ class C pixels need be considered for full analysis. This represents a substantial improvement in efficiency, especially if the images are retrieved over a slow network.

Thus, our notion of quality-control is the following ability: for a given image, a given compression method, and a given quality metric, compress the image precisely to a given rate target or a given quality target.

The mapping between compression parameters and resulting rate and quality may not be obvious. If an image I is compressed using a compressor $\mathcal{C}$ that takes tuning parameter θ, then the compressed image $\mathcal{C}(I, \theta)$ has a certain rate and a certain quality, determined by θ. Figuring out a θ for a given rate or quality target is often a hard problem. Thus, to do quality-control, we need *enabling technologies* specific to image compression methods that can meet the following demands:

1. Given a rate or quality target, find θ such that $\mathcal{C}(I, \theta)$ meets the target precisely.

2. Make sure that the resulting rate-quality tradeoff is close to optimal.

For some compression methods, these two objectives are hard to achieve (such as with JPEG), whereas for others they are easier (such as with common wavelet-based compression schemes [Sha93, SP96a, ZASB95]).

1.2 The QCLIC-Image object

A *QCLIC-Image* is an object that supports two basic methods:

1. *QCLIC-GetCurve*($\mathcal{C}$, $\mathcal{Q}$, *curve-constraints*), and

2. *QCLIC-Compress*($\mathcal{C}$, $\mathcal{Q}$, *target*, *compression-constraints*).

Here, $\mathcal{C}$ specifies a compression method, and $\mathcal{Q}$ is a quality metric. *QCLIC-GetCurve* returns a rate-quality curve for the image. The parameter *curve-constraints* specifies restrictions such as minimum/maximum rate/quality, number of curve points needed, etc. The curve would be returned as a sequence of (rate, quality) pairs. A good implementation needs to also ensure that the curve returned is optimal or nearly optimal.

The method *QCLIC-Compress* returns compressed images according to the specified target and constraints. *QCLIC-Compress*($\mathcal{C}$, $\mathcal{Q}$, *target*, *compression-constraints*) returns a compressed image using compression method $\mathcal{C}$ and quality metric $\mathcal{Q}$. The parameter *target* specifies either a rate target, or a quality target. The parameter *compression-constraints* can specify several constraints, such as a tolerance allowed in achieving *target*. For selective retrieval, *compression-constraints* can also specify a particular *piece* of the compressed image to be returned. For example, for using progressive transmission, the first call can demand just the first 1000 bytes of the compressed image—subsequent calls will demand the remaining bytes progressively. Once again, a good implementation will try to optimize the rate-quality tradeoff in choosing the tuning parameters.

Specific application-based implementations of the QCLIC framework need only support a limited number of possibilities for the parameters $\mathcal{C}$, $\mathcal{Q}$, *curve-constraints*, and *compression-constraints*, depending on their needs and available resources. For example, the QCLIC framework also supports retrieval of the original image, with lossless or no compression. Specific implementations may choose not to support this functionality, if they do not have the storage resources and/or do not need the lossless image. Typically, scientific applications (where each pixel intensity is potentially meaningful data) such as medical imagery require the lossless image to be retained.

1.2.1 QCLICS: QCLIC-Image Sets

There are several applications where *sets* of images need to be compressed together. Using the QCLIC framework, and aggregate quality metrics for sets of images, we can also optimize rate-quality tradeoffs *across* several images, and provide quality-control. The rate-quality curve and related concepts extend naturally to sets of images using aggregate quality metrics such as "average quality." As a simple example, consider the multimedia application of packing a large number of images on a CD-ROM. The total available space must be distributed over the images so as to maximize the set quality.

At a conceptual level, a set of QCLIC-Images is also an object very similar to the QCLIC-Image object. It supports the methods:

1. *QCLICS-GetCurve*($\mathcal{C}$, $\mathcal{Q}_S$, *curve-constraints*), and

2. *QCLICS-Compress*($\mathcal{C}$, $\mathcal{Q}_S$, *target*, *compression-constraints*).

These have exactly the same form as those for a single QCLIC-Image. The parameter $\mathcal{C}$ specifies a compression method, and $\mathcal{Q}_S$ is a quality metric for sets of images, defined in terms of an underlying quality metric $\mathcal{Q}$ for individual images. The parameters *curve-constraints* and *compression-constraints* can specify (in addition to the constraints discussed for QCLIC-Image),

1. Minimum and maximum constraints on rate and quality for some or all of the individual images,

2. Weights for the rates of individual images (quality weights are included in the quality metric $\mathcal{Q}_S$), and

3. Selection and interleaving constraints, which specify which pieces of the compressed images are to be returned, and how they are to be interleaved (this

applies only to *QCLICS-Compress*).

We will show how to build the QCLICS functionality using the methods of QCLIC-Image, and discuss some useful quality metrics for sets of images. Note that the set may be a dynamically constructed subset of a large number of images, as in the following example. Consider an image database server application. Remote clients can make queries such as:

> Find all images of potential airforce bases in this geographical area.

The server processes the query using meta-data and/or content-based searching. The result is a set of images that is to be shipped to the client, over an available bandwidth, with a response-time constraint. The server can use QCLICS to figure out how much each image is to be compressed, using an overall quality metric. If the server simply divides the available bandwidth equally among the images, easy-to-compress images might hog too much bandwidth while some hard-to-compress images might be delivered at extremely poor quality.

Another application of QCLICS is for digital cameras. A digital camera needs to store the captured images in a limited amount of on-device memory. Typically, vendors like to promise the consumer that the camera can hold a certain number of images (say 24) before they are downloaded to his home computer. Like the previous application, here too it is inappropriate to simple divide the available memory equally over the 24 images. On the other hand, if 23 images have been shot, and the 24th happens to be a hard-to-compress image, some quality must be "chipped off" the existing images to create more room for the last image. The QCLICS methods can be used for both partitioning the currently available memory, as well as chipping off to create more room, so that the overall quality is maximized.

1.3 Methodology and Thesis Organization

The functionality of the QCLIC-Image object may not be directly available for the compression method and quality metric used. To realize the QCLIC framework, enabling technologies are needed for specific compression methods and specific quality metrics. Our methodology was to consider specific image compression techniques and devise technologies to allow their use in the QCLIC framework, for a variety of quality metrics. Several applications were implemented to serve real needs and to test and evaluate the QCLIC framework. The image data used in our tests included the standard "image-compression test images" as well as a large number of images gathered from various scientists at the University of Wisconsin-Madison.

Chapter 2 presents a review of the theory behind commonly used image compression tools, and surveys several quality metrics and specific compression techniques. In the context of uniform scalar quantization, which is a key tuning step in several compression methods, we present an analysis of entropy-constrained quantization for the Laplacian probability density, which is a good model of the transformed image data used in image compression. We use the analysis to optimize a class of quantizers for the Laplacian pdf.

Chapter 3 presents the QCLIC framework, and the issues involved in realizing it, for a number of image compression methods. For most of these, the requisite enabling technologies are evident and are described in their entirety. Chapter 3 also presents algorithms for implementing the QCLICS functionality for sets of QCLIC-Images. The performance of the QCLIC framework is evaluated from the standpoint of several systems that we have implemented.

For the JPEG image compression standard, which is the most commonly used image compression technique at present, the QCLIC framework is non-trivial to realize. We have developed a comprehensive enabling technology, the RD-OPT algorithm, for achieving the QCLIC functionality with JPEG. RD-OPT forms the subject of Chapter 4. Chapter 5 presents our conclusions along with related work and directions for future research.

Chapter 2

Image Compression: Theory and Techniques

In this chapter we review some common image compression techniques. Digital images can be produced by a variety of sources such as scanners, CCD cameras, magnetic resonance imaging, tomography, etc. Usually, the original ("real world") image is converted from analog to digital by the image-generating device. There is a quality loss associated with this sampling, and it is important to design it carefully [Pra91]. Here, we assume that the sampled rectangular arrays of pixel intensities are the original data, and think of quality of compressed images with respect to this array as the reference. An additional source of digital images is image-creating software, which directly produces the "rectangular array of samples."

2.1 Notation

A digital image I is a $W \times H$ rectangular array of pixel intensities, with width W and height H. The pixel values (intensities) are referred to as $I(i, j)$, where $0 \leq i \leq H - 1$ and $0 \leq j \leq W - 1$. For "grayscale" images, the pixel intensities are integers in some range $[0, M]$. For "color" images, each pixel intensity $I(i, j)$ is a vector in some color space, such as (Red, Green, Blue). We identify the pixel values in a particular color plane (say R, or red) using the notation $I_R(i,j)$. The pixel intensities in each plane are integers in some range $[0, M]$. Typically, $M = 2^m - 1$, so that each color of a pixel can be represented using m bits. The most commonly used precision is $m = 8$ ($M = 255$).

We use the term "compression technique" loosely, to describe general concepts as well as specific methods. We use the term "compression method" to strictly imply a fixed compression algorithm and decompression algorithm. Thus, a compression method $\mathcal{C}$ consists of an encoder (also denoted by $\mathcal{C}$), and a decoder (denoted by $\mathcal{D}_{\mathcal{C}}$). The compressed image produced by a compression method $\mathcal{C}$ from an image I is denoted by $\mathcal{C}(I)$. In addition, $\mathcal{C}$ might require some parameters (the "tuning" parameters), and if they are not evident from the context, we will add them as arguments, as in $\mathcal{C}(I, \theta)$. The *rate* of compression for the compressed image $\mathcal{C}(I)$ is $\frac{|\mathcal{C}(I)|}{WH}$ bpp, where $|\mathcal{C}(I)|$ is the size of $\mathcal{C}(I)$ in bits.

Usually, the decompressed image is uniquely determined by a compressed image, except for variations resulting from floating-point computations. For simplicity, we will often refer to pixels $I'(i,j)$ of a compressed image $I' = \mathcal{C}(I)$, when strictly speaking, we should use $I'' = \mathcal{D}_{\mathcal{C}}(\mathcal{C}(I))$ instead of I', where $\mathcal{D}_{\mathcal{C}}$ is the decompression algorithm.

2.2 Quality Metrics

As described in Chapter 1, the rate-quality curve for lossy compression depends upon four factors: compression method, image characteristics, quality metric, and tuning parameters. This section describes some commonly used quality metrics.

We identify three uses of image quality metrics:

1. For a given image and compression method, measure and compare the information content at different rates.

2. Evaluate and optimize different compression methods so as to maximize the quality of an image at any particular rate. Thus, for a compression method $\mathcal{C}$ with tuning parameter θ, the quality metric can be used to choose θ such that the measured quality is maximized for the resulting rate. Further, to compare compression methods $\mathcal{C}_1$ and $\mathcal{C}_2$, the quality metric can be compared at a fixed rate.

3. Serve as a comparison method across images being compressed together, so that the total available space or bandwidth can be divided equitably.

There is no single metric well suited for all three uses. Here, we survey some of the commonly used metrics (and propose one new metric), and present evaluations in terms of the above three uses. A complete discussion of the merits and demerits of various quality metrics is beyond the scope of this thesis. Image quality metrics constitute an active research area. Here, we simply present an overview.

A quality metric is supposed to quantify the information-content of a compressed image. Unfortunately, the term "information-content" is hard to define for images. Moreover, the notion of "information" for an image can differ from application to application. For example, in satellite imagery, clouds may be noise to geologists and information to meteorologists. For this reason, scientific applications typically need to define their own domain-specific metrics.

For common images used simply as "pictures," the notion of information is intuitively obvious: the quality of a compressed picture is determined by the amount of difference the human eye can discern between it and the uncompressed original. Thus, a good quality metric needs to incorporate the characteristics and limitations of the human visual system (HVS) [Kel89]. Unfortunately, there is no precise way of doing that, as the HVS is highly subjective and hard to model [Kel89, Pra91]. Several empirical approximations exist, with widely varying complexities. In practice, the simple, tractable metrics are more commonly used rather than the complex but arguably better ones.

Quality metrics for color images (in terms of the HVS) aren't as well understood as for grayscale images. A color fidelity measure should quantitatively measure just noticeable color differences between images, taking into account the complex processes occurring in the HVS [Kel89, Pra91]. Color images are usually compressed after conversion to the (Y,Cb,Cr) color space, where the Y plane (luminance) characterizes the brightness of the image and the chrominance planes (Cb and Cr) determine the colors. Each pixel (R,G,B) is mapped via a simple transformation to a (Y,Cb,Cr) triplet [Sch85]. This is useful for two reasons: the correlation between the Y, Cb, and Cr planes is lesser than that between R, G, and B. Further, the human eye is less sensitive to high frequencies in the chrominance planes [VB67, Mul85]. The latter fact is usually utilized by subsampling the chrominance planes before encoding. In the presence of such subsampling, we will measure quality relative to the subsampled image itself.

2.2.1 Distortion-Based Metrics

The most popular quality metrics are based on *distortion*[1], or mean-square error (MSE). For a $W \times H$ image plane I, approximated by a compressed image plane I', the distortion $D(I', I)$ is computed as:

$$D(I', I) = \frac{1}{WH} \sum_{0 \le i < H} \sum_{0 \le j < W} \left(I'(i,j) - I(i,j)\right)^2.$$

[1]Usually, *distortion* is used synonymously with "quality loss" in general, but we will consistently use it to mean MSE only.

Higher distortion implies poor quality (low information-content). The popularity of distortion-based metrics stems from the tractability of distortion. Not only is the mean-square error easy to compute, it can be tracked, analyzed and optimized through the compression process.

The most widely used distortion-based quality metric is Peak Signal to Noise Ratio (PSNR) (measured in decibels), which, for grayscale images, is defined as

$$\mathrm{PSNR} = 10 \log_{10} \frac{M^2}{D(I', I)} \ \mathrm{dB}.$$

Here M is the peak signal (the maximum possible value of a pixel), and "dB" is short for decibels.

For color images, the simplest way to calculate the PSNR is to use average distortion over all the color planes. If $D_Y(I', I)$, $D_{Cb}(I', I)$, and $D_{Cr}(I', I)$ are the mean-square errors for the Y, Cb, and Cr planes, respectively, then:

$$\mathrm{PSNR} = 10 \log_{10} \frac{3M^2}{D_Y(I', I) + D_{Cb}(I', I) + D_{Cr}(I', I)} \ \mathrm{dB}.$$

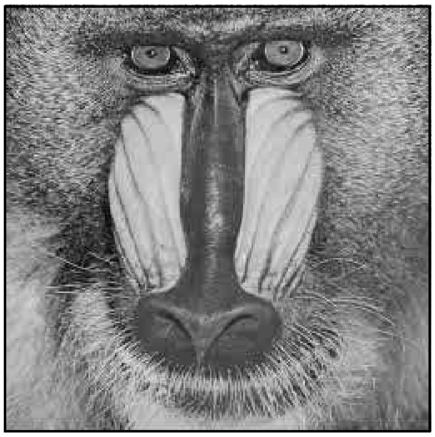
If the chrominance planes have been subsampled, then PSNR is calculated by weighing the distortions according to the subsampled plane sizes. We will use this as our PSNR calculation technique for color images, unless otherwise specified.

This PSNR extension for color images is ad-hoc, like almost every quality metric for color images. For some applications (for example those requiring very low rates), it is useful to give more weight to the luminance distortion. Sometimes it is more natural to use a vector of PSNR's rather than combining the different plane-PSNR's into a single value.

For a given compression method, PSNR is a useful measurement of the amount of information at a given rate. This is especially the case for scientific imagery. PSNR

Lena

PSNR: 30.00 dB, SNR: 24.31 dB
PSNR/N: 0.80, PQS: 0.97



Baboon

PSNR: 30.00, SNR: 24.55 dB
PSNR/N: 1.21, PQS: 2.69

Sailboat

PSNR: 30.00 dB, SNR: 24.85 dB
PSNR/N: 0.93, PQS: 1.91

Splash

PSNR: 30.00 dB, SNR: 23.15 dB
PSNR/N: 0.81, PQS: -1.53

Figure 3: Quality comparison at constant PSNR.

does not correlate too well with the response of the HVS [MKA96]. The reason is that PSNR does not take into account the *structure* in the error (each pixel error is equally important), whereas for the HVS, errors in certain areas of the image (such as uniform backgrounds) are less tolerable than others. Artificial examples can be easily constructed to show that an improvement in PSNR may actually be a degradation in visual quality. In practice, this seldom happens, except for very small changes in PSNR. A compressed image with higher PSNR does have better visual quality, as any reasonable compression technique itself utilizes the image structure. Usually, the problem is that *for roughly the same PSNR*, visual quality can be substantially better if the coding pays more attention to the visual structure [AE97]. This problem can be fixed to some extent by using a *perceptually weighted* distortion in the PSNR calculation. For example, for image compression using the discrete cosine transform (DCT) [ANR74], distortion can be calculated in the DCT coefficient space, with more weight given to the lower frequencies. An example of a perceptual weighing scheme of this kind appears in [CVC95]. Perceptually weighted PSNR offers a good way to approximate visual quality as it can also be analyzed, tracked, and optimized through the compression process.

Another problem with PSNR is that PSNR comparisons across images are usually inconclusive: two different images with identical PSNR's can have radically different visual qualities. Figure 3 shows four test images compressed to the JPEG format, each at a PSNR of 30 dB. Each image is a 256×256 grayscale 8-bit image. For images such as "Baboon" that have a lot of small detail ignored by the eye, a PSNR of 30 dB is fairly good, visually. For an image with a large, uniform background, such as "Splash," 30 dB is a rather poor quality. The other quality metric values listed in the figure (SNR, PSNR/N, PQS) will be discussed shortly.

Normalized PSNR

As seen in Figure 3, PSNR is not a good comparative measure across images. One commonly used approach for normalizing distortion is to use *Signal to Noise Ratio (SNR)*, which, for grayscale images, is defined as

$$\text{SNR} = 10 \log_{10} \frac{S(I)}{D(I', I)} \text{ dB},$$

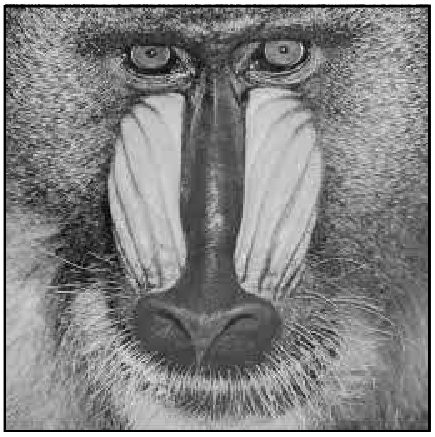
where $S(I)$ is the mean-square pixel intensity for the uncompressed image I. That is, $S(I) = \frac{1}{WH} \sum_{0 \le i \le H-1} \sum_{0 \le j \le W-1} (I(i,j))^2$. For color images, we will take the mean-square pixel value over all the color planes as $S(I)$.

Unfortunately, this normalization is often not good enough. Figure 4 shows that at a constant SNR value of 25 dB, the four test images exhibit different visual qualities, mimicking the constant-PSNR case. The problem is that the normalizing factor, $S(I)$, is not enough to capture the overall "complexity" of an image, which depends on the spatial location and correlation between pixel values too.

We propose a normalized quality measure called PSNR/N, which uses a normalization technique based directly on the rate-PSNR curve for any specific compression method. As observed before, the rate-quality curve typically exhibits a sharp rise, followed by a leveling-off. Intuitively, the sharp rise lasts while the highly structured, high-level information of the image is being coded, and the leveling-off represents coding of the finer details. We use the knee of the curve (the transition between sharp-rise and leveling-off) to normalize the curve. For a given compression method, we obtain the optimal (or nearly optimal) rate-PSNR curve for the entire span of rates possible. We approximate this curve using a least-squares fit with two line segments, and use

Lena

PSNR: 30.68 dB, SNR: 25.00 dB
PSNR/N: 0.82, PQS: 1.20

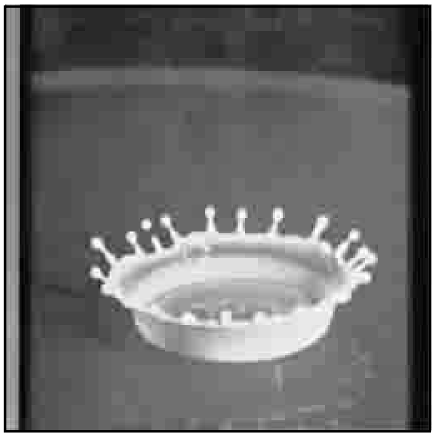


Baboon

PSNR: 30.47 dB, SNR: 25.00 dB
PSNR/N: 1.22, PQS: 2.78

Sailboat

PSNR: 30.14 dB, SNR: 25.00 dB
PSNR/N: 0.93, PQS: 1.97



Splash

PSNR: 31.87 dB, SNR: 25.00 dB
PSNR/N: 0.86, PQS: -0.29

Figure 4: Quality comparison at constant SNR.

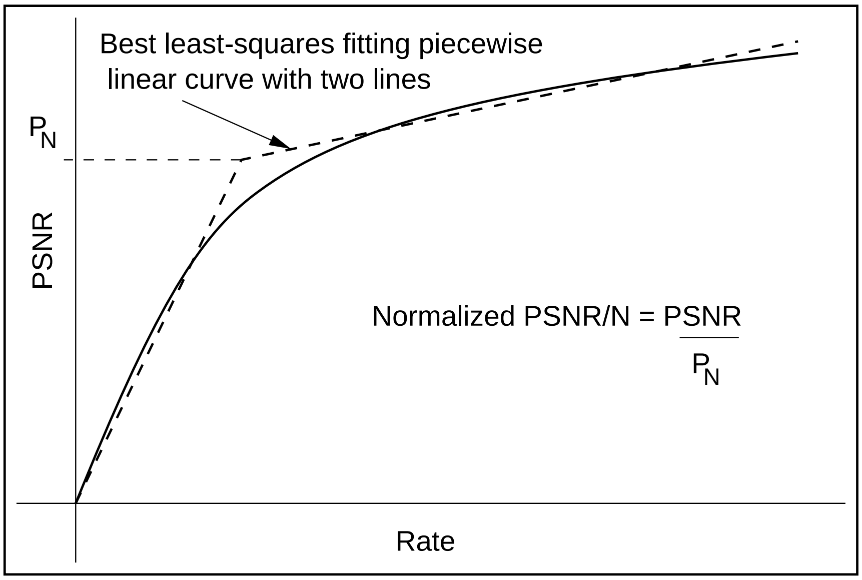


Figure 5: PSNR/N calculation illustrated.

the PSNR at the intersection of the two lines to normalize. If P_N is the PSNR at the knee, that is, at the intersection of the two lines, then a PSNR value of P is normalized to a PSNR/N value of $\frac{P}{P_N}$. Figure 5 illustrates the calculation of PSNR/N, and Figure 6 shows the rate-PSNR curves for the four test images (using optimized JPEG) along with the least-squares fitting line segments. Figure 7 shows the four test images compressed at a constant PSNR/N value of 1.00 (i.e., at the knee). The correlation in visual qualities is seen to be much better than that with SNR or PSNR. However, the "Baboon" image seems to be at a slightly lower visual quality than the others (the coding of the "easy" part of the "Baboon" image lasted till a very low "knee" PSNR, P_N, before the rest of the hard-to-code detail took over). The computation complexity of this normalization is not much: given a rate-PSNR curve with n points, we obtain the least-squares fits for each possible partitioning of the points into two consecutive

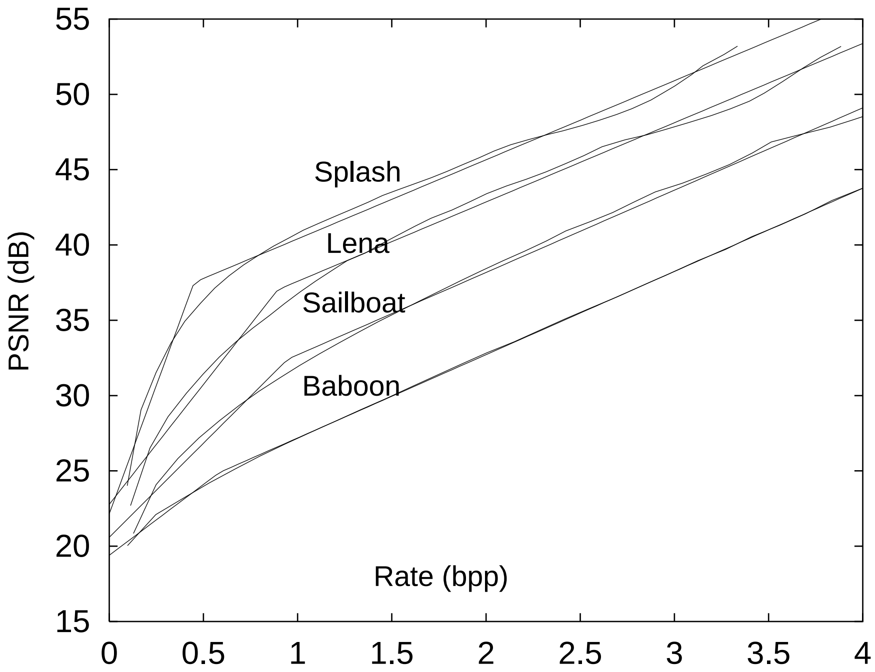
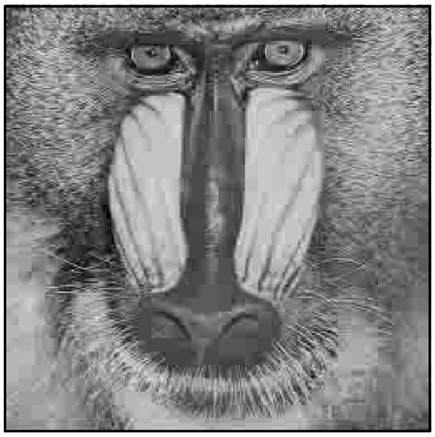


Figure 6: Knee PSNR (P_N) for the four test images.

parts, and retain the best fit. For each partitioning, the coefficients of the linear approximations and the resulting errors are obtained easily using closed-form expressions involving sums, sums of products, and sums of squares of the (x, y) values. By building incremental tables of these sums, the fit for each partition can be computed in constant time. To illustrate this, consider the term $S_x^{m,n} = \sum_{i=m}^{n} \text{bpp}_i$, needed for the computation of the linear fit. The sum-of-x table ($S_x[0 \ldots n]$), given by $S_x[i] = \sum_{j=1}^{i} \text{bpp}_j$, can be computed once, and then used to compute any $S_x^{m,n}$ as $S_x[n] - S_x[m-1]$ in constant time. The time taken to build all the tables is $O(n)$. Hence the overall complexity is also $O(n)$, rather than $O(n^2)$.

Lena

PSNR: 37.52 dB, SNR: 31.81 dB
PSNR/N: 1.00, PQS: 3.29



Baboon

PSNR: 24.89 dB, SNR: 19.44 dB
PSNR/N: 1.00, PQS: 1.37

Sailboat

PSNR: 32.42 dB, SNR: 27.27 dB
PSNR/N: 1.00, PQS: 2.56

Splash

PSNR: 37.04 dB, SNR: 30.13 dB
PSNR/N: 1.00, PQS: 2.51

Figure 7: Quality comparison at constant PSNR/N.

Rate-Distortion Theory

For analysis of compression techniques using distortion-based quality metrics, there is a richly developed theory. In 1948, Shannon showed that the minimum rate (bits per sample) required to code a sequence of discrete samples with a given probability distribution (a "discrete source") can be calculated using the *entropy* function [Sha48]. Moreover, he showed in [Sha48] that the entropy bound could be achieved arbitrarily closely ("noiseless coding theorem"). We will discuss entropy in the context of encoding transformed and/or quantized images with the fewest number of bits, in Section 2.4. In 1959, Shannon extended this to include a fidelity criterion [Sha59], and this paper was the foundation for rate-distortion theory. Rate-distortion theory was originally developed in the context of signal transmission, and deals with the following problem: "Given a sequence of signal values with a known probability density function, what is the minimum rate required to encode the sequence at a given level of distortion?" The minimum rate required for distortion D is represented by the *rate-distortion function*, $R(D)$ [Dav72, Ber71].

Image compression techniques commonly transform the pixel data to decorrelate it. The transformed data can usually be approximated by a Gaussian or Laplacian distribution [Pra91, RG83]. The rate-distortion function for a Gaussian sequence (of independent identically distributed Gaussian elements with variance σ^2) has been found to be [Ber71]:

$$R(D) = \begin{cases} \frac{1}{2}\log_2(\frac{\sigma^2}{D}) & \sigma^2 > D \\ 0 & \sigma^2 \leq D \end{cases}$$

This has also been extended to stationary Gaussian sources with a known covariance matrix. The rate-distortion function for arbitrary distributions is hard to compute, but the Gaussian case can be used to provide both upper and lower bounds [Ber71]. Practical image compression methods can be evaluated in terms of how close they come to the theoretical rate-distortion function. Notice the similarity of the Gaussian rate-distortion function with PSNR and SNR. The rate-PSNR curve for images, after an initial sharp rise, levels off to nearly a straight line. The linear shape can be understood in terms of the $R(D)$ function above.

2.2.2 Psycho-Visual Metrics

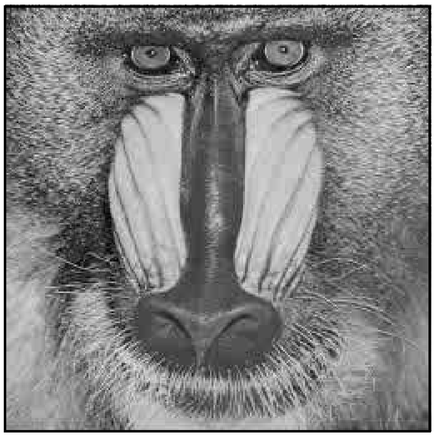
These metrics try to directly assess the HVS response to lossy image compression by modeling the results of extensive subjective tests [AP92, Wat93, MKA96]. Here, we review the Picture Quality Scale metric being developed by Algazi *et al* [MKA96].

The PQS quality metric is designed to correlate with the Mean Observation Score (MOS) evaluation of a large set of test images. For each test image, a number of observers were asked to give a rating on a "scale of impairment" from 0 (very poor quality) to 5 (excellent quality). Intermediate points on the scale were labeled as "perceptible and very annoying loss," "perceptible and slightly annoying loss," "perceptible but not annoying loss," etc. The average MOS rating for each image was calculated. The PQS uses five different carefully designed measures of quality loss (including distortion). A principal components analysis was done on these to find uncorrelated factors, and then multiple regression analysis was used to find weights for these eigen-factors that gave the best correlation with the MOS scores. The current PQS implementation works best for 256×256 grayscale images, at moderate to high qualities. The MOS itself did not have enough range to distinguish between low qualities.

Figure 8 shows the test images JPEG-compressed to a constant PQS (2.51 on a

Lena

PSNR: 34.39 dB, SNR: 28.71 dB
PSNR/N: 0.92, PQS: 2.51

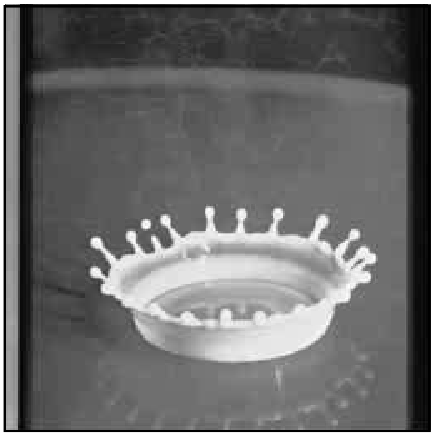


Baboon

PSNR: 29.27 dB, SNR: 23.82 dB
PSNR/N: 1.18, PQS: 2.51

Sailboat

PSNR: 32.17 dB, SNR: 27.03 dB
PSNR/N: 0.99, PQS: 2.51



Splash

PSNR: 36.99 dB, SNR: 30.09 dB
PSNR/N: 1.00, PQS: 2.51

Figure 8: Quality comparison at constant PQS.

scale from 0 to 5, which translates to "perceptible and mildly annoying artifacts"). The correlation between the visual qualities of the four images is much better than with the other metrics (PSNR, SNR, PSNR/N). Note that in some of the previous images (Figures 3 and 4), the PQS values are negative, as some of the "factors" were outside the design range at such poor qualities. This limited application range (moderate to high quality, 256×256 grayscale images) may be relaxed as more research is done. A more serious problem is that PQS measurement is computationally complex, and certainly not amenable to analysis through the compression process. Compression algorithms can optimize PQS only through searching for the optimal parameters by evaluating the PQS at each point (and navigating the search space in some ordered fashion, such as following the best gradient). This is very expensive, and in practice, the best use of PQS is as a calibration tool to uniformly assess qualities across images and compression methods.

We note that of the three quality metrics, PSNR, SNR, and PSNR/N, PSNR/N offers the best compromise in terms of tractability and correlation with visual quality (both subjectively and in terms of PQS values), as seen in Figure 7.

2.2.3 Other Quality Metrics

For scientific applications, the quality of a lossy image is the utility of an analysis done on it, when compared to the same analysis done on the uncompressed original. There has been a lot of work done on evaluating compressed images for medical use [LRF+92, MDS+91, CGO94]. In situations where the image analysis is done by a person (such as medical diagnosis), quality metrics are evaluated by conducting extensive subjective tests (on the "experts" in the field). One common approach for evaluating quality for diagnostic use is to use *Receiver Operating Characteristic (ROC)* analysis which measures the tradeoff between “false positive results” and “true positive results.”

If the intended use of an image is some analysis done by a computer, then it’s easier to measure the quality of the analysis (compared to, say medical diagnosis by doctors). Often, the accuracy of the analysis can be bounded in terms of the mean-square error. Such correlations (between mean-square error and application-specific image quality) can be extremely useful. Using the QCLIC framework, clients can retrieve compressed images from remote servers, run their analysis software on the compressed images, and use the error bounds to recover only a small subset of “interesting” images for full analysis.

2.3 Image Compression Framework

In this section we present an overview of the common elements of image compression. There are many excellent references for detailed discussions of these [NH95, Jai89, JR94, Pra91, BK95, JN84].

Image compression relies on two fundamental traits of pixel intensities in image data: *redundancy* and *irrelevancy* [JN84]. Redundancy relates to statistical properties of images, while irrelevancy relates to the “observer” (human or computer) using the image. Redundancy can be spatial (due to correlation between neighboring pixels) or spectral (due to correlation between color planes or spectral bands). In addition, for video imagery, there can be temporal redundancy as well, due to correlation between consecutive frames. Irrelevancy results from the insensitivity of the HVS or the analysis algorithm to very small details in the image, which include the noise introduced in the image acquisition process.

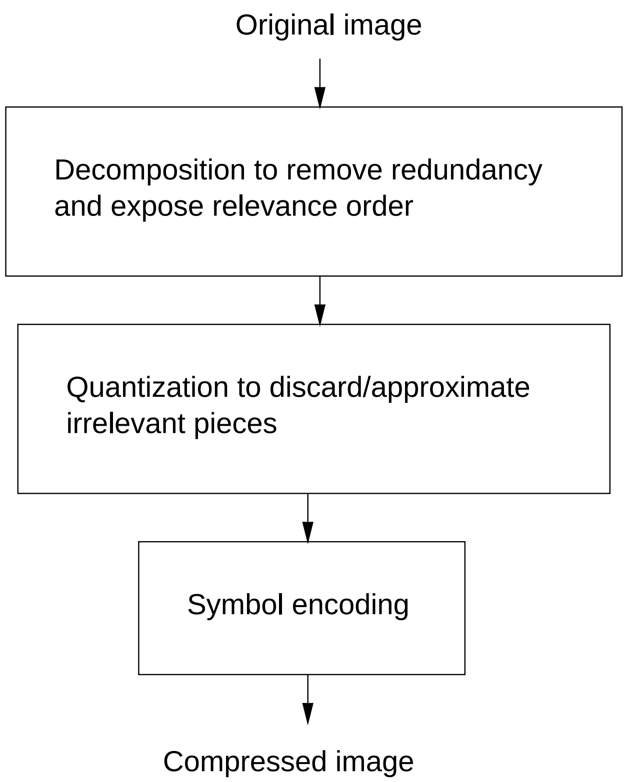


Figure 9: General lossy image compression framework.

A good compression technique needs to decompose the image data so as to remove the redundancy and order the data in order of relevance. This decomposition is usually done by transforming the image data to a "spatial frequency" domain. It is usually lossless, except for floating point calculation errors in practical implementations. This organized information is then subjected to *quantization* which exploits the relevance order to discard or approximately retain the less relevant pieces. The quantization step produces the loss in quality. Typically, the tuning parameters for a compression method are just the parameters for a particular quantization strategy. Finally, the quantized data is encoded into bits that form the compressed image. This encoding does not introduce any further loss. These steps are shown in Figure 9. These steps are highly interdependent. For example, symbol encoding can be done much more efficiently via entropy coding techniques (such as Huffman coding [Huf52]), if the reorganization and quantization produce a representation with low entropy. The efficacy of quantization depends upon the accuracy of the relevance order imposed by the decomposition step.

The heart of an image compression technique is its decomposition step: symbol encoding and quantization are rather general, and common approaches are used in all image compression techniques. For this reason, we discuss symbol encoding and quantization first. Subsequent sections present some compression techniques along with their decomposition mechanisms and the specific flavors of quantization and symbol encoding used by them.

2.4 Symbol Encoding

This step involves the coding of a sequence of symbols (produced by quantization of the decomposed image) into a small number of bits, so that the input sequence can be recovered exactly from the coded bits. This process is also known as *lossless* or *noiseless* encoding.

2.4.1 Fixed-Length Codes

Some applications require the use of fixed-length codewords, for simpler implementations (such as simple buffering requirements) or because of other constraints. For example, if it is important for an image compression application to randomly access arbitrary pieces of an image, fixed-length codewords may be more suitable than variable-length codewords. For an input sample alphabet with N symbols, the rate required, using fixed-length codewords, is $\lceil \log_2 N \rceil$ bits per sample. If N is a power of 2, then this is the least possible rate with fixed-length codewords. If N is not a power of 2, the rate can be improved by grouping together blocks of input symbols. If blocks of k symbols are used, then the rate is $\frac{\lceil \log_2 N^k \rceil}{k}$. It is readily seen that this approaches $\log_2 N$ as $k \to \infty$. Thus, increasing the block size leads to more efficient coding, at the expense of increased complexity and reduced "random-access" capability. Beyond a certain size, the gain in rate anyway becomes insignificant.

2.4.2 Variable-Length Codes

Variable-length codes encode the information produced by a source more efficiently by assigning shorter codewords to frequently occurring symbols, minimizing the rate (the expected codeword length). The best rate that can be achieved in this fashion is bounded by the *entropy* function. For a discrete memoryless stationary source S of symbols over an alphabet of size N, the entropy, $H(S)$, is defined as

$$H(S) = -\sum_{i=1}^{N} p_i \log_2 p_i \quad \text{bits/symbol.}$$

Here p_i is the probability of occurrence of the i^{th} symbol[2]. Shannon showed that this is a lower bound for the rate required to encode the source. For sources with memory, the entropy definition is easily extended using joint probabilities. Note that the entropy is lower for "skewed" distributions that have high probabilities for a very small number of symbols.

Shannon also showed that the entropy bound can be achieved arbitrarily closely, using block coding with variable-length codes. The coding techniques used for this are also known as *entropy coding* techniques. We briefly review two of the most common entropy coding techniques.

[2]For entropy calculations, $0 \log_2 0 \equiv 0$.

Huffman Coding

Huffman coding [Huf52] produces codes which satisfy the *prefix condition:* No codeword is the prefix of any other codeword. Thus, a sequence of codewords can be uniquely mapped to a sequence of symbols. To construct the Huffman codes for a given source, the original alphabet is repeatedly shrunk by merging the two symbols with least probabilities, until only two symbols remain. One of these is assigned the codeword '0' and the other is assigned '1'. The composite symbols are then broken up (in reverse order of the merging), with codewords $\alpha 0$ and $\alpha 1$ assigned to the symbols that were merged to form the composite symbol with codeword α.

It can be shown that if all the symbol probabilities are negative powers of two, Huffman coding achieves the entropy bound exactly. In practice, symbol probabilities can be arbitrary, and Huffman coding may be inefficient as each codeword length is an integer (the optimal codeword length for a symbol with probability p_i is $-\log_2 p_i$). In this case, it can be shown that by using Huffman coding on blocks (of length k) of symbols, the entropy bound is achieved in the limit as $k \rightarrow \infty$. However, this convergence may be slow, and the implementation complexity grows exponentially with k. A better strategy, in some cases, is to use a composite symbol alphabet that groups together runs of frequently occurring symbols into one composite symbol (this is known as *run-length encoding*).

Often, symbol probabilities may not be stationary (for example, in an image, they may change from area to area). In this case, Huffman coding, which assumes a stationary source, is suboptimal. *Dynamic Huffman coding* schemes [Knu85, Vit87], adaptively adjust the codewords through the encoding process, to take into account varying symbol probabilities, but their implementations are complex.

Practical implementations of Huffman coding usually make several simplifications (such as limiting the maximum codeword length) for reducing the complexity, at the cost of a slight increase in rate.

Arithmetic Coding

In Huffman coding there is a codeword for each symbol or block of symbols. Arithmetic coding directly assigns a full code to the entire sequence of symbols. The idea, due to Elias [Eli63], is to partition the interval $[0, 1)$ into non-overlapping subintervals, one for each possible sequence of symbols, such that the length of the subinterval corresponding to a sequence is equal to its probability of occurrence. A sequence is coded by specifying the corresponding subinterval. The subinterval partitioning is done iteratively: using a fixed ordering of the symbols in the alphabet, the current interval is partitioned into pieces whose lengths are proportional to the probabilities of occurrence of the alphabet symbols. Out of these newly created subintervals, the one corresponding to the next input symbol is chosen. This process is repeated for the entire sequence of input symbols. Figure 10 illustrates the interval partitioning step in arithmetic coding.

An input sequence is coded by specifying the corresponding subinterval. It can be shown that if the probability of occurrence of the sequence s is p_s, then the subinterval for s can be specified using $\lceil -\log_2 p_s \rceil$ bits. Thus, each sequence can be coded within one bit of its ideal length.

As presented here, arithmetic coding is hard to implement, because the precision required to carry out the arithmetic for subinterval partitioning increases with the length of the sequence to be coded. Practical implementations use several simplifications (such

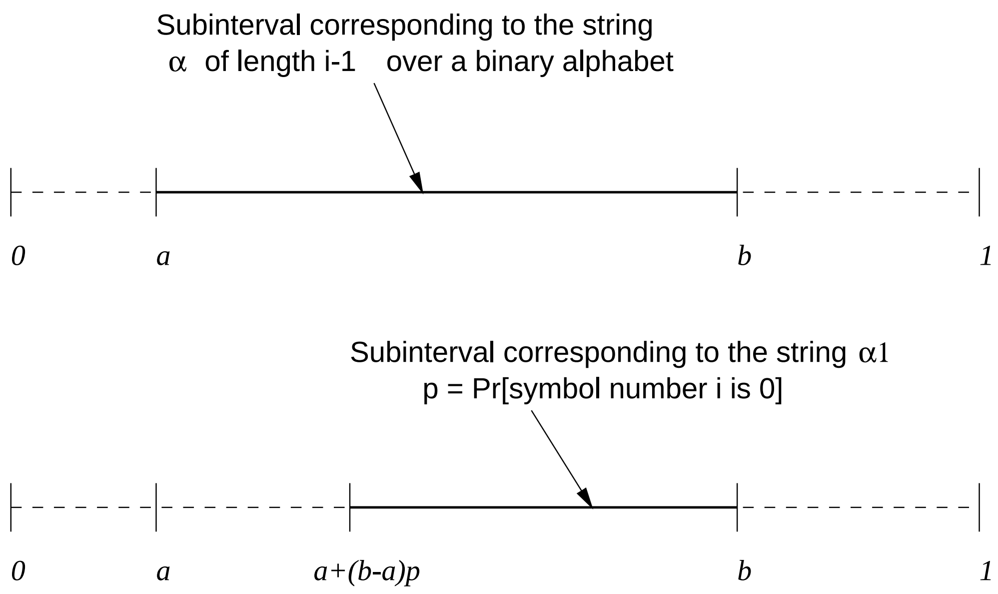


Figure 10: Interval partitioning step in arithmetic coding.

as normalizing the interval size at each step and rounding) to use fixed-precision arithmetic [PMLA88, Lan84, WNC87]. Arithmetic coding can easily adapt to changing source statistics. In fact, IBM's Q-coder [PMLA88] and the QM-coder used in JPEG [PM93] are arithmetic coders with adaptive estimation of symbol probabilities being an integral part of the encoder itself. In practice, arithmetic coding can perform up to 10% better (in terms of rate) than Huffman coding, but it is used less often because it is a patented technique.

2.5 Quantization

The symbols encoded by entropy coding are the output of the quantization step, which is the key loss-controlling step in image compression. The task of the quantization step is to take the decomposed image data and produce an approximation of it, using a smaller number of symbols, while allowing a minimal loss in quality. Typically, the image decomposition step produces a structured representation, with a relevance ordering of the different pieces of the structure. For example, a discrete cosine transform decomposition produces blocks of spatial frequency coefficients, with the low frequency coefficients containing more information than the higher frequency coefficients. Quantization must take into account this ordering to maximize the quality while minimizing the rate.

In this section, we review some commonly used quantization techniques. Quantization can be classified as *scalar quantization* or *vector quantization*. Scalar quantization quantizes each input signal value individually, while vector quantization jointly quantizes blocks (or vectors) of signal values. Scalar quantization is a special case of vector quantization, and is simpler to implement. For memoryless sources (where the signal values are independent), scalar quantization performs nearly as well as vector quantization [GG92]. Usually, the image decomposition step produces largely uncorrelated data. While lack of correlation does not necessarily imply independence (except for Gaussian sources), in practical cases, performance gains of vector over scalar quantization of decorrelated data are not enough to warrant the added complexity [JR94]. Vector quantization is typically used as an image compression technique by itself (without the decomposition step), as the raw pixel intensities are highly correlated. For this reason, we just discuss scalar quantization here, while vector quantization is discussed as a complete image compression technique.

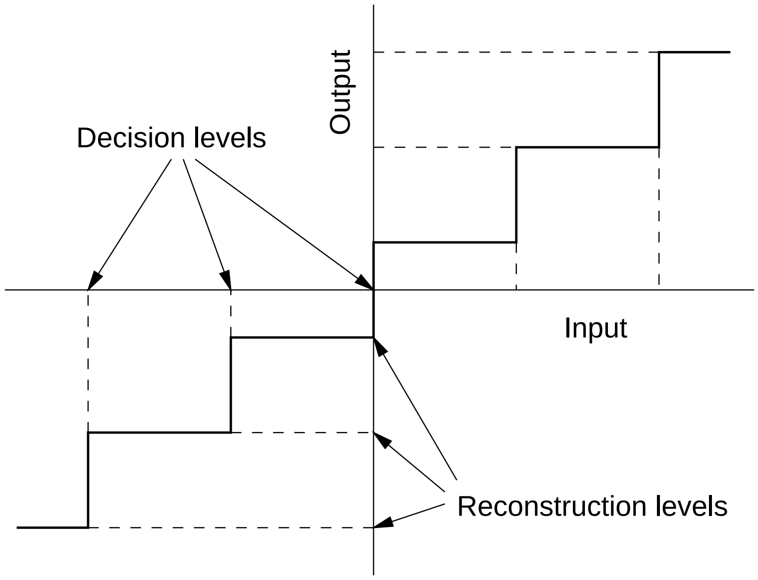


Figure 11: Scalar quantization.

2.5.1 Scalar Quantization

A scalar quantizer maps real numbers to a countable set of *reconstruction levels*, r_i, based on ordered *decision levels*, d_i. If the input x is in the *decision interval* $(d_i, d_{i+1}]$ then the output y is r_i (Figure 11). Each reconstruction level r_i also lies in the corresponding decision interval, $(d_i, d_{i+1}]$.

In practice, the total number of reconstruction levels is finite. For an input signal with a given probability density function $p(x)$, the goal is to design a scalar quantizer that will minimize the distortion while achieving a particular rate. Note that we use distortion to mean MSE. Other metrics can also be used, but the MSE is best suited for analysis of quantizers and hence is most commonly used.

The kind of scalar quantization used depends upon the symbol encoding technique used. If fixed-length codes are used, then for a given number N of reconstruction levels, the rate is fixed, and d_i, r_i need to be designed so as to minimize the distortion.

This is done using the Lloyd-Max technique [Llo82, Max60]. If variable-length codes are used, then the rate will be close to the entropy of the quantizer output. In this case, it is useful to minimize both entropy and distortion, using "entropy-constrained" quantization.

Lloyd-Max Quantizer

When fixed-length symbol encoding is used, the tuning parameter for the quantizer is simply N, the number of reconstruction levels to be used. In this case, the quantizer can be designed using the Lloyd-Max algorithm [Llo82, Max60] to minimize the distortion. The optimal quantizer, Q, is one where $d_0, \ldots, d_N$ and $r_0, \ldots, r_{N-1}$ minimize the distortion D, where,

$$D = \int_{-\infty}^{\infty} (Q(x) - x)^2 p(x) dx.$$

It can be shown that the optimal d_i, and r_i satisfy the constraint that r_i is the mean of the signal conditional on the fact that x lies in the interval $(d_i, d_{i+1}]$. That is,

$$r_i = \frac{\int_{d_i}^{d_{i+1}} x p(x) dx}{\int_{d_i}^{d_{i+1}} p(x) dx}.$$

Further, for $1 \leq i \leq N-1$,

$$d_i = \frac{r_{i-1} + r_i}{2}.$$

The Lloyd-Max technique starts with an arbitrary initial partitioning, d_i, computes the conditional means r_i, then recomputes the decision levels as the averages of neighboring reconstruction levels. This process is repeated until the distortion drop falls below some threshold. In most cases, convergence is rapid [JR94].

Entropy-Constrained Quantization

When variable-length coding is used, the rate of the quantized output is not fixed simply by a choice of N, the number of quantizer levels. Instead, the rate is closely approximated by the entropy of the quantized output. Unlike the Lloyd-Max quantizers, entropy-constrained quantizers try to minimize the entropy as well as the distortion. For a particular quantization strategy determined by parameters Θ, the entropy-constrained quantization (ECQ) optimization problem can be formulated as follows.

> Given a target rate R^*, find Θ such that the resulting rate $R(\Theta)$ is no more than the target R^*, and the distortion $D(\Theta)$ is minimized.

In its most general form, the parameter Θ can be any tuple $(N, \{d_i\}, \{r_i\})$ with N any positive integer or infinity, d_i and r_i arbitrary real numbers such that $d_i \leq r_i \leq d_{i+1}$. In practice, a particular quantization *strategy* is used, which imposes restrictions on the possible values of Θ. For example, the commonly used *uniform scalar quantization* strategy imposes the constraint that the interval width $d_{i+1} - d_i$ is constant.

We present a brief overview of ECQ theory here, and then present our analysis of the Laplacian density case, which is very important from an image compression perspective. All the signal sources in the analyses presented in this chapter are assumed to be sequences of independent, identically distributed random variables.

The rate $R(\Theta)$ corresponding to a choice of the parameters $\Theta = (N, \{d_i\}, \{r_i\})$ is calculated as the entropy,

$$R(\Theta) = -\sum_i p_i \log_2 p_i,$$

where p_i is the probability that the signal value x is in $(d_i, d_{i+1}]$. That is,

$$p_i = \int_{d_i}^{d_{i+1}} p(x)dx. \tag{2.1}$$

The distortion is similarly calculated as,

$$D(\Theta) = \sum_i \int_{d_i}^{d_{i+1}} (x - r_i)^2 p(x) dx.$$

The ECQ optimization problem can be solved by minimizing the Lagrangian $R(\Theta) + \lambda D(\Theta)$. It can be shown that for any $\lambda \geq 0$, a solution Θ_L to the Lagrangian mini-

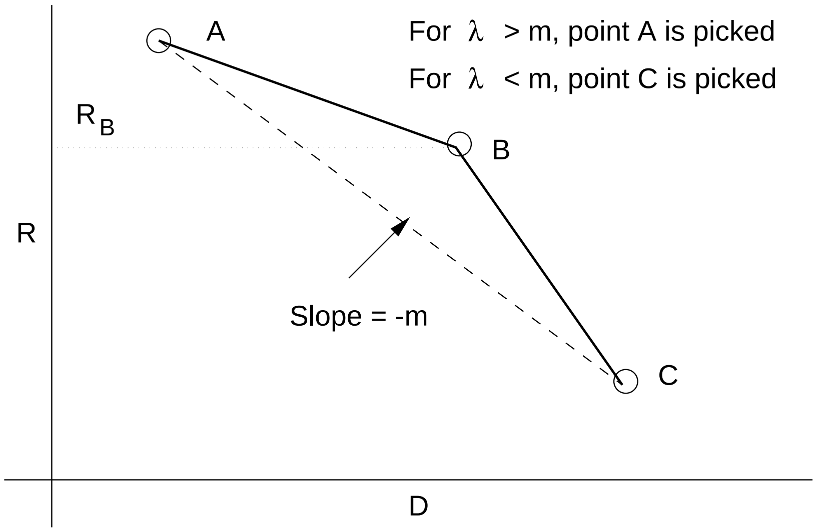


Figure 12: Lagrangian minimization only picks points on the convex hull.

mization problem is also a solution to the ECQ optimization problem, for $R^* = R(\Theta_L)$. Not all optimal solutions to the ECQ optimization problem are necessarily solutions to the Lagrangian minimization problem. The Lagrangian will only be minimized at points on the convex hull of the R-D curve. In Figure 12, point B will never be picked by Lagrangian minimization, even though it is a solution to the ECQ optimization problem for $R^* = R_B$. In practice, the points that minimize the Lagrangian are dense enough to achieve any target R^* very closely. Also, if the optimal rate versus distortion curve (optimal with respect to the choice of Θ out of a possibly restricted set) is continuous with slope increasing towards zero, then every solution to the ECQ optimization problem can be obtained via Lagrangian minimization. This can be easily

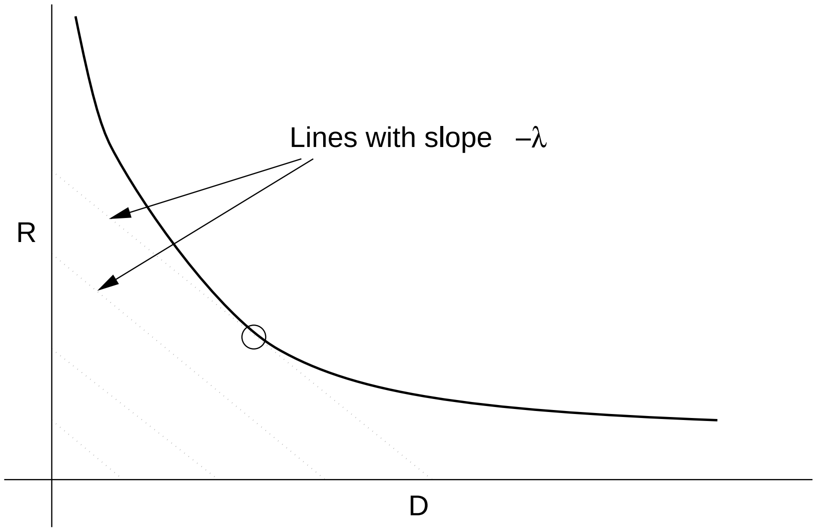


Figure 13: The point that minimizes $R + \lambda D$.

seen from the geometrical interpretation of Lagrangian minimization (Figure 13). If we draw lines with slope $-\lambda$ at increasing distance from the origin, the first point where the rate-distortion curve is touched will minimize the Lagrangian.

Consider the general case with no restrictions on possible choices for Θ. For a particular choice of d_i and d_{i+1}, the probability p_i does not depend on the reconstruction level r_i. Hence, the choice of r_i does not affect the rate, and should be made so as to minimize the distortion. Then,

$$r_i = \frac{\int_{d_i}^{d_{i+1}} xp(x)dx}{\int_{d_i}^{d_{i+1}} p(x)dx}. \tag{2.2}$$

Taking partial derivatives of the Lagrangian $R+\lambda D$ with respect to each d_i and equating with zero results in a set of non-linear simultaneous equations:

$$\log_2\left(\frac{p_i}{p_{i-1}}\right) = \lambda[(r_i - r_{i-1})(r_i + r_{i-1} - 2d_i)] \tag{2.3}$$

Equations 2.2 and 2.3, first presented by Berger [Ber72], provide necessary conditions for a quantizer to be optimal. If d_{i-1}, d_i and λ are known, then equations 2.1, 2.2, and 2.3 can be solved numerically to compute d_{i+1}. Given some boundary conditions (such as the first and last decision level), and λ, this offers an iterative algorithm to solve for the optimal quantizer [Ber72, FM84]. Unfortunately, there can be two solutions for each d_{i+1}, for fixed values of d_{i-1}, d_i, and λ. If each possible solution is examined, the complexity of the iterative algorithm grows exponentially. In practice, the lower of the two solutions is used, based on empirical evidence that it usually is the right choice [Ber72, Ber82, FM84].

2.6 ECQ for Laplacian Density

The Laplacian pdf is of special interest in image compression, as most image compression techniques use decompositions that produce data closely approximated by a Laplacian density [RG83, MHY82, Tes79].

The Laplacian density is characterized by the pdf,

$$p(x) = \frac{\alpha}{2} e^{-\alpha|x|}, \quad -\infty < x < \infty.$$

The variance σ^2 of the distribution is $2/\alpha^2$.

We characterize the performance of several quantization strategies on the Laplacian pdf. We assume that N, the number of quantizer levels, is not bounded. Placing a sufficiently large bound on N does not change rate or distortion appreciably, as the contributions made by the "tails" become negligible for large N. Moreover, the results can be easily extended to the case when N is small.

Berger has shown in [Ber72] that a uniform quantizer with 0 as one of the decision levels (a *mid-riser*[3] uniform quantizer) satisfies the necessary conditions for optimality given by equations 2.2 and 2.3. For a quantizer step size of q, the decision levels for

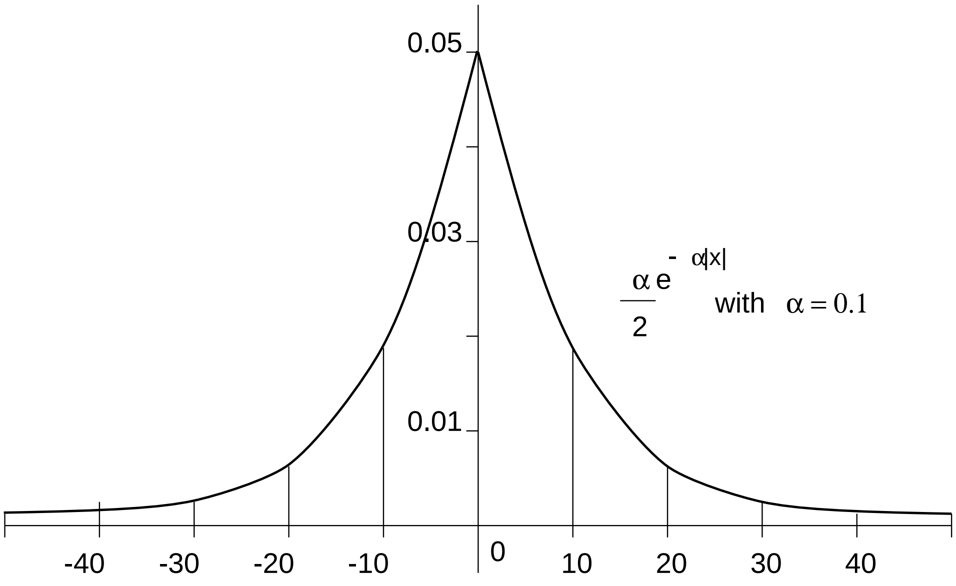


Figure 14: Mid-riser uniform quantizer for a Laplacian source with $\alpha = 0.1$. Decision levels are shown for $q = 10$.

this quantizer are $d_i = iq$, for $i = 0, \pm 1, \pm 2, \ldots$ Figure 14 shows the Laplacian density for $\alpha = 0.1$, and shows the decision levels for $q = 10$. Let θ denote $e^{-\alpha q}$. Then the reconstruction levels are given by $r_i = iq + \alpha^{-1} - q\theta(1-\theta)^{-1}$, for $i \geq 0$. For $i < 0$, r_i are symmetrically obtained. In terms of the parameter θ, the rate and distortion are [Ber72]:

$$R(\theta) = (1-\theta)^{-1}\left[-\theta \log_2 \theta - (1-\theta)\log_2(1-\theta)\right] + 1, \tag{2.4}$$

$$D(\theta) = \frac{2}{\alpha^2}\left[\frac{1-\theta(1-\theta)^{-2}\log_e^2 \theta}{2}\right]. \tag{2.5}$$

Note that the distortion expression has been written so that the "normalized distortion," D/σ^2, is apparent. We will refer to this quantizer as the MR-U-C quantizer

[3]The name derives from the fact that the quantizer output versus input "staircase" plot rises at zero.

(Mid-Riser, Uniform, reconstruction at Conditional mean). As the parameter $\theta = e^{-\alpha q}$ runs from 0 to 1, $R(\theta)$ runs from 1 to ∞ and $D(\theta)$ from $1/\alpha^2$ to 0. Since 0 is one of the decision levels, even in the limit as $\theta \to 0$ (that is, $q \to \infty$), there are two reconstruction levels, one to the right of the origin and one to the left. The rate, for symmetric quantizers with zero as a decision level and working on symmetric pdf's, is always ≥ 1, (a proof can be found in [FM84]). Intuitively, in the limit there will be two symbols to be coded with probability 0.5 for each, which has entropy 1.

Unfortunately, the MR-U-C quantizer is not always optimal for the Laplacian pdf, even for $R > 1$. Figure 15 shows the rate-distortion performance of the MR-U-C quantizer on a Laplacian source. All the other quantizers shown (which will be described shortly) perform better, except at very low distortions, where all the curves coincide. The $R_L(D)$ curve shown is the generalized Shannon lower bound to the theoretical rate-distortion function, which is obtained as the $R(D)$ curve of a Gaussian density with equal differential entropy [Ber71].

When zero is one of the reconstruction levels, the quantizer is classified as a "mid-tread quantizer." For image compression applications, mid-tread quantizers are preferred over mid-riser ones, as a mid-riser quantizer quantizes even a zero input to a non-zero output. The signal being quantized is usually the coefficient corresponding to some spatial frequency, and approximating a zero with a non-zero corresponds to introducing a frequency when there isn't any, which results in visible artifacts.

The simplest mid-tread quantizer is again the uniform quantizer, with each decision interval having a constant width, q. Figure 16 shows a uniform mid-tread quantizer with $q = 10$. It is more convenient to adopt the following notation for the decision levels: "bin n" refers to the decision interval centered at nq, for $n = 0, \pm 1, \pm 2, \ldots$ Endpoints

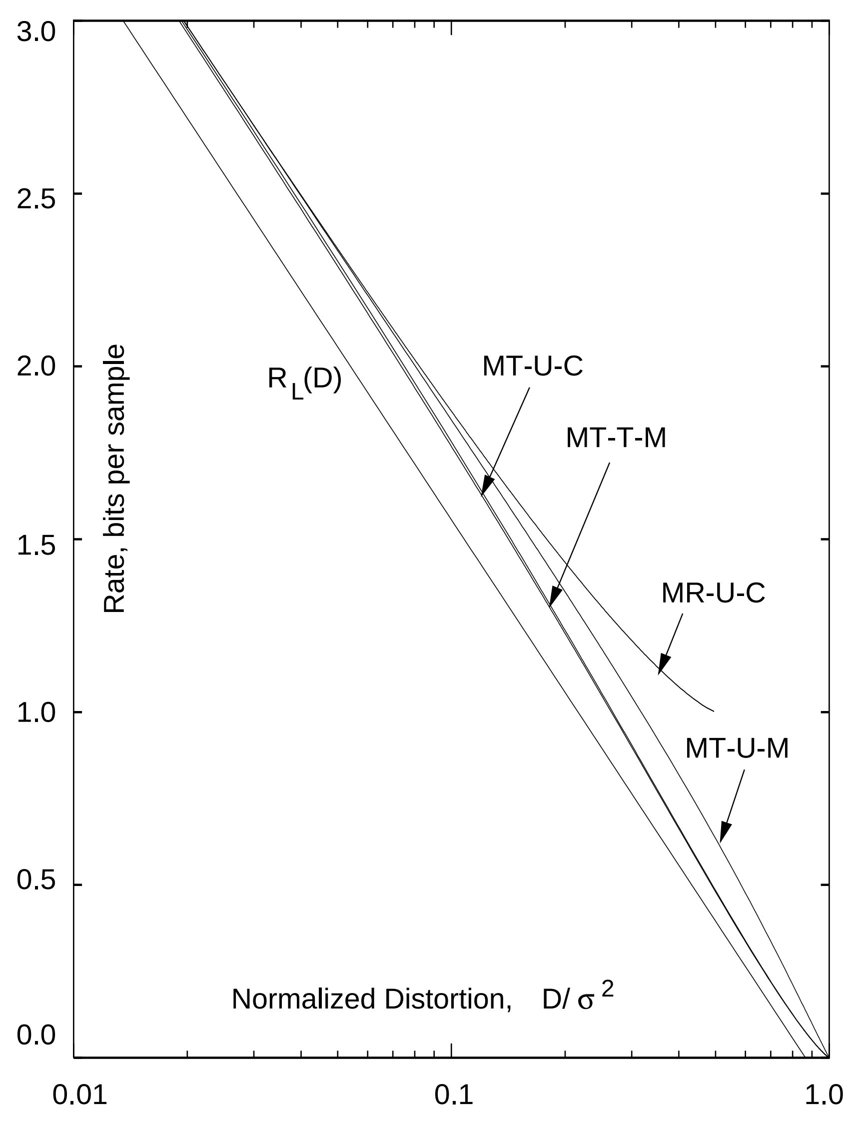


Figure 15: Performance of different quantization strategies on a Laplacian pdf.

are assigned to decision intervals as follows. The decision interval for bin 0 is taken to be $(-q/2, q/2)$. For $n > 0$, the decision interval is taken to be $[nq - q/2, nq + q/2)$, and for $n < 0$, the decision interval is $(nq - q/2, nq + q/2]$. The *quantized value* of an input x is the value n such that bin n contains x. Note that n can be calculated very simply by rounding x/q to the nearest integer. The symbol p_n now denotes the probability that an input x is in bin n. Given a step size q, the reconstruction level for bin n that

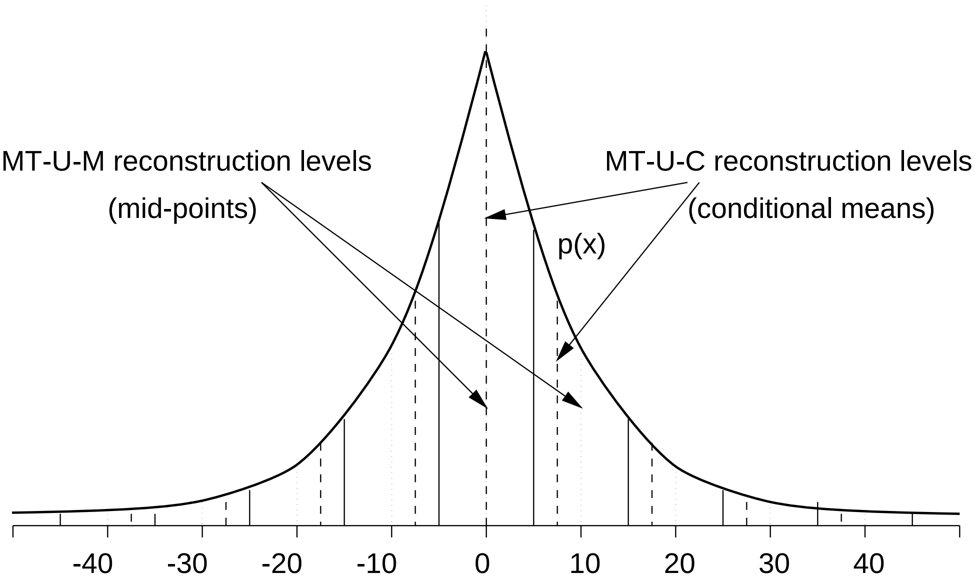


Figure 16: Mid-tread uniform quantizer for a Laplacian source. Decision levels are shown for $q = 10$.

minimizes the distortion is the conditional mean given by

$$c_n = \begin{cases} 0, & n = 0, \\ nq - q/2 + \delta, & n > 0, \\ nq + q/2 - \delta, & n < 0. \end{cases} \tag{2.6}$$

Here $\delta = \alpha^{-1} - q\beta^2(1-\beta^2)^{-1}$, with β denoting $e^{-\alpha q/2}$ (using $e^{-\alpha q/2}$ as the parameter simplifies subsequent notation). We refer to this quantizer as the MT-U-C quantizer

(Mid-Tread, Uniform, reconstruction at Conditional mean). The MT-U-C quantizer does not satisfy the necessary conditions for optimality. Its performance is nevertheless better than the MR-U-C quantizer, as can be seen from its rate-distortion curve shown in Figure 15. We have calculated the rate and distortion for the MT-U-C quantizer, in terms of the parameter $\beta = e^{-\alpha q/2}$, as:

$$\begin{aligned} R(\beta) &= \beta\left[1 - \log_2(1+\beta) - \log_2\beta\left(\frac{1+\beta^2}{1-\beta^2}\right)\right] - \log_2(1-\beta), & (2.7)\\ D(\beta) &= \frac{2}{\alpha^2}\left[1 - \frac{\beta}{2} + \beta\log_e(\beta) - \frac{\beta}{2}\log_e^2\beta\left(\frac{1+\beta^2}{1-\beta^2}\right)^2\right]. & (2.8)\end{aligned}$$

The reconstruction levels, c_n, depend upon the parameter α of the Laplacian pdf. A very commonly used simplification is to use the mid-points of bins as reconstruction levels, instead of the conditional means. Thus, the reconstruction level m_n for bin n becomes

$$m_n = nq. \tag{2.9}$$

In this case, the decoder does not need to know the value of α. Further, the decoder simply needs to multiply a quantized value n by q to get the reconstruction value $m_n = nq$. We refer to this quantizer as the MT-U-M quantizer (Mid-Tread, Uniform, reconstruction at Mid-point). Figure 16 also shows the reconstruction levels for the MT-U-M quantizer. The MT-U-M quantizer is probably the most commonly used quantizer in image compression, because of its simplicity. The JPEG image compression standard, for example, uses MT-U-M quantizers. In terms of the parameter $\beta = e^{-\alpha q/2}$, we have calculated the rate and distortion for this quantizer as:

$$\begin{aligned} R(\beta) &= \beta\left[1 - \log_2(1+\beta) - \log_2\beta\left(\frac{1+\beta^2}{1-\beta^2}\right)\right] - \log_2(1-\beta), & (2.10)\\ D(\beta) &= \frac{2}{\alpha^2}\left[\frac{1}{2} + \frac{\beta\log_e\beta}{(1-\beta^2)}\right]. & (2.11)\end{aligned}$$

The rate-distortion curve for the MT-U-M quantizer is also shown in Figure 15. At rates greater than about 2.5 bits per sample, the performance is very close to that of the MT-U-C quantizer. At moderate to low rates, however, the MT-U-C quantizer performs considerably better. Note that the X-axis is normalized distortion ($D/\sigma^2 = D\alpha^2/2$), on a logarithmic scale. At a typical rate of 0.5 bits per sample, the MT-U-M distortion is about twenty percent greater than the MT-U-C distortion.

In conjunction with our work on JPEG optimization (Chapter 4), we found that a simple modification to the MT-U-M quantizer improves its performance considerably. The intuition is as follows. For a Laplacian density quantized using an MT-U-M quantizer, the most probable symbol is zero, and its expected information ($-p_0 \log_2 p_0$) is the dominant component of the total rate. Thus, keeping bin 0 constant, if we reduce the widths of all other bins, the rate will not increase much, whereas the distortion might decrease considerably. Expressed another way, the Lagrangian $R + \lambda D$ may be lowered for some λ if all bins n are kept constant for $n \neq 0$, and bin 0 is expanded (that is, the *zeroing threshold* is increased from $q/2$). We refer to this as an MT-T-M quantizer (Mid-Tread, Thresholding, reconstruction at Mid-point). Figure 17 illustrates the MT-T-M quantizer. The additional parameter (apart from q, the step size) that determines this quantizer is a positive real zeroing threshold T, with $T \geq q/2$. For an input x, if x is in the interval $(-T, T)$, then its quantized value is zero. For $x \geq T$, it is quantized as in the MT-U-M case. That is, its quantized value (bin) is x/q rounded to the nearest integer. A very useful fact is that the decoder is exactly the same as in the MT-U-M case: a quantized value is simply multiplied by q to get the reconstruction level. Thus, any compression technique that uses the MT-U-M quantization strategy can easily be extended to use MT-T-M, without changing the decoder.

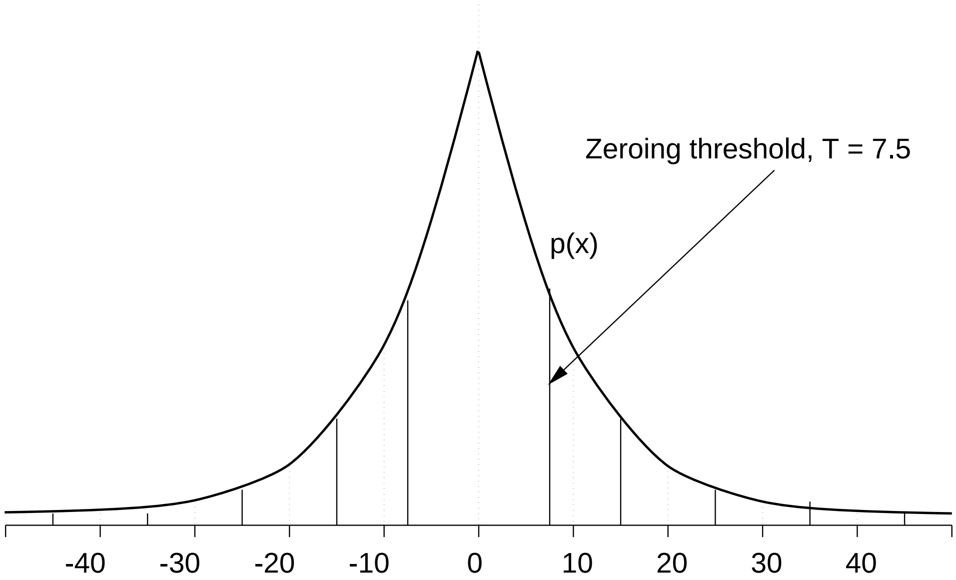


Figure 17: Mid-tread thresholding quantizer (MT-T-M) for a Laplacian source. Decision levels are shown for $q = 10, T = 7.5$. Reconstruction levels are at $10n$, for each bin n.

To characterize and optimize the performance of the MT-T-M quantization strategy, we investigate the rate and distortion for a given value of q and $T \geq q/2$. Let k be the largest positive integer such that $kq - q/2 \leq T$. The bins k and $-k$ are the closest bins to zero which aren't completely merged into bin zero. Let t denote $T - (kq - q/2)$. Then, rate and distortion can be expressed in terms of the parameters $\beta = e^{-\alpha q/2}$, k, and $\gamma = e^{-\alpha t}$, as follows.

$$\begin{aligned} R(\beta, k, \gamma) &= -(1 - \gamma\beta^{2k-1}) \log_2(1 - \gamma\beta^{2k-1}) \\ &\quad -(\gamma\beta^{2k-1} - \beta^{2k+1}) \left[(2k-1)\log_2 \beta + \log_2(\gamma - \beta^2) - 1\right] \\ &\quad -\beta^{2k+1} \left[\log_2 \beta \left(2k + \frac{1+\beta^2}{1-\beta^2}\right) + \log_2(1-\beta^2) - 1\right], \qquad (2.12) \\ D(\beta, k, \gamma) &= \frac{2}{\alpha^2} \left[1 + 2\beta^{2k-1} \log_e \beta \left(k\gamma(1 - \log_e(\gamma\beta^{k-1})) + \frac{\beta^2}{1-\beta^2}\right)\right]. \qquad (2.13) \end{aligned}$$

The ranges for the parameters are: $0 < \beta < 1$, k is an integer ≥ 1, and $\beta^2 < \gamma \leq 1$.

Unlike the quantization strategies discussed earlier, several different choices of β, k, and γ might result in the same rate or the same distortion. We have developed an algorithm to choose optimal β, k, and γ, in the entropy-constrained optimization sense. We present the results obtained, and defer a complete description of the optimization algorithm to the next subsection.

Figure 15 shows the performance of the MT-T-M quantizer optimized using our algorithm. The curve coincides almost exactly with that of the MT-U-C quantizer. At high rates and low distortions, the curves for all the quantizers coincide, as uniform quantizers are optimal in the limit as $D \to 0$ [GP68]. At moderate rates, there is a nearly imperceptible gap between the MT-T-M and MT-U-C curves. The interesting result is that the curves coincide exactly at low rates. Thus, the large gap between the MT-U-M and MT-U-C curves can be avoided by using the MT-T-M quantizer, which can use the same decoder as the MT-U-M quantizer. The gains achieved by using conditional means seem to equal the gains achieved by using thresholding. This suggests the possibility of further improvement, by using thresholding on the MT-U-C quantizer. We used an optimization algorithm similar to the one described below to do this, but the improvements were extremely minimal.

2.6.1 Optimizing the MT-T-M Quantizer for Laplacian Density

To optimize the rate-distortion performance, we investigate the Lagrangian minimization problem for the MT-T-M quantizer.

Given a $\lambda \geq 0$, we must choose β, k, and γ so as to minimize the Lagrangian $R(\beta, k, \gamma) + \lambda D(\beta, k, \gamma)$. First we describe the optimization of γ for fixed β and k. This can then be used to easily optimize all three parameters.

For a fixed value of β and k, the rate $R_{\beta,k}(\gamma)$, distortion $D_{\beta,k}(\gamma)$, and Lagrangian $L_{\beta,k}(\gamma) = R_{\beta,k}(\gamma) + \lambda D_{\beta,k}(\gamma)$ are functions of γ determined by equations 2.12 and 2.13. For simplifying the analysis notation, we temporarily drop the subscript β, k. To find the minimum value achieved by $L(\gamma) = R(\gamma) + \lambda D(\gamma)$ over $(\beta^2, 1]$, we consider its derivative. For $dR(\gamma)/d\gamma$, we note that only the contributions to $R(\gamma)$ from bins $-k$,, 0, and k depend on γ.

$$
\begin{aligned}
&\frac{dR(\gamma)}{d\gamma} \\
&= \frac{d}{d\gamma}\left[-p_0 \log_2 p_0 - p_k \log_2 p_k - p_{-k} \log_2 p_{-k}\right] \\
&= \frac{d}{d\gamma}\left[-p_0 \log_2 p_0 - 2p_k \log_2 p_k\right] \\
&= -\frac{1}{\log_e 2}\left[(1+\log_e p_0)\frac{dp_0}{d\gamma} + 2(1+\log_e p_k)\frac{dp_k}{d\gamma}\right].
\end{aligned}
$$

Here p_n is the probability of the input x falling in bin n. Thus,

$$
\begin{aligned}
&\frac{dp_0}{d\gamma} \\
&= \frac{d}{d\gamma}\left[2\int_0^T \frac{\alpha}{2} e^{-\alpha x} dx\right] \\
&= \frac{d}{d\gamma}\left[1 - e^{-\alpha T}\right] \\
&= \frac{d}{d\gamma}\left[1 - \gamma\beta^{2k-1}\right] \quad (\text{as } T = (2k-1)q/2 + t) \\
&= -\beta^{2k-1}.
\end{aligned}
$$

Similarly,

$$
\begin{aligned}
&\frac{dp_k}{d\gamma} \\
&= \frac{d}{d\gamma}\left[\int_T^{(2k+1)q/2} \frac{\alpha}{2} e^{-\alpha x} dx\right]
\end{aligned}
$$

$$= \frac{d}{d\gamma}\left[(e^{-\alpha T} - e^{-\alpha(2k+1)q/2})/2\right]$$
$$= \frac{d}{d\gamma}\left[(\gamma\beta^{2k-1} - \beta^{2k+1})/2\right]$$
$$= \beta^{2k-1}/2.$$

Hence,

$$\frac{dR(\gamma)}{d\gamma}$$
$$= -\frac{1}{\log_e 2}\left[(1 + log_e p_0)(-\beta^{2k-1}) + 2(1 + log_e p_k)(\beta^{2k-1}/2)\right]$$
$$= \beta^{2k-1}\log_2\frac{p_0}{p_k}.$$

Substituting $p_0 = 1 - \gamma\beta^{2k-1}$ and $p_k = (\gamma\beta^{2k-1} - \beta^{2k+1})/2$ yields,

$$\frac{dR_{\beta,k}(\gamma)}{d\gamma} = \beta^{2k-1}\log_2\frac{2(1-\gamma\beta^{2k-1})}{\beta^{2k-1}(\gamma-\beta^2)}, \quad \text{for } \beta^2 < \gamma \leq 1. \tag{2.14}$$

To find $dD(\gamma)/d\gamma$, we work directly from equation 2.13.

$$\frac{dD(\gamma)}{d\gamma}$$
$$= \frac{2}{\alpha^2}\frac{d}{d\gamma}\left[1 + 2\beta^{2k-1}\log_e\beta\left(k\gamma(1-\log_e(\gamma\beta^{k-1})) + \frac{\beta^2}{1-\beta^2}\right)\right]$$
$$= \frac{4k\beta^{2k-1}\log_e\beta}{\alpha^2}\frac{d}{d\gamma}\left[\gamma(1-\log_e(\gamma\beta^{k-1}))\right]$$
$$= \frac{4k\beta^{2k-1}\log_e\beta}{\alpha^2}\left[1-(1+\log_e\gamma)-\log_e\beta^{k-1}\right].$$

Hence,

$$\frac{dD_{\beta,k}(\gamma)}{d\gamma} = -\frac{4k\beta^{2k-1}\log_e\beta}{\alpha^2}\log_e(\gamma\beta^{k-1}), \quad \text{for } \beta^2 < \gamma \leq 1. \tag{2.15}$$

Hence, the slope of the Lagrangian with respect to γ is:

$$\frac{dL(\gamma)}{d\gamma}$$
$$= \frac{dR(\gamma)}{d\gamma} + \lambda\frac{dD(\gamma)}{d\gamma}$$

$$= \beta^{2k-1}\left[\log_2 \frac{2(1-\gamma\beta^{2k-1})}{\beta^{2k-1}(\gamma-\beta^2)} + -\frac{4k\lambda \log_e \beta}{\alpha^2}\log_e(\gamma\beta^{k-1})\right]$$
$$= \beta^{2k-1}\left[\log_2 \frac{2(1-\gamma\beta^{2k-1})}{\beta^{2k-1}(\gamma-\beta^2)} + s\log_2(\gamma\beta^{k-1})\right].$$

Here s denotes the constant $-\frac{4k\lambda \log_e \beta \log_e 2}{\alpha^2}$. Note that $s \geq 0$. Thus, we have,

$$\frac{dL_{\beta,k}(\gamma)}{d\gamma} = \beta^{2k-1}\log_2 \frac{2\gamma^s\beta^{s(k-1)}(1-\gamma\beta^{2k-1})}{\beta^{2k-1}(\gamma-\beta^2)}, \quad \text{for } \beta^2 < \gamma \leq 1. \tag{2.16}$$

Over the interval of interest, $(\beta^2, 1]$, $\gamma-\beta^2 > 0$. Hence, $\frac{dL(\gamma)}{d\gamma}$ is negative, zero, or positive whenever $\left[2\gamma^s\beta^{s(k-1)}(1-\gamma\beta^{2k-1})\right] - \left[\beta^{2k-1}(\gamma-\beta^2)\right]$ is negative, zero, or positive. This expression can be simplified as:

$$\left[2\gamma^s\beta^{s(k-1)}(1-\gamma\beta^{2k-1})\right] - \left[\beta^{2k-1}(\gamma-\beta^2)\right]$$
$$= -2\gamma^{s+1}\beta^{(s+1)(k-1)+k} + 2\gamma^s\beta^{s(k-1)} - \gamma\beta^{2k-1} + \beta^{2k+1}$$
$$= \beta^{2k-1}\left[-2\gamma^{s+1}\beta^{s(k-1)} + 2\gamma^s\beta^{s(k-1)-(2k-1)} - \gamma + \beta^2\right].$$

Let $F(\gamma)$ denote the function in brackets above. That is,

$$F(\gamma) = -2\gamma^{s+1}\beta^{s(k-1)} + 2\gamma^s\beta^{s(k-1)-(2k-1)} - \gamma + \beta^2. \tag{2.17}$$

Then the derivative $\frac{dL(\gamma)}{d\gamma}$ has the same sign as $F(\gamma)$ over $(\beta^2, 1]$. We can solve the equation $F(\gamma) = 0$ to minimize the Lagrangian, but we first investigate the second derivative, which leads to useful simplifications.

Since we are only interested in the sign of the second derivative of $L(\gamma)$, we use the notation "$=_\pm$" to mean "has the same sign as." Thus, $\frac{dL(\gamma)}{d\gamma} =_\pm F(\gamma)$ means that $\frac{dL(\gamma)}{d\gamma}$ is negative, zero, or positive, whenever $F(\gamma)$ is negative, zero, or positive. Now,

$$\frac{d^2L(\gamma)}{d\gamma^2}$$
$$= \beta^{2k-1}\frac{d}{d\gamma}\log_2 \frac{2\gamma^s\beta^{s(k-1)}(1-\gamma\beta^{2k-1})}{\beta^{2k-1}(\gamma-\beta^2)}$$

$$
\begin{aligned}
&= \frac{\beta^{2k-1}}{\log_e 2}\frac{d}{d\gamma}\left[s\log_e\gamma + \log_e(1-\gamma\beta^{2k-1}) - \log_e(\gamma-\beta^2)\right] \\
&=_{\pm} \frac{s}{\gamma} - \frac{\beta^{2k-1}}{1-\gamma\beta^{2k-1}} - \frac{1}{\gamma-\beta^2} \\
&=_{\pm} s(1-\gamma\beta^{2k-1})(\gamma-\beta^2) - \gamma(\gamma-\beta^2)\beta^{2k-1} - \gamma(1-\gamma\beta^{2k-1}) \quad (\text{as } \gamma > \beta^2) \\
&= [-s\beta^{2k-1} - \beta^{2k-1} + \beta^{2k-1}]\gamma^2 + [s + s\beta^{2k+1} + \beta^{2k+1} - 1]\gamma + [-s\beta^2] \\
&= -s\beta^{2k-1}\gamma^2 + [s(1+\beta^{2k+1}) - (1-\beta^{2k+1})]\gamma - s\beta^2.
\end{aligned}
$$

Thus, $\frac{d^2L(\gamma)}{d\gamma^2}$ has the same sign as the function $G(\gamma)$ over the interval $(\beta^2, 1]$, where,

$$G(\gamma) = -s\beta^{2k-1}\gamma^2 + [s(1+\beta^{2k+1}) - (1-\beta^{2k+1})]\gamma - s\beta^2. \tag{2.18}$$

The function $G(\gamma)$ is quadratic in γ, and hence the equation $G(\gamma) = 0$ can be easily solved. If there are no real roots, then $G(\gamma)$ never changes its sign for $-\infty < \gamma < \infty$, and hence is always negative (as $G(0) = -s\beta^2$ is negative). If there are real roots γ_1 and γ_2, with $\gamma_1 \leq \gamma_2$, then $G(\gamma) = -s\beta^{2k-1}(\gamma-\gamma_1)(\gamma-\gamma_2)$. Hence $G(\gamma)$ is zero at γ_1 and γ_2, less than zero over $(-\infty, \gamma_1)$ and (γ_2, ∞), and greater than zero over (γ_1, γ_2).

We know that $\frac{d^2L(\gamma)}{d\gamma^2} =_{\pm} G(\gamma)$, for $\beta^2 < \gamma \leq 1$. If $G(\gamma)$ has no real roots, or if the intervals $(\beta^2, 1]$ and $[\gamma_1, \gamma_2]$ are disjoint, then $\frac{d^2L(\gamma)}{d\gamma^2}$ is always negative over $(\beta^2, 1]$, and the solutions to $\frac{dL(\gamma)}{d\gamma} = 0$ are local maxima. Thus, in this case, the Lagrangian will either be minimized at $\gamma = 1$, or in the limit as $\gamma \to \beta^2$. The case $\gamma = \beta^2$, which corresponds to $t = q$, need not be considered if higher values of k are later going to be considered, as it coincides with $\gamma = 1$ when k is changed to $k+1$. In any case, the limiting Lagrangian as $\gamma \to \beta^2$ is the actual value of the Lagrangian at $T = (2k-1)q/2 + q$, which can be easily computed from equations 2.12 and 2.12.

When $G(\gamma)$ has real roots and the intervals $(\beta^2, 1]$ and $[\gamma_1, \gamma_2]$ are not disjoint, then $\frac{dL(\gamma)}{d\gamma} = 0$ can have at most one root in the interval of intersection, and that root can be the only local minima for $L(\gamma)$ over $(\beta^2, 1]$. If the interval of intersection is $[\gamma_1', \gamma_2']$, then there is a root if and only if $F(\gamma_1') \leq 0$ and $F(\gamma_2') \geq 0$. In this case, the bisection method can be used to solve the equation $F(\gamma) = 0$. Note that in the case when $\gamma_1' = \beta^2$, the interval of intersection is $(\beta^2, \gamma_2']$. But the bisection method can still be used, as the sign of $\lim_{\gamma \to \beta^2} \frac{dL(\gamma)}{d\gamma}$ (from the right) is the same as the sign of $F(\beta^2)$.

Thus, the following algorithm can be used to minimize the Lagrangian, for given values of β and k:

Algorithm RangeMinLagrangian

  Inputs: $\alpha, \lambda, \beta, k$

  Outputs: γ, L

begin

$\gamma \leftarrow 1$

$L \leftarrow R_{\beta,k}(1) + \lambda D_{\beta,k}(1)$

$L' \leftarrow R_{\beta,k}(\beta^2) + \lambda D_{\beta,k}(\beta^2)$

if $L' < L$ **then**

  $L \leftarrow L'$

  $\gamma \leftarrow \beta^2$

if ($G(\gamma) = 0$ has no real solutions) **then**

  return

$\gamma_1 \leftarrow$ smaller root of $G(\gamma) = 0$

$\gamma_2 \leftarrow$ larger root of $G(\gamma) = 0$

if $\beta^2 > \gamma_1$ **then**

  $\gamma_1 \leftarrow \beta^2$

if $1 < \gamma_2$ **then**

  $\gamma_2 \leftarrow 1$

if $\gamma_2 < \gamma_1$ **then**

  return

if $F(\gamma_1) \leq 0$ **and** $F(\gamma_2) \geq 0$ **then**

  $\gamma_r \leftarrow$ solution to $F(\gamma) = 0$ over $[\gamma_1, \gamma_2]$, using bisection method

  $L_r \leftarrow R_{\beta,k}(\gamma_r) + \lambda D_{\beta,k}(\gamma_r)$

  if $L_r < L$ **then**

    $L \leftarrow L_r$

    $\gamma \leftarrow \gamma_r$

end

Now, given λ and β, the Lagrangian can be minimized successively for $k = 1, 2, \ldots$, and the best value can be chosen. In practice, the Lagrangian stops decreasing after a point, and the iteration can be terminated. Given just a value of λ, the Lagrangian can be minimized over a sufficiently large set of β values. Finally, given a target rate or a target distortion, the bisection method can be used to find λ such that the target is achieved.

Having discussed symbol encoding and scalar quantization, we present overviews of some common image compression techniques, which include the kind of decomposition they use, along with the specific symbol encoding and quantization strategies. We will conclude the chapter with a comparative evaluation of the techniques.

2.7 The JPEG Standard

The JPEG (Joint Photography Experts Group) standard [JPG] is the most commonly used image compression technique today. JPEG uses the discrete cosine transform

(DCT) [ANR74] to decompose the image. Recall that image decomposition serves two purposes: removing redundancy and exposing a relevance order. The pixels in an image are usually highly correlated. For a random vector with a known auto-correlation matrix, the Karhunen-Loeve transform (KLT) [Kar47, Loe60] is a transform that produces random vectors with uncorrelated components. Moreover, the KLT packs most of the signal energy in a small number of components, and orders the components in decreasing order of variance. Thus, the KLT is ideally suited for removing redundancy as well as for relevance ordering. Unfortunately, the KLT is hard to compute, and also requires a computation of the auto-correlation matrix. A practical transform for the purpose of image compression should preferably be image-independent and have a fast implementation, while maintaining the decorrelating property. The DCT performs nearly as well as the KLT (asymptotic equivalence can be shown for certain Markov processes [RY90]), in theory and in practice, and is independent of the image. The DCT is computationally complex enough that its computation for the entire image is prohibitively expensive. In practice, the DCT is applied to smaller units, or blocks, of image data, and in that case it can be done very efficiently, even in hardware. JPEG uses 8×8 blocks, and we present the DCT using that block size.

2.7.1 The Discrete Cosine Transform

The two-dimensional DCT decomposes an image block (in a single color plane) using a fixed set of basis blocks. The basis blocks consist of cosines with different spatial frequencies. The transformed block consists of coefficients for each basis block. For the 8×8 block size, there are 64 basis blocks, indexed by (u, v), for $0 \leq u, v \leq 7$. Let an image block be denoted by f, with individual pixels identified as $f(i, j)$, for

$0 \leq i, j \leq 7$. Then the DCT coefficients $F(u,v)$, for the specific kind of DCT used in JPEG, are computed by taking the dot product of f with the basis block for (u,v) [PM93]:

$$\mathrm{DCT}(f)(u,v) = F(u,v) = \frac{C(u)C(v)}{4} \sum_{i=0}^{7} \sum_{j=0}^{7} f(i,j) \cos \frac{(2i+1)u\pi}{16} \cos \frac{(2j+1)v\pi}{16}. \tag{2.19}$$

The normalizing constants $C(w)$ are:

$$C(w) = \begin{cases} 1/\sqrt{2} & \text{for } w = 0 \\ 1 & \text{for } w \neq 0. \end{cases}$$

The inverse DCT (IDCT) transforms a block of coefficients back into a block of image pixels:

$$\mathrm{IDCT}(F)(i,j) = f(i,j) = \frac{1}{4} \sum_{u=0}^{7} \sum_{v=0}^{7} C(u)C(v)F(u,v) \cos \frac{(2i+1)u\pi}{16} \cos \frac{(2j+1)v\pi}{16}. \tag{2.20}$$

The two-dimensional DCT is separable; it can be computed by first applying a one-dimensional DCT on the rows, and then on the columns. There are many fast algorithms and implementations for computing the DCT. An excellent survey can be found in [RY90].

The coefficient $F(u,v)$ corresponds to the amount of the corresponding spatial frequency present in the image. The coefficient $F(0,0)$ is simply eight times the mean pixel value in the block, and is referred to as the *DC* coefficient. The remaining coefficients are called the *AC* coefficients. As u and v increase from 0 to 7, the coefficient $F(u,v)$ corresponds to higher spatial frequency (that is, finer detail). A higher value of u corresponds to greater fluctuation in the vertical direction, while a higher value of v corresponds to greater fluctuation in the horizontal direction. For example, consider the image block f (taken from an actual image) whose pixel values are:

$$f = \begin{bmatrix} 30 & 35 & 36 & 43 & 59 & 95 & 115 & 134 \\ 38 & 38 & 45 & 63 & 80 & 110 & 126 & 139 \\ 45 & 50 & 56 & 75 & 97 & 121 & 139 & 149 \\ 47 & 57 & 74 & 95 & 114 & 132 & 142 & 153 \\ 54 & 70 & 83 & 105 & 126 & 141 & 146 & 153 \\ 70 & 76 & 94 & 118 & 135 & 143 & 147 & 150 \\ 73 & 87 & 105 & 126 & 145 & 147 & 150 & 149 \\ 84 & 100 & 115 & 133 & 147 & 151 & 148 & 150 \end{bmatrix}.$$

Figure 18 shows this image block, magnified. The DCT coefficients, DCT(f) are

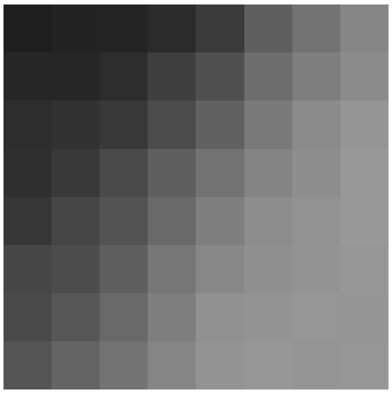


Figure 18: The 8×8 image block used to illustrate the DCT. Each small square corresponds to a single pixel in the block.

obtained as:

$$\mathrm{DCT}(f) = \begin{bmatrix} 819.1 & -263.2 & -8.5 & 1.7 & 0.6 & -4.5 & 0.7 & -4.8 \\ -155.8 & -38.3 & 54.9 & 0.8 & -4.1 & 2.5 & 3.6 & -4.5 \\ -22.1 & 19.2 & 11.4 & -1.2 & -3.3 & -0.2 & 1.1 & -1.0 \\ -17.7 & 0.6 & 2.6 & -0.4 & -0.4 & 0.3 & 1.1 & -0.3 \\ -2.4 & 2.0 & 1.1 & -5.0 & -3.9 & -4.1 & 1.6 & -2.0 \\ -4.1 & -1.8 & 0.7 & -2.6 & -0.5 & -0.1 & -0.5 & 2.2 \\ -0.7 & 2.9 & 6.1 & 0.7 & 0.2 & 0.2 & -0.1 & -0.2 \\ -1.3 & -2.4 & 2.8 & 0.8 & -3.2 & -3.1 & -2.8 & 0.3 \end{bmatrix}.$$

Observe that the DC coefficient has the largest absolute magnitude, and the higher frequency coefficients are all nearly zero. The DCT preserves the total *energy*. That is,

$$\sum_{i,j} f(i,j)^2 = \sum_{u,v} F(u,v)^2.$$

Most images are fairly smooth, and hence most of the energy of the image block is captured by the low-frequency coefficients in the transformed block. Thus, the relevance ordering implicitly imposed by the DCT is that the lower frequency coefficients are typically greater in absolute magnitude than the high frequency coefficients. Moreover, the human visual system is less sensitive to higher frequencies than lower frequencies [VB67]. These two facts can be used to achieve high compression, by retaining only a coarse approximation of the coefficients, i.e., by quantization.

2.7.2 JPEG Quantization and Encoding

The JPEG standard does not specify any fixed color space. Usually, color images are converted to the (Y,Cb,Cr) space, and the chrominance planes are subsampled by a factor of two horizontally as well as vertically. Conceptually, further encoding of each plane is done independently (the encoded blocks are interleaved in practice). Thus, we can describe the JPEG compression steps in terms of a single color plane I, of width W and height H.

The image I is first divided into 8×8 non-overlapping blocks. To each block f, the DCT is applied to get the 8×8 block F of DCT coefficients. MT-U-M quantization of each coefficient is done. Recall that MT-U-M quantization uses a uniform step size q, and each input is quantized by dividing it by q and rounding to the nearest integer. Reconstruction is done in the decompression process by multiplying each quantized value by q.

Since different DCT frequencies have different perceptual significance, and since lower frequencies are likely to have greater absolute magnitudes, JPEG uses different step sizes for quantizing different coefficients. An 8×8 table of integers, called the *quantization table* Q, specifies the quantizer step size for each coefficient. JPEG allows Q to have entries in the range 1 to 255.

The block of quantized DCT coefficients, F_Q, is given by,

$$F_Q(u, v) = F(u, v) /\!\!/ Q(u, v),$$

where $/\!\!/$ represents division followed by rounding to the nearest integer. The quantization table Q is the tuning parameter for JPEG compression, and is included in the compressed image so that the decoder can use it for dequantization. Greater values of $Q(u, v)$ result in poorer quality and lower rate. For the example block f given previously, the DCT coefficients block F, when quantized by a quantization table Q with all entries $Q(u, v)$ equal to 10, results in the quantized block given by:

$$F_Q = \begin{bmatrix} 82 & -26 & -1 & 0 & 0 & 0 & 0 & 0 \\ -16 & -4 & 5 & 0 & 0 & 0 & 0 & 0 \\ -2 & 2 & 1 & 0 & 0 & 0 & 0 & 0 \\ -2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}.$$

Observe that most of the quantized coefficients are zero. The DCT did a good job of exposing the relevance order by packing most of the signal energy in low-frequency coefficients, which are the only ones with non-zero quantized values. It is apparent that the transformed and quantized blocks can be compressed tremendously.

There are two modes of symbol encoding allowed in JPEG: the arithmetic coding mode and the Huffman coding mode. The arithmetic coding mode uses the QM-coder to code the sequence of quantized coefficient blocks, by using a dynamic probability model for each $F_Q(u, v)$ [PM93]. In practice, the arithmetic coding mode is not used very widely, because arithmetic coding is a patented technique, and the legal issues involved have led implementors to choose the other mode, the Huffman coding mode, which gives marginally higher rates.

The DCT coefficients are largely uncorrelated, as the DCT performs similarly as the KLT, on certain Markov processes which are fair approximations of image pixel blocks [RY90]. As mentioned before, Huffman coding may perform badly when symbol probabilities are not negative powers of two. JPEG solves this problem by grouping together runs of zeros into composite symbols, prior to Huffman coding. This is done by listing the quantized coefficients in each block in a fixed "zig-zag" order, and then rewriting this zig-zag sequence as a sequence of composite symbols of the form (run-length, next-value). Here *run-length* is the number of consecutive zeros encountered before the next non-zero value, *next-value*. The zig-zag ordering lists low-frequency coefficients before high-frequency coefficients, thus making long runs of zero more likely. The composite symbols are then coded using Huffman coding. This is a slightly simplified description of JPEG symbol encoding, the exact details can be found in [PM93].

The compression steps for JPEG are summarized in Figure 19. For decompression,

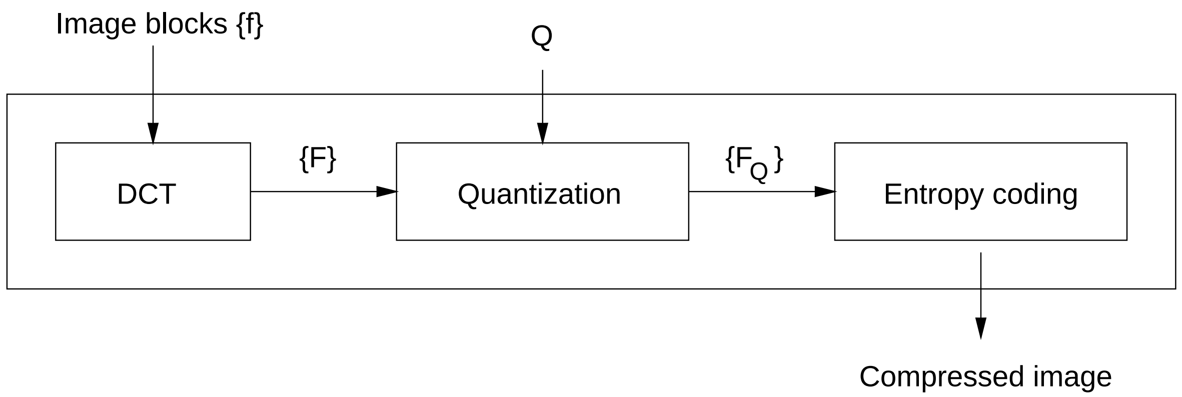


Figure 19: JPEG compression steps.

these steps are simply reversed. The entropy-coded stream is decoded to exactly recover the sequence of quantized coefficient blocks. Each quantized block F_Q is dequantized to get the decompressed approximation F' of the original coefficient block as:

$$F'(u,v) = F_Q(u,v) \cdot Q(u,v).$$

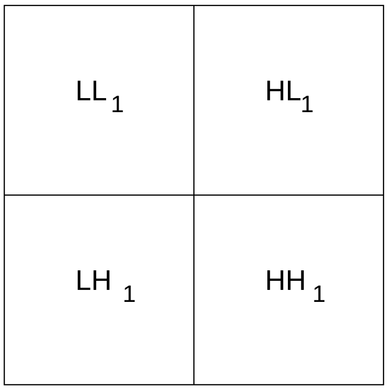
Finally, IDCT is applied to each F' to get the decompressed image blocks $f' = \mathrm{IDCT}(F')$.

The rate-quality tradeoff in JPEG compression is determined by the quantization table Q. Smaller entries lead to higher quality but poorer compression. The RD-OPT algorithm developed by us addresses the previously open problem of quality-control in JPEG (that is, choosing the tuning parameter Q to optimize rate-quality tradeoff while achieving any rate/quality target). RD-OPT will be described in detail in Chapter 4.

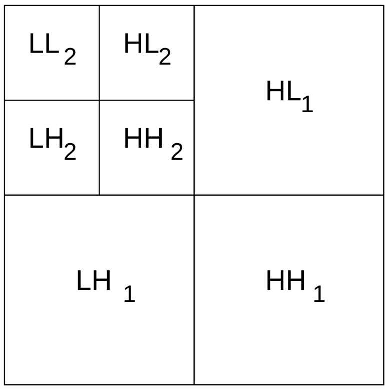
2.8 Image Compression with Wavelets

Image compression using wavelet transforms has received considerable attention, and is likely to become standardized in the near future (the "JPEG-2000" standard is expected to be primarily wavelet based). A *wavelet* ("little wave") is, roughly speaking, an oscillating function modulated by a decaying envelope. A discrete wavelet transform (DWT) uses scalings and translations of a "mother" wavelet as its set of basis functions. Fine details are captured by coefficients of smaller-scale basis functions, while global trends are captured by coefficients of the larger-scale ones. Several mother wavelets, and corresponding DWT's are discussed in [ABMD92].

From an image compression perspective, it is convenient to view the DWT coefficients as a *hierarchical subband representation* of the image. The underlying mother wavelet defines two filters, $l(\cdot)$ (a low-pass filter) and $h(\cdot)$ (a high-pass filter). The DWT can be implemented on an image plane as follows. To begin the decomposition, $l(\cdot)$ and $h(\cdot)$ are applied to each row, and the results are downsampled by 2. (The filters must satisfy certain conditions to ensure that this step is lossless; the reader is referred to [ABMD92] for details.) Next, the same process is applied to the columns. This decomposes the image into four subbands, as shown in Figure 20. The subbands LH_1, HL_1, and HH_1 represent the DWT coefficients at the finest scale. The subband



(a) First stage



(b) Second stage

Figure 20: First two stages of a discrete wavelet transform.

LL_1 is the original image itself, at a lower resolution (since it is obtained by applying a low-pass filter along both directions). The coefficients in the subbands LH_1, HL_1, and HH_1 are largely uncorrelated, and are not processed further. Each coefficient corresponds to a roughly 2×2 area of the original image, at this finest scale. The DWT coefficients at the next coarser scale are obtained by applying the $l(\cdot)$ and $h(\cdot)$ filters to the LL_1 subband, dividing it into four more subbands. Each coefficient at this scale corresponds to a roughly 2×2 area of the LL_1 image, that is, a roughly 4×4 area of the original image. This step is also illustrated in Figure 20. This process is an example of *multiresolution analysis*, as, at each resolution, the subbands other than LL capture the details at that resolution, while the LL subband represents the image at the next coarser resolution.

For a square image whose width is 2^k, this process can be repeated until, at the coarsest scale, there are just four coefficients, one each in LL_k, LH_k, HL_k, and HH_k. In practice, image dimensions are increased by padding with rows and columns, until both height and width are multiples of 2^k and the multiresolution analysis is applied for k stages, with k typically chosen as 5 or 6.

Note that the DWT can be easily implemented, as filters $l(\cdot)$ and $h(\cdot)$ with fairly small "tap" sizes (i.e., the number of inputs used to compute an output) perform very well on images. The variations of DWT commonly used in image compression can be expressed as transformation matrices that are unitary or "almost unitary" [Sha93, SP96a]. Thus, the DWT can be assumed to preserve L_2 norms (mean-square intensities), for all practical purposes.

The high-pass filter in the DWT achieves the decorrelating property desired in image-compression. In early wavelet-based techniques, the relevance ordering used for quantization was simply, "each coarser subband has more energy and is more important visually, for common images" (in [ABMD92], for example). DeVore *et al* observed that if quantization is done by simply discarding a subset of the coefficients, then the MSE will be lowest by retaining the coefficients with the largest absolute magnitude [DJL92]. Moreover, they showed that when uniform scalar quantization is used on each level of coefficients, the quantizer step size should be increased by a factor of two as one goes from a level to the next finer level, when MSE is used as the error criterion.

Shapiro observed and utilized an important structure inherent in the wavelet coefficients [Sha93]. Shapiro's results constituted an important breakthrough, and led to the development of compression techniques whose performance greatly surpassed all other previous techniques. Shapiro's idea is based on an empirical observation, that if a coefficient at some coarse scale is small in absolute magnitude, then it is likely to be small in absolute magnitude at the next finer scale too. We can organize the coefficients in a tree structure, with groups of four coefficients corresponding to a particular spatial location and subband being the children of the coefficient for the same spatial location and subband at the next coarser scale. At the coarsest level, each LL coefficient is the parent of three coefficients (one each in the LH, HL, and HH subbands)[4]. The tree structure is illustrated for an 8×8 image, in Figure 21. In terms of the tree structure,

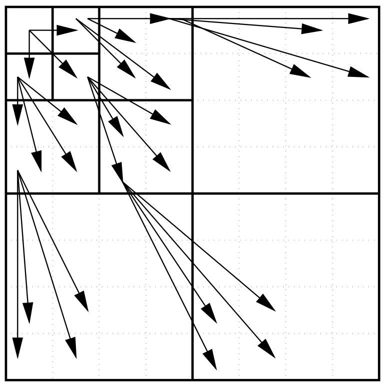


Figure 21: The tree structure of wavelet coefficients.

Shapiro's observation is that if a coefficient c is absolutely less than some threshold T, then all its descendants are also likely to be absolutely less than T. If quantization is done simply by comparing with a threshold, then the encoder can utilize this *zerotree structure* by using a special symbol to indicate that a particular coefficient is zero along with all its descendants. The descendants do not have to be encoded separately, in such cases. An image is compressed by using a succession of thresholds, and encoding the zerotrees resulting from the thresholds using Huffman or arithmetic coding. The performance of the resulting coding technique (called the EZW technique) was comparable or superior to other image compression techniques, and had the nice property that the compressed image at a particular rate was simply a prefix of the compressed image at any higher rate. Codes with this property are called *embedded* codes, and are very useful for progressive transmission of images.

[4]Thus, the structure is really a *forest,* except for square images whose width is a power of two.

Said and Pearlman extended Shapiro's technique by simplifying the zeroing thresholds (they used powers of two, which correspond to a bit-planes ordering), but using a more sophisticated technique for encoding the zerotree structure, called SPIHT (Set Partitioning In Hierarchical Trees) [SP96a]. The SPIHT encoder has the best or nearly the best rate-PSNR performance among all current image compression techniques.

2.8.1 Quantization and Tuning Parameters in SPIHT

SPIHT is also an embedded coding technique: the compressed image at any rate is simply a prefix of the compressed image at any higher rate. The quantization is implicit: the level of reconstruction of a coefficient depends upon the bit-planes encoded up to a given rate. The tuning parameter is, trivially, just a rate or quality target. In case of a rate target, the encoder simply continues its encoding process until the desired rate has been met. For distortion-based quality metrics too, the encoder knows when to stop, by keeping track of the current distortion and adjusting it incrementally. SPIHT was designed to minimize distortion, or MSE. Even with fairly complicated quality metrics, the SPIHT encoder can meet a target by periodically stopping to evaluate the quality.

The reason that SPIHT and EZW perform so well, even with such simple quantization strategies, is that the wavelet transform, together with the zerotree structure, does an excellent job of redundancy removal and relevance ordering. In fact, the performance of the SPIHT algorithm is only marginally lowered even if no entropy coding is used [SP96a].

2.9 Vector Quantization

In vector quantization (VQ), vectors of signal values are quantized together. When applied directly to image pixel intensities, image blocks are the vectors to be quantized. Let the block size (= block-width $\times$ block-height) be denoted by n (for color images, VQ can be applied to each plane, or composite vectors can be formed using some or all color planes). If the pixel values are integers in the range 0 to M, then the total number of possible vectors is $(M+1)^n$. The idea in VQ is to use a small set of representative vectors, and code each input vector as its closest approximation in this small set. The set of representative vectors is called a *codebook.* The *optimal codebook,* for a given image, N, and n, is one which minimizes the distortion resulting from approximating each image block by its closest codebook entry.

An important result from rate-distortion theory is that as $n \to \infty$, the quantizer output entropy approaches $\log_2 N$, and the distortion (using optimal codebooks) approaches the theoretical rate-distortion bound for that entropy [GG92]. Thus, in theory, with sufficiently large block size, VQ becomes so efficient that entropy-coding is obviated. In practice, implementation complexity becomes prohibitively high very soon as the block size is increased.

If image-specific codebooks are used, then the codebook needs to be transmitted along with the compressed image. The codebook size is roughly $Nn \log_2 M$ bits, and for large n, this overhead is substantial. A common strategy is to use a shared codebook for large classes of similar images.

Designing good codebooks is an important issue in VQ. The most commonly used approach is the Linde-Buzo-Gray (LBG) algorithm [LBG80], which is similar to the Lloyd-Max algorithm for scalar quantizer design. The LBG algorithm uses a *training* *set* of images to optimize the codebook. Initially, some starting codebook is used, and the training set vectors are clustered according to their closest codebook entries. The next, better codebook is constructed by replacing each codebook entry by the centroid of its cluster. This process is repeated until the distortion drop falls below some threshold. The LBG algorithm can get trapped in local optima, and it is important to use a good starting codebook with spread-out entries.

In practice, for block sizes that are amenable to efficient implementation, the rate-quality performance of VQ is considerably lower than that of JPEG or wavelet-based techniques. The primary advantage of practical VQ techniques is that decoding is extremely fast, as it just involves table look-ups.

Encoding an image using a given codebook is also expensive, as all the entries need to be examined in order to find the best match for each image block. By imposing more structure on the codebook, the encoding time can be improved. For example, tree-structured VQ (TSVQ) techniques use a codebook where the search for the best match can be done by following a decision path along a tree with N leaves. Hierarchical VQ techniques make the encoding even faster by reducing it too to a sequence of table look-ups [CVC95]. Imposing structure on the codebook improves encoding time, but the restrictions imply poorer rate-quality tradeoffs.

2.10 Fractal-Based Image Compression

There is a very large degree of self-similarity in nature, and hence in natural images. A *fractal image* is an image in which every "piece" of the image can be obtained by transforming some other "piece." Thus, a fractal image can be obtained by applying a suitable transformation to itself. When this transformation is contractive[5], the image is the fixed point of the transformation, that is, the image can be obtained by repeatedly applying the transformation to any arbitrary image [Bar93]. Figure 22 shows an example of fixed-point convergence, using a crude fractal approximation of the 256×256 grayscale image, Lena.

Since a tractable encoding algorithm cannot afford to search over arbitrary transformations on arbitrary domains and ranges within a given image, constraints have to be placed on the kind of transformations, domains, and ranges that are allowed [Jac92]. These constraints have taken away a lot from the theoretical power of fractal compression. We have shown that even for highly self-similar images, fractal compression may perform poorly [RFT94]. Thus, even though they are computationally expensive, typical fractal compression techniques do not perform as well as other standard image compression schemes [JFB92]. The idea of fractal compression came from Barnsley, and his company, Iterated Systems, makes compression software with better performance than the published techniques, but the Iterated Systems techniques have not been published.

The contractivity requirement is merely a technicality for enabling decompression via convergence to the fixed point. We have investigated ways to exploit self-similarity in images without requiring the transformations to be contractive [FPR95]. The techniques developed performed well at very low bit-rates, when compared to block-based coding schemes such as JPEG, but their computation complexity is very high, and they are primarily of theoretical interest.

[5] A function f is said to be contractive if for some distance metric d, $\forall x_1, x_2\ d(f(x_1), f(x_2)) < d(x_1, x_2)$.

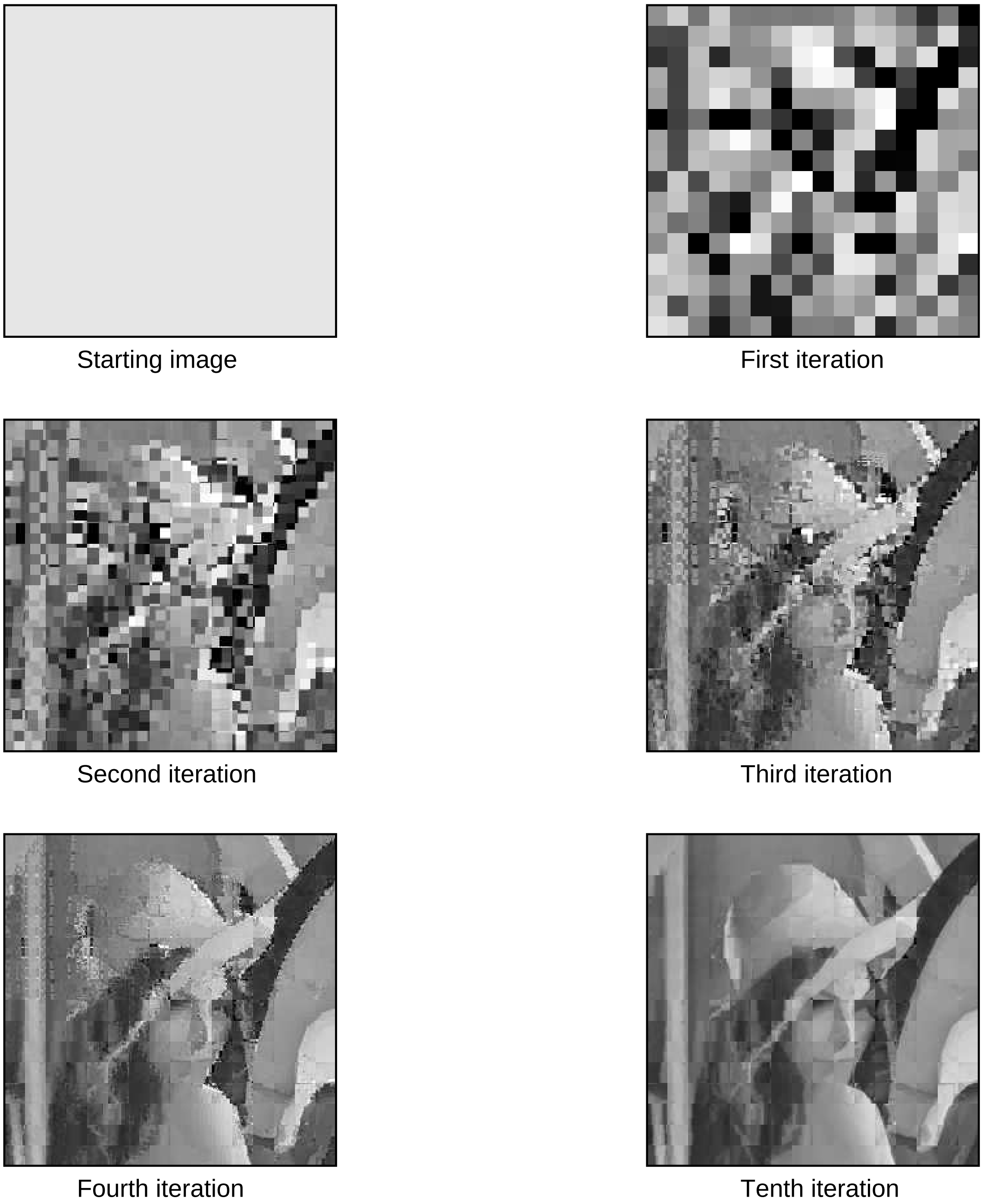


Figure 22: Fractal decompression via convergence to fixed-point.

2.11 Progressive Compression

The wavelet-based compression techniques, SPIHT and EZW, are embedded coded techniques, and are very useful for progressive transmission. Recently, embedded compression techniques using reversible wavelet transforms have also been developed [SP96b, ZASB95]. Reversible transforms can be implemented exactly using fixed-point arithmetic. The compression techniques S+P-SPIHT [SP96b] and CREW [ZASB95] are *lossless* if the whole compressed image is used. Even when only a prefix of the whole compressed image is used, the decompressed image is of comparable quality to the best lossy compressed image at that rate. Thus, a single compressed image can be pruned to achieve any target rate or quality. This property is very useful for realizing the QCLIC framework.

JPEG also supports progressive compression, but the coding is not embedded. In the progressive mode of JPEG, the DCT coefficients can be grouped into “scans”. Scans can also progressively select bit-planes of coefficients. A prefix of a coded image may only contain the first few coefficients of each block and/or the most significant bit-planes of the coefficients. The rate-quality tradeoff achieved by using a pruned progressive JPEG image is not as good as that obtained by JPEG optimized for its rate. Still, for the QCLIC framework, the progressive mode of JPEG is very useful, as it allows quality-control to be implemented by storing only a few (even one) compressed images, and pruning to achieve intermediate rates and qualities.

2.12 A Comparison of Various Image Compression Techniques

We conclude this chapter by listing and comparing the main features of the compression techniques discussed. Fractal compression scores high only on mathematical elegance, and is omitted from this comparison.

Rate-quality tradeoff. State-of-the-art wavelet-based techniques perform the best, for nearly all quality metrics. At moderate to high bit rates, there isn't much difference subjectively between JPEG and wavelet-based techniques, but at low rates, the subjective quality is better with wavelets. Quality with VQ is substantially lower than both JPEG and wavelets.

Complexity. Fast software as well as hardware implementations are available or possible for DCT and DWT, forward and inverse. In case of the DWT, sophisticated implementations that do not require image-sized memory buffers are also possible [ZASB95]. Entropy coding can be simplified for implementation with a slight degradation in rate. The decoding complexity is lowest for VQ. Encoding complexity is lowest for hierarchical VQ.

Error tolerance. If the compressed image is to be transmitted over a noisy channel, susceptibility to channel errors is an important issue. In block based compression techniques such as JPEG and VQ, the effect of a single bit error will typically be restricted to a single block. For wavelet-based techniques, a single bit error can manifest itself over the entire image.

Random access. To access a small area of the image from compressed JPEG data, the compressed data must be parsed to locate the corresponding blocks. Still, the JPEG syntax localizes blocks corresponding to a small area to some extent. When random-access is very important, indices can be built to quickly locate blocks. In case of SPIHT, EZW, and CREW, even the various bits of the coefficients corresponding to a small area are scattered all over the compressed data. For achieving random access with wavelet-based techniques, the sophisticated restructuring associated with bit-plane ordering and zerotrees needs to be sacrificed. With VQ, random access (up to block granularity) is trivial. In general, random access can be more easily implemented when fixed-length codewords are used for entropy coding.

Decoding at multiple resolutions. The wavelet transform naturally decomposes an image into various resolutions. As with random access, the problem is that because of the sophisticated organization techniques, extracting the coefficients only up to a target resolution is complicated. However, the situation is not too bad, as the zerotree structure gives information about finer resolutions in the coarser-resolution part of the compressed data. This means that when extracting the image up to a particular resolution, some "unnecessary" information about higher resolutions will also be extracted. The only complication is that arising from separation of bit-planes (or threshold significance, for EZW), which can be alleviated by building a small index. In case of JPEG, the DC coefficients of the blocks give a low-resolution image (downsampling by 8 in both directions). For extracting just this low-resolution image, the compressed data needs to be parsed, but the inverse DCT need not be applied, as the DC coefficient is the mean of the 8×8

block (scaled by a constant). With progressive JPEG, the DC coefficients are always constrained to be in a single scan all by themselves, and their extraction is cheaper. If other resolutions are also needed, then they must be separately created and compressed, for JPEG. The JPEG syntax does allow this multiresolution representation, in the so-called *hierarchical mode*. With VQ, multiple resolutions need multiple codebooks, which can be incremental.

Progressive transmission. Wavelet-based techniques support progressive transmission naturally. With JPEG, it can be done in the progressive mode, but the rate-quality performance of intermediate points (i.e., prefixes) is not the best that can be achieved at their rates. With VQ, a sequence of codebooks with a tree structure can be used. Such a tree can be efficiently designed by starting with a codebook of size 2, and repeatedly "splitting" each codebook entry [LBG80].

Compatibility. Currently, JPEG is the only image compression standard. A JPEG-based file format, called FlashPix, is evolving as an internet standard (information about FlashPix can be found at the Kodak web-site, `www.kodak.com`). A wavelet-based JPEG standard is expected around the year 2000. Applications that choose non-standard compression methods are likely to have limited compatibility across platforms and users.

Sensitivity to image type. The DCT and DWT work fairly well across all kinds of images. For VQ, if the image is significantly different from those in the training set used for codebook generation, quality may be poor.

Chapter 3

The QCLIC Framework

This chapter describes the QCLIC framework for image compression in detail. We first describe the general structure of information associated with an image in the QCLIC framework. This is followed by a discussion of the enabling technologies needed for specific compression techniques, to generate the structure needed for the QCLIC framework. The QCLICS algorithm for achieving the QCLICS functionality for sets of QCLIC-Images is described. Finally a few test-bed implementations, a production-mode image compressor, an image browser, and an image server, are described.

3.1 The Structure of a QCLIC-Image

We first present the QCLIC framework from a conceptual point of view, to define its elements formally. A *compression method* $\mathcal{C}$ is a specific compression algorithm, along with an implied decompression algorithm $\mathcal{D}_{\mathcal{C}}$. A QCLIC-Image, I, consists of the following pieces of derived information. For each compression method, $\mathcal{C}$, and each quality metric, $\mathcal{Q}$, QC-Crv$(\mathcal{C}, \mathcal{Q})$ is the optimal rate-quality curve for I, using $\mathcal{C}$ and $\mathcal{Q}$. The curve QC-Crv$(\mathcal{C}, \mathcal{Q})$ is made up of a discrete number of points. The curve has a minimum rate point and a maximum rate point. The separation between successive points is $1/WH$ bpp on the rate axis, where W and H are the width and height of the image (if a particular rate is not achievable exactly, we can pad the previous lower-rate compressed image with zeros). Each point on this curve, denoted by QC-Pt($\mathcal{C}, \mathcal{Q}, R, Q$), has an associated value θ of the tuning parameter, such that $\mathcal{C}(I, \theta)$ has rate R (possibly after padding) and quality Q. Each point is optimal, that is, there is no θ' with the rate of $\mathcal{C}(I, \theta')$ no more than R but the quality of $\mathcal{C}(I, \theta')$ greater than Q.

If we look at just a prefix of the compressed image, with S ($\leq RWH$) bits, then that corresponds to a rate $R' = S/WH$ ($\leq R$) bpp. Assume for a moment that this prefix can also be decompressed, with quality Q'. Then, associated with each point QC-Pt($\mathcal{C}, \mathcal{Q}, R, Q$), there is another "internal" rate-quality curve, denoted by QC-Pt-Crv($\mathcal{C}, \mathcal{Q}, R, Q$), made up of rate-quality points QC-Pt-Pt($\mathcal{C}, \mathcal{Q}, R, Q, R', Q'$), with $0 \leq R' \leq R$ (in steps of $1/WH$ bpp). Figure 23 shows these elements which constitute a QCLIC-Image object.

When the decompression algorithm $\mathcal{D}_\mathcal{C}$ supports progressive decoding, QC-Pt-Crv and QC-Pt-Pt are well-defined. Even if that is not the case, in practice, we can always stop the decompression algorithm before it has processed the entire compressed data stream, even if that implies that no information at all has been received about some pixels. To be rigorous, we define the quality of a prefix ρ of $\mathcal{C}(I, \theta)$ as follows. Let l_ρ be the least number of bits such that there is a binary string σ of length l_ρ such that $\rho\sigma$ can be decoded by the decompression algorithm $\mathcal{D}_\mathcal{C}$ as a $W \times H$ image with the same number of color planes and same pixel-value range as I. Clearly, at least one such σ exists, as ρ can always be extended to form $\mathcal{C}(I, \theta)$. Call such a string σ a *minimal valid suffix* of ρ. Then the quality of ρ is defined as the quality of $\rho\sigma_\rho$, where σ_ρ is the lexicographically smallest minimal valid suffix. Observe that if the decompression algorithm does support progressive decoding, then the definition is consistent, as the empty string is a minimal valid suffix.

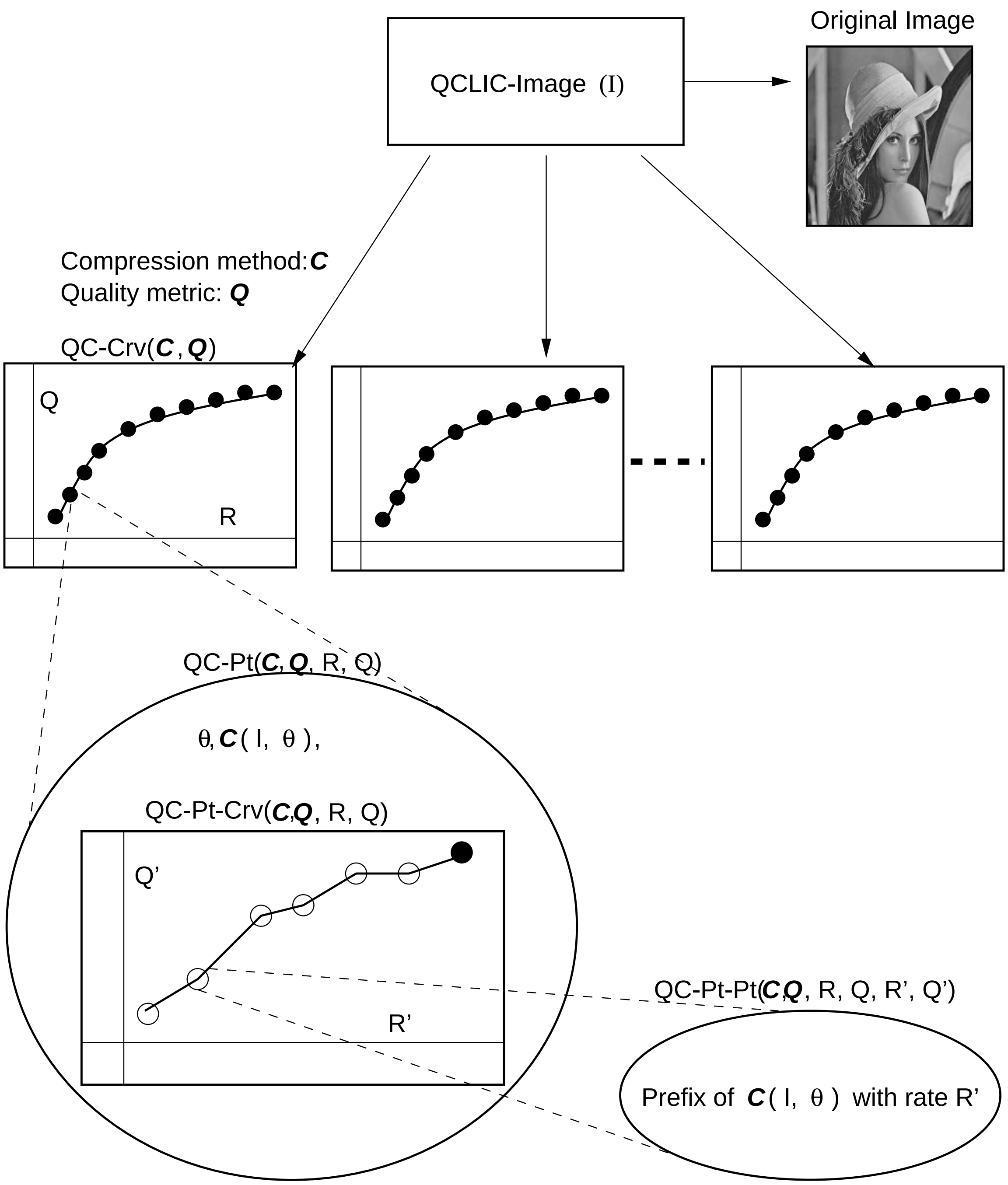


Figure 23: Structured elements of a QCLIC-Image.

A *realization* of the QCLIC framework, for an image I, compression method $\mathcal{C}$, and quality metric $\mathcal{Q}$, is a way to approximate the elements of the QCLIC-Image structure for I, $\mathcal{C}$, and $\mathcal{Q}$, using suitable enabling technology. We can now define the semantics of the methods *QCLIC-GetCurve* and *QCLIC-Compress*, introduced in Chapter 1.

3.1.1 The QCLIC-Image Methods

For a QCLIC-Image I, the result of *QCLIC-GetCurve*$(\mathcal{Q}, \mathcal{C},$ *curve-constraints*$)$ is derived from the realization of the curve QC-Crv$(\mathcal{Q}, \mathcal{C})$ using the constraints specified in *curve-constraints*. The parameter *curve-constraints* can specify any or all of:

1. Minimum and maximum rate, R_l, R_h. The returned curve should be for the rate range $[R_l, R_h]$. Defaults are the minimum and maximum rates achieved by $\mathcal{C}$ on I.

2. Minimum and maximum quality, Q_l, Q_h. The returned curve should be for the quality range $[Q_l, Q_h]$. Defaults are the minimum and maximum qualities achieved by $\mathcal{C}$ on I. When both rate and quality extrema are specified, the requested range is that common to both. Also, the range thus specified will be clipped if it extends beyond that actually achievable.

3. Number of curve points, N_P. Default is the maximum possible number of points (which depends on how the QCLIC framework is realized).

4. Rate tolerances, δ_R^-, δ_R^+. These specify acceptable errors in the reported rates. If a rate is reported as R, but actually achieved as R_A, then these constraints enforce the condition, $R_A - \delta_R^- \leq R \leq R_A + \delta_R^+$. This is useful, as realizations of the QCLIC framework may often store *estimates* of rate and quality for a

large number of points. Defaults are ∞, which simply allow the method to use the available estimates without bothering to verify their accuracy (which might require actual compression).

5. Quality tolerances, δ_Q^-, δ_Q^+. These are tolerances for quality, similar to the rate tolerances describe above. Defaults are ∞.

We will use the convention of denoting complex objects such as *curve-constraints* by listing the values of fields inside square brackets (for example, $[R_l = 0, R_h = 1.0, N_P = 10]$).

The minimum and maximum rates and qualities restrict the curve to a range of rates, $[R_1, R_2]$, which is divided nearly equally (the rate at each point has to be a multiple of $1/WH$) into the number of points specified, N_P. *QCLIC-GetCurve* returns a sequence of rate-quality pairs, one for each point. For simplifying the implementation of *QCLIC-Compress*, *QCLIC-GetCurve* also returns a "point-identifier," $\mathbf{id}_i$, for each rate-quality point returned. The point-identifier is unique for each rate-quality point possible in the QCLIC realization for I, $\mathcal{C}$, and $\mathcal{Q}$. ($\mathbf{id}_i$ can be used as a "handle" for subsequent compression using *QCLIC-Compress*.) Thus, *QCLIC-GetCurve* returns a sequence of the form $\{(R_i, Q_i, \mathbf{id}_i) \mid i = 0, \ldots, N_P - 1\}$.

The method *QCLIC-Compress*($\mathcal{Q}, \mathcal{C}$, *target*, *compression-constraints*) returns I compressed using method $\mathcal{C}$, according to the restrictions imposed by the parameters *target* and *compression-constraints*. The parameter *target* can be one of the following:

Rate target, R_T, or, Quality target, Q_T, or, Point-identifier, $\mathbf{id}_T$.

In case of a rate target, R_T, the rate of the compressed image should be no more than R_T (within a tolerance possibly specified in *compression-constraints*). In case of a quality target, Q_T, the quality of the compressed image should be at least Q_T (within a possible tolerance). A point-identifier target, $\mathbf{id}_T$, is simply a handle previously returned by *QCLIC-GetCurve* or *QCLIC-Compress*.

The constraints specified in the parameter *compression-constraints* can be any or all of:

1. Starting bit offset, B_s, specifying an "internal piece". This specifies that the whole compressed image need not be returned; only the piece beginning at bit offset B_s should be returned. This, along with the next parameter, is useful for incremental retrieval. Default is 0. Note that a bit offset is simply another way of expressing rate, but it is free of floating-point imprecision.

2. Ending internal piece rate, R'_e, or bit offset, B_e, or quality, Q'_e. This specifies where the returned piece of the compressed image should be terminated. Defaults are the values corresponding to the whole compressed image. When an ending quality, Q'_e is specified, tolerances $\delta^-_{Q'_e}$ and $\delta^+_{Q'_e}$ can also be specified (default ∞). Note that the rates and qualities used for these piece-specifying parameters are simply the rates and qualities on the internal curve, QC-Pt-Crv.

3. Rate tolerances, δ^-_R, δ^+_R. These define an acceptable level of error in meeting a rate target. Defaults are ∞.

4. Quality tolerances, δ^-_Q, δ^+_Q. These define an acceptable level of error in meeting a quality target. Defaults are ∞.

Along with the compressed image ($\mathcal{C}(I)$), *QCLIC-Compress* also returns its exact rate (R), quality (Q), point-identifier ($\mathbf{id}$), and the starting and ending bit-offsets

(B_s, B_e). Thus, the returned value from *QCLIC-Compress* can be represented as $(\mathcal{C}(I), R, Q, \mathbf{id}, B_s, B_e)$.

3.1.2 An Illustrative Example

Consider a simple, two-step image browser that uses some progressive compression method $\mathcal{C}$. Suppose that a quality value Q (in some metric $\mathcal{Q}$) corresponds to visually lossless quality, but the browser only retrieves a piece with a low quality Q_B first, and retrieves the whole compressed image only if the low quality prefix looks interesting to the user. Using QCLIC methods, the browser can be expressed as:

$(\rho_1, R, Q, \mathbf{id}, B_{s1}, B_{e1}) \leftarrow I.\mathit{QCLIC\text{-}Compress}(\mathcal{C}, \mathcal{Q}, [Q_T = Q], [Q'_e = Q_B])$

Decompress and show the image $I_1 = \mathcal{D}_{\mathcal{C}}(\rho_1)$

if (I_1 looks interesting to user) **then**

$(\rho_2, R, Q, \mathbf{id}, B_{s2}, B_{e2}) \leftarrow I.\mathit{QCLIC\text{-}Compress}(\mathcal{C}, \mathcal{Q}, [\mathbf{id}_T = \mathbf{id}], [B_s = B_{e1} + 1])$

Decompress and show the image $I_2 = \mathcal{D}_{\mathcal{C}}(\rho_1 \rho_2)$

Note that the point-identifier ($\mathbf{id}$) returned by the first call is given as the target to the second call. Also, the bit after the ending bit offset returned by the first call (B_{e1}) is used as the starting bit offset for the second call.

3.2 Compression Methods and Enabling Technologies

To realize the QCLIC framework for a given compression method $\mathcal{C}$ and quality metric $\mathcal{Q}$, we need fairly efficient tuning parameter selection. There are several issues that an enabling technology must address for this, and the complexity of the techniques used is constrained by the schedule of realization. When the QCLIC framework is realized "off-line," complex techniques can be adopted, but if the realization has to be "on-the-fly," then the simplest approaches may have to be used. For example, if images are to be archived and made available to remote users, the QCLIC framework can be realized nearly optimally by pre-computing and storing all the necessary information. On the other hand, a digital camera must use its limited storage space and time to do the best it can, every time a new picture is shot. There are three broad concerns for an enabling technology: mapping between quality and tuning parameters, mapping between rate and tuning parameters, and optimization of rate-quality tradeoff.

For complex quality metrics (such as PQS), the mapping between tuning parameters and quality can be obtained only by going through the compression-decompression process and evaluating the quality. For distortion-based quality metrics, the exact mapping, or a close approximation, can often be predicted more easily.

The mapping between rate and tuning parameters is easily obtained when fixed-length codes are used, or (trivially) when rate itself is a tuning parameter (such as in embedded codes). Otherwise a rate-model may be used as an approximation.

Optimization of the rate-quality tradeoff also varies in complexity. If both rate and quality (for the metric being used) can be easily computed directly from the tuning parameters, in the absence of other techniques, a "best-gradient" search strategy may be reasonable. For complex quality metrics, instead of optimizing the metric directly, it might be more efficient to optimize a simpler metric and simply re-calibrate the resulting quality values. For better performance, the simpler quality metric can be designed to have good correlation with the complex metric. For example, for using PQS with any orthonormal transform-based coding method, distortion can be weighted in the transform domain, with the weights calculated to maximize the correlation with PQS.

Optimization is trivial when the compression method itself is such that all possible tuning parameters are completely ordered in terms of quality. For example, in SPIHT compression, there is no parameter choice to be made other than a "stopping rate," and for any (reasonable!) quality metric, quality will always increase as the stopping rate is pushed higher. Even for the JPEG compression technique, we can use compression *methods* that produce JPEG images using only a restricted number of ordered possibilities for the quantization tables. Thus, even when the optimization problem is hard for a method, it can be made easier by restricting the parameter set. Of course, the resulting solutions will be suboptimal in terms of the original, unrestricted method.

3.2.1 JPEG Image Compression

The RD-OPT algorithm (to be described in Chapter 4) developed by us is an efficient enabling technology for JPEG and other DCT-based compression methods for a number of quality metrics. RD-OPT produces an image-specific rate-quality curve that is nearly optimal, and maps it to the tuning parameters needed. Further, RD-OPT also produces an internal rate-quality curve for each point. The rates given by RD-OPT are fairly good estimates that are computed using a rate model. The qualities given by RD-OPT are very accurate. The time taken to run RD-OPT is about two seconds, for typical images and settings. Moreover, the complexity of RD-OPT can be fine-tuned to meet application constraints, while scaling down rate-quality performance fairly. We now describe some possibilities for using RD-OPT as an enabling technology to realize the QCLIC framework, differentiating using the time when RD-OPT is applied.

"Off-line" and "on-the-fly" refer to the time when RD-OPT is used in relation to the times when the QCLIC methods, *QCLIC-GetCurve* and *QCLIC-Compress* are used. The distinction is not very strict, as the first call to a QCLIC method might invoke RD-OPT on-the-fly, with the QCLIC structure created being used again for later calls.

Off-line RD-OPT

For applications such as image archives, RD-OPT can be run off-line to generate a sequence of nearly optimal tuning parameters spanning the entire rate range possible. For each point, the JPEG compressed image and the exact rate and quality values can be precomputed and stored, along with the exact internal curves. If the original image can be retained, and fast compression is available (in hardware, for example), just the tuning parameters and the exact values of rates and qualities may be retained. When storage space is scarce, RD-OPT can be used to generate a single nearly optimal, high-quality compressed image, and its internal rate-quality curve can be used to prune it to meet any smaller rate/quality target. If progressive JPEG is used, pruning is trivial. Otherwise, pruning involves parsing, removing some coefficient bits, and re-packing (which is substantially cheaper than re-compressing). This can result in rather poor qualities at low rates. If more than one compressed image can be stored, the quality at low rates can be improved: a target can be met by pruning the closest compressed image with better rate/quality.

RD-OPT On-the-fly

If the application does not require rate targets to be met extremely accurately, RD-OPT can be run whenever a *QCLIC-Compress* call is made. If the tuning parameter returned by RD-OPT does not meet the target within the specified tolerances, then the bisection method can be used to meet the target accurately, by repeatedly asking RD-OPT to give tuning parameters for corrected targets and re-compressing. Note that RD-OPT is run only once, and after it is done with its computations, it can be repeatedly requested to give tuning parameters for arbitrary targets.

3.2.2 Wavelet-Based Image Compression

Most state-of-the-art wavelet methods have excellent performance, even with the restricted, implied quantization used for embedded coding. To realize the QCLIC framework, it is only necessary to compress an image once, up to the maximum possible rate, while recording the rates and qualities at intermediate points. This curve, along with either or both of the compressed image and the original image, can be stored to efficiently implement both *QCLIC-GetCurve* and *QCLIC-Compress*. With reversible methods such as CREW [ZASB95] and S+P-SPIHT [SP96b], the compressed image at maximum rate itself is lossless. Note that the internal curve for any point is the same as the "external" rate-quality curve up to that point.

If wavelet-based compression methods are to be used when features such as random-access and error tolerance are important, the sophisticated structures used by the embedded methods must be abandoned. In such cases, rate-quality performance can be improved by using a conventional, JPEG-like quantization strategy. RD-OPT can always be modified to do efficient quality control whenever a decorrelating orthonormal transform is followed by quantization.

3.2.3 Vector Quantization

The tuning parameter, in case of VQ, is the codebook. The most commonly used techniques use only a fixed, quality-ordered set of codebooks (such as a set of codebooks of increasing size with identical vector dimension), with structural restrictions on the codebooks for fast encoding (such as TSVQ), and fixed-length codes. In this case, the optimization problem is trivial. With fixed-length codes, each codebook has a fixed rate. The quality for each codebook can be evaluated off-line to generate and store the rate-quality curve. A rate-target in *QCLIC-Compress* can always be met on-the-fly.

When a quality-target needs to be met on-the-fly, the bisection method can be used to find the appropriate codebook. Instead of evaluating the quality by actually finding the closest codebook entry for each block (which is the same as encoding, in terms of complexity), estimates of quality can be used for each codebook. One way to do that is to calibrate each codebook in terms of the average quality it produces on images grouped into various classes. Then, the quality of an image using a codebook can be estimated as the quality of the class the image is in.

In practice, VQ methods are not very widely used, as the other, transform-based techniques (such as with DCT or DWT), perform much better. When VQ *is* used, it is usually with a restricted set of codebooks (as mentioned above), designed using a training set of images similar to the ones actually used in the application. In this case, the average quality achieved by a codebook on the training set is a good quality estimate. For the general case, when there is a number of possible codebooks for each rate, more research is needed. For example, techniques to efficiently classify images with respect to a set of codebooks can be developed, to restrict the search range as well as to evaluate quality more efficiently.

3.3 QCLICS: Sets of QCLIC-Images

In this section, we address the problem of quality-controlled compression of sets of QCLIC-Images. Images may be grouped together into sets for many possible reasons. For example, a set of arbitrary images to be put on a single CD-ROM, the results of a query in an image database, the contents of a web document, etc. In addition to optimizing compression-quality tradeoffs for individual images in a set, it is important to allocate rate-quality quotas optimally to the individual images out of a fixed total chunk. An example application that illustrates this is multimedia CD-ROM production, where the goal would be to find, for instance, the best way to pack a set of 1000 images in the available disk space of 20 Megabytes. Several constraints, such as limits on worst acceptable individual image quality or size, may also be placed. For example, a multimedia archive delivering images across a network may be asked to provide Quality-of-Service guarantees, such as transmitting a certain number of images per second across a certain bandwidth. This would translate to limits on maximum rate for individual images.

3.3.1 Illustration of Rate-Quality Tradeoff Across Images

Consider two images, I_1 and I_2, with three rate-quality points for each image, denoted by $\{(R', Q_i'), (R, Q_i), (R'', Q_i'') \mid i = 1, 2\}$, as shown in Figure 24. Let the rates R', R, and R'' be such that $R = (R' + R'')/2$. If we are to compress these two images such that the average rate is equal to R, the straightforward way would be to use the points (R, Q_1) and (R, Q_2). This would give an average quality of $(Q_1 + Q_2)/2$. If both images have identical rate-quality curves, this average quality cannot be improved while keeping the rate equal to R, as the slope of the rate-quality curve typically

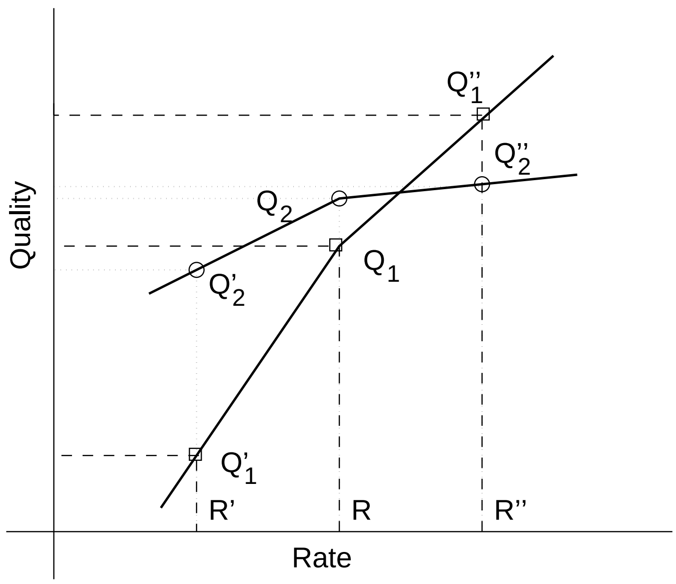


Figure 24: Rate-quality tradeoff across two images.

decreases as rate increases. However, if the slope for one image beyond R is greater than the slope of the other before R, then the average quality can be improved. In the example shown, compressing I_2 at rate R' reduces its quality to Q'_2, but this reduction is more than offset by compressing I_1 at (R'', Q''_1), thereby giving an average quality $(Q''_1 + Q'_2)/2 > (Q_1 + Q_2)/2$. Note that this is a desirable exchange of quality, as the image with greater slope is probably harder to compress (it is farther away from its leveling-off stage in the rate-quality curve). In general, the improvements obtained by optimizing rate-quality tradeoffs across images are greater for image sets containing widely different images.

3.3.2 Measuring Rate and Quality For Image Sets

In this section, we fix some notation and address the issue of measuring compression and quality for image sets. We assume that the quality metric $\mathcal{Q}_S$ for the image set is built as an aggregate using an underlying quality metric $\mathcal{Q}$ for the individual images. Further, the individual images themselves are QCLIC-Image objects. Without loss of generality, we assume that a single compression method is to be used for all the images in the set. If different compression methods are to be used, the QCLIC-Image methods *QCLIC-GetCurve* and *QCLIC-Compress* will use the appropriate method for each image, and our simplification is only notational.

Let $\mathcal{I} = \{I_1, I_2, \ldots, I_N\}$ be a set of N QCLIC-Images, with P_i being the total number of pixels in the i^{th} image. Let S_i (R_i) denote the size (rate) and Q_i the quality of the compressed images.

We can measure compression for the set $\mathcal{I}$ in two ways, averaging either the rates or the sizes. These two will be different if the number of pixels in different images is different. Averaging rate assigns equal importance to each pixel in each image, while averaging size assigns equal importance to each image, irrespective of its number of pixels. Assuming that the set $\mathcal{I}$ is not a collection of images grouped together arbitrarily, it should be possible to determine which average to use. Usually, averaging size is appropriate, as all images are equally important. In certain special situations, such as when the set $\mathcal{I}$ shows the same scene at many different resolutions, average rate should be used. In general, the *rate* R of the compressed set $\mathcal{I}$ is given by

$$R = \sum_{i=1}^{N} r_i,$$

where $r_i = R_i w_i^r$, w_i^r is a non-negative weight assigned to the i^{th} image. We will use average size for our examples, for which,

$$w_i^r = P_i/P,$$

where $P = \sum_{j=1}^{N} P_j$. In this case, the total size of the compressed set $\mathcal{I}$ is RP bits. We allow *constraints* in the form of bounds on r_i, along with a target value for $R = \sum r_i$.

Another possibility for measuring compression is to take the maximum of all R_i (or S_i). Keeping this maximum under a target while maximizing quality, however, reduces simply to choosing equal rates (or sizes) for all images.

Quality of a compressed set of images can also be measured by averaging the individual qualities. Again, different weights may be assigned to different images.

$$Q = \sum_{i=1}^{N} q_i,$$

where $q_i = Q_i w_i^q$, w_i^q being a non-negative weight assigned to the i^{th} image. For our examples, we use the simple average, in which case, $w_i^q = 1/N$. In fact, for the dynamic programming solution to the rate-quality optimization problem for sets, which we present in this section, aggregate quality can be any general "composing function" of the form

$$g_N(Q_N, g_{N-1}(Q_{N-1}, \ldots, g_2(Q_2, g_1(Q_1)) \ldots)).$$

We allow constraints to be placed on the qualities of individual images too.

Here too, the minimum quality can be used as the measure of quality of the set, but keeping this minimum above a given target while maximizing compression reduces to choosing equal qualities for all images.

3.3.3 Rate-Quality Optimization for Sets of Images

Given a set $\mathcal{I}$ of N QCLIC-Images, non-negative weights w_i^r and w_i^q, quality constraints q_i^L, q_i^H, and rate constraints r_i^L, r_i^H, the optimization problem can be phrased in two ways.

1. Given a target rate R^*, compress the image set $\mathcal{I}$ such that $R \leq R^*$ and Q is maximized.

2. Given a target quality Q^*, compress the image set $\mathcal{I}$ such that $Q \geq Q^*$ and R is minimized.

In both cases, the individual compression rates ($r_i = R_i w_i^r$) and qualities ($q_i = Q_i w_i^q$) must satisfy the constraints:

1. $r_i^L \leq r_i \leq r_i^H$, and

2. $q_i^L \leq q_i \leq q_i^H$, for each $i = 1, \ldots, N$.

Using efficient methods to solve this problem, the methods *QCLICS-GetCurve* and *QCLICS-Compress* (introduced in Chapter 1), can be easily implemented, using strategies similar to those described for single images in the previous section. The parameters *curve-constraints* and *compression-constraints* are similar to those for single QCLIC-Images, with added constraints $r_i^L, r_i^H, q_i^L, q_i^H$. The parameter *compression-constraints* can also specify interleaving constraints for the resulting compressed images. Similar to the single image case, point-identifiers can be provided by *QCLICS-GetCurve* (or *QCLICS-Compress*, when the compressed image set is incrementally retrieved), and used as a "handle" by *QCLICS-Compress*. We simply present the QCLICS algorithm, which solves the rate-quality optimization problem for sets as formulated here; the rigorous definitions of *QCLICS-GetCurve* and *QCLICS-Compress* are omitted.

3.3.4 The QCLICS Algorithm

The QCLICS algorithm has three steps:

1. *QCLIC-GetCurve* for each image.

2. Dynamic programming to combine the curves.

3. Compression using *QCLIC-Compress*.

We use *QCLIC-GetCurve* to get a set of constraint-satisfying rate-quality points for each image, and then use a dynamic programming approach to combine these. This results in the selection of compression parameters for each image, such that the rate/quality objective is met, and the rate-quality tradeoff is nearly optimal. Finally, the images are compressed according to their selected parameters. We now present each of these steps in detail.

The *QCLIC-GetCurve* Step

The task of this step is to get a piece of the optimal (or nearly optimal) rate-quality curve for each image, such that each point on the curve satisfies the constraints for that image. Let n be a user-supplied positive integer, denoting the granularity which each curve is to be examined. Then the QCLIC-Image method,

$$I_i.\mathit{QCLIC\text{-}GetCurve}(\mathcal{C}, \mathcal{Q},$$
$$[N_P = n, R_l = r_i^L/w_i^r, R_h = r_i^H/w_i^r, Q_l = q_i^L/w_i^q, Q_h = q_i^H/w_i^q]),$$

returns n rate-quality points (which we scale using the QCLICS weights) and point-identifiers, $\{(r_i^j, q_i^j, \mathbf{id}_i^j) \mid j = 1, \ldots, n\}$, such that the constraints

$$r_i^L \leq r_i^j \leq r_i^H, \text{ and, } q_i^L \leq q_i^j \leq q_i^H$$

are satisfied at each point, and the rates r_i^j are spread uniformly over the range enforced by the constraints. Higher values of n result in finer granularity of the search space, but increase the time complexity of the dynamic programming step.

The dynamic programming step

Discretize each r_i^j as $\rho_i^j = \mathit{RoundOff}(r_i^j \times B)$, where B is a large integer. Let ρ^M be the discretized maximum total rate for $\mathcal{I}$. If a target rate R^* for $\mathcal{I}$ has been specified then $\rho^M = \mathit{RoundOff}(R^* \times B)$, while if a target quality has been specified then $\rho^M = \sum_{i=1}^{N} \rho_i^n$.

Let BestQ be a table with N rows and ρ^M columns. The entries in BestQ have the following interpretation: $\mathit{BestQ}(i,\rho)$ is the highest sum of qualities for images I_1 through I_i such that the sum of their discretized rates is exactly ρ. Initialize BestQ to be $-\infty$ everywhere.

The table BestQ is easily filled, starting from row 1, and marching down the rows, using the information about the i^{th} image and the values in the $(i-1)^{\text{th}}$ row to fill the i^{th} row:

$$\mathit{BestQ}(i,\rho) = \max\left\{ q_i^j + \mathit{BestQ}(i-1,\rho') \;\middle|\; \begin{array}{l} 1 \le j \le n \\ \rho' \ge 0 \\ \rho' + \rho_i^j = \rho \end{array} \right\}.$$

The choice of j made for each $\mathit{BestQ}(i,\rho)$ is recorded in the $N \times \rho^M$ array Choice.

1. Initialize each $\mathit{BestQ}(i,\rho)$ to $-\infty$.

2. Initialize row 1:

   for $j := 1$ **to** n

       if $q_1^j > \mathit{BestQ}(1,\rho_1^j)$ **then**

           $\mathit{BestQ}(1,\rho_1^j) := q_1^j$

           $\mathit{Choice}(1,\rho_1^j) := j$

3. Fill remaining rows:

   for $i := 2$ **to** N

       for $j := 1$ **to** n

for $\rho' := 1$ **to** $\rho^M - \rho_i^j$

if $q_i^j + BestQ(i-1, \rho') > BestQ(i, \rho_i^j + \rho')$ **then**

$BestQ(i, \rho_i^j + \rho') := q_i^j + BestQ(i-1, \rho')$

$Choice(i, \rho_i^j + \rho') := j$

Now, if a target rate R^* has been specified, find the highest value of ρ such that $\rho \leq RoundOff(R^* \times B)$ and $BestQ(N, \rho) > -\infty$. If a target quality Q^* has been specified, then find the least ρ such that $BestQ(N, \rho) \geq Q^*$. From this starting point in the N^{th} row, the choice j_i^* for each image i can be recovered as follows:

for $i := N$ **down to** 1

$j_i^* := Choice(i, \rho)$

$\rho := \rho - \rho_i^{j_i^*}$

Note that after the dynamic programming step, *any* target rate/quality is readily obtained for the given constraints. Thus, the *QCLICS-GetCurve* method can also be realized using the QCLICS algorithm.

The compression step

Compress each image at the selected rate-quality point. This is done by simply using the method:

$$I_i.QCLIC\text{-}Compress(\mathcal{C}, \mathcal{Q}, [\mathbf{id}_T = \mathbf{id}_i^{j_i^*}], []).$$

If the *compression-constraints* parameter passed to the QCLICS method specified interleaving and/or incremental retrieval, use appropriate constraints for individual images in the calls to *QCLIC-Compress* and interleave the results as desired.

3.3.5 Performance

We present performance results for four sets of images, compressed using JPEG to optimize average PSNR. These sets are:

1. Brain: 11 grayscale images of cat's brains.

2. Cell: 31 grayscale images of a cell.

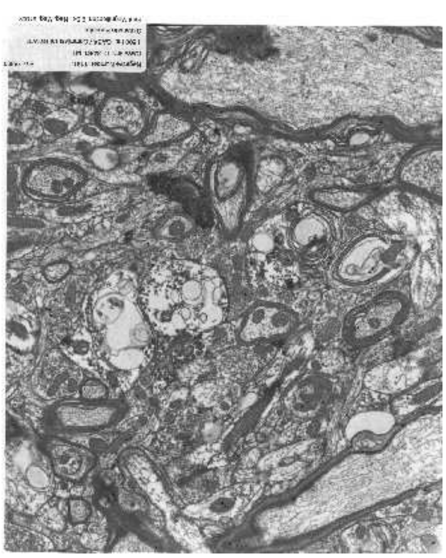
3. Crystal: 14 grayscale images of various crystals.

4. Venus: 17 grayscale images of Venus.

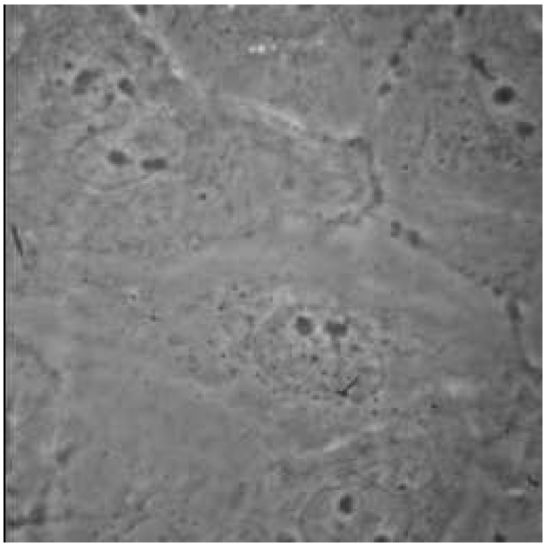
A sample image from each set is shown in Figure 25. For each set, we ran QCLICS with several constraints and objectives. Average sizes were used for rate (that is, each $w_i^r = P_i/P$), and simple averaging was done for quality (that is, each $w_i^q = 1/N$). When a target rate R^* was specified, each R_i was constrained to be in the range $(R^* - \delta_r, R^* + \delta_r)$, and when a target quality Q^* was specified, each Q_i was constrained to be in the range $(Q^* - \delta_q, Q^* + \delta_q)$, for various values of δ_r and δ_q. The number of rate-quality points used per image (n) was 50, and the discretization constant B was 5000. RD-OPT was used on-the-fly for each image to realize the *QCLIC-GetCurve* and *QCLIC-Compress* calls, and the rates predicted by RD-OPT were corrected by compressing the image once at a median rate and observing the error.

Table 1 shows the performance of QCLICS when R^* was specified, and Table 2 shows the performance of QCLICS when Q^* was specified.

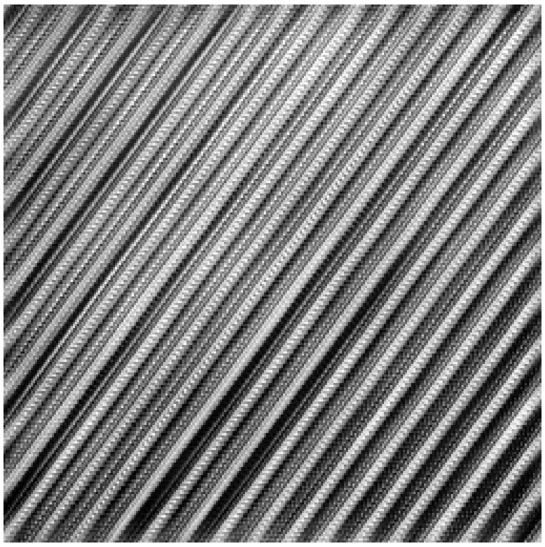
Table 3 shows the average running time per image for each step of QCLICS for each image set, on a 200 MHz Pentium running Solaris 2.5. The time spent in the dynamic programming step of QCLICS is linear in n and B and N. As the number



(a) Brain



(b) Cell



(c) Crystal

(d) Venus

Figure 25: Sample images used for testing QCLICS.

Name	R^* (bpp)	δ_r (bpp)	Predicted R (bpp)	Actual R (bpp)	Predicted Q (dB)	Actual Q (dB)
Brain	1.20	0.30	1.20	1.21	29.6	29.6
Cell	0.75	0.25	0.75	0.75	49.8	49.2
Crystal	1.50	0.50	1.50	1.42	35.7	35.8
Venus	0.75	0.25	0.75	0.74	58.0	59.3

Table 1: QCLICS performance with target R^*.

Name	Q^* (dB)	δ_q (dB)	Predicted R (bpp)	Actual R (bpp)	Predicted Q (dB)	Actual Q (dB)
Brain	37.0	04.0	1.79	1.86	33.0	33.0
Cell	50.0	04.0	0.83	0.79	50.0	49.4
Crystal	35.0	10.0	1.08	1.02	35.0	35.0
Venus	55.0	05.0	0.33	0.38	55.0	55.5

Table 2: QCLICS performance with target Q^*.

Name	*QCLIC-GetCurve*	Dynamic programming	*QCLIC-Compress*	Total
Brain	2.1	0.2	1.7	4.0
Cell	0.6	0.2	0.7	1.5
Crystal	1.7	0.2	1.8	3.7
Venus	0.7	0.1	0.6	1.4

Table 3: QCLICS average running time per image in seconds.

of images N in the set grows, B must be increased too, otherwise the rates will be discretized too coarsely. Thus the QCLICS algorithm as presented here does not scale very well. However, instead of using dynamic programming, we can use Lagrangian minimization. With Lagrangian minimization, the time taken to "process" each image will be independent of the number of images (unlike the dynamic programming case where it depends upon B which must be increased with N). The disadvantage with the Lagrangian technique is that if the rate-quality curves are produced by interpolating the quality over a very limited number of points (as would be done by typical QCLIC realizations), the intermediate points will never be considered.

3.4 Implementation and Evaluation

We have implemented the QCLIC framework for several test-bed applications. In most cases, we have used JPEG compression, with RD-OPT as the enabling technology. The design choices to be made depend upon response time and media constraints, compression method, and quality metrics.

3.4.1 Rate-Quality Curve Based Interactive Compression

This is a simple GUI to enable users to do JPEG compression at any quality or rate, interactively. The user selects an image, and is presented with the rate-quality curve (any of the distortion-based metrics can be used in the current implementation) generated using RD-OPT. Users can click on any part of the curve to compress the image at that rate/quality and see the results or save them in a file. The knee-normalization described in Chapter 2 (PSNR/N) can also be done. This is in contrast to standard interfaces (such as that provided by the Independent JPEG Group's widely used JPEG software library) for JPEG compression which require the user to input the tuning parameters directly, rather than choose a rate/quality target.

This compression tool can also be used in production mode, non-interactively. If a large number of images are to be routinely compressed in a pipeline, then this tool (using RD-OPT for JPEG) can be used to ensure that good choices of tuning parameters are made, and that rate/quality targets are closely met. When compression is done without human interaction, it is important to use a quality target using a metric that is uniform across images. We have found it useful to use PSNR/N in such cases, for example, obliviously compressing each new image to a PSNR/N of 1.2. This is very useful in production mode compression using any other technique (such as SPIHT), even when tuning parameters can be easily selected, to automatically select a good level of quality.

3.4.2 QclicBrowse: An Image Browser With Quality Control

QclicBrowse is an image browser that uses the QCLIC framework to retrieve images efficiently over networks. For each image, the QCLIC structure is generated off-line, and stored at the server. Browsing can be done at any minimum quality (using a number of quality metrics), or minimum rate. The images can be incrementally retrieved while browsing, with the user interactively clicking on a rate-quality curve. Figure 26 shows a snapshot of QclicBrowse at work, browsing a large set of images of cells. The GUI was written in Tcl/Tk, while the underlying server code, and the incremental image-display code, is in C running on various UNIX platforms. The user can also retrieve sets of images with quality-control, and animate sequences of images with quality-control. A stack of “interesting” images can be set aside, for detailed viewing. For each image, the original, lossless image is also available for retrieval. At the server end, all the QCLIC information is stored in a simple format that we have developed, which is also used by other systems that we have built. This format has also been successfully stored and used for providing quality-controlled image data in the ZOO experiment management system [ILGP96] being developed in the UW-Madison Computer Sciences Department.

3.4.3 TASVIR: an Image Server

TASVIR (Transform, Arrange, Scale and View Incremental Rasters) is an image server whose main task is to allow applications to display images efficiently, without worrying about format conversions and display intricacies. The implementation is in C, and the image display is done using the X library on UNIX platforms. Clients request TASVIR to display an arbitrary image in an arbitrary rectangle of any X window. The image source could be a file, or a process, or a remote web site (specified as a URL). TASVIR retrieves the image and displays it, doing sophisticated color mapping to optimally use the available color palette. TASVIR understands the QCLIC format, and can do quality control for QCLIC images. Even when the QCLIC information is not available, TASVIR can build approximations on-the-fly to do quality-control. The current implementation supports JPEG, progressive JPEG, and SPIHT (for grayscale images only) compression formats, as well as a number of uncompressed image formats (PNM, TIFF, GIF, etc.). TASVIR can also save images in various formats (including PostScript), and perform simple image processing operations such as brightness and contrast control. TASVIR is used as a component in the DEVise (Data Exploration and Visualization) [LRB+97] tool being developed in the UW-Madison Computer Sciences

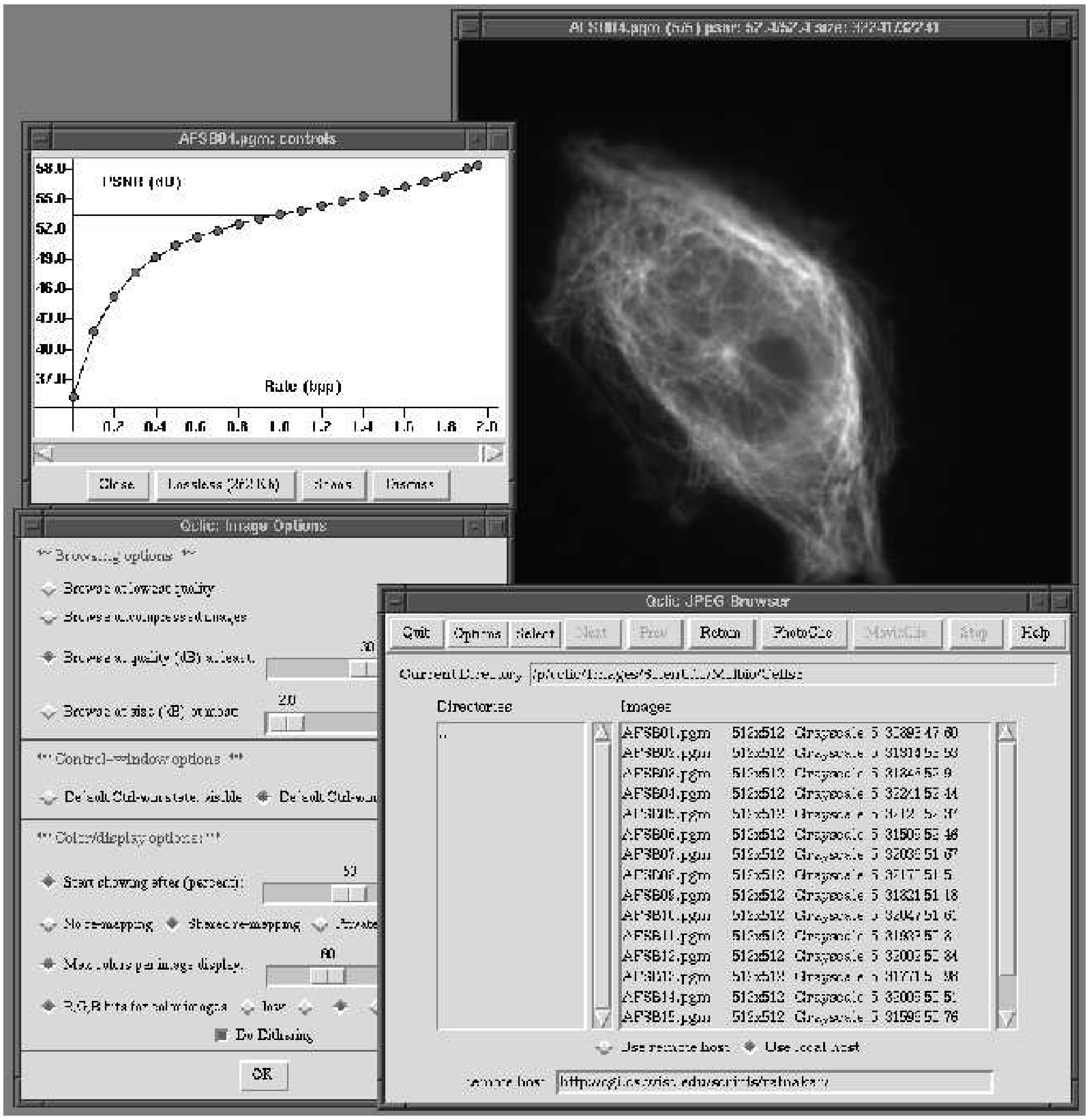


Figure 26: QclicBrowse snapshot.

Department. DEVise allows users to efficiently visualize data, and when some data fields are images, DEVise uses TASVIR for displaying them. Several sophisticated "visual queries" are supported by DEVise, the results of which could require large sets of images to be dynamically displayed. TASVIR can use the QCLIC framework to apportion rate/quality among the images to quickly display the entire set, and then add details to the images later. For every single image displayed by DEVise, a TASVIR "control panel" can be invoked, which can be used to perform quality-control, image manipulation, and format interchange. Figure 27 shows a snapshot of DEVise visualization of Mars imagery data using TASVIR, with the control panel for one of the images.

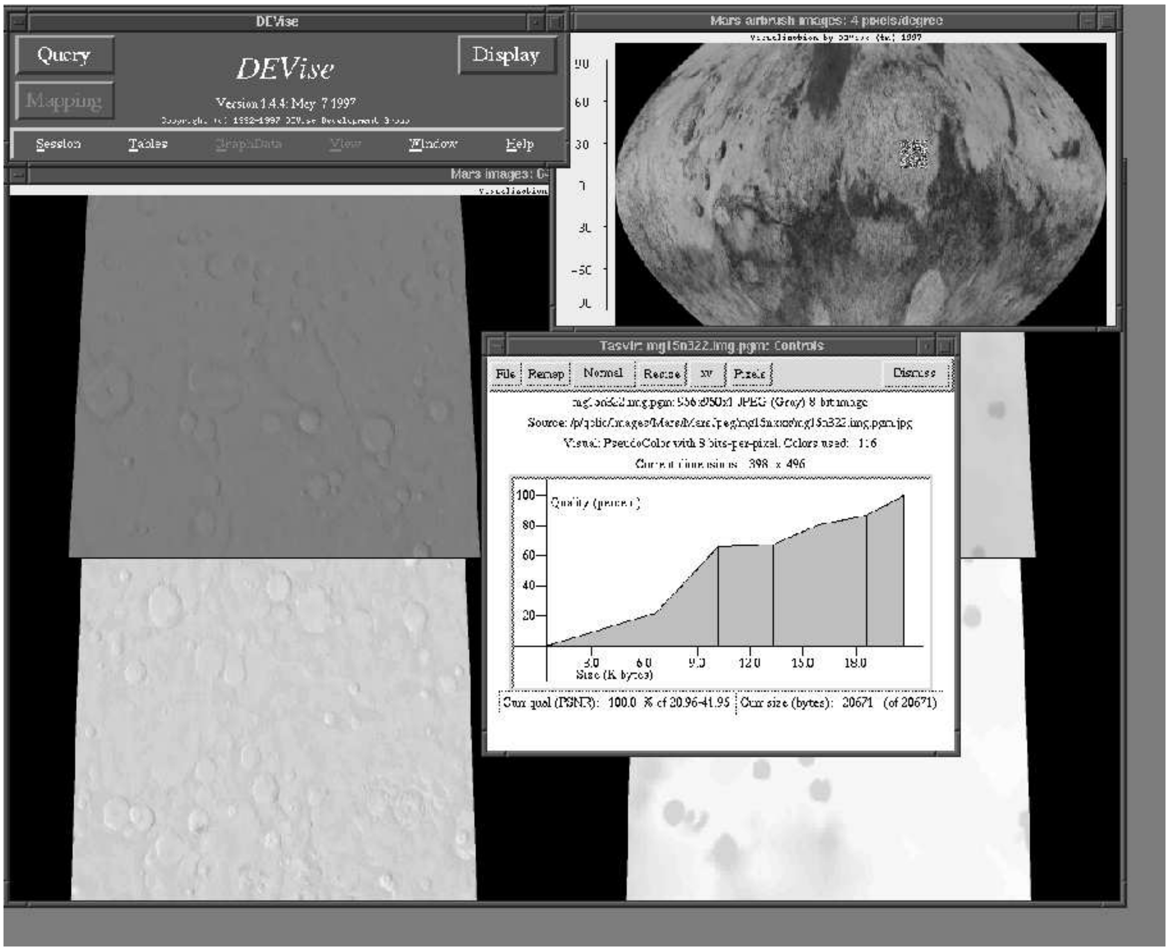


Figure 27: DEVise snapshot showing Tasvir controls.

Chapter 4

RD-OPT: QCLIC for JPEG

As seen in Chapter 2, the tuning parameter in JPEG and other DCT-based image compression techniques is the quantization table. The extent of compression achieved depends upon the coarseness of quantization of the transform coefficients. The mapping between a quantization table and the resulting rate-quality point is image-specific and fairly complex, and there were no techniques to efficiently solve the problem of choosing an optimal quantization table to closely meet a specified rate or quality target. RD-OPT addresses this problem, and is the key enabling technology for using the QCLIC framework with JPEG-compressed images.

4.1 Previous Work

Several approaches have been tried in order to select and optimize quantization tables for particular rate/quality specifications. Each quantization table entry can be any integer between 1 and 255. Thus, the size of the search-space is 255^{64}, and exhaustive searching is clearly out of the question.

The simplest and most commonly used approach is to use a default table and scale it up or down by a scalar multiplier. The multiplier is varied, and at each point, the image is compressed to evaluate the rate and then decompressed to evaluate the quality.

This process is repeated until the desired target is met. This is very inefficient. Moreover, we have shown in [RL94] that this might give rather poor rate-quality tradeoffs. Other approaches include psycho-visual model based quantization [AP92, Wat93], and stochastic optimization techniques [MS93]. In each of these, a particular quantization table is evaluated by going through the entire compression-decompression cycle. The search space is navigated using some heuristic. The best results (with PSNR as quality metric) achieved using such searching algorithms have been reported by Wu and Gersho [WG93], by using a search strategy that is equivalent to moving in the direction that minimizes the Lagrangian, $R + \lambda D$.

Ramachandran and Crouse extended the work in [WG93] to search for an optimal *zeroing strategy,* along with a quantization table [CR95]. Zeroing strategy refers to adaptively setting some nonzero-quantized coefficients to zero, in order to improve the rate-distortion tradeoff. The results obtained in [CR95] are the best known JPEG compression results, with PSNR as quality metric.

The major drawback of the search strategy in [WG93] and [CR95] is its computational complexity. For each quantization table (and zeroing strategy) tried, the entire compression-decompression cycle is used as a "black box" (except for the forward DCT, which need not be repeated every time). Moreover, the whole search process results in a single quantization table, corresponding to the value of λ in the Lagrangian. If a single target is to be achieved, the bisection method needs to be used to search for a λ that will meet the target. For getting the rate-quality curve over a wide span of rates, λ must be swept through a wide range of values.

The RD-OPT algorithm [RL95, RL96] proposed by us overcomes these shortcomings, while achieving equally good rate-quality tradeoffs. If only the quantization table is searched for, the performance coincides with that of the algorithm in [WG93] for all the test images used in that work. RD-OPT can also optimize a restricted zeroing strategy, and the performance is only very slightly lower than that reported in [CR95].

4.2 Overview

RD-OPT is an algorithm to efficiently optimize rate-quality tradeoffs and achieve rate/quality targets for JPEG, in an image-specific way. That is, RD-OPT can be used for efficient quality-control for JPEG, in the QCLIC framework. RD-OPT can work with any of the distortion-based quality metrics described in Section 2.2, and with several other quality metrics. For example, RD-OPT can also be used with the quality evaluations used by Watson in [Wat93], which were based on extensive psycho-visual experiments done to evaluate the visibility of the 64 DCT basis functions. We will present RD-OPT in terms of distortion, and then describe extensions for other quality metrics.

The main idea in RD-OPT is to evaluate quantization tables more efficiently, rather than using the entire JPEG compression-decompression cycle. The DCT is a linear orthonormal transform [RY90]. Hence the distortion can be measured in the DCT coefficients themselves, as follows. For an image block f, let the DCT coefficients block be $F = \mathrm{DCT}(f)$. If F' is an approximation of F, such as that produced by quantizing and dequantizing, and $f' = \mathrm{IDCT}(F')$ is the corresponding approximation of f, then $\mathrm{DCT}(F') = f'$. From the linearity of DCT, it follows that,

$$\begin{aligned} \mathrm{DCT}(f - f') &= \mathrm{DCT}(f) - \mathrm{DCT}(f') \\ &= F - F'. \end{aligned}$$

Since DCT is an orthonormal transform,

$$
\begin{aligned}
\sum_{i=0}^{7}\sum_{j=0}^{7}(f-f')(i,j)^2 &= \sum_{u=0}^{7}\sum_{v=0}^{7}(F-F')(u,v)^2, \text{ that is,} \\
\sum_{i=0}^{7}\sum_{j=0}^{7}(f(i,j)-f'(i,j))^2 &= \sum_{u=0}^{7}\sum_{v=0}^{7}(F(u,v)-F'(u,v))^2.
\end{aligned}
$$

Thus, the distortion can be calculated by adding distortions in the 64 DCT coefficients. We use a simple *rate model* to decompose the total rate into a sum over the 64 coefficients. Recall that the DCT has a strong decorrelating property for typical images. We approximate the rate by adding together the measured entropies of the 64 quantized DCT coefficients. If the coefficients were statistically independent, this sum would be the entropy of the quantized coefficient blocks, and would be a lower bound to the rate achievable with either of Huffman coding or arithmetic coding. In practice, the coefficients are not entirely independent, and the entropy of the blocks is lower than that calculated as a coefficient-wise sum. However, the actual rate resulting from any real implementation of arithmetic coding or Huffman coding is slightly greater than the entropy lower bound (more so in the case of Huffman coding). This difference compensates for the fact that our entropy calculation results in an over-estimate. We conducted extensive experiments with several images and the results indicated that the coefficient-wise sum of measured entropies of the quantized coefficients is a very good estimate of the rate for JPEG [RFVK94]. Moreover, the performance of RD-OPT, which has been used by us and by a very large number of other users and researchers, has confirmed the accuracy of the rate model. Of course, there are instances when the rate model deviates from the actual rate by more than the allowed tolerance. We will describe corrective measures used for the QCLIC framework, for such cases.

By using coefficient-wise sums of entropies and distortions, we can avoid doing the symbol encoding at each step to evaluate the performance of a quantization table. RD-OPT uses a novel technique to calculate rate and entropy, using histograms of DCT coefficient statistics, thereby avoiding quantization/dequantization at each step too. The statistics are measured once, and then used to build rate and distortion tables that can be used to efficiently calculate the performance of any quantization table. Further, the decomposition of rate and distortion into coefficient-wise sums allows the individual components (for each of the 64 coefficients) to be optimized separately, thus avoiding the search through a space of exponentially large size.

4.3 Thresholding

The distribution of each AC DCT coefficient of an image is approximately Laplacian [RG83]. We showed in Section 2.6 that the MT-U-M quantization strategy used in JPEG can be enhanced considerably by adaptively optimizing the zeroing threshold (the MT-T-M strategy) for each coefficient. This does not change the decoder at all—a quantized coefficient is still dequantized by multiplying by the corresponding quantization table entry. The RD-OPT algorithm can optimize thresholding along with the quantization tables. We refer to this as *global thresholding*, as the zeroing threshold for a particular DCT coefficient is the same irrespective of the block it belongs to. The zeroing strategy used in [CR95] decides on a per-coefficient basis whether it is advantageous to set it to zero, and we refer to it as *local thresholding*. Our results indicate that the gains obtained by local thresholding are only marginally better than those obtained by global thresholding, while the computational complexity of optimizing global thresholding with RD-OPT is substantially lower than that of optimizing local thresholding, especially for large images.

The gain in rate-distortion performance resulting from thresholding can be of two kinds. For each coefficient, there is the gain obtained because the entropy is lowered considerably while the distortion is only marginally increased. This is the type of gain we discussed and analyzed in Section 2.6. We refer to this as Type-I gain.

When Huffman coding is used on runs of zeros, as in the JPEG Huffman coding mode, thresholding can also improve rate-distortion performance in another way. If a sequence of quantized coefficients (in the zig-zag scan order) contains only one or two isolated non-zeros, it might be useful to zero off those coefficients to achieve a larger run of zeros, as it might reduce the Huffman coding rate more than enough to offset the increase in distortion. In most cases, the reduction in actual rate would be modeled well by the reduction in measured entropy for the coefficient(s) that are set to zero, and the gain would be included in Type-I. However, in some blocks, because of the residual correlation among coefficients, the gain might be significantly greater, and we classify this type of gain as Type-II. The object of this nomenclature is to understand the slight improvement in rate-distortion performance resulting from optimal local thresholding over optimal global thresholding. Local thresholding can examine neighboring coefficients for each coefficient in the image, and hence can take into account Type-II gains too, which global thresholding cannot.

When global thresholding is used, the compression process needs two tuning parameters: the quantization table and the *threshold table*. A threshold table T is an 8×8 table of positive real numbers. Let $/\!\!/$ denote division followed by rounding to the nearest integer. Define,

$$x /\!\!/\!\!/ (q,t) = \begin{cases} 0 & \text{if } \ |x| < t \\ x /\!\!/ q & \text{otherwise.} \end{cases}$$

Then, coefficient quantization with a quantization table Q and a threshold table T is given by:

$$F_{Q,T}(u,v) = F(u,v) /\!\!/\!\!/ \left(Q(u,v), T(u,v)\right).$$

The table T need not be included with the compressed image, as the decompressor does not need to know the thresholds.

4.4 RD-OPT Details

We consider a single image plane I first. For a quantization table Q and threshold table T, let $D(Q,T)$ and $R(Q,T)$ represent the distortion and modeled rate (the sum of measured entropies of quantized coefficients), respectively, resulting from JPEG compression of I using Q and T. For simplifying notation, we refer to individual DCT coefficients in a block by numbering them in raster order. $F(u,v)$ is referred to as $F[n]$, where $n = 8u + v$. Thus, $F[0]$ is the DC coefficient, and $F[63]$ is the last AC coefficient. Let $D_n(q,t)$, the contribution to total distortion from coefficient number n, be defined as:

$$D_n(q,t) = \frac{1}{64}\text{Mean}\left\{\left(F[n] - (F[n] /\!\!/\!\!/ (q,t)) \cdot q\right)^2\right\}, \tag{4.1}$$

where the mean is taken over all the DCT blocks. Similarly, define $R_n(q,t)$, the rate contribution as,

$$R_n(q,t) = \frac{1}{64}\text{Entropy}\{(F[n] /\!\!/\!\!/ (q,t))\}, \tag{4.2}$$

where the entropy is measured over all the DCT blocks[1]. Thus,

$$D(Q,T) \;=\; \sum_{n=0}^{63} D_n\left(Q[n], T[n]\right), \tag{4.3}$$

[1] If $(F[n] /\!\!/\!\!/ (q,t))$ takes the value v in a fraction $p_v > 0$ of all blocks, then this entropy is $-\sum_v p_v log_2 p_v$.

$$R(Q,T) = \sum_{n=0}^{63} R_n\left(Q[n],T[n]\right). \tag{4.4}$$

Given the image I, the rate-distortion optimization problem (under our rate model) is to find Q and T such that the rate $R(Q,T) \leq R^*$, and $D(Q,T)$ is minimized, for a given rate budget R^*.

RD-OPT uses histograms of DCT coefficient distributions to simplify the calculations of the coefficient-wise components $D_n(q,t)$ and $R_n(q,t)$. It then finds Q and T by minimizing the sum $\sum_n D_n(\cdot,\cdot)$ against the sum $\sum_n R_n(\cdot,\cdot)$ using either dynamic programming or Lagrangian minimization.

4.4.1 Gathering Histograms

Equations 4.1 and 4.2 can be used to calculate $D_n(q,t)$ and $R_n(q,t)$ for any q and t, by actually quantizing/dequantizing each block's n^{th} coefficient, $F[n]$, and calculating the distortion and the counts of various quantized values. However, we can do this much more efficiently (albeit with a slight loss in accuracy of $D_n(q,t)$ values), using histograms of DCT coefficient distributions. We split the range of values of coefficients into buckets of width 0.5, and measure occurrence counts for each bucket. For $i = 1, 2, \ldots$, let $c_n(i)$ be the number of blocks f for which $F[n] \in [0.5(i-1), 0.5i)$. For $i = -1, -2, \ldots$, let $c_n(i)$ be the number of blocks f for which $F[n] \in (0.5i, 0.5(i+1)]$, except that for $i = -1$, the bucket is $(-0.5, 0)$ rather than $(-0.5, 0]$, to avoid counting zero twice. If μ_i is the midpoint of bucket i, and l_i is the endpoint closer to zero for bucket i, then for any $F[n]$ in the i^{th} bucket,

$$\begin{aligned} |F[n] - \mu_i| &\leq 0.25, \\ F[n] /\!\!/ q &= l_i /\!\!/ q. \end{aligned}$$

Thus, each coefficient value in a bucket can be approximated within ± 0.25, and is quantized identically by any q. Since we are interested in optimizing thresholds $T[n]$ too, we note that

$$F[n] /\!/\!/ (q,t) = l_i /\!/ (q,t), \tag{4.5}$$

as long as t is a multiple of 0.5.

4.4.2 Building Rate and Distortion Tables

Let Q_m and Q_M be quantization tables that respectively represent the minimum and maximum values possible for each quantization table entry. That is, $Q_m[n] \leq Q[n] \leq Q_M[n]$. Further, we constrain the threshold table entries to vary in steps of 0.5, from $Q[n]/2$ to $(Q[n] + T_M[n])/2$. Here Q_m, Q_M, and T_M are user-supplied tables of integers that determine the operating range of RD-OPT. Note that $Q_M[n]$ need only be at most $\lfloor 2v + 1 \rfloor$, where v is the maximum absolute value of the n^{th} coefficient in the image.

Using the histograms calculated earlier, RD-OPT builds tables of $D_n(q,t)$ and $R_n(q,t)$, for each (q,t) in the above range. Given q and t, (with t being a multiple of 0.5), the distortion $D_n(q,t)$ is calculated as follows:

$D_n(q,t) := 0$

for each bucket i with $c_n(i) > 0$

$\quad D_n(q,t) := D_n(q,t) + c_n(i)(\mu_i - q \cdot (l_i /\!/\!/ (q,t)))^2$

divide $D_n(q,t)$ by the total number of pixels

Similarly, the rate $R_n(q,t)$ is calculated by using the non-zero bucket counts to calculate the frequency of occurrence of each quantized value, and then calculating the entropy and dividing it by 64. Note that only a few incremental calculations are needed to calculate the table entries for $(q, t + 0.5)$ from those for (q,t), as increasing the threshold from t to $t + 0.5$ affects the quantization of only 2 buckets. The details are omitted here for simplicity.

4.4.3 Optimizing $R(Q, T)$ against $D(Q, T)$

We have used both dynamic programming and Lagrangian minimization to minimize $R(Q, T)$ and $D(Q, T)$. With dynamic programming, there is a loss in accuracy resulting from discretization of rates to form table indices. Also, dynamic programming is much slower in practice than Lagrangian minimization, unless a very coarse discretization is used. It would be more advantageous to use dynamic programming if the $(R_n(\cdot, \cdot), D_n(\cdot, \cdot))$ points were sparse, as Lagrangian minimization can only pick points on the convex hull of the set of these points. In practice, these points are dense enough, and the Lagrangian minimization approach is more efficient. We only describe the Lagrangian minimization approach here. Details on the dynamic programming approach for RD-OPT can be found in [RL95, RL96].

Lagrangian Minimization

The rate-distortion optimization problem is equivalent to that of minimizing the Lagrangian $R(Q, T) + \lambda D(Q, T)$, in the sense that solutions to the latter for any non-negative λ are solutions to the former for some rate budget R^*.

Using equations 4.3 and 4.4, the Lagrangian minimization problem reduces to minimizing

$$\sum_{n=0}^{63} R_n \left(Q[n], T[n]\right) + \lambda D_n \left(Q[n], T[n]\right),$$

for $Q_m[n] \leq Q[n] \leq Q_M[n]$ (in steps of 1) and $Q[n]/2 \leq T[n] \leq (Q[n] + T_M[n])/2$ (in steps of 0.5). For each coefficient, We obtain a subset of the operating points by sorting, such that $R_n(\cdot,\cdot)$ is strictly decreasing, and $D_n(\cdot,\cdot)$ is strictly increasing. This removes some of the points (q,t) from consideration. We then use the Graham scan algorithm [Gra72] to get the (lower half of) convex hull of the $(R_n(\cdot,\cdot), D_n(\cdot,\cdot))$ points. This is done by ensuring that as we move down the R-D curve, we only make left turns. We also remove the points where no turn is made, that is, whose slope is the same with respect to both neighbors. Figure 28 illustrates the resulting set of operating points. Let the values (q,t) that give these h_n remaining points on the hull be represented by

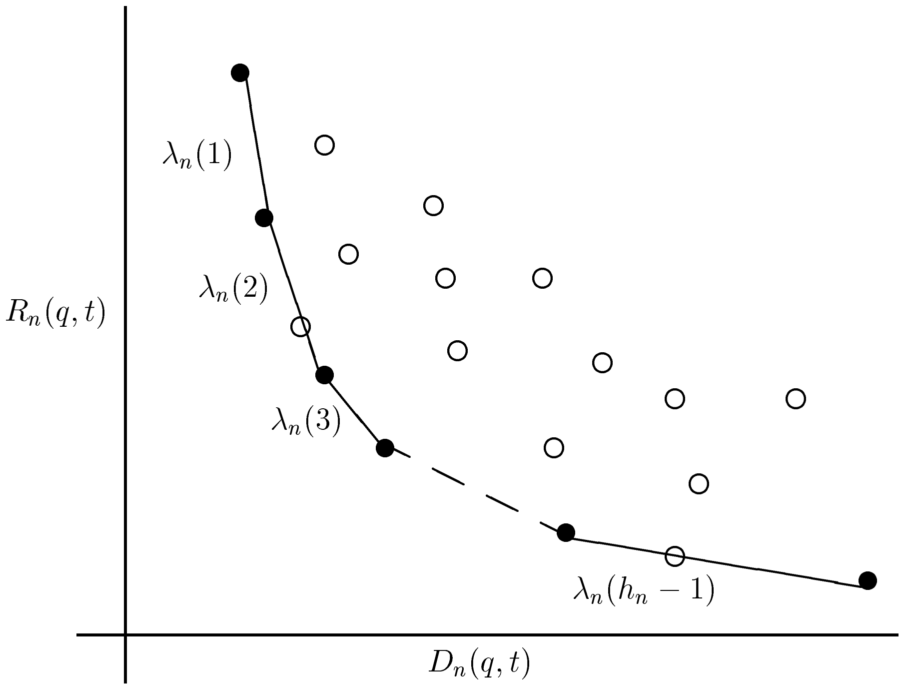


Figure 28: The convex hull of R-D points that is retained by RD-OPT for Lagrangian minimization.

$(q_n(1), t_n(1))$ through $(q_n(h_n), t_n(h_n))$. Define the slopes of the R-D curve for the n^{th} coefficient at these h_n points as:

$$\lambda_n(k) = \frac{R_n\left(q_n(k), t_n(k)\right) - R_n\left(q_n(k+1), t_n(k+1)\right)}{D_n\left(q_n(k+1), t_n(k+1)\right) - D_n\left(q_n(k), t_n(k)\right)},$$

for $k = 1, 2, \ldots, h_n - 1$, and set $\lambda_n(h_n) = 0$. It follows that

$$\lambda_n(k) > \lambda_n(k+1), \quad \text{for } k = 1, 2, \ldots, h_n - 1.$$

Further, for any given $\lambda \geq 0$, the Lagrangian,

$$R_n(q,t) + \lambda D_n(q,t)$$

is minimized with $(q,t) = (q_n(k), t_n(k))$, where k is the least index for which $\lambda \geq \lambda_n(k)$.

Thus, for any given $\lambda \geq 0$, we can use 64 binary searches to efficiently find Q and T in the operating range such that $R(Q,T) + \lambda D(Q,T) = \sum_0^{63} R_n(Q[n], T[n]) + \lambda D_n(Q[n], T[n])$ is minimized. This solves the rate-distortion minimization problem for $R^* = R(Q,T)$. If we are given a target rate R^* (or distortion D^*) we can use the bisection method to efficiently search for a λ that would match the target up to any desired tolerance.

For color images, combined histograms are gathered for each set of planes that are to use the same quantization and threshold table. For example, the Y, Cb, and Cr planes can use 3 different (Q,T), or the Cb and Cr planes can be quantized identically, using a single (Q,T). In any case, for each plane or set of planes to be quantized together, a (Q,T) is found by RD-OPT by using the same λ, to optimize the sum of rates over the planes against the sum of distortions over the planes.

4.5 Performance

For compression of 8-bit images using baseline JPEG, the pixel values are in the range [0..255] and the quantization table entries are in the range [1..255]. We present the performance of RD-OPT using compression results for three images. These are the well-known grayscale images Lena, Baboon, and Peppers. We show the PSNR-rate curves for JPEG compression of these images using RD-OPT generated quantization and threshold tables in Figures 29, 30, and 31. Also shown are the PSNR-rate plots for these images compressed using the "default" table used by most JPEG compressors (the example table in the standard [PM93]) scaled by different values. In addition, the plots also show the performance of RD-OPT without thresholding, i.e., just quantization table optimization. The rates shown in these plots are the actual rates resulting from JPEG compression with Huffman coding, and not entropy estimates. For each image,

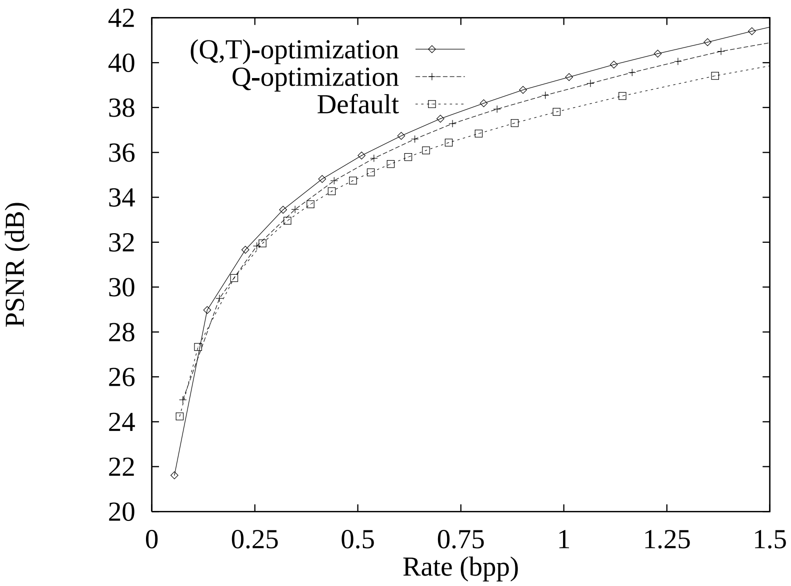


Figure 29: Performance of RD-OPT with and without thresholding, for Lena.

RD-OPT with thresholding results in PSNR gains of up to 2 dB, compared to the scaled default table. The PSNR values achieved by RD-OPT are nearly as good as those obtained by the best wavelet encoders. Algazi and Estes have evaluated the improvements achieved by RD-OPT using the PQS metric [AE97]. At moderately-low to moderately-high bit rates, RD-OPT improves the PQS right up to the level achieved

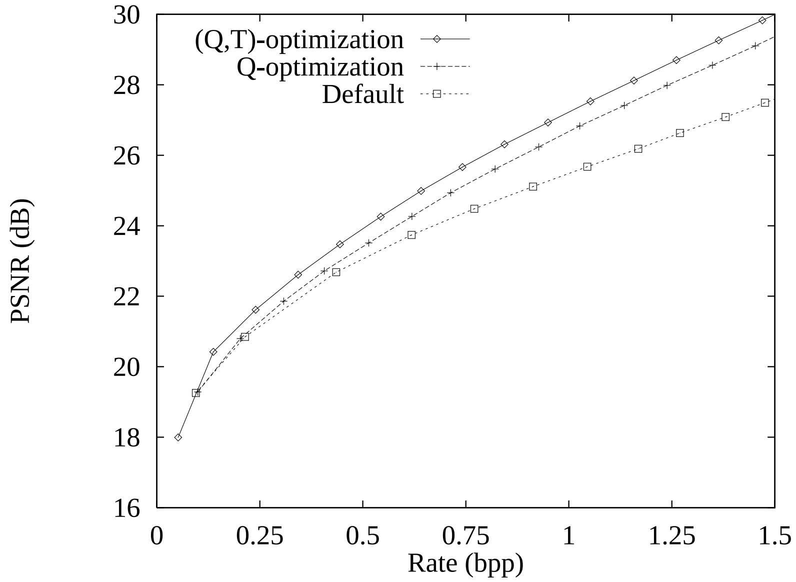


Figure 30: Performance of RD-OPT with and without thresholding, for Baboon.

by the best wavelet encoders. At high bit rates, RD-OPT manages to improve the PQS by about half the gap between the wavelet encoder and JPEG with scaled default tables.

Figure 32 shows the quantization and thresholding tables obtained by RD-OPT for Lena at about 1.0 bit per pixel (bpp). Note that with thresholding, the quantization table entries are more uniform across coefficients. Since all coefficients are equally important for distortion (in the sense that the i^{th} bit of the first coefficient has as much weight as the i^{th} bit of the 50^{th}), every coefficient should be reconstructed with nearly the same accuracy (from purely a distortion perspective). However, it is more advantageous to quantize higher frequencies coarsely as it would reduce the rate much more than coarse quantization of lower frequencies would. This is reflected in the quantization table obtained by RD-OPT without thresholding. With thresholding, the rate reduction is obtained via thresholding, and hence the quantization table entries

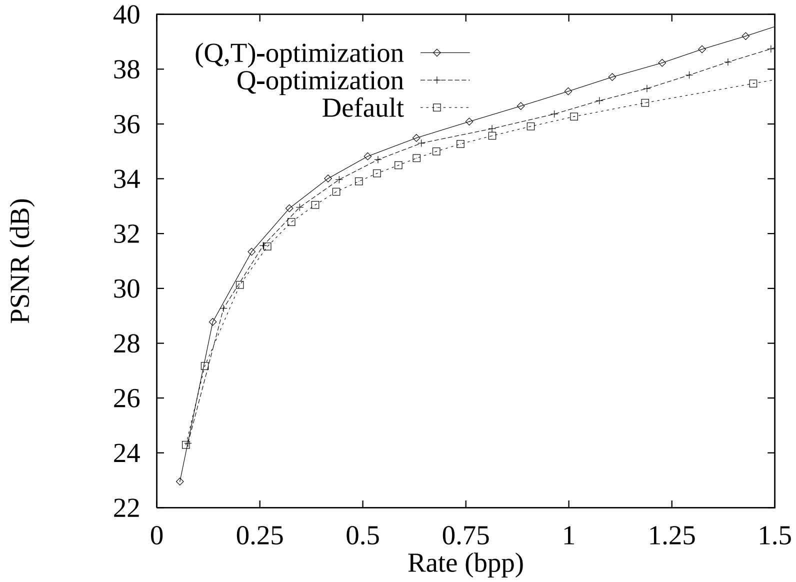


Figure 31: Performance of RD-OPT with and without thresholding, for Peppers.

for higher frequencies are comparable to those for lower frequencies.

4.5.1 Thresholding Gains

The performance curves show that substantial improvements in rate-distortion performance can be achieved with the use of global thresholding as used by RD-OPT, compared to only quantization table optimization. From the rate-distortion curves for MT-U-M and MT-T-M quantization strategies on Laplacian density (shown in Figure 15), one can see that the difference between the logarithms of distortion is about 0.1 in the range 0.25–1.25 bits per sample. If all the quantized coefficients were encoded in this rate range, the PSNR improvement achieved by thresholding should be about 1 dB (over the no-thresholding case). Because of the energy compaction done by the DCT, the rates for different coefficients vary, and only a few of the coefficients are in the 0.25–1.25 range where thresholding gains are maximum. For example, when the overall

Optimized Q (38.8 dB)							
9	9	9	9	13	13	13	13
9	9	13	9	13	13	13	13
9	13	13	13	13	13	13	13
13	13	13	13	13	13	17	17
13	13	13	13	13	13	17	21
13	13	13	13	17	21	21	21
17	21	17	17	17	30	26	25
25	29	25	31	29	31	25	21

Optimized Q/T (39.3 dB)							
9/5.0	9/5.0	9/5.5	10/6.0	10/6.5	10/6.5	9/6.5	10/7.0
9/5.0	9/5.5	10/7.0	9/6.0	10/7.0	10/6.5	11/7.5	10/7.5
10/6.0	10/6.5	10/6.5	10/6.5	10/7.5	9/7.0	11/7.5	10/8.0
10/6.5	10/6.5	10/6.5	11/7.0	10/7.0	10/7.0	11/8.0	10/7.5
10/7.0	10/7.0	11/7.0	10/7.0	10/7.0	12/8.0	11/8.0	10/8.0
10/7.0	10/7.5	12/8.0	11/7.5	11/8.5	11/8.5	10/8.0	6/9.0
11/8.0	10/8.5	11/8.0	10/8.0	10/8.5	11/8.5	11/9.0	11/9.5
10/8.5	11/9.0	11/9.0	11/9.0	11/9.0	11/9.5	5/9.5	1/10.5

Figure 32: Optimized quantization table and quantization/threshold table for Lena at 1.0 bpp.

rate is 1.0 bpp for Lena, the DC coefficient is coded at about 6 bits per coefficient, whereas the highest frequency coefficient is coded at about 0.003 bits per coefficient. In fact, for images with a high level of detail (such as Baboon), the variances (and hence rates) are more uniform over the 64 coefficients, and hence thresholding gains are greater.

To compare the performance improvement resulting from global thresholding with that from local thresholding, we present a comparison between RD-OPT results and the published results in [CR95], for the Lena image, in Table 4. We also show the results with scaled "default" quantization tables, and with just quantization table optimization, so that the improvements from local and global thresholding can be compared.

The numbers show that the additional gains afforded by local thresholding are small,

Rate (bpp)	"Default"	No Thresh (RD-OPT)	Global Thresh (RD-OPT)	Local Thresh (Ramachandran+Crouse)
0.25	31.5	31.83	32.15	32.3
0.50	34.8	35.41	35.77	35.9
0.75	36.6	37.46	37.87	38.1
1.00	37.8	38.86	39.35	39.6

Table 4: Comparison of PSNR's for local and global thresholding at various rates for Lena.

compared to the gains resulting from global thresholding. In fact they could be even smaller than the numbers shown here—we used the actual compressed Lena image (at 1.0 bpp) obtained from the authors of [CR95], and calculated the PSNR. It turned out to be 39.47 dB, instead of 39.6 dB, and the difference is probably due to variations in the original image itself (many incarnations of Lena exist) and minor differences resulting from floating point calculation discrepancies. This confirms that most of the gains of thresholding are of Type-I. Further, RD-OPT offers the additional advantage over the local thresholding algorithm of [CR95] in that it jointly optimizes over the entire range of operating points. That is, after one execution, RD-OPT is ready to produce quantization and thresholding tables for all rate and distortion values, whereas search methods like those in [CR95] and [WG93] have to be re-run for each new value of rate/distortion.

4.5.2 Complexity

The running time of RD-OPT can be split into three main activities: computing the DCT and building histograms, filling the prediction tables, and, computing the slopes $\lambda_n(k)$. The time taken to do the DCT and gather coefficient histograms is the only component that depends upon the dimensions of the image. Building the prediction tables and doing the Lagrangian optimization require time roughly linear in the size of the operating range. The complexity of generating the 64 convex hulls is $O(K \log K)$, where K is the number of operating points. In practice, the overall time is dominated by the time taken to compute the rate and distortion prediction tables from the histograms, which is linear in K.

Thus, the running time grows nearly linearly with the size of the operating range determined by Q_m, Q_M, and T_M. We achieve further reduction in running time by searching coarsely. That is, instead of examining each possible quantizer value between $Q_m[n]$ and $Q_M[n]$, we can skip over some values. Our implementation of RD-OPT uses a parameter called "coarseness" which determines how coarsely the span of quantizer values is examined. It is useful to examine smaller values more finely than larger values, and we have built that into the coarseness parameter. Table 5 shows the quantizer ranges for each value of coarseness in our implementation (the numbers in brackets indicate the step size for a particular subrange). Figure 33 shows the running time

Coarseness	#	Values
0	255	1-255 (1)
1	128	1-16 (1), 18-238 (2), 255
2	100	1-16 (1), 18-112 (2), 116-252 (4), 255
3	53	2-32 (2), 36-96 (4), 104-248 (8), 255
4	37	2-16 (2), 20-48 (4), 56-176 (8), 192-240 (16), 255
5	28	2-16 (2), 20-40 (4), 48-88 (8), 104-152 (16), 184-248 (32), 255
6	19	2-16 (2), 20-40 (4), 64, 96, 128, 176, 255
7	14	2-10 (2), 14, 18, 22, 30, 38, 46, 96, 128, 255
8	8	2, 6, 10, 14, 18, 32, 128, 255

Table 5: Number and values of quantizers at each coarseness.

for our implementation of RD-OPT running on a 200 MHz Pentium running Solaris

2.5, as a function of the coarseness, for the Lena image. The threshold range t in the plots is the number of zeroing thresholds examined for each quantization table entry ($t = 0$ corresponds to no thresholding). For the coarsest search (coarseness = 8) and

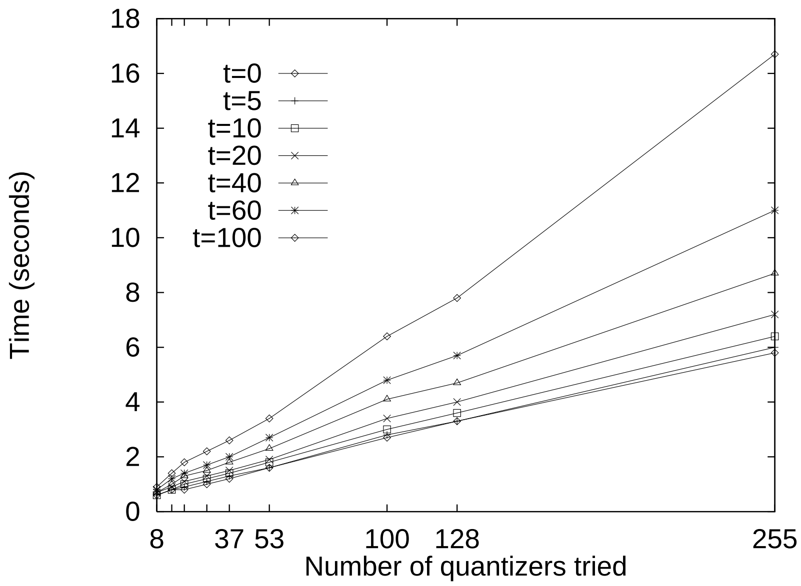


Figure 33: RD-OPT running time as a function of coarseness of search, at various levels of threshold range t.

no thresholding, the running time is 0.6 seconds. The running time increases very slowly as the number t of thresholds tried increases. This is because of the incremental computation of rate and distortion tables from one threshold to the next. Thus, even with $t = 20$, the running time is within about one second of the time for $t = 0$, at each value of coarseness. We evaluate the improvements in rate-PSNR curves as coarseness is reduced using a fixed threshold range, $t = 20$. Figure 34 shows the rate-PSNR curves for Lena, obtained using $t = 20$ at various coarseness settings. The curve points out the utility of using thresholding. With thresholding, RD-OPT can afford to search for quantization table entries at a very coarse level, without losing anything in

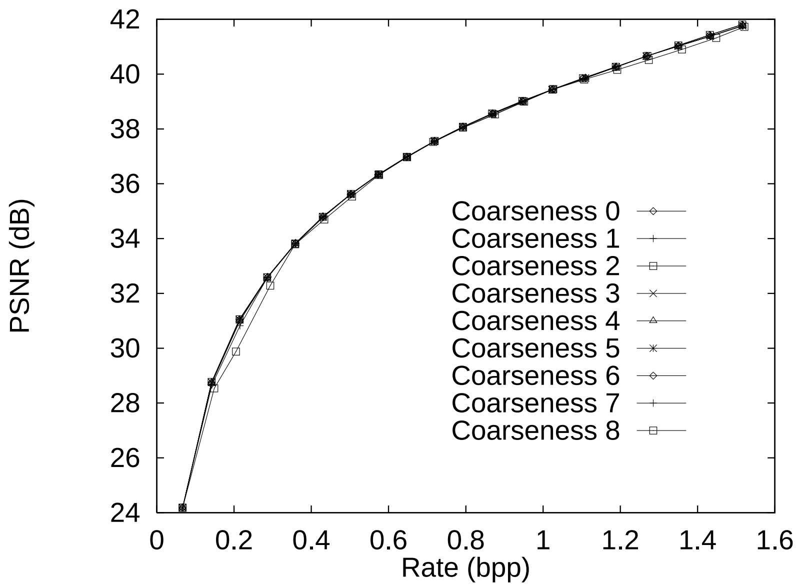


Figure 34: Rate-PSNR plots at various levels of coarseness, with $t = 20$.

the rate-PSNR tradeoff, but with a much smaller running time. The only curve that is perceptibly separate from the rest is that for coarseness 8, the highest coarseness, where only 8 different quantizers are tried for each coefficient. At the next lower coarseness of 7, only 14 different quantizers are tried, and yet the performance is the same as for coarseness 0, where all 255 quantizers are tried. Of course, if the no thresholding is done, ($t = 0$), there is much more variation in rate-PSNR performance with coarseness.

In practice one doesn't need to use a very high value of t, as all the thresholding advantages are achieved at very small values of t. This is shown in Figure 35, which shows the improvement in PSNR at 1.0 bpp for Lena, as a function of t, with coarseness set to zero. The curve shows that $t \approx 10$ is good enough to achieve all thresholding gains, at 1.0 bpp. Our experiments indicate that in practice, a coarseness setting of 6 with a threshold range $t = 20$, works well at all rates, while being very fast.

Finally, we show the dependence of running time on the size of the image. Only

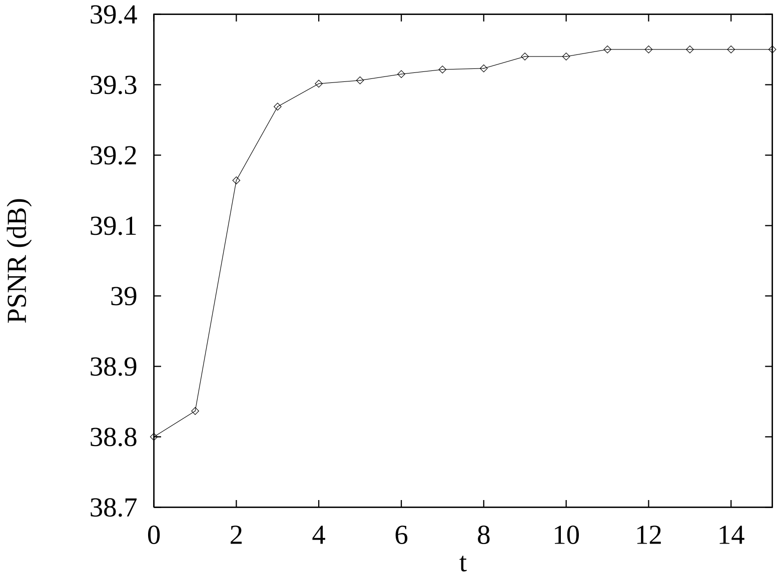


Figure 35: PSNR increases as the number of thresholds tried increases, but achieves its maximum very soon (Lena at 1.0 bpp).

the statistics-gathering step (which just applies DCT to each image block and builds histograms of coefficient values) depends upon the size of the image. Figure 36 shows the running time as a function of the number of pixels in the image, at coarseness 8 and threshold range $t = 0$. Thus, running time increases by about 0.3 seconds for every 4096 8×8 blocks. Note that this holds irrespective of coarseness and threshold range. Our implementation of DCT is reasonably optimized, but uses floating point computations with double precision. The running time can be improved by using a faster DCT algorithm.

Memory Requirement

When gathering image statistics, RD-OPT needs memory to store the histograms. For building the prediction tables, RD-OPT needs the histograms as well as memory for

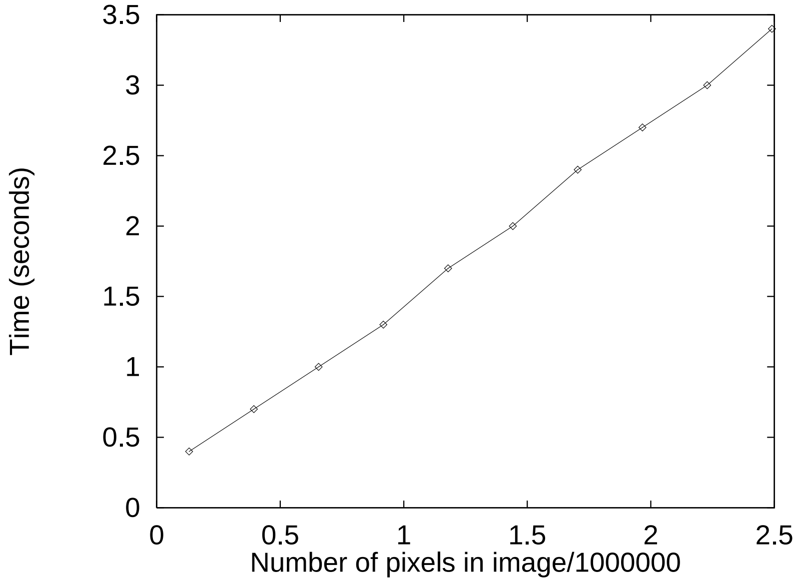


Figure 36: RD-OPT running time as a function of image size.

the tables themselves. After the prediction tables have been built, the histograms are no longer needed and can be freed.

The size of the prediction table is linear in the size of the operating range, and is small for reasonable settings such as a coarseness setting of 6 with threshold range $t = 20$. The size of the histograms depends upon the maximum and minimum values achieved by the coefficients. For 8-bit images, the value span is 4096 for AC coefficients, and since we use buckets of width 0.5, we need at most 8192 buckets per coefficient. For typical images, the high frequency coefficients have a much smaller value span, and require about 50-100 buckets each. If memory is scarce, RD-OPT can be made to run compactly (at the cost of moving away from optimality) by reducing the operating range and by increasing the bucket width from 0.5. The latter can be done without affecting the accuracy of the rates calculated, by restricting the quantizers and thresholds. For example, a bucket width of 4 can be used by restricting the quantizers to be multiples of 8, and using thresholds in steps of 4. Essentially, we need to ensure that equation 4.5 is satisfied, for the buckets, quantizers, and thresholds used. The distortion estimates deteriorate, however, as the bucket width is increased.

4.5.3 Accuracy

The PSNR values predicted by RD-OPT are very accurate, as each coefficient value is accurate to within ± 0.25 using the histograms. If D is the distortion predicted by RD-OPT, and $\tilde{D}$ is the actual distortion, then it follows from the triangle inequality that

$$\sqrt{D} - 0.25 \leq \sqrt{\tilde{D}} \leq \sqrt{D} + 0.25.$$

Figure 37 shows the actual PSNR versus the PSNR predicted by RD-OPT, for the three test images.

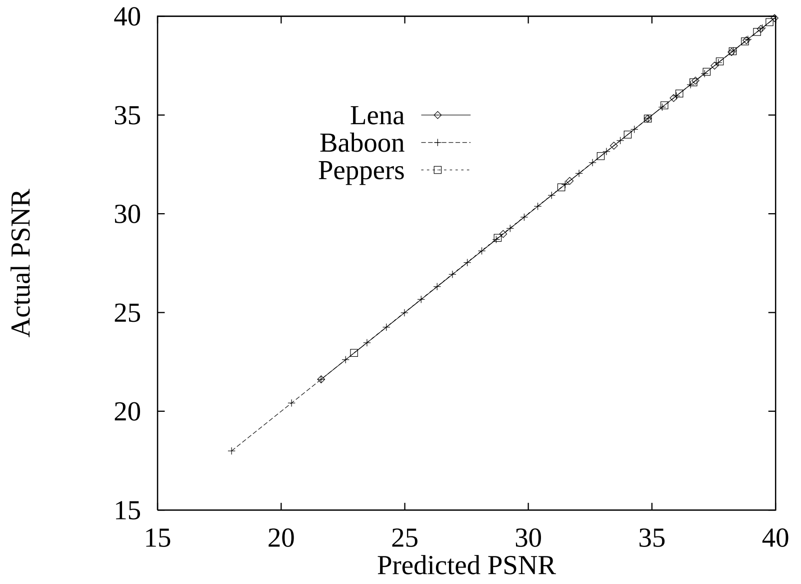


Figure 37: Actual vs. predicted PSNR for the three test images.

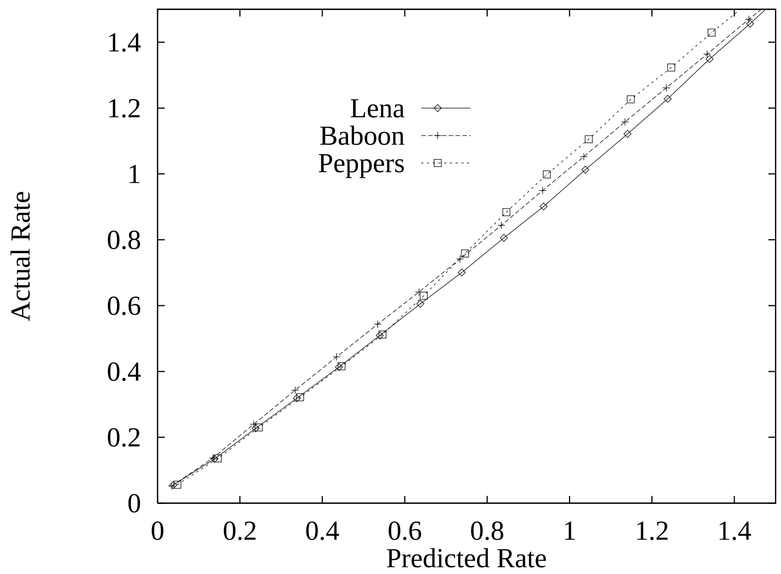


Figure 38: Actual vs. predicted rate for the three test images.

The rate estimates, however, may vary more from their actual values. Still, they are usually within about 0.02 of the actual rates, and hence are reasonably accurate estimates. This can be seen from Figure 38, which shows the actual rates against the predicted rates for the three test images.

4.5.4 Progressive JPEG

The progressive mode of JPEG allows the zig-zag scan of coefficients to be broken up into segments known as *scans*. The DC coefficient must occur in a scan by itself. For example, one way of breaking up a JPEG image into 4 progressive scans is to have the DC scan, followed by coefficients 1 through 5, followed by 6 through 32, and finally 32 through 63. This enables efficient browsing of images across networks—the browser need not fetch all the scans of “uninteresting” images. Since RD-OPT provides estimates of total rate and distortion in terms of sums of coefficient-wise contributions, it is straight-forward to determine scan boundaries subject to any specification of rate/distortion distribution over scans. For example, we can use RD-OPT to set the scan boundaries such that all the scans (except the DC) are nearly equi-sized. Figure 39 shows the progressive PSNR-rate plots for the three test images, when RD-OPT was asked to produce quantization tables for 1.0 bpp, and to split the AC coefficients into four equi-sized scans. The plot shows that the AC scans are, indeed, nearly equi-sized. In fact, the coefficient-wise break-up of the rate and distortion (for each pair of quan-

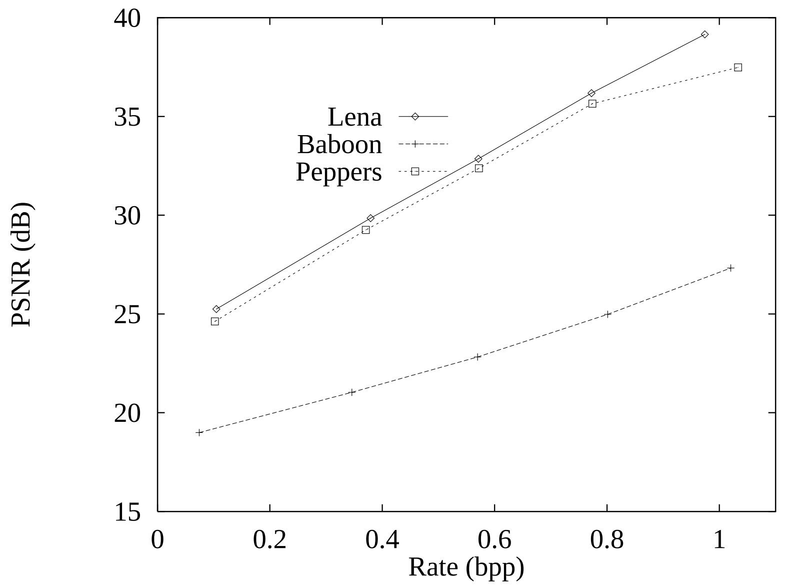


Figure 39: Progressive JPEG compression of the test images at around 1.0 bpp, with four AC scans of comparable sizes.

tization and thresholding tables) is a very good estimate of the internal rate-quality curve, QC-Pt-Crv, in the QCLIC framework.

4.5.5 Other Quality Metrics

RD-OPT can be easily extended to use any different flavor of distortion-based quality metrics. For example, different weights can be assigned to different DCT coefficients, based on their perceptual significance. Moreover, different weights can be assigned to different color planes. Such weights are especially useful if the image is to be compressed at a low rate. In such cases, it is useful to give large weights to just the lowest few DCT coefficients, and to the luminance plane over the chrominance planes for color images.

RD-OPT can work for any quality metric for which a prediction table similar to $D_n(q,t)$ can be built and used. For example, Watson's DCT quality metric, which was built by evaluating the visibility of DCT basis functions using extensive testing of the human vision [Wat93], can be easily used in RD-OPT, as the quality metric is defined by composing together contributions from the 64 coefficients.

4.5.6 Achieving Rate Targets Exactly

Usually, the rate predicted by RD-OPT is within a few percent of the actual rate resulting from JPEG compression. If a greater degree of accuracy is needed, two approaches are possible.

The first approach is to simply compress repeatedly until the target rate is met within the desired tolerance. If the predicted rate is R but the actual rate is R', in the neighborhood of R, RD-OPT is underestimating rate by about $R' - R$. In the next iteration, RD-OPT is asked to produce quantization (and threshold) tables for a target rate of $R - (R' - R)$. Note that RD-OPT is run only once: after it is done with all its steps, it is ready to quickly produce quantization/threshold tables for any number of targets.

Usually, a lower-than-actual prediction of rate is unacceptable. When progressive JPEG is used or allowed, it is easy to simply "trim" the compressed data till the target rate is met within the tolerance. The trimming is not done by simply chopping away some bytes off the compressed data stream. Progressive JPEG allows the coefficients to be divided up into scans. Further, bit-planes of the coefficients can also form scans. Using the difference between predicted and actual rate, and the rate predictions for individual coefficients, RD-OPT can fairly accurately guess the level (coefficient and bit-plane) where the trimming should be done. This is a very useful technique in the QCLIC framework, when only a few curve points are stored.

Chapter 5

Conclusion

Efficient and optimal quality-controlled lossy image compression is very important for efficient management of large amounts of image data. The QCLIC framework, presented in this thesis, is a way to use rate-quality tradeoff as a fundamental attribute of digital images. In spite of the lack of any universal quality metric, simple metrics can be employed to provide useful calibrations of images to best satisfy application needs while meeting media and other constraints. We discussed several useful quality metrics, and presented a survey of the tools and techniques used in image compression. We compared the major compression techniques available at present using several criteria that are important for applications.

We defined the QCLIC-Image object along with its methods, *QCLIC-Compress* and *QCLIC-GetCurve*, in terms of fundamental properties of images, QC-Crv, QC-Pt, QC-Pt-Crv, QC-Pt-Pt. To realize the QCLIC framework in practice for specific image compression techniques and quality metrics, we presented the notion of an enabling technology. An enabling technology is a way to efficiently and closely approximate the elements of the QCLIC-Image structure, within applicable constraints. We discussed common strategies that can be used to design enabling technologies.

We used the QCLIC framework to address the problem of quality-control for sets of images, noting that there are rate-quality tradeoffs across images too, when they are compressed together. We presented the QCLICS algorithm to optimize aggregate measures of rate and quality for sets of images.

For JPEG compression (and other DCT-based techniques), we addressed the previously open problem of optimizing quantization tables and meeting rate-quality targets efficiently. The RD-OPT algorithm described in this thesis can be used for efficient, nearly optimal selection of DCT quantization tables. We described ways to use the RD-OPT algorithm as an enabling technology, for efficient realization of the QCLIC framework for JPEG.

We described several useful applications that we have built to meet practical imagery needs, implementing the QCLIC framework. The QCLIC framework has been used to design components of an elaborate data visualization tool (DEVise) to allow efficient use of images transparently, and as easily as traditional data types. These implementations confirm the utility of QCLIC as a fundamental way of looking at images.

5.1 Contributions

The work presented in this thesis has made the following contributions:

1. QCLIC as a fundamental framework for using digital images.

2. QCLICS: a generic algorithm for quality-controlled compression of sets of images.

3. Knee-normalized PSNR as a uniform and tractable quality metric.

4. Analysis and optimization of uniform scalar quantization with thresholding for Laplacian source distribution.

5. RD-OPT algorithm: QCLIC enabling technology for JPEG (and any orthonormal block transform coding method).

6. Blueprints for designing QCLIC enabling technology for several compression techniques and quality metrics.

7. Design and implementation of applications using the QCLIC framework, used in systems such as DEVise and ZOO.

5.2 Future Work

The main area of future work is in applying the QCLIC framework to design real systems to meet specific application needs. For example, a hardware implementation of limited QCLICS functionality to dynamically pack images in the memory of a digital camera, should be an interesting and rewarding project. Building image servers for shipping images to portable devices, using QCLICS to apportion rate/quality among the various contents of a multimedia web document, are other interesting and useful applications.

Currently used image quality metrics can be improved with more research. The Picture Quality Scale being developed by Algazi *et al* [MKA96] can be used as a benchmark to design simpler metrics. For example, the perceptual weights assigned to errors in the DCT domain can be designed to maximize the correlation with PQS. This should be straightforward, once the PQS metric development stabilizes (it is currently applicable only in a limited quality range, and for 256×256 images).

For scientific applications, designing analysis-specific quality metrics or bounding analysis errors in terms of existing metrics is important. We have done some preliminary investigations in this regard, by analyzing the performance of as unsupervised classification algorithm on imagery obtained from soil scientists, in terms of the accuracy of analysis as a function of distortion [RLNK95]. The results indicate that the classification errors (assuming the classification on the uncompressed image to be the ground truth) are small and random, for moderate distortion. For image-database applications as well, similar analyses need to be done to evaluate the viability of doing content-based search on compressed images, and to bound the errors in these search algorithms in terms of a quality metric.

For vector quantization using large sets of possible codebooks, there is no efficient enabling technology to provide quality control. For applications where extremely fast decoding is critical, vector quantization may be the compression technique of choice, in spite of its poorer rate-quality performance when compared with other techniques. For such applications, rate-quality performance can be improved by using larger codebook sets, and hence the design of QCLIC enabling technologies is important. Techniques to efficiently compute some image statistics and use those to classify and calibrate images with respect to a set of codebooks could be useful in this regard.

In terms of the optimal entropy-constrained uniform quantization problem for Laplacian density, we have shown that the uniform quantizer with optimal thresholding (MT-T-M) works very well, in spite of mid-point reconstruction. But this quantizer, and the MT-U-C quantizer (which has comparable or better performance), do not satisfy the necessary conditions for global optimality. While it appears that their performance *is* close to optimal, obtaining a theoretically optimal quantizer remains an open problem.

Bibliography

[ABMD92] Antonini, M., Barlaud, M., Mathieu, P., and Daubechies, I. Image Coding Using Wavelet Transform. *IEEE Transactions on Image Processing*, 1(2):205–220, April 1992.

[AE97] Algazi, V. R. and Estes Jr., R. R. Comparative Performance of Wavelet and JPEG Coders at High Quality. *Very High Resolution and Quality Imaging, Proc. of the SPIE*, 3025, 1997.

[ANR74] Ahmed, N., Natarajan, T., and Rao, K. R. Discrete Cosine Transform. *IEEE Trans. Computers*, C-2390-3, Jan. 1974.

[AP92] Ahumada Jr., A. J. and Peterson, H. A. Luminance-Model-Based DCT Quantization for Color Image Compression. *Human Vision, Visual Processing, and Digital Display III, B. E. Rogowitz, ed. (Proceedings of the SPIE)*, 1992.

[Bar93] Barnsley, M. F. *Fractals Everywhere.* Academic Press Professional, Cambridge, MA, second edition, 1993.

[Ber71] Berger, T. *Rate Distortion Theory: A Mathematical Basis for Data Compression.* Prentice-Hall, Englewood Cliffs, NJ, 1971.

[Ber72] Berger, T. Optimum Quantizers and Permutation Codes. *IEEE Trans. Inform. Theory*, IT-18(6):759–765, November 1972.

[Ber82] Berger, T. Minimum Entropy Quantizers and Permutation Codes. *IEEE Trans. Inform. Theory*, IT-28(2):149–157, March 1982.

[BK95] Bhaskaran, V. and Konstantinides, K. *Image and Video Compression Standards.* Kluwer Academic Publishers, Boston, MA, 1995.

[CGO94] Cosman, P., Gray, R., and Olshen, R. Evaluating quality of compressed medical images: SNR, subjective rating, and diagnostic accuracy. *Proceedings of the IEEE*, 82:919–932, June 1994.

[CR95] Crouse, M. and Ramchandran, K. JPEG optimization using an entropy-constrained quantization framework. *Proceedings of Data Compression Conference*, pages 342–351, 1995.

[CVC95] Chaddha, N., Vishwanath, M., and Chou, P. A. Hierarchical Vector Quantization of Perceptually Weighted Block Transforms. *Proceedings of Data Compression Conference*, pages 3–12, 1995.

[Dav72] Davisson, L. D. Rate-distortion theory and applications. *Proc. IEEE*, 60(7):800–808, 1972.

[DJL92] DeVore, R. A., Jawerth, B., and Lucier, B. J. Image Compression Through Wavelet Transform Coding. *IEEE Trans. Inform. Theory*, 38(2):719–746, March 1992.

[Eli63] Elias, P. *Information Theory and Coding, ed. by Abramson, N.* McGraw-Hill, New York, 1963.

[FM84] Farvardin, N. and Modestino, J. W. Optimum quantizer performance for a class of non-Gaussian memoryless sources. *IEEE Trans. Inform. Theory*, IT-30(3):485–497, 1984.

[FPR95] Feig, E., Peterson, H., and Ratnakar, V. Image Compression Using Spatial Prediction. *Proc. Inter. Conf. Acoustics, Speech and Signal Processing*, May 1995.

[GG92] Gersho, A. and Gray, R. M. *Vector Quantization and Signal Compression.* Kluwer Academic Publishers, Boston, 1992.

[GP68] Gish, H. and Pierce, J. N. Asymptotically efficient quantizing. *IEEE Trans. Inform. Theory*, IT-14(5):676–683, 1968.

[Gra72] Graham, R. L. An Efficient Algorithm for Determining the Convex Hull of a Finite Planar Set. *Information Processing Letters*, 1(4):132–133, 1972.

[Huf52] Huffman, D. A. A Method for the Construction of Minimum Redundancy Codes. *Proc. IRE.*, 40(9):1098–101, Sept. 1952.

[ILGP96] Ioannidis, Y., Livny, M., Gupta, S., and Ponnekanti, N. ZOO: A Desktop Experiment Management Environment. *Proc. 22nd International VLDB Conference, Bombay, India*, pages 274–285, September 1996.

[Jac92] Jacquin, A. Image Coding Based on a Fractal Theory of Iterated Contractive Image Transformations. *IEEE Transactions on Image Processing*, 1:18–30, 1992.

[Jai89] Jain, A. K. *Fundamentals of Digital Image Processing.* Prentice Hall, Englewood Cliffs, NJ, 1989.

[JFB92] Jacobs, E. W., Fisher, Y., and Boss, R. D. Image compression: A study of the iterated transform method. *Signal Processing*, 29(3):251–263, December 1992.

[JN84] Jayant, N. S. and Noll, P. *Digital Coding of Waveforms.* Prentice Hall, Englewood Cliffs, NJ, 1984.

[JPG] ISO 10918-1 JPEG Draft International Standard and CCITT Recommendation T.81.

[JR94] Jones, P. W. and Rabbani, M. Digital Image Compression. *Digital Image Processing Methods, edited by Dougherty, E. R.*, pages 261–325, 1994.

[Kar47] Karhunen, K. Ueber lineare methoden in der Wahrscheinlichkeitsrechnung. *Ann. Acad. Sci Fenn. Ser A.I. Math. Phys.*, 37, 1947.

[Kel89] Kelly, D. H. Spatial and temporal interactions in color vision. *J. Imag. Technol.*, 15(2):82–89, 1989.

[Knu85] Knuth, D. E. Dynamic Huffman coding. *J. Algorithms*, 6:163–180, 1985.

[Lan84] Langdon, G. G. An introduction to arithmetic coding. *IBM J. Res. Develop.*, 28(2):135–149, 1984.

[LBG80] Linde, Y., Buzo, A., and Gray, R. M. An algorithm for vector quantizer design. *IEEE Trans. Commun.*, COM-28:84–95, January 1980.

[Llo82] Lloyd, S. P. Least squares quantization in PCM. *IEEE Trans. Inform. Theory*, IT-28:129–137, 1982.

[Loe60] Loeve, M. *Probability Theory.* Van Nostrand, Princeton, NJ, Second edition, 1960.

[LRB+97] Livny, M., Ramakrishnan, R., Beyer, K., Chen, G., Donjerkovic, D., Lawande, S., Myllymaki, J., and Wenger, K. DEVise: Integrated Querying and Visual Exploration of Large Datasets. *Proceedings of ACM SIGMOD*, May 1997.

[LRF+92] Lee, H., Rowberg, A. H., Frank, M. S., Choi, H. S., and Kim, Y. Subjective evaluation of compressed image quality. *SPIE Proceedings of Medical Imaging VI: Image Capture, Formatting, and Display*, 1653:241–251, Feb. 1992.

[Max60] Max, J. Quantizing for minimum distortion. *IRE Trans. Inform. Theory*, IT-6(1):7–12, 1960.

[MDS+91] MacMohan, H., Doi, K., Sanada S., Montner, S., Giger, M., Metz, C., Nakamori, N., Yiu, F., Xu, X., Yonekawa, H., and Takeuchi, H. Data compression: effect on diagnostic accuracy in digital chest radiographs. *Radiology*, 178:175–179, 1991.

[MHY82] Murakami, H., Hatori, Y., and Yamamoto, H. Comparison between DPCM and Hadamard transform coding in the composite coding of the NTSC color TV signal. *IEEE Trans. Commun.*, COM-30:469–479, March 1982.

[MKA96] Miyahara, M., Kotani, K., and Algazi, V. R. Objective Picture Quality Scale (PQS) For Image Coding. *Technical report, Center for Image Processing and Integrated Computing, University of California, Davis*, 1996.

[MS93] Monro, D. M. and Sherlock, B. G. Optimum DCT Quantization. *Proceedings of Data Compression Conference*, pages 188–194, 1993.

[Mul85] Mullen, K. T. The Contrast Sensitivity of Human Color Vision to Red-Green and Blue-Yellow Chromatic Gratings. *J. Physiol.*, 359381-400, 1985.

[NH95] Netravali, A. N. and Haskell, B. G. *Digital Pictures.* Plenum Press, New York, 1995.

[PM93] Pennebaker, W. B. and Mitchell, J. L. *JPEG Still Image Data Compression Standard.* Van Nostrand Reinhold, New York, 1993.

[PMLA88] Pennebaker, W. B., Mitchell, J. L., Langdon Jr., G. G., and Arps, R. B. An overview of the basic principles of the Q-coder adaptive binary arithmetic coder. *IBM J. Res. Develop.*, 32(6):717–726, 1988.

[Pra91] Pratt, W. K. *Digital Image Processing.* John Wiley & Sons, New York, 1991.

[RFT94] Ratnakar, V., Feig, E., and Tiwari, P. Fractal Based Hybrid Compression Schemes. *Proceedings of SPIE's VCIP*, 1994.

[RFVK94] Ratnakar, V., Feig, E., Viscito, E., and Kalluri, S. Runlength encoding of quantized DCT coefficients. *IBM RC 19693 (87318) 8/5/94 (Also in Proceedings of SPIE '95)*, 1994.

[RG83] Reininger, R. C. and Gibson, J. D. Distributions of the Two-Dimensional DCT Coefficients for Images. *IEEE Trans. Communications*, COM-31(6):835–839, June 1983.

[RL94] Ratnakar, V. and Livny, M. Performance of Customized DCT Quantization Tables on Scientific Data. *Science Information Management and Data Compression Workshop Proceedings, NASA Conference Publication 3277*, pages 1–8, Sept 1994.

[RL95] Ratnakar, V. and Livny, M. RD-OPT: An Efficient Algorithm For Optimizing DCT Qunatization Tables. *Proceedings of Data Compression Conference (Also, Technical Report 1257, Dept of Computer Sciences, UW-Madison)*, pages 332–341, 1995.

[RL96] Ratnakar, V. and Livny, M. Extending RD-OPT with Global Thresholding for JPEG Optimization. *Proceedings of Data Compression Conference*, pages 379–386, 1996.

[RLNK95] Ratnakar, V., Livny, M., Norman, J. M., and Kucharik, K. Classification on compressed images with bounded loss. *International Geoscience and Remote Sensing Symposium Proceedings*, July 1995.

[RY90] Rao, K. R. and Yip, P. *Discrete Cosine Transform: Algorithms, Advantages, Applications*. Academic Press, Inc, San Diego, California, 1990.

[Sch85] Schafer, R. High-definition television production standard: an opportunity for optimal color processing. *SMPTE (Society of Motion Picture and Television Engineers) J.*, pages 749–758, July 1985.

[Sha48] Shannon, C. E. The Mathematical Theory of Communications. *Bell Syst. Tech. J.*, 27:379–423,635–656, 1948.

[Sha59] Shannon, C. E. Coding theorems for a discrete source with a fidelity criterion. *IRE National Convention Record, Part 4*, pages 142–163, 1959.

[Sha93] Shapiro, J. M. Embedded Image Coding Using Zerotrees of Wavelet Coefficients. *IEEE Trans. Signal Processing*, 41(12):3445–3462, December 1993.

[SP96a] Said, A. and Pearlman, W. A. A New, Fast, and Efficient Image Codec Based on Set Partitioning in Hierarchical Trees. *IEEE Trans. Circuits Syst. Video Technol.*, 6(3):243–250, June 1996.

[SP96b] Said, A. and Pearlman, W. A. An Image Multiresolution Representation for Lossless and Lossy Compression. *IEEE Transactions on Image Processing*, 5(9):1303–1310, September 1996.

[Tes79] Tescher, A. G. Transform image coding. *Advances in Electronics and Electron Physics*, Suppl. 12:113–115, 1979.

[VB67] Van Ness, F. I. and Bouman, M. A. Spatial Modulation Transfer in the Human Eye. *Journal of the Optical Society of America*, 57(3):401–406, March 1967.

[Vit87] Vitter, J. S. Design and analysis of dynamic Huffman codes. *J. Assoc. Comput. Mach.*, 34(4):825–845, 1987.

[Wat93] Watson, A. B. DCT quantization matrices visually optimized for individual images. *Human Vision, Visual Processing, and Digital Display IV, B. E. Rogowitz, ed. (Proceedings of the SPIE)*, 1993.

[WG93] Wu, S. and Gersho, A. Rate-constrained picture-adaptive quantization for JPEG baseline coders. *Proc. Inter. Conf. Acoustics, Speech and Signal Processing*, 5:389–392, April 1993.

[WNC87] Witten, I. H., Neal, R. M., and Cleary, J. G. Arithmetic coding for data compression. *Commun. ACM*, 30(6):520–540, 1987.

[ZASB95] Zandi, A., Allen, J. D., Schwartz, E. L., and Boliek, M. CREW: Compression with Reversible Embedded Wavelets. *Proceedings of Data Compression Conference*, pages 212–221, March 1995.