Ashok C. Popat
Google, Inc.
March 28, 2007

Image Compression: Part 2

Ashok C. Popat
Google, Inc.
March 28, 2007

Image Compression

- Represent an image (a rectangular array of pixels) using as few bits as possible, while still allowing sufficiently faithful reproduction from those bits

Image Compression

- Represent an image (a rectangular array of pixels) using as few bits as possible, while still allowing sufficiently faithful reproduction from those bits
- How well can we expect to do?
- How might one compress images if only an approximate replica of the image is required?
- How might one compress images if a perfect replica is required?

Image Compression

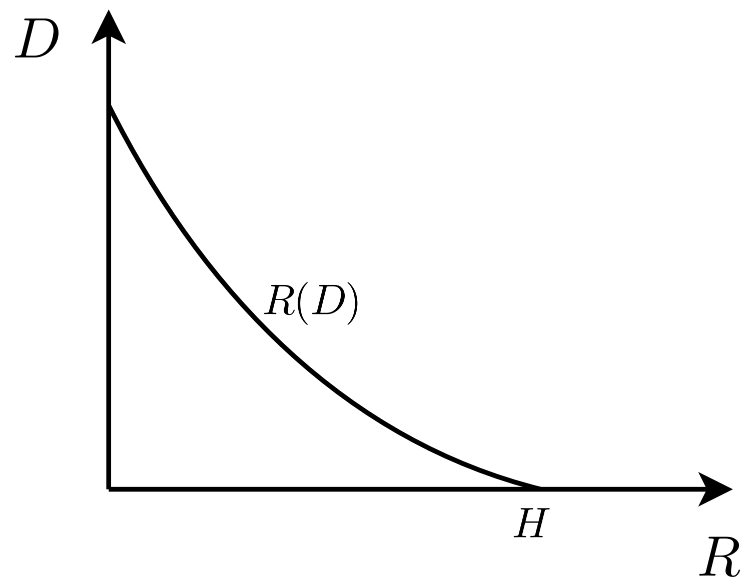
- Represent an image (a rectangular array of pixels) using as few bits as possible, while still allowing sufficiently faithful reproduction from those bits
- How well can we expect to do?
- How might one compress images if only an approximate replica of the image is required?
- How might one compress images if a perfect replica is required?
- Brief review of March 7 talk; pick up where left off, then move to lossless compression

Evaluating quality of reconstructed image

$$\mathrm{PSNR} = 10 \log_{10} \frac{255^2}{\sigma_{\mathrm{err}}^2}$$

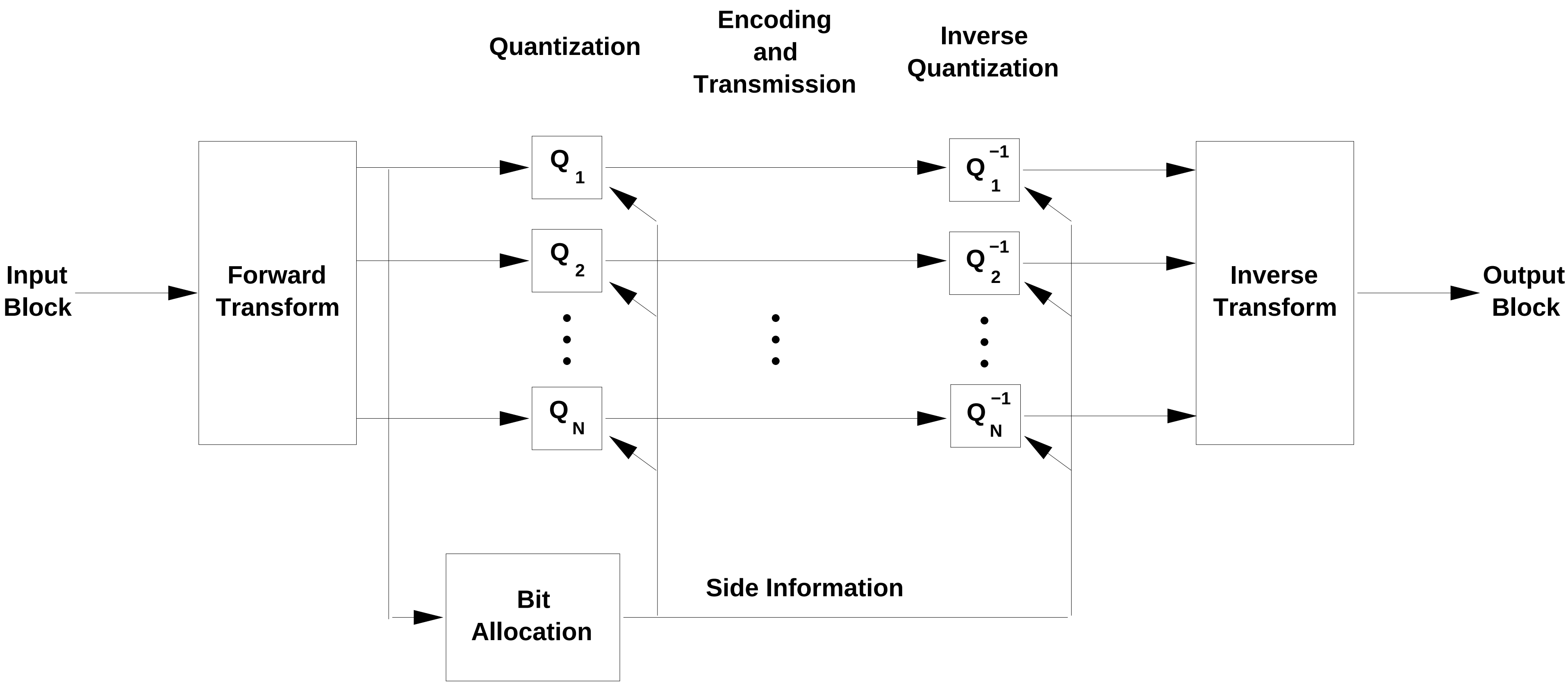
Theoretical limit: rate-distortion bound

$$R(D^*) = \min_{P(Y|X): D < D^*} \sum_{X,Y} P(X,Y) \log_2 \frac{P(Y|X)}{P(Y)}$$



- Can't do better than $R(D)$, assuming the model is right and the distortion measure makes sense.

Transform Coding Block



Choice of transform

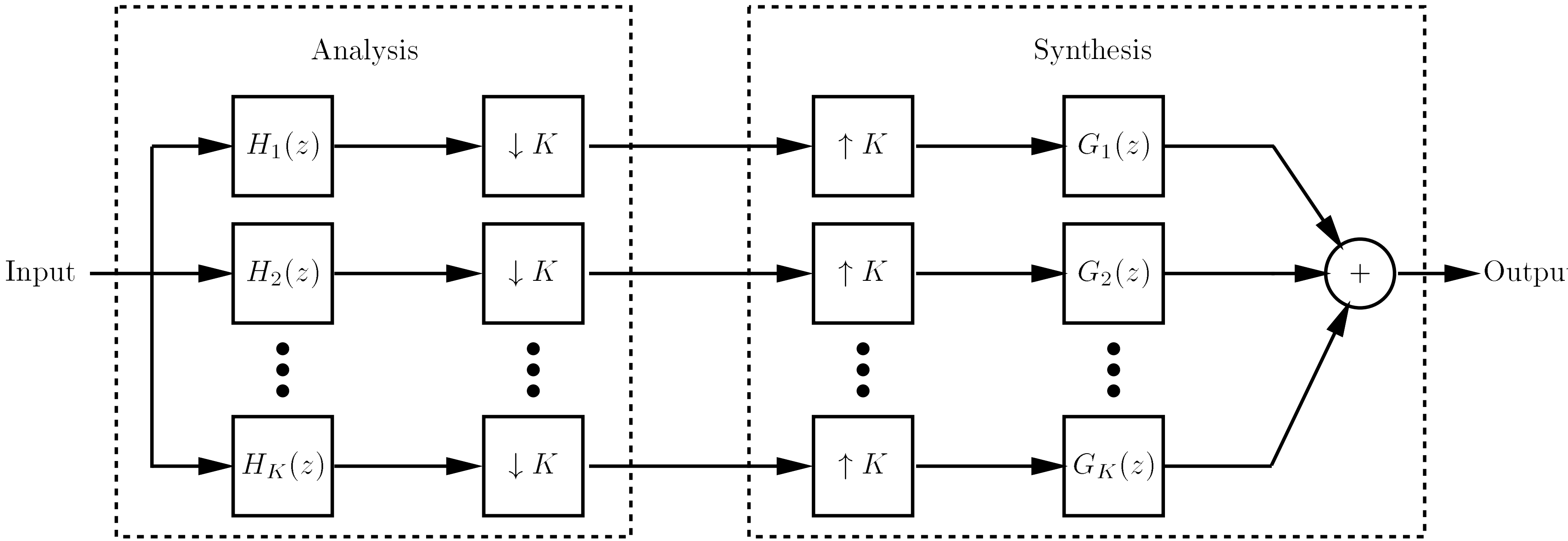
- Transform: linear, orthogonal
- Karhounen-Loeve transform is data-dependent, relatively hard to compute; decoder needs to be told what it is
- Discrete cosine transform (DCT) provides good energy compaction for positively correlated sources; used in JPEG
- DCT can be interpreted in terms of critically sampled filter bank

Quantization and Bit Allocation

- Bit allocation: After transform has compacted energy into few components, allocate the available bits to the most important components

- Implicit rate allocation: use a fixed-step-size scalar quantizer followed by entropy coding
- Coefficients that have high energy will use more quantizer output levels, increasing bit usage
- Last time: implicit rate allocation nearly optimal

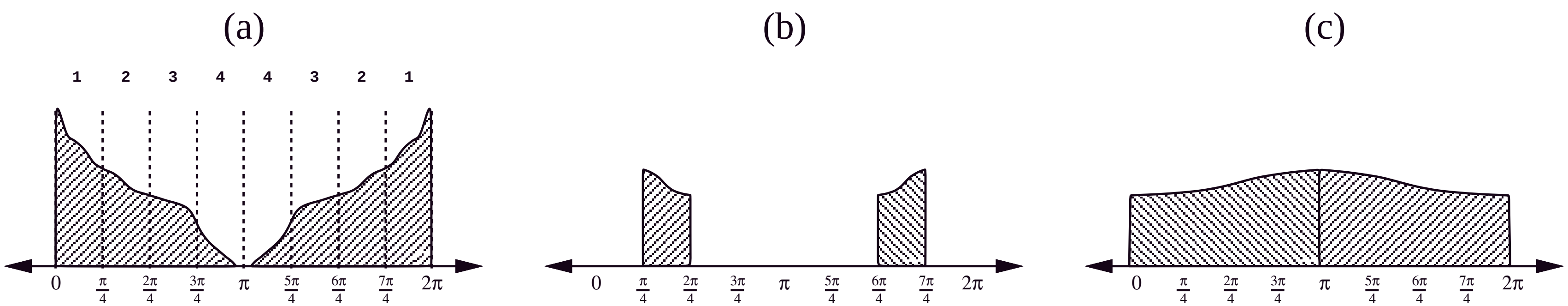
Critically sampled multi-rate filter bank



- Separable processing of vertical and horizontal
- When convolution kernels have length not greater than K, corresponds to a block transform.
- When filter length $>$ number of subbands $= K$, implements a "lapped orthogonal transform" – blocks overlap

Motivation in terms of power spectral densities

- Uncorrelated $\longleftrightarrow$ flat power spectral density
- Highly correlated $\longleftrightarrow$ highly non-flat PSD
- Suggests splitting up the spectrum and allocating more bits to the higher-energy segments

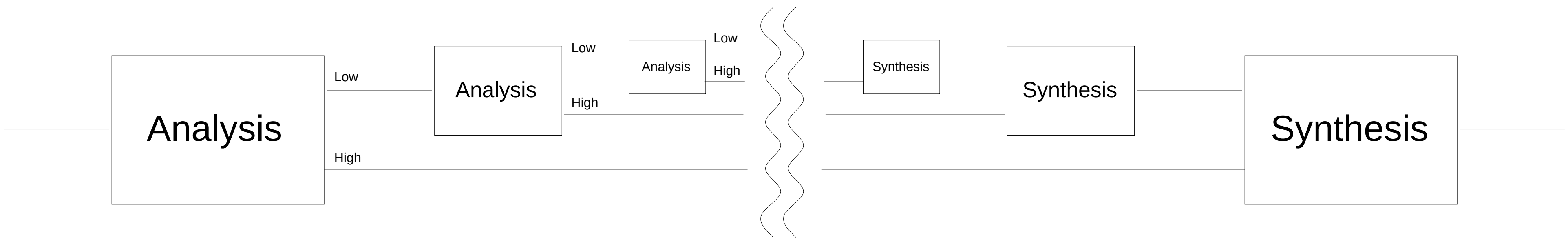


Two Opportunities for Improving Block-Transform Coding

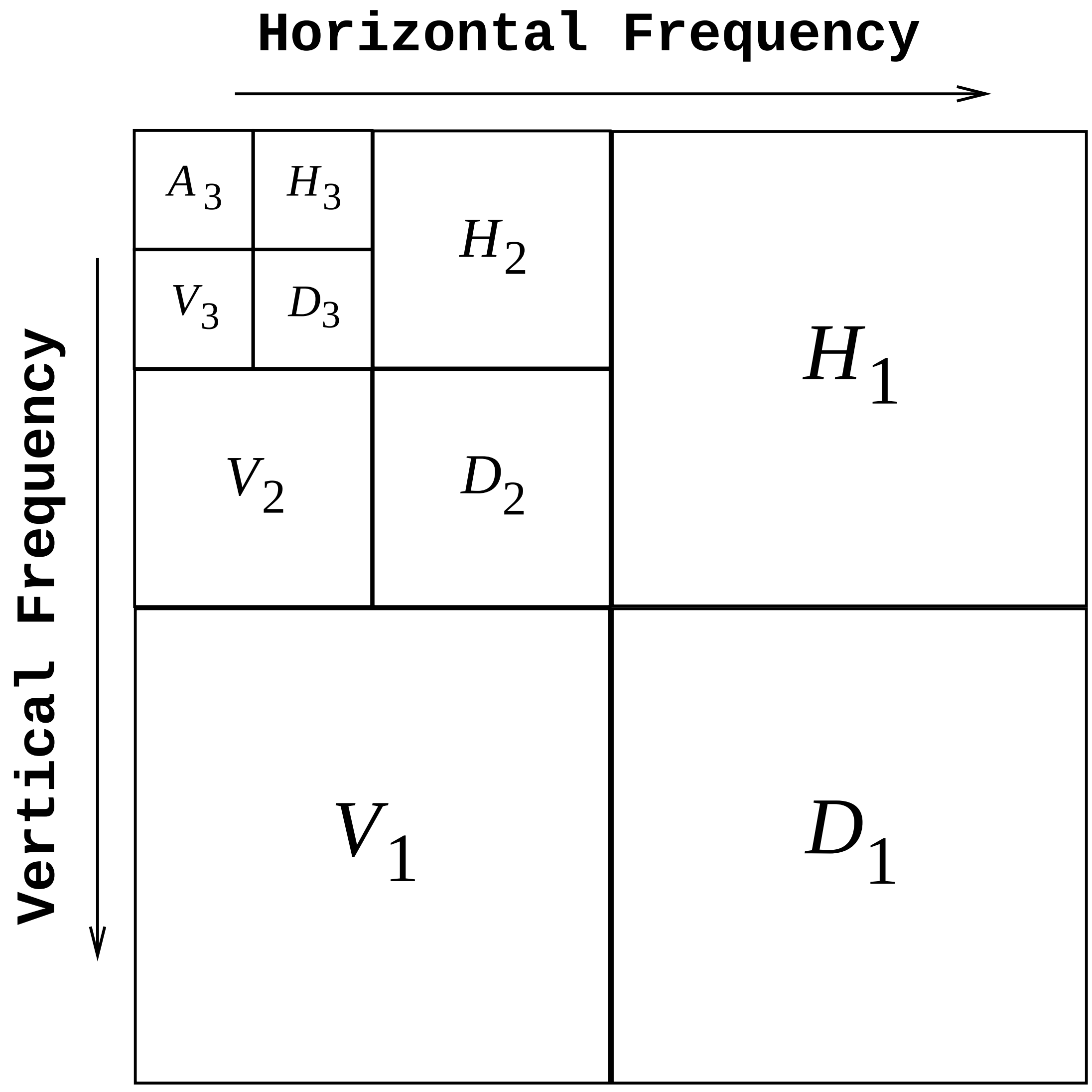
- Allowing filter kernels to extend beyond block boundaries can mitigate blocking artifacts.
- Using non-uniform sub-band decompositions can provide spectral resolution where needed (low frequency regions; interiors of objects) and spatial resolution where needed (high-frequency edges).

Tree-Structured Filter Banks

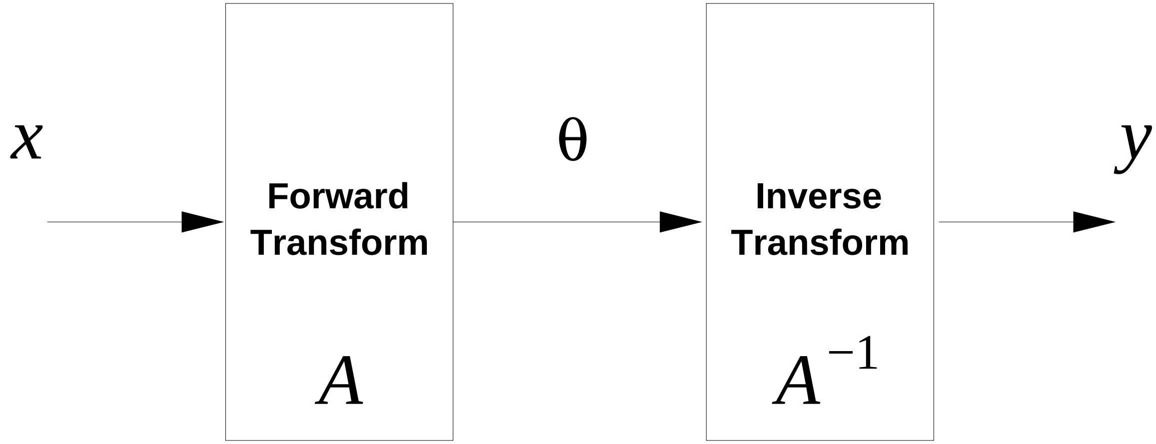
- Recursively subdivide lowest-frequency subband
- Allows high-frequency bands to have short spatial support (e.g., to analyze edges) while allowing low-frequency bands to have long spatial support (e.g., to compress low-activity/low-contrast regions effectively)
- Sometimes called quadrature-mirror filters, discrete wavelet transform



Separable Application to Images

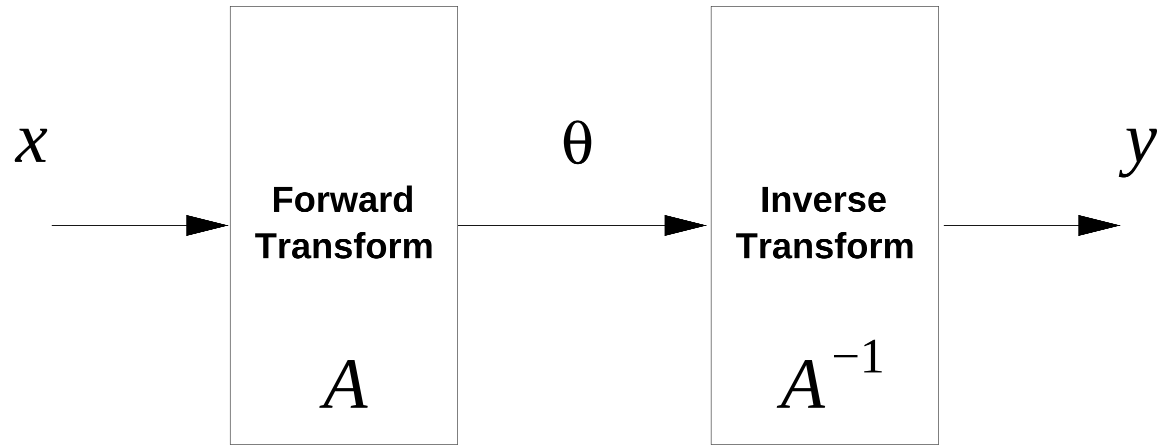


Reflections on Transform Coding



Does the transform make the input more compressible?

Reflections on Transform Coding

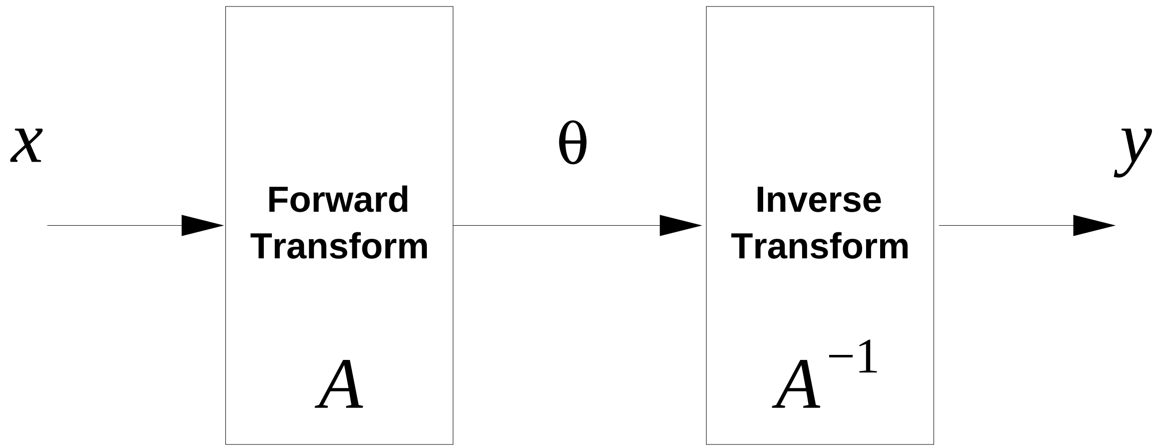


Does the transform make the input more compressible?

Given that the transform is invertible, how does the rate-distortion function of θ compare with that of x?

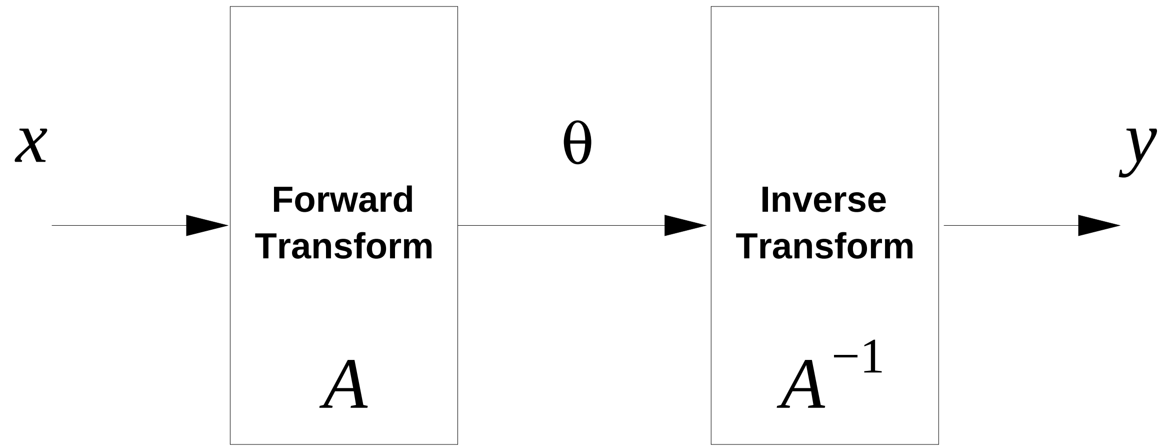
In light of this, what is the role of the transform?

Reflections on Transform Coding



Transform does not compress; *it simplifies subsequent compression*

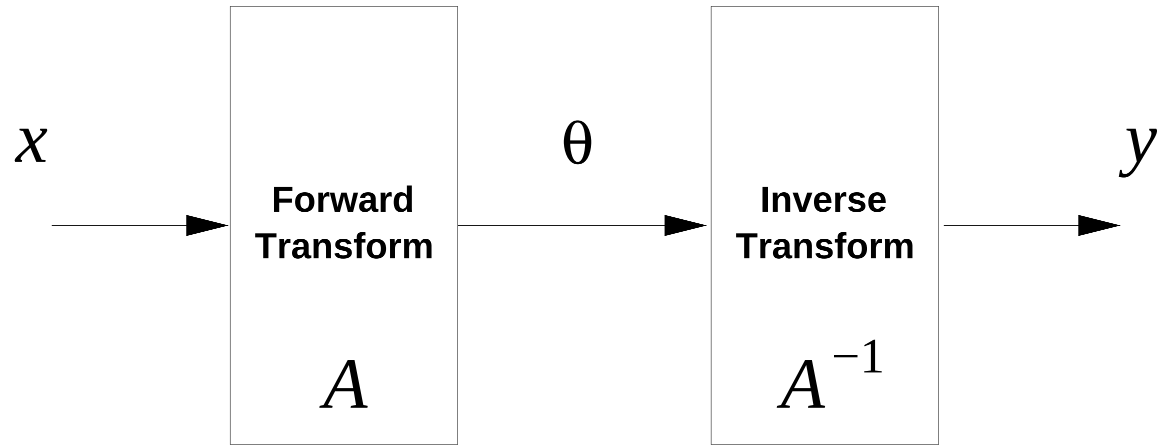
Reflections on Transform Coding



Transform does not compress; *it simplifies subsequent compression*

Transform puts signal's energy in predictable places

Reflections on Transform Coding

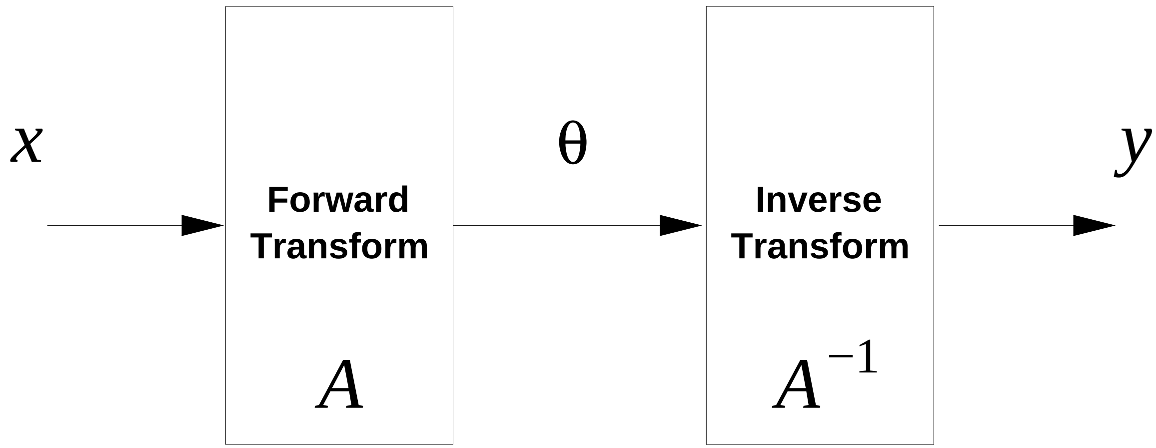


Transform does not compress; *it simplifies subsequent compression*

Transform puts signal's energy in predictable places

Transform enables simple scalar quantization to be effective on the transformed data

Reflections on Transform Coding



Transform does not compress; *it simplifies subsequent compression*

Transform puts signal's energy in predictable places

Transform enables simple scalar quantization to be effective on the transformed data

What if I'm willing to do something more complex?

Vector quantization

- Treat an (e.g.) 8×8 block as a point in 64-dimensional space
- Build a "codebook" of N reproduction vectors
- Encode each input point using $\lceil \log_2 N \rceil$ bits (fixed-rate), or
- Use variable-rate coding

Fixed-rate codebook design (Lloyd, k-means)

1. Initialize codebook
2. Assign example points to nearest codebook entry
3. Re-compute codebook entries as centroids of assigned points
4. Return to Step 2

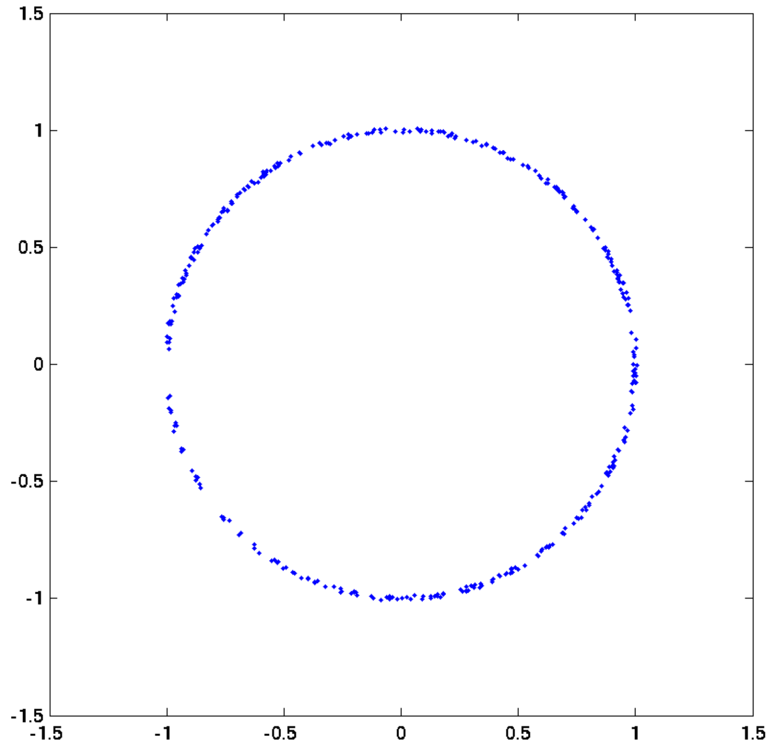
Vector quantization

- Does not figure prominently in standards
- May be used in proprietary schemes (interest in hierarchical VQ re-invigorated around 1997 for video)
- Related to cluster analysis and classification / regression trees
- Unlike transform coding, can approach the RDF

Why not Vector Quantize Transformed Image?

- Why is scalar quantization effective on transform coefficients?
- Components of the Vector Quantization Advantage: dependence (most), PDF shape (some), space-filling (some)
- Transform coefficients are uncorrelated, both spatially and across frequency band
- PDF shape gain small over entropy-constrained scalar quantization
- Is there any role for Vector Quantization in transform/subband/wavelet coding?

Exploiting Nonlinear Statistical Dependence



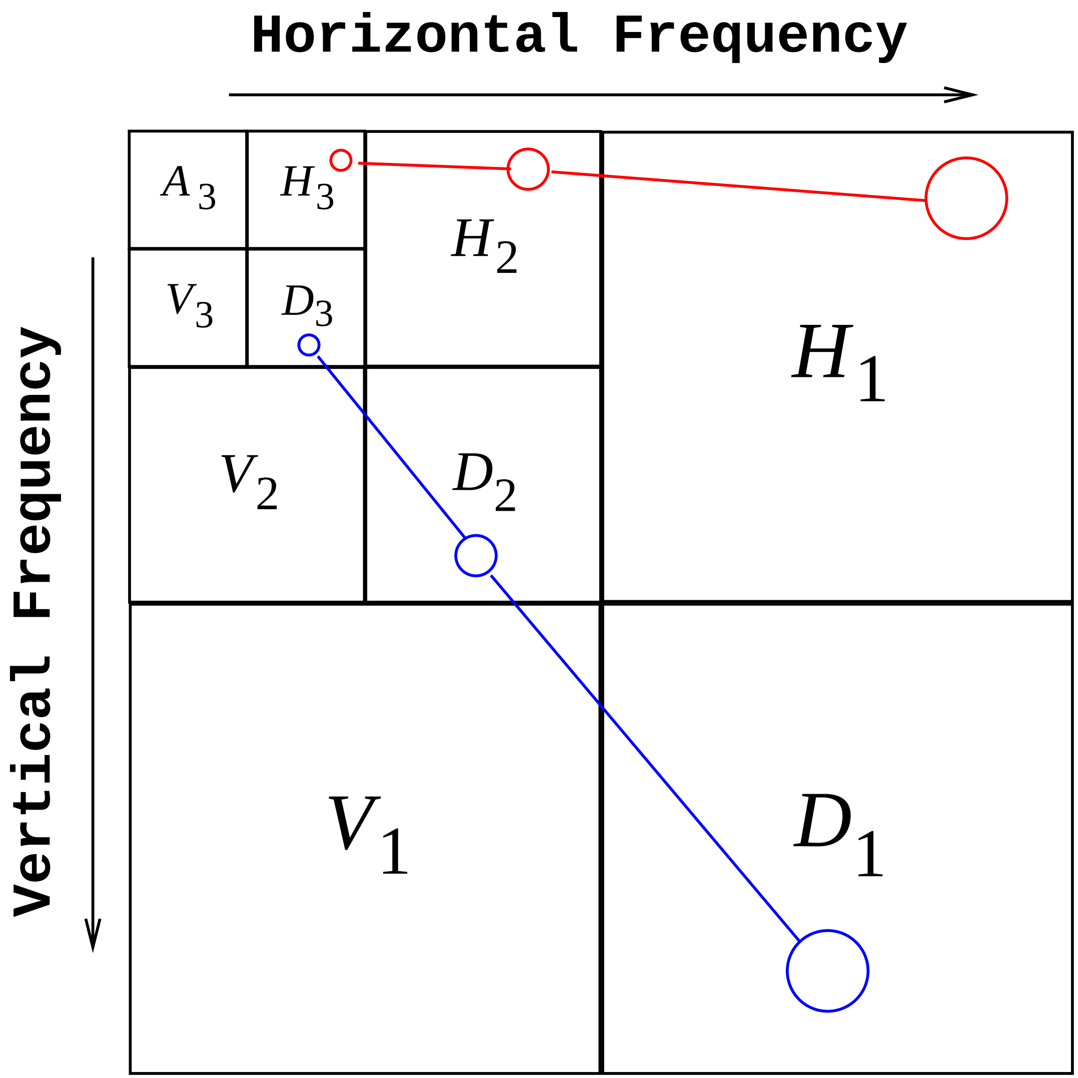
- Orthogonal linear transform (rotation) will not make the regularity amenable to scalar quantization
- Vector quantization would be quite effective here

Exploiting Nonlinear Statistical Dependence

Three-by-three subband decomposition:

- Subbands are uncorrelated, but obviously dependent
- Suggests that there is potential benefit in jointly coding transform coefficients

Predicting Inactivity across Subbands



Embedded Zerotrees Wavelet Coding

- Described by Jerry Shapiro, "Embedded image coding using zerotrees of wavelet coefficients." *IEEE Transactions on Signal Processing,* 41(12):3445–3462, Dec. 1993.
- Several neat ideas in one paper
- Most important contribution: effective means of exploiting nonlinear statistical dependence among wavelet coefficients
- Exploits following empirical finding: inactive spatial regions of non-DC low-resolution subbands are likely to remain inactive in higher-resolution counterparts

Lossless compression

- Decompressed image must be bit-for-bit identical to the original
- Why would someone want lossless compression?
- Why might lossless compression be hard?

Lossy versus lossless compression of text images

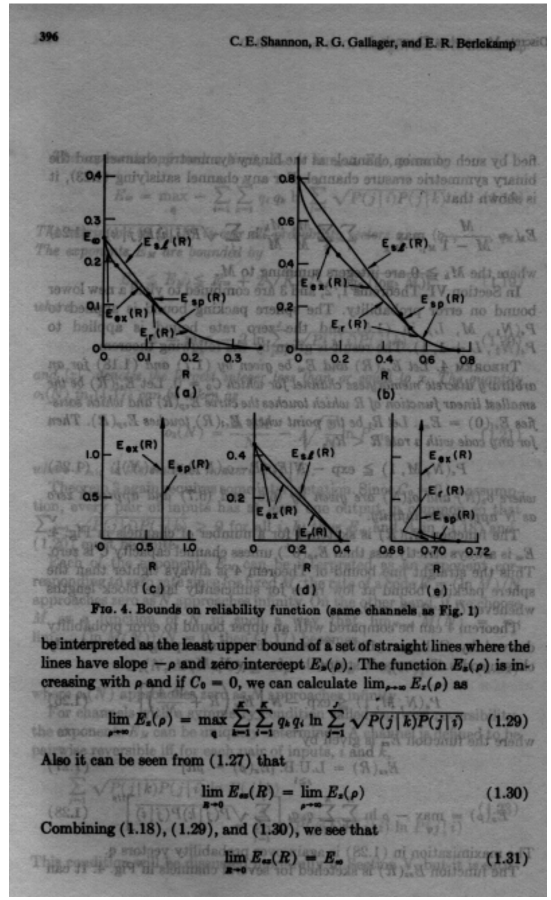
- Original (lossless)

Lossy versus lossless compression of text images

- Original (lossless)

- Result of JBIG2 (lossy) compression

Why might lossless compression be hard?



396 C. E. Shannon, R. G. Gallager, and E. R. Berlekamp

FIG. 4. Bounds on reliability function (same channels as Fig. 1)

be interpreted as the least upper bound of a set of straight lines where the lines have slope $-\rho$ and zero intercept $E_x(\rho)$. The function $E_x(\rho)$ is increasing with ρ and if $C_0 = 0$, we can calculate $\lim_{\rho\to\infty} E_x(\rho)$ as

$$\lim_{\rho\to\infty} E_x(\rho) = \max_{q} \sum_{k=1}^{K}\sum_{i=1}^{K} q_k q_i \ln \sum_{j=1}^{J} \sqrt{P(j|k)P(j|i)} \qquad (1.29)$$

Also it can be seen from (1.27) that

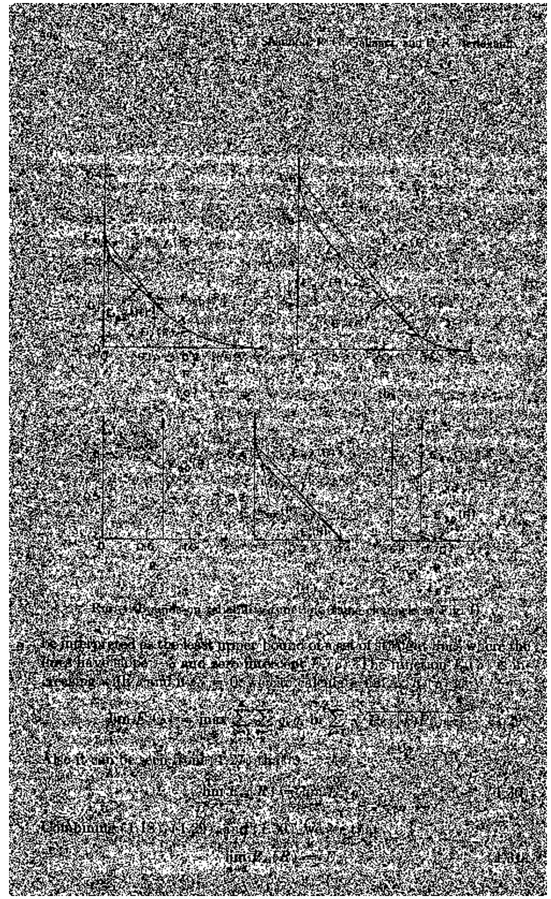
$$\lim_{R\to 0} E_{ex}(R) = \lim_{\rho\to\infty} E_x(\rho) \qquad (1.30)$$

Combining (1.18), (1.29), and (1.30), we see that

$$\lim_{R\to 0} E_{ex}(R) = E_\infty \qquad (1.31)$$

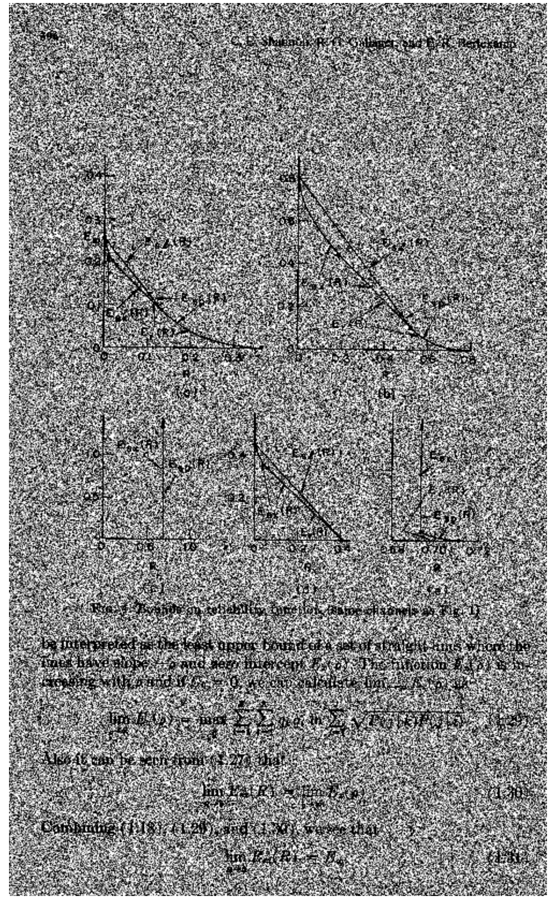
Original

Why might lossless compression be hard?



Least significant bit plane

Why might lossless compression be hard?



Two least significant bits

Why might lossless compression be hard?

396 C. E. Shannon, R. G. Gallager, and E. R. Berlekamp

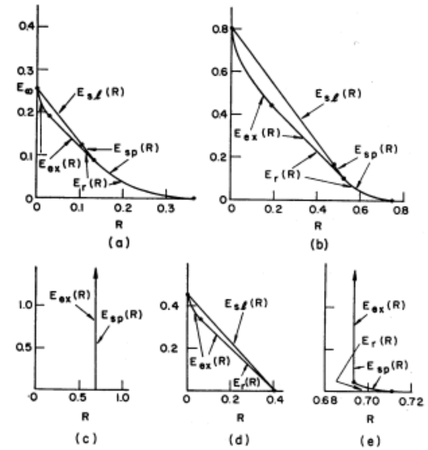


FIG. 4. Bounds on reliability function (same channels as Fig. 1)

be interpreted as the least upper bound of a set of straight lines where the lines have slope $-\rho$ and zero intercept $E_x(\rho)$. The function $E_x(\rho)$ is increasing with ρ and if $C_0 = 0$, we can calculate $\lim_{\rho\to\infty} E_x(\rho)$ as

$$\lim_{\rho\to\infty} E_x(\rho) = \max_{q} \sum_{k=1}^{K} \sum_{i=1}^{K} q_k q_i \ln \sum_{j=1}^{J} \sqrt{P(j\mid k)P(j\mid i)} \quad (1.29)$$

Also it can be seen from (1.27) that

$$\lim_{R\to 0} E_{ex}(R) = \lim_{\rho\to\infty} E_x(\rho) \quad (1.30)$$

Combining (1.18), (1.29), and (1.30), we see that

$$\lim_{R\to 0} E_{ex}(R) = E_\infty \quad (1.31)$$

Binarized

Lossless compression basics

- More probable images get short codewords;
- Less probable images get longer codewords.

Lossless compression basics

- More probable images get short codewords;
- Less probable images get longer codewords.
- Why not use short codewords for everything?

Lossless compression basics

- More probable images get short codewords;
- Less probable images get longer codewords.
- Why not use short codewords for everything?
- How long is the ideal codeword for image I?

Lossless compression basics

- More probable images get short codewords;
- Less probable images get longer codewords.
- Why not use short codewords for everything?
- How long is the ideal codeword for image I?
- Best code length: $-\log_2 P(I)$

Lossless compression basics

- More probable images get short codewords;
- Less probable images get longer codewords.
- Why not use short codewords for everything?
- How long is the ideal codeword for image I?
- Best code length: $-\log_2 P(I)$
- On average: $-\sum_I P(I) \log_2 P(I)$

Lossless compression basics

- More probable images get short codewords;
- Less probable images get longer codewords.
- Why not use short codewords for everything?
- How long is the ideal codeword for image I?
- Best code length: $-\log_2 P(I)$
- On average: $-\sum_I P(I) \log_2 P(I)$
- Given a probability model for images, how can we design an encoder that approximates this ideal?

Typical sets and the law of large numbers

- Of all of the events that are *possible*, a smaller set of has almost all of the probability
- Effect becomes stronger when working with large blocks (law of large numbers)
- Example: flip a biased coin (Pr(Head)=0.1) 1000 times. The set of sequences that have about 100 heads will have almost all of the probability. They are equally likely and need $\log_2 \frac{1000!}{100!900!}$ bits to index.
- From Stirling's approximation:

$$\begin{aligned} \log_2 \frac{n!}{k!(n-k)!} &\approx -k \log_2 \frac{k}{n} - (n-k) \log_2 \frac{n-k}{n} \\ &= nH(\frac{k}{n}) \end{aligned}$$

Kraft Inequality

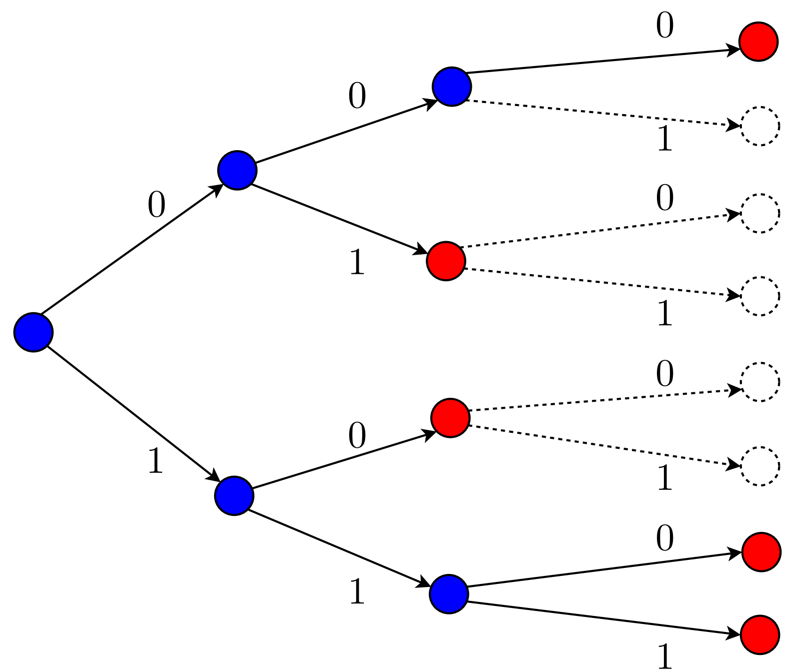
- Why not use short codewords for everything?

Kraft Inequality

- Why not use short codewords for everything?
- Prefix-free: no codeword is a prefix of another
- Example: 000, 01, 10, 110, 111
- Is this a good code?

Kraft Inequality (cont.)

- Example: 000, 01, 10, 110, 111



- Let l_i be the length of the codeword i. Then

$$\sum_i 2^{-l_i} \leq 1$$

Lossless compression of sequential data

- Fixed length to fixed length (no compression)
- Fixed to variable (Huffman)
- Variable to fixed (Tunstall; run-length)
- Variable to variable (Run-length / Huffman; Arithmetic)

Arithmetic coding

- Decompose probability of entity $\mathbf{x}$ to be compressed into a product of conditional probabilities of its constituents $x_1, \ldots, x_N$

Arithmetic coding

- Decompose probability of entity $\mathbf{x}$ to be compressed into a product of conditional probabilities of its constituents $x_1, \ldots, x_N$
- Divide the unit interval into subintervals of width $P(x_1)$ for each possible value of x_1. Commit to the subinterval corresponding to the value of x_1 that actually occurs.

Arithmetic coding

- Decompose probability of entity $\mathbf{x}$ to be compressed into a product of conditional probabilities of its constituents $x_1, \ldots, x_N$
- Divide the unit interval into subintervals of width $P(x_1)$ for each possible value of x_1. Commit to the subinterval corresponding to the value of x_1 that actually occurs.
- Subdivide the new interval into subintervals of width $P(x_1)P(x_2|x_1)$ for each possible value of x_2. Commit to the subinterval corresponding to the value of x_2 that actually occurs.

Arithmetic coding

- Decompose probability of entity $\mathbf{x}$ to be compressed into a product of conditional probabilities of its constituents $x_1, \ldots, x_N$
- Divide the unit interval into subintervals of width $P(x_1)$ for each possible value of x_1. Commit to the subinterval corresponding to the value of x_1 that actually occurs.
- Subdivide the new interval into subintervals of width $P(x_1)P(x_2|x_1)$ for each possible value of x_2. Commit to the subinterval corresponding to the value of x_2 that actually occurs.
- Continue until the entire sequence has been processed. The width of the final subinterval will be $P(\mathbf{x})$.

Arithmetic coding

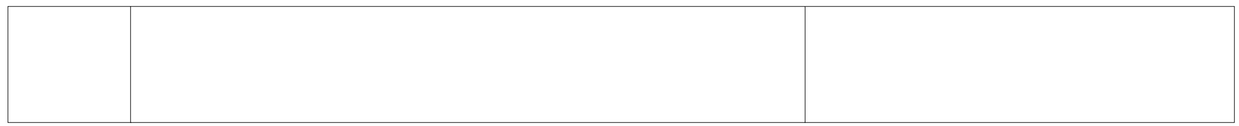
- Decompose probability of entity $\mathbf{x}$ to be compressed into a product of conditional probabilities of its constituents $x_1, \ldots, x_N$
- Divide the unit interval into subintervals of width $P(x_1)$ for each possible value of x_1. Commit to the subinterval corresponding to the value of x_1 that actually occurs.
- Subdivide the new interval into subintervals of width $P(x_1)P(x_2|x_1)$ for each possible value of x_2. Commit to the subinterval corresponding to the value of x_2 that actually occurs.
- Continue until the entire sequence has been processed. The width of the final subinterval will be $P(\mathbf{x})$.
- This final subinterval will contain at least one binary fraction that can be written exactly using $L = \lceil -\log_2 P(\mathbf{x}) \rceil$ bits, which can be used to uniquely identify $\mathbf{x}$

Arithmetic coding

- Decompose probability of entity $\mathbf{x}$ to be compressed into a product of conditional probabilities of its constituents $x_1, \ldots, x_N$
- Divide the unit interval into subintervals of width $P(x_1)$ for each possible value of x_1. Commit to the subinterval corresponding to the value of x_1 that actually occurs.
- Subdivide the new interval into subintervals of width $P(x_1)P(x_2|x_1)$ for each possible value of x_2. Commit to the subinterval corresponding to the value of x_2 that actually occurs.
- Continue until the entire sequence has been processed. The width of the final subinterval will be $P(\mathbf{x})$.
- This final subinterval will contain at least one binary fraction that can be written exactly using $L = \lceil -\log_2 P(\mathbf{x}) \rceil$ bits, which can be used to uniquely identify $\mathbf{x}$
- Can build up this identifying number using sequential arithmetic operations

Arithmetic coding (cont.)

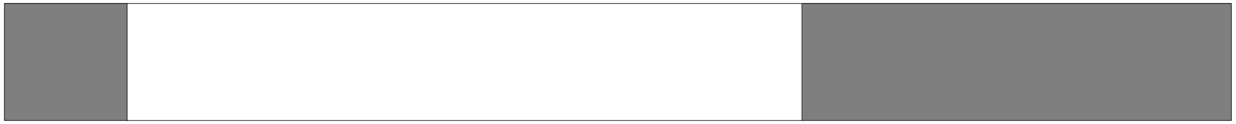
Encoding *bac*. . .



subdivide
Width $= 1.0$

Arithmetic coding (cont.)

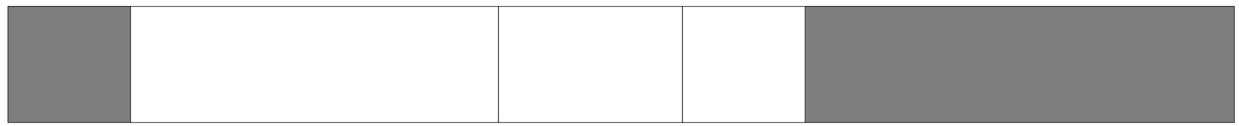
Encoding *bac*. . .



commit to $x_1 = b$
Width $= P(b)$

Arithmetic coding (cont.)

Encoding *bac*. . .



subdivide

Width $= P(b)$

Arithmetic coding (cont.)

Encoding *bac*. . .

commit to $x_2 = a$

Width $= P(a|b)P(b)$

Arithmetic coding (cont.)

Encoding *bac*...

subdivide

Width $= P(a|b)P(b)$

Arithmetic coding (cont.)

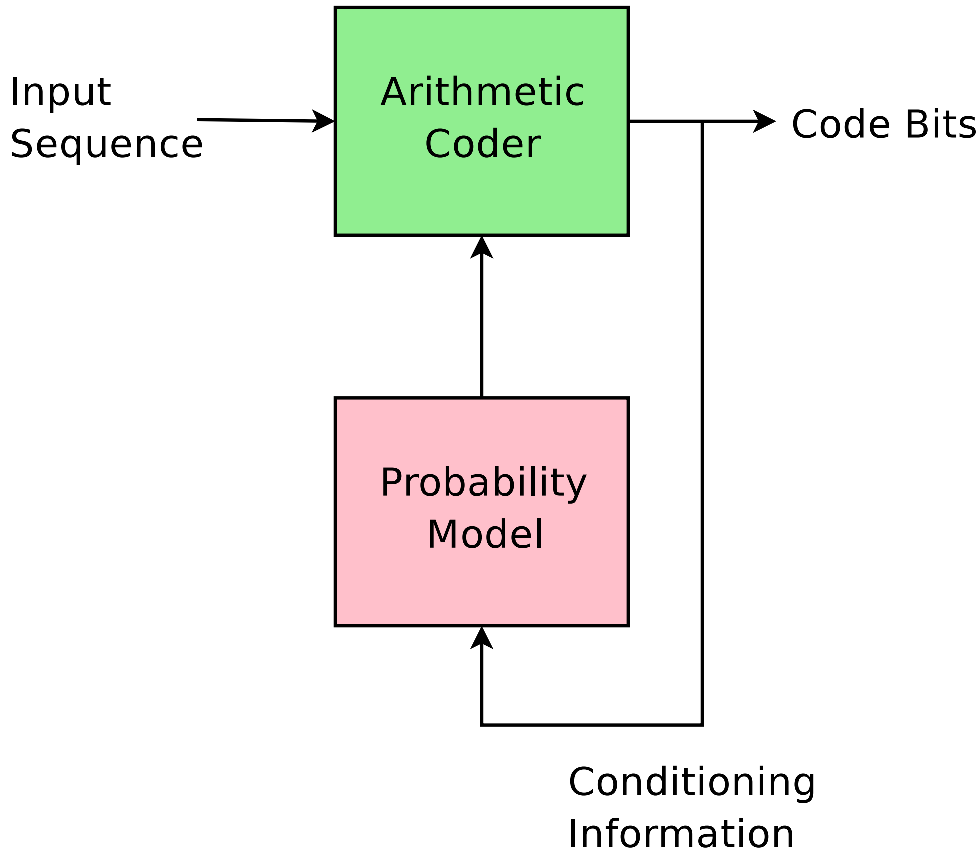
Encoding *bac*. . .

commit to $x_3 = c$
Width $= P(c|ba)P(a|b)P(b)$

Arithmetic coding (cont.)

- In practice, need to prevent numerical underflow, allow FIFO operation, prevent carry propagation, etc.
- Separates modeling (coming up with the probabilities) from the actual encoding.
- Patents on the basic technique have expired.
- Compression efficiency depends on predictive accuracy of model

Arithmetic coding (cont.)



Can we do better by "cooking" the probabilities?

- Assume true probabilities are $P(x)$ and assumed probabilities are $Q(x)$
- Average code length will be about

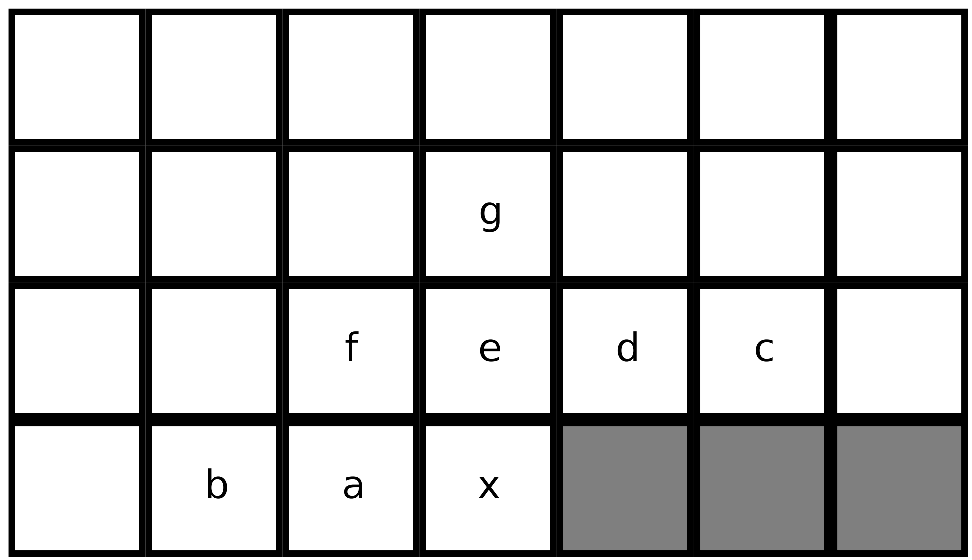
$$-\sum_x P(x) \log_2 Q(x) \geq -\sum_x P(x) \log_2 P(x)$$

by the non-negativity of relative entropy

Context coding

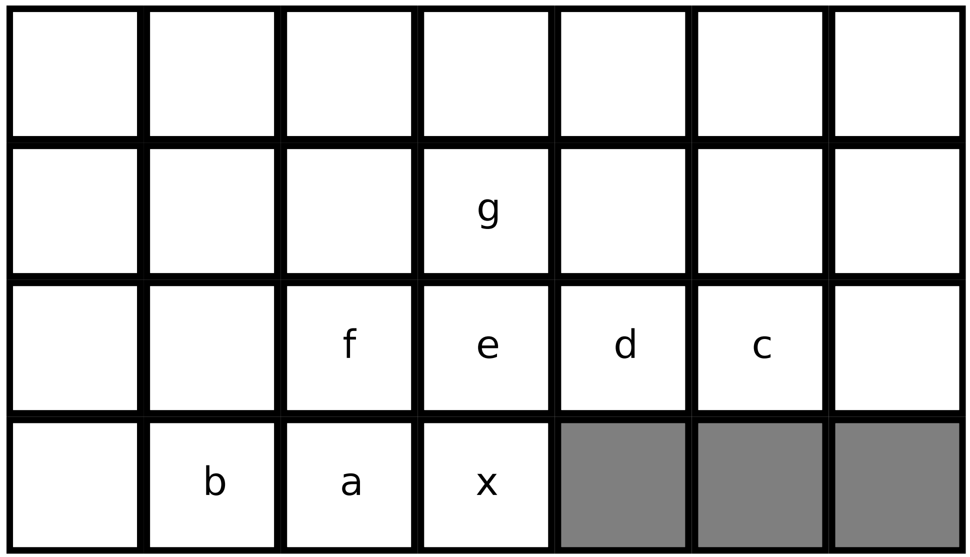
- Arithmetic-code pixels in raster order
- Use probability estimates conditioned on previous values

$$P(x|a, b, c, d, e, f, g)$$



Context coding (cont.)

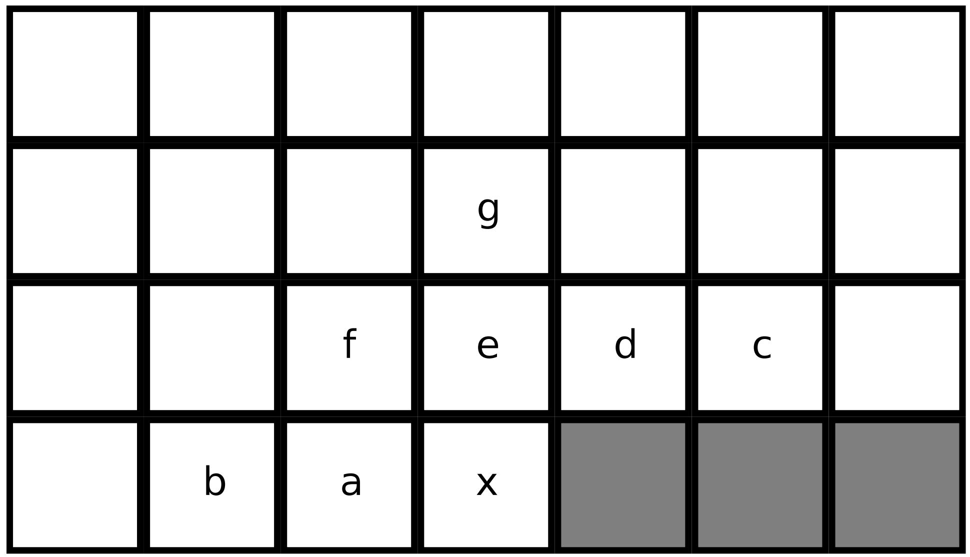
$$P(x|a, b, c, d, e, f, g)$$



- Why not use very large contexts?

Context coding (cont.)

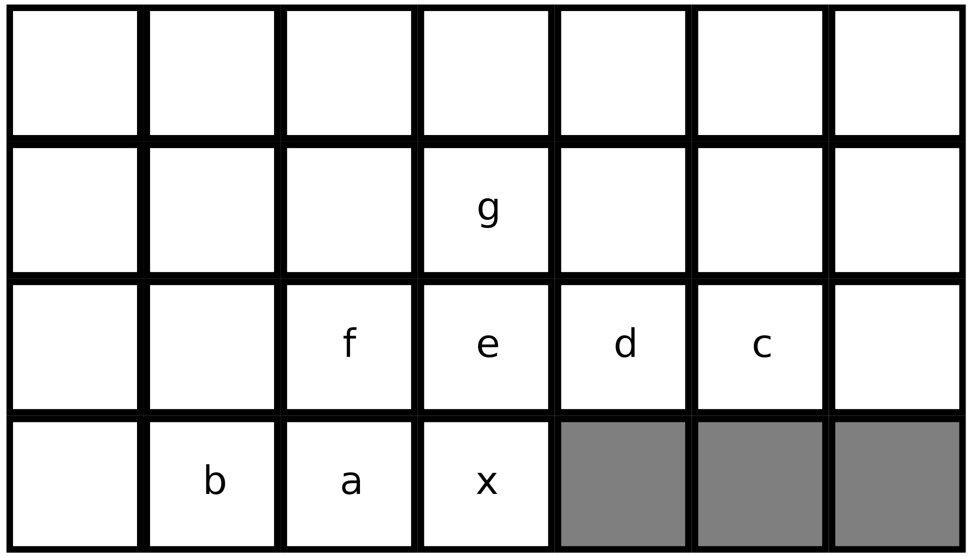
$$P(x|a, b, c, d, e, f, g)$$



- Why not use very large contexts?
- Why not use very small contexts?

Context coding (cont.)

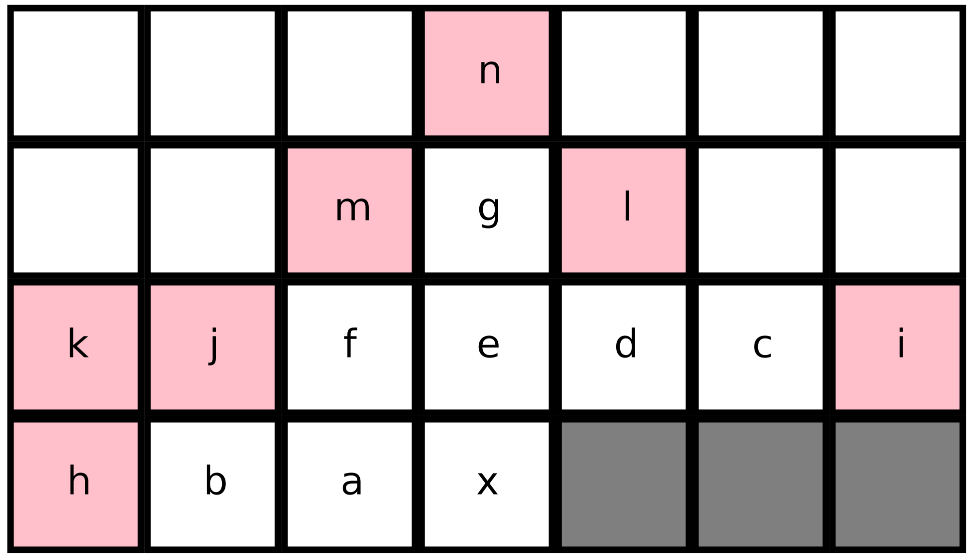
$$P(x|a,b,c,d,e,f,g)$$



- Why not use very large contexts?
- Why not use very small contexts?
- Why not use non-contiguous conditioning pixels?

Context coding (cont.)

- Two-level context coding



- Try bigger neighborhood first; if counts sufficient, use it
- Otherwise, back off to smaller neighborhood
- Decoder can do the same

Summary

- Non-linear statistical dependence among sub-bands can be exploited
- Lossless compression can be challenging; not always clear why one would want to do it
- Arithmetic coding is efficient and allows separating modeling from coding
- Context coding is one approach to lossless compression (basis of JBIG (1))