Computation-Distortion Characteristics of JPEG Encoding and Decoding

Vivek K Goyal[1] Martin Vetterli[1,2]

[1]Dept. of Electrical Engineering and Computer Sciences
University of California, Berkeley

[2]Laboratoire de Communications Audiovisuelles
École Polytechnique Fédérale de Lausanne, Switzerland

E-mail: v.goyal@ieee.org, Martin.Vetterli@de.epfl.ch

Abstract

A distortion-computation function $D(C)$ *is defined as the minimum expected distortion in computing some quantity—using an algorithm from a predefined set of algorithms—while using no more than* C *computational units. When the computational problem is to encode at rate* R*, this gives slices of a* computation-rate-distortion surface. *This framework is used in the analysis of a family of JPEG coders that use output-pruned DCT calculations in place of some full DCT calculations. For encoding the Lena image at 0.5 bits/pixel, this yields a 30% reduction in complexity while lowering the PSNR by only 0.4 dB. The decoding complexity can be similarly reduced.*

1 Introduction

It is clear to all source coding practictioners that computational complexity is of substantial importance, and there is no question that a lot of effort has gone into optimizing certain calculations which are common in source coding, *e.g.* matrix multiplications, filtering operations, and discrete cosine transforms. But, at least amongst theoreticians, there seems to be a bit of a gap between the design of coding algorithms and the computational optimization of them: One occassionally finds discussions of computational trade-offs in papers on coding algorithms, but rarely finds precise numerical comparisons or justifications.

In a recent paper [6], we introduced a flexible framework for systematically studying the trade-off between computational complexity and coding performance. The value of the framework itself is to provide a common vocabulary; it does not intrinsically aid in the analysis. The bulk of this paper is devoted to one particular application of this framework: JPEG encoding and decoding. Through the analysis of a set of "simplified" encoding and decoding algorithms, a precise characterization of an achievable set of rate-PSNR-complexity triples is found. (By "simplified" we mean that certain calculations which would be performed in a standard JPEG encoder/decoder are omitted even though this will generally impair the rate-distortion performance.) In particular, we find that for a fixed rate, the computation vs. distortion trade-off exhibits a diminishing returns characteristic, where the computation can be reduced somewhat with negligible effect on image quality.

This paper reviews the framework from [6] in Section 2 and then applies this framework to JPEG encoding and decoding in Section 3. Another application relevant to image coding, namely to entropy pruned tree-structured vector quantization [3], is briefly outlined in Section 4.

2 Computation-Rate-Distortion Framework

Let $\mathcal{P}$ be a set of computational problems which are posed according to some underlying probability distribution and let ρ be a distortion measure on approximate solutions to problems in $\mathcal{P}$. Suppose also there is a computational cost function on algorithms for (approximately) solving $P \in \mathcal{P}$, $c : \mathcal{A} \times \mathcal{P} \to \mathbb{R}^+$, where $\mathcal{A}$ is a set of such algorithms. Then define the *distortion-computation function* of algorithms $\mathcal{A}$ for problems $\mathcal{P}$ by

$$D(C) = \inf_{\{A \in \mathcal{A} : Ec(A,P) \leq C\}} E\rho(P, A(P)).$$

In the context of source coding, we can specialize the definition. Consider the problem to be finding a variable length approximate representation of a source with expected length bounded above by R. Denote the source and the reproduction by x and $\hat{x}$, respectively. Then, define the *distortion-computation function at rate* R by

$$D_R(C) = \inf_{\{A \in \mathcal{A} : Ec(A,x) \leq C, \ell(\hat{x}) \leq R\}} E\rho(x, \hat{x}),$$

1058-6393/98 $10.00

where $\ell(\hat{x})$ is the length of the representation of $\hat{x}$ in bits. Notice that in contrast to the definition of a rate distortion function [4], we do not use the mutual information between x and $\hat{x}$. Doing so would implicitly assume that the entropy coding of $\hat{x}$ is ideal; instead, we would like to remain open to the possibility that the entropy coding is included in the computational cost. Varying the parameter R yields a *computation-rate-distortion surface*.

This framework was introduced in [6]. That paper provided a detailed comparison of the use of the Karhunen-Loève Transform (KLT) and the Discrete Cosine Transform (DCT) for transform coding of a Gauss-Markov source. In this same context, Gormish and Gill [5] made earlier mention of the concept of a computation-rate-distortion surface, though in an operational sense.

A few properties are obvious. $D(C)$ must be nonincreasing and $D_R(C)$ must be nonincreasing with respect to both R and C. If the set of algorithms produces a discrete set of complexities, as is the case in Section 3, then $D(C)$ may be only piecewise continuous. If we allow time- or probabilistic-multiplexing, then operational $D(C)$ and $D_R(C)$ can always be made convex.

3 Application to JPEG Coding

3.1 Analysis

Of the major steps in JPEG coding (DCT, scalar quantization, zigzag scanning, runlength coding, and entropy coding), only the computation of the DCT seems to be computation scalable; thus, we focus on this step. In order to utilize the framework of the previous section, we must explicitly define $\mathcal{P}$, $\mathcal{A}$, ρ, and c. We do this as follows:

$\mathcal{P}$ = JPEG-compatible encoding at R bits/pixel

$\mathcal{A}$ = Approximate DCT followed by standard JPEG quantization and entropy coding. The approximate DCT is described below.

ρ = MSE

c = Number of multiplications per block in the approximate DCT computation

The set of approximate DCT algorithms that we consider are algorithms that compute a subset of the DCT coefficients and assume that the remaining coefficients are zero. The image blocks in JPEG are 8×8, so there are 2^{64} approximate algorithms. Because of the frequency domain characteristics of natural images, it is clear that a small number of the possible algorithms are if interest. In this investigation, we will limit the sets of calculated coefficients to be a triangle or rectangle of low frequency coefficients, as shown in Figure 1. The use of the rectangular regions is motivated by the trade-off between the number of coefficients calculated and the computational complexity. Triangular regions are motivated by the zigzag scanning of AC coefficients.

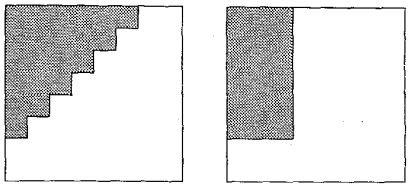

Figure 1. The DCT coefficients computed form either a triangle or rectangle. The lowest frequencies are in the upper left.

k	multiplications	additions
1	0	7
2	7	20
3	10	25
4	11	27
5	12	29
6	12	29
7	12	29
8	12	29

Table 1. Number of multiplications and additions to compute the first k coefficents of a length-8 DCT using algorithms designed through output pruning.

The DCT calculation itself is done separably using *output pruned* decimation-in-frequency 1-D DCT algorithms, by which we mean decimation in frequency length-8 DCT algorithms [11] which are simplified because only a subset of the eight coefficients are desired. Figure 2 shows an output pruned DCT where only the first two coefficients ($X[0]$ and $X[1]$) are desired. Because of the shapes of the regions considered (see Figure 1), we always require the k lowest frequency DCT coefficients, $1 \leq k \leq 8$. The computational complexities for these output-pruned 1-D DCTs are given in Table 1.

From Table 1 it might seem that significant computational savings are achieved only if, say, two of eight DCT coefficients are calculated. A large fraction of the savings comes from the 2-D separable nature of the computation. For example, computing the 4×4 block of lowest frequency coefficients requires eight horizontal DCTs from which the first four coefficients are desired (8 times 11 multiplications) and *four* vertical DCTs from which the first four coefficients are desired (4 times 11 multiplications) for a total of 132 multiplications. This is roughly two-thirds of the 192 multi-

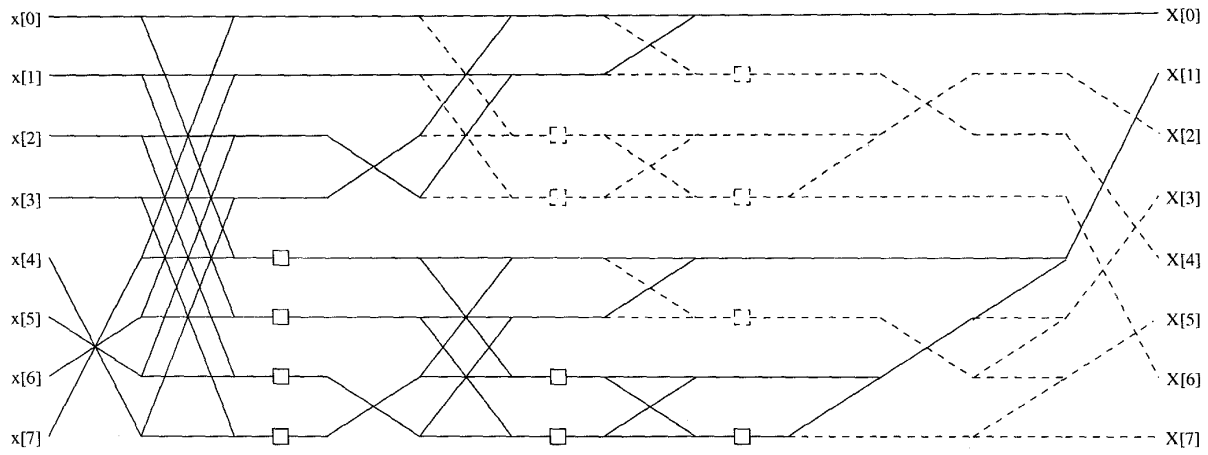

Figure 2. Output pruned signal flow graph for computing the first two coefficients of a length-8 DCT. The full signal flow graph is from [11, p. 61] and represents a decimation-in-frequency algorithm. (Boxes represent multiplications other than by ± 1 and subtractions are not distinguished from additions.) The dotted curves represent calculations that can be eliminated because we desire only $X[0]$ and $X[1]$. The computational complexity is reduced from 12 multiplications and 29 additions for the full calculation to 7 multiplications and 20 additions.

plications for a full 8×8 DCT. The situation is very similar for decoding. If we assume that the DCT coefficients outside of a certain region are zero, we can use input pruned 1-D inverse DCT algorithms. The signal flow graph would be like a left-right flipped version of Figure 2.

3.2 Results

A large number of computation-rate-distortion points were determined for the standard *Lena* image using approximate DCT calculations as we have described. The remaining components (quantization, zigzag scanning, and entropy coding) were implemented as in an Independent JPEG Group encoder[1] with "quality factors" 5, 10, ..., 95, and 99. The points with rates less than 2 bits per pixel which are presumably on the computation-rate-distortion surface are shown in Figure 3, where the connected points are at the same complexity.[2] Figure 4 shows the computation-distortion for several rates. We find that for low- to moderate-rate coding, the distortion as a function of computation becomes very flat. For example, at 0.5 bits/pixel, one can have a 19% complexity reduction while lowering the PSNR by less than 0.1 dB, or have a 30% complexity reduction while lowering the PSNR by less than 0.4 dB.

[1] To download software, follow links from http://www.ijg.org/.

[2] We use "presumably" because we did not try *every* algorithm in $\mathcal{A}$.

3.3 Comments

Not computing certain coefficients and setting them to zero seems a very rudimentary way to produce an approximate DCT algorithm, but it works reasonably well for this application. Because of the runlength and entropy coding used in JPEG, even when a high frequency DCT coefficient has a nonzero quantized value, coding that coefficient (as opposed to rounding it to zero) may not be wise in a rate-distortion sense [10]. The approximate DCT considered here forces longer runs of zeros and hence gets good coding efficiency.

Output pruning is not the optimal way to produce the desired 1-D DCT algorithms, but was done for conceptual transparency and so that the set of algorithms $\mathcal{A}$ could be precisely defined. Lengwehasatit [7] has provided the operation counts for hand-optimized algorithms which assume integer valued inputs and use some bit shifting. These operation counts are given in Table 2 and yield the $D(C)$ curves in Figure 5. The complexity reductions in Figure 5 are even more dramatic than those in Figure 4. At 0.5 bits/pixel, 0.1 dB and 0.4 dB PSNR losses occur with 38% and 46% reductions in complexity, respectively.

One approach to improving on the results presented here is to use a *variable complexity algorithm* (VCA). The algorithms we have discussed process each block identically, regardless of how many DCT coefficients of the block are zero. For decoding, blocks with many zero coefficients are

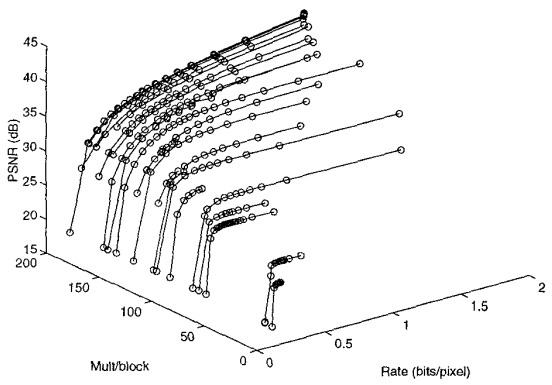

Figure 3. Operational $C - R - D$ for JPEG encoding of Lena with approximate DCT algorithms designed through output pruning.

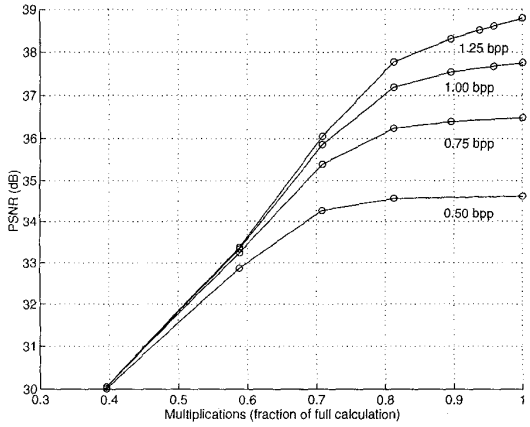

Figure 4. Operational $D(C)$ at various bit rates for JPEG encoding of Lena with approximate DCT algorithms designed through output pruning.

k	multiplications	additions
1	0	7
2	4	17
3	6	21
4	8	25
5	8	26
6	9	27
7	10	28
8	11	29

Table 2. Operation counts to compute the first k coefficents of a length-8 DCT [7].

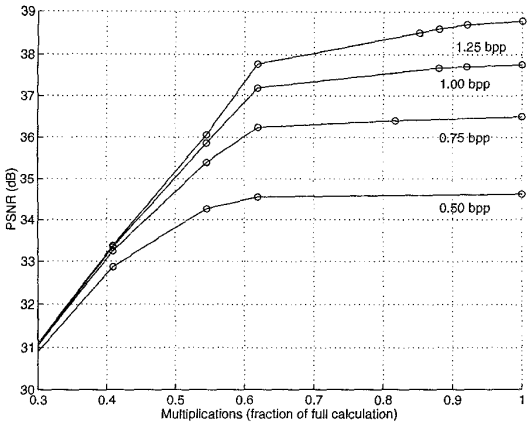

Figure 5. Operational $D(C)$ at various bit rates for JPEG encoding of Lena with computation counts from Table 2.

clearly easiest to handle, assuming that they can be identified efficiently. Lengwehasatit and Ortega [8, 9] have developed a decoder which classifies blocks based on their patterns of zero and non-zero coefficients and uses inverse DCT algorithms optimized for each class. The definitions of the classes themselves have been computationally optimized, including the cost of classifying. The VCA work has a distinct advantage over the present work: the complexity is reduced with no degradation of rate-distortion performance. On the other hand, the degree of complexity reduction with this approach is limited.

4 Application to Entropy Pruned Tree-Structured Vector Quantization

An eventual goal of this research is to provide meaningful comparisons between vector quantization (VQ) based coding schemes and transform coding schemes. A requisite step would be to characterize the computation-distortion behavior of VQ coding. For VQ with unconstrained codebook design and full-search encoding, the rate determines the codebook size and hence determines the computational complexity of encoding. Therefore to have variable computational load requires varying the vector dimension.[3] This is not pursued here.

In contrast to the case of full-search VQ, the complexity of entropy pruned tree-structured VQ (EPTSVQ) [3] is not determinable from the output rate. Thus we can attempt to

[3]The situation is slightly more complicated with entropy coding, but the fact remains that for a fixed output entropy it is difficult to vary the size of the codebook in a meaningful way.

optimize simultaneously for rate, distortion, and computation.

In tree-structured VQ (TSVQ) [2], a binary tree is constructed with a codeword at each node. In the encoding process, one starts at the root of the tree and iteratively traverses the branch to the child node whose codeword is closest to the source vector. Coding terminates when a leaf node is reached. In the simplest form of TSVQ, the route from root to leaf is the channel codeword (say, each left child selected gives a zero and each right child gives a one). The rate can be lowered by using an entropy code on the leaf nodes. This is called entropy coded TSVQ. EPTSVQ is a design method for entropy coded TSVQ. The idea is to start with a deep TSVQ tree and prune it to minimize $J = R + \lambda D$, where R is the entropy coded rate, D is the distortion, and λ controls the trade-off between R and D. (Optimality is within the set of all subtrees of the original tree.) The pruning uses the greedy algorithm of Breiman, Freidman, Olshen, and Stone (BFOS) [1] which in general is not optimal, but is optimal in this case because the objective functions are monotonic, affine tree functionals [3].

Assuming a pair of distance determinations and a comparison takes one unit of computation, the average computational complexity of TSVQ encoding is the weighted average of the depths of the leaf nodes. Because the average tree depth is a monotonic, affine tree functional, $J' = R + \lambda D + \mu C$ can also be minimized with the BFOS algorithm. Thus, we have an efficient method to find the lower convex hull of computation-rate-distortion points in entropy coded TSVQ. However, because of the close coupling between rate and computation, the optimal pruning does not depend much on the relative weighting of rate and computation. Experiments for image coding at 1 bit per pixel show that computation can be reduced by about 5% while incurring an increase in distortion of about 5%.

5 Concluding Comments

Traditionally, image coding techniques have been compared almost exclusively on the basis of their rate vs. distortion performances, with consideration of computational complexity, if any, reduced to binary determinations: "too complex" or "not too complex." Through a simple example, this paper provides a glimpse at what happens when a measure of computational complexity is thrown into the mix. As a philosophical ideal, comparisons between algorithms should be done by comparing their performances with a fixed computational budget. Since many coding methods are at least partially computation-scalable this is a sensible objective.

We have used a very precise framework to define the optimal trade-off between complexity and distortion. However, it should be noted that the ability to draw precise conclusions is dependent on the set of algorithms $\mathcal{A}$ and the computational complexity metric c. In Section 3, $\mathcal{A}$ is a relatively small discrete set, so by using an (effectively) exhaustive search, we were able to (essentially) exactly characterize $D_R(C)$. If we were to enlarge $\mathcal{A}$, then we would have only an upper bound for $D_R(C)$.

References

[1] L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone. *Classification and Regression Trees. The Wadsworth Statistics/Probability Series*. Wadsworth, Belmont, CA, 1984.

[2] A. Buzo, A. H. Gray, Jr., R. M. Gray, and J. D. Markel. Speech coding based upon vector quantization. *IEEE Trans. Acoust. Speech Signal Proc.*, ASSP-28:562–574, Oct. 1980.

[3] P. A. Chou, T. Lookabaugh, and R. M. Gray. Optimal pruning with applications to tree-structured source coding and modeling. *IEEE Trans. Info. Theory*, IT-35(2):299–315, Mar. 1989.

[4] T. M. Cover and J. A. Thomas. *Elements of Information Theory*. John Wiley & Sons, New York, 1991.

[5] M. J. Gormish and J. T. Gill. Computation-rate-distortion in transform coders for image compression. In *Proc. SPIE Conf. Image and Video Proc.*, volume 1903, pages 146–152, 1993.

[6] V. K. Goyal and M. Vetterli. Computation-distortion characteristics of block transform coding. In *Proc. IEEE Int. Conf. Acoust., Speech, and Signal Proc.*, pages 2729–2732, Munich, Germany, Apr. 1997.

[7] K. Lengwehasatit. Personal communication. October 1997.

[8] K. Lengwehasatit and A. Ortega. DCT computation with minimal average number of operations. In *Proc. SPIE Conf. on Vis. Commun. and Image Proc.*, volume 3024, San Jose, California, Feb. 1997.

[9] K. Lengwehasatit and A. Ortega. Distortion/decoding time tradeoffs in software DCT-based image coding. In *Proc. IEEE Int. Conf. Acoust., Speech, and Signal Proc.*, volume 4, pages 2725–2728, Munich, Germany, Apr. 1997.

[10] K. Ramchandran and M. Vetterli. Rate-distortion optimal fast thresholding with complete JPEG/MPEG decoder compatibility. *IEEE Trans. Image Proc.*, IP-3(5):700–704, Sept. 1994.

[11] K. R. Rao and P. Yip. *Discrete Cosine Transform: Algorithms, Advantages, Applications*. Academic Press, 1990.