RD-OPT: An Efficient Algorithm For Optimizing DCT Quantization Tables*

Viresh Ratnakar
University of Wisconsin-Madison
Computer Sciences Department
Madison, WI 53706
Phone: (608) 262-6627
Email: ratnakar@cs.wisc.edu

Miron Livny
University of Wisconsin-Madison
Computer Sciences Department
Madison, WI 53706
Phone: (608) 262-0856
Email: miron@cs.wisc.edu

Abstract

The Discrete Cosine Transform (DCT) is widely used in lossy image and video compression schemes such as JPEG and MPEG. In this paper we describe RD-OPT, an efficient algorithm for constructing DCT quantization tables with optimal rate-distortion tradeoffs for a given image. The algorithm uses DCT coefficient distribution statistics in a novel way and uses a dynamic programming strategy to produce optimal quantization tables over a wide range of rates and distortions. It can be used to compress images at any desired signal-to-noise ratio or compressed size.

1 Introduction

The Discrete Cosine Transform [ANR74] lies at the heart of most commonly used lossy image and video compression schemes [PM93, MP91]. The extent of compression achieved depends upon the coarseness of quantization of the transform coefficients. The coarser the quantization, the lesser the entropy of the quantized coefficients. But coarse quantization also leads to poor quality of the decompressed image. Thus, the quantization table used directly determines the rate-distortion tradeoff, i.e., the compression-quality tradeoff.

Several approaches have been tried in order to design quantization tables for particular distortion or rate specifications. The most common of these is to use a default table and scale it up or down by a scalar multiplier to vary quality and compression. We have shown in [RL94] that this might not give the best tradeoff possible. Other approaches include psycho-visual model based quantization [AP92, Wat93], rate-distortion model based quantization [Jai89], and stochastic optimization techniques [MS93].

In this paper we present RD-OPT, an efficient algorithm for optimum quantization table design that does not rely on visual or rate-distortion models. The algorithm admits a wide range of quality measures (including PSNR, weighted PSNR) and produces quantization tables optimizing the tradeoff between quality and compressed size. A key feature of the algorithm is that it simultaneously optimizes quantization tables over a wide range of rates and distortions.

2 Image compression based on the Discrete Cosine Transform

The human visual system is not very sensitive to sudden changes in intensity across an image [VB67]. Lossy image compression techniques strive to discard that part of the image structure which is less perceptible to the eye. The two-dimensional Discrete Cosine Transform (referred to hereafter as DCT) offers an efficient way to break up the underlying structure of an image into different spatial-frequency components. The high-frequency components are less perceptible to the eye compared to the low-frequency components.

*This work was partially supported by NASA grant NAGW-3914 and NSF grant IRI-9224741.

Thus, the DCT orders the information-content of an image into parts with different degrees of visual importance. These parts can then be selectively discarded or stored with varying degrees of accuracy, for lossy compression of the image. The compression-ratio increases as more and more information is thrown away. DCT-based compression techniques typically allow the user to specify a table (called the quantization table) that stipulates the level of accuracy with which each spatial-frequency component is to be stored.

We now briefly describe the basic steps used in DCT-based image compression and decompression, and thereby present some notation. Details on various standards can be found in the standard documents. Details on DCT itself can be found in [RY90].

Let I be a $W \times H$ image with pixel values in the range $[0 \ldots M]$. The DCT-based compression process consists of the following steps.

1. The image is divided into 8×8 blocks. To each image block f, the DCT is applied to get an 8×8 block $\hat{f}$ of DCT coefficients. Each coefficient represents the amount of a particular spatial-frequency content in f. The lowest frequency coefficient (also called the DC coefficient) is $\hat{f}(0,0)$ while the highest frequency coefficient is $\hat{f}(7,7)$.

2. An 8×8 matrix of integers Q, called the quantization table, is used to quantize the coefficients in $\hat{f}$ to form $\hat{f}_Q$. For notational convenience, we number the 64 entries in each 8×8 image block and each 8×8 block of DCT coefficients in raster-scan order, and use this ordering to refer to individual entries in the various blocks. Thus, $f(u,v)$ is referred to as $f[8u+v]$. Using this notation,

   $$\hat{f}_Q[n] = \hat{f}[n] /\!\!/ Q[n], \quad 0 \leq n \leq 63.$$

   Here $/\!\!/$ represents division followed by rounding to the nearest integer[1].

3. The block $\hat{f}_Q$ is entropy-coded, using (for example) Huffman codes to exploit similarities across blocks, to give the compressed block $E(\hat{f}_Q)$ [PM93]. The sequence of these compressed blocks forms the compressed image.

The decompression process reverses these steps as follows:

1. Each entropy-coded block $E(\hat{f}_Q)$ is decoded to get the corresponding block of quantized coefficients $\hat{f}_Q$.

2. Dequantization is done to construct the block $\hat{f}'$, as follows:

   $$\hat{f}'[n] = \hat{f}_Q[n] \cdot Q[n], \quad 0 \leq n \leq 63.$$

3. The two-dimensional Inverse Discrete Cosine Transform (IDCT), is applied to $\hat{f}'$ to get the decompressed image block f'. These decompressed blocks form the decompressed image I'.

The compression and decompression processes can be summarized as:

$$\boxed{\begin{array}{ccccccc} f & \xrightarrow{\text{DCT}} & \hat{f} & \xrightarrow{\text{Quantization}} & \hat{f}_Q & \xrightarrow{\text{Entropy-coding}} & E(\hat{f}_Q) \\ & & & & & & \downarrow \\ f' & \xleftarrow{\text{IDCT}} & \hat{f}' & \xleftarrow{\text{Dequantization}} & \hat{f}_Q & \xleftarrow{\text{Decoding}} & E(\hat{f}_Q) \end{array}}$$

The lossiness of the compression is essentially because of the quantization step ($\hat{f} \rightarrow \hat{f}_Q$), as in general,

$$\hat{f}'[n] = \hat{f}_Q[n] \cdot Q[n] = (\hat{f} /\!\!/ Q[n]) \cdot Q[n] \neq \hat{f}[n].$$

[1] For any $b > 0$, $a /\!\!/ b = \begin{cases} \lfloor \frac{a}{b} + 0.5 \rfloor & \text{if } a \geq 0 \\ -\lfloor \frac{-a}{b} + 0.5 \rfloor & \text{if } a < 0. \end{cases}$

This causes differences in pixel values between the original image block f and its approximation, the decompressed image block f'. We refer to the mean-squared-error between f and f' as *distortion* in f caused by Q and denote it by $d(f, Q)$.

$$d(f, Q) = \frac{1}{64} \sum_{n=0}^{63} (f[n] - f'[n])^2.$$

The distortion $D(I, Q)$ between an image I and its approximation I' due to quantization by Q is similarly defined as the mean-squared-error between corresponding pixel values. Clearly, $D(I, Q)$ is the mean value of all $d(f, Q)$ over all constituent blocks f in I. The distortion $D(I, Q)$ is used to judge the "quality" of I', in quality measures such as Signal-to-Noise Ratio (SNR) and Peak Signal-to-Noise Ratio (PSNR):

$$\begin{aligned} \text{PSNR} &= 10 \log_{10} \frac{M^2}{D(I, Q)}, \\ \text{SNR} &= 10 \log_{10} \frac{S}{D(I, Q)}. \end{aligned}$$

Here S is the mean-squared pixel value over I. Higher distortion implies poorer quality and vice versa. Clearly, as entries in Q are made higher, $D(I, Q)$ tends to increase.

We can also define the distortion between the DCT coefficient-block $\hat{f}$ and its approximation $\hat{f}'$ resulting from quantization by Q, as

$$d(\hat{f}, Q) = \frac{1}{64} \sum_{n=0}^{63} (\hat{f}[n] - \hat{f}'[n])^2.$$

Our approach to designing rate-distortion-optimal quantization tables exploits the following nice property of the DCT:

$$d(f, Q) = d(\hat{f}, Q).$$

That is, the mean squared error in pixel-domain is the same as the mean squared error in DCT-domain. This can be seen as follows:

$$\begin{array}{rrcll} & \text{DCT}(f) &=& \hat{f} & \\ & \text{IDCT}(\hat{f}') &=& f' & \\ \Rightarrow & \text{DCT}(f') &=& \hat{f}' & \\ \Rightarrow & \text{DCT}(f - f') &=& \hat{f} - \hat{f}', & \text{using linearity of DCT} \\ \Rightarrow & \sum_{n=0}^{63} (f[n] - f'[n])^2 &=& \sum_{n=0}^{63} (\hat{f}[n] - \hat{f}'[n])^2, & \text{since DCT preserves } L^2 \text{ norms} \\ \Rightarrow & d(f, Q) &=& d(\hat{f}, Q). & \end{array}$$

This implies that $D(I, Q)$ can be split as a sum into distortions in various DCT coefficients. Let $D_n(I, q)$ be defined as

$$D_n(I, q) = \frac{1}{64} \text{Mean}\{(\hat{f}[n] - \hat{f}'[n])^2\},$$

Where $\hat{f}'[n] = (\hat{f}[n] /\!\!/ q) \cdot q$, and the mean is taken over all the blocks in the image. Then

$$D(I, Q) = \sum_{n=0}^{63} D_n(I, Q[n]).$$

Since we are mainly concerened with the effects of different quantization tables on a given image, we denote the distortion $D(I, Q)$ simply as $D(Q)$. Similarly, we denote the distortion $D_n(I, q)$ in the n^{th} coefficient by $D_n(q)$. Then,

$$D(Q) = \sum_{n=0}^{63} D_n(Q[n]). \tag{1}$$

This decomposition of $D(Q)$ into contributions from individual coefficients is important as it simplifies the task of minimizing $D(Q)$ to that of minimizing a sum, each of whose components depends on just one entry in Q. We now show how the rate (or compressed size) resulting from Q can also be split similarly into a sum. These two properties of the DCT allow the problem of optimizing the rate-distortion tradeoff to be solved using a dynamic programming approach for minimizing two sums.

2.1 Bit-rates for DCT-based compression

The degree of compression achieved is usually expressed in terms of the *rate* of the compressed image, which is the number of bits used per pixel:

$$\text{rate} = \frac{\text{size of compressed image in bits}}{\text{number of pixels in the image}}.$$

Low rates are achieved when the quantized blocks $\hat{f}_Q$ have similar entries (low entropy). The most common case is that of a coefficient being quantized to zero. The more zeros there are in $\hat{f}_Q$, the fewer bits it would take to store it. Thus, increasing the entries in Q tends to decrease the rate. We denote the rate resulting with the use of a particular quantization table Q as $R(Q)$.

DCT has the nice property of being very close to the Karhunen-Loeve-Hotelling tranform, a transform that produces uncorrelated coefficients [ANR74]. The lack of correlation between coefficients allows the rate to be decomposed as a sum of contributions from individual coefficients. It has been shown in [RFVK94] that the coefficient-wise average of entropies of the quantized DCT coefficients is a very good estimate of the rate resulting from two-pass Huffman coding of runlengths. This allows us to approximate $R(Q)$ as a sum of rates of individual coefficients. Let $R_n(q)$ be defined as

$$R_n(q) = \frac{1}{64}\text{Entropy}\{(\hat{f}[n] /\!\!/ q)\},$$

Where the entropy is measured over all the blocks in the image[2]. Then

$$R(Q) \approx \sum_{n=0}^{63} R_n(Q[n]). \tag{2}$$

Thus, $R(Q)$ can also be decomposed into a sum of contributions from individual coefficients, just like $D(Q)$.

3 The RD-OPT algorithm

In this section, we present the RD-OPT algorithm for constructing quantization tables with optimal rate-distortion tradeoffs for a given image. It is desirable to have low rate (high compression) and low distortion (high quality). However, varying Q has opposite effects on $D(Q)$ and $R(Q)$. The distortion $D(Q)$ tends to increase and the rate $R(Q)$ tends to decrease as the entries in Q are made larger. The tradeoff between $D(Q)$ and $R(Q)$ is different for different images.

The problem of choosing Q to optimize the rate-distortion tradeoff for a given image I can be stated in two ways:

1. Given a target distortion Δ, find Q such that $D(Q) \leq \Delta$ and the rate $R(Q)$ is minimized.
2. Given a target rate B bits per pixel (bpp), find Q such that $R(Q) \leq B$ and the distortion $D(Q)$ is minimized.

We call the quantization tables Q that satisfy these conditions (for some Δ or B) *RD-optimal.*

RD-OPT takes an image I as input and produces RD-optimal quantization tables for a wide range of rates and distortions. The contributions to rate and distortion of individual coefficients ($R_n(Q[n])$ and $D_n(Q[n])$, respectively) just depend on the entry $Q[n]$ of Q. RD-OPT first calculates $R_n(Q[n])$ and $D_n(Q[n])$ for each possible value of $Q[n]$, and then uses a dynamic programming approach to minimize sums of $R_n(Q[n])$ against sums of $D_n(Q[n])$. To calculate $R_n(q)$ and $D_n(q)$ for each possible q, a preliminary pass through the image is run to gather DCT statistics which are used in a novel way. RD-OPT can be described at a high level as:

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

Algorithm RD-OPT

Input: An image I of width W and height H, with pixel values in the range $[0 \ldots M]$.

Output: RD-optimal DCT quantization tables Q.

Step 1. Gather DCT statistics for the image (Procedure `GatherStats`).

Step 2. Use the statistics to calculate $R_n(q)$ and $D_n(q)$ for each possible q (Procedures `FillR` and `FillD`).

Step 3. Use dynamic programming to optimize $R(Q)$ against $D(Q)$ (Procedure `FillLeastD`).

We now present each of the three steps in RD-OPT in detail. Pseudo-code for all the procedures described in this section can be found in Appendix A.

3.1 Step 1: Gathering DCT statistics

The task for this step is to gather DCT statistics for the image, which can be used to efficiently answer the questions:

1. How many times does the n^{th} coefficient get quantized to value v when $Q[n] = q$?
2. What is the mean-squared error for the n^{th} coefficient when $Q[n] = q$?

For any real number c, let

$$\text{Bucket}(c) = \begin{cases} \lfloor 2c \rfloor & \text{if } c \geq 0 \\ -\lfloor -2c \rfloor & \text{if } c < 0. \end{cases}$$

It can be shown that for any integer $q \geq 1$,

$$c /\!\!/ q \;=\; \text{Bucket}(c) /\!\!/ (2q).$$

Hence, it suffices to gather statistics by counting the number of times each DCT coefficient takes a value in a particular bucket, as this count can then be used to calculate the number of times a particular quantized coefficient takes a particular value. The unquantized coefficient value itself can be approximated to within[3] ± 0.25.

Let $\texttt{OccursCount}[0 \ldots 63][-2\texttt{VMAX} \ldots 2\texttt{VMAX}]$ be an array with the value $\texttt{OccursCount}[n][v]$ being the number of blocks where the n^{th} DCT coefficient c_n is such that $\text{Bucket}(c_n) = v$. The constant `VMAX` is the maximum absolute value any DCT coefficient can take (for 1-byte samples, $M = 255$ and $\texttt{VMAX} = 2048$). The array `OccursCount` is filled by the procedure `GatherStats`.

`GatherStats` works by going through each 8×8 block f in I and calculating its Discrete Cosine Transform g. For each coefficient $g[n]$, the count $\texttt{OccursCount}[n][\text{Bucket}(g[n])]$ is incremented by one.

3.2 Step 2: Calculating $R_n(q)$ and $D_n(q)$.

Let the possible range of values of any quantization table entry q be $1 \leq q \leq \texttt{MAXQ}$. Let $\texttt{R}[0 \ldots 63][1 \ldots \texttt{MAXQ}]$ and $\texttt{D}[0 \ldots 63][1 \ldots \texttt{MAXQ}]$ be two-dimensional arrays. The task of this step is to fill these arrays using the array `OccursCount` such that,

$$\begin{aligned} \texttt{R}[n][q] &= R_n(q) \\ \texttt{D}[n][q] &= D_n(q) \end{aligned}$$

This is accomplished by procedures `FillR` and `FillD`.

`FillR` fills the array `R` by calculating the entropy of the n^{th} coefficient when quantized by q, for all n and q. For each possible quantized value (`QuantizedVal`), the variable `Count` is used to compute the number of times the n^{th} coefficient gets quantized (by q) to `QuantizedVal`. `Count` is simply the sum of all

[3] To make sure this is the case, bucket 0 which corresponds to values in the interval $(-0.5, 0.5)$ must be split into two buckets, but this detail is ignored here for clarity.

$\mathtt{OccursCount}[n][v]$ such that $v /\!\!/ (2q)$ is equal to $\mathtt{QuantizedVal}$. If F is the total number of blocks in I, then, the entropy is calculated as:

$$\mathtt{Entropy} = -\sum_{\mathtt{QuantizedVal}} \frac{\mathtt{Count}}{F} \log_2 \frac{\mathtt{Count}}{F}.$$

$\mathtt{FillD}$ fills the array $\mathtt{D}$ by calculating the error in quantizing the n^{th} coefficient by q, for each n and q. For each integer v in the range $-2\mathtt{VMAX} \ldots 2\mathtt{VMAX}$, the n^{th} coefficient gets quantized to the value $\mathtt{QuantizedVal}$ $(= v /\!\!/ (2q))$ in $\mathtt{OccursCount}[n][v]$ blocks. The actual (unquantized) value of the coefficient in each of these blocks is estimated by the variable $\mathtt{OriginalVal}$, to within ± 0.25. For each v, the error is incremented by $\mathtt{OccursCount}[n][v] \cdot (\mathtt{OriginalVal} - q \cdot \mathtt{QuantizedVal})^2$.

3.3 Step 3: Finding RD-optimal Q

The arrays $\mathtt{R}$ and $\mathtt{D}$ are used in this step to find rate-distortion-optimal quantization tables, using a dynamic programming (DP) approach. For this, one of $R(Q)$ and $D(Q)$ needs to be discretized to integral values, to be used as an index in the DP table. This introduces some error in the quantity discretized, which is analyzed in the next section.

Let $\mathtt{BPPSCALE}$ be a large integer constant. We discretize $R(Q)$ to integral values by multiplying each $\mathtt{R}[n][q]$ with $\mathtt{BPPSCALE}$ and rounding off. This is done in Procedure $\mathtt{FillR}$ itself. For the rest of this section, we will be referring to the discretized values, when we refer to *rate*, and to $R(Q)$ and $R_n(q) (= \mathtt{R}[n][q])$.

Let $\mathtt{MAXRATE}$ be the discretized value of the highest rate for which we are interested in finding an RD-optimal quantization table. Let $\mathtt{LeastD}[0\ldots 63][0\ldots\mathtt{MAXRATE}]$ be an array whose entries have the following definition: $\mathtt{LeastD}[n][s]$ is the least total distortion for coefficients numbered 0 through n such that the total (discretized) rate (for these coefficients) is exactly s. That is, $\mathtt{LeastD}[n][s]$ is the least value (over all Q) of $\sum_{k=0}^{n} \mathtt{D}[k][Q[k]]$ subject to the constraint

$$\sum_{k=0}^{n} \mathtt{R}[k][Q[k]] = s.$$

Theorem 1 states the property of $\mathtt{LeastD}$ that allows it to be computed using a dynamic programming approach.

Theorem 1 *For each n, $1 \le n \le 63$, and each s, $0 \le s \le \mathtt{MAXRATE}$, let $\mathcal{D}(n, s)$ be the set*

$$\mathcal{D}(n,s) = \left\{ \mathtt{D}[n][q] + \mathtt{LeastD}[n-1][s'] \;\middle|\; \begin{array}{l} 1 \le q \le \mathtt{MAXQ}, \\ s' = s - \mathtt{R}[n][q], \\ s' \ge 0 \end{array} \right\}.$$

Then,

$$\mathtt{LeastD}[n][s] = \begin{cases} \min \mathcal{D}(n,s) & \text{if } \mathcal{D}(n,s) \text{ is non-empty} \\ \infty & \text{otherwise.} \end{cases}$$

Proof: (See Appendix B).

The array $\mathtt{LeastD}$ is filled by procedure $\mathtt{FillLeastD}$ using Theorem 1. $\mathtt{FillLeastD}$ starts with each entry in $\mathtt{LeastD}$ set to ∞ and then fills the rows one by one. Row number n is filled using row number $n-1$, $\mathtt{D}[n][\ldots]$, and $\mathtt{R}[n][\ldots]$. For each q $(0 \le q \le \mathtt{MAXQ})$ and each s' $(0 \le s' \le \mathtt{MAXRATE})$, $\mathtt{D}[n][q] + \mathtt{LeastD}[n-1][s']$ is compared with $\mathtt{LeastD}[n][s]$, where $s = s' + \mathtt{R}[n][q]$. If the former is lesser, then it replaces the latter. To keep track of the choices made at any point, $\mathtt{FillLeastD}$ maintains another data structure, $\mathtt{QChoice}[0\ldots 63][0\ldots\mathtt{MAXRATE}]$. $\mathtt{QChoice}[n][s]$ is set to the value q that gives the entry in $\mathtt{LeastD}[n][s]$. That is, $\mathtt{QChoice}[n][s]$ is set to q whenever $\mathtt{D}[n][q] + \mathtt{LeastD}[n-1][s']$ is entered in $\mathtt{LeastD}[n][s]$.

Now, if a total distortion requirement Δ is to be met, it's straightforward to find the least s such that $\mathtt{LeastD}[63][s] \le \Delta$. Similarly, if a rate requirement B is to be met, it's easy to find s such that $s \le B$ and $\mathtt{LeastD}[63][s]$ is the minimum over all such s. Thus, in both cases one can find a starting point s in the

63^{rd} row. To recover the desired quantization table Q from that point, procedure **RecoverQ** is used. This procedure recovers the quantization table Q for target rate s by setting $Q[63]$ to `QChoice`$[63][s]$ and then working its way up the rows as follows. For going from row number n to row number $n-1$, s is decremented by `R`$[n][Q[n]]$ and then $Q[n-1]$ is set to `QChoice`$[n-1][s]$.

4 Analysis of RD-OPT

In this section we discuss the running time of RD-OPT and the errors resulting from discretization.

4.1 Complexity

`GatherStats` runs in time about that required to apply the DCT once to each block in the image. `FillD` and `FillR` each run in time less than a constant times $64 \times$ `MAXQ` $\times$ `VMAX`. `FillLeastD` runs in time less than a constant times $64 \times$ `MAXQ` $\times$ `MAXRATE`. In any practical implementation, these times can be substantially reduced. The necessary optimizations are straightforward and are omitted here for clarity.

4.2 Error analysis

In calculating the distortion in `FillD`, the value `OriginalVal` (see line 6 of the pseudo-code in Appendix A) is an estimate of the actual DCT coefficient, accurate to within ± 0.25. Let I_ϵ be the image for which the DCT coefficients are the same as those obtained as `OriginalVal` in `FillD`. Thus, if a particular DCT coefficient for I has the value c, the corresponding coefficient for I_ϵ would have the value $\text{Bucket}(c)/2 + 0.25$ if $c \geq 0$ and $\text{Bucket}(c)/2 - 0.25$ if $c < 0$. The absolute difference between this value and c is at most 0.25. The distortion obtained by RD-OPT for any quantization table Q is $D(I_\epsilon, Q)$ rather than $D(I, Q)$. The triangle inequality can be used to show that

$$\sqrt{D(I_\epsilon, Q)} - 0.25 \quad \leq \quad \sqrt{D(I, Q)} \quad \leq \quad \sqrt{D(I_\epsilon, Q)} + 0.25.$$

In the most common case, RD-OPT will be asked to optimize PSNR. For $M = 255$, the reported PSNR, P_ϵ, will be:

$$P_\epsilon = 10 \log_{10} \frac{255^2}{D(I_\epsilon, Q)}.$$

Hence, the actual PSNR, P, is bounded as follows:

$$P_\epsilon - 20 \log_{10}(1 + \alpha(P_\epsilon)) \leq P \leq P_\epsilon - 20 \log_{10}(1 - \alpha(P_\epsilon)),$$

where $\alpha(P_\epsilon) = \frac{10^{P_\epsilon/20}}{1020}$.

The total error in $R(Q)$ is at most

$$64 \times 0.5/\texttt{BPPSCALE}$$

plus the error in estimating rate as the coefficient-wise sum of entropies. The latter component goes down as $R(Q)$ increases, and is typically around 0.02 bits per pixel [RFVK94].

5 Performance results

We have implemented RD-OPT in C on various platforms. Figure 1 shows the performance of RD-OPT running on an IBM POWERstation 370. The test image used was the well-known 512×512 grayscale image of Lena. The figure tabulates running times and accuracy of results for various values of `BPPSCALE`. In each case, RD-OPT was asked to compute optimal tables over the range 0.0 to 1.0 bits per pixel. These tables were used to compress the image using the Independent JPEG Group's JPEG compression software. The table shows the actual rate and actual PSNR when Lena was compressed using the quantization table picked by RD-OPT with a target rate of 0.8 bits per pixel. The scaled default JPEG table gave a PSNR of 34.90 dB at 0.796 bits per pixel.

BPPSCALE	Running time (secs.)	Predicted Rate (bits per pixel)	Actual Rate (bits per pixel)	Predicted PSNR (dB)	Actual PSNR (dB)
1000	14.5	0.800	0.822	35.809	35.798
2500	19.8	0.800	0.801	35.726	35.714
5000	28.8	0.800	0.798	35.708	35.695
7500	37.6	0.800	0.796	35.705	35.694
10000	46.7	0.800	0.795	35.701	35.690
12500	56.0	0.800	0.794	35.700	35.687
15000	64.6	0.800	0.794	35.699	35.688
20000	82.2	0.800	0.795	35.699	35.687
50000	189.8	0.800	0.794	35.697	35.686

Figure 1: Running time and accuracy for different values of `BPPSCALE`

The running time in each case includes the time spent in `GatherStats`, `FillD`, and `FillR`, which is independent of `BPPSCALE`, and is about 10.91 seconds. Out of 10.91 seconds, about 2.5 seconds are spent in `GatherStats`, and the rest in `FillD` and `FillR`. These procedures haven't been optimized in the current implementation, and we expect to bring the running time down even more, in the near future.

We have used RD-OPT on a wide variety of images, with similar results.

References

[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.

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

[MP91] MPEG I draft: Coding of Moving Pictures and associated audio for digital storage, 1991. Document ISO/IEC-CD-11172.

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

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

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

[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.

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

[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.

[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.

A RD-OPT Procedures

```
Procedure GatherStats

   Input:  Image I
   Output: Array OccursCount[0..63][-2VMAX..2VMAX]

  1. Initialize OccursCount to 0 everywhere
  2. For each 8x8 block f in I
  3.   g := DCT(f)
  4.   For n := 0 to 63
  5.     v := Bucket(g[n])
  6.     OccursCount[n][v]++
```

```
Procedure FillR

   Input:  Array OccursCount[0..63][-2VMAX..2VMAX]
   Output: Array R[0..63][1..MAXQ]

  1. F := Number of 8x8 blocks in the image
  2. For n := 0 to 63
  3.   For q := 1 to MAXQ
  4.     Entropy = 0
  5.     For QuantizedVal := (-VMAX) // q to VMAX // q
           /* QuantizedVal is the quantized value */
  6.       Count := 0 /* Count is the # of times the value QuantizedVal occurs */
  7.       For each v such that v // (2q) == QuantizedVal
  8.         Count := Count + OccursCount[n][v]
  9.       Probab := Count/F
 10.       If (Probab > 0) then
 11.         Entropy := Entropy - (Probab * Log2(Probab))
 12.     R[n][q] :=  ((Entropy / 64.0) * BPPSCALE) // 1
```

```
Procedure FillD

   Input:  Array OccursCount[0..63][-2VMAX..2VMAX]
   Output: Array D[0..63][1..MAXQ]

  1. N := Number of pixels in the image
  2. For n := 0 to 63
  3.   For q := 1 to MAXQ
  4.     D[n][q] := 0
  5.     For v := -2VMAX to 2VMAX
           /* OriginalVal is the original coefficient value, within 0.25 */
  6.       OriginalVal = v/2.0 + ((v < 0) ? -0.25 : 0.25)
           /* QuantizedVal is the quantized value */
  7.       QuantizedVal = v // (2q)
  8.       Error := OccursCount[n][v] * Square(OriginalVal - q*QuantizedVal)
  9.       D[n][q] := D[n][q] + Error
 10.     D[n][q] := D[n][q]/N
```

```
Procedure FillLeastD

   Input:  Arrays D[0..63][1..MAXQ], R[0..63][1..MAXQ]
   Output: Arrays LeastD[0..63][0..MAXSIZE], QChoice[0..63][0..MAXSIZE]

 /* Initializations */
 For n := 0 to 63
   For s := 0 to MAXRATE
     LeastD[n][s] := INFINITY

 /* Fill row number zero */
 For q := 1 to MAXQ
 If (D[0][q] < LeastD[0][R[0][q]]) then
   LeastD[0][R[0][q]] := D[0][q]
   QChoice[0][R[0][q]] := q

 /* Main loop */
 For n := 1 to 63
   For q := 1 to MAXQ
     For s' := 0 to MAXRATE
       If (D[n][q] + LeastD[n-1][s'] < LeastD[n][s' + R[n][q]]) Then
         s := s' + R[n][q]
         LeastD[n][s] := D[n][q] + LeastD[n-1][s']
         QChoice[n][s] := q
```

```
Procedure RecoverQ

   Input:  Arrays QChoice[0..63][0..MAXSIZE], R[0..63][1..MAXQ]; Target rate s
   Output: Quantization table Q[0..63]

 For n := 63 downto 0
   Q[n] := QChoice[n][s]
   s := s - R[n][Q[n]]
```

B Proof of Theorem 1

Suppose $\mathcal{D}(n,s)$ is empty or $\min \mathcal{D}(n,s)$ is ∞. Then, for every q ($1 \leq q \leq \texttt{MAXQ}$), either $\texttt{R}[n][q] > s$, or $\texttt{LeastD}[n-1][s-\texttt{R}[n][q]] = \infty$. In either case, the rate s cannot be achieved from coefficients 0 through n, implying $\texttt{LeastD}[n][s] = \infty$.

Now assume $\mathcal{D}(n,s)$ is non-empty and that d is the minimum value in $\mathcal{D}(n,s)$, achieved by setting $Q[n]$ to q. Assume $\texttt{LeastD}[n][s] = d' < d$. Then the distortion d' must be achieved with some value, say q' for $Q[n]$. Let $d'' = d' - \texttt{D}[n][q']$. Then the distortion d'' must be achievable from coefficients 0 through $n-1$, with their rate being exactly equal to $s - \texttt{R}[n][q']$. But then, $d'' = \texttt{LeastD}[n-1][s-\texttt{R}[n][q']]$, as otherwise d' can be improved, contradicting $d' = \texttt{LeastD}[n][s]$. Hence $d' = \texttt{D}[n][q'] + \texttt{LeastD}[n-1][s-\texttt{R}[n][q']$ implying $d' \in \mathcal{D}(n,s)$. Thus, $d \leq d'$, which contradicts $d' < d$. $\square$