COMPUTATION-DISTORTION CHARACTERISTICS OF BLOCK TRANSFORM CODING

Vivek K Goyal *Martin Vetterli*
Dept. of Electrical Engineering and Computer Sciences
University of California, Berkeley
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 while using no more than C computational units. In a communication framework, where the computational problem is to determine a representation that can be transmitted with expected rate not exceeding R, this gives slices of a *rate-distortion-computation surface*. The convexity of distortion-computation functions and rate-distortion-computation surfaces is asserted. Transform coding is studied as a particular instance of this theory. Explicit comparisons between the efficacies of the Karhunen-Loève Transform and the Discrete Cosine Transform for coding of a Gauss-Markov source are given. Results are also given on joint optimization of the block length and the computational precision.

1. INTRODUCTION

Computational complexity is by no means a revolutionary concern—in general or in source coding. All coding standards, *e.g.* JPEG and MPEG, are compromises between computational complexity and performance. It seems, however, that consideration of complexity in source coding is often reduced to a binary determination: "too complex" or "not too complex." This is not satisfactory; 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 (for example by changing the block size in block transform coding or vector quantization), this is a sensible objective.

In this paper we propose a systematic way to study the relationship between computational complexity and coding performance. Our definitions create a framework analogous and complementary to rate-distortion theory. The similarity to rate-distortion theory is intended to make the definitions seem familiar to those working in information theory and communications.

This theory is applied to assess the efficacy of the Karhunen-Loève Transform (KLT) and the Discrete Cosine Transform (DCT) for block transform coding of a Gauss-Markov source. Stochastic simulations are avoided through the use of reasonable approximations and explicit calculations.

2. DEFINITIONS AND BASIC PROPERTIES

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* for $\mathcal{P}$ by

$$D(C) = \min_{\{A \in \mathcal{A} : Ec(A,P) \le C\}} E\rho(P, A(P)). \tag{1}$$

In the context of source coding, we can specialize the definition. Consider the problem to be finding an approximate representation of a source with expected rate bounded above by R. Denote the source and the reproduction by x and $\hat{x}$, respectively. Then, taking R as a parameter,

$$D_R(C) = \min_{\{A \in \mathcal{A} : Ec(A,x) \le C, \ell(\hat{x}) \le R\}} E\rho(x, \hat{x}), \tag{2}$$

where $\ell(\hat{x})$ is the entropy rate of $\hat{x}$. Since

$$\lim_{C \to \infty} D_R(C) = D(R),$$

a point on the distortion-rate function, (2) describes a 3-D surface with the property that in the limit $C \to \infty$ one obtains the distortion-rate function.

Assuming the set of algorithms allows multiplexing between operating points, the following basic theorems can be proven by using probabilistic multiplexing arguments:[1]

Theorem 1: $D(C)$ is convex.

Theorem 2: The *D-R-C* surface is convex, *i.e.* given $C_1 \le C_2$ and $R_1 \le R_2$,

$$D_{R_\theta}(\theta C_1 + (1-\theta)C_2) \le \theta D_{R_1}(C_1) + (1-\theta) D_{R_2}(C_2),$$

for all $\theta \in [0,1]$, where $R_\theta = \theta R_1 + (1-\theta)R_2$.

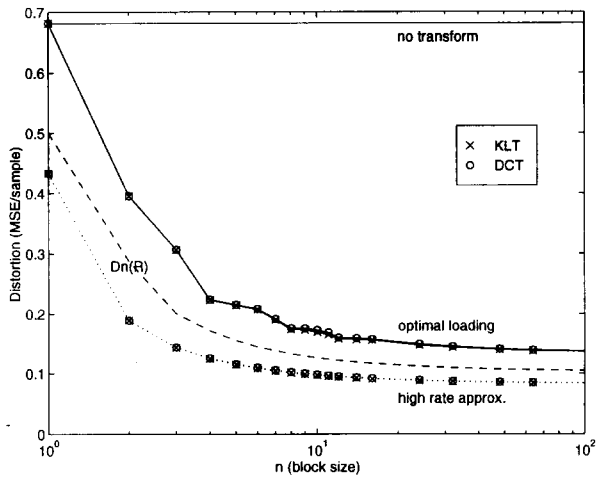
As in rate-distortion theory, a system that is described by parameters chosen from discrete sets will not necessarily have a convex *operational* distortion-computation function (or rate-distortion-computation surface). However, as above, convexity can be assured by multiplexing.

3. TRANSFORM CODING

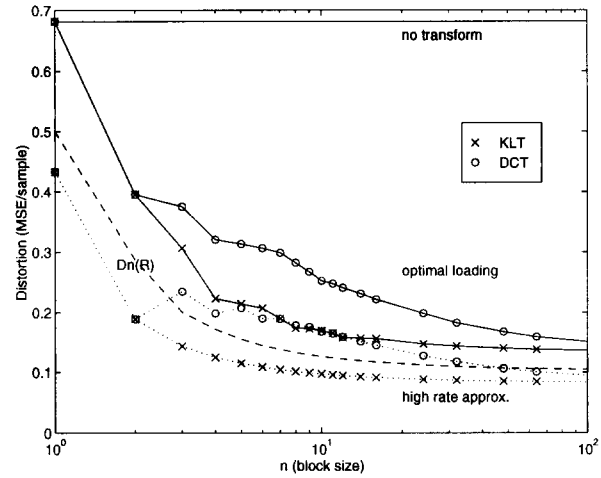
The usual justification of using the Discrete Cosine Transform (DCT) rather than the Karhunen-Loève Transform (KLT) in transform coding is that the DCT is a fixed transform which can be implemented with a fast algorithm. Thus, even if the KLT would give better rate-distortion performance than the DCT for a fixed block size n, the DCT may be preferable because it makes larger values of n feasible. The distortion-computation function approach allows a precise characterization of this phenomenon.

As an example, we will consider the transform coding (with scalar quantization) of a Gaussian first-order autoregressive source X with correlation coefficient α, *i.e.* a source with autocorrelation sequence $r_X(m) = \alpha^{|m|}$. Distortion is

[1] The proofs require that, at a minimum, a single Bernoulli random variable can be generated with a finite computational cost.



(a) Source correlation coefficient $\alpha = 0.9$



(b) Source correlation coefficient $\alpha = -0.9$

Figure 1. *Operational $D(n)$ for DCT and KLT coding of an AR(1) source at rate 0.5 bits/sample. A high resolution approximation, performance with no transform, and the theoretical bound are also shown. Note that the block size is given on a logarithmic scale.*

measured by MSE per sample. We will first consider algorithms based on an exactly computed transform followed by quantization, and we will measure complexity by the number of general multiplications.[2] This case allows for many precise statements but, as will be discussed, is of limited practical consequence. We then consider the more practical case where a transform is computed with finite precision and the result is then quantized.

3.1. Coding with exact computations

3.1.1. Computing distortion

As a preliminary to finding operational distortion-computation functions for DCT and KLT coding, we first study the $D(R)$ performance of these methods as the block size n is varied. In this step we must make certain assumptions about the coding process, namely in relation to the bit allocation and design of the scalar quantizers, but we need not make any assumptions about the computational model. For comparison we also exhibit the performance attained without any transform and the optimal performance attainable for any method that forms n-tuples from the source and codes them independently.[3]

Denote the KLT for block size n by T_n, *i.e.* $T_n R_X T_n^T = \Lambda$, where Λ is a diagonal matrix with nonincreasing entries. Let U denote a DCT matrix given elementwise by[4]

$$u_{ij} = \begin{cases} \sqrt{\frac{2}{n}} \cos(\frac{\pi}{n}(i-1)(j-\frac{1}{2})) & i = 2, 3, \ldots, n \\ \sqrt{\frac{1}{n}} \cos(\frac{\pi}{n}(i-1)(j-\frac{1}{2})) & i = 1 \end{cases}$$

Since the quantization is scalar, the performance depends only on the transformed component variances, which are given (without regard to ordering) by $(\lambda_1, \lambda_2, \ldots, \lambda_n) = \mathrm{diag}(T_n R_X T_n^T)$ and $(\mu_1, \mu_2, \ldots, \mu_n) = \mathrm{diag}(U R_X U^T)$ for KLT and DCT coding, respectively.

For large n we can make the approximation

$$\lambda_k \approx \mu_k \approx S_X(\tfrac{2\pi k}{n}), \qquad (3)$$

where $S_X(\omega) = \frac{1-\alpha^2}{1-2\alpha\cos\omega+\alpha^2}$ is the power spectral density of X [4, 5]. This approximation, however, dismisses the coding gain difference between the KLT and the DCT and obscures the dependence on n, so in the remainder of the paper we will use the exact values for the λ_k's and μ_k's.

Using high rate approximations and assuming optimal scalar quantization leads to the following optimal (arbitrary real) bit allocation for coding at rate R bits/sample:

$$b_i = R + \frac{1}{2}\log_2\frac{\lambda_i}{\rho^2}, \quad \text{where } \rho^2 = \left(\prod_{i=1}^{n}\lambda_i\right)^{1/n} \qquad (4)$$

Under these assumptions, the distortion is given by the following expressions [6]:

$$D_{KLT} = \frac{\sqrt{3}}{2}\left(\prod_{i=1}^{k}\lambda_i\right)^{1/n} 2^{-2R}$$

$$D_{DCT} = \frac{\sqrt{3}}{2}\left(\prod_{i=1}^{k}\mu_i\right)^{1/n} 2^{-2R}$$

Using the true component variances we obtain the dotted $D(n)$ curves shown in Fig. 1. By abandoning high rate approximations one can obtain more precise results. In particular, instead of using approximate expressions for optimal companding, we computed the performance using nonnegative integer bit allocation according to a greedy algorithm [6, §8.4] and uniform quantization with optimal loading. These results are also shown in Fig. 1, along with the performance obtained with no transform and $D_n(R)$. Note that for $\alpha = 0.9$, the performance of KLT coding is virtually indistinguishable from that of DCT coding. On the other hand, the performance gap is significant for $\alpha = -0.9$, although as in the previous case, the DCT is asymptotically equivalent to the KLT. Also, the high rate approximations are quite poor; they even give distortion values below the theoretical minimum. The discrepancy arises largely from not enforcing a nonnegativity constraint on the bit allocation. Distortions calculated based on greedy nonnegative integer bit allocation and optimally loaded quantizers are used below.

[2] Multiplications in which one or more multiplicands are rational are not counted.

[3] The latter quantities are $R_n(D)$ points as defined in [1].

[4] This is the "original" DCT first reported in [2] and classified as DCT-II in [3].

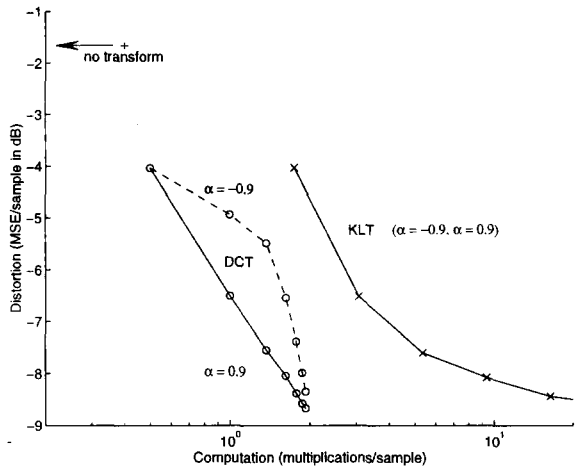


Figure 2. *Operational $D(C)$ for DCT and KLT coding of AR(1) sources with correlation coefficients $\alpha = -0.9$ and $\alpha = 0.9$ at rate 0.5 bits/sample. The complexity is based on implementations that minimize the number of multiplications. The block sizes are powers of two.*

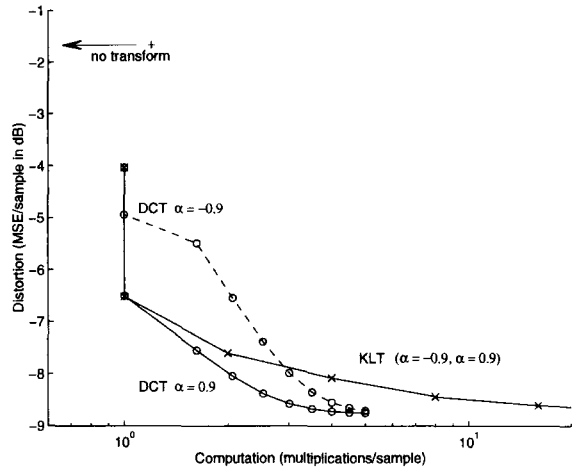


Figure 3. *Operational $D(C)$ for DCT and KLT coding of AR(1) sources with correlation coefficients $\alpha = -0.9$ and $\alpha = 0.9$ at rate 0.5 bits/sample. The complexity is based on implementations without an inordinate number of additions. The block sizes are powers of two.*

3.1.2. Estimating computational load

Consider first the computational complexity measure given by the number of multiplications between arbitrary real numbers per input sample. This model is familiar because of its connection to Winograd convolution algorithms and provable complexity lower bounds [7, 8].

The computation of any square linear transform of size n can be viewed as a multiplication between an $n \times n$ matrix and an $n \times 1$ vector. Since the KLT does not in general have a structure conducive to a fast algorithm, the KLT algorithms that minimize the number of multiplications are simply those that use the most efficient matrix multiplication techniques. For convenience, we will assume that $n = 2^s$, $s \in \mathbb{Z}^+$. To code n vectors at a time would entail multiplying pairs of $n \times n$ matrices, which can be done with $n^{\log_2 7}$ multiplications using Strassen's method [7]. Normalizing by the n^2, the number of samples transformed in each multiplication, gives a multiplicative complexity of

$$C_{KLT} = n^{(\log_2 7)-2} \text{ multiplies/sample.} \qquad (5)$$

On the other hand, the special structure of the DCT allows calculations to be done much more efficiently than as a general matrix multiplication, especially when n is a power of two. The minimum number of multiplications to compute a length-n DCT is $2n - \log_2 n - 2$ [8]. Normalizing gives

$$C_{DCT} = 2 - \tfrac{1}{n}(\log_2 n - 2) \text{ multiplies/sample.} \qquad (6)$$

When using moderate block sizes and usual computer architectures, algorithms that minimize the number of multiplications are generally not efficient. For example, while Strassen's algorithm for multiplying a pair of 2×2 matrices uses only 7 multiplications (instead of the usual 8), it increases the number of additions from 4 to 18. Similarly, DCT algorithms that have very low numbers of multiplications tend to have more additions and more complicated data flow. Therefore we would like to also compare multiplicative complexity for typical implementations of the KLT and DCT.

Performing the KLT using typical matrix-vector multiplication requires n^2 multiplications. If $m < n$ of the transformed components are allocated bits in the coding, the computational load can be reduced to mn multiplications (or m multiplications per sample) by computing only the components that will be coded. For fixed n, m is determined explicitly by the bit allocation algorithm. One approximation of m is the number of components allocated at least one bit in (4). For large n, combining this with (3) yields $\frac{m}{n} \approx \frac{\omega^*}{\pi}$, where $\omega^* \in [0, \pi)$ is a solution of $(1-\alpha)^2 2^{2R} = 1 - 2\alpha\cos\omega + \alpha^2$. Since this approximation masks the difference in energy compaction between the KLT and the DCT, explicitly computed greedy nonnegative integer bit allocations were used instead.

For n a power of two, one possible implementation of the DCT (which does not have an inordinate number of additions) has $\frac{1}{2} n \log_2 n$ multiplications [3]. To maintain an analogy with the multiplication count for the KLT, we should consider pruned computations that determine only the DCT coefficients with positive bit allocations. Since the complexities of pruned DCT algorithms are not easily captured in a single expression, we use the following estimate:

$$C_{DCT} = \min\{m, \tfrac{1}{2}\log_2 n\} \text{ multiplies/sample,} \qquad (7)$$

where as above m is the number of coefficients allocated at least one bit. This reflects the strategy of using a matrix multiplication when it is more efficient than a full DCT.

3.1.3. Operational distortion-computation

Having described $D(n)$ and $C(n)$, we have parametric descriptions of the operational $D(C)$ for KLT and DCT coding. Throughout this subsection, $D(n)$ for optimally loaded uniform quantization and greedy nonnegative integer bit allocation are used. Fig. 2 shows operational $D(C)$ curves for coding at 0.5 bits/sample when computation is estimated using (5)–(6). This graph shows $D_{DCT}(C) < D_{KLT}(C)$ for all computational budgets C. The graph also shows the distortion that is obtained when no transform is used, and the signal is simply subjected to uniform scalar quantization. This is indicated with an arrow because zero multiplications is off the left edge of the plot.

The operational $D(C)$ curves look somewhat different when the computational complexity is measured using (7) and the corresponding expression for the KLT. These operational curves are shown in Fig. 3. For $\alpha = 0.9$, as before, the DCT is superior for all computational budgets. On the other hand, for $\alpha = -0.9$ the DCT is superior only when the block size is larger than about 64.

3.1.4. Limitation of the computational model

The calculations made thus far pertain only to *operational* $D(C)$ for various implementations of two particular transform methods. The true distortion-computation function over the class of algorithms that follows a linear transform with scalar quantization—where complexity is measured by the number of general multiplications—has a very simple form. For any block size n, one can approximate the KLT of the source by a rational matrix. Since multiplication by rational numbers has no cost, the computational complexity of using this transform is zero. By making n arbitrarily large and using an arbitrarily good approximation of the KLT, we find that $D(C) = d$ for all C, where d is the distortion obtained with ideal IIR linear prediction of the source.

The infeasibility of using the coding strategy described above highlights the importance of having a good computational complexity metric. Good metrics would reflect actual costs in some application environment, *e.g.* execution time with particular hardware or hardware costs to meet certain performance specifications. Yet at the same time, when the implementations used are limited to practical schemes as in Fig. 3, this framework can give a reasonable basis of comparison between algorithms.

3.2. Coding with finite precision computations

Regardless of the transform used, transform coefficients are coarsely quantized in low rate coding, This is *prima facie* evidence that there is no point in computing these coefficients with high accuracy. In this section we analyze the relative benefits of spending computational resources on accurate transform coefficient calculation and on increased block lengths.

Consider a KLT-based coding system which uses nonnegative integer bit allocation. As before, denote by m the number of transform coefficients that have positive bit allocations. If the i-th transform coefficient is computed with a B_i-bit mantissa, a reasonable cost function for a hardware implementation is

$$C = \sum_{i=1}^{m} B_i^2 + B_i(1 - \tfrac{1}{n}).$$

This is based on costs of B^2 and B for each B-bit multiplication and addition, respectively. Modeling each roundoff as the addition of uniformly distributed noise and using the central limit theorem leads to the approximation $\hat{y}_i = y_i(1+\delta_i)$ for each computed transform coefficient, where y_i is the exact transform coefficient and $\delta_i \sim \mathcal{N}(0, 2^{-2B_i}/3n)$. An optimization was performed based on the additional assumption that the quantization error is independent of the errors from finite precision computations. The resulting operational $D(C)$ is as shown in Fig. 4.

4. COMMENTS

The computational complexity calculations in Section 3.1.2 used the fact that transform coefficients that are not allocated any bits need not be calculated. This can be viewed as a special case of the principle that transform coefficient calculations need only be accurate enough to determine proper *quantized* values.[5] Obviously, determining which of a finite set of intervals a transform coefficient lies in can not be harder than calculating the coefficient precisely. In DCT-based image coding, quantized transform coefficients are often zero, in part because of the use of deadzone quantizers. Methods to efficiently identify coefficients that quantize to zero are currently under investigation. Such coefficients need not be explicitly computed, and a corresponding pruned DCT algorithm can be used.

[5] Here we are assuming computational optimization that does not effect rate-distortion performance. More generally, distortion incurred due to low precision calculations should be measured after quantization.

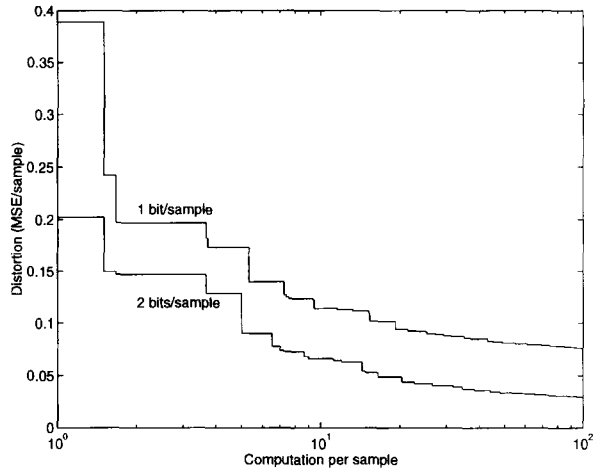


Figure 4. *Operational $D(C)$ for transform coding of an AR(1) source with correlation coefficient $\alpha = 0.9$ in which the block length and computational precisions of each transform coefficient are jointly optimized.*

In this paper only the encoding operations associated with transform coding were considered. The scheme suggested in the previous paragraph is the dual of a method used to optimize *decoders*; many decoders for DCT-based image coding enjoy substantial speedups by eliminating the regular inverse DCT when an all-zero row or column is detected. The design of such input-dependent algorithms is generally done in an ad hoc fashion. A notable exception is work presented in [9], where the following approach to input-dependent inverse DCT computation is proposed: classify each input block based on its sparsity structure and use a pruned DCT algorithm optimized for that sparsity structure. The classes are designed to minimize the average computational load over a training set.

REFERENCES

[1] T. Berger. *Rate Distortion Theory.* Prentice-Hall, 1971.

[2] N. Ahmed, T. Natarajan, and K. R. Rao. Discrete cosine transform. *IEEE Trans. Comp.*, 23:90–93, Jan. 1974.

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

[4] U. Grenander and G. Szegö. *Toeplitz Forms and Their Applications.* Univ. California Press, 1958.

[5] R. M. Gray. On the asymptotic eigenvalue distribution of Toeplitz matrices. *IEEE Trans. Info. Theory*, 18(6):725–730, Nov. 1972.

[6] A. Gersho and R. M. Gray. *Vector Quantization and Signal Compression.* Kluwer Acad. Pub., 1992.

[7] R. E. Blahut. *Fast Algorithms for Digital Signal Processing.* Addison-Wesley, 1985 (with corrections 1987).

[8] M. T. Heideman. *Multiplicative Complexity, Convolution, and the DFT.* Springer-Verlag, 1988.

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