Theoretical Foundations of Transform Coding

Vivek K Goyal

Divide and conquer. This principle is central to many endeavors, ranging from children managing inconsistent parents to colonial powers controlling native peoples. In engineering and computational science it means breaking a big problem into smaller problems that can be more easily understood and solved. Putting the pieces back together gives a modular design, which is advantageous for implementation, testing, and component reuse.

In signal processing, we are familiar with the divide and conquer strategy because its recursive application is the essence of fast Fourier transform (FFT) algorithms. Recursive division leads also to efficient algorithms for searching, sorting, convolutions, and eigenvalue computations. Thus, in both recursive and nonrecursive forms, this strategy is fundamental.

Everyday compression problems are unmanageable without a divide and conquer approach. Effective compression of images, for example, depends on the tendencies of pixels to be similar to their neighbors or to differ in partially predictable ways. These tendencies, arising from the continuity, texturing, and boundaries of objects, the similarity of objects in an image, gradual lighting changes, an artist's technique and color palette, etc., may extend over an entire image with a quarter million pixels. Yet the most general way to utilize the probable relationships between pixels (later described as unconstrained source coding) is infeasible for this many pixels. In fact, 16 pixels is a lot for an unconstrained source code.

To conquer the compression problem—allowing, for example, more than 16 pixels to be encoded simultaneously—state-of-the-art lossy compressors divide the encoding operation into a sequence of three relatively simple steps: the computation of a linear transformation of the data designed primarily to produce uncorrelated coefficients, separate quantization of each scalar coefficient, and entropy coding. This process is called transform coding. In image compression, a square image with N pixels is typically processed with simple linear transforms (often discrete wavelet transforms) of size $\sqrt{N}$.

This article explains the fundamental principles of transform coding; these principles apply equally well to images, audio, video, and various other types of data, so abstract formulations are given. Much of the material presented here is adapted from [14, Chap. 2, 4]. The details on wavelet transform-based image compression and the JPEG2000 image compression standard are given in the following two articles of this special issue [38], [37].

Source Coding

Source coding is to represent information in bits, with the natural aim of using a small number of bits. When the information can be exactly recovered from the bits, the source coding or compression is called lossless; otherwise, it is called lossy. The transform codes in this article are lossy. However, lossless entropy codes appear as components of transform codes, so both lossless and lossy compression are of present interest.

In our discussion, the "information" is denoted by a real column vector $x \in \mathbb{R}^N$ or a sequence of such vectors. A vector might be formed from pixel values in an image or by sampling an audio signal; $K \cdot N$ pixels can be arranged as a sequence of K vectors of length N. The vector length N is defined such that each vector in a sequence is encoded independently. For the purpose of building a mathematical theory, the source vectors are assumed to be realizations of a random vector $\mathbf{x}$ with a known distribution. The distribution could be purely empirical.

A source code is comprised of two mappings: an encoder and a decoder. The encoder maps any vector $x \in \mathbb{R}^N$ to a finite string of bits, and the decoder maps any of these strings of bits to an approximation $\hat{x} \in \mathbb{R}^N$. The encoder mapping can always be factored as $\gamma \circ \alpha$, where α is a mapping from $\mathbb{R}^N$ to some discrete set $\mathcal{I}$ and γ is an invertible

1053-5888/01/$10.00©2001IEEE

The standard theoretical model for transform coding has strict modularity , meaning that the transform, quantization, and entropy coding blocks operate independently.

mapping from $\mathcal{I}$ to strings of bits. The former is called a lossy encoder and the latter a lossless code or an entropy code. The decoder inverts γ and then approximates x from the index $\alpha(x) \in \mathcal{I}$. This is shown in the top half of Fig. 1. It is assumed that communication between the encoder and decoder is perfect. (The last article of this issue [13] describes techniques that work when some transmitted bits are lost.)

To assess the quality of a lossy source code, we need numerical measures of approximation accuracy and description length. The measure for description length is simply the expected number of bits output by the encoder divided by N; this is called the rate in bits per scalar sample and denoted by R. Here we will measure approximation accuracy by squared Euclidean norm divided by the vector length

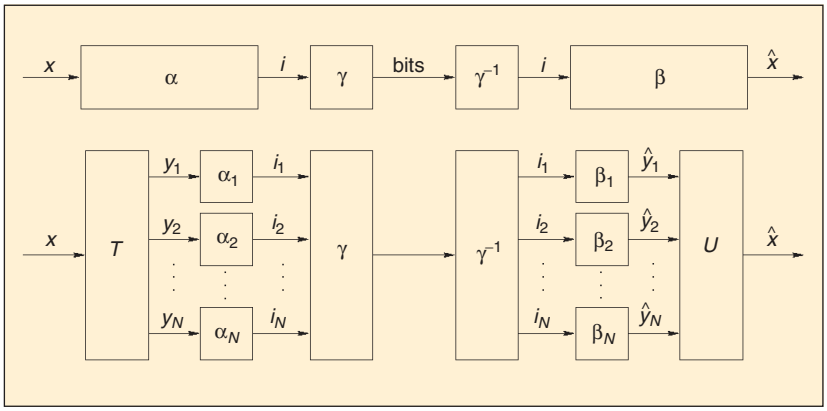
$$d(x,\hat{x}) = \frac{1}{N}\|x-\hat{x}\|^2 = \frac{1}{N}\sum_{i=1}^{N}(x_i-\hat{x}_i)^2 .$$

This accuracy measure is conventional and usually leads to the easiest mathematical results, though the theory of source coding has been developed with quite general measures [1]. The expected value of $d(\mathbf{x},\hat{\mathbf{x}})$ is called the mean-squared error (MSE) distortion and is denoted by $D = E[d(\mathbf{x},\hat{\mathbf{x}})]$. The normalizations by N make it possible to fairly compare source codes with different lengths.

Fixing N, a theoretical concept of optimality is straightforward: A length-N source code is *optimal* if no other length-N source code with at most the same rate has lower distortion. This concept is of dubious value. First, it is very difficult to check the optimality of a source code. Local optimality—being assured that small perturbations of α and β will not improve performance—is often the best that can be attained [16]. Second, and of more practical consequence, a system designer gets to choose the value of N. It can be as large as the total size of the data set—like the number of pixels in an image—but can also be smaller, in which case the data set is interpreted as a sequence of vectors.

There are conflicting motives in choosing N. Compression performance is related to the predictability of one part of x from the rest. Since predictability can only increase from having more data, performance is usually improved by increasing N. (Even if the random variables producing each scalar sample are mutually independent, the optimal performance is improved by increasing N; however, this "packing gain" effect is relatively small [16].) The conflict comes from the fact that the computational complexity of encoding is also increased. This is particularly dramatic if one looks at complexities of optimal source codes. The obvious way to implement an optimal encoder is to search through the entire codebook, giving running time exponential in N. Other implementations reduce running time while increasing memory usage [29].

State-of-the-art source codes result from an intelligent compromise. There is no attempt to realize an optimal code for a given value of N because encoding complexity would force a small value for N. Rather, source codes that are good, but plainly not optimal, are used. Their lower complexities make much larger N's feasible. A tutorial in an earlier issue of this *Magazine* [5] called this "the power of imperfection." The paradoxical conclusion is that the best codes to use in practice are suboptimal.



▲ *1. Any source code can be decomposed so that the encoder is $\gamma \circ \alpha$ and the decoder is $\beta \circ \gamma^{-1}$, as shown at top. γ is an entropy code and α and β are the encoder and decoder of an N-dimensional quantizer. In a transform code, α and β each have a particular constrained structure. In the encoder, α is replaced with a linear transform T and a set of N scalar quantizer encoders. The intermediate y_is are called transform coefficients. In the decoder, β is replaced with N scalar quantizer decoders and another linear transform U. Usually $U = T^{-1}$.*

Constrained Source Coding

Transform codes are the most used source codes because they are easy to apply at any rate and even with very large values of N. The essence of transform coding is the modularization shown in the bottom half of Fig. 1. The mapping α is implemented in two steps. First, an invertible linear transform of the source vector x is computed, producing $y = Tx$. Each component of y is called a transform coefficient. The N transform coefficients are then quantized independently of each other by N scalar quantizers. This is called scalar quantization since each scalar component of y is treated separately. Finally, the quantizer indexes that correspond to the transform coefficients

are compressed with an entropy code to produce the sequence of bits that represent the data.

To reconstruct an approximation of x, the decoder essentially reverses the steps of the encoder. The action of the entropy coder can be inverted to recover the quantizer indices. Then the decoders of the scalar quantizers produce a vector $\hat{y}$ of estimates of the transform coefficients. To complete the reconstruction, a linear transform is applied to $\hat{y}$ to produce the approximation $\hat{x}$. This final step usually uses the transform T^{-1}, but for generality the transform is denoted U.

Most source codes cannot be implemented in the two stages of linear transform and scalar quantization. Thus, a transform code is an example of a constrained source code. Constrained source codes are, loosely speaking, source codes that are suboptimal but have low complexity. The simplicity of transform coding allows large values of N to be practical. Computing the transform T requires at most N^2 multiplications and $N(N-1)$ additions. Specially structured transforms—like discrete Fourier, cosine, and wavelet transforms—are often used to reduce the complexity of this step, but this is merely icing on the cake. The great reduction from the exponential complexity of a general source code to the (at most) quadratic complexity of a transform code comes from using linear transforms and scalar quantization. See Box 1 for more on constrained source codes.

The Standard Model and Its Components

The standard theoretical model for transform coding looks like the bottom of Fig. 1. It has the strict modularity shown, meaning that the transform, quantization and entropy coding blocks operate independently. In addition, the entropy coder can be decomposed into N parallel entropy coders so that the quantization and entropy coding operate independently on each scalar transform coefficient.

This section briefly describes the fundamentals of entropy coding and quantization to provide background for our later focus on the optimization of the transform. The final part of this section addresses the allocation of bits among the N scalar quantizers. Sources for additional information include [3], [8], [14], [16].

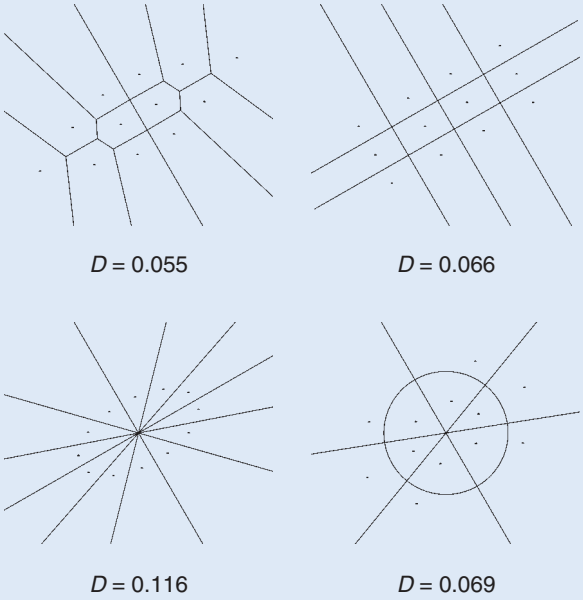
Entropy Codes

Entropy codes are used for lossless coding of discrete random variables. Consider the discrete random variable $\mathbf{z}$ with alphabet $\mathcal{I}$. An entropy code γ assigns a unique binary string, called a *codeword*, to each $i \in \mathcal{I}$. (See Fig. 1.)

Since the codewords are unique, an entropy code is always invertible. However, we will place more restrictive conditions on entropy codes so they can be used on sequences of realizations of $\mathbf{z}$. The extension of γ maps the finite sequence $(z_1, z_2, \ldots, z_k)$ to the concatenation of the outputs of γ with each input, $\gamma(z_1)\ \gamma(z_2), \ldots, \gamma(z_k)$. A code is called uniquely decodable if its extension is one-to-one. A uniquely decodable code can be applied to message se-

Box 1
Constrained and Unconstrained Source Codes

The difference between constrained and unconstrained source codes is shown by partition diagrams. In these diagrams, the cells indicate which source vectors are encoded to the same index and the dots are the reconstructions computed by the decoder. Four locally optimal fixed-rate source codes with 12-element codebooks were constructed. The two-dimensional, jointly Gaussian source is the same as that used in Fig. 2 and Box 6.



The upper-left plot shows an unconstrained code. The partition shares the symmetries of the source density but is otherwise complicated because the cell shapes are arbitrary. Encoding is difficult because there is no simple way to get around using both components of the source vector simultaneously in computing the index.

Encoding with a transform code is easier because after the linear transform the coefficients are quantized separately. This gives the structured alignment of partition cells and of reconstruction points in the upper-right plot.

It is fair to ask why the transform is linear. In two dimensions, one might imagine quantizing in polar coordinates. Two examples of partitions obtained with separate quantization of radial and angular components are shown in the lower plots, and these are as elegant as the partition obtained with a linear transform. Yet nonlinear transformations—even transformations to polar coordinates—are rarely used in source coding. With arbitrary transformations, the approximation accuracy of the transform coefficients does not easily relate to the accuracy of the reconstructed vectors. This makes designing quantizers for the transform coefficients more difficult. Also, allowing nonlinear transformations reintroduces the design and encoding complexities of unconstrained source codes.

Constrained source codes need not use transforms to have low complexity. Techniques described in [8] and [16] include those based on lattices, sorting, and tree-structured searching; none of these techniques is as popular as transform coding.

Box 2
Huffman Codes

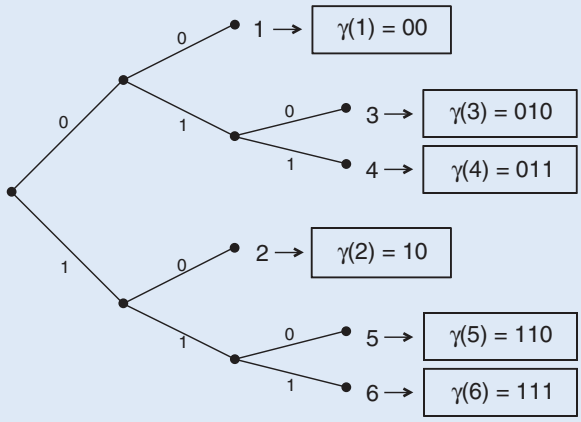
There is a simple algorithm, due to Huffman [22], for constructing optimal entropy codes. One starts with a graph with one node for each symbol and no edges. These nodes will become the leaves of a tree as edges are added to make a connected graph.

At each step of the algorithm, the probabilities of the disconnected sets of nodes are sorted and the two least probable sets are merged through the addition of a parent node and edges to each of the two sets. The edges are assigned labels of 0 and 1. When a tree has been formed, codewords are assigned to each leaf node by concatenating the edge labels on the path from the root to the leaf.

Shown below is a Huffman code tree for symbols {1, 2, 3, 4, 5, 6} with respective probabilities {0.3, 0.26, 0.14, 0.13, 0.09, 0.08}. The codewords are boxed. Computing a weighted sum of the codeword lengths gives the expected code length

$$L = 0.3\cdot 2 + 0.26\cdot 2 + 0.14\cdot 3 + 0.13\cdot 3 + 0.09\cdot 3 + 0.08\cdot 3 = 2.44 \text{ bits.}$$

This is quite close to the entropy of 2.41 bits obtained by evaluating (2).



quences without adding any "punctuation" to show where one codeword ends and the next begins. In a prefix code, no codeword is the prefix of any other codeword. Prefix codes are guaranteed to be uniquely decodable.

A trivial code numbers each element of $\mathcal{I}$ with a distinct index in $\{0, 1, \ldots, |\mathcal{I}|-1\}$ and maps each element to the binary expansion of its index. Such a code requires $\lceil \log_2 |\mathcal{I}| \rceil$ bits per symbol. This is considered the lack of an entropy code. The idea in entropy code design is to minimize the mean number of bits used to represent $\mathbf{z}$ at the expense of making the worst-case performance worse. The expected code length is given by

$$L(\gamma) = E[\ell(\gamma(\mathbf{z}))] = \sum_{i\in\mathcal{I}} p_{\mathbf{z}}(i)\,\ell(\gamma(i)),$$

where $p_{\mathbf{z}}(i)$ is the probability of symbol i and $\ell(\gamma(i))$ is the length of $\gamma(i)$. The expected length can be reduced if short code words are used for the most probable symbols—even if it means that some symbols will have codewords with more than $\lceil \log_2 |\mathcal{I}| \rceil$ bits.

The entropy code γ is called optimal if it is a prefix code that minimizes $L(\gamma)$. Huffman codes, described in Box 2, are examples of optimal codes. The performance of an optimal code is bounded by

$$H(\mathbf{z}) \le L(\gamma) < H(\mathbf{z}) + 1 \tag{1}$$

where

$$H(\mathbf{z}) = -\sum_{i\in\mathcal{I}} p_{\mathbf{z}}(i) \log_2 p_{\mathbf{z}}(i) \tag{2}$$

is the entropy of $\mathbf{z}$.

The up to one bit gap in (1) is ignored in the remainder of the article. If $H(\mathbf{z})$ is large, this is justified simply because one bit is small compared to the code length. Otherwise note that $L(\gamma) \approx H(\mathbf{z})$ can attained by coding blocks of symbols together; this is detailed in any information theory or data compression textbook.

Quantizers

A quantizer q is a mapping from a source alphabet $\mathbb{R}^N$ to a reproduction codebook $\mathcal{C} = \{\hat{x}_i\}_{i\in\mathcal{I}} \subset \mathbb{R}^N$, where $\mathcal{I}$ is an arbitrary countable index set. Quantization can be decomposed into two operations $q = \beta \circ \alpha$, as shown in Fig. 1. The lossy encoder $\alpha: \mathbb{R}^N \to \mathcal{I}$ is specified by a partition of $\mathbb{R}^N$ into partition cells $S_i = \{x \in \mathbb{R}^N \,|\, \alpha(x) = i\}, i \in \mathcal{I}$. The reproduction decoder $\beta: \mathcal{I} \to \mathbb{R}^N$ is specified by the codebook $\mathcal{C}$. If $N = 1$, the quantizer is called a scalar quantizer; for $N > 1$, it is a vector quantizer.

The quality of a quantizer is determined by its distortion and rate. The MSE distortion for quantizing random vector $\mathbf{x} \in \mathbb{R}^N$ is

$$D = E[d(\mathbf{x}, q(\mathbf{x}))] = N^{-1} E\left[\, \|\mathbf{x} - q(\mathbf{x})\|^2 \,\right].$$

The rate can be measured in a few ways. The lossy encoder output $\alpha(\mathbf{x})$ is a discrete random variable that usually should be entropy coded because the output symbols will have unequal probabilities. Associating an entropy code γ to the quantizer gives a variable-rate quantizer specified by (α, β, γ). The rate of the quantizer is the expected code length of γ divided by N. Not specifying an entropy code (or specifying the use of fixed-rate binary expansion) gives a fixed-rate quantizer with rate $R = N^{-1} \log_2 |\mathcal{I}|$. Measuring the rate by the idealized performance of an entropy code gives $R = N^{-1} H(\alpha(\mathbf{x}))$; the quantizer in this case is called entropy constrained.

The optimal performance of variable-rate quantization is at least as good as that of fixed-rate quantization, and entropy-constrained quantization is better yet. Entropy coding adds complexity, however, and variable length output can create difficulties such as buffer overflows.

Furthermore, entropy-constrained quantization is only an idealization since an entropy code will generally not meet the lower bound in (1).

Optimal Quantization

An optimal quantizer is one that minimizes the distortion subject to an upper bound on the rate or minimizes the rate subject to an upper bound on the distortion. Because of simple shifting and scaling properties, an optimal quantizer for a scalar $\mathbf{x}$ can be easily deduced from an optimal quantizer for the normalized random variable $\mathbf{w} = (\mathbf{x} - \mu_x)/\sigma_x$, where μ_x and σ_x are the mean and standard deviation of $\mathbf{x}$, respectively. One consequence of this is that optimal quantizers have performance

$$D = \sigma^2 \, g(R), \tag{3}$$

where σ^2 is the variance of the source and $g(R)$ is the performance of optimal quantizers for the normalized source. Equation (3) holds, with a different function g, for any family of quantizers that can be described by its operation on a normalized variable, not just optimal quantizers.

Optimal quantizers are difficult to design, but locally optimal quantizers can be numerically approximated by an iteration in which α, β, and γ are separately optimized, in turn, while keeping the other two fixed. For details on each of these optimizations and the difficulties and properties that arise, see [5] and [16].

Note that the rate measure affects the optimal encoding rule because $\alpha(x)$ should be the index that minimizes a Lagrangian cost function including both rate and distortion; for example

$$\alpha(x) = \operatorname*{argmin}_{i \in \mathcal{I}} \left[\frac{1}{N} \ell(\gamma(i)) + \lambda \frac{1}{N} \| x - \beta(i) \|^2 \right]$$

is an optimal lossy encoder for variable-rate quantization. (By fixing the relative importance of rate and distortion, the Lagrange multiplier λ determines a rate-distortion operating point among those possible with the given β and γ.) Only for fixed-rate quantization does the optimal encoding rule simplify to finding the index corresponding to the nearest codeword.

In some of the more technical discussions that follow, one property of optimal decoding is relevant: The optimal decoder β computes

$$\beta(i) = E[\mathbf{x} | \mathbf{x} \in S_i],$$

which is called centroid reconstruction. The conditional mean of the cell, or centroid, is the minimum MSE estimate [31].

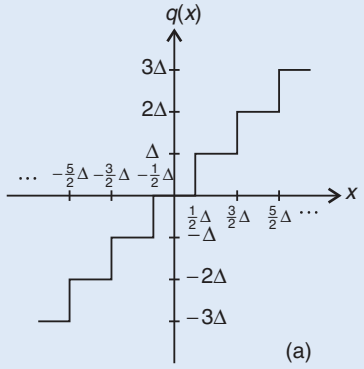
High Resolution Quantization

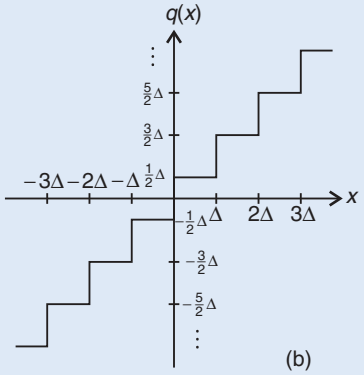
For most sources, it is impossible to analytically express the performance of optimal quantizers. Thus, aside from

Box 3
Uniform Quantization

The design of quantizers has a deep and fascinating theory. Nevertheless, the fact remains [16]: "Most quantizers today are indeed uniform and scalar, but are combined with prediction or transforms."

Though definitions of uniform quantization vary somewhat, the archetype of rounding is always an example of uniform quantization. Shown in (a) is the input-output relationship of a device that rounds to the nearest integer multiple of *step size* Δ. To describe this in formal notation, the encoder could be $\alpha(x) = \text{round}(x/\Delta)$, where round$(\cdot)$ denotes rounding to the nearest integer, with corresponding decoder $\beta(i) = i\Delta$.





The other common uniform quantizer is shown in (b). This is a shifted version of the previous uniform quantizer. Variations in the definition of uniform quantization sometimes allow only the encoder to have equal length cells or only the decoder to have evenly spaced outputs and may also allow the decoder outputs to be shifted from the centers of the partition cells.

Uniform quantization of a uniform source provides a setting to see the typical trade-off between rate and distortion. Consider $\mathbf{x}$ uniformly distributed on the interval [0,1). A fixed-rate uniform quantizer, as on the right above, with K cells and step size $\Delta = 1/K$ quantizes $x \in [(m-1)\Delta, m\Delta)$ to $(m - 1/2)\Delta$ for $m = 1, 2, \ldots, K$. It has rate $R = \log_2 N = -\log_2 \Delta$ and distortion

$$D = \int_0^1 (x - \hat{x})^2 \, dx = \sum_{m=1}^{N} \int_{(m-1)\Delta}^{m\Delta} \left(x - \left(m - \frac{1}{2} \right) \Delta \right)^2 dx = \frac{1}{12} \Delta^2.$$

Thus, $D = (1/12) 2^{-2R}$. When using the MSE distortion measure, there will almost always be a 2^{-2R} factor.

The motivating principle of transform coding is that simple coding may be more effective in the transform domain than in the original signal space.

using (3), approximations must suffice. Fortunately, approximations obtained when it is assumed that the quantization is very fine are reasonably accurate even at low to moderate rates [10], [40]. Details on this "high resolution" theory for both scalars and vectors can be found in [7] and [16] and references therein.

Let $f_{\mathbf{x}}(x)$ denote the probability density function (pdf) of the scalar random variable $\mathbf{x}$. High resolution analysis is based on approximating $f_{\mathbf{x}}(x)$ on the interval S_i by its value at the midpoint. Assuming $f_{\mathbf{x}}(x)$ is smooth, this approximation is accurate when each S_i is short.

Optimization of scalar quantizers turns into finding the optimal lengths for the S_is, depending on the pdf $f_{\mathbf{x}}(x)$. One can show that the performance of optimal fixed-rate quantization is approximately

$$D \approx \frac{1}{12}\left(\int_{\mathbb{R}} f_{\mathbf{x}}^{1/3}(x)\, dx\right)^3 2^{-2R}. \tag{4}$$

Evaluating this for a Gaussian source with variance σ^2 gives

$$D \approx \frac{\sqrt{3}\pi}{2}\sigma^2 2^{-2R}. \tag{5}$$

For entropy-constrained quantization, high resolution analysis shows that it is optimal for each S_i to have equal length [9]. A quantizer that partitions with equal-length intervals is called uniform (see Box 3). The resulting performance is

$$D \approx \frac{1}{12} 2^{2h(\mathbf{x})} 2^{-2R}, \tag{6}$$

where

$$h(\mathbf{x}) = -\int_{\mathbb{R}} f_{\mathbf{x}}(x) \log_2 f_{\mathbf{x}}(x)\, dx$$

is the differential entropy of $\mathbf{x}$. For Gaussian random variables, (6) simplifies to

$$D \approx \frac{\pi e}{6}\sigma^2 2^{-2R}. \tag{7}$$

Summarizing (4)-(7), the lesson from high resolution quantization theory is that quantizer performance is described by

$$D \approx c\sigma^2 2^{-2R}, \tag{8}$$

where σ^2 is the variance of the source and c is a constant that depends on the normalized density of the source and the type of quantization (fixed rate, variable rate, or entropy constrained). This is consistent with (3).

The computations we have made are for scalar quantization. For vector quantization, the best performance in the limit as the dimension N grows is given by the distortion rate function [1] (see [13, Box 3]). For a Gaussian source this bound is $D = \sigma^2 2^{-2R}$. The approximate performance given by (7) is only worse by a factor of $\pi e/6$ (≈ 1.53 dB). This can be expressed as a redundancy $(1/2)\log_2(\pi e/6) \approx 0.255$ bits. Furthermore, a numerical study has shown that for a wide range of memoryless sources, the redundancy of entropy-constrained uniform quantization is at most 0.3 bits per sample at all rates [6].

Bit Allocation

Coding (quantizing and entropy coding) each transform coefficient separately splits the total number of bits among the transform coefficients in some manner. Whether done with conscious effort or implicitly, this is a bit allocation among the components.

Bit allocation problems can be stated in a single common form: One is given a set of quantizers described by their distortion-rate performances as

$$D_i = g_i(R_i), \quad R_i \in \mathcal{R}_i,\ i = 1, 2, \ldots, N.$$

Each set of available rates $\mathcal{R}_i$ is a subset of the nonnegative real numbers and may be discrete or continuous. The problem is to minimize the average distortion $D = N^{-1}\sum_{i=1}^{N} D_i$ given a maximum average rate $R = N^{-1}\sum_{i=1}^{N} R_i$.

As is often the case with optimization problems, bit allocation is easy when the parameters are continuous and the objective functions are smooth. Subject to a few other technical requirements, parametric expressions for the optimal bit allocation can be found [27], [35]. The techniques used when the $\mathcal{R}_i$s are discrete are quite different and play no role in forthcoming results [36].

If the average distortion can be reduced by taking bits away from one component and giving them to another, the initial bit allocation is not optimal. Applying this reasoning with infinitesimal changes in the component rates, a necessary condition for an optimal allocation is that the slope of each g_i at R_i is equal to a common constant value. A tutorial treatment of this type of optimization appeared in an earlier issue of this *Magazine* [30].

The approximate performance given by (8) leads to a particularly easy bit allocation problem with

$$g_i = c_i\sigma_i^2 2^{-2R_i}, \quad \mathcal{R}_i = [0, \infty),\ i = 1, 2, \ldots, N. \tag{9}$$

Ignoring the fact that each component rate must be nonnegative, an equal-slope argument shows that the optimal bit allocation is

$$R_i = R + \frac{1}{2}\log_2 \frac{c_i}{\left(\prod_{i=1}^{N} c_i\right)^{1/N}} + \frac{1}{2}\log_2 \frac{\sigma_i^2}{\left(\prod_{i=1}^{N} \sigma_i^2\right)^{1/N}}.$$

With these rates, all the D_is are equal and the average distortion is

$$D = \left(\prod_{i=1}^{N} c_i\right)^{1/N} \left(\prod_{i=1}^{N} \sigma_i^2\right)^{1/N} 2^{-2R}. \tag{10}$$

This solution is valid when each R_i given above is nonnegative. For lower rates, the components with smallest $c_i \cdot \sigma_i^2$ are allocated no bits and the remaining components have correspondingly higher allocations.

Bit Allocation with Uniform Quantizers

With uniform quantizers, bit allocation is nothing more than choosing a step size Δ_i for each of the N components. The equal-distortion property of the analytical bit allocation solution gives a simple rule: Make all of the step sizes equal. This will be referred to as "lazy" bit allocation.

Our development indicates that lazy allocation is optimal when the rate is high. Numerical studies have shown that lazy allocation is nearly optimal as long as the minimal allocated rate is at least 1 bit [14], [15]. Entropy-constrained uniform quantization with lazy bit allocation is used in the numerical examples in the following section.

Optimal Transforms

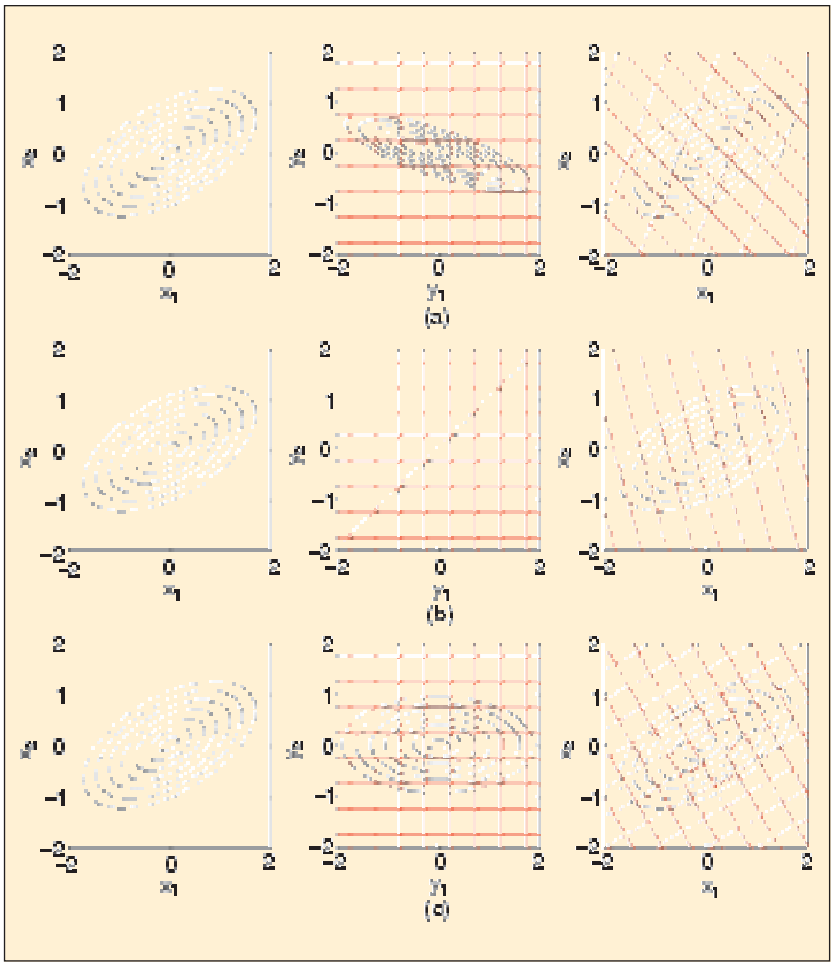
It has taken some time to set the stage, but we are now ready for the main event of designing the analysis transform T and the synthesis transform U. Throughout this section the source $\mathbf{x}$ is assumed to have mean zero, and R_x denotes the covariance matrix $E[\mathbf{x}\mathbf{x}^T]$, where ^{T} denotes the transpose. The source is often, but not always, jointly Gaussian.

A signal given as a vector in $\mathbb{R}^N$ is implicitly represented as a series with respect to the standard basis. An invertible analysis transform T changes the basis. A change of basis does not alter the information in a signal, so how can it affect coding efficiency? Indeed, if arbitrary source coding is allowed after the transform, it does not. The motivating principle of transform coding is that simple coding may be more effective in the transform domain than in the original signal space. In the standard model, "simple coding" corresponds to the use of scalar quantization and scalar entropy coding.

Visualizing Transforms

Beyond two or three dimensions, it is difficult to visualize vectors—let alone the action of a transform on vectors. Fortunately, most people already have an idea of what a linear transform does: it combines rotating, scaling, and shearing such that a hypercube is always mapped to a parallelepiped.

In two dimensions, the level curves of a zero-mean Gaussian density are ellipses centered at the origin with collinear major axes, as shown in the left panels of Fig. 2. The middle panel of Fig. 2(a) shows the level curves of the joint density of the transform coefficients after a more or less arbitrary invertible linear transformation. A linear transformation of an ellipse is still an ellipse, though its eccentricity and orientation (direction of major axis) may have changed.



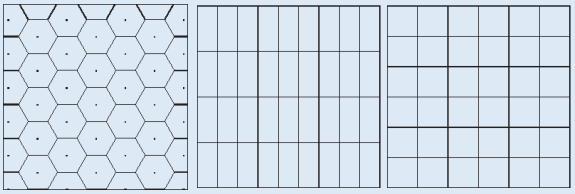
▲ *2. Illustration of various basis changes. The source is depicted by level curves of the pdf (left). The transform coefficients are separately quantized with uniform quantizers (center). The induced partitioning is then shown in the original coordinates (right). (a) A basis change generally induces a nonhypercubic partition. (b) A singular transformation gives a partition with unboundedn cells. (c) A Karhunen -Loève transform is an orthogonal transform that aligns the partitioning with the axes of the source pdf.*

Box 4
Shapes of Partition Cells

The quality of a source code depends on the shapes of the partition cells $\{\alpha^{-1}(i), i \in \mathcal{I}\}$ and on varying the sizes of the cells according to the source density. When the rate is high, and either the source is uniformly distributed or the rate is measured by entropy ($H(\alpha(\mathbf{x}))$), the sizes of the cells should essentially not vary. Then, the quality depends on having cell shapes that minimize the average distance to the center of the cell.

For a given volume, a body in Euclidean space that minimizes the average distance to the center is a sphere. But spheres do not work as partition cell shapes because they do not pack together without leaving interstices. Only for a few dimensions N is the best cell shape known [2]. One such dimension is $N = 2$, where the hexagonal packing shown below (left) is best.

The best packings (including the hexagonal case) cannot be achieved with transform codes. Transform codes can only produce partitions into parallelepipeds, as shown for $N = 2$ in Fig. 2. The best parallelepipeds are cubes. We get a hint of this by comparing the two rectangular partitions of a unit-area square shown below. Both partitions have 36 cells, so every cell has the same area. The partition with square cells gives distortion $1/432 \approx 2.31 \times 10^{-3}$, while the other gives $97/31104 \approx 3.12 \times 10^{-3}$. (The calculations are easy; see Box 3.)



This simple example can also be interpreted as a problem of allocating bits between the horizontal and vertical components. The "lazy" bit allocation arising from equal quantization step sizes for each component is optimal. This holds generally for high-rate entropy-constrained quantization of components with the same normalized density.

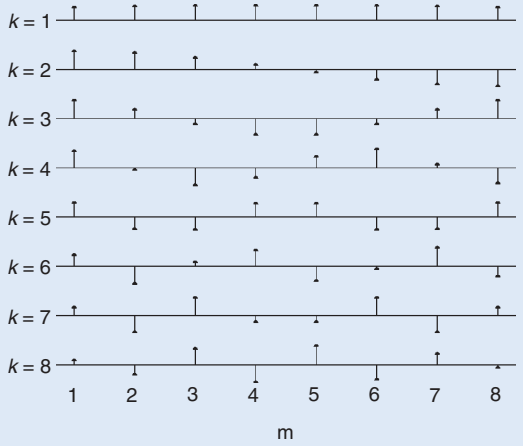
Box 5
Autoregressive Source

Autoregressive models are popular in many branches of signal processing. Under such a model, a sequence is generated as

$$\mathbf{x}[k] = \rho \mathbf{x}[k-1] + \mathbf{z}[k]$$

where k is a time index, $\mathbf{z}[k]$ is a white sequence, and $\rho \in [0,1)$ is called the *correlation coefficient*. It is a crude but useful model for the samples along any line of a grayscale image, with $\rho \approx 0.9$.

An $\mathbb{R}^N$-valued source $\mathbf{x}$ can be derived from a scalar autoregressive source by forming blocks of N consecutive samples. With normalized power, the covariance matrix is given elementwise by $(R_{\mathbf{x}})_{ij} = \rho^{|i-j|}$. For any particular N and ρ, a numerical eigendecomposition method can be applied to $R_{\mathbf{x}}$ to obtain a KLT. (An analytical solution happens to also be possible [17], [34].) For $N = 8$ and $\rho = 0.9$, the KLT can be depicted as below:



Each subplot (value of k) gives a row of the transform or, equivalently, a vector in the analysis basis. This basis is superficially sinusoidal and is approximated by a discrete cosine transform (DCT) basis. There are various asymptotic equivalences between KLTs and DCTs for large N and $\rho \to 1$ [23], [33]. These results are often cited in justifying the use of DCTs.

The grid in the middle panel indicates the cell boundaries in uniform scalar quantization, with equal step sizes, of the transform coefficients. The effect of inverting the transform is shown in the right panel; the source density is returned to its original form and the quantization partition is linearly deformed. The partition in the original coordinates, as shown in the right panel, is what is truly relevant. It shows which source vectors are mapped to the same symbol, thus giving some indication of the average distortion. Looking at the number of cells with appreciable probability gives some indication of the rate.

A singular transform is a degenerate case. As shown in the middle panel of Fig. 2(b), the transform coefficients have probability mass only along a line. (A line segment is an ellipse with unit eccentricity.) Inverting the transform is not possible, but we may still return to the original coordinates to view the partition induced by quantizing the transform coefficients. The cells are unbounded in one direction, as shown in the right panel. This is undesirable unless variation of the source in the direction in which the cells are unbounded is very small.

Although better than unbounded cells, the parallelogram-shaped partition cells that arise from arbitrary invertible transforms are inherently suboptimal (see Box 4). To get rectangular partition cells, the basis vectors must be orthogonal. For square cells, when quantization step sizes are equal for each transform coefficient, the basis vectors should in addition to being orthogonal have equal

Transform codes are easy to implement because of a divide and conquer strategy; the transform exploits dependencies in the data so that the quantization and entropy coding can be simple.

lengths. When orthogonal basis vectors have unit length, the resulting transform is called an orthogonal transform. (It is regrettable that of a matrix or transform, "orthogonal" means orthonormal.)

A Karhunen-Loève transform (KLT) is a particular type of orthogonal transform that depends on the covariance of the source. An orthogonal matrix T represents a KLT of $\mathbf{x}$ if $TR_{\mathbf{x}}T^T$ is a diagonal matrix. The diagonal matrix $TR_{\mathbf{x}}T^T$ is the covariance of $\mathbf{y}=T\mathbf{x}$; thus, a KLT gives uncorrelated transform coefficients. KLT is the most commonly used name for these transforms in signal processing, communication, and information theory, recognizing the works [24] and [26]; among the other names are Hotelling transforms [19] and principal component transforms.

A KLT exists for any source because covariance matrices are symmetric, and symmetric matrices are orthogonally diagonalizable; the diagonal elements of $TR_{\mathbf{x}}T^T$ are the eigenvalues of $R_{\mathbf{x}}$. KLT's are not unique: any row of T can be multiplied by ±1 without changing $TR_{\mathbf{x}}T^T$, and permuting the rows leaves $TR_{\mathbf{x}}T^T$ diagonal. If the eigenvalues of $R_{\mathbf{x}}$ are not distinct, there is additional freedom in choosing a KLT. An example of a KLT is given in Box 5.

For Gaussian sources, KLTs align the partitioning with the axes of the source pdf, as shown in Fig. 2(c). It appears that the forward and inverse transforms are rotations, though actually the symmetry of the source density obscures possible reflections.

The Easiest Transform Optimization

Consider a jointly Gaussian source, and assume U and T are orthogonal and $U=T^{-1}$. The Gaussian assumption is important because any linear combination of jointly Gaussian random variables is Gaussian. Thus, any analysis transform gives Gaussian transform coefficients. Then, since the transform coefficients have the same normalized density, for any reasonable set of quantizers, (3) holds with a single function $g(R)$ describing all of the transform coefficients. Orthogonality is important because orthogonal transforms preserve Euclidean lengths, which gives $d(x,\hat{x})=d(y,\hat{y})$.

With these assumptions, for any rate and bit allocation a KLT is an optimal transform.

Box 6
Optimizing the Synthesis Transform *U*

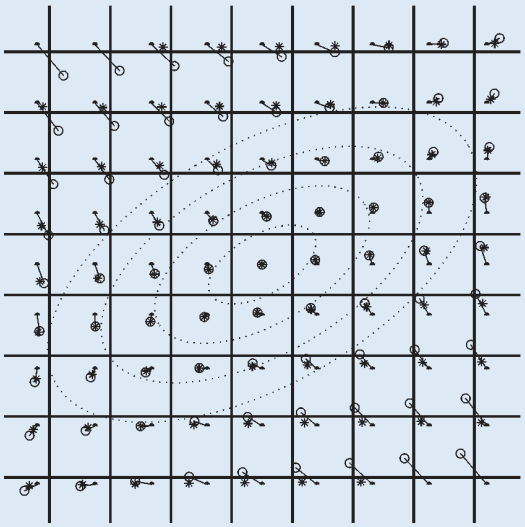
From a glance at Fig. 1, it is natural to assume that the synthesis transform should be $U=T^{-1}$. Theorem 3 provides sufficient conditions for this to be optimal, but these are not always satisfied.

As an example, consider a two-dimensional, zero-mean, jointly Gaussian source with

$$R_{\mathbf{x}}=\frac{1}{16}\begin{bmatrix}13 & 3\sqrt{3}\\ 3\sqrt{3} & 7\end{bmatrix}.$$

Level curves of the pdf are shown with dotted ellipses. Rotating by 30° would give independent transform coefficients, but suppose no transform (or the identity transform) is used. The conditions of Theorem 3 are not satisfied: Even if the β_i's (scalar quantizer decoders) are optimal, a synthesis transform other than $T^{-1}=I$ may be optimal.

With α_i's (scalar quantizer encoders) that are uniform with step size $\Delta=1$, the partitioning is as shown.



The outputs of the scalar quantizers (•) are not at the centers of the cells, but they still form a separable codebook; the cell centroids (*) are not separable. The optimal synthesis transform

$$U=\begin{bmatrix}0.970 & 0.089\\ 0.083 & 0.944\end{bmatrix}$$

gives a better codebook (◦) and reduces the distortion by 4.2%.

Theorem 1 ([11], [15]): Consider a transform coder with orthogonal analysis transform T and synthesis transform $U=T^{-1}=T^T$. Suppose there is a single function g to describe the quantization of each transform coefficient through

$$E\left[(\mathbf{y}_i-\hat{\mathbf{y}}_i)^2\right]=\sigma_i^2\, g(R_i),\quad i=1,2,\ldots,N,$$

where σ_i^2 is the variance of $\mathbf{y}_i$ and R_i is the rate allocated to y_i. Then for any bit allocation $(R_1, R_2, \ldots, R_N)$ there is a KLT that minimizes the distortion. In the typical case where g is nonincreasing, a KLT that gives $(\sigma_1^2, \sigma_2^2, \ldots, \sigma_N^2)$ sorted in the same order as the bit allocation minimizes the distortion.

Since it holds for any bit allocation and many families of quantizers, Theorem 1 is stronger than several earlier transform optimization results. In particular, it subsumes the low-rate results of Lyons [28] and the high-rate results that are reviewed presently.

Recall that with a high average rate of R bits per component and quantizer performance described by (9), the average distortion with optimal bit allocation is given by (10). With Gaussian transform coefficients that are optimally quantized, the distortion simplifies to

$$D = c\left(\prod_{i=1}^{N} \sigma_i^2\right)^{1/N} 2^{-2R}, \tag{11}$$

where $c = \pi e/6$ for entropy-constrained quantization or $c = \sqrt{3}\pi/2$ for fixed-rate quantization. The choice of an orthogonal transform is thus guided by minimizing the geometric mean of the transform coefficient variances.

Theorem 2: The distortion given by (11) is minimized over all orthogonal transforms by any KLT.

Proof: Applying Hadamard's Inequality [18, 7.8.1] to R_y gives

$$(\det T)(\det R_x)(\det T^T) = \det R_y \le \prod_{i=1}^{N} \sigma_i^2 .$$

Since $\det T = 1$, the left-hand side of this inequality is invariant to the choice of T. Equality is achieved when a KLT is used. Thus KLTs minimize the distortion.

Equation (11) can be used to define a figure of merit called the coding gain. The coding gain of a transform is a function of its variance vector, $(\sigma_1^2, \sigma_2^2, \ldots, \sigma_N^2)$, and the variance vector without a transform, $\mathrm{diag}(R_x)$:

$$\text{coding gain} = \frac{\left(\prod_{i=1}^{N} (R_x)_{ii}\right)^{1/N}}{\left(\prod_{i=1}^{N} \sigma_i^2\right)^{1/N}}.$$

The coding gain is the factor by which the distortion is reduced because of the transform, assuming high rate and optimal bit allocation. The foregoing discussion shows that KLTs maximize coding gain. Related measures are the variance distribution, maximum reducible bits, and energy packing efficiency or energy compaction. All of these are optimized by KLTs [33].

More General Results

The results of the previous section are straightforward, and Theorem 2 is well known. However, KLTs are not always optimal. With some sources, there are nonorthogonal transforms that perform better than any orthogonal transform. And, depending on the quantization, $U = T^{-1}$ is not always optimal, even for Gaussian sources. This section provides results that apply

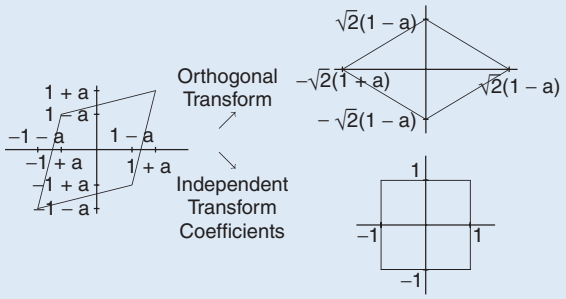
Box 7
Orthogonality versus Independence

Two qualitative intuitions arise from high resolution transform coding theory: a) try to have independent transform coefficients; and b) use orthogonal transforms. Sometimes you can only have one or the other.

Suppose the source $\mathbf{x}$ is uniformly distributed on the rhombus shown below left. For $a \neq 0$, there is no orthogonal transform that gives independent transform coefficients. The best orthogonal transform is a 45° rotation (right top) and the nonorthogonal transform

$$T = \frac{1}{1-a^2}\begin{bmatrix} 1 & -a \\ -a & 1 \end{bmatrix}$$

gives independent transform coefficients (right bottom).



By computing differential entropies of the transform coefficients and using (6), one can show that the optimal high-rate performance using the orthogonal transform is

$$D = \frac{1}{6}e(1-a^2)2^{-2R}.$$

Computing the effect of nonsquare cells as in Box 4, one can show that the performance with the nonorthogonal transform is

$$D = \frac{1}{3}(1+a^2)2^{-2R}.$$

For $a\sqrt{(e-2)/(e+2)}$, it is better to use the nonorthogonal transform; otherwise, it is better to use the KLT. The optimal transform is neither orthogonal nor produces independent transform coefficients.

The situation is yet more complicated if we do not use high resolution theory. For $a = 0$ and very low rates, the rotated coordinates are better than the original coordinates even though the components are independent in the original coordinates [14].

without the presumption of Gaussianity or orthogonality and examples to show the limitations of these results.

The Synthesis Transform U

Instead of assuming the decoder structure shown in the bottom of Fig. 1, let us consider for a moment the best way to decode given only the encoding structure of a transform code. The analysis transform followed by quantization induces some partition of $\mathbb{R}^N$, and the best decoding is to associate with each partition cell its centroid. Generally, this decoder cannot be realized with a linear transform applied to $\hat{y}$. For one thing, some scalar quantizer decoder β_i could be designed in a plainly wrong way; then it would take an extraordinary (nonlinear) effort to fix the estimates.

The difficulty is actually more dramatic because even if the β_is are optimal, the synthesis transform T^{-1} applied to $\hat{y}$ will generally not give optimal estimates. In fact, unless the transform coefficients are independent, there may be a linear transform better suited to the reconstruction than T^{-1}.

Theorem 3 ([14]): In a transform coder with invertible analysis transform T, suppose the transform coefficients are independent. If the component quantizers reconstruct to centroids, then $U = T^{-1}$ gives centroid reconstructions for the partition induced by the encoder. As a further consequence, T^{-1} is the optimal synthesis transform.

Examples where the lack of independence of transform coefficients or the absence of optimal scalar decoding makes T^{-1} a suboptimal synthesis transform are given in [14]. One of these examples is presented in Box 6.

The Analysis Transform T

Now consider the optimization of T under the assumption that $U = T^{-1}$. The first result is a counterpart to Theorem 2. Instead of requiring orthogonal transforms and finding uncorrelated transform coefficients to be best, it requires independent transform coefficients and finds orthogonal basis vectors to be best. It does not require a Gaussian source; however, it is only for Gaussian sources that R_y being diagonal implies that the transform coefficients are independent.

Theorem 4 ([14]): Consider a transform coder in which analysis transform T produces independent transform coefficients, the synthesis transform is T^{-1}, and the component quantizers reconstruct to their respective centroids. To minimize the MSE distortion, it is sufficient to consider transforms with orthogonal rows, i.e., T such that TT^T is a diagonal matrix.

The scaling of a row of T is generally irrelevant because it can be completely absorbed in the quantizer for the corresponding transform coefficient. Thus, Theorem 4 implies furthermore that it suffices to consider orthogonal transforms that produce independent transform coefficients. Together with Theorem 1, it still falls short of showing that a KLT is necessarily an optimal transform—even for a Gaussian source.

Heuristically, independence of transform coefficients seems desirable because otherwise dependencies that would make it easier to code the source are "wasted." Orthogonality is beneficial for having good partition cell shapes. A firm result along these lines requires high resolution analysis.

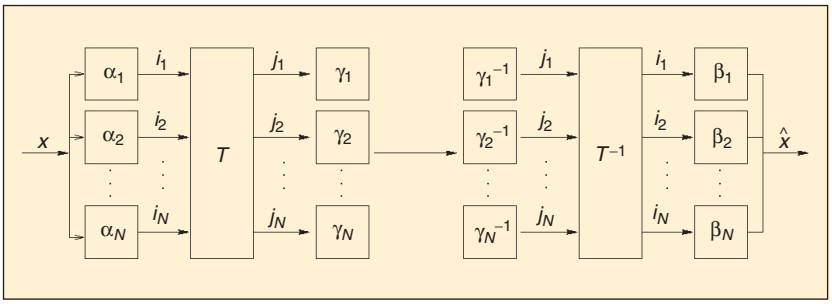
Theorem 5 ([14]): Consider a high-rate transform coding system employing entropy-constrained uniform quantization. A transform with orthogonal rows that produces independent transform coefficients is optimal, when such a transform exists. Furthermore, the norm of the *i*th row divided by the *i*th quantizer step size is optimally a constant. Thus, normalizing the rows to have an orthogonal transform and using equal quantizer step sizes is optimal.

For Gaussian sources there is always an orthogonal transform that produces independent transform coefficients—the KLT. For some other sources there are only nonorthogonal transforms that give independent transform coefficients, but for most sources there is no linear transform that does so. Through an example, Box 7 considers whether orthogonality or independent transform coefficients is more important, when you have to choose between the two. There is no unequivocal answer.

More examples that explore the limitations of Theorem 5 are given in [14]. In particular, it is demonstrated that even an orthogonal transform that produces independent transform coefficients is not necessarily optimal when the rate is low.

Departures from the Standard Model

Practical transform coders differ from the standard model in many ways. This is discussed in detail in two articles of this *Magazine* [37], [38]. One particular change has significant implications for the relevance of the conventional analysis and has lead to new theoretical developments: Transform coefficients are often not entropy coded independently.



▲ *3. An alternative transform coding structure introduced in [12]. The transform T operates on a vector of discrete quantizer indices instead of on the continuous-valued source vector, as in the standard model. Scalar entropy coding is explicitly indicated. For a Gaussian source, a further simplification with* $\gamma_1 = \gamma_2 = \cdots = \gamma_N$ *can be made with no loss in performance.*

Allowing transform coefficients to be entropy coded together, as it is drawn in Fig. 1, throws the theory into disarray. Most significantly, it eliminates the incentive to have independent transform coefficients. As for the particulars of the theory, it also destroys the concept of bit allocation because bits are shared among transform coefficients.

The status of the conventional theory is not quite so dire, however, because the complexity of unconstrained joint entropy coding of the transform coefficients is prohibitive. Assuming an alphabet size of K for each scalar component, an entropy code for vectors of length N has K^N codewords. The problems of storing this codebook and searching for desired entries prevent large values of N from being feasible. Entropy coding without explicit storage of the codewords—as, for example, in arithmetic coding—is also difficult because of the number of symbol probabilities that must be known or estimated.

Analogous to constrained lossy source codes, joint entropy codes for transform coefficients are usually constrained in some way. In the original JPEG standard [32], the joint coding is limited to transform coefficients with quantized values equal to zero. This type of joint coding does not eliminate the optimality of the KLT (for a Gaussian source); in fact, it makes it even more important for a transform to give a large fraction of coefficients with small magnitude. The empirical fact that wavelet transforms have this property for natural images (more abstractly, for piecewise smooth functions) is a key to their current popularity. Another article in this issue [39] discusses this and related developments at length.

Returning to the choice of a transform assuming joint entropy coding, the high rate case gives an interesting result: Quantizing in any orthogonal analysis basis and using uniform quantizers with equal step sizes is optimal. All the meaningful work is done by the entropy coder. Given that the transform has no effect on the performance, it can be eliminated. There is still room for improvement, however.

Producing transform coefficients that are independent allows for the use of scalar entropy codes, with the attendant reduction in complexity, without any loss in performance. A transform applied to the quantizer outputs, as shown in Fig. 3, can be used to achieve or approximate this. For Gaussian sources, it is even possible to design the transform so that the quantized transform coefficients have approximately the same distribution, in addition to being approximately independent. Then the same scalar entropy code can be applied to each transform coefficient [12]. Some of the lossless codes used in practice include transforms, but they are not optimized for a single scalar entropy code.

Historical Notes

Transform coding was invented as a method for conserving bandwidth in the transmission of signals output by the analysis unit of a ten-channel vocoder ("voice coder") [4]. These correlated, continuous-time, continuous-amplitude signals represented estimates, local in time, of the power in ten contiguous frequency bands. By adding modulated versions of these power signals, the synthesis unit resynthesized speech. The vocoder was publicized through the demonstration of a related device, called the Voder, at the 1939 World's Fair.

Kramer and Mathews [25] showed that the total bandwidth necessary to transmit the signals with a prescribed fidelity can be reduced by transmitting an appropriate set of linear combinations of the signals instead of the signals themselves. Assuming Gaussian signals, KLTs are optimal for this application.

The technique of Kramer and Mathews is not source coding because it does not involve discretization. Thus, one could ascribe a later birth to transform coding. Huang and Schultheiss [20], [21] introduced the structure shown in the bottom of Fig. 1 that we have referred to as the standard model. They studied the coding of Gaussian sources while assuming independent transform coefficients and optimal fixed-rate scalar quantization. First they showed that $U = T^{-1}$ is optimal and then that T should have orthogonal rows. These results are subsumed by Theorems 3 and 4. They also obtained high-rate bit allocation results.

Summary

The theory of source coding tells us that performance is best when large blocks of data are used. But this same theory suggests codes that are too difficult to use—because of storage, running time, or both—if the block length is large. Transform codes alleviate this dilemma. They perform well, though not optimally, but are simple enough to apply with very large block lengths.

Transform codes are easy to implement because of a divide and conquer strategy; the transform exploits dependencies in the data so that the quantization and entropy coding can be simple. For jointly Gaussian sources, this works perfectly: the transform can produce independent transform coefficients, and then little is lost by using scalar quantization and scalar entropy coding.

As with source coding generally, it is hard to make use of the theory of transform coding with real-world signals. Nevertheless, the principles of transform coding certainly do apply, as evidenced by the dominance of transform codes in audio, image, and video compression.

Acknowledgments

Comments from M. Effros, A.K. Fletcher, anonymous reviewers, and the guest editors are gratefully acknowledged.

Vivek K Goyal received the B.S. in mathematics and the B.S.E. in electrical engineering (both with highest distinction), in 1993 from the University of Iowa, Iowa City. He received the M.S. and Ph.D. degrees in electrical engineering from the University of California, Berkeley, in 1995 and 1998, respectively. He was a Research Assis-

tant in the Laboratoire de Communications Audiovisuelles at École Polytechnique Fédérale de Lausanne, Switzerland, in 1996. He was with Lucent Technologies' Bell Laboratories as an intern in 1997 and again as a Member of Technical Staff from 1998 to 2001. He is currently a Senior Research Engineer for Digital Fountain, Inc., Fremont, CA. His research interests include source coding theory, quantization theory, and practical robust compression. He is a member of the IEEE, Phi Beta Kappa, Tau Beta Pi, Sigma Xi, Eta Kappa Nu and SIAM. In 1998 he received the Eliahu Jury Award of the University of California, Berkeley.

References

[1] T. Berger, *Rate Distortion Theory*. Englewood Cliffs, NJ: Prentice-Hall, 1971.

[2] J.H. Conway and N.J.A. Sloane, *Sphere Packings, Lattices and Groups*, 3rd ed. (*Grundlehren der mathematischen Wissenschaften*, vol. 290). New York: Springer-Verlag, 1998.

[3] T.M. Cover and J.A. Thomas, *Elements of Information Theory*. New York: Wiley, 1991.

[4] H.W. Dudley, "The vocoder," *Bell Lab. Rec.*, vol. 18, pp. 122-126, Dec. 1939.

[5] M. Effros, "Optimal modeling for complex system design," *IEEE Signal Processing Mag.*, vol. 15, pp. 51-73, Nov. 1998.

[6] N. Farvardin and J.W. Modestino, "Optimum quantizer performance for a class of non-Gaussian memoryless sources," *IEEE Trans. Inform. Theory*, vol. IT-30, pp. 485-497, May 1984.

[7] A. Gersho, "Asymptotically optimal block quantization" *IEEE Trans. Inform. Theory*, vol. IT-25, pp. 373-380, July 1979.

[8] A. Gersho and R.M. Gray, *Vector Quantization and Signal Compression*. Norwell, MA: Kluwer, 1992.

[9] H. Gish and J.P. Pierce, "Asymptotically efficient quantizing," *IEEE Trans. Inform. Theory*, vol. IT-14, pp. 676-683, Sept.1968.

[10] T.J. Goblick, Jr. and J.L. Holsinger, "Analog source digitization: A comparison of theory and practice," *IEEE Trans. Inform. Theory*, vol. IT-13, pp. 323-326, Apr. 1967.

[11] V.K. Goyal, "High-rate transform coding: How high is high, and does it matter?," in *Proc. IEEE Int. Symp. Inform. Theory*, Sorrento, Italy, June 2000, p. 207.

[12] V.K. Goyal, "Transform coding with integer-to-integer transforms," *IEEE Trans. Inform. Theory*, vol. 46, pp. 465-473, Mar. 2000.

[13] V.K. Goyal, "Multiple description coding: Compression meets the network," *IEEE Signal Processing Mag.*, vol. 18, pp. 74-93, Sept. 2001.

[14] V.K. Goyal. *Single and Multiple Description Transform Coding with Bases and Frames*. Philadelphia, PA: SIAM, 2001.

[15] V.K. Goyal, J. Zhuang, and M. Vetterli, "Transform coding with backward adaptive updates," *IEEE Trans. Inform. Theory*, vol. 46, pp. 1623-1633, July 2000.

[16] R.M. Gray and D.L. Neuhoff, "Quantization," *IEEE Trans. Inform. Theory*, vol. 44, pp. 2325-2383, Oct. 1998.

[17] U. Grenander and G. Szegö, *Toeplitz Forms and Their Applications*. Berkeley, CA: Univ. of California Press, 1958.

[18] R.A. Horn and C.R. Johnson, *Matrix Analysis*. Cambridge, U.K.: Cambridge Univ. Press, 1985. (Reprinted with corrections 1987.)

[19] H. Hotelling, "Analysis of a complex of statistical variables into principal components," *J. Educ. Psychology*, vol. 24, pp. 417-441, 498-520, 1933.

[20] J.-Y. Huang, "Quantization of correlated random variables," Ph.D. dissertation, Yale Univ., New Haven, CT, 1962.

[21] J.J.Y. Huang and P.M. Schultheiss, "Block quantization of correlated Gaussian random variables," *IEEE Trans. Commun. Syst.*, vol. 11, pp. 289-296, Sept. 1963.

[22] D.A. Huffman, "A method for the construction of minimum redundancy codes," *Proc. IRE*, vol. 40, pp.1098-1101, Sept. 1952.

[23] A.K. Jain, "A sinusoidal family of unitary transforms," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. PAMI-1, pp. 356-365, Oct. 1979.

[24] K. Karhunen, "Über lineare methoden in der Wahrscheinlichkeitsrechnung," *Ann. Acad. Sci. Fenn., Ser. A.1.: Math.-Phys.*, vol. 37, pp. 3-79, 1947.

[25] H.P. Kramer and M.V. Mathews, "A linear coding for transmitting a set of correlated signals," *IRE Trans. Inform. Theory*, vol. 23, no. 3, pp. 41-46, Sept. 1956.

[26] M. Loève, "Fonctions aleatoires de seconde ordre," in *Processus Stochastiques et Mouvement Brownien*, P. Levy, Ed . Paris, France: Gauthier-Villars, 1948.

[27] D.G. Luenberger, *Optimization by Vector Space Methods*. New York: Wiley, 1969.

[28] D.F. Lyons III, "Fundamental limits of low-rate transform codes," Ph.D. dissertation, Univ. Michigan, Ann Arbor, MI, 1992.

[29] N. Moayeri and D.L. Neuhoff, "Time-memory tradeoffs in vector quantizer codebook searching based on decision trees," *IEEE Trans. Speech Audio Processing*, vol. 2, pp. 490-506, Oct. 1994.

[30] A. Ortega and K. Ramchandran, "Rate-distortion methods for image and video compression," *IEEE Signal Processing Mag.*, vol. 15, pp. 23-50, Nov. 1998.

[31] A. Papoulis, *Probability, Random Variables, and Stochastic Processes,* 3rd ed. New York: McGraw-Hill, 1991.

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

[33] K.R. Rao and P. Yip, *Discrete Cosine Transform: Algorithms, Advantages, Applications*. San Diego, CA: Academic, 1990.

[34] W.D. Ray and R.M. Driver, "Further decomposition of the Karhunen-Loève series representation of a stationary random process," *IEEE Trans. Inform. Theory*, vol. IT-16, pp. 663-668, Nov. 1970.

[35] A. Segall, "Bit allocation and encoding for vector sources," *IEEE Trans. Inform. Theory*, vol. IT-22, pp. 162-169, Mar. 1976.

[36] Y. Shoham and A. Gersho, "Efficient bit allocation for an arbitrary set of quantizers," *IEEE Trans. Acoust. Speech Signal Processing*, vol. 36, pp. 1445-1453, Sept. 1988.

[37] A. Skodras, C. Christopoulos, and T. Ebrahimi, "The JPEG 2000 still image compression standard," *IEEE Signal Processing Mag.*, vol. 18, pp. 36-58, Sept. 2001.

[38] B.E. Usevitch, "A tutorial on modern lossy wavelet image compression: Foundations of JPEG 2000," *IEEE Signal Processing Mag.*, vol. 18, pp. 22-35, Sept. 2001.

[39] M. Vetterli, "Wavelets, approximation, and compression," *IEEE Signal Processing Mag.*, vol. 18, pp. 59-73, Sept. 2001.

[40] R.C. Wood, "On optimum quantization," *IEEE Trans. Inform. Theory*, vol. IT-15, pp. 248-252, Mar. 1969.