Chapter 10

Image Coding

Contents

(Related to Ch. 10 of Lim.)

Introduction

Entire books devoted to this problem, *e.g.*, [1].

Coding goal: represent an image with "as few bits as possible" while preserving the "quality" (meaning visual quality usually) required for the particular application.

Course goal: familiarity with the *elements* of image coding systems at a level suitable for reading current literature.

Outline

- quantization (scalar, vector)
- codeword assignment
- waveform coding
- transform coding
- image model coding
- interframe image coding, color, channel errors, ...

Relevant MATLAB commands: `dct2`, `idct2`, `dctdemo`, `blkproc`, and many more.

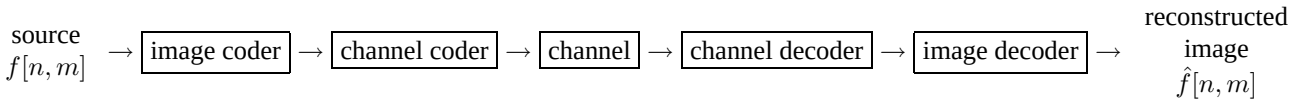
Overview

- Image coding methods fall into two categories
  - **lossless** - the recovered image exactly matches the original image
  - **lossy** aka "information preserving" - the recovered image does not exactly match the original image, but the "essential information" is retained.

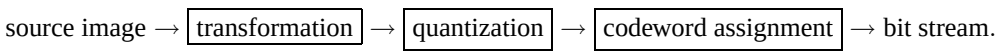
  Unfortunately, there is no widely accepted metric for "essential information," so the usual measure of image fidelity is **peak signal-to-noise ratio** (**PSNR**):

$$\text{PSNR} = 10 \log_{10} \frac{255^2}{\sum_{n=0}^{N-1} \sum_{m=0}^{M-1} (f[n,m] - \hat{f}[n,m])^2}.$$

- Typical image coding system



  - If the bits are stored on a reliable medium like hard disk, then channel coding is unnecessary.
  - If the bits are transmitted over a noisy communication channel, like a telephone line, then channel coding may be essential.
  - A good image coder will produce what appears to the channel coder to be a random bit sequence. In this type of scenario, the channel coding method can be designed independently of the source coding method. Channel coding is a separate topic considered in EECS 650. In this course we will focus on the image coding problem, which is a special case of the more general source coding problem covered in EECS 651.
- The image coder typically has the following form:



  The transformation extracts image information prior to quantization. The transformation usually takes one of three forms.
  - Waveform Coder: The intensity information itself is coded.
  - Transform Coder: Some integral transform of the image, such as the DCT, is computed.
  - Model Coder: A model of the image is formulated and parameters of the model are estimated for a particular image (this is analogous to LPC coding in speech processing). The model parameters are passed to the quantizer.
- The steps of transformation, quantization, and codeword assignment are often closely linked.
- However, the general principles governing quantization and codeword assignments are separable from the choice of the image transformation. We will study quantization and codeword assignment procedures independently before looking at different transformation procedures and complete image coding systems.

10.1

Quantization

The process of representing a real-valued image $f[n, m]$ (or some other array of real numbers such as transform coefficients or model parameters) by a finite set of bits is called **quantization**.

- If we quantize each value independently, then the procedure is called **scalar quantization**.
- If we group together multiple values and quantize them simultaneously, it is called **vector quantization** (as discussed later).

10.1.1

Scalar quantization

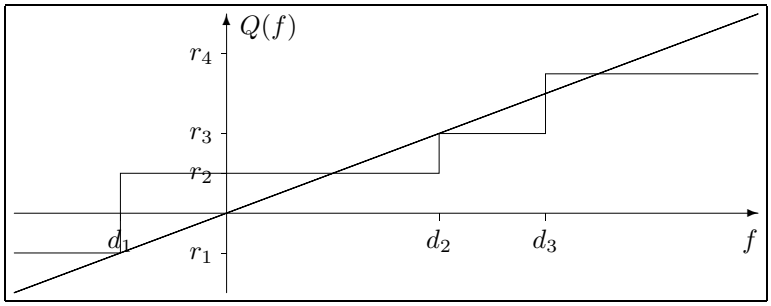
Let f denote a continuous scalar quantity. We wish to represent f with a finite number of bits (m) resulting in one of $L = 2^m$ possible levels. Assigning a specific f to one of L levels is called **amplitude quantization**.

Let $\hat{f}$ denote the quantized value of f. We consider deterministic quantization strategies, where

$$\hat{f} = Q(f) \text{ and } Q : \mathbb{R} \to \{r_1, r_2, \ldots, r_L\}.$$

We call Q the **quantization operator**. The r_i's are called the **reconstruction levels**.

Example. The following figure shows a case where $L = 4$.



Any such operation will always involve **quantization error** or **quantization noise**:

$$\hat{f} = Q(f) = f + e_Q \text{ where } e_Q \triangleq \hat{f} - f \tag{10.2}$$

If we choose the r_i's properly, then as the number of levels L increases, "on average" the quantization error should decrease.

Key questions:

- What is the "best" choice of the r_i's?
- How should we choose $Q(f)$, *i.e.*, how do we choose the regions $\mathcal{C}_i = \{f : Q(f) = r_i\}$..
- How many levels L are required for a given "accuracy"?

In 1D one can argue easily that the $\mathcal{C}_i$ regions should be contiguous intervals, so we generally form Q as follows:

$$Q(f) = \begin{cases} r_1, & d_0 < f \le d_1 \\ \vdots & \vdots \\ r_i, & d_{i-i} < f \le d_i \\ \vdots & \vdots \\ r_L, & d_{L-1} < f \le d_L, \end{cases}$$

where the $L + 1$ values $\{d_i\}$ are called **decision boundaries**. (It is possible to have $d_0 = -\infty$ and $d_L = \infty$.)

The simplest choice is *equally spaced* decision boundaries and to choose r_i to be the **midpoint** of each interval. This simple approach (which is rarely optimal in terms of quantization error) is called **uniform scalar quantization**:

$$d_i - d_{i-1} = \Delta, \qquad r_i = \frac{d_i + d_{i-1}}{2}, \quad i = 1, \ldots, L.$$

Although the form of (10.2) might suggest otherwise, in general the quantization error e_Q will be *signal dependent*. However, whether the quantization error is *correlated* or not with the signal depends on the specifics of the distribution of f and the quantization scheme.

Design of reconstruction levels and decision boundaries

As a general rule, to minimize quantization error, one would like to assign more reconstruction levels (the r_i's) near the values of f that are more likely. To formulate objective criteria for choosing the "best" r_i's and d_i's, we must choose a method for quantifying the quantization error.

The first ingredient is a **distortion measure** $d(f, \hat{f})$. Typical choices are $|f - \hat{f}|$ or $|f - \hat{f}|^2$. To obtain a scalar cost function, we must somehow "average" over the possible values of f. To proceed, we *assume* that f is a random variable with pdf $\mathsf{p}(f)$. We then define the **average distortion** to be

$$D = \mathsf{E}\left[d(f, \hat{f})\right] = \int d(f, Q(f)) \, \mathsf{p}(f) \, \mathrm{d}f \, .$$

If we choose the quadratic distortion measure $d(f, \hat{f}) = (f - \hat{f})^2$, then minimizing the average distortion is equivalent to a MMSE approach:

$$D = \int |f - Q(f)|^2 \, \mathsf{p}(f) \, \mathrm{d}f = \sum_{i=1}^{L} \int_{d_{i-1}}^{d_i} (f - r_i)^2 \, \mathsf{p}(f) \, \mathrm{d}f \, .$$

To minimize this average distortion, we can equate to zero the partial derivatives of D with respect to the design parameters (the r_i's and the d_i's). Specifically:

$$0 = \frac{\partial}{\partial r_i} D = \int_{d_{i-1}}^{d_i} 2(f - r_i) \, \mathsf{p}(f) \, \mathrm{d}f = 2 \int_{d_{i-1}}^{d_i} f \, \mathsf{p}(f) \, \mathrm{d}f - 2 r_i \int_{d_{i-1}}^{d_i} \mathsf{p}(f) \, \mathrm{d}f,$$

and by **Leibniz's rule**:

$$0 = \frac{\partial}{\partial d_i} D = (d_i - r_i)^2 \, \mathsf{p}(d_i) - (d_{i+1} - r_{i+1})^2 \, \mathsf{p}(d_i), \quad i = 1, \ldots, L-1.$$

These lead to the following set of equations:

$$\begin{aligned}
r_i &= \frac{\int_{d_{i-1}}^{d_i} f \, \mathsf{p}(f) \, \mathrm{d}f}{\int_{d_{i-1}}^{d_i} \mathsf{p}(f) \, \mathrm{d}f}, \quad i = 1, \ldots, L \text{ (weighted centroid)} \\
d_i &= \frac{r_i + r_{i+1}}{2}, \quad i = 1, \ldots, L-1 \text{ (midpoint)} \\
d_0 &= \min \{ f : \mathsf{p}(f) > 0 \} \\
d_L &= \max \{ f : \mathsf{p}(f) > 0 \} \, .
\end{aligned} \tag{10.9}$$

Unfortunately, these equations are usually nonlinear and must be solved numerically.

Example. If f is uniformly distributed over $[a, b]$, then the MMSE decision boundaries and reconstruction levels are given by $d_i = a + (b - a)\frac{i}{L}$ and $r_i = d_i - \frac{b-a}{2L}$ respectively, which corresponds to **uniform scalar quantization**.

Exercise. Show that the MSE in the above example is σ_f^2 / L^2. Naturally, as L increases, the average distortion decreases.

Example. Suppose we apply uniform scalar quantization to variables having the exponential distribution $\mathsf{p}(f) = \mathrm{e}^{-f}\, u(f)$, using levels $r_i = ia$, $i = 1, \ldots, L$. What should a be?
The distortion is (***(Picture)***):

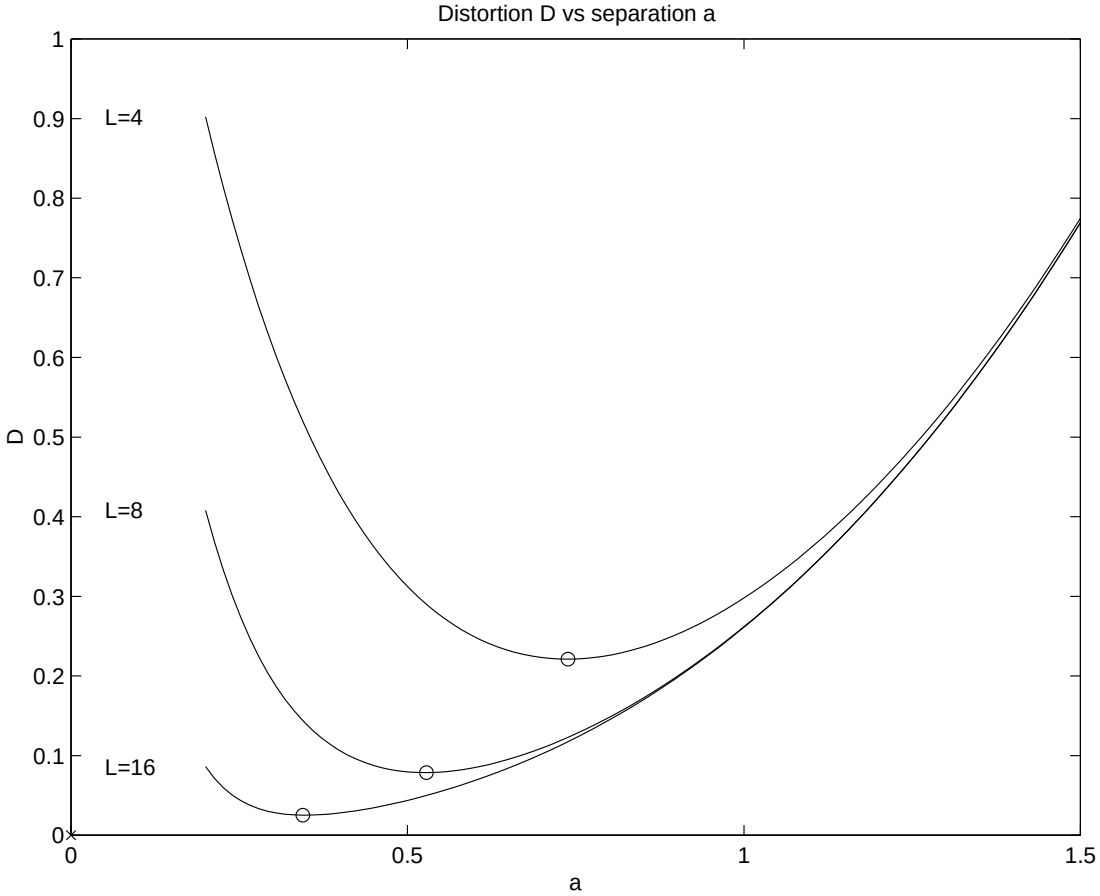
$$D = \int_0^{\frac{3}{2}a} (f-a)^2\, \mathrm{e}^{-f}\, \mathrm{d}f + \sum_{i=2}^{L-1} \int_{(i-1/2)a}^{(i+1/2)a} (f-ia)^2\, \mathrm{e}^{-f}\, \mathrm{d}f + \int_{(L-1/2)a}^{\infty} (f-La)^2\, \mathrm{e}^{-f}\, \mathrm{d}f\,.$$

Using `int` and `diff` from MATLAB's symbolic toolbox yields:

$$\frac{\partial}{\partial a} D = \mathrm{e}^{-a/2}\,(a^2/8 - a + 2) - 2 + 2a + \sum_{i=1}^{L-1} \left[(a^2/4 + a + 2) a\, \mathrm{e}^{-(i+1/2)a} - (a^2/4 - a + 2) a\, \mathrm{e}^{-(i-1/2)a} \right]$$

$$+ \left[a/2 - 1 - (L-1/2)(a^2/4 - a + 2) \right] \mathrm{e}^{-(L-1/2)a}$$

It is unlikely that we will find an analytical form for the minimizer. We can use `fmin` or `fsolve` to minimize D numerically.



- Minimum distortion D decreases as L increases, as expected.
- Optimal a does not halve as L doubles, unlike for uniform distribution.
  (Therefore redesign of quantizer is needed for each L of interest.)

As a consequence of the central limit theorem, Gaussian random variables frequently arise in many problems (although not necessarily in image coding problems...). The book compares the MSE of uniform scalar quantization and the MSE of optimal (MMSE) quantization for a Gaussian random variable. On average, the MMSE approach saves about 1/2 bits (per coded value) for 7 bit accuracy (128 levels). (The savings are more dramatic with vector quantization.)

See [2] for a thorough analysis of uniform scalar quantization.

Companding (another scalar quantization method)

Since uniform scalar quantization is MSE-optimal for uniformly distributed random variables, a natural quantization approach is to first transform to a uniform random variable, and then apply uniform scalar quantization to the result.

Illustration of companding approach:

$$f \to \boxed{T} \to g \to \boxed{\text{Uniform quantizer}} \to \hat{g} \to \boxed{T^{-1}} \to \hat{f}.$$

The two key questions are the following.
- What should the companding function $T(f)$ be?
- Is the resulting system optimal?

We have seen previously that if random variable X has cdf $F_X(x)$, then $Y = F_X(X)$ has a uniform distribution. Thus

$$g = \int_{-\infty}^{f} \mathsf{p}(f') \, df' \triangleq T(f)$$

is uniformly distributed over $[0, 1]$. So the appropriate companding function is just the CDF of f (which we might need to learn from training data).

Applying uniform scalar quantization to g will minimize

$$\mathsf{E}\big[(g - Q(g))^2\big] = \mathsf{E}\big[(T(f) - Q(T(f)))^2\big] \neq \mathsf{E}\big[(f - Q(f))^2\big],$$

so this **companding** approach is not MSE optimal in terms of f. However, the MMSE criterion is somewhat arbitrary anyway, and chosen primarily for its analytical convenience, so in practice minimizing $\mathsf{E}\big[(T(f) - Q(T(f)))^2\big]$ may produce reasonable coded images.

Example.
Suppose f has the pdf $\mathsf{p}(f) = 2/f^2$ for $f \in [1, 2]$, and we wish to assign L levels using companding. The required transformation for this random variable is

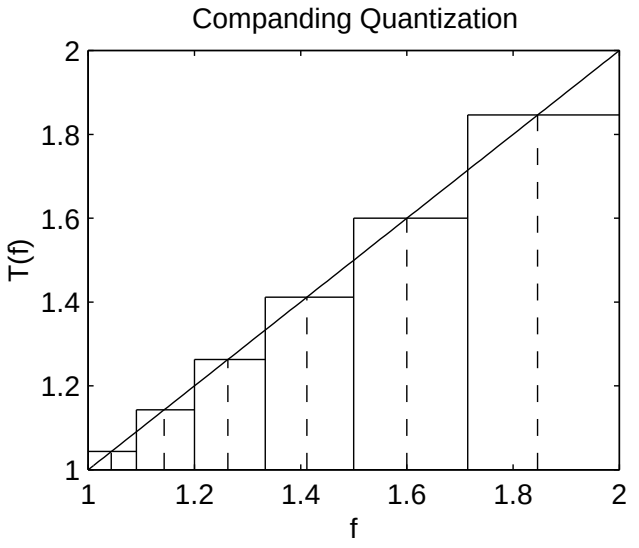
$$g = \int_1^f 2/x^2 \, \mathrm{d}x = 2 - 2/f.$$

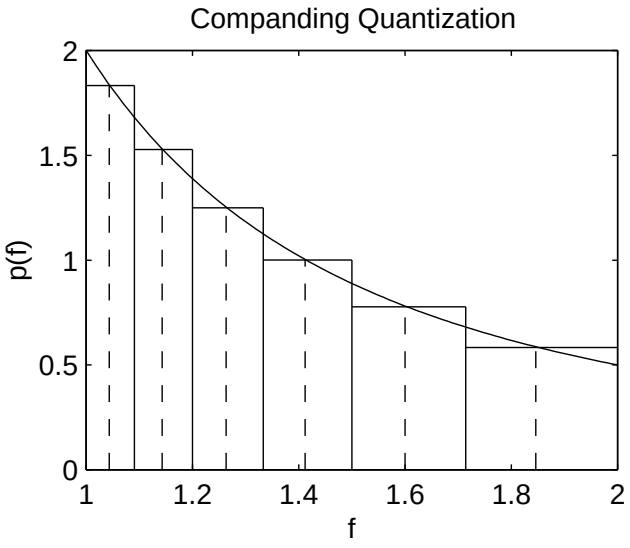
The associated reconstruction levels and decision boundaries (in terms of g) are

$$d_i^{(g)} = i/L, \qquad r_i^{(g)} = (i - 1/2)/L.$$

Since $2/f = 2 - g$ we have $f = \frac{2}{2-g}$, so the associated reconstruction levels and decision boundaries (in terms of f) are

$$d_i = \frac{2}{2 - i/L}, \qquad r_i = \frac{2}{2 - (i - 1/2)/L}.$$





High-rate scalar quantization

As shown by Gersho [3], for k-dimensional quantization with distortion using $\left\| \boldsymbol{f} - \hat{\boldsymbol{f}} \right\|_2^r$ the optimal centroid density is proportional to $\mathsf{p}^{k/(k+r)}$, for large rate, *i.e.*, large L.

In particular, in 1D ($k = 1$) using MSE ($r = 2$) distortion, the optimal density is $\mathsf{p}^{1/3}$.

Thus, for large L, instead of solving for the r_i values using the coupled system (10.9), one can choose the centroids r_i non-iteratively as $r_i = G^{-1}\left(\frac{i-1/2}{L}\right)$, where $G(t) = \int_{-\infty}^{t} \mathsf{p}^{1/3}(f)\,\mathrm{d}f$.

Some related (?) MATLAB commands.

QUANT Discretize values as multiples of a quantity.
QUANTIZE An example M-File S-function vectorized quantizer
lvq.m: learning vector quantization.
NEWLVQ Create a learning vector quantization network.
NNT2LVQ Update NNT 2.0 learning vector quantization network to NNT 3.0.
LEARNLVQ Learning vector quantization rule.
VMQUANT Variance Minimization Color Quantization.
VMQUANTC Variance Minimization Color Quantization (MEX file).

Bit allocation

If we choose $L_i = 2^{B_i}$ reconstruction levels for quantization the ith scalar f_i, then coding f_i alone would require B_i bits.

If we are coding N statistically identically distributed scalars f_i, then if we assign the same number bits B_i to each variable, the total number of bits will be $B = \sum_{i=1}^{N} B_i$. For waveform coding, using equal bits for each pixel may be reasonable. For transform coding, however, the different transform coefficients can have very different energies / variances (indeed the point of the transform is to compact most of the signal energy into a few coefficients). When the variances are nonuniform, we should also allocate the bits nonuniformly to the different coefficients.

This **bit allocation** problem is very important. If the pdf's of the f_i's have the same *form* but different variances, then an approximate solution to problem of allocating B total bits to N variables is given by

$$B_i = \frac{B}{N} + \frac{1}{2} \log_2 \frac{\sigma_i^2}{\left[\prod_{j=1}^{N} \sigma_j^2\right]^{1/N}}.$$

- If the σ_i values are all identical, then $B_i = B/N$, which is uniform allocation.
- Otherwise, *the higher variance values are allocated more bits.*

10.2

Vector quantization

Rather than quantization each scalar one at a time, it is often advantageous to group scalars into "blocks" and jointly quantize the block. This type of approach is called **vector quantization** (**VQ**).

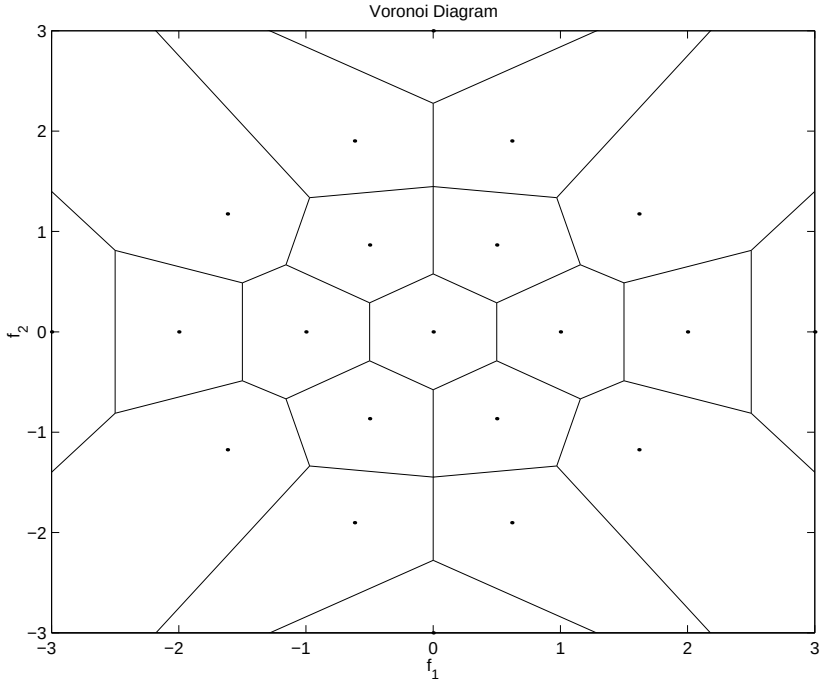
Why consider VQ? **??**

Let $\boldsymbol{f} = [f_1 \ \dots \ f_N]^T$ denote an N-dimensional vector consisting of N real-valued elements. In vector quantization, we map $\boldsymbol{f}$ to one of L "**reconstruction levels**" $\boldsymbol{r}_i$, each of which is also an N-dimensional vector:

$$\hat{\boldsymbol{f}} = Q(\boldsymbol{f}) = \begin{cases} \boldsymbol{r}_1, & \boldsymbol{f} \in \mathcal{C}_1 \\ \vdots & \vdots \\ \boldsymbol{r}_i, & \boldsymbol{f} \in \mathcal{C}_i \\ \vdots & \vdots \\ \boldsymbol{r}_L, & \boldsymbol{f} \in \mathcal{C}_L, \end{cases} \quad \text{where } Q : \mathbb{R}^N \to \{\boldsymbol{r}_1, \dots, \boldsymbol{r}_L\},$$

where the set $\mathcal{C}_i$ is called the ith **cell**.

Graphical display of Q is more complicated in VQ than in scalar quantization. The case $N = 2$ is about all that we can visualize easily. The MATLAB command `voronoi` computes the **Voronoi diagram** that, given the $\boldsymbol{r}_i$'s, displays them as dots in the plane, and displays the boundaries of the $\mathcal{C}_i$'s, assuming a **nearest neighbor** association rule.



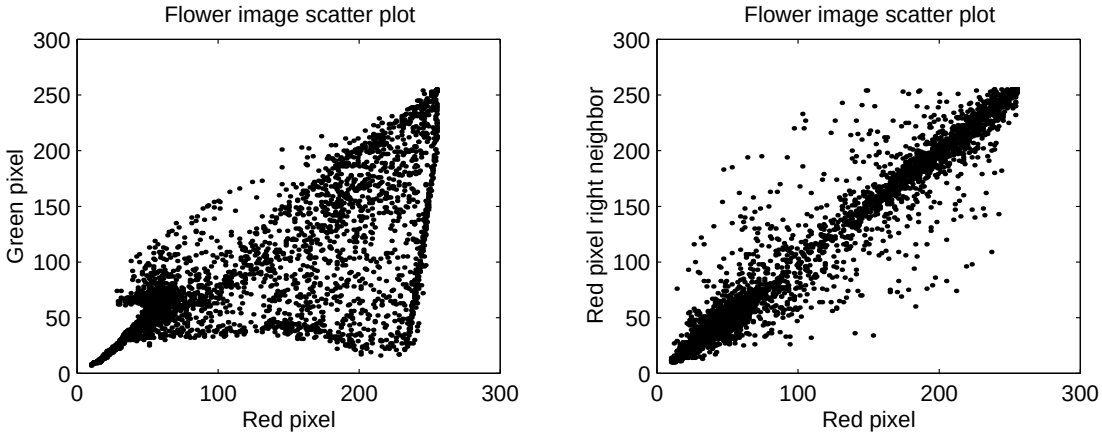
VQ Design

The same questions arise as in the scalar case, plus some additional ones.

- How many "levels" L are required to achieve a specified "accuracy."
- What should the $\boldsymbol{r}_i$'s be?
- How should we choose the $\mathcal{C}_i$'s?
- What should N be?
- When does VQ outperform scalar quantization?

The advantage of VQ is that it can exploit **statistical dependence** between elements of the vector $\boldsymbol{f}$ to achieve either
- Lower average distortion than scalar quantization with the same number of levels, or
- fewer levels (better compression) for the same average distortion.

The following values were extracted from the `flower.tif` image in MATLAB's image processing toolbox.



Scalar quantization of each variable corresponds to a rectilinear or "plaid" decomposition of the 2D plane. Such a decomposition would needlessly allocate cells in the upper left hand corner. A VQ design can concentrate the cells where they are needed. The advantages can increase further as N increases beyond 2.

To quantify the average distortion, we first must define a distortion measure $d(\boldsymbol{f}, \hat{\boldsymbol{f}})$, and then, again assuming the $\boldsymbol{f}$ is a random vector with pdf $\mathsf{p}(\boldsymbol{f})$, we can define the **average distortion** as follows:

$$D = \mathsf{E}[d(\boldsymbol{f}, Q(\boldsymbol{f}))] = \int d(\boldsymbol{f}, Q(\boldsymbol{f}))\, \mathsf{p}(\boldsymbol{f})\, \mathrm{d}\boldsymbol{f} = \sum_{i=1}^{L} \int_{\mathcal{C}_i} d(\boldsymbol{f}, \boldsymbol{r}_i)\, \mathsf{p}(\boldsymbol{f})\, \mathrm{d}\boldsymbol{f},$$

where the integrals are N-dimensional.

Codebook design

The set $\{\boldsymbol{r}_1, \ldots, \boldsymbol{r}_L\}$ is called a **codebook** since given the sequence of bits that describes an index $i \in \{1, \ldots, L\}$, you "look up" the corresponding $\boldsymbol{r}_i$ in the codebook to reconstruct the image (or its coefficients or model parameters).

Both the transmitter and the receiver must have a copy of the codebook through some "prior agreement" (*e.g.*, it is built into the system design) or somehow transmitted separately.

The goal of VQ codebook design is to choose the $\boldsymbol{r}_i$'s (and the $\mathcal{C}_i$'s) to minimize the average distortion D for a given class of images (*i.e.*, for a given pdf $\mathsf{p}(\boldsymbol{f})$).

Designing a VQ codebook is very challenging! (This problem alone has generated many dissertations.)

The following two conditions are necessary for a given design to be optimal.

Condition 1.

$$Q(\boldsymbol{f}) = \boldsymbol{r}_i \text{ iff } d(\boldsymbol{f}, \boldsymbol{r}_i) \le d(\boldsymbol{f}, \boldsymbol{r}_j) \text{ for } j = 1, \ldots, i-1, i+1, \ldots, L$$

In words, for every input vector, the quantizer must always choose the reconstruction level that minimizes the distortion.
(The proof is easy; just consider the alternative.)

Condition 2. Each reconstruction level $\boldsymbol{r}_i$ must minimize the average distortion in the corresponding cell $\mathcal{C}_i$:

$$\boldsymbol{r}_i = \arg\min_{\boldsymbol{r}} \mathsf{E}[d(\boldsymbol{f}, \boldsymbol{r}) \mid \boldsymbol{f} \in \mathcal{C}_i]\,. \qquad \text{"generalized centroid"}$$

Again, if this condition did not hold for a given supposedly optimal quantizer, we could easily construct a quantizer with lower average distortion, contradicting the purported optimality.

These two conditions suggest the initial design of an iterative procedure for determining the $\boldsymbol{r}_i$'s and the $\mathcal{C}_i$'s given the pdf $\mathsf{p}(\boldsymbol{f})$ and the desired number of levels L.

- If we somehow knew the $\boldsymbol{r}_i$'s, then in principle we could find the $\mathcal{C}_i$'s using Condition 1:

$$\mathcal{C}_i = \{\boldsymbol{f} : d(\boldsymbol{f}, \boldsymbol{r}_i) \le d(\boldsymbol{f}, \boldsymbol{r}_j) \text{ for } j = 1, \ldots, i-1, i+1, \ldots, L\}\,. \tag{10-3}$$

- On the other hand, if we knew the $\mathcal{C}_i$'s we could in principle find the $\boldsymbol{r}_i$'s using Condition 2.
  In particular, suppose the distortion measure is the quadratic function:

$$d(\boldsymbol{f}, \hat{\boldsymbol{f}}) = \left\| \boldsymbol{f} - \hat{\boldsymbol{f}} \right\|^2 .$$

  We know from estimation theory that the quantity that minimizes the expected squared error is the **conditional mean**:

$$\arg\min_{\boldsymbol{r}} \mathsf{E}\left[\|\boldsymbol{f} - \boldsymbol{r}\|^2 \mid \boldsymbol{f} \in \mathcal{C}_i \right] = \mathsf{E}[\boldsymbol{f} \mid \boldsymbol{f} \in \mathcal{C}_i] = \frac{\int_{\mathcal{C}_i} \boldsymbol{f}\, \mathsf{p}(\boldsymbol{f})\, \mathrm{d}\boldsymbol{f}}{\int_{\mathcal{C}_i} \mathsf{p}(\boldsymbol{f})\, \mathrm{d}\boldsymbol{f}} \tag{10-4}$$

  (*cf.* (10.9)).

Since in practice we know neither the r_i's nor the $\mathcal{C}_i$'s, we make an initial guess of the r_i's and then iterate from there.

Example. We wish to find the MMSE quantization method (into L levels) for the 1D exponential distribution $\mathsf{p}(f) = \mathrm{e}^{-f}\, u(f)$. We can begin with an initial guess of the r_i's, say $r_i = i/2$, $i = 1, \ldots, L$. The iterative algorithm then proceeds as follows.

- Given the most recent estimates of the r_i's, we evaluate (10-3) to estimate the $\mathcal{C}_i$'s. In this simple 1D case the $\mathcal{C}_i$'s are just intervals, and clearly the endpoints of those intervals are simply the midpoints between each of the r_i values as follows:

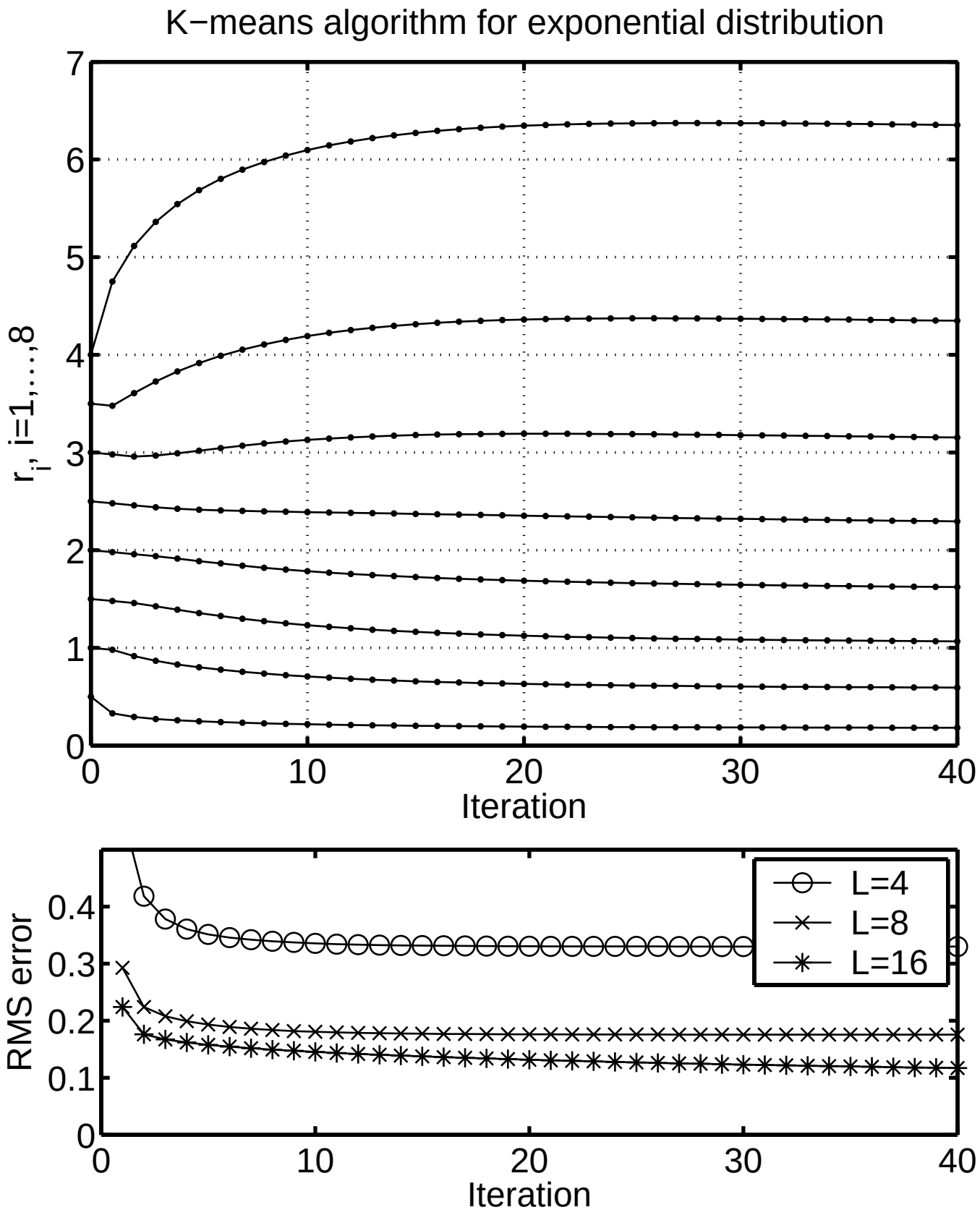
$$\mathcal{C}_i = \begin{cases} \left[0, \dfrac{r_1 + r_2}{2}\right], & i = 1 \\ \left[\dfrac{r_i + r_{i-1}}{2}, \dfrac{r_i + r_{i+1}}{2}\right], & i = 2, \ldots, L-1 \\ \left[\dfrac{r_{L-1} + r_L}{2}, \infty\right], & i = L. \end{cases}$$

- Given those most recent estimates of the $\mathcal{C}_i$'s, we evaluate (10-4) to compute new r_i's. Each $\mathcal{C}_i$ is an interval of the form $[a_i, b_i]$, where the a_i's and b_i's depend on the previous r_i's. Thus

$$r_i^{\text{new}} = \mathsf{E}[f \mid f \in \mathcal{C}_i] = \frac{\int_{\mathcal{C}_i} f\, \mathsf{p}(f)\, \mathrm{d}f}{\int_{\mathcal{C}_i} \mathsf{p}(f)\, \mathrm{d}f} = \frac{\int_{a_i}^{b_i} f\, \mathrm{e}^{-f}\, \mathrm{d}f}{\int_{a_i}^{b_i} \mathrm{e}^{-f}\, \mathrm{d}f} = \frac{(1 + a_i)\, \mathrm{e}^{-a_i} - (1 + b_i)\, \mathrm{e}^{-b_i}}{\mathrm{e}^{-a_i} - \mathrm{e}^{-b_i}}.$$

(This is a "textbook" type of example. Rarely will we get such a simple form.)

The following plot shows the r_i's as a function of iteration, starting from equally-spaced values. As the iterations proceed, the average distortion decreases towards a *local* minimum. There is no guarantee in general that the limiting r_i's produced by this method will be the *global* minimizers of the average distortion. To increase the chances of finding a global minimizer, one must initialize the algorithm multiple times with different initial guesses for the r_i's, and use the run that yields the lowest MSE.



Notice that the algorithm puts a higher density of r_i values where the pdf $\mathsf{p}(f)$ is larger.

The K-means algorithm

There is a large practical problem with the above approach though. In practice, we rarely know $\mathsf{p}(\boldsymbol{f})$; we just have **training data** consisting of images that are (hopefully) representative of the class of interest.

The **K-means algorithm** ($K = L$ here), discovered in the late 50's, is a practical solution to the problems mentioned above. It is also called the **LBG** algorithm after Linde, Buzo, and Gray who popularized it in the 80's.

The K-means algorithm is `kmeans` in MATLAB. See also the related c-means method `fcm` and `fcmdemo`.

Why "means?" If $d(\cdot, \cdot)$ is the quadratic distortion measure, then the expectation in Condition 2 is the **conditional mean** (10.9).

If we have a sample $\{\boldsymbol{f}_j\}_{j=1}^M$ of training vectors (*e.g.*, 2×2 blocks of pixels from an image), then the **maximum likelihood** estimate of the pdf $\mathsf{p}(\boldsymbol{f})$ is given by:

$$\hat{\mathsf{p}}(\boldsymbol{f}) = \frac{1}{M} \sum_{j=1}^{M} \delta_N(\boldsymbol{f} - \boldsymbol{f}_j),$$

where δ_N denotes the N-dimensional Dirac impulse. Substituting this estimator in for $\mathsf{p}(\boldsymbol{f})$ into the condition mean expression (10.9) yields

$$\frac{\int_{\mathcal{C}_i} \boldsymbol{f}\, \hat{\mathsf{p}}(\boldsymbol{f}) \,\mathrm{d}\boldsymbol{f}}{\int_{\mathcal{C}_i} \hat{\mathsf{p}}(\boldsymbol{f}) \,\mathrm{d}\boldsymbol{f}} = \frac{\int_{\mathcal{C}_i} \boldsymbol{f} \sum_{j=1}^{M} \delta_N(\boldsymbol{f} - \boldsymbol{f}_j) \,\mathrm{d}\boldsymbol{f}}{\int_{\mathcal{C}_i} \sum_{j=1}^{M} \delta_N(\boldsymbol{f} - \boldsymbol{f}_j) \,\mathrm{d}\boldsymbol{f}} = \frac{\sum_{j:\boldsymbol{f}_j \in \mathcal{C}_i} \boldsymbol{f}_j}{\sum_{j:\boldsymbol{f}_j \in \mathcal{C}_i} 1} = \text{sample mean of the } \boldsymbol{f}_j\text{'s in } \mathcal{C}_i,$$

which is an intuitive choice for $\boldsymbol{r}_i$— simply the empirical average of the associated $\boldsymbol{f}_j$'s.

Replacing (10.9) with the above empirical average in the iterative algorithm described above yields the following practical algorithm, called the **K-means algorithm**, also called the **clustering algorithm** in the pattern recognition literature. (We begin with an initial guess of $\boldsymbol{r}_i$'s.)

- Let $\mathcal{C}_i$ be the set of $\boldsymbol{f}_j$'s that are closer to $\boldsymbol{r}_i$ than to any of the other $\boldsymbol{r}$'s.
  (This step is expensive since it requires $O(LM)$ operations.)
- Let $\boldsymbol{r}_i^{\text{new}}$ be the empirical average of the $\boldsymbol{f}_j$'s in $\mathcal{C}_i$.
- Repeat...

LBG showed that this iterative algorithm converges to a local minimum of the *empirical* average distortion D, defined by:

$$D = \mathsf{E}_{\hat{\mathsf{p}}}[d(\boldsymbol{f}, Q(\boldsymbol{f}))] = \int d(\boldsymbol{f}, Q(\boldsymbol{f}))\, \hat{\mathsf{p}}(\boldsymbol{f}) \,\mathrm{d}\boldsymbol{f} = \frac{1}{M} \sum_{j=1}^{M} \|\boldsymbol{f}_j - Q(\boldsymbol{f}_j)\|^2 .$$

Again, running with multiple initial guesses can lower D at the price of increased computation.

Use of VQ codebooks

- Transmission (coding): compare a given $\boldsymbol{f}$ to all of the $\boldsymbol{r}_i$'s, finding the $\boldsymbol{r}_i$ that is the closest.
  Then transmit bits that describe the index i.
  The "compare" part of the coding operation is called a **full search** and is very computationally expensive!
  (Computations grow exponentially with the vector dimension N.)
- Decoding: look up in the codebook the $\boldsymbol{r}_i$ associated with the received bits describing the index i.

To reduce coding computation, one can design the codebook (the $\boldsymbol{r}_i$'s) using certain constraints, such as requiring a **tree codebook**, which facilitates a **binary search** rather than a **full search** for finding the closest $\boldsymbol{r}_i$.

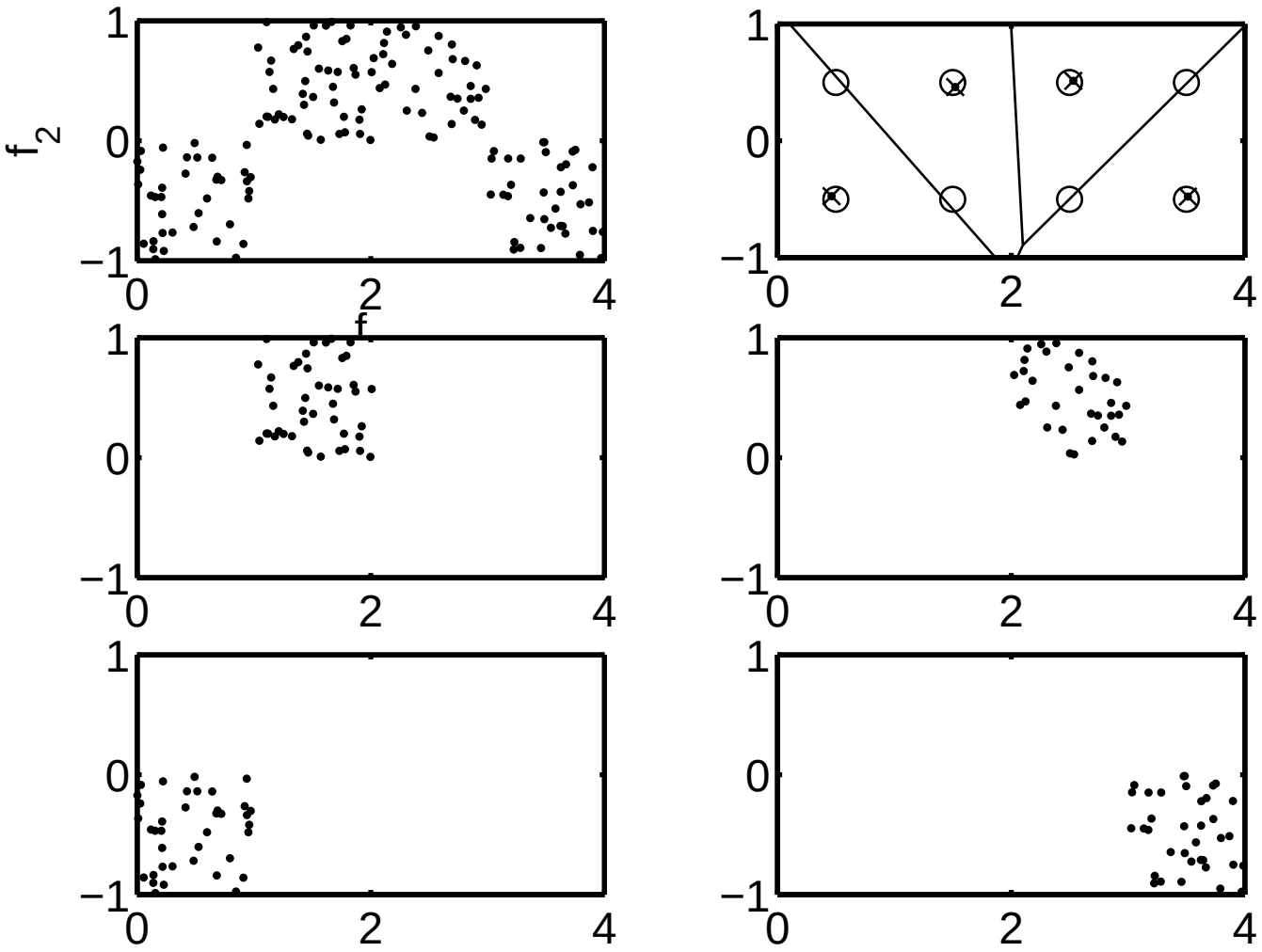
What will be the tradeoff of constraining the structure of the $\boldsymbol{r}_i$'s in the interest of saving computation? **??**

Summary of VQ

- Better performance (lower average distortion D) than scalar quantization for same bit rate.
- Greater complexity / computation.
- Appropriate for broadcast applications with one expensive central transmitter and many simpler receivers.

The principles here are useful not only for vector quantization, but also for **clustering** of data.

Example. Here is a distribution (with $N = 2$) for which vector quantization is significantly more efficient than uniform scalar quantization.



10.2

Codeword assignment

Recall:

$$\text{source image} \rightarrow \boxed{\text{transformation}} \rightarrow \boxed{\text{quantization}} \rightarrow \boxed{\text{codeword assignment}} \rightarrow \text{bit stream.}$$

After the quantizer has represented parameters of the image in terms of L possible levels, we must assign each level a specific **codeword** (*i.e.*, a sequence of 0s and 1s). The receiver in an image coding system inverts this relationship: it takes each received codeword and from it identifies the corresponding reconstruction level $\boldsymbol{r}_i$.

For the receiver to be able to uniquely identify a reconstruction level, we must assign each level $\{1, \ldots, L\}$ to a different codeword. The transmitter sends all of an image's codewords as a long sequence, so the codewords must be uniquely identifiable when received as a long sequence. Such a code is called **uniquely decodable.**

Some simple codes are not uniquely decodable.

Example. Consider the case where $L = 4$. The obvious natural codeword assignment is simply

$$\begin{bmatrix} \boldsymbol{r}_1 \rightarrow 00 \\ \boldsymbol{r}_2 \rightarrow 01 \\ \boldsymbol{r}_3 \rightarrow 10 \\ \boldsymbol{r}_4 \rightarrow 11 \end{bmatrix}$$

This obvious type of codeword assignment is called **uniform-length codeword assignment**. This choice guarantees that the sequence will be uniquely decodable.

When would you want to use anything other than uniform-length codeword assignment? **??**

If $\boldsymbol{r}_1$ and $\boldsymbol{r}_2$ occur more frequently in the class of images of interest, then one might think of trying the following codeword assignment

$$\begin{bmatrix} \boldsymbol{r}_1 \rightarrow 0 \\ \boldsymbol{r}_2 \rightarrow 1 \\ \boldsymbol{r}_3 \rightarrow 10 \\ \boldsymbol{r}_4 \rightarrow 11 \end{bmatrix},$$

so that fewer bits are allocated to the more likely centroids.

Unfortunately, the second approach is not uniquely decodable. If we receive the sequence 010, we cannot tell whether it corresponds to $\boldsymbol{r}_1, \boldsymbol{r}_3$ or $\boldsymbol{r}_1, \boldsymbol{r}_2, \boldsymbol{r}_1$. One way to fix this problem is to use a codeword assignment like the following:

$$\begin{bmatrix} \boldsymbol{r}_1 \rightarrow 0 \\ \boldsymbol{r}_2 \rightarrow 10 \\ \boldsymbol{r}_3 \rightarrow 110 \\ \boldsymbol{r}_4 \rightarrow 111 \end{bmatrix}.$$

The **average bit rate** of a coding scheme is given by

$$\bar{b} = \sum_{i=1}^{L} \mathsf{p}_i \, b_i,$$

where p_i is the probability of occurrence of the of the ith reconstruction level (aka message), and b_i is the number of bits assigned to the ith message. In a uniform codeword assignment, $b = \log_2 L$, so $\bar{b} = b$ since $\sum_{i=1}^{L} \mathsf{p}_i = 1$.

10.2.2 Variable-length codeword assignment

The latter example uses **variable-length codeword assignment**, which can be based on the relative probabilities (the p_i values) of the L "messages." If the p_i values are nonuniform, then we can usually reduce the average bit rate to be lower than $\log_2 L$ by using shorter codewords for the more probable messages.

What is the lowest possible average bit rate? Information theory shows that the average bit rate is never lower than the **entropy** of the messages, defined by:

$$H = -\sum_{i=1}^{L} \mathsf{p}_i \log_2 \mathsf{p}_i \,.$$

It can be shown that

$$0 \le H \le \log_2 L \text{ and } \bar{b} \ge H. \tag{10.32}$$

When is the entropy equal to $\log_2 L$? **??**

When is the entropy equal to 0? **??**

From (10.32) we can conclude that **uniform-length codeword assignment** is optimal for $L = 2^{\bar{b}}$ equally probable messages.

Which scalar quantization method always yields equally probable reconstruction levels? **??**

To assign codewords in a way that would achieve the lower limit on average bit rate, ideally we would use

$$b_i = -\log_2 \mathsf{p}_i$$

bits for the ith message, so that the average bit rate would be

$$\bar{b} = \sum_i \mathsf{p}_i \, b_i = \sum_i \mathsf{p}_i (-\log_2 \mathsf{p}_i) = H.$$

Unfortunately, however, $-\log_2 \mathsf{p}_i$ is usually *not* an integer, so we must do some "rounding" which will result in a code that will not quite minimize the average bit rate[1].

[1] This problem can be mitigated by grouping multiple messages together into a block and assigning codewords to each possible block.

Huffman coding

What is the lowest possible average bit rate if we constrain ourselves to a **uniquely decodable** code that (of course) must have an integer number of bits per message?

Although there may not be a simple analytical expression for this constrained minimizer, the **Huffman coding** procedure gives us an *algorithm* for designing the code (given the probabilities p_i) that is guaranteed to be optimal in that constrained sense. The following figure (from Lim) illustrates the approach:

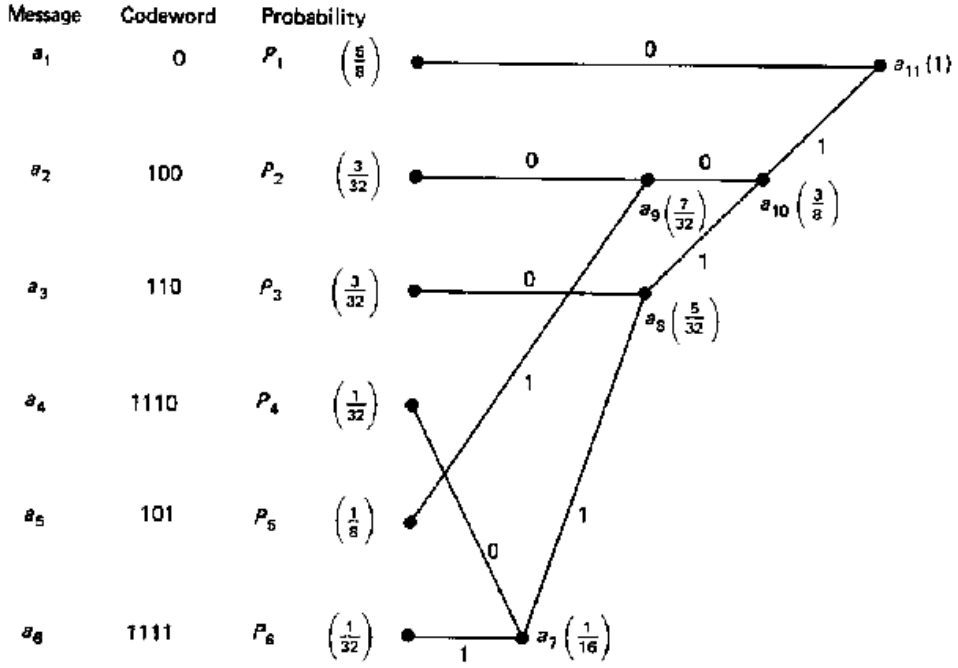


Figure 10.16 Illustration of codeword generation in Huffman coding. Message possibilities with higher probabilities are assigned with shorter codewords.

Algorithm:
- Start with the two messages with the lowest probability (a_4 and a_6).
- Assign a single bit to distinguish these messages from each other and combine the messages to form a new message a_7.
- Based on the new "reduced" set of messages $(a_1, a_2, a_3, a_5, a_7)$, assign a single bit to distinguish two messages with the lowest probability (a_3, a_7).
- New bits are added to the left of previously assigned bits.
- Continue this procedure until the most probable level is assigned.

The final code word assignment is shown above.

For this example, the entropy is

$$H = -\left[\frac{5}{8}\log_2\frac{5}{8} + \frac{3}{32}\log_2\frac{3}{32} + \cdots + \frac{1}{32}\log_2\frac{1}{32}\right] \approx 1.752 \text{ bits / message},$$

where as the average bit rate of the Huffman code is

$$\bar{b} = \frac{5}{8}1 + \frac{3}{32}3 + \cdots + \frac{1}{32}4 \approx 1.813 \text{ bits / message}.$$

What would the average bit rate be for uniform-length coding? **??**
(But this example is a bit unfair since 6 is not a power of 2.)
Since $\log_2 6 \approx 2.585$, Huffman coding is still better than what would be achieved by uniform-length coding of long blocks.

A disadvantage of variable-length coding is that the **instantaneous bit rate** differs from the **average bit rate**, so buffering is required at the receiver, which adds complexity. This is a concern in real-time image communication systems, but not a significant issue when coding for image archiving to storage.

By grouping multiple messages into a block, one can design variable-length codes whose average bit rate is as close as desired to the lower limit specified by the entropy. In practice the probabilities of the reconstruction levels are not exactly known anyway, but must be approximated by a training image set, so designing super-complicated codes to approach the theoretical entropy limit is probably pointless.

10.2.3 Joint optimization of quantization and codeword assignment

So far we have considered *separately* the issues of quantizer design and codeword assignment. In reality, these issues are coupled.

Example. If one intends to use uniform-length codes to avoid buffering issues, then the only logical choice for L is $L = 2^{\bar{b}}$, where $\bar{b}$ is the specified average bit rate. Using any L that is not a power of two would be pointless for uniform-length codes (unless block coding is applied of course).

- A uniform quantization method will usually yield *nonuniform* probabilities, so **variable-length codeword assignment** may significantly reduce the average bit rate.
- A nonuniform quantization method (companding in particular) may result in approximately equal probabilities, in which case **uniform-length codeword assignment** is likely to be adequate.

However, in our consideration of quantizer design, we tried to minimize the **average distortion** D subject to a given number of **reconstruction levels** L. In practice, it would often be preferable to minimize the average distortion *subject to a given constraint on the **average bit rate** $\bar{b}$.* (Depending on the probability distributions, different L's could yield the same $\bar{b}$ with **variable-length codeword assignment**.)

This problem is called **joint optimization** of the **quantizer** and the **codeword assignment**. However, joint optimization is a challenging nonlinear problem.

Overall, the goal is to use the minimum number of bits to code an image with average distortion that is lower than some given specification.

- In practical systems, most of the burden is usually put on either the quantization stage or the codeword assignment stage. For example, a nonuniform quantizer is used so that each reconstruction level has the same probability (*i.e.*, all messages are approximately equally probable). A simple uniform coder can be used for this system.
- In contrast, for the same level of overall distortion at the same bit rate, we could use a uniform quantizer with many more levels, and then use variable-length codewords to compensate for the fact that uniform quantization will yield unequally likely messages for any distribution of values other than uniform.
- Often practical considerations such as buffering size, computational requirements, etc., must be factored into any design and optimization procedures.

10.3

Waveform coding

Now that we have summarized quantization and codeword assignment, we return to the issue of what image variables are actually quantized and coded.

In this section we consider **waveform coding**, where the image pixel values themselves are coded. We consider **transform coding** and **model coding** in subsequent sections.

10.3.1

Pulse code modulation

The simplest waveform coding method is **pulse code modulation** (**PCM**), which is just another name for **uniform scalar quantization** of the pixel values.

Typical image acquisition devices like CCD detectors are essentially PCM systems since the A/D converter maps continuous-valued electrical signals (corresponding to image intensities at each pixel) into a finite set of levels, *e.g.*, $L = 2^8$ for an 8-bit A/D converter.

8 bits/pixel is typical for most monochrome applications, although 12 bits/pixel is also common for medical image applications. The book shows pictures illustrating the **contouring effects** of quantizing to small number of bits/pixels.

Using **nonuniform scalar quantization** can improve PCM system performance, in the spirit of **companding** as discussed earlier. Logarithmic transforms are one "common" choice.

Given an 8-bit image, how does one apply uniform scalar quantization to yield an image with only 32 levels? Do you just divide by 8? **??**

Robert's pseudonoise technique

Quantization noise is signal dependent, as discussed earlier.

To reduce the contouring effects of quantization noise, one can apply **Robert's pseudonoise technique**, which makes the noise more signal independent.

$$f[n,m] \to \underset{\uparrow}{\oplus} \to \boxed{\text{Uniform scalar quantizer}} \to \underset{\uparrow}{\oplus} \to \hat{f}[n,m]\,.$$
$$w[n,m] \qquad\qquad -w[n,m]$$

The white noise $w[n,m]$ should have a uniform distribution over $[-\Delta/2, \Delta/2]$, where Δ is the quantization interval.

Basically, we add noise before quantization and then attempt to subtract it back out after quantization. Since quantization is nonlinear, this subtract does not work perfectly, and the resulting $\hat{f}[n,m]$ will certainly have errors. But the visual characteristics of those errors may be less objectionable than the contouring effects.

Both the encoder and the decoder "must" share identical copies of $w[n,m]$. (More practically, they must share a common method for generating $w[n,m]$.)

One can apply noise reduction methods to the output image to reduce the resulting additive noise since it is approximately signal independent.

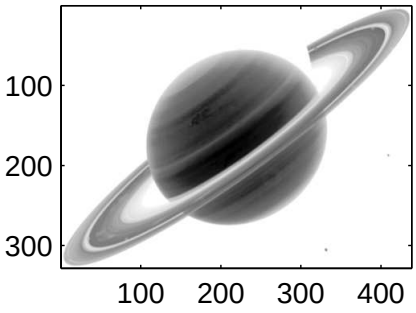
(This idea does not easily apply to the CCD situation where the input image is actually analog.)

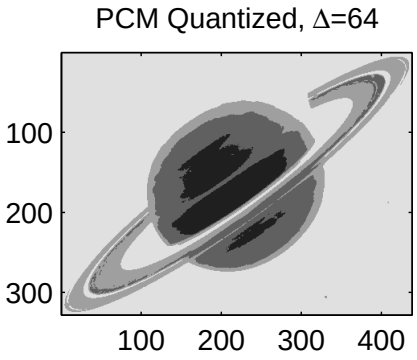
The plots on the following page show the application of Robert's technique to the Saturn image. The quantization error plots ($\hat{f} - f$ vs f) suggest that the quantization noise is independent of of the signal f.

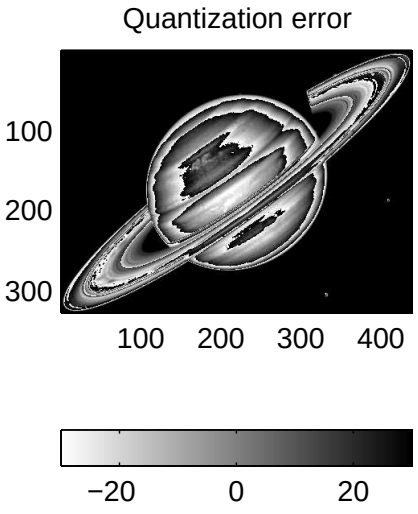
The quantized image is only 2 bits per pixel. With even 4 bits per pixel it looks pretty reasonable, since the eye is good at averaging over the local signal-independent noise.

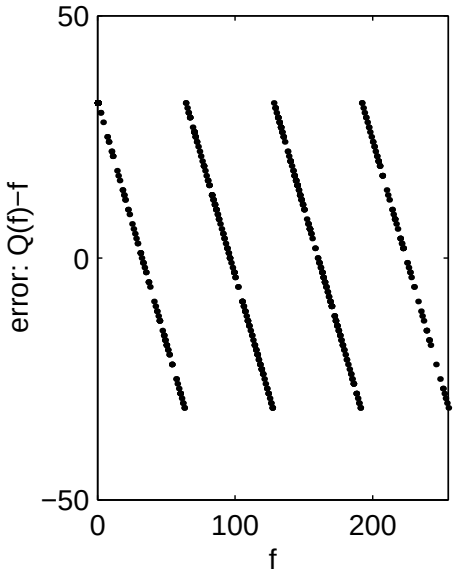
However, note the unnecessary noise in the uniform background region.

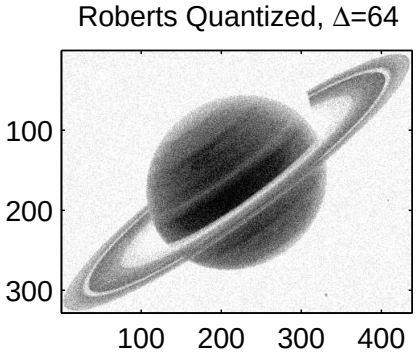
The marginal distribution of the error is essentially uniform $[-\Delta/2, \Delta/2]$ in both cases.

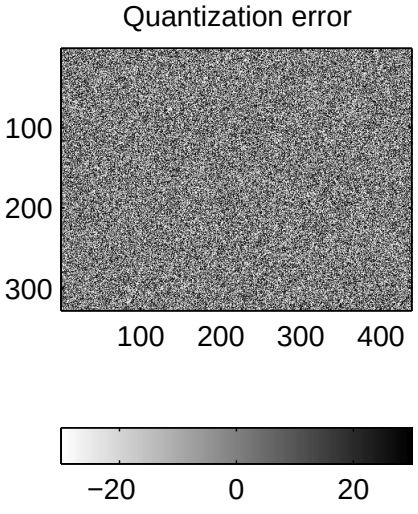

100
200
300
100 200 300 400

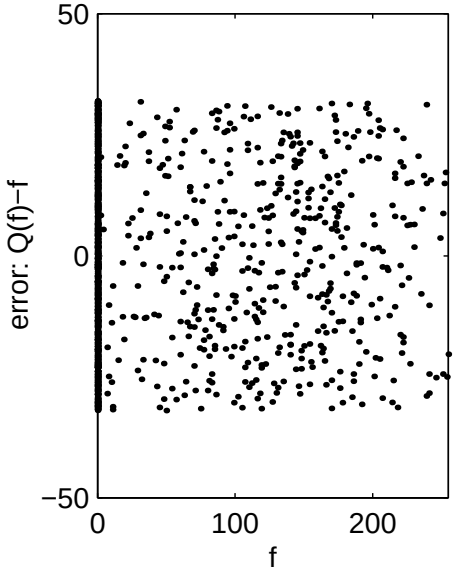

PCM Quantized, Δ=64
100
200
300
100 200 300 400


Quantization error
100
200
300
100 200 300 400
−20 0 20


50
0
−50
error: Q(f)−f
0 100 200
f


Roberts Quantized, Δ=64
100
200
300
100 200 300 400


Quantization error
100
200
300
100 200 300 400
−20 0 20


50
0
−50
error: Q(f)−f
0 100 200
f

10.3.2 Delta modulation

The PCM quantization method completely ignores the presence of correlation (*cf.* flower scatter plot shown earlier) between neighboring pixel values, since each pixel is coded independently.

One way to exploit correlation, while still using scalar quantization, is to use **delta modulation**, which codes the *difference* between neighboring pixel values.

In fact, it is even possible (and often the case) that one can use a single bit for quantizing the difference. (A one bit quantizer is particularly easy to implement.)

skim...

10.3.3 Differential PCM

Like DM but using multiple bits to code the error, and using an autoregressive prediction filter.

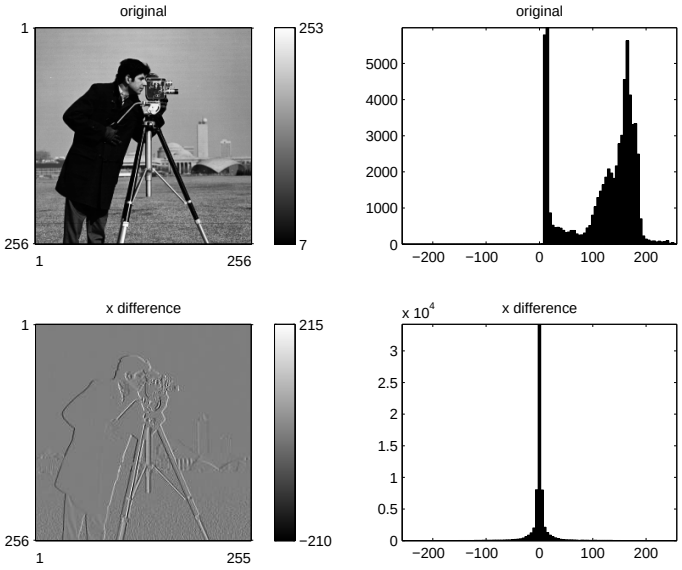
skim...

10.3.4 Two-channel coders

Separately code the low-frequency and high-frequency components, using different bit-rates and tolerable distortions for each channel.

skim...

Analysis of delta modulation



For gaussian random variables, as well as many others, entropy is a monotonically increasing function of variance. So methods that reduce variance will reduce entropy and thus improve the rate-distortion tradeoff.

Let $x[n]$ denote a WSS random process with autocorrelation function $\mathsf{R}_x[n]$. Now define the process $y[n] = x[n] - x[n-1]$ based on difference between neighboring sample values. When is the variance of $y[n]$ smaller than that of $x[n]$?

$$\mathsf{R}_y[0] = \mathsf{E}\Big[|y[n]|^2\Big] = \mathsf{E}\big[(x[n] - x[n-1])\,(x[n] - x[n-1])^*\big] = 2\,\mathsf{R}_x[0] - 2\,\mathrm{real}(\mathsf{R}_x[1]),$$

so

$$\mathsf{Var}\{y[n]\} = 2\,\mathsf{Var}\{x[n]\}\,(1 - \rho_1) \text{ where } \rho_1 = \frac{\mathrm{real}(\mathsf{R}_x[1])}{\mathsf{Var}\{x[n]\}}.$$

So $\mathsf{Var}\{y[n]\} < \mathsf{Var}\{x[n]\}$ if and only if $\rho_1 > 1/2$, *i.e.*, if there is sufficient positive correlation between neighboring signal values.

10.3.5

Pyramid coding

This is another method for exploiting correlation between pixels, both neighboring and farther away.

The basic idea is that if we appropriately down-sample an image and then upsample, we should get the original image back except for some fine details.

In **pyramid coding** the image is viewed as having several different levels of resolution. Only the differences between the different resolution levels are transmitted. Also, at each level of resolution the image is subsampled producing an image with smaller dimensions. This type of processing is a classic example of **sub-band coding**.

Pyramid coding is based on successive subsampling. Subsampling is performed by first low pass filtering the image:

$$f_0^L[n,m] = h_L[n,m] \,**\, f_0[n,m] \,.$$

Following filtering, the image is decimated at a 2:1 ratio, *i.e.*,

$$f_1[n,m] = \begin{cases} f_0^L[2n,2m], & 0 \le n \le 2^{K-1},\ 0 \le m \le 2^{K-1} \\ 0, & \text{otherwise.} \end{cases}$$

We assume $N = M = 2^K + 1$ and we use an odd number of points for each subimage so that the filter $h_L[n,m]$ is symmetric about the subsampled points, and $f_K[n,m]$ consists of a single point in the sub-band image.

The image $f_0[n,m]$ is called the **base of the pyramid**, and $f_1[n,m]$ is called the first-level image of the pyramid.

If $f_0[n,m]$ has $(2^K+1) \times (2^K+1)$ pixels, then $f_1[n,m]$ has $(2^{K-1}+1) \times (2^{K-1}+1)$ pixels.

We can repeat this process as illustrated below

$$f_k[n,m] \rightarrow \boxed{\begin{array}{c}\text{lowpass} \\ h_L[n,m]\end{array}} \rightarrow f_k^L[n,m] \rightarrow \boxed{\begin{array}{c}\text{subsample} \\ \downarrow 2\end{array}} \rightarrow f_{k+1}[n,m]$$

There are a variety of possible choices for $h_L[n,m]$, each leading to a different "type" of pyramid.

The Gaussian and Laplacian pyramid representations

The so-called **Gaussian pyramid** is based on the zero-phase separable choice $h_L[n,m] = h[n]\,h[m]$, where

$$h[n] = \left[\begin{array}{ccccccc} \frac{1}{4} - \frac{a}{2} & \frac{1}{4} & \underline{a} & \frac{1}{4} & \frac{1}{4} - \frac{a}{2} \end{array} \right],$$

and a is a free parameter. For $a = 0.4$, the filter looks like [0.05 0.25 $\underline{0.4}$ 0.25 0.05], which is roughly samples of a Gaussian function, hence the name.

Using this filter, the pyramid process can be graphically represented in 1-D as follows

(Picture)

The kth image in the pyramid has $(2^{K-k}+1) \times (2^{K-k}+1)$ pixels. If $N = M = 2^K + 1$, then the Kth image in the pyramid is just one pixel.

Earlier in the course we saw that we can approximately recover a downsampled image by using interpolation-based upsampling. Such interpolation will not *exactly* recover the original image, but the differences will small and somewhat randomly distributed. The basic idea behind of pyramid coding (for a single level) is to code the low resolution version, and the **difference image** between the original image (at that level) and its prediction based on interpolation from the coarser level.

Let I denote the interpolation operator (linear, or whatever, as long as the coder and decoder agree!). The error between an image $f_k[n,m]$ and its prediction is given as follows:

$$e_k[n,m] \triangleq f_k[n,m] - I\{f_{k+1}[n,m]\} \,.$$

If there were no quantization applied, we could recover $f_k[n,m]$ exactly at any stage by

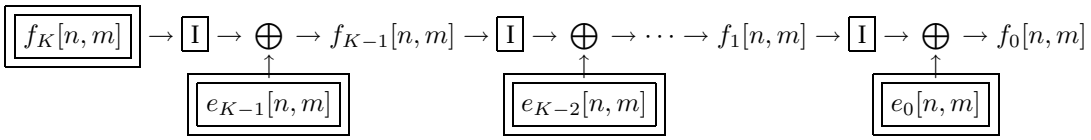
$$\underbrace{f_k[n,m] = I\{f_{k+1}[n,m]\} + e_k[n,m]}\,.$$

(Low resolution **prediction** plus additional **detail**s.)

Applying such ideas recursively, we can recover the original image $f_0[n,m]$ exactly using $f_K[n,m]$ and $e_k[n,m]$ for $k = 0, \ldots, K-1$.

The images $f_K[n,m]$ and $e_k[n,m]$ for $k = 0, \ldots, K-1$ form a **Laplacian pyramid** (so called for reasons to be described soon), where $f_K[n,m]$ is the top level of the pyramid (coarsest resolution), and $e_k[n,m]$ is the kth level of the pyramid.

The original base-level image $f_0[n,m]$ is reconstructed as follows:



The double-boxed items are quantized, coded, and transmitted.

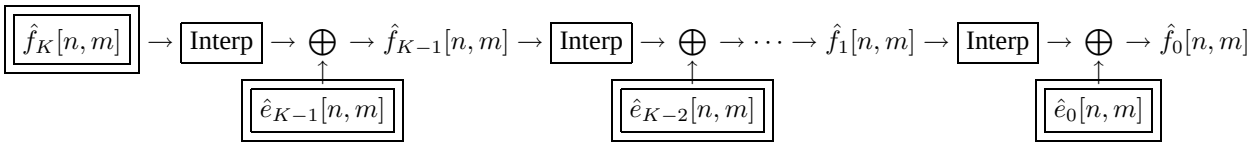
Coding issues

So far we have said little about coding.

Indeed, so far we have actually *expanded* the number of pixels by about 33%, since e_0 is the same size as f_0, and e_1 is about a quarter of that size, and e_2 is another quarter of that, so roughly $1 + 1/4 + 1/16 + \cdots = \frac{1}{1-1/4} = 4/3$. At first glance this hardly looks like progress towards **image compression**!

- Because $f_K[n,m]$ and $e_k[n,m]$ have different characteristics (see analysis below) they are quantized differently.
- Higher level (coarser) images generally have higher variances than lower level images. This means that more bits per pixel must be assigned to these images. However, these images are smaller, resulting in a small number of total bits needed.
- Lower level (finer) images ($e_0[n,m]$ and $e_1[n,m]$) will have many zeros. These images, although large, require few bits for coding if some sophisticated entropy coding scheme is used.
- "Optimal" allocation of the bits between the different pyramid levels is an interesting and challenging problem.
- Quite faithful reproduction of images can be produced using pyramid coding at very low bit rates (*e.g.*, 0.5 bit/pixel). Combinations of run-length encoding and other sophisticated entropy coding schemes are needed to reach the 0.5 bit/pixel level. Nevertheless, the complicated processing inherent in pyramid coding does result in high compression ratios.
- Pyramid coding is suitable for progressive data transmission. At any level of the reconstruction we can always produce a blurred representation of the image.

After quantizing, coding, and transmitting (or archiving) the bits corresponding to the base image and the error images, the decoder can (approximately) reconstruct the original base-level image $f_0[n,m]$ as follows:



Analysis of the Laplacian pyramid

The Laplacian pyramid is just an invertible method for **image representation** (in fact it could be called an **image transformation**), and this representation is useful for other image processing tasks like **edge detection** and **object recognition**.

What are the properties of the $e_k[n,m]$ images?

Book's analysis ***skip***

Suppose for discussion purposes that the interpolation method recovers the (lowpass filtered) image at the finer scale, *i.e.*

$$I\{f_{k+1}[n,m]\} \approx f_k^L[n,m] = h[n,m] \,**\, f_k[n,m] \,.$$

Then

$$\begin{aligned} e_k[n,m] &= f_k[n,m] - I\{f_{k+1}[n,m]\} \\ &\approx f_k[n,m] - h[n,m] \,**\, f_k[n,m] \\ &= f_k[n,m] \,**\, (\delta_2[n,m] - h[n,m]). \end{aligned}$$

In particular

$$e_0[n,m] \approx f_0[n,m] \,**\, (\delta_2[n,m] - h[n,m]).$$

Since $h[n,m]$ is a (zero phase) lowpass filter, $e_k[n,m]$ will have highpass characteristics (including zero DC component).

At the next level of the pyramid:

$$e_1[n,m] = f_1[n,m] - I\{f_2[n,m]\} \approx f_1[n,m] - h[n,m] \,**\, f_1[n,m] \,.$$

To put this back in terms of the base image's coordinates, ***the book claims that "making a few additional approximations, we obtain"*** after interpolating:

$$\begin{aligned} I\{e_1[n,m]\} &\approx I\{f_1[n,m]\} - I\{h[n,m] \,**\, f_1[n,m]\} \\ &\approx f_0[n,m] \,**\, h[n,m] - h[n,m] \,**\, f_0[n,m] \,**\, h[n,m] \\ &= f_0[n,m] \,**\, (h[n,m] - h[n,m] \,**\, h[n,m]) \,. \end{aligned}$$

Since $h[n,m]$ is lowpass, $h[n,m] - h[n,m] \,**\, h[n,m]$ ***the book claims that*** is a bandpass filter. ***But that is incorrect when*** $h[n,m]$ ***is the ideal lowpass!*** Continuing to the next level, ***the book claims that we get***

$$I\{I\{e_2[n,m]\}\} \approx f_0[n,m] \,**\, (h[n,m] \,**\, h[n,m] - h[n,m] \,**\, h[n,m] \,**\, h[n,m]) \,.$$

In each case, the filters that relate the $e_k[n,m]$ images to the base image are **bandpass filters** of different center frequencies and bandwidths. As k increases, the filter has effectively a smaller bandwidth with lower passband frequencies.

Since a Gaussian convolved with a Gaussian is still Gaussian, the filters are approximately the difference of two Gaussians, which in turn is approximately the **Laplacian of Gaussian** discussed previously, hence the name **Laplacian pyramid**.

My analysis (in 1D for simplicity)

The **down sampling** step:

$$f_k^L = h_L * f_k \overset{\text{DSFT}}{\longleftrightarrow} F_k^L = H_L F_k$$

What should $H_L(\omega)$ be ideally? **??**

$$\begin{aligned} f_{k+1} = \text{downsample}(f_k^L) \overset{\text{DSFT}}{\longleftrightarrow} F_{k+1}(\omega) &= \frac{1}{2}\left[F_k^L\left(\frac{\omega}{2}\right) + F_k^L\left(\frac{\omega}{2} \pm \pi\right)\right] \\ &= \frac{1}{2}\left[H_L\left(\frac{\omega}{2}\right) F_k\left(\frac{\omega}{2}\right) + H_L\left(\frac{\omega}{2} \pm \pi\right) F_k\left(\frac{\omega}{2} \pm \pi\right)\right]. \end{aligned}$$

In particular, if $H_L(\omega) \approx \text{rect}\left(\frac{\omega}{\pi}\right)_{(2\pi)}$, then for $|\omega| < \pi$ we have the following approximation:

$$\begin{aligned} F_k(\omega) \approx \frac{1}{2} H_L\left(\frac{\omega}{2}\right) F_{k-1}\left(\frac{\omega}{2}\right) &\approx \frac{1}{2} H_L\left(\frac{\omega}{2}\right)\left[\frac{1}{2} H_L\left(\frac{\omega}{4}\right) F_{k-1}\left(\frac{\omega}{4}\right)\right] \\ &= \frac{1}{4} H_L\left(\frac{\omega}{2}\right) F_{k-1}\left(\frac{\omega}{2}\right) \approx \cdots \approx \frac{1}{2^k} H_L\left(\frac{\omega}{2}\right) F_0\left(\frac{\omega}{2^k}\right). \end{aligned}$$

In particular, $F_k(\omega)$ consists of those frequency components of $F_0(\omega)$ up to $\pi/2^k$.

The **interpolation** step:

$$f_{k+1}^I \triangleq I\{f_{k+1}\} = h_I * f_{k+1}^U$$

where f_{k+1}^U is upsampling by zero insertion, *i.e.*,

$$f_{k+1}^U[n] = \begin{cases} f_{k+1}[n/2], & n \text{ even} \\ 0, & \text{otherwise} \end{cases} \overset{\text{DSFT}}{\longleftrightarrow} F_{k+1}^U(\omega) = F_{k+1}(2\omega),$$

and h_I is the lowpass filter associated with the interpolation. (Typically $h_I = 2h_L$.) In the frequency domain:

$$\begin{aligned} F_{k+1}^I(\omega) &= H_I(\omega) F_{k+1}^U(\omega) \\ &= H_I(\omega) F_{k+1}(2\omega) \\ &= H_I(\omega) \frac{1}{2}\left[H_L(\omega) F_k(\omega) + H_L(\omega \pm \pi) F_k(\omega \pm \pi)\right] \\ &\approx H_L^2(\omega) F_k(\omega), \end{aligned}$$

assuming $h_I = 2h_L$ and making the very reasonable approximation (which is exact for ideal lowpass filters) that

$$H_I(\omega) H_L(\omega \pm \pi) \approx 0.$$

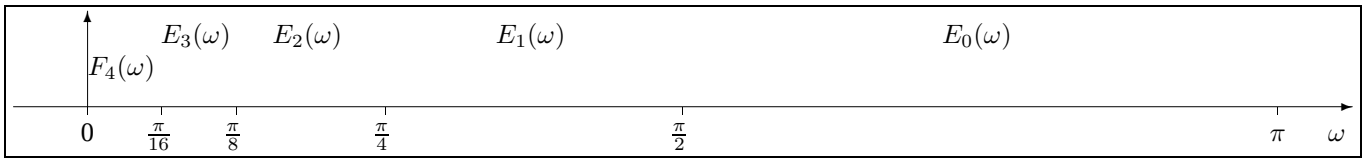
The **error** step:

$$\begin{aligned} e_k = f_k - I\{f_{k+1}\} = f_k - f_{k+1}^I \overset{\text{DSFT}}{\longleftrightarrow} E_k(\omega) = F_k(\omega) - F_{k+1}^I(\omega) &\approx \left[1 - H_L^2(\omega)\right] F_k(\omega) \\ &\approx \left[1 - H_L^2(\omega)\right] \frac{1}{2^k} H_L\left(\frac{\omega}{2}\right) F_0\left(\frac{\omega}{2^k}\right), \end{aligned}$$

for $|\omega| < \pi$. In particular, if $H_L(\omega) = \text{rect}\left(\frac{\omega}{\pi}\right)_{(2\pi)}$, then for $|\omega| < \pi$:

$$E_k(\omega) = \frac{1}{2^k} F_0\left(\frac{\omega}{2^k}\right) \mathbf{1}_{\{\pi/2 < |\omega| < \pi\}}.$$

Plugging in the limits of the indicator function into $F\left(\omega/2^k\right)$ shows that $E_k(\omega)$ contains information about the original image $F_0(\omega)$ over the frequency band $\frac{\pi}{2^{k+1}} < \omega < \frac{\pi}{2^k}$, which is closely related to **subband coding**.



More analysis of pyramid decomposition (for ideal filters and interpolators) ***for interested readers***

First, lowpass: $f_k^L[n,m] = f_k[n,m] \ast\ast h[n,m]$ where $h[n,m] = \frac{1}{4}\operatorname{sinc}_2\left(\frac{n}{2}, \frac{m}{2}\right)$. Hence

$$F_k^L(\omega_{\mathrm{X}}, \omega_{\mathrm{Y}}) = F_k(\omega_{\mathrm{X}}, \omega_{\mathrm{Y}})\, H(\omega_{\mathrm{X}}, \omega_{\mathrm{Y}}) = F_k(\omega_{\mathrm{X}}, \omega_{\mathrm{Y}})\ \mathrm{rect}_2\left(\frac{\omega_{\mathrm{X}}}{\pi}, \frac{\omega_{\mathrm{Y}}}{\pi}\right)_{(2\pi, 2\pi)}.$$

Second, downsample: $f_{k+1}[n,m] = f_k^L[2n, 2m]$. Hence from homework:

$$\begin{aligned} F_{k+1}(\omega_{\mathrm{X}}, \omega_{\mathrm{Y}}) &= \frac{1}{4}\left[F_k^L\left(\frac{\omega_{\mathrm{X}}}{2}, \frac{\omega_{\mathrm{Y}}}{2}\right) + F_k^L\left(\frac{\omega_{\mathrm{X}}}{2} - \pi, \frac{\omega_{\mathrm{Y}}}{2}\right) + F_k^L\left(\frac{\omega_{\mathrm{X}}}{2}, \frac{\omega_{\mathrm{Y}}}{2} - \pi\right) + F_k^L\left(\frac{\omega_{\mathrm{X}}}{2} - \pi, \frac{\omega_{\mathrm{Y}}}{2} - \pi\right)\right] \\ &= \frac{1}{4}\left[F_k\left(\frac{\omega_{\mathrm{X}}}{2}, \frac{\omega_{\mathrm{Y}}}{2}\right)\ \mathrm{rect}_2\left(\frac{\omega_{\mathrm{X}}}{2\pi}, \frac{\omega_{\mathrm{Y}}}{2\pi}\right)_{(2\pi, 2\pi)} + \cdots\right]. \end{aligned}$$

To save writing, hereafter we only consider the range $-\pi \le \omega_{\mathrm{X}}, \omega_{\mathrm{Y}} \le \pi$ for the arguments on the left-hand side. Thus $F_{k+1}(\omega_{\mathrm{X}}, \omega_{\mathrm{Y}}) = \frac{1}{4} F_k\left(\frac{\omega_{\mathrm{X}}}{2}, \frac{\omega_{\mathrm{Y}}}{2}\right)\ \mathrm{rect}_2\left(\frac{\omega_{\mathrm{X}}}{2\pi}, \frac{\omega_{\mathrm{Y}}}{2\pi}\right)$, since the rect's eliminate the other terms over this range.

Applying this relationship recursively yields the following pyramid of lowpass filtered images:

$$F_k(\omega_{\mathrm{X}}, \omega_{\mathrm{Y}}) = \left(\frac{1}{4}\right)^k F_0\left(\frac{\omega_{\mathrm{X}}}{2^k}, \frac{\omega_{\mathrm{Y}}}{2^k}\right)\ \mathrm{rect}_2\left(\frac{\omega_{\mathrm{X}}}{2\pi}, \frac{\omega_{\mathrm{Y}}}{2\pi}\right), \quad k = 0, 1, 2, \ldots.$$

Third, interpolate: $f_{k+1}^I[n,m] = \sum_{k=-\infty}^{\infty} \sum_{l=-\infty}^{\infty} f_{k+1}[n,m] \operatorname{sinc}_2(n/2 - k, m/2 - l)$, so from homework:

$$F_{k+1}^I(\omega_{\mathrm{X}}, \omega_{\mathrm{Y}}) = 4\, F_{k+1}(2\omega_{\mathrm{X}}, 2\omega_{\mathrm{Y}})\ \mathrm{rect}_2\left(\frac{\omega_{\mathrm{X}}}{\pi}, \frac{\omega_{\mathrm{Y}}}{\pi}\right) = F_k(\omega_{\mathrm{X}}, \omega_{\mathrm{Y}})\ \mathrm{rect}_2\left(\frac{\omega_{\mathrm{X}}}{\pi}, \frac{\omega_{\mathrm{Y}}}{\pi}\right).$$

Fourth, error image: $e_k[n,m] = f_k[n,m] - f_{k+1}^I[n,m]$, so

$$E_k(\omega_{\mathrm{X}}, \omega_{\mathrm{Y}}) = F_k(\omega_{\mathrm{X}}, \omega_{\mathrm{Y}}) - F_{k+1}^I(\omega_{\mathrm{X}}, \omega_{\mathrm{Y}}) = F_k(\omega_{\mathrm{X}}, \omega_{\mathrm{Y}})\left[1 - \mathrm{rect}_2\left(\frac{\omega_{\mathrm{X}}}{\pi}, \frac{\omega_{\mathrm{Y}}}{\pi}\right)\right].$$

Thus, using $k = 0$ we have: $\boxed{E_0(\omega_{\mathrm{X}}, \omega_{\mathrm{Y}}) = F_0(\omega_{\mathrm{X}}, \omega_{\mathrm{Y}})\left[1 - \mathrm{rect}_2\left(\frac{\omega_{\mathrm{X}}}{\pi}, \frac{\omega_{\mathrm{Y}}}{\pi}\right)\right]}$ (for $|\omega_{\mathrm{X}}, \omega_{\mathrm{Y}}| \le \pi$, and periodic otherwise).

Suppose we were to form an image $g_k[n,m] = I^k\{e_k[n,m]\}$ by taking the error image $e_k[n,m]$ at the kth level of the pyramid and performing k applications of the interpolation method given above to it (for $k \ge 1$).

Relate the spectrum of $g_k[n,m]$ to the spectrum of the base image $f_0[n,m]$. From above, for $k \ge 1$:

$$\begin{aligned} E_k(\omega_{\mathrm{X}}, \omega_{\mathrm{Y}}) &= F_k(\omega_{\mathrm{X}}, \omega_{\mathrm{Y}})\left[1 - \mathrm{rect}_2\left(\frac{\omega_{\mathrm{X}}}{\pi}, \frac{\omega_{\mathrm{Y}}}{\pi}\right)\right] \\ &= \left(\frac{1}{4}\right)^k F_0\left(\frac{\omega_{\mathrm{X}}}{2^k}, \frac{\omega_{\mathrm{Y}}}{2^k}\right)\left[\mathrm{rect}_2\left(\frac{\omega_{\mathrm{X}}}{2\pi}, \frac{\omega_{\mathrm{Y}}}{2\pi}\right) - \mathrm{rect}_2\left(\frac{\omega_{\mathrm{X}}}{\pi}, \frac{\omega_{\mathrm{Y}}}{\pi}\right)\right]. \end{aligned}$$

With $g_k[n,m] = I^k\{e_k[n,m]\}$ we have (for $k \ge 1$):

$$\begin{aligned} G_k(\omega_{\mathrm{X}}, \omega_{\mathrm{Y}}) &= 4^k E_k[2^k\omega_{\mathrm{X}}, 2^k\omega_{\mathrm{Y}}]\ \mathrm{rect}_2\left(\frac{\omega_{\mathrm{X}}}{\pi/2^{k-1}}, \frac{\omega_{\mathrm{Y}}}{\pi/2^{k-1}}\right) \\ &= F_0(\omega_{\mathrm{X}}, \omega_{\mathrm{Y}})\left[\mathrm{rect}_2\left(\frac{2^k\omega_{\mathrm{X}}}{2\pi}, \frac{2^k\omega_{\mathrm{Y}}}{2\pi}\right) - \mathrm{rect}_2\left(\frac{2^k\omega_{\mathrm{X}}}{\pi}, \frac{2^k\omega_{\mathrm{Y}}}{\pi}\right)\right]\ \mathrm{rect}_2\left(\frac{\omega_{\mathrm{X}}}{\pi/2^{k-1}}, \frac{\omega_{\mathrm{Y}}}{\pi/2^{k-1}}\right). \end{aligned}$$

Thus $\boxed{G_k(\omega_{\mathrm{X}}, \omega_{\mathrm{Y}}) = F_0(\omega_{\mathrm{X}}, \omega_{\mathrm{Y}})\left[\mathrm{rect}_2\left(\frac{\omega_{\mathrm{X}}}{\pi/2^{k-1}}, \frac{\omega_{\mathrm{Y}}}{\pi/2^{k-1}}\right) - \mathrm{rect}_2\left(\frac{\omega_{\mathrm{X}}}{\pi/2^k}, \frac{\omega_{\mathrm{Y}}}{\pi/2^k}\right)\right]}$ (for $|\omega_{\mathrm{X}}, \omega_{\mathrm{Y}}| \le \pi$, and periodic otherwise), which is a set of concentric dyadic rectangular annular subbands.

10.3.6 Adaptive image coding and vector quantization ***(skim)***

Small sub-blocks of an image can be directly quantized using vector quantization.

Typical block sizes range from 2 x 2 to 4 x 4. Above 4 x 4, an optimal vector quantizer gets very expensive.

Direct vector quantization is capable of modest reproduction but at very low bit rates (*e.g.*, less than 0.5 bits/pixel).

10.4

Transform image coding

- Transform coding techniques exploit the property that for typical images (and for appropriate transforms) most of the signal energy is concentrated in a small fraction of the transform coefficients. Because of this property, it is possible to code only a fraction of the transform coefficients without seriously affecting the image.
- The transform coefficients can be quantized and coded in any way. Traditionally, simple scalar quantization with uniform length coding has been used, where the number of reconstruction levels decreases away from the region of energy compaction.
- Vector quantization, especially of higher order coefficients, is starting to be used in real systems.
- To date, primarily linear transforms have been used for image coding.
- An ideal transform (like the KL transform) would make the coefficients uncorrelated (or even independent), in which case scalar quantization would be quite natural. (There still would be a benefit of going to VQ even in the case of independent coefficients because of the "sphere packing" problem.)

The basic block diagram of a transform codec is the following.

$$\boldsymbol{x} \rightarrow \boxed{\text{Transform}} \xrightarrow{\boldsymbol{X}} \boxed{\text{Quantize}} \xrightarrow{\hat{\boldsymbol{X}}} \boxed{\text{Transmit/Store}} \xrightarrow{\hat{\boldsymbol{X}}} \boxed{\text{"Inverse Transform"}} \rightarrow \hat{\boldsymbol{x}}$$

The vector $\boldsymbol{x} \in \mathbb{R}^N$ might represent the entire image, or, more typically, it represents some block of pixel values, *e.g.*, in JPEG one uses 8×8 blocks.

For simplicity of decoding, typically one uses a linear "inverse transform" in which the final image is represented as a linear combination of basis vectors:

$$\hat{\boldsymbol{x}} = \sum_{k=1}^{K} \boldsymbol{b}_k \hat{X}_k,$$

where the vectors $\{\boldsymbol{b}_1, \ldots, \boldsymbol{b}_K\}$ are sometimes called the **dictionary** or **library**. The traditional choice uses a dictionary where $K = N$, but more recently **overcomplete** expansions where $K > N$ have also been proposed.

For simplicity, we consider the case where $K = N$ hereafter, in which case if the $\boldsymbol{b}_k$ vectors are a basis for $\mathbb{R}^N$, and hence linearly independent, then we can define a $N \times N$ transformation matrix

$$\boldsymbol{W} = [\boldsymbol{b}_1 \ \ldots \ \boldsymbol{b}_N]^{-1},$$

so that the transform and inverse transform arg given by:

$$\boldsymbol{X} = \boldsymbol{W}\boldsymbol{x}, \qquad \hat{\boldsymbol{x}} = \boldsymbol{W}^{-1}\hat{\boldsymbol{X}}.$$

What are desirable properties of transformations?

- invertible
- easily computed
  - separable (for reduced computation via **row-column decomposition**)
  - real, fast, etc.
- correlation reduction (to facilitate scalar quantization)
- energy compaction
- energy preservation (like Parseval's theorem):

  $$\|\boldsymbol{W}\boldsymbol{x}\|^2 = c\,\|\boldsymbol{x}\|^2$$

  for some constant c independent of x.
  This property holds with $c = 1$ if $\boldsymbol{W}$ is a **unitary matrix**, *i.e.*, corresponding to a **orthonormal transformation**. In this case $\boldsymbol{W}^{-1} = \boldsymbol{W}'$, $\boldsymbol{Q}^{-1} = \boldsymbol{Q}'$, where $\boldsymbol{W}'$ is the **Hermitian transpose** of $\boldsymbol{W}$, and

  $$\|\boldsymbol{X}\|^2 = \|\boldsymbol{W}\boldsymbol{x}\|^2 = \boldsymbol{x}'\boldsymbol{W}'\boldsymbol{W}\boldsymbol{x} = \boldsymbol{x}'\boldsymbol{x} = \|\boldsymbol{x}\|^2.$$

  This property is desirable since it assures us that small transform coefficients contribute only a small amount to the reconstructed signal's energy, so discarding a small transform coefficient (*i.e.*, quantizing them to zero) will not seriously degrade the image. (Of course, discarding a *lot* of small coefficients will have a synergistic effect that is likely to be noticeable.)

There is often a *tradeoff* between the energy compaction property and computational considerations. For example, the 2D DFT is very easy to compute using the row-column 2D FFT. But image regions containing edges are poorly compacted by the DFT.

Karhunen-Loève Transform

Suppose that you are willing to assume that your image (or image block) $\boldsymbol{x}$ is a realization of a random vector with known covariance function

$$\boldsymbol{K} = \mathsf{Cov}\{\boldsymbol{x}\} = \mathsf{E}[(\boldsymbol{x} - \mathsf{E}[\boldsymbol{x}])(\boldsymbol{x} - \mathsf{E}[\boldsymbol{x}])'] \,.$$

Under this (often artificial) assumption, the *linear* transformation (actually, it is an **affine** transformation) that is optimal in terms of the energy compaction property, is the **Karhunen-Loève** (**KL**) transform (**KLT**).

The KLT solves the following problem. We wish to approximate $\boldsymbol{x} \in \mathbb{C}^N$ using the following affine expression:

$$\hat{\boldsymbol{x}} = \boldsymbol{B}\boldsymbol{\alpha} + \boldsymbol{v},$$

where $\boldsymbol{B} = [\boldsymbol{b}_1 \ \ldots \boldsymbol{b}_M]$ is a $N \times M$ matrix with $M \leq N$ orthonormal columns that does not depend on any give $\boldsymbol{x}$, but $\boldsymbol{v} \in \mathbb{C}^N$ is also independent of $\boldsymbol{x}$, $\boldsymbol{B}'\boldsymbol{v} = \boldsymbol{0}$, and each coefficient vector $\boldsymbol{\alpha} \in \mathbb{C}^M$ may depend (possibly even nonlinearly) on $\boldsymbol{x}$. We want to choose $\boldsymbol{B}$, $\boldsymbol{\alpha}$, and $\boldsymbol{v}$ to minimize the MSE $\mathsf{E}\left[\|\hat{\boldsymbol{x}} - \boldsymbol{x}\|^2\right]$.

We derive shortly that the optimal choice for $\boldsymbol{v}$ is $\boldsymbol{v} = (\boldsymbol{I} - \boldsymbol{B}\boldsymbol{B}')\boldsymbol{\mu}$, the optimal choice for $\boldsymbol{\alpha}$ is $\boldsymbol{\alpha} = \boldsymbol{B}'\boldsymbol{x}$, and most importantly, the basis vectors $\boldsymbol{b}_k$ must be the M eigenvectors of $\boldsymbol{K}$ corresponding to the largest eigenvalues.

In principle, using lexicographic ordering of finite-sized images we could use MATLAB's `eig` function to find the required eigenvalues and eigenvectors. In practice this can be nontrivial for large image sizes.

Alternatively, one can find the KL representation of small (*e.g.*, 8×8) blocks of images from a representative training set (which are used to form an empirical estimate of the covariance matrix). This approach only requires finding the eigenvectors of a 64×64 covariance matrix, which is trivial. The eigenvectors found this way will be optimal for the training set; whether they will have good energy compaction for a subsequent image of interest will depend on how "representative" that image is of the images in the training set.

If $\boldsymbol{K}$ is circulant, *i.e.*, if the image is circularly WSS, then what are the eigenvectors and eigenvalues? **??**

The sense in which the KL transform is optimal is that if we order the eigenvalues from largest to smallest, and then look at the *expected error* in a truncated expansion that only includes the first, say, M coefficients, then the KL transform will have the minimum such MSE over all linear transformations. In other words, the first M components describe as much of the variance of the signal (energy compaction) as is possible with a linear orthonormal transformation.

Furthermore, the KL coefficients are completely uncorrelated, as shown by the following argument for a finite-extent signal $\boldsymbol{x}$.

Denote the covariance matrix of the lexicographically ordered image vector $\boldsymbol{x}$ by

$$\boldsymbol{K} = \mathsf{E}[(\boldsymbol{x} - \mathsf{E}[\boldsymbol{x}])(\boldsymbol{x} - \mathsf{E}[\boldsymbol{x}])'] \,.$$

Since $\boldsymbol{K}$ is symmetric nonnegative definite, it has an orthogonal eigen-decomposition:

$$\boldsymbol{K} = \boldsymbol{V}\boldsymbol{\Lambda}\boldsymbol{V}',$$

where the columns of $\boldsymbol{V}$ are its eigenvectors, and $\boldsymbol{\Lambda}$ is a diagonal matrix with its eigenvalues λ_k.

The KL transform in matrix vector form is given by:

$$\boldsymbol{X} = \boldsymbol{V}'(\boldsymbol{x} - \mathsf{E}[\boldsymbol{x}])$$

and the inverse transform is given by

$$\boldsymbol{X} = \mathsf{E}[\boldsymbol{x}] + \boldsymbol{V}\boldsymbol{X}.$$

(Note the small detail here that the transform is actually **affine** rather than **linear** if the mean is nonzero.)

To show that the KL coefficients are uncorrelated, we examine the covariance matrix of the coefficient vector $\boldsymbol{X}$:

$$\mathsf{Cov}\{\boldsymbol{X}\} = \mathsf{Cov}\{\boldsymbol{V}'\boldsymbol{x}\} = \boldsymbol{V}' \,\mathsf{Cov}\{\boldsymbol{x}\}\, \boldsymbol{V} = \boldsymbol{V}'\boldsymbol{K}\boldsymbol{V} = \boldsymbol{V}'(\boldsymbol{V}\boldsymbol{\Lambda}\boldsymbol{V}')\boldsymbol{V} = \boldsymbol{\Lambda},$$

which is diagonal. So the KL coefficients are perfectly **uncorrelated**. They also have the best **energy compaction** as shown below.

However, for the reasons described above, the KL transform is *impractical*, so is rarely used in practical image coding. It is useful for analysis and insight.

KLT derivation

Let $\boldsymbol{x}$ be a random vector of length N with known $N \times N$ covariance matrix $\boldsymbol{K}$ and known mean $\boldsymbol{\mu}$.

Let $\boldsymbol{T} = [\boldsymbol{\phi}_1 \ \ldots \boldsymbol{\phi}_N]'$ denote an $N \times N$ **orthonormal** transformation with basis functions $\boldsymbol{\phi}_i$, $i = 1, \ldots, N$, such that $\boldsymbol{T}'\boldsymbol{T} = \boldsymbol{T}\boldsymbol{T}' = \boldsymbol{I}$, *i.e.*, $\|\boldsymbol{\phi}_i\| = 1$ and $\boldsymbol{\phi}_i'\boldsymbol{\phi}_j = 0, \ i \neq j$.

We want to form an approximation to $\boldsymbol{x}$ using just $M < N$ components:

$$\hat{\boldsymbol{x}}(\boldsymbol{\alpha}, \boldsymbol{\nu}) = \sum_{i=1}^{M} \alpha_i \boldsymbol{\phi}_i + \boldsymbol{\nu},$$

where the α_i's can depend on $\boldsymbol{x}$, but $\boldsymbol{\nu}$ cannot (we must preselect $\boldsymbol{\nu}$), and where $\boldsymbol{\nu}'\boldsymbol{\phi}_i = 0$ for $i = 1, \ldots, M$.

Goal: choose $\boldsymbol{\nu}$, $\boldsymbol{\alpha}$, and $\boldsymbol{T}$ (the $\boldsymbol{\phi}_i$'s) to minimize the MSE between $\hat{\boldsymbol{x}}$ and $\boldsymbol{x}$, subject to the constraint $\boldsymbol{T}\boldsymbol{T}' = \boldsymbol{T}'\boldsymbol{T} = \boldsymbol{I}$.

The approximation error is

$$\varepsilon(\boldsymbol{\alpha}, \boldsymbol{\nu}) = \|\hat{\boldsymbol{x}}(\boldsymbol{\alpha}, \boldsymbol{\nu}) - \boldsymbol{x}\|^2 = \left\| \sum_{i=1}^{M} \alpha_i \boldsymbol{\phi}_i + \boldsymbol{\nu} - \boldsymbol{x} \right\|^2$$

and we first minimize this error over $\boldsymbol{\alpha}$:

$$\min_{\boldsymbol{\alpha}} \varepsilon(\boldsymbol{\alpha}, \boldsymbol{\nu}) = \min_{\boldsymbol{\alpha}} \|\hat{\boldsymbol{x}}(\boldsymbol{\alpha}, \boldsymbol{\nu}) - \boldsymbol{x}\|^2 = \min_{\boldsymbol{\alpha}} \left\| \sum_{i=1}^{M} \alpha_i \boldsymbol{\phi}_i + \boldsymbol{\nu} - \boldsymbol{x} \right\|^2$$

$$\frac{\partial}{\partial \alpha_i} \varepsilon(\boldsymbol{\alpha}, \boldsymbol{\nu}) = \boldsymbol{\phi}_i' \left[\sum_{k=1}^{M} \alpha_k \boldsymbol{\phi}_k + \boldsymbol{\nu} - \boldsymbol{x} \right] = \alpha_i - \boldsymbol{\phi}_i' \boldsymbol{x},$$

since $\boldsymbol{T}'\boldsymbol{T} = \boldsymbol{I}$. Equating the partial derivatives to zero yields

$$\alpha_i = \boldsymbol{\phi}_i' \boldsymbol{x}, \ i = 1, \ldots, M.$$

Thus we have

$$\hat{\boldsymbol{x}}(\boldsymbol{\nu}) = \sum_{i=1}^{M} \boldsymbol{\phi}_i \boldsymbol{\phi}_i' \boldsymbol{x} + \boldsymbol{\nu}$$

with corresponding error:

$$\varepsilon(\boldsymbol{\nu}) \triangleq \mathsf{E}\left[\|\hat{\boldsymbol{x}}(\boldsymbol{\nu}) - \boldsymbol{x}\|^2 \right] = \mathsf{E}\left[\left\| \sum_{i=1}^{M} \boldsymbol{\phi}_i \boldsymbol{\phi}_i' \boldsymbol{x} + \boldsymbol{\nu} - \boldsymbol{x} \right\|^2 \right] = \mathsf{E}\left[\left\| \boldsymbol{\nu} - \sum_{i=M+1}^{N} \boldsymbol{\phi}_i \boldsymbol{\phi}_i' \boldsymbol{x} \right\|^2 \right] \equiv \|\boldsymbol{\nu}\|^2 - 2\boldsymbol{\nu}' \sum_{i=M+1}^{N} \boldsymbol{\phi}_i \boldsymbol{\phi}_i' \boldsymbol{x}.$$

Taking the gradient with respect to $\boldsymbol{\nu}$ yields

$$\frac{1}{2} \nabla' \varepsilon(\boldsymbol{\nu}) = \boldsymbol{\nu} - \sum_{i=M+1}^{N} \boldsymbol{\phi}_i \boldsymbol{\phi}_i' \boldsymbol{\mu}.$$

Equating this gradient to zero yields:

$$\boldsymbol{\nu} = \sum_{i=M+1}^{N} \boldsymbol{\phi}_i \boldsymbol{\phi}_i' \boldsymbol{\mu}.$$

Fortunately, this solution for the best $\boldsymbol{\nu}$ satisfies $\boldsymbol{\phi}_i'\boldsymbol{\nu} = 0$, $i = 1, \ldots, M$, otherwise we would have to minimize over $\boldsymbol{\nu}$ using Lagrange multipliers.

We now have our MMSE (over the α_i's and $\boldsymbol{\nu}$) affine approximation:

$$\hat{\boldsymbol{x}} = \sum_{i=1}^{M} \boldsymbol{\phi}_i \boldsymbol{\phi}_i' \boldsymbol{x} + \sum_{i=M+1}^{N} \boldsymbol{\phi}_i \boldsymbol{\phi}_i' \boldsymbol{\mu} = \sum_{i=1}^{M} \boldsymbol{\phi}_i \boldsymbol{\phi}_i' (\boldsymbol{x} - \boldsymbol{\mu}) + \boldsymbol{\mu},$$

so we can turn to the more interesting and challenging problem of asking: what should the $\boldsymbol{\phi}_i$'s be (for $i = 1, \ldots, M$) to minimize the MSE? The MSE is

$$\begin{aligned}
\varepsilon &= \mathsf{E}\left[\|\hat{\boldsymbol{x}} - \boldsymbol{x}\|^2\right] = \mathsf{E}\left[\left\|\sum_{i=1}^{M} \boldsymbol{\phi}_i \boldsymbol{\phi}_i'(\boldsymbol{x} - \boldsymbol{\mu}) + \boldsymbol{\mu} - \boldsymbol{x}\right\|^2\right] = \mathsf{E}\left[\left\|\sum_{i=M+1}^{N} \boldsymbol{\phi}_i \boldsymbol{\phi}_i'(\boldsymbol{x} - \boldsymbol{\mu})\right\|^2\right] \\
&= \sum_{i=M+1}^{N} \mathsf{E}\left[(\boldsymbol{\phi}_i'(\boldsymbol{x} - \boldsymbol{\mu}))^2\right] = \sum_{i=M+1}^{N} \boldsymbol{\phi}_i' \, \mathsf{E}[(\boldsymbol{x} - \boldsymbol{\mu})(\boldsymbol{x} - \boldsymbol{\mu})'] \, \boldsymbol{\phi}_i = \sum_{i=M+1}^{N} \boldsymbol{\phi}_i' \boldsymbol{K} \boldsymbol{\phi}_i
\end{aligned}$$

To minimize over $\boldsymbol{\phi}_i$, we must include the constraint $\|\boldsymbol{\phi}_i\|^2 = 1$ using a Lagrange multiplier:

$$\nabla' \varepsilon - \lambda_i \nabla' \left(\|\boldsymbol{\phi}_i\|^2 - 1 \right) = 2\boldsymbol{K}\boldsymbol{\phi}_i - \lambda_i 2 \boldsymbol{\phi}_i$$

Equating to zero yields:

$$\boldsymbol{K}\boldsymbol{\phi}_i = \lambda_i \boldsymbol{\phi}_i,$$

which means that the $\boldsymbol{\phi}_i$'s must be **eigenvectors** of the covariance matrix $\boldsymbol{K}$.

Substituting this relationship back into the error expression yields:

$$\varepsilon = \sum_{i=M+1}^{N} \lambda_i.$$

Clearly then, to minimize the MSE for any M we should arrange the λ_i's (and hence the corresponding eigenvectors, the $\boldsymbol{\phi}_i$'s) in *decreasing order*.

Thus, we have shown that, over all (affine) orthonormal transformations, truncated to $M < N$ terms, the KLT has the MMSE, assuming that the covariance matrix $\boldsymbol{K}$ and mean $\boldsymbol{\mu}$ are known.

Example. Generally it is impractical to apply the KLT to an entire image. However, one can extract small blocks of pixels from an image (*e.g.*, 8×8) and arrange those as 64-element vectors $\boldsymbol{x}_l$ for $l = 1, \ldots, L$ where L is the number of such blocks available in a training set. From the training set, we can *estimate* the covariance matrix $\boldsymbol{K}$ using the sample covariance:

$$\hat{\boldsymbol{K}} = \frac{1}{L} \sum_{l=1}^{L} (\boldsymbol{x}_l - \bar{\boldsymbol{x}})(\boldsymbol{x}_l - \bar{\boldsymbol{x}})' \text{ where the sample mean vector is: } \bar{\boldsymbol{x}} = \frac{1}{L} \sum_{l=1}^{L} \boldsymbol{x}_l.$$

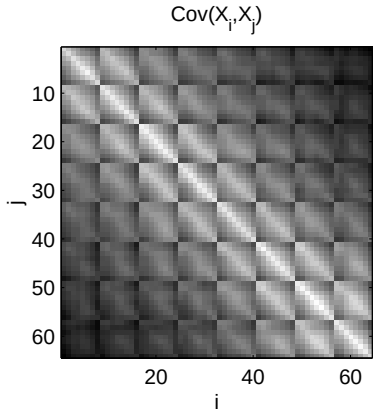
The empirical covariance matrix $\hat{\boldsymbol{K}}$ is nonnegative definite, so we can use its (orthonormal) eigenvectors as an **empirical KL transform**.

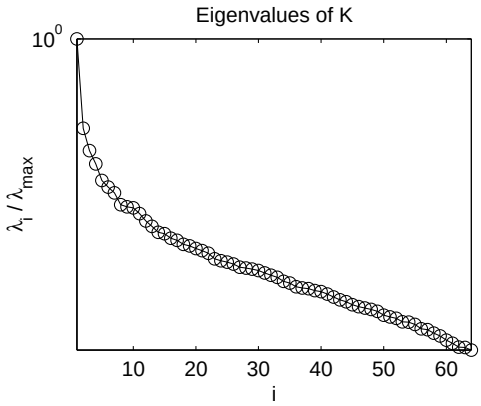
A practical disadvantage of this approach is that the basis would be different for every image (or class of images), so in a heterogeneous environment like the internet, one would have to transmit the basis along with the image, which would add some overhead and complexity. Using a "generic" predetermined basis like the DCT avoids the need for communicating a different basis for each image.

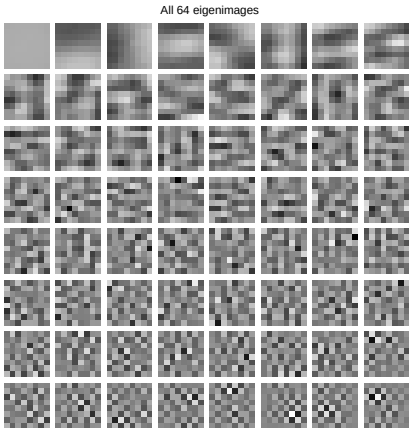
Summary: the KLT gives the MMSE basis for a truncated representation of a random vector with known mean and covariance matrix. There was no consideration of quantization errors in this analysis. However, for "high resolution" (asymptotic) bit rates, one can generalize this conclusion to include the effects of quantization errors. (See EECS 651...)

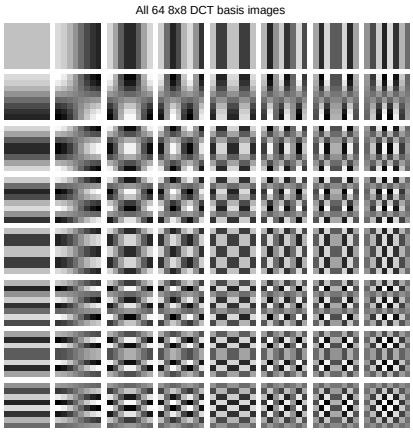
Example.

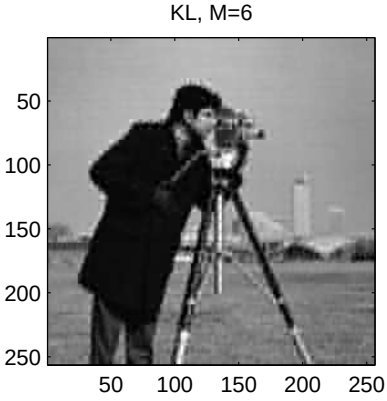
I took the 256x256 `camerman.tif` image and partitioned it into $L = 32^2$ subimages of size 8×8, which I arranged as a collection of L vectors of length $N = 64$ and passed it to MATLAB's `cov` routine, which computes the sample covariance $\hat{\boldsymbol{K}}$ described above. The first figure below shows that 64×64 covariance matrix. I then passed the sample covariance matrix to MATLAB's `eig` function which computed its eigenvalues (shown below) and eigenvectors. I `reshape`d each 64×1 eigenvector into an 8×8 eigenimage, and all 64 eigenimages are shown below. For comparison purposes, I also show the 64 basis images associated with the DCT command (using `kron` and `dctmtx`). There is a remarkable degree of similarity.

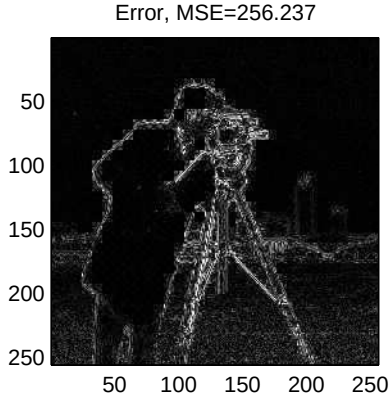

Cov(X_i,X_j)
10
20
30
40
50
60
j
20
40
60
i

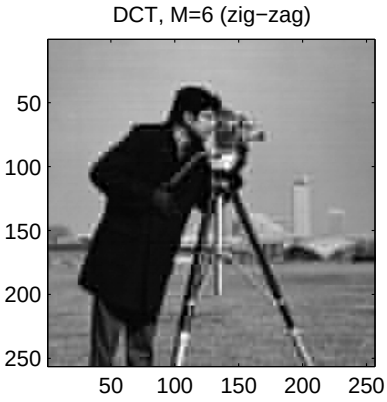

Eigenvalues of K
10^0
λ_i / λ_max
10
20
30
40
50
60
i

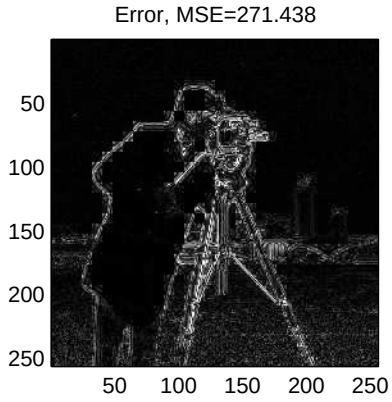

All 64 eigenimages


All 64 8x8 DCT basis images


KL, M=6
50
100
150
200
250
50
100
150
200
250


Error, MSE=256.237
50
100
150
200
250
50
100
150
200
250


DCT, M=6 (zig−zag)
50
100
150
200
250
50
100
150
200
250


Error, MSE=271.438
50
100
150
200
250
50
100
150
200
250

Other transforms

DCT, DFT, and Hadamard are the most common.

The **Hadamard transform** is computationally the simplest. Its $N \times N$, basis matrices are denoted $\boldsymbol{H}_n$ where $N = 2^n$, $n = 1, 2, 3, \ldots$. They can be generated easily by the core matrix

$$\boldsymbol{H}_1 = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$$

and the recursion

$$\boldsymbol{H}_n = \boldsymbol{H}_{n-1} \otimes \boldsymbol{H}_1 = \frac{1}{\sqrt{2}} \begin{bmatrix} \boldsymbol{H}_{n-1} & \boldsymbol{H}_{n-1} \\ \boldsymbol{H}_{n-1} & -\boldsymbol{H}_{n-1} \end{bmatrix}.$$

The Hadamard transform is **unitary**.

If we ignore the $\sqrt{2}$'s, computing the transform simply requires additions and subtractions!

However, the Hadamard transform, does not usually have the energy compaction properties of either the DFT or DCT.
There usually is a tradeoff between complexity and energy compaction.

We have previously discussed the fact that the DCT often has better energy compaction than the DFT because of the way edge conditions are handled.

10.4.2

Practical considerations in transform image coding

Because of computational constraints, most transform coding systems work on **subimages**.

However, the subimages must not be chosen too small or there will be little exploitation of correlations between image regions.

Typically 8 x 8 or 16 x 16 sub-images are used.

All coefficients of the transform are not coded with equal accuracy.

There are two basic approaches to transferring reduced information about transform coefficients.

- **Zonal and threshold coding**
  - Only encode a limited set of transform coefficients.
  - Each coefficient of the reduced set is coded with high accuracy.
  - The set of coefficients to be coded can be determined as follows.
    - Use predetermined **zones**, as shown in Fig. 10.43.
    - Select coefficients based on a threshold condition.

    Obviously, the threshold approach produces a better representation, but must somehow also encode information about the positions of the selected frequencies.
    Run-length coding and variations can be applied for this purpose.
- **Bit allocation methods**
  Bits for coding subimages are allocated based on the expected variance of the transform coefficients. Coefficients with a large variance are assigned a large number of bits, and coefficients with small variances are assigned a small number of bits. An example of such an allocation for DCT coding at 0.5 bit/pixel is shown in Fig. 10.44.
  So that the same quantization step size can be used for all coefficients, typically the coefficients are first normalized by their "expected" standard deviation prior to scalar quantization.

Degradations

- Quantization noise can causes "graininess"
- Coefficients corresponding to high spatial frequencies are usually eliminated or assigned fewest bits, resulting in a loss of **spatial resolution**.
- Thus use of subimages causes **blocking effect** due to artificial discontinuities at block boundaries that are very noticeable to the human eye.

Summary of design parameters

- transform
- subimage size
- selection of coefficient set
- bit allocation
- quantization levels

One practical issue is whether a variable bit rate is tolerable. For example, using a threshold to select coefficients will yield different numbers of coded coefficients for different blocks.

10.4.3 Reduction of blocking effect

Blocking artifacts can be reduced by overlapping subimages. However, in image coding large overlaps are not acceptable because of the greatly increased number of coefficients. Consequently, only small overlap, or no overlap at all, is used in practice for transform image coding.

Blocking artifacts can also be reduced by low pass filtering in the neighborhood of subimage boundaries.

10.4.4 Hybrid transform coding

In a hybrid coder, one spatial dimension is waveform coded and the other is transform coded, as illustrated below

$$f[n,m] \rightarrow \boxed{\text{1D transform along rows}} \rightarrow \boxed{\text{1D ``waveform'' coding along columns}} \rightarrow \boxed{\text{codeword assignment}} \rightarrow$$

$$\rightarrow \boxed{\text{decoding}} \rightarrow \boxed{\text{1D ``waveform'' reconstruction along columns}} \rightarrow \boxed{\text{1D inverse transform along row}} \rightarrow \hat{f}[n,m]$$

Although conceptually simple, hybrid systems have not been used very much in practice. This is because at very low bit rates, transform coders work best, but at very high bit rates where faithful reproductions are needed, a hybrid system does not perform any better than a simple waveform coder. Thus, hybrid coders have limited applicability in 2D.

10.4.5 Adaptive coding schemes

Bit allocations, zones, etc. can all be selected adaptively, either per image, or per block or group of blocks.

What is the tradeoff of adaptive methods? **??** The overhead associated with adaptive methods usually is worth the overall savings! A classic example of this is JPEG, which uses adaptive coding of DCT coefficients in 8×8 blocks.

JPEG coding, a case-study in practical transform image coding (from [4])

Goals:
- Close to state-of-the art (circa 1990) in terms of compression rate
- Generally applicable (grayscale, color, multiple component)
- User selectable quality level
- Tractable computation
- Four modes (4 separate codecs since no single method adequately works for all situations):
  - Sequential - left-to-right, top-to-bottom scan
  - Progressive - blurry-to-clear recovery. First pass, low frequency DCT coefficients. Subsequent passes, additional frequency bands of DCT coefficients. Decoded images are all the same size at each pass.
  - Lossless - exact recovery (albeit with smaller compression ratios)
  - Hierarchical - access coarsely sampled version without decoding high resolution information.
    (similar to **pyramid coding** considered in HW)
    Useful for display on monitor before printing (thumbnail).

Design choices

1. **Transform**

   Tradeoff between complexity and energy compaction.
   What are the two extremes?
   - KL is best compaction, most complexity since it is image (covariance class) dependent.
   - Waveform coding is lowest complexity but no energy compaction.

   Practical compromises
   - A low complexity choice is the **Hadamard transform**, discussed above, which requires only additions and subtractions. But its energy compaction underperforms the DFT and DCT.
   - The **JPEG** committee chose the **DCT** based on a (blinded) competition using subjective image quality.
     We would expect DCT to outperform DFT based on earlier discussion of edge conditions for DCT (mirroring) vs DFT (periodic).
   - Accuracy specifications on DCT (due to cosines)
   - 8 or 12 bit input images (easy to add front-end for floats)
   - For 8-bit input images, the DCT coefficients are in the range $-2^{10}, 2^{10} - 1$.

2. **Subimage size**

   Tradeoff between computation and exploitation of correlation.
   - Large subimages (or entire image) exploit more correlation, possibly yielding better compression ratios, although the "economies of scale" in things like shared quantization tables are reduced.
   - Small subimages require less computation.
     If an $N \times N$ image is broken into blocks of size $M \times M$, the computation required is $O(M^2 \log M)$ per block for $(N/M)^2$ blocks for a total of $O(N^2 \log M)$, so the smaller M is the less computation.
   - The **JPEG** committee chose 8×8, again based on blinded visual quality.

3. **Quantization / coefficient subset**

   The 64 DCT coefficients for each block have different variances/energies, so one should quantize them differently.

   Some subset of coefficients will even be discarded (quantized to zero).
   See earlier discussion of predetermined **zones** (predetermined subset) vs **threshold coding** where the subset is chosen based on the coefficient magnitudes.

   Tradeoff: threshold coding preserves more signal energy, but requires extra bits to code indices of nonzero subset for every block.

   The **JPEG** standard:
   - 8 by 8 table describing (uniform scalar) quantizer step size for each of the 64 DCT coefficients, from 1 to 255.
     In terms of the subset of nonzero coefficients, is this **zones** or **threshold based**? **??**
   - Adaptive: specified by "user" for each image. (Default quantization tables are available based on psychovisual experiments.)

4. **Coding** (bit allocation)
   - DC coefficient coded by difference from previous block (DM idea)
     Why? Usually large (0-255 image values), and often varies slowly spatially.
   - Remaining coefficients ordered in a zig-zag sequence from low to high spatial frequency.
   - Entropy coding (Huffman or arithmetic) of (bits describing) quantized coefficients.
     Requires separate "user supplied" Huffman code table be shared (*e.g.*, encoded) between the coder and decoder.

Other considerations
- Color images etc. (each plane compressed separately)
- Multiple quantization tables and coding tables supported, so they can differ for each component. (But only one table within a component.)

Performance (starting with 8-bit images)
- 0.25 - 0.5 bits/pixel: moderate to good quality, sufficient for some applications;
- 0.5 - 0.75 bits/pixel: good to very good quality, sufficient for many applications;
- 0.75 - 1/5 bits/pixel: excellent quality, sufficient for most applications;
- 1.5 - 2.0 bits/pixel: usually indistinguishable from the original, sufficient for the most demanding applications.
- Lossless mode (prediction from Left, Above, Above-Left neighbors) yields 2:1 compression of moderately complex color scenes.

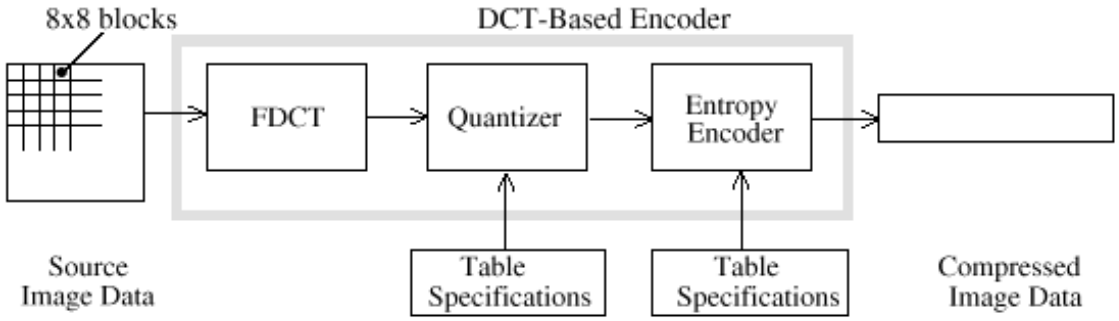


Figure 1. DCT-Based Encoder Processing Steps

Issue: "block artifacts" - many papers on correction methods.

10.5

Image model coding

The block diagram of a model coder is shown below.

Transmitter:

$$f[n,m] \rightarrow \boxed{\text{analysis}} \rightarrow \text{model parameters} \rightarrow \boxed{\text{encoder}} \rightarrow$$

Receiver:

$$\rightarrow \boxed{\text{decoder}} \rightarrow \text{quantized model parameters} \rightarrow \boxed{\text{synthesis}} \rightarrow \hat{f}[n,m]$$

In principle, one can use any model of a class of images to generate a reduced set of model parameters capturing image content.

Example. Replace "grass" parts of an image with synthetically generate grass texture.

Example. (Fig. 10.53) A portion of the mandrill's facial hair replaced with pixels synthesized by a simple first-order Markov model. Hundreds of pixels reduced to just a few parameters.

- Geometry models
  Objects in the image are contained in a finite set of possible objects.
- Texture models.
  Image texture often can be modeled as a random field with some statistical description. Psychophysical measurements have shown that random fields with the same 2nd order statistics (*i.e.*, covariance matrices) appear similar to a human observer.
- Combined geometry and texture.
  Segment an image into uniform regions based on edge information and generate statistically based texture within identified regions.
- Fractals.
  Use fractal measures to capture complex geometry using a small number of parameters. This method is similar to texture based methods, where a specific fractal model is used to capture statistics.

The **synthesis** part of such a system is quite challenging, and typically requires a decoder with sizable computational capability.

10.6

Interframe image coding

There are several ways that we can exploit interframe correlations to improve performance and/or reduce average bit rates.

In the simplest system, either direct waveform coding, such as DPCM, or transform coding, such as DCT, can be directly extended to three dimensions. Direct extension to 3D often is not optimal, especially because of the increased computation.

A hybrid system, unlike in 2D, can be very effective for interframe coding. A 3D hybrid system is shown below.

Transmit:

$$f(n,m,z) \rightarrow \boxed{\text{2D transform for each z}} \rightarrow T(k,l,z) \rightarrow \boxed{\text{“Waveform coding” along z}} \rightarrow \boxed{\text{codeword assignment}} \rightarrow$$

Receive:

$$\rightarrow \boxed{\text{decoding}} \rightarrow \boxed{\text{1D waveform reconstruction along z}} \rightarrow \boxed{\text{2D inverse transform for each z}} \rightarrow \hat{f}(n,m,z)$$

The waveform coding portion is usually DPCM so that only one or two frames must be stored at the decoder.

The hybrid system greatly reduces computations because only those DCT coefficients which are retained need be coded by the interframe DPCM system. That is, waveform coding is only applied to a fraction of the DCT coefficients.

There is no major delay in a hybrid system. For example, in a full 3D transform coder, one frame cannot be reconstructed until all transform coefficients within a 3D block are received. This means there is at least an N-frame delay in reconstructions for a 3D transform coder using N consecutive frames of information.

Some simple waveform coders based on DM or DPCM only transmit interframe differences that exceed a certain threshold. The instantaneous bit rate fluctuates, but the average rate can be very small. With proper buffering at the receiver, this simple type of system is capable of significantly reduced bit rates.

As discussed in the previous chapters, interframe motion can be estimated. Using estimates of the motion, we can transmit only the interframe difference between the expected, motion compensated image and the current image can be transmitted. Also, the motion parameters must be transmitted. The motion is estimated by the receiver using multiple frames, assuming the same motion parameters between frames. Thus, motion does not need to be transmitted given that the decoder has access to the previous estimated frames.

10.6.2 Color image coding ***skim***

10.6.3 Channel error effects

Errors in communication have different effects on different image coding schemes. In PCM coding, a bit reversal only affects the particular pixel whose intensity is represented by the bit in error. This type of noise, at reasonable low bit rates, can be removed using median filtering, or other nonlinear filtering methods, or often just ignored.

In DPCM coding, bit reversals affect multiple pixel values. Because reconstructed intensities are used recursively, a bit reversal affects all subsequent pixel intensities from that point on. Because of the deleterious effect of a small number of bit reversals on DPCM reconstruction, these systems are often used with error correcting codes. Error correction, which is necessary to maintain fidelity, increases the bits/pixel needed to code the image.

A bit reversal in transform codings affects one particular coefficient. Each coefficient, however, affects all pixels within the subimage. This, the entire subimage may be corrupted solely by a single bit reversal. Again, as with DPCM, error correcting codes are often used with transform coders to eliminate the effect of channel noise. The addition of error correction once again increases the bit/pixel needed to code the image.

Summary

This chapter described methods for image coding. Image (and video) coding are very active research areas, due to the many commercial, medical, and scientific applications of digital image.

The JPEG2000 standard is based on wavelets, which offer many advantages including **progressive decoding** (*cf.* pyramid decoder).

10.7 Final comments on image coding

If a highly accurate reconstruction of an image is needed, then there is little difference between the bit rates of waveform and transform coders. Consequently, because of their simplicity and robustness to channel noise, waveform coders are usually selected for these applications.

Transform coders, such as JPEG, yield good performance at low bit rates (*i.e.*, in the 0.5-1 bit/pixel range). These coders are usually computationally complex, but can be justified in circumstances where the cost of the encoder and decoder are small compared to the cost of communication bandwidth (*e.g.*, the Web).

Model coders can yield modest performance but at greatly reduced bit rate (*i.e.*, < 0.1 bit/pixel). These coders are most commonly used in applications where intelligibility is the major concern and bandwidths are minimal.

Adaptive coding, although conceptually expensive, often can enhance the performance of almost any image coding system. For most applications, the increased complexity and slight loss of bandwidth due to communication of adaptive parameters can often be justified given the level of improvement (*e.g.*, JPEG adaptive bit allocation).

Interframe coding is very useful in applications such as HDTV where the sequence of frames has significant temporal correlation.

A general rule is that as the bits/pixel decreases, the complexity increases. The major design tradeoff usually is between the complexity of the encoder vs. decoder given a particular communications bandwidth.

Based on the way errors propagate, the decoder must be designed to be robust against channel noise. Error-correcting codes are often used to reduce the effect of channel noise.

Bibliography

[1] M. A. Tekalp. *Digital video processing*. Prentice-Hall, New York, 1996.
[2] R. M. Gray. Quantization noise spectra. *IEEE Trans. Info. Theory*, 36(6):1220–44, November 1990.
[3] A. Gersho. Asymptotically optimal block quantization. *IEEE Trans. Info. Theory*, 25(4):373–380, July 1979.
[4] G. K. Wallace. The JPEG still picture compression standard. *IEEE Trans. Consumer Electronics*, 38(1):xviii–xxxiv, February 1992.