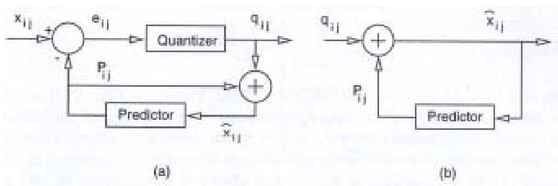
Lossy Image Compression

Lossy Image Compression

- Decompression yields an imperfect reconstruction of the original image data
- Focus on the basic concepts of lossy compression adopted in practice and in standards
- Special emphasis on Discrete Cosine Transform-based compression schemes
  - DCT-based schemes form the basis of all the image and video compression standards
- Sample-based vs. Block-based coding

Sample-Based Coding

- Compressed on a sample-by-sample basis
  - Samples can be either in the spatial domain or in the frequency domain
- Example: Differential Pulse Code Modulation (DPCM)

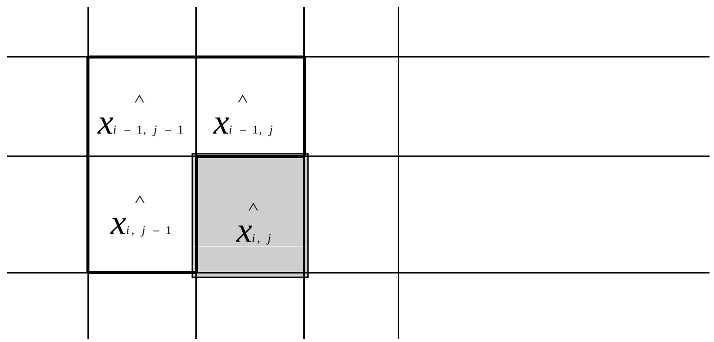


$\hat{x}_{ij} = P_{ij} + q_{ij}$

Decoder's estimate for x_{ij}

Quantizer:
irreversible
main cause of information loss

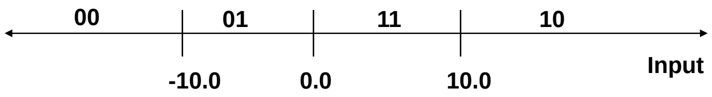
Prediction Kernel for DPCM



$$P_{ij} = w_1 X_{ij-1} + w_2 X_{i-1j-1} + w_3 X_{i-1j}$$

Quantization

- **Quantization:**
  - the process of representing a large (possibly infinite) set of values with a much smaller set



2-bit encoder

- Input: scalars or vectors

Quantization

$$q_{ij} = \text{round}(\frac{e_{ij}}{\Delta})$$

- Δ : quantization step
- variance(e_{ij}) < variance(x_{ij})
  - quantization will not introduce significant distortion
- Covariance function between pixels at distances longer than 8 decays rapidly
  - No benefit to using more than an eight-pixel neighborhood when forming P_{ij}

Computational Complexity

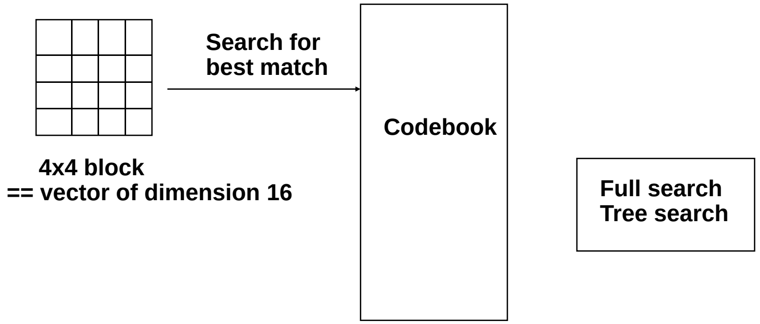
- Example: Three prior samples
  - P_{ij} requires 3 multiplications, 2 additions
  - eij requires 1 subtraction
  - q_{ij} requires 1 multiplications
  - Multiplications involve small constants
    - » Look-up table methods can be used for faster encoding.

Block-Based Coding

- Block-based coding schemes yield:
  - better compression ratios for the same-level of distortion or
  - less distortions for the same compression ratio.
- Spatial-domain vs. Transform-domain block coding

Spatial-Domain Block Coding

- Pixels are grouped into blocks.
- Blocks are compressed in the spatial domain.
- Example: Vector-Quantization method



Transform-Domain Block Coding

- Pixels are grouped into blocks.
- Blocks are transformed to another domain.
- Why transforming X to Y?
  - Hopefully Y is a more compact representation of X
- Lossy transform-based compression:
  - First, perform transformation from X to Y
  - Second, discard the less important information in Y

Compaction Efficiency for Various Transformations

- KLT is the optimum transform:
  - It packs the most energy in the least numbe rof elements in Y.
- KLT has implementation-related deficiencies:
  - Basis functions are image dependent
- In practice, DCT is used widely
  - Basis functions are image INDEPENDENT.
  - Compaction efficiency is close to that of the KLT.

DCT-Based Coding

- Basic tool: N x N image block from the spatial domain to the DCT domain
- What value for N?
  - N = 8 for compression standards
  - Why 8?
    - » *Implementation*: 8x8 is appropriate considering the memory requirements and computational complexity
    - » *Compaction Efficiency*: a blocksize larger than 8x8 does not offer significant improvements

Discrete Cosine Transform

$$F[u,v] = \frac{4C(u)C(v)}{n^2} \sum_{j=0}^{n-1} \sum_{k=0}^{n-1} f(j,k) \cos\left[\frac{(2j+1)u\pi}{2n}\right] \cos\left[\frac{(2k+1)v\pi}{2n}\right]$$

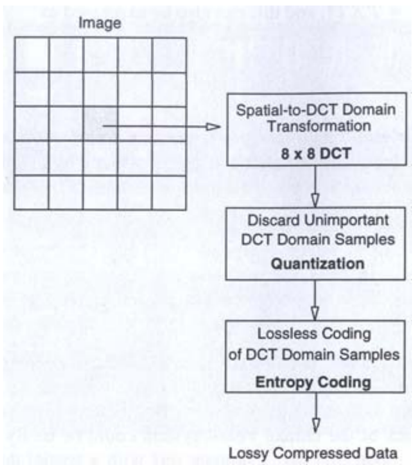
where

$$C(w) = \begin{cases} \frac{1}{\sqrt{2}} & \text{for } w = 0 \\ 1 & \text{for } w = 1, 2, \ldots, n-1 \end{cases}$$

Benefits of DCT

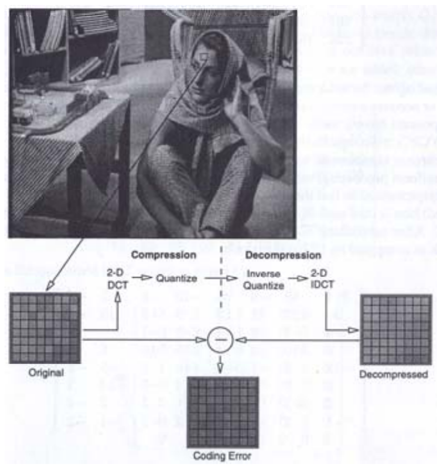
- For highly correlated images, DCT compaction efficiency ~= KLT compaction efficiency
- 2-D DCT and IDCT are separable transformations
  - 2-D DCT can be obtained by row-wise 1-D DCTs followed by column-wise 1-D DCTs
- DCT basis is image independent.
- There exist fast algorithms that require fewer operations than those required by the definition.

Generic DCT-based Image Coding System

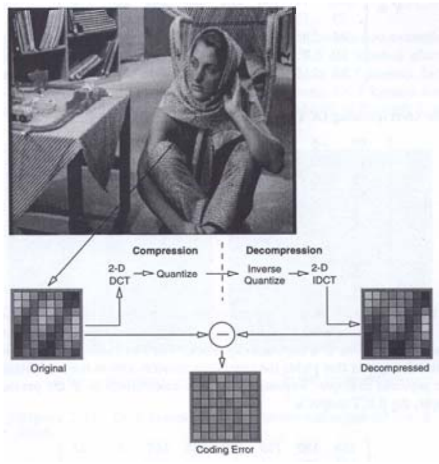


- Entropy coder combines a run-length coder with a Huffman coder

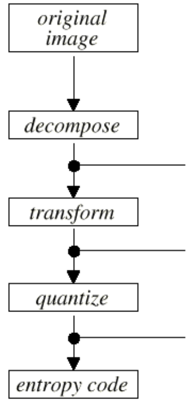
a block from a low-activity region



a block from a high-activity region



Block-Based Coding

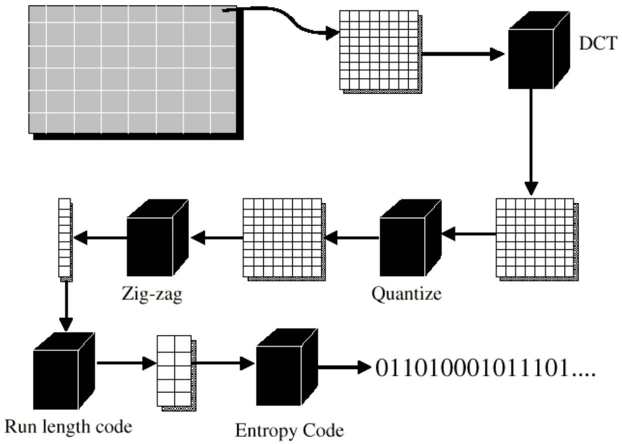


sequence of 8x8 blocks - different planes treated separately (i.e., RGB, YUV, etc.)

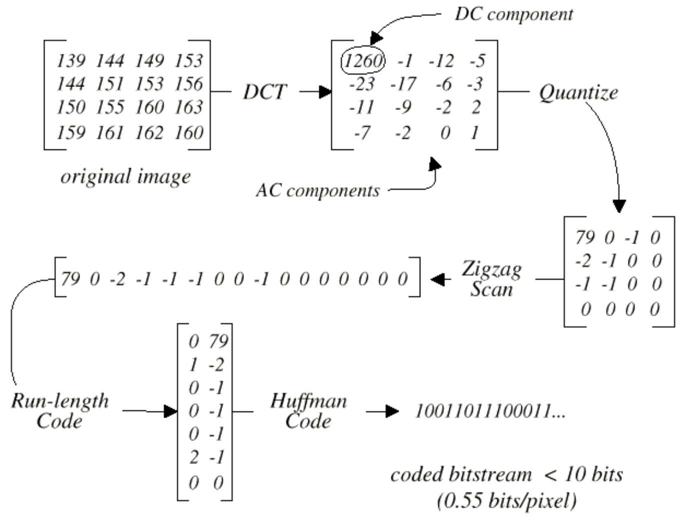
transformed blocks reduce redundancy and concentrate signal energy into a few coefficients: *Discrete Cosine Transform* (DCT)

blocks with discarded information - goal is to smooth picture and throw away information that will not be missed such as high frequencies

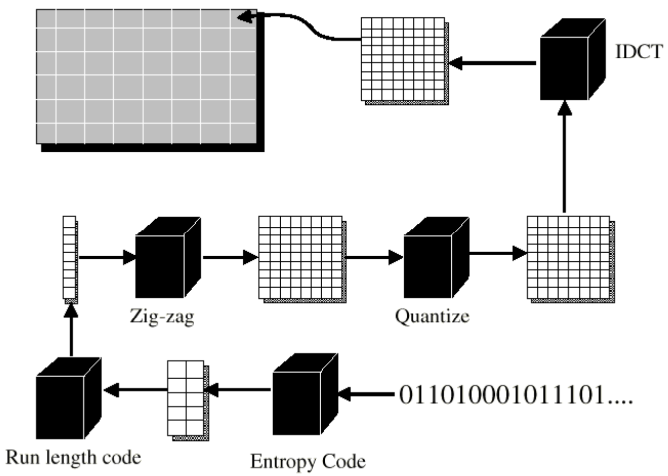
Block-Based Coding: Encoding



Block-Based Coding: Encoding



Block-Based Coding: Decoding



Block-Based Coding: Decoding Errors

$$\begin{bmatrix} 139 & 144 & 149 & 153 \\ 144 & 151 & 153 & 156 \\ 150 & 155 & 160 & 163 \\ 159 & 161 & 162 & 160 \end{bmatrix}$$

original block

$$\begin{bmatrix} 144 & 146 & 149 & 152 \\ 148 & 150 & 152 & 154 \\ 155 & 156 & 157 & 158 \\ 160 & 161 & 161 & 162 \end{bmatrix}$$

reconstructed block

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

errors

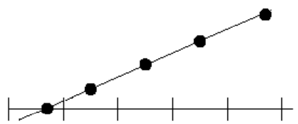
Discrete Cosine Transform

$$F[u,v] = \frac{4C(u)C(v)}{n^2} \sum_{j=0}^{n-1} \sum_{k=0}^{n-1} f(j,k) \cos\left[\frac{(2j+1)u\pi}{2n}\right] \cos\left[\frac{(2k+1)v\pi}{2n}\right]$$

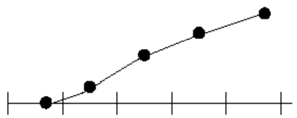
where

$$C(w) = \begin{cases} \frac{1}{\sqrt{2}} & \text{for } w = 0 \\ 1 & \text{for } w = 1, 2, ..., n-1 \end{cases}$$

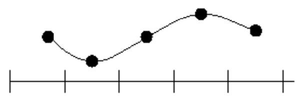
DCT vs. DFT



Original Signal

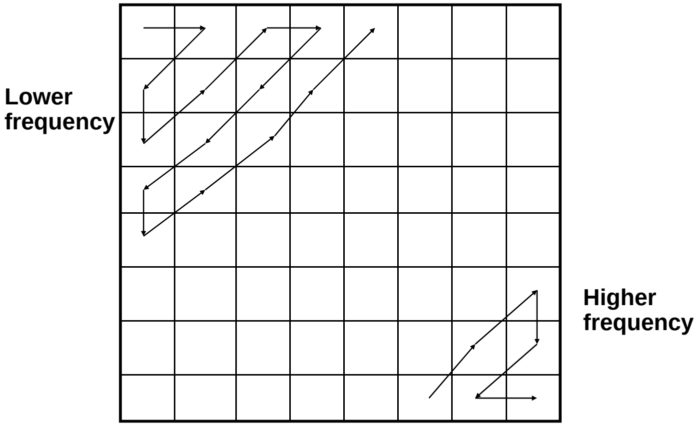


Recovered from DCT



Recovered from DFT

ZigZag Scan



Fast DCT Algorithms

- **Direct Computation**
  - Each DCT coefficient requires 64 multiplications and 64 additions
  - For an 8x8 block, 4096 (= 64x64) multiplications and 4096 additions
- **Separable Implementation**
  - Eight 1-D row-wise DCTs followed by eight 1-D column-wise DCTs
    - » For each 1-D DCT coefficient, 8 Xs and 8 +s
    - » For a 1x8 block, 64 Xs and 64 +s
  - For 16 1-D DCTs, 1024 Xs and 1024 +s

- **These numbers are still quite high**
- **For real-time implementation, need faster algorithms**

Lee's 1-D IDCT Algorithm

$$x(k) = \sum_{n=0}^{N-1} \overline{X(n)}\, C_{2N}^{(2k+1)n}, \quad k = 0, 1, \ldots, N-1$$

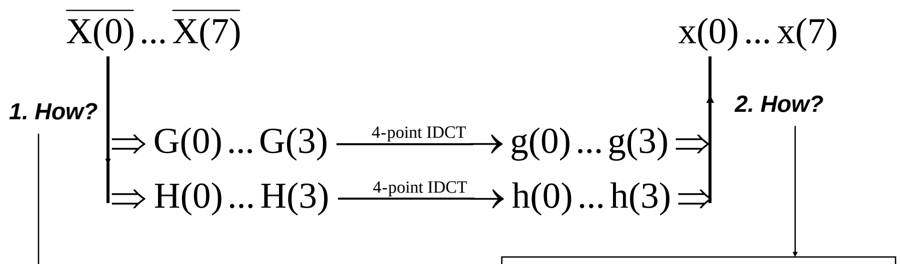
$$\text{where } \overline{X(n)} = e(n)\, X(n),$$

$$e(n) = \begin{cases} 1/\sqrt{2}, \text{ if } n = 0 \\ 1, \text{ otherwise} \end{cases}$$

$$C_{2N}^{(2k+1)n} = \cos\left(\frac{\pi(2k+1)n}{2N}\right)$$

Lee's 1-D IDCT Algorithm

Main idea: Recursively Divide & Conquer



$$G(n) = \overline{X(2n)}$$

$$H(n) = \overline{X(2n+1)} + \overline{X(2n-1)}$$

$$x(k) = g(k) + \frac{1}{2C_{16}^{2k+1}} h(k)$$

$$x(7-k) = g(k) - \frac{1}{2C_{16}^{2k+1}} h(k)$$

Lee's 1-D IDCT Algorithm

- **From Figure 1 of Lee's paper**
  - For N=8,
    - » # of multiplications = 12
    - » # of additions = 29

Non-trivial constant

- **Input sequence: in bit-reversed order**
- **Output sequence:**
  - Start with (0, 1)
  - Add the prefix 0 to each element (*0*0, *0*1)
  - Obtain the rest of elements by complementing the existing ones (00, 01) -> (00, 01, *11, 10)*

Computational Complexity of DCT Algorithms

- **Disadvantages of 2-D DCT methods**
  - Storage for up to 128 elements is required
    - » For systems with the small number of registers, not feasible
  - Data addressing is highly irregular
    - » additional overhead for address calculations