Coding with distortion

We have a signal $x_n,\ n = 1 \dots N$ to code. The alphabet is a subset of the real numbers $\mathcal{A} \subseteq \mathbb{R}$. The alphabet can be continuous.

If we don't have the demand that the decoded signal should be exactly the same as the original signal we can get a lower data rate than if we have lossless coding. Typically the signal is described using a smaller alphabet than the original signal uses (quantization).

In the case where the original alphabet is continuous, in general an infinite number of bits is required to describe the signal losslessly.

The more bits that are used, the closer to the original signal the decoded signal $\hat{x}_n$ will be.

Distortion measure

We need a measure of how much error we have in the decoded signal, the so called *distortion*.

The most common measure is a quadratic error measure, combined with averaging over the whole sequence

$$D = \frac{1}{N} \sum_{n=1}^{N} (x_n - \hat{x}_n)^2$$

This is the *mean square error* of the decoded sequence.

Distortion measure, cont.

Often we want to consider the distortion (or noise power) relative to the signal power, the so called *signal to noise ratio* (SNR)

$$\sigma_x^2 = \frac{1}{N} \sum_{n=1}^{N} x_n^2$$

$$\mathrm{SNR} = \frac{\sigma_x^2}{D}$$

SNR is usually expressed in dB

$$\mathrm{SNR} = 10 \cdot \log_{10} \frac{\sigma_x^2}{D}$$

PSNR

When coding still images and video we usually use the *peak-to-peak signal to noise ratio* (PSNR)

$$\mathrm{PSNR} = 10 \cdot \log_{10} \frac{x_{pp}^2}{D}$$

where x_{pp} is the difference between the maximum and minum values of the signal.

For example, if the data to be coded is a grayscale image quantized to 8 bits, the signal can assume values between 0 and 255. The PSNR is then

$$\mathrm{PSNR} = 10 \cdot \log_{10} \frac{255^2}{D}$$

Random signal models

A signal can be modelled as an amplitude continuous stationary random process X_n, with distribution function $F_X(x)$ and density function $f_X(x)$.

$$F_X(x) = Pr(X \leq x)$$

$$f_X(x) = \frac{d}{dx} F_X(x)$$

$$f_X(x) \geq 0 \ , \quad \forall x$$

$$\int_{-\infty}^{\infty} f_X(x) dx = 1$$

$$Pr(a \leq X \leq b) = F_X(b) - F_X(a) = \int_a^b f_X(x) dx$$

Random signal models, cont.

Mean value

$$m_X = E\{X_n\} = \int_{-\infty}^{\infty} x \cdot f_X(x) dx$$

Quadratic mean value

$$E\{X_n^2\} = \int_{-\infty}^{\infty} x^2 \cdot f_X(x) dx$$

Variance

$$\sigma_X^2 = E\{(X_n - m_x)^2\} = E\{X_n^2\} - m_x^2$$

In most of our cases we will use signal models with mean value 0. In those cases the variance is equal to the quadratic mean value.

The variance (or rather the quadratic mean value) is a measure of the signal power.

Common distributions

Uniform distribution

$$f_X(x) = \begin{cases} \frac{1}{b-a} & a \leq x \leq b \\ 0 & \text{otherwise} \end{cases}$$

Mean value $m = \frac{a+b}{2}$, variance $\sigma^2 = \frac{(b-a)^2}{12}$

Gaussian distribution (normal distribution)

$$f_X(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x-m)^2}{2\sigma^2}}$$

Laplace distribution

$$f_X(x) = \frac{1}{\sqrt{2}\sigma} e^{-\frac{\sqrt{2}|x-m|}{\sigma}}$$

Random signal models, cont.

The dependence of the signal value in two times instances n och m is given by the twodimensional density function $f_{X_n X_m}(x_n, x_m)$.

If we can write this as a product $f_X(x_n) \cdot f_X(x_m)$ we say that the signal in the two time instances are *independent*.

A signal where all time instances are independent of each other is a *memoryless* signal or a *white* signal.

In most cases we will describe the dependence using the *correlation* $E\{X_n \cdot X_m\}$.

If $E\{X_n \cdot X_m\} = E\{X_n\} \cdot E\{X_m\}$ we say that the signal in the two time instances are *uncorrelated*. Independent signals are uncorrelated, but the reverse is not necessarily true.

Memory sources

Markov source of order k

$$f(x_n|x_{n-1}x_{n-2}\ldots) = f(x_n|x_{n-1}\ldots x_{n-k})$$

Linear models, ϵ_n white (memoryless) noise.
AR(N)

$$x_n = \sum_{i=1}^{N} a_i \cdot x_{n-i} + \epsilon_n$$

MA(M)

$$x_n = \sum_{j=1}^{M} b_i \cdot \epsilon_{n-j} + \epsilon_n$$

ARMA(N, M)

$$x_n = \sum_{i=1}^{N} a_i \cdot x_{n-i} + \sum_{j=1}^{M} b_i \cdot \epsilon_{n-j} + \epsilon_n$$

Random signal models, cont.

The correlation properties of the signal is usually expressed using the *auto correlation function*, which for a stationary process is given by

$$R_{XX}(k) = E\{X_n X_{n+k}\}$$

The auto correlation function is symmetric: $R_{XX}(-k) = R_{XX}(k)$.

We also have: $|R_{XX}(k)| \leq R_{XX}(0) = E\{X_n^2\}$.

For a memoryless (white) process we have

$$R_{XX}(k) = \sigma_X^2 \cdot \delta(k) = \begin{cases} \sigma_X^2 & k = 0 \\ 0 & \text{otherwise} \end{cases}$$

For an AR(1) process we have

$$R_{XX}(k) = a^{|k|} \cdot \sigma_X^2 \qquad (|a| < 1)$$

Multidimensional signals

The auto correlation function can of course also be defined for multidimensional signals. For instance, for a twodimensional stationary random process $X_{i,j}$ the auto correlation function is given by

$$R_{XX}(k,l) = E\{X_{i,j}X_{i+k,j+l}\}$$

The auto correlation function is symmetric: $R_{XX}(-k,-l) = R_{XX}(k,l)$

We also have: $|R_{XX}(k,l)| \leq R_{XX}(0,0) = E\{X_{i,j}^2\}$

Random signal models, cont.

For a random signal X_n that is coded and then decoded to $\hat{X}_n$, the distortion is given by

$$D = E\{(X - \hat{X})^2\} = \int_{-\infty}^{\infty} (x - \hat{x})^2 f_X(x) dx$$

The signal power is (given mean zero)

$$E\{X^2\} = \sigma_X^2 + (E\{X\})^2 = \sigma_X^2$$

and SNR as before

$$\text{SNR} = 10 \cdot \log_{10} \frac{\sigma_X^2}{D}$$

Theoretical limit

The rate-distortion function $R(D)$ for a source gives the theoretically lowest rate R we can use to code the source, on the condition that the maximum allowed distortion is D. Compare to the entropy limit for source coding.

Example: White gaussian process with variance σ^2

$$R(D) = \begin{cases} \frac{1}{2}\log\frac{\sigma^2}{D} & 0 < D \leq \sigma^2 \\ 0 & \text{otherwise} \end{cases}$$

Ie, if we allow a distortion that is larger than the variance of the process, we don't need to transmit any bits at all. The decoder can just set the decoded signal equal to the mean value at each time instance, which will give a distortion equal to the variance.
We can also se that $R \to \infty$ when $D \to 0$

Gaussian source with memory

For gaussian sources with memory, the rate-distortion function can be calculated from the power spectral density.

$$\Phi(\theta) = \mathcal{F}\{R_{XX}(k)\} = \sum_{k=-\infty}^{\infty} R_{XX}(k) \cdot e^{-j2\pi\theta k}$$

The rate-distortion function is parametrically given by

$$R = \int_{-1/2}^{1/2} \max\{\frac{1}{2}\log\frac{\Phi(\theta)}{\lambda}, 0\}\ d\theta$$

and

$$D = \int_{-1/2}^{1/2} \min\{\lambda, \Phi(\theta)\}\ d\theta$$

The integration can of course be done over any interval of size 1, since the power spectral density is a periodic function.

Quantization

Mapping from a continuous alphabet to a discrete alphabet (or mapping from a large discrete alphabet to a smaller one). After quantization we have a discrete signal, on which we can use our source coding methods (Huffman, arithmetic coding, et c.)

A general M level quantizer is specified by $M + 1$ *decision borders* b_i ; $i = 0 \dots M$ and M *reconstruction levels (or reconstruction points)* y_i ; $i = 1 \dots M$.

The quantization operator $Q(x)$ is given by

$$Q(x) = y_i \quad \text{if} \quad b_{i-1} < x \leq b_i$$

And the reconstructed signal is thus

$$\hat{x}_n = Q(x_n)$$

Quantization

Sometimes it can be useful to see quantization and reconstruction as two separate operations instead of just one operation.

Quantization: $x \rightarrow j$ such that $b_{j-1} < x \leq b_j$

Reconstruction: $\hat{x} = y_j$

A sequence of x thus gives a sequence of indices j that can then be coded by a source coder

The receiver decodes the index sequence and the maps the indices to the corresponding reconstruction points.

Quantization, cont.

Given a random signal model the distortion is

$$\begin{aligned} D &= E\{(X-\hat{X})^2\} = \\ &= \int_{-\infty}^{\infty} (x-Q(x))^2 f_X(x)dx = \\ &= \sum_{i=1}^{M} \int_{b_{i-1}}^{b_i} (x-y_i)^2 f_X(x)dx \end{aligned}$$

If no special source coding is used, ie if we just code the quantized signal using a fixed length code, the rate is

$$R = \lceil \log_2 M \rceil$$

Uniform quantization

The distance between two reconstruction points is constant

$$y_j - y_{j-1} = \Delta$$

Δ is the *stepsize* of the quantizer.

The reconstruction points are in the middle of their intervals, which means that all decision regions (apart from the end intervals in some cases) also are of the same size

$$b_i - b_{i-1} = \Delta$$

Uniform quantization, cont.

To simplify the calculations we can assume that the number of reconstruction points is given by $M = 2^R$ and that the quantizer is symmetric around the origin. The following results can easily be generalized to arbitrary M.

The reconstruction point belonging to the interval $[(j-1)\Delta, j\Delta]$ is

$$y_j = \frac{2j-1}{2}\Delta$$

The simplest case is when the input distribution is uniform on the interval $[-A, A]$:

$$\Delta = \frac{2A}{M}$$

Uniform quantization, cont.

The distortion for uniform quantization of a uniform distribution:

$$D = \sum_{i=-M/2+1}^{M/2} \int_{(i-1)\Delta}^{i\Delta} (x - \frac{2i-1}{2}\Delta)^2 \frac{1}{2A} dx = M \cdot \frac{1}{2A} \cdot \frac{\Delta^3}{12} = \frac{\Delta^2}{12}$$

$$\sigma_X^2 = \frac{(2A)^2}{12} = \frac{\Delta^2 M^2}{12}$$

$$\begin{aligned} \text{SNR} &= 10 \cdot \log_{10} \frac{\sigma_X^2}{D} = 10 \cdot \log_{10} M^2 = \\ &= 10 \cdot \log_{10} 2^{2R} = 20 \cdot R \cdot \log_{10} 2 \approx 6.02 \cdot R \end{aligned}$$

For every bit added to the quantizer (ie for every doubling of the number of reconstruction points) we will get approximately 6 dB higher SNR.

Uniform quantization, cont.

For unlimited distributions (eg a gaussian distribution) the two end intervals will be infinitely large (in the calculations below we assume that that M is even and that the quantizer is symmetric around the origin).

$$\begin{aligned} D &= \sum_{i=-M/2+1}^{M/2} \int_{(i-1)\Delta}^{i\Delta} (x - \frac{2i-1}{2}\Delta)^2 f_X(x)dx + \\ &+ \int_{(M/2)\Delta}^{\infty} (x - \frac{M-1}{2}\Delta)^2 f_X(x)dx + \\ &+ \int_{-\infty}^{-(M/2)\Delta} (x - \frac{-M+1}{2}\Delta)^2 f_X(x)dx \end{aligned}$$

The last two terms are called the *overload distortion* of the quantizer.

Uniform quantization, cont.

To find the best choice of Δ (the one that minimizes the distortion) we have to solve

$$\frac{\partial}{\partial \Delta} D = 0$$

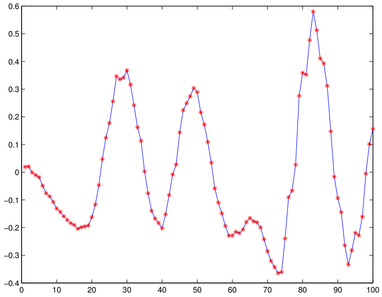
which in the general case is a hard problem. Normally we will have to find a numeric solution.

If the number of quantization levels M is large and Δ is chosen so that the overload distortion is small compared to the total distortion, the distortion is approximately

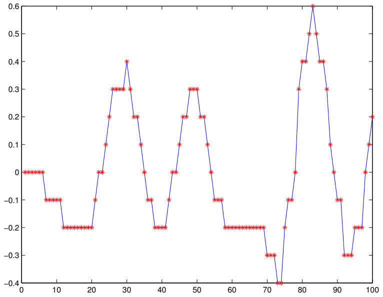
$$D \approx \frac{\Delta^2}{12}$$

Uniform quantization, example

Uniform quantization of a speech signal



Original signal



Quantized using $\Delta = 0.1$

Measured distortion: $D \approx 0.0008517$
(Compare to $\Delta^2/12 \approx 0.0008333$)

Quantization using source coding

The probability $P(j)$ of being in interval j is

$$P(j) = \int_{b_{j-1}}^{b_j} f_X(x)dx$$

In the general case these probabilities are different for different intervals. We could thus get a lower rate than $\log M$ by using some form of source coding.

Finding the optimal quantizer given an allowed rate R after source coding is a hard problem. However, it can be shown that for sufficiently large R (fine quantization) the optimal quantizer is a uniform quantization. Thus, if we are using some form of source coding, it is enough to use the simplest form of quantization.

Fine quantization

When we have *fine quantization*, ie when the number of quantization levels is large, the distortion is approximatively given by

$$D \approx c \cdot \sigma_X^2 \cdot 2^{-2R}$$

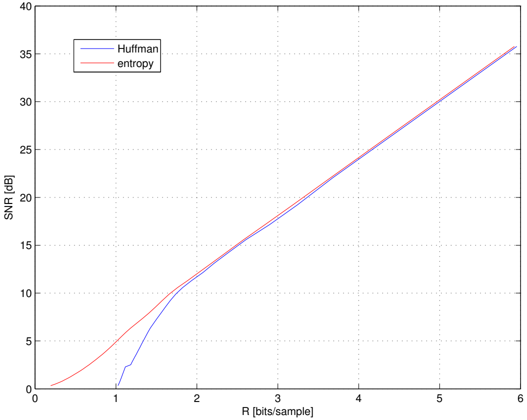
where σ_X^2 is the signal variance, R is the rate and c is a constant depending on the type of quantization and the distribution of the signal.

When we have fine quantization, the SNR as a function of the rate is approximately given by

$$SNR = 10 \cdot \log_{10} \frac{\sigma_X^2}{D} \approx 6.02 \cdot R - 10 \cdot \log_{10} c$$

Example

A mono music file is coded using a uniform quantizer (midtread), followed by Huffman coding. The rate is varied by varying the quantizer stepsize. No limitation of the number of levels is done. For comparison, we have also measured the entropy of the quantized signal.



Transform coding

1. Split the signal into blocks of size N (or $N \times N$ if the signal is twodimensional). Transform the blocks using a suitable, reversible transform to a new sequence.
2. Quantize the transform components.
3. Use some kind of source coding on the quantized transform components (fixed length coding, Huffman, arithmetic coding et c.)

Matrix description

The transform and the inverse transform can be written in matrix form as

$$\bar{\theta} = \mathbf{A} \cdot \bar{\mathbf{x}} \quad ; \quad \bar{\mathbf{x}} = \mathbf{B} \cdot \bar{\theta}$$

where

$$\bar{\mathbf{x}} = \begin{pmatrix} x_0 \\ x_1 \\ \vdots \\ x_{N-1} \end{pmatrix} \quad ; \quad \bar{\theta} = \begin{pmatrix} \theta_0 \\ \theta_1 \\ \vdots \\ \theta_{N-1} \end{pmatrix}$$

and the matrix element at position (i, j) is given by

$$[\mathbf{A}]_{i,j} = a_{i,j} \quad ; \quad [\mathbf{B}]_{i,j} = b_{i,j}$$

The matrices **A** and **B** are the inverses of each other, ie $\mathbf{B} = \mathbf{A}^{-1}$.

Orthonormal transforms

We are usually only interested in *orthonormal* transforms, ie transforms where $\mathbf{B} = \mathbf{A}^{-1} = \mathbf{A}^T$.

Orthonormal transforms are energy preserving, ie the sum of the squares of the transformed signal is equal to the sum of the squares of the original signal

$$\begin{aligned}
\sum_{i=0}^{N-1} \theta_i^2 &= \bar{\theta}^T \bar{\theta} \\
&= (\mathbf{A}\bar{\mathbf{x}})^T \mathbf{A}\bar{\mathbf{x}} \\
&= \bar{\mathbf{x}}^T \mathbf{A}^T \mathbf{A}\bar{\mathbf{x}} \\
&= \bar{\mathbf{x}}^T \bar{\mathbf{x}} = \sum_{i=0}^{N-1} x_i^2
\end{aligned}$$

Parseval's identity

The transform as a basis change

The transform can be seen as describing the signal in another basis, ie as a linear combination of new basis vectors

$$\begin{aligned}
\bar{\mathbf{x}} &= \mathbf{A}^T \bar{\theta} \\
&= \begin{pmatrix} a_{00} & \cdots & a_{N-1,0} \\ \vdots & \ddots & \vdots \\ a_{0,N-1} & \cdots & a_{N-1,N-1} \end{pmatrix} \begin{pmatrix} \theta_0 \\ \vdots \\ \theta_{N-1} \end{pmatrix} \\
&= \theta_0 \begin{pmatrix} a_{00} \\ \vdots \\ a_{0,N-1} \end{pmatrix} + \ldots + \theta_{N-1} \begin{pmatrix} a_{N-1,0} \\ \vdots \\ a_{N-1,N-1} \end{pmatrix}
\end{aligned}$$

The rows of the transform matrix (or the columns in the inverse transform matrix) are the basis vectors of the new basis.

Properties

Some desirable properties of the transform

- The transform should concentrate the signal energy to as few components as possible.
- The transform should decorrelate the transform components, ie if possible we want $E\{\theta_i \cdot \theta_j\} = 0,\ i \neq j$. This means that we remove all dependance (memory) between the transform components.
- The transform should be robust with respect to changes in source statistics.
- The transform should be simple and fast to calculate.

All of these properties can not be found in one transform.

The Karhunen-Loève-transform (KLT)

The KLT is a transform that will completely decorrelate the transform components and also give maximal energy concentration.

Assuming we have an input signal that is modelled as a stationary random process X_n with mean zero and auto correlation function $R_{XX}(k) = E\{X_n X_{n+k}\}$. Given a block size of N, we have signal vectors

$$\bar{\mathbf{x}} = \begin{pmatrix} X_n \\ X_{n+1} \\ \vdots \\ X_{n+N-1} \end{pmatrix}$$

The *correlation matrix* $\mathbf{R}_X$ is the matrix

$$\mathbf{R}_X = E\{\bar{\mathbf{x}}\bar{\mathbf{x}}^T\}$$

KLT, cont.

The correlation matrix can be expressed using the auto correlation function

$$\mathbf{R}_X = \begin{pmatrix} R_{XX}(0) & R_{XX}(1) & \cdots & R_{XX}(N-1) \\ R_{XX}(1) & R_{XX}(0) & \cdots & R_{XX}(N-2) \\ \vdots & \vdots & \ddots & \cdots \\ R_{XX}(N-1) & R_{XX}(N-2) & \cdots & R_{XX}(0) \end{pmatrix}$$

The correlation matrix $\mathbf{R}_\theta$ of the transformed signal, given a transform $\mathbf{A}$, is given by

$$\mathbf{R}_\theta = E\{\bar{\theta}\bar{\theta}^T\} = E\{\mathbf{A}\bar{\mathbf{x}}(\mathbf{A}\bar{\mathbf{x}})^T\} = \mathbf{A}\mathbf{R}_X\mathbf{A}^T$$

If we want the transform to decorrelate the signal, ie diagonalize $\mathbf{R}_\theta$ (all values zero except for the main diagonal), we should choose the basis vectors (rows of $\mathbf{A}$) as the normalized eigenvectors of $\mathbf{R}_X$.

KLT, cont.

For a KLT, the variances of the transform components will be equal to the eigenvalues of the signal correlation matrix.

In addition to decorrelating the source, the KLT will also be the transform that gives the maximum energy concentration to a few transform components. This is the same as saying that the KLT is the transform that minimizes the geometric mean of the transform component variances

$$(\prod_{i=0}^{N-1} \sigma_i^2)^{1/N}$$

A disadvantage of the KLT is that it is signal dependent, so it has to be transmitted as side information. There is usually also no fast way to perform the transform.

The discrete cosine transform (DCT)

The transform matrix **C** is given by

$$[\mathbf{C}]_{ij} = \begin{cases} \sqrt{\frac{1}{N}} & ;\ i = 0 \\ \sqrt{\frac{2}{N}} \cos \frac{(2j+1)i\pi}{2N} & ;\ i = 1, \ldots, N-1 \end{cases}$$

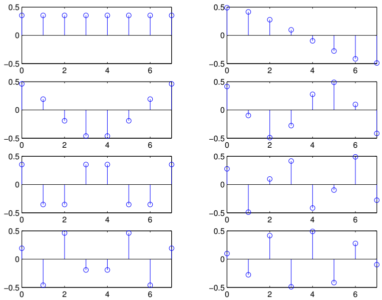
The DCT is a close relative of the discrete fourier transform (DFT). There are fast ways of doing a DCT, in the same way that there are fast fourier transforms (FFT).

The DCT will usually be very close to a KLT for sources where there is a high correlation between consecutive samples, which includes most natural audio and image sources.

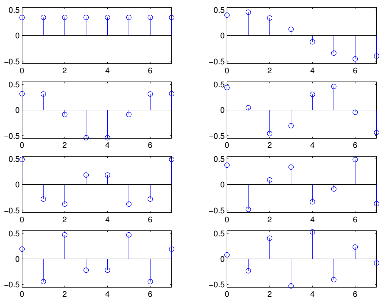
The DCT is the most commonly used transform in image and video coding. For instance it is used in the JPEG and MPEG standards.

DCT and KLT

Basis vectors for 8-point DCT and a KLT adapted to a music signal



DCT



KLT

Twodimensional signals

For a twodimensional signal (eg an image) we take blocks of size $N \times N$ to transform.

In general we can view this block as a vector of N^2 samples and use a transform matrix of size $N^2 \times N^2$.

Usually a *separable* transform is used. We then consider the block as a matrix $\mathbf{X}$ instead of a vector. A onedimensional transform is applied first the the rows of $\mathbf{X}$ and then on the columns (or the other way, the order will not matter). The resultat is a matrix $\boldsymbol{\Theta}$ of transform components

$$\boldsymbol{\Theta} = \mathbf{A}\mathbf{X}\mathbf{A}^T$$

The inverse transform is given by

$$\mathbf{X} = \mathbf{A}^T\boldsymbol{\Theta}\mathbf{A}$$

Twodimensional signals

We can view the block $\mathbf{X}$ as a linear combination of new basis matrices α_{ij} given by

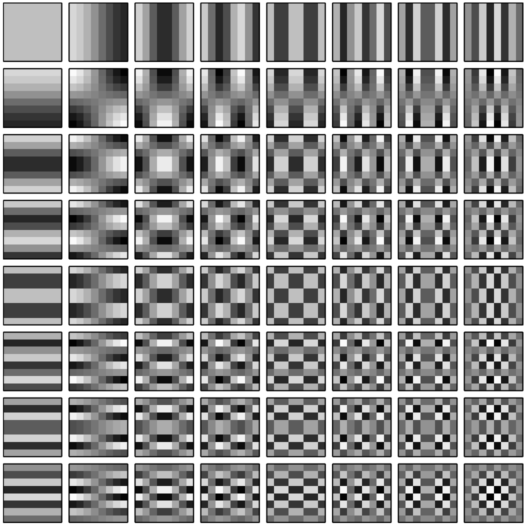
$$\alpha_{ij} = \bar{\mathbf{a}}_i^T \bar{\mathbf{a}}_j$$

where $\bar{\mathbf{a}}_i$ and $\bar{\mathbf{a}}_j$ are the i:th and j:th rows of $\mathbf{A}$.

$$\mathbf{X} = \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} [\mathbf{\Theta}]_{ij} \cdot \alpha_{ij}$$

A separable transform can always be written as a general transform applied to a vector of N^2 elements, but the reverse is not true.

Basis matrices for a 8 $\times$ 8 DCT



Block size

How should we choose the block size N?

A large N will give better concentration of the energy, but the transform will be more complicated to calculate. It will also be harder to adapt the coder if the source has different statistics in different parts (eg foreground and background in an image or different parts of a music signal). Large transforms can also give rise to more noticable quantization errors.

Typical block size for image coding is 8×8 pixels (JPEG, MPEG, DV)

Typical block sizes for audio coding are 256-2048 samples (Dolby Digital, MPEG AAC, Ogg Vorbis)

Distortion

For orthonormal transforms the distortion in the transform domain will be the same as the distortion in the signal domain.

Assume that we quantize and reconstruct the transform vector to $\hat{\theta}$ and inverse transform to the reconstructed vector $\hat{\mathbf{x}}$. The distortion is then

$$\begin{aligned} D &= \frac{1}{N}||\bar{\mathbf{x}} - \hat{\mathbf{x}}||^2 = \frac{1}{N}(\bar{\mathbf{x}} - \hat{\mathbf{x}})^T(\bar{\mathbf{x}} - \hat{\mathbf{x}}) \\ &= \frac{1}{N}(\mathbf{A}^T\bar{\theta} - \mathbf{A}^T\hat{\theta})^T(\mathbf{A}^T\bar{\theta} - \mathbf{A}^T\hat{\theta}) \\ &= \frac{1}{N}(\bar{\theta} - \hat{\theta})^T\mathbf{A}\mathbf{A}^T(\bar{\theta} - \hat{\theta}) \\ &= \frac{1}{N}(\bar{\theta} - \hat{\theta})^T(\bar{\theta} - \hat{\theta}) = \frac{1}{N}||\bar{\theta} - \hat{\theta}||^2 \end{aligned}$$

The same reasoning also applies for random signals, with expectation.

Zonal coding

In zonal coding (or zonal sampling) we split the transformed vector (or block) into a number of parts (zones). All coefficients in the same zone are coded using the same quantizer and the same source coder.

If we have K zones and zone j has N_j coefficients, we of course have $N_1 + N_2 + \ldots + N_K = N$.

Given that zone j has the rate R_j bits/sample, the average rate R for the whole coder is

$$R = \frac{\sum_{j=1}^{K} N_j \cdot R_j}{N}$$

The zone division, quantization and source coding can be fixed for all blocks, or we can also switch when we needed. This gives us a better possibility to adapt the coder to a varying signal, but it also means that we get more side information to transmit.

Quantization, zonal coding

From now on, assume that we let each transform coefficient be its own zone (ie all $N_j = 1$) and that we keep the coders fixed and don't switch coders between blocks.

Transform component k is quantized and coded to R_k bits, with a resulting distortion D_k. Assuming fine quantization, the distortion can be approximated by

$$D_k \approx c_k \cdot \sigma_k^2 \cdot 2^{-2R_k}$$

We want to find the bit allocation that minimizes the average distortion

$$D = \frac{1}{N} \sum_{k=0}^{N-1} D_k \approx \frac{1}{N} \sum_{k=0}^{N-1} c_k \cdot \sigma_k^2 \cdot 2^{-2R_k}$$

under the condition that the average rate is fixed

$$R = \frac{1}{N} \sum_{i=0}^{N-1} R_k$$

Bit allocation, zonal coding

For simplicity we assume that all transform components have the same type of distribution and that we use the same type of quantization and source coding. Then all c_k are equal. Lagrange optimization gives (see Sayood for details)

$$R_k = R + \frac{1}{2}\log_2 \frac{\sigma_k^2}{\left(\prod_{i=0}^{N-1}\sigma_i^2\right)^{1/N}}$$

Note that this can give some components a negative rate. In that case we set the rate for those components to 0, and redo the bit allocation for the other components, such that the average rate is still R.

For some types of quantization and coding (Lloyd-Max quantization, quantization followed by fixed length coding) we might have the condition that rates should be integers.

Distortion, zonal coding

For optimal bit allocation the distortion for each component (given that our fine quantization assumption still holds) is

$$\begin{aligned} D_k &\approx c \cdot \sigma_k^2 \cdot 2^{-2R_k} = \\ &= c \cdot \sigma_k^2 \cdot 2^{-2R - \log_2 \frac{\sigma_k^2}{(\prod_{i=0}^{N-1} \sigma_i^2)^{1/N}}} = \\ &= c \cdot \sigma_k^2 \cdot \frac{(\prod_{i=0}^{N-1} \sigma_i^2)^{1/N}}{\sigma_k^2} \cdot 2^{-2R} = \\ &= c \cdot (\prod_{i=0}^{N-1} \sigma_i^2)^{1/N} \cdot 2^{-2R} \end{aligned}$$

We will thus get the same distortion for each transform component. The average distortion will of course also take this value.

Threshold coding

For each transform block we tell which transform components that have a magnitude over a threshold value. Only these components are quantized and coded, the rest are set to zero. Which components that are above the threshold needs to be transmitted as side information for every block.

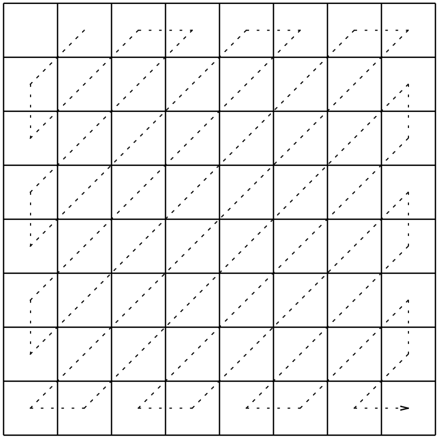
Often runlength coding of the zeros are used for this side information.

For twodimensional transforms a zigzag scanning of the components are usually performed, to get a onedimensional signal, before the runlength coding.

In practice, usually no separate thresholding is done. Instead, the components that are quantized to zero are the ones that are considered to be below the threshold.

Zigzag scanning

Zigzag scanning for 8×8 transform. The DC level in the upper left corner is usually treated separately.



JPEG

ISO standard (1990) for still image coding.

Uses DCT of size 8×8 pixels.

1-4 colour components.

Either 8 or 12 bits per colour components. The common file formats JFIF and EXIF only allow 8 bits per component.

No explicit thresholding, uniform quantization. The step size can be choosen freely for each of the 64 transform components. Typically the high frequency components are quantized harder than the low frequency components.

The source coding is either runlength coding of zeros followed by Huffman coding, or arithmetic coding. Since the arithmetic coder in the standard was protected by several patents, only Huffman coding is used in practice.

JPEG

Image quality is controlled by the choice of the step sizes of the 64 quantizers. Since we can choose them freely and independently of each other, it might be hard to find the best choice of step sizes for a given average rate or a given average distortion.

In order to simplify, most JPEG coders (eg digital cameras) only let the user choose one quality parameter. Each quality parameter will correspond to a pre-chosen matrix of step sizes. A quantization matrix might look like this

16	11	10	16	24	40	51	61
12	12	14	19	26	58	60	55
14	13	16	24	40	57	69	56
14	17	22	29	51	87	80	62
18	22	37	56	68	109	103	77
24	35	55	64	81	104	113	92
49	64	78	87	103	121	120	101
72	92	95	98	112	100	103	99

JPEG, coding of the DC level

The difference d from the DC level in the previous block is coded. The Huffman coding is not done directly on the difference values. Instead a category is formed according to

$$k = \lceil \log(|d| + 1) \rceil$$

Stastics are gathered for all categories and a Huffman code is constructed.
The codeword for a difference d consists of the Huffman codeword for k followed by k extra bits to exactly specify d.

k	d	extra bits
0	0	$-$
1	$-1, 1$	$0, 1$
2	$-3, -2, 2, 3$	$00, 01, 10, 11$
3	$-7, \dots, -4, 4, \dots, 7$	$000, \dots, 011, 100, \dots, 111$
$\vdots$	$\vdots$	$\vdots$

JPEG, coding of other components

The components are ordered in zigzag order. All runs of zeros are replaced by the length of the run (min 0, max 15). Just as for the DC component, we form the category for each non-zero component l as

$$k = \lceil \log(|l| + 1) \rceil$$

A new symbol alphabet is constructed, consisting of pairs (runlength, category). We gather statistics for the pairs and build a Huffman code for the new alphabet. Just as for the DC level, the codeword for each pair is followed by k bits that exactly tells us what value the non-zero component has.

JPEG

In the Huffman code we also have two special symbols, (End Of Block) which is used when all the remaining components in a block are zero and ZRL (Zero Run Length) which is used when we have to code a run of zeros that is longer than 15. ZRL means 16 zeros. For example, a run of 19 zeros followed by category 5 is described as (ZRL)(3,5).

Example image

768 $\times$ 512 pixels, 8 bits/pixel

JPEG, example

One block from the image

Pixel values:

$$\begin{array}{rrrrrrrr}
6 & 13 & 26 & 54 & 45 & -33 & -56 & 12 \\
21 & 23 & 46 & 60 & 24 & -53 & -38 & 22 \\
32 & 47 & 62 & 39 & -15 & -69 & -20 & 33 \\
48 & 52 & 37 & -1 & -54 & -57 & 12 & 20 \\
51 & 30 & -7 & -49 & -61 & -6 & 17 & 31 \\
9 & -22 & -52 & -68 & -18 & 14 & 29 & 65 \\
-42 & -58 & -77 & -32 & 12 & 31 & 71 & 59 \\
-72 & -63 & -25 & 9 & 35 & 86 & 74 & 25
\end{array}$$

128 has been removed from all pixel values, so black is -128 and white is 127.

JPEG, example

After DCT (rounded to integers for clarity)

$$\begin{matrix}
42 & -36 & 68 & -50 & 33 & 0 & -8 & 6 \\
37 & 213 & -35 & -116 & 65 & -20 & -3 & 6 \\
12 & -95 & -143 & 25 & 36 & -11 & 1 & -2 \\
-37 & -25 & 43 & 50 & 20 & -31 & 12 & -3 \\
8 & 24 & -14 & 12 & -33 & 11 & 8 & -11 \\
-12 & -12 & 9 & -7 & 9 & 2 & -21 & 5 \\
4 & 3 & -5 & -5 & -2 & -14 & 15 & 3 \\
0 & 0 & 4 & 1 & -1 & 0 & 0 & -5
\end{matrix}$$

JPEG, example

After quantization with step size 30 for all components (divide with the step size and round to integer)

$$\begin{matrix} 1 & -1 & 2 & -2 & 1 & 0 & 0 & 0 \\ 1 & 7 & -1 & -4 & 2 & -1 & 0 & 0 \\ 0 & -3 & -5 & 1 & 1 & 0 & 0 & 0 \\ -1 & -1 & 1 & 2 & 1 & -1 & 0 & 0 \\ 0 & 1 & 0 & 0 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{matrix}$$

Order in zigzag order (DC component removed)

-1 1 0 7 2 -2 -1 -3 -1 0 -1 -5 -4 1 0 2 1 1 1 0 0 0 0 2 1 -1 0 0 0 0 1 0 0 0 0 0 0 0 -1 -1 0 0 0 0 0 0 0 0 0 0 0 0 0 -1 0 0 0 0 0 0 0 0

JPEG, example

Gather statistics for the DC categories and construct a Huffman code. For the example test image we get the codeword lengths:

category	codeword length
0	2
1	2
2	2
3	3
4	4
5	5
6	5

The DC level in our quantized block is 1, which is category 1. Thus we will use 2+1 bits to code it.

JPEG, example

-1 1 0 7 2 -2 -1 -3 -1 0 -1 -5 -4 1 0 2 1 1 1 0 0 0 0 2 1 -1 0 0 0 0 1 0 0 0 0 0 0 0 -1 -1 0 0 0 0 0 0 0 0 0 0 0 0 0 -1 0 0 0 0 0 0 0 0

Code as pairs (runlength, non-zero component)

(0,-1) (0,1) (1,7) (0,2) (0,-2) (0,-1) (0,-3) (0,-1) (1,-1) (0,-5) (0,-4) (0,1) (1,2) (0,1) (0,1) (0,1) (4,2) (0,1) (0,-1) (4,1) (7,-1) (0,-1) (14,-1) (EOB)

Code the non-zero components as category plus extra bits

(0,1) 0 (0,1) 1 (1,3) 111 (0,2) 10 (0,2) 01 (0,1) 0 (0,2) 00 (0,1) 0 (1,1) 0 (0,3) 010 (0,3) 011 (0,1) 1 (1,2) 10 (0,1) 1 (0,1) 1 (0,1) 1 (4,2) 10 (0,1) 1 (0,1) 1 (4,1) 1 (7,1) 1 (0, 1) 0 (14,1) 0 (EOB)

Do the same for all blocks in the image, gather statistics for the pairs (runlength, category) and construct a Huffman code for them.

JPEG, example

Lengths of Huffman codewords (rows categories 1-7, columns runlengths 0-15):

2	5	6	7	6	6	5	4	7	8	10	11	11	10	9	8
4	7	8	9	8	8	6	7	12	-	-	-	-	14	14	14
6	10	10	9	10	9	8	9	-	-	-	-	-	-	-	-
5	12	12	12	12	12	10	12	-	-	-	-	-	-	-	-
4	-	-	-	-	15	15	-	-	-	-	-	-	-	-	-
4	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
10	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

EOB is coded with 2 bits, ZRL with 8 bits.

For our block we will need in total 3 bits for the DC level and 125 bits for the AC components.

JPEG, example

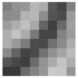
The decoder recreates the following transform block (multiply decoded components with the step sizes)

$$\begin{array}{rrrrrrrr}
30 & -30 & 60 & -60 & 30 & 0 & 0 & 0 \\
30 & 210 & -30 & -120 & 60 & -30 & 0 & 0 \\
0 & -90 & -150 & 30 & 30 & 0 & 0 & 0 \\
-30 & -30 & 30 & 60 & 30 & -30 & 0 & 0 \\
0 & 30 & 0 & 0 & -30 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & -30 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{array}$$

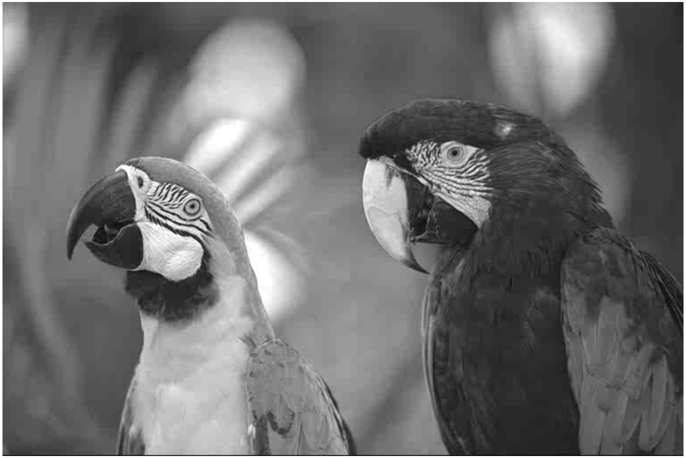
JPEG, example

The block is then inverse transformed to the following block

$$\begin{array}{rrrrrrrr}
-1 & 17 & 22 & 54 & 44 & -41 & -59 & 0 \\
12 & 14 & 51 & 55 & 13 & -45 & -52 & 26 \\
28 & 47 & 63 & 41 & -28 & -65 & -22 & 32 \\
46 & 67 & 44 & -6 & -59 & -59 & 2 & 27 \\
47 & 26 & -4 & -53 & -55 & -16 & 9 & 43 \\
10 & -29 & -61 & -54 & -14 & 18 & 36 & 59 \\
-43 & -57 & -66 & -21 & 19 & 43 & 67 & 45 \\
-73 & -71 & -22 & 4 & 22 & 73 & 71 & 20
\end{array}$$

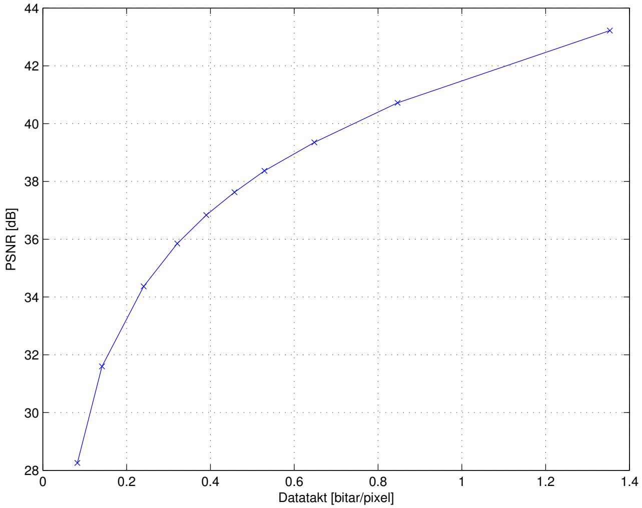
which looks like 

Decoded image


0.35 bits/pixel

JPEG coding

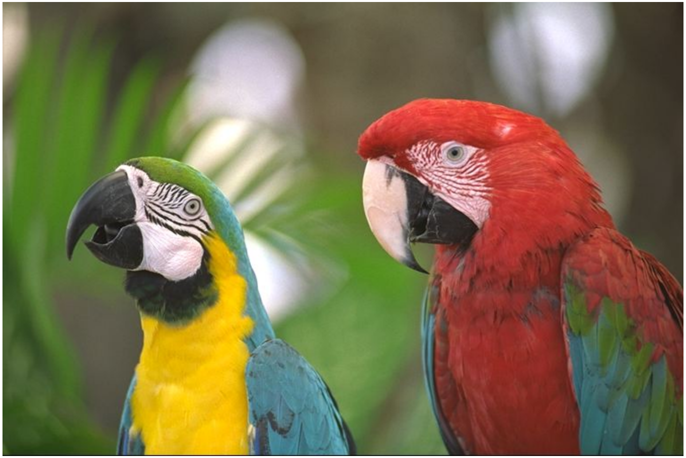
PSNR as a function of the rate by JPEG-coding of the grayscale test image.



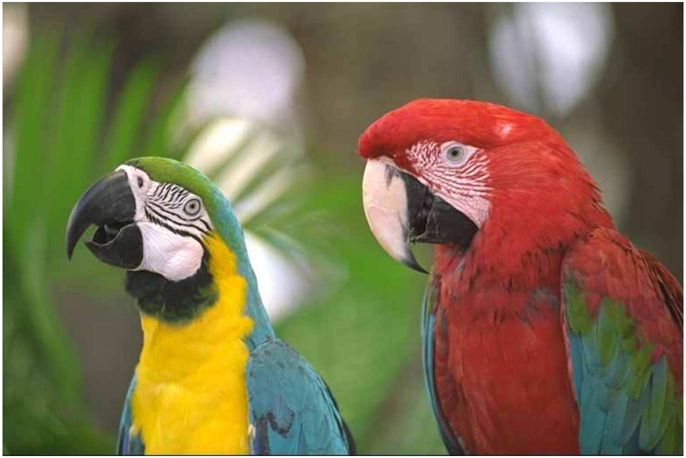
Original colour image

24 bits/pixel

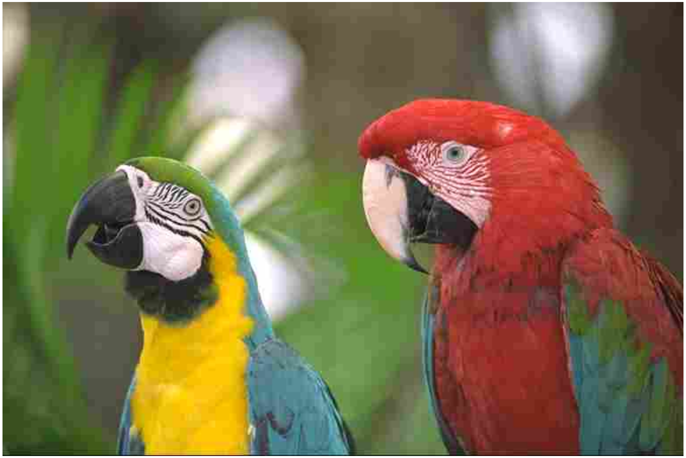
JPEG


0.96 bits/pixel (compression ratio 25)

JPEG

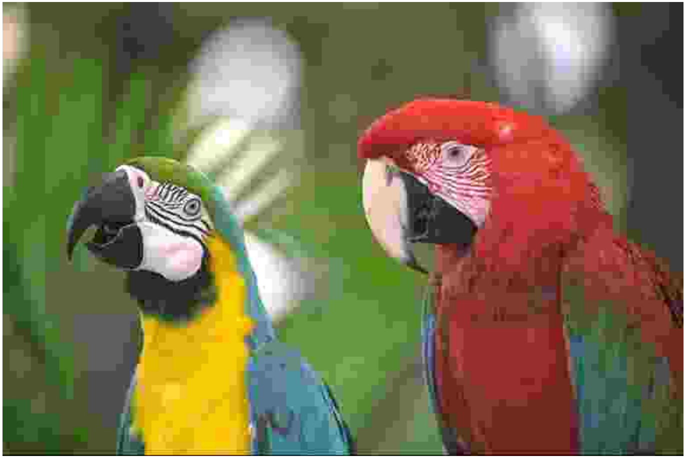

0.48 bits/pixel (compression ratio 50)

JPEG



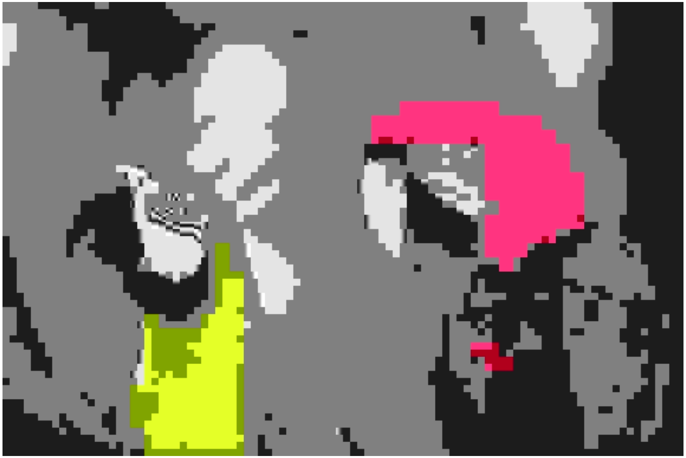
0.24 bits/pixel (compression ratio 100)

JPEG



0.12 bits/pixel (compression ratio 200)

JPEG


0.06 bits/pixel (compression ratio 400)