Optimizing DCT Quantization Tables: an Efficient Approach

Viresh Ratnakar
University of Wisconsin-Madison
Computer Science Department
Madison, WI

Miron Livny
University of Wisconsin-Madison
Computer Science Department
Madison, WI

Abstract

In this paper we describe an algorithm for constructing quantization tables with optimal quality-compression tradeoffs. The Discrete Cosine Transform (DCT) is widely used in image and video compression schemes such as JPEG and MPEG. The quality and compression ratio of DCT-based schemes depend on the table(s) used to quantize the 64 DCT coeficients. We present an algorithm to design quantization tables that optimize the quality-compression tradeoff for a certain class of quality and compression measures. Signal-to-Noise ratios (SNR and PSNR) fall within the class of allowed quality measures. Sum of entropies of quantized coeficients falls within the class of allowed compression measures and is a very close approximation of the actual bit-rate resulting from two-pass huffman coding. The algorithm can be used to achieve either a target quality or a target compression ratio, optimizing the other quantity.

1 Introduction

We are developing an environment for production-mode compression of still-image and video data, where the user can specify constraints on the desired quality and compression ratio, and the compressor produces the best results under those constraints without any human assistance. The Discrete Cosine Transform [ANR74] lies at the heart of most commonly used 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 reconstructed image. Thus, the quantization table used directly determines the quality-compression tradeoff. It is crucial that the quantization tables used in production-mode compression utilize the particular characteristics of particular images and video sequences so as to offer optimal compression-quality tradeoffs to users.

Several approaches have been tried in order to design quantization tables for particular quality or compression specifications. The most common of these is to use a default table and scale it up or down by a scalar multiplier to achieve varying quality or 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 an algorithm for optimum quantization table design that does not rely on visual or rate-distortion models and is not very expensive in terms of computation time. 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 produces an array that can be used to read off optimal quantization tables for a wide range of quality and compression specifications.

2 DCT-based compression

Let I be a $R \times W$ image with pixel values in the range $[1 \ldots M]$. The image is divided into 8×8 blocks. To each image block f, the Discrete Cosine Transform is applied to get an 8×8 block $\hat{f}$ of coefficients. An

8×8 matrix Q, called the quantization table, is then used to quantize the coefficients in $\hat{f}$ in a manner to be described shortly. Let $\hat{f}_Q$ represent the resulting 8×8 block of quantized coefficients. Then,

$$\hat{f}_Q(u,v) = \hat{f}(u,v) /\!\!/ Q(u,v), \quad 0 \le u,v \le 7.$$

Here $/\!\!/$ represents division followed by rounding to the nearest integer. That is, for any $b > 0$,

$$a /\!\!/ b = \begin{cases} \lfloor \frac{a}{b} + 0.5 \rfloor & \text{if } a \ge 0 \\ -\lfloor \frac{-a}{b} + 0.5 \rfloor & \text{if } a < 0. \end{cases}$$

The errors in reconstruction are essentially due to this quantization step. The block $\hat{f}_Q$ is then entropy-coded to give the compressed block $E(\hat{f}_Q)$. The decoder applies the Inverse Discrete Cosine Transform (IDCT) to the block $\hat{f}'$ constructed by dequantization:

$$\hat{f}'(u,v) = \hat{f}_Q(u,v) \cdot Q(u,v) \quad 0 \le u,v \le 7.$$

The block of pixels f' thus obtained is the final reconstructed block. The encoding and decoding process can be summarized as:

$$f \overset{\text{DCT}}{\longrightarrow} \hat{f} \overset{\text{Quantization}}{\longrightarrow} \hat{f}_Q \overset{\text{Entropy-coding}}{\longrightarrow} E(\hat{f}_Q),$$

$$E(\hat{f}_Q) \overset{\text{Decoding}}{\longrightarrow} \hat{f}_Q \overset{\text{Dequantization}}{\longrightarrow} \hat{f}' \overset{\text{IDCT}}{\longrightarrow} f'.$$

Let $I, \hat{I}, \hat{I}_Q, E(\hat{I}_Q), \hat{I}', I'$ represent respectively the original image, the collection of DCT coefficient blocks, the collection of quantized coefficient blocks, the compressed image, the collection of dequantized coefficient blocks, and the reconstructed image. Given a target size S for the compressed image the problem is to choose Q such that the difference between I and I' is minimized and size of $E(\hat{I}_Q)$ is no more than S. Alternately, given a maximum tolerable error Δ, the problem is to choose Q such that the difference between I and I' is no more that Δ and the size of $E(\hat{I}_Q)$ is minimized.

3 A general formulation of the problem

In this section we present a general formulation of the problem. The next section will present an algorithm for solving the general problem. Subsequently, we will show how to use the algorithm for PSNR-entropy, SNR-entropy, and weighted PSNR-entropy tradeoffs. We will also show how each of these can be modified to optimize tables to be used with a per-macroblock adaptive quantization scheme as in MPEG [MP91].

Let the 64 DCT coefficients be numbered $0 \ldots 63$, with the $(u,v)^{\text{th}}$ coefficient being numbered $8u+v$, $0 \le u,v \le 7$. The main assumption is that quality and compressed size for an image compressed using a quantization table $Q[0 \ldots 63]$ are sums of coefficient-wise qualities and sizes. Let MAXQ be the maximum allowed value for any $Q[n]$, $0 \le n \le 63$. Let Size$[0 \ldots 63][1 \ldots \text{MAXQ}]$ and Quality$[0 \ldots 63][1 \ldots \text{MAXQ}]$ be arrays with the following interpretations: The number Size$[n][q]$ is the contribution of the n^{th} DCT coefficient towards total compressed size, when $Q[n] = q$. Further, each Size$[n][q]$ is a positive integer. Similarly, Quality$[n][q]$ is the contribution of the n^{th} DCT coefficient towards total quality. The arrays Quality and Size are image-dependent and are the inputs to the optimization algorithm.

For an image compressed using quantization table Q, the total quality and compressed size, as functions of Q, are:

$$\text{Quality}(Q) = \sum_{n=0}^{63} \text{Quality}[n][Q[n]] \tag{1}$$

$$\text{Size}(Q) = \sum_{n=0}^{63} \text{Size}[n][Q[n]]. \tag{2}$$

For any Q with entries in the range $1 \ldots \text{MAXQ}$, Size(Q) is an integer in the range $0 \ldots \text{MAXSIZE}$.

Given the arrays Quality$[0 \ldots 63][1 \ldots \text{MAXQ}]$ and Size$[0 \ldots 63][1 \ldots \text{MAXQ}]$ as inputs, the problem can be stated in two ways:

1. Given a target quality P, find Q such that Quality(Q) $\geq P$ and Size(Q) is minimized.
2. Given a target size S, find Q such that Size(Q) $\leq S$ and Quality(Q) is maximized.

4 The algorithm

We use a dynamic-programming approach to solve the problem. Let BestQuality[0...63][0...MAXSIZE] be a table whose entries have the following meaning: BestQuality[n][s] is the best (highest) total quality for coefficients numbered 0 through n such that the total size (for these coefficients) is exactly s. That is, BestQuality[n][s] is the maximum value of $\sum_{k=0}^{n}$ Quality[k][$Q[k]$] subject to the constraint

$$\sum_{k=0}^{n} \text{Size}[k][Q[k]] = s.$$

The algorithm starts with each entry in BestQuality set to UNDEFINED and then fills the rows one by one. The key idea is described in Theorem 1.

Theorem 1 *For each n, $1 \leq n \leq 63$, and each s, $0 \leq s \leq MAXSIZE$, let $X(n,s)$ be the set*

$$X(n,s) = \left\{ Quality[n][q] + BestQuality[n-1][s'] \;\middle|\; \begin{array}{l} 1 \leq q \leq MAXQ, \\ s' = s - Size[n][q], \\ s' \geq 0, \\ BestQuality[n-1][s'] \neq UNDEFINED \end{array} \right\}.$$

Then,

$$BestQuality[n][s] = \begin{cases} \max X(n,s) & \text{if } X(n,s) \text{ is non-empty} \\ UNDEFINED & \text{otherwise.} \end{cases}$$

Proof: Suppose $X(n,s)$ is empty. Then clearly, the size s cannot be achieved from coefficients 0 through n. Now suppose $X(n,s)$ is non-empty and that p is the maximum value in $X(n,s)$, achieved by setting $Q[n]$ to q. Assume BestQuality[n][s] $= p' > p$. Then the quality p' must be achieved with some value, say q' for $Q[n]$. Let $p'' = p' -$ Quality[n][q']. Then the quality p'' must be achievable from coefficients 0 through $n-1$, with size exactly equal to $s -$ Size[n][q']. But then, $p'' =$ BestQuality[$n-1$][$s -$ Size[n][q']], as otherwise p' can be improved, contradicting $p' =$ BestQuality[n][s]. Hence $p' =$ Quality[n][q'] + BestQuality[$n-1$][$s -$ Size[n][q'] implying $p' \in X(n,s)$. Thus, $p \geq p'$, which contradicts $p' > p$. □

To recover the quantization table for any desired quality or size, we maintain another data structure QChoice[0...63][0...MAXSIZE]. QChoice[n][s] stores the value q that gave the entry in BestQuality[n][s].

We now present the algorithm in pseudocode. For ease of presentation, we assume that the constant UNDEFINED behaves as minus infinity ($-\infty$) in comparison tests.

```
Algorithm FillBestQuality

  Input:  Arrays Quality[0..63][1..MAXQ], Size[0..63][1..MAXQ]
  Output: Arrays BestQuality[0..63][0..MAXSIZE], QChoice[0..63][0..MAXSIZE]
/* Initializations */
 1. For n := 0 to 63
 2.   For s := 0 to MAXSIZE
 3.     BestQuality[n][s] := UNDEFINED

/* Fill row number zero */
 4. For q := 1 to MAXQ
 5. If (Quality[0][q] > BestQuality[0][Size[0][q]]) then
 6.   BestQuality[0][Size[0][q]] := Quality[0][q]
 7.   QChoice[0][Size[0][q]] := q
```

```
/* Main loop */
 8. For n := 1 to 63
 9.   For q := 1 to MAXQ
10.     For s' := 0 to MAXQUAL
11.       If (Quality[n][q] + BestQuality[n-1][s'] > BestQuality[n][s' + Size[n][q]]) Then
12.         s := s' + Size[n][q]
13.         BestQuality[n][s] := Quality[n][q] + BestQuality[n-1][s']
14.         QChoice[n][s] := q
```

Now, if a total quality requirement P is to be met, it's straightforward to find the least s such that BestQuality[63][s] $\geq P$. Similarly, if a size requirement S is to be met, it's easy to find s such that $s \leq S$ and BestQuality[63][s] is the maximum 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, the following procedure is used:

```
Procedure RecoverQ

  Input:  Arrays BestQuality[0..63][0..MAXSIZE], QChoice[0..63][0..MAXSIZE]; Target size s
  Output: Quantization table Q[0..63]

 1. For n := 63 downto 0
 2.   Q[n] := QChoice[n][s]
 3.   s := s - Size[n][Q[n]]
```

4.1 Complexity

Algorithm FillBestQuality clearly runs in time less than a constant times $64 \times \text{MAXQ} \times \text{MAXSIZE}$. In any practical implementation, this can be substantially reduced. The loop range in line 9 can be made 1 to MAXQ(n) where MAXQ(n) is the minimum of MAXQ and the least value of $Q[n]$ that will make the n^{th} coefficient zero everywhere in the image. The loop range in line 10 for s' can be made 1 to the last entry in the $(n-1)^{\text{th}}$ row which is not marked UNDEFINED. Further, if only one given target quality or target size is to be met, then the loops can be pruned to exclude cases which will clearly be outside the given specifications. Also, note that only two rows of the table BestQuality need to be maintained at any point: the current row and the previous row.

The key idea is entirely symmetric, in the sense that we can maintain an array BestSize instead of BestQuality and use an algorithm FillBestSize analogous to FillBestQuality. In this case, quality will need to be integral. The choice of the algorithm to be used depends on various factors such as the range of values spanned by quality and size, and the errors incurred by quantizing them to integers.

The algorithm has another interesting feature: the final results are independent of the order in which the coefficients are considered. This implies that it can be readily parallelized. The 64 rows can be pairwise combined, then the 32 "composite" rows can be pairwise combined, and so on.

5 PSNR-entropy formulation

In this section, we present the details on using the algorithm for optimizing the PSNR-entropy tradeoffs for a particular image (or video sequence).

For an image I compressed using quantization table Q, we use PSNR as the quality measure for the reconstruction I'. Recall that pixel values are in the range $0 \ldots M$.

$$\text{PSNR}(Q) = 10 \log_{10}(\frac{M^2}{\text{MSE}_Q}),$$

where MSE_Q is the mean squared error between the pixel values in I and I'. For any Q, MSE_Q is readily obtained using some fundamental properties of DCT (see [RY90], for example), as follows. Recall the notation in Section 2. For any image block f, we have,

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

Using linearity of DCT,

$$\mathrm{DCT}(f - f') = \hat{f} - \hat{f}'.$$

Further, since DCT preserves L^2 norms,

$$\begin{aligned} \sum_{0 \le i,j \le 7} (f(i,j) - f'(i,j))^2 &= \sum_{0 \le u,v \le 7} (\hat{f}(u,v) - \hat{f}'(u,v))^2 \\ &= \sum_{0 \le u,v \le 7} (\hat{f}(u,v) - (Q(u,v) \cdot (\hat{f}(u,v) /\!\!/ Q(u,v))))^2 \end{aligned}$$

Hence, MSE_Q can be expressed as a coefficient-wise sum of quantization errors.

To estimate the size of the compressed image, we use the coefficient-wise sum of entropies. This provides a very good estimate for bits-per-pixel used (BPP), as reported in [RFVK94].

$$\mathrm{BPP}(Q) = (1/64) \sum_{n=0}^{63} H_Q(n),$$

Where $H_Q(n)$ is the measured entropy of the values the n^{th} coefficient takes after quantization by $Q[n]$. If $p_n(q,v)$ is the fraction of blocks where the n^{th} coefficient has the value v after quantization by q, then

$$H_Q(n) = - \sum_{\text{all } v} p_n(Q[n], v) \, \log_2(p_n(Q[n], v)).$$

In terms of the general formulation of the previous section, we let $\mathrm{Quality}(Q)$ be the negative of the mean sqaured error incurred using quantization table Q. That is,

$$\mathrm{Quality}(Q) = -\mathrm{MSE}_Q.$$

For $\mathrm{Size}(Q)$, we quantize $\mathrm{BPP}(Q)$ to integer values by multiplying with a large constant BPPSCALE and rounding to the nearest integer.

$$\mathrm{Size}(Q) = \mathrm{Round}(\mathrm{BPPSCALE} \cdot \mathrm{BPP}(Q)).$$

We will see shortly how to choose BPPSCALE to meet any desired level of accuracy.

To construct the arrays $\mathrm{Quality}[0 \ldots 63][1 \ldots \mathrm{MAXQ}]$ and $\mathrm{Size}[0 \ldots 63][1 \ldots \mathrm{MAXQ}]$ such that $\mathrm{Quality}(Q)$ and $\mathrm{Size}(Q)$ can be expressed as coefficient-wise sums as in equations 1 and 2, we use a preliminary pass through the image to gather the DCT statistics, as follows. The procedure GatherStats fills an array $\mathrm{OccursCount}[0 \ldots 63][-2\mathrm{VMAX} \ldots 2\mathrm{VMAX}]$. The constant VMAX is the maximum absolute value any DCT coefficient can take (for 1-byte samples, $M = 255$ and $\mathrm{VMAX} = 2048$). The value $\mathrm{OccursCount}[n][v]$ at the end is the number of blocks where the n^{th} DCT coefficient c_n is such that $\lfloor 2c_n \rfloor = v$ (for $c_n \ge 0$) or $-\lfloor -2c_n \rfloor = v$ (for $c_n < 0$). Thus, OccursCount is the histogram of DCT coefficients in steps of 0.5. It can be shown that for any real c and integer $q \ge 1$, $((c /\!\!/ 1) /\!\!/ q) = (x /\!\!/ 2q)$ where

$$x = \begin{cases} \lfloor 2c \rfloor & \text{if } c \ge 0 \\ -\lfloor -2c \rfloor & \text{if } c < 0. \end{cases}$$

The function TruncateTowardsZero used in Procedure GatherStats is defined as

$$\mathrm{TruncateTowardsZero}(x) = \begin{cases} \lfloor x \rfloor & \text{if } x \ge 0 \\ -\lfloor -x \rfloor & \text{if } x < 0. \end{cases}$$

```
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 := TruncateTowardsZero(2*g[n])
 6.     OccursCount[n][v]++
```

The array OccursCount can then be used to find, for any coefficient n, any quantization table entry q, and any integer value v, the number of times the n^{th} coefficient quantized by q takes the value v. This information is used to fill the arrays Quality and Size using the procedures FillQuality and FillSize, as follows.

```
Procedure FillQuality

  Input:  Array OccursCount[0..63][-2VMAX..2VMAX]
  Output: Array Quality[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.     Quality[n][q] := 0
 5.     For v := -2VMAX to 2VMAX
 6.       OriginalVal = v/2.0 + ((v < 0) ? -0.25 : 0.25)
 7.       QuantizedVal = v // (2q)
 8.       error := OccursCount[n][v] * Square(OriginalVal - QuantizedVal)
 9.       Quality[n][q] := Quality[n][q] - error
10.     Quality[n][q] := Quality[n][q]/N

Procedure FillSize

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

 1. N := 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
 6.       /* QuantizedVal is the quantized value */
 7.       count := 0 /* count is the # of times the value QuantizedVal occurs */
 8.       For each v such that v // (2q) == QuantizedVal
 9.         count := count + OccursCount[n][v]
10.       prob := count/N
11.       If (prob > 0) then
12.         entropy := entropy - (prob * Log2(prob))
13.     Size[n][q] := (BPPSCALE * entropy / 64) // 1
```

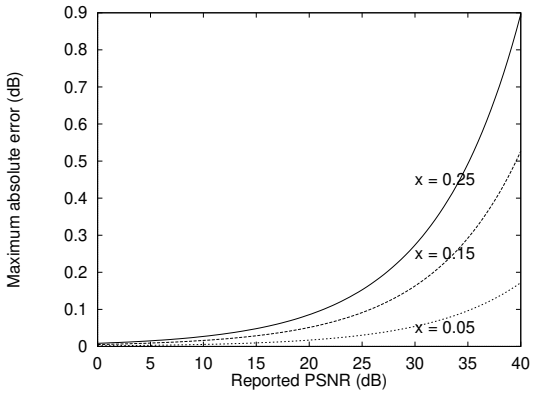

Figure 1: Accuracy of reported PSNR

The loop ranges in these procedures can also be reduced, depending upon image characteristics.

5.1 Error analysis

Since OccursCount is the histogram of discretized DCT coefficients, Quality(Q) will not be exactly equal to $-\text{MSE}_Q$. This is because the quantity

$$(c - (c /\!\!/ q) \cdot q)^2$$

is being approximated by

$$((c + \delta) - (c /\!\!/ q) \cdot q)^2,$$

where $-0.25 \leq \delta \leq +0.25$. Let I_δ be the image for which the DCT coefficients are the same as those for I, except that they are discretized to steps of 0.5 as in Procedure GatherStats. Hence the PSNR obtained will be the PSNR with respect to I_δ, rather than than the original image I. Let $\hat{I}_\delta$ represent the DCT coefficients for I_δ. Let P_δ be the PSNR reported by the algorithm. Let the mean squared error between $\hat{I}'$ and $\hat{I}_\delta$ be denoted by Δ_δ. Let the actual mean squared error (between $\hat{I}'$ and $\hat{I}$) be denoted by Δ. The error between any coefficient in $\hat{I}_\delta$ and the corresponding coefficient in $\hat{I}$ is at most 0.25. Hence the triangle inequality implies,

$$\begin{aligned} \sqrt{\Delta_\delta} - 0.25 \quad &\leq \sqrt{\Delta} \leq \quad \sqrt{\Delta_\delta} + 0.25 \\ \Rightarrow 1 - \frac{0.25}{\sqrt{\Delta_\delta}} \quad &\leq \sqrt{\tfrac{\Delta}{\Delta_\delta}} \leq \quad 1 + \frac{0.25}{\sqrt{\Delta_\delta}}. \end{aligned}$$

Since

$$P_\delta = 10 \log_{10} \frac{255^2}{\Delta_\delta},$$

the actual PSNR, P, is bounded as follows:

$$P_\delta - 20 \log_{10}(1 + \alpha(P_\delta)) \leq P \leq P_\delta - 20 \log_{10}(1 - \alpha(P_\delta)), \tag{3}$$

where $\alpha(P_\delta) = \frac{10^{P_\delta/20}}{1020}$. If lower errors are desired, OccursCount must be stored with finer accuracy. If the error in estimating any coefficient value is at most x, then the error bounds on P_δ can be obtained using 3 with $\alpha(P_\delta) = \frac{x 10^{P_\delta/20}}{255}$. Figure 5.1 shows the error bound versus P_δ for various values of x. For $x = 0.25$, the error is at most 0.9 dB at $P_\delta = 40$dB, and at most 0.3 dB at $P_\delta = 30$dB.

The total error in Size(Q) is at most

$$64 \cdot 0.5/\text{BPPSCALE}.$$

For example, if we set BPPSCALE to 10000, then the error is at most 0.0032 bits per pixel. This gives a row size of 10000 in FillBestQuality.

5.2 Modifications

Instead of PSNR, the final table BestQuality can be used to pick a table for any desired SNR.

$$\text{SNR} = 10 \log_{10}(\frac{\text{Mean squared pixel value}}{\text{Mean squared error}}).$$

Similarly, it is straightforward to use weighted mean squared error instead of mean squared error, by assigning different weights to errors in different frequencies.

For better visual quality, it is sometimes useful to to do adaptive quantization which gives more bits for encoding regions in the image that are perceptually more significant. This is done in MPEG by scaling the quantization table up or down on a per-macroblock basis [MP91]. Thus, for any block f, the quantization table used is $Q \cdot \text{qscale}_f$, where Q is a nominal quantization table and qscale_f is a factor that depends upon the macroblock containing f. The value of qscale_f is typically chosen based upon characteristics such as texture, total energy, presence of edges, etc. However, qscale_f does not depend upon Q. Hence, while gathering statistics (procedure GatherStats) qscale_f can be determined for each block. The entry OccursCount$[n][v]$ can be filled by setting v to be the actual value of the n^{th} coefficient divided by qscale_f for the block under consideration. Then, procedure FillBestQuality will optimize Q to give the best quality-compression tradeoff for the adaptive quantization scheme.

6 Performance results

We have implemented the algorithm in C on various platforms. The total running time for Algorithm FillBestQuality (including procedures FillSize and FillQuality) on an IBM POWERstation 370 is about 47 seconds. The time for gathering statistics is that required to do DCT on the entire image or video sequence.

FILL figures, tables

7 Conclusion

FILL

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.

[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.