# Transform Coding

National Chiao Tung University
Chun-Jen Tsai
11/24/2014

# Transform Domain Data Analysis

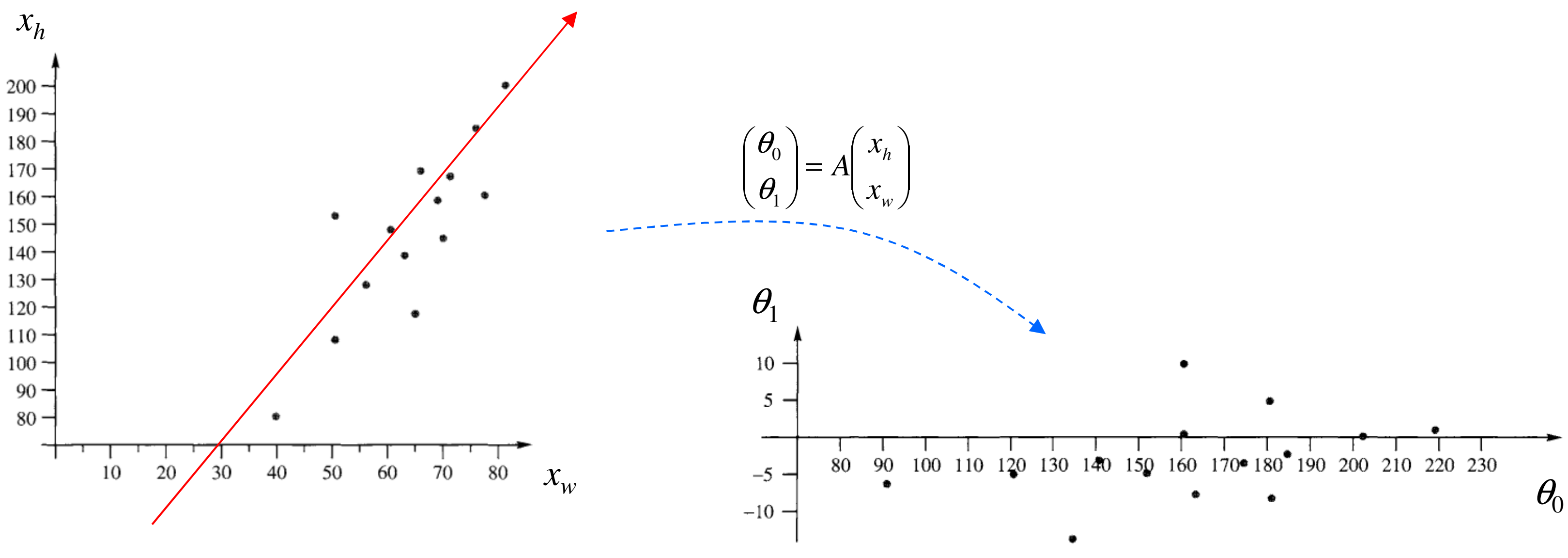
- Given an invertible transform $A$, the entropy of a source $\mathbf{x}$ does not change subject to $A$, i.e. $A\mathbf{x}$ has the same entropy as $\mathbf{x}$.
- However, there are several reasons why we want to perform lossy compression on $A\mathbf{x}$, instead of $\mathbf{x}$:
  - Input data sequence can be interpreted with more insights
  - Input data possibly are de-correlated in transform domain
  - The original time-ordered sequence of data can be decomposed into different categories

# Example: Height-Weight Data (1/3)

- The height-weight data pair tends to cluster alone the line $x_h = 2.5x_w$. A rotation transform

$$A = \begin{pmatrix} \cos\phi & \sin\phi \\ -\sin\phi & \cos\phi \end{pmatrix}, \quad \phi = 68.02^\circ,$$

can simplify the data representation :



# Example: Height-Weight Data (2/3)

- If we set $\theta_1$ to zeros for all the data pairs, and transform the data back to $x_h$–$x_w$ domain, we have the reconstruction errors as follows:

Original data

| Height | Weight |
|---|---|
| 65 | 170 |
| 75 | 188 |
| 60 | 150 |
| 70 | 170 |
| 56 | 130 |
| 80 | 203 |
| 68 | 160 |
| 50 | 110 |
| 40 | 80 |
| 50 | 153 |
| 69 | 148 |
| 62 | 140 |
| 76 | 164 |
| 64 | 120 |

Reconstructed data

| Height | Weight |
|---|---|
| 68 | 169 |
| 75 | 188 |
| 60 | 150 |
| 68 | 171 |
| 53 | 131 |
| 81 | 203 |
| 65 | 162 |
| 45 | 112 |
| 34 | 84 |
| 60 | 150 |
| 61 | 151 |
| 57 | 142 |
| 67 | 168 |
| 50 | 125 |

# Example: Height-Weight Data (3/3)

- Note that, in original data, both $x_h$ and $x_w$ have non-negligible variances, however, for $\theta_0$ and $\theta_1$, only $\theta_0$ has large variance
- Variance (or energy) of a source and its information has a positive relation; larger source variance, higher entropy
  - For Gaussian source, the differential entropy is $(\log_2 \pi e \sigma^2)/2$.
- The error introduced into the reconstructed sequence of $\{x\}$ is equal to the error introduced into the transform-domain sequence $\{\theta\}$.

# Transform Coding Principle

- Transform step:
  - The source $\{x_n\}$ is divided into blocks of size $N$. Each block is mapped into a transform sequence $\{e_n\}$ using a reversible mapping
  - Most of the energy of the transformed block was contained in few elements of the transformed values
- Quantization step:
  - The transformed sequence is quantized based on the following strategy:
    - The desired average bit rate
    - The statistics of the various transformed elements
    - The effect of distortion on the reconstructed sequence
- Entropy coding step:
  - The quantized data are entropy-coded using Huffman, AC, or other techniques

# Transform Formulation

- For media coding, only linear transforms are used
  - The forward transform can be denoted by

$$\theta_n = \sum_{i=0}^{N-1} x_i a_{n,i}.$$

  - The inverse transform is

$$x_n = \sum_{i=0}^{N-1} \theta_i b_{n,i}.$$

- The selection of $N$ is application-specific
  - Complexity of transform is lower for small $N$
  - Large $N$ adapts to fast-changing statistics badly
  - Large $N$ produces better resolution in transform domain

# 2-D Forward Transform

- For 2-D signals $X_{i,j}$, a general linear 2-D transform of block size $N \times N$ is given as

$$\Theta_{k,l} = \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} x_{i,j} a_{i,j,k,l}.$$

- If separable transform is used; the formulation can be simplified to

$$\Theta_{k,l} = \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} a_{k,i} x_{i,j} a_{j,l} = \sum_{i=0}^{N-1} a_{k,i} \left( \sum_{j=0}^{N-1} x_{i,j} a_{j,l} \right).$$

- In matrix form, the separable transform becomes

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

# Orthonormal Transform

- All the transforms used in multimedia compression are orthonormal transforms. Thus, $A^{-1} = A^T$.
  In this case, $\boldsymbol{\Theta} = A\boldsymbol{X}A^T$ becomes $\boldsymbol{\Theta} = A\boldsymbol{X}A^{-1}$.

- Orthonormal transforms are energy preserving

$$\sum_{i=0}^{N-1} \theta_i^2 = \theta^T \theta = (\mathbf{Ax})^T \mathbf{Ax}$$

$$= \mathbf{x}^T \mathbf{A}^T \mathbf{Ax} = \mathbf{x}^T \mathbf{x} = \sum_{n=0}^{N-1} x_n^2 .$$

# Energy Compaction Effect

- The efficiency of a transform depends on how much energy compaction is provided by the transform
- The amount of energy compaction can be measured by the ratio of the arithmetic mean of the variances to their geometric means:

$$G_{TC} = \frac{\frac{1}{N}\sum_{i=0}^{N-1}\sigma_i^2}{\left(\prod_{i=0}^{N-1}\sigma_i^2\right)^{\frac{1}{N}}},$$

where $\sigma_i^2$ is the variance of the $i$th coefficients.

Note: The wider the spread of $\sigma_i^2$ w.r.t. their arithmetic mean, the smaller the value of the geometric mean will be → better energy compaction!

# Decomposition of 1-D Input

- Transform decomposes an input sequence into components with different characteristics. If

$$A = \frac{1}{\sqrt{2}}\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix},$$

input $\mathbf{x} = [x_1, x_2]$, the transformed output is

$$A\mathbf{x} = \left[\frac{(x_1 + x_2)}{\sqrt{2}}, \frac{(x_1 - x_2)}{\sqrt{2}}\right].$$

The first transformed component computes the average (i.e. low-pass) behavior of the input sequence, while the 2$^{nd}$ component captures the differential (i.e. high-pass) behavior of the input.

# Decomposition of 2-D Input

- If $A$ in previous example is used for 2-D transform and $X$ is a 2-D input, we have $X = A^T \Theta A$:

$$\begin{bmatrix} x_{00} & x_{01} \\ x_{10} & x_{11} \end{bmatrix} = \frac{1}{2}\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}\begin{bmatrix} \theta_{00} & \theta_{01} \\ \theta_{10} & \theta_{11} \end{bmatrix}\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$$

$$= \frac{1}{2}\begin{bmatrix} \theta_{00}+\theta_{01}+\theta_{10}+\theta_{11} & \theta_{00}-\theta_{01}+\theta_{10}-\theta_{11} \\ \theta_{00}+\theta_{01}-\theta_{10}-\theta_{11} & \theta_{00}-\theta_{01}-\theta_{10}+\theta_{11} \end{bmatrix}$$

$$= \theta_{00}\alpha_{0,0} + \theta_{01}\alpha_{0,1} + \theta_{10}\alpha_{1,0} + \theta_{11}\alpha_{1,1},$$

  where $\alpha_{i,j}$ is the outer product of $i$th and $j$th rows of $A$.
- How do you interpret $\theta_{0,0}$, …, $\theta_{1,1}$?
  - $\theta_{0,0}$ is the DC coefficient, and other $\theta_{i,j}$ are AC coefficients.

# Karhunen-Loeve Transform (KLT)

- KLT consists of the eigenvectors of the autocorrelation matrix: $[R]_{i,j} = E[X_n X_{n+|i-j|}]$.
- KLT minimizes the geometric means of the variance of the transform coefficients $\rightarrow$ provides maximal $G_{TC}$
- Issues with KLT
  - For non-stationary inputs, the autocorrelation function is time varying; computation of KLT is relatively expensive
  - KLT matrix must be transmitted to the decoder
  - If the input statistics change slowly, and the transform size can be kept small, the KLT can be useful

# Example: KLT

- For $N = 2$, the autocorrelation matrix for a stationary process is

$$R = \begin{bmatrix} R_{xx}(0) & R_{xx}(1) \\ R_{xx}(1) & R_{xx}(0) \end{bmatrix},$$

The eigenvectors of $R$ are

$$v_1 = \begin{bmatrix} \alpha \\ \alpha \end{bmatrix}, \ v_2 = \begin{bmatrix} \beta \\ -\beta \end{bmatrix}.$$

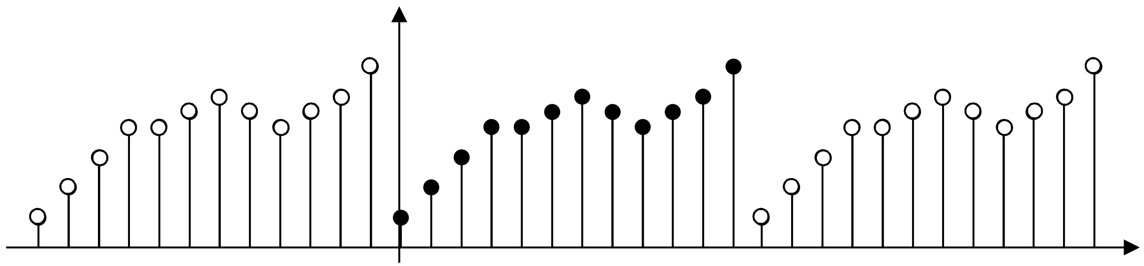
With orthonormal constraint, the transform matrix is

$$\mathbf{K} = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}.$$

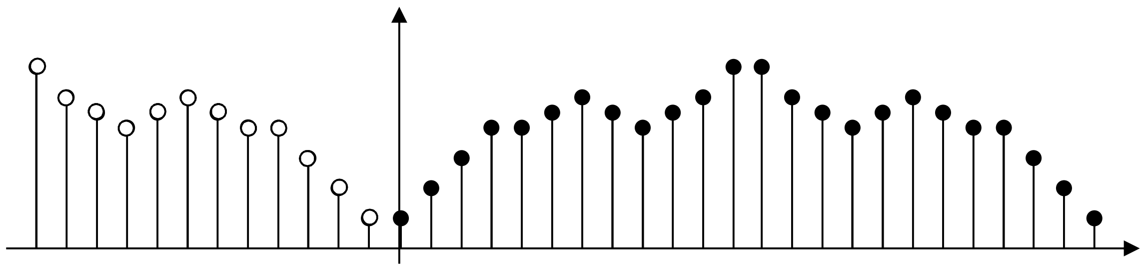
# Discrete Cosine Transform

- DCT is derived from the Discrete Fourier Transform (DFT) by first perform an even-function extension to the input data, then compute its DFT:
  - Only real number operations are required
  - Better energy compaction than DFT

DFT



DCT



# DCT Formulation

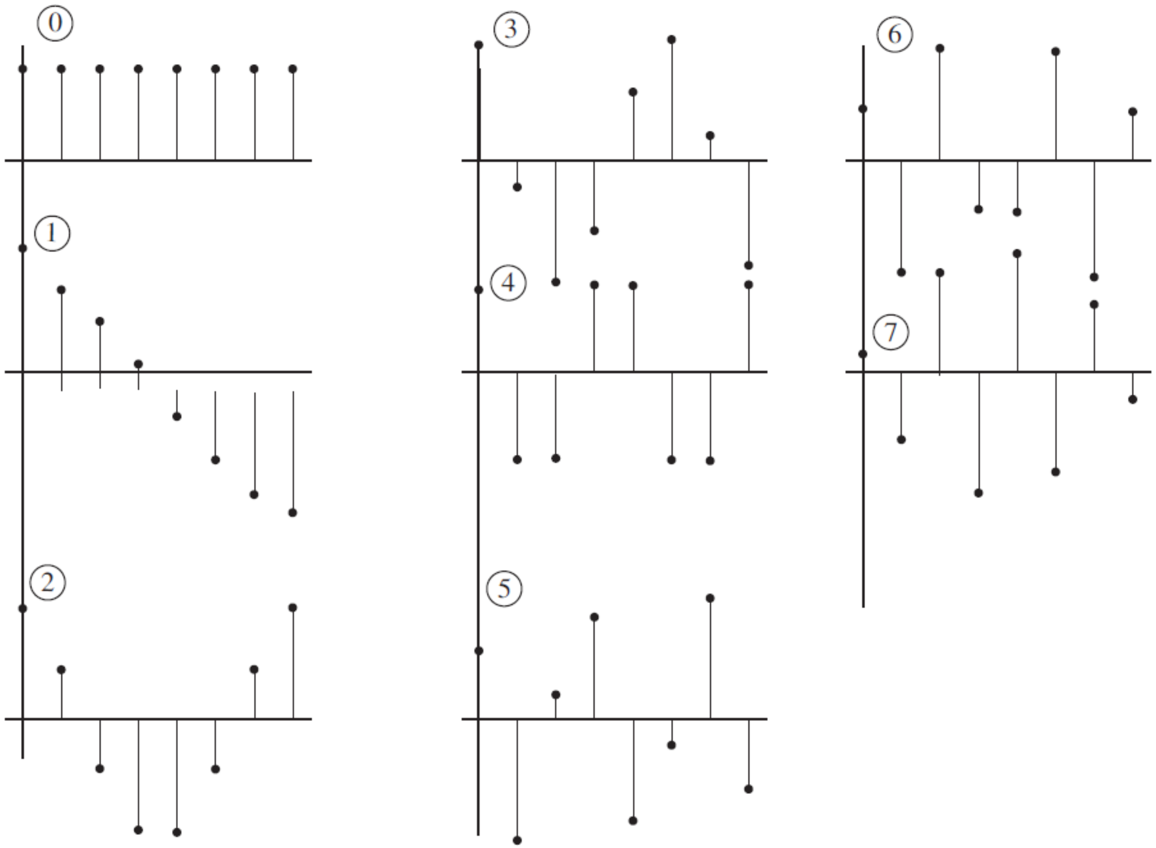
- The rows of DCT matrix is composed of cosine functions of different frequencies:

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

- The inner product of the input signal with each row of the matrix is the projection of the input signal onto a cosine function of fixed frequency
  - The larger $N$ is, the better the frequency resolution is

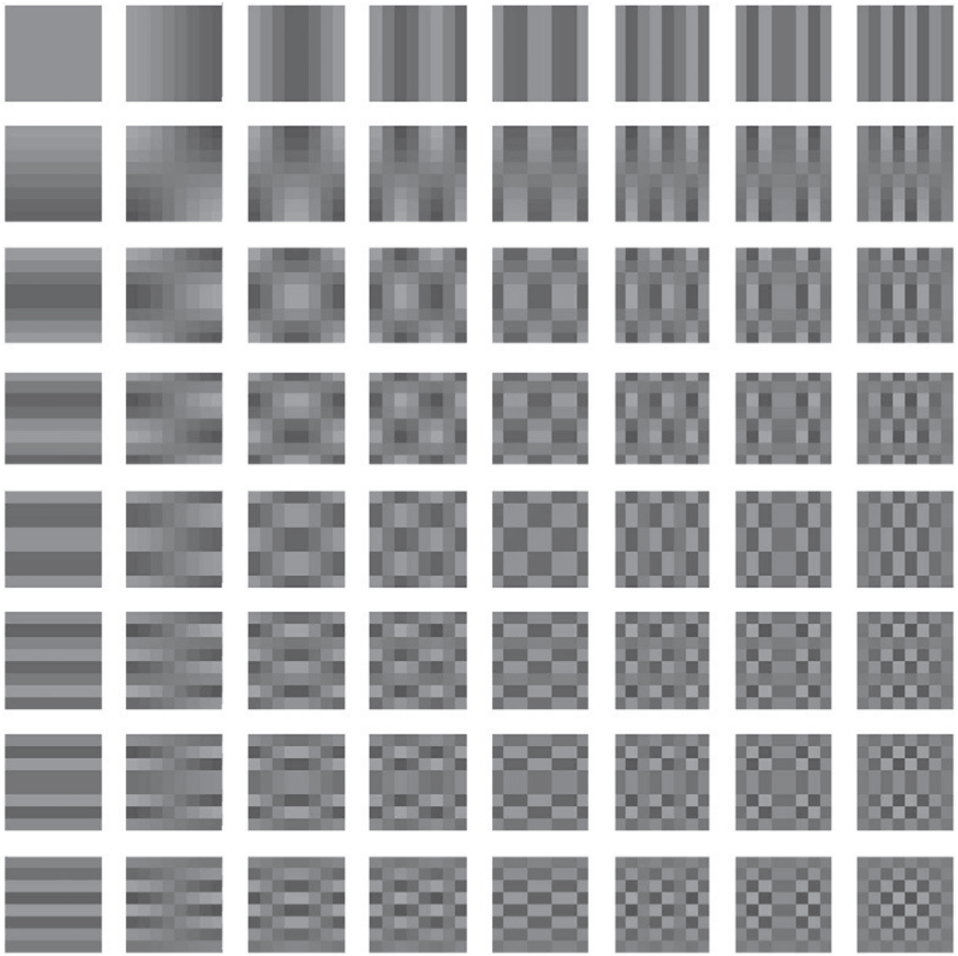
# Basis Functions of 8-Point DCT

- ❑ Each column of the DCT matrix is a basis function:



# Basis Images of 8-Point 2-D DCT

- DCT can be extended to a 2-D transform:



# Performance of DCT

- For Markov sources with high correlation coefficient $\rho$,

$$\rho = \frac{E[x_n x_{n+1}]}{E[x_n^2]},$$

  the compaction ability of DCT is close to that of KLT

- As many sources can be modeled as Markov sources with high values for $\rho$, DCT is the most popular transform for multimedia compression

# Discrete Walsh-Hadamard Trans.

- The Hadamard transform is defined by an $N \times N$ matrix $H$ with the property $HH^T = NI$.
  - Simple to compute while still separate low frequency from high frequency components of the input data
- The Hadamard matrix is recursively defined as:

$$H_{2N} = \begin{bmatrix} H_N & H_N \\ H_N & -H_N \end{bmatrix}, \text{ and } H_1 = [1].$$

- The DWHT transform matrix is obtained by
  - Normalize the matrix by $1/N^{½}$ so that it is orthonormal
  - Re-arrange the rows according to number of sign changes

# Coding of Transform Coefficients

- Different transform coefficients should be quantized and coded differently based on the amount of information it carries
  - Information is related to the variance of each coefficients
- The bit allocation problem tries to determine the level of quantizer to use for different transform coefficients
- The Lagrange multiplier optimization technique is often used to solve the optimal bit allocation

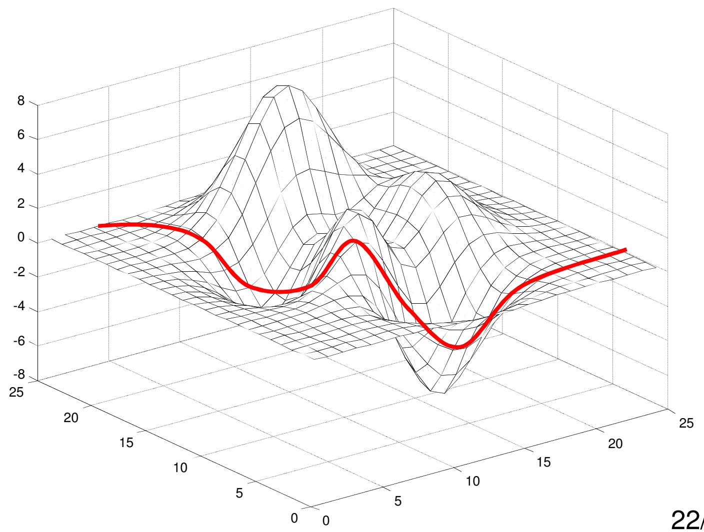
# Lagrange Multiplier

- A constrained optimization problem tries to minimize a cost function $f(x, y)$ subject to some constraints on the parameter $x$ and $y$: $g(x, y) = c$
- The Lagrange cost function is defined as follows:

$$J(x, y, \lambda) = f(x, y) - \lambda \cdot \left\| g(x, y) - c \right\|^2.$$

- Solution: solve

$$\nabla_{x,y,\lambda} J(x, y, \lambda) = 0.$$



# Rate-Distortion Optimization (1/3)

- If the rate per coefficient is $R$ and the rate per $k$th coefficient is $R_k$ , then

$$R = \frac{1}{M}\sum_{k=1}^{M} R_k ,$$

where $M$ is the number of transform coefficients

- The error variance for the $k$*th* quantizer $\sigma_{r_k}{}^2$, is related to the $k$*th* input variance $\sigma_{\theta_k}{}^2$, by:

$$\sigma_{r_k}^2 = \alpha_k 2^{-2R_k} \sigma_{\theta_k}^2 ,$$

where $\alpha_k$ depends on input distribution and quantizer

- The total reconstruction error is given by

$$\sigma_r^2 = \sum_{k=1}^{M} \alpha_k 2^{-2R_k} \sigma_{\theta_k}^2 .$$

# Rate-Distortion Optimization (2/3)

- The objective of the bit allocation procedure is to find $R_k$ to minimize $\sigma_r^2$ subject to total rate constraint $R$.
- If we assume that $\alpha_k$ is a constant $\alpha$ for all $k$, we can set up the minimization problem in terms of Lagrange multipliers as

$$J = \alpha \sum_{k=1}^{M} 2^{-2R_k} \sigma_{\theta_k}^2 - \lambda \left( R - \frac{1}{M} \sum_{k=1}^{M} R_k \right).$$

- Taking the derivative of $J$ with respect to $R_k$ and setting it to zero, we obtain the expression for $R_k$:

$$R_k = \frac{1}{2} \log_2 (2\alpha \ln 2 \sigma_{\theta_k}^2) - \frac{1}{2} \log_2 \lambda.$$

# Rate-Distortion Optimization (3/3)

- Substituting $R_k$ to the expression for $R$, we have:

$$\lambda = \prod_{k=1}^{M} \left(2\alpha \ln 2\sigma_{\theta_k}^2\right)^{\frac{1}{M}} 2^{-2R}.$$

- Therefore, the individual bit allocations for each transform coefficients is:

$$R_k = R + \frac{1}{2}\log_2 \frac{\sigma_{\theta_k}^2}{\prod_{k=1}^{M}\left(\sigma_{\theta_k}^2\right)^{\frac{1}{M}}}.$$

- Note that $R_k$ may not be integers or positive numbers
  - Negative $R_k$'s are set to zero
  - Positive $R_k$'s are reduced to a smaller integer value

# Zonal Sampling

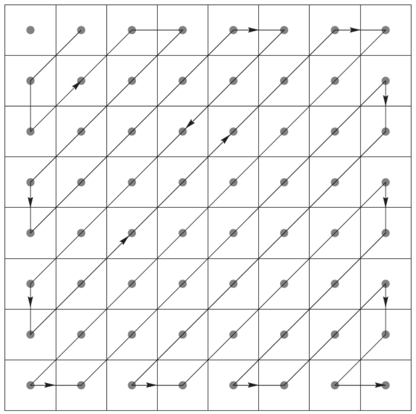
- Zonal sampling is a simple bit allocation algorithm:
  1. Compute $\sigma_{\theta_k}^2$ for each coefficient.
  2. Set $R_k = 0$ for all $k$ and set $R_b = MR$, where $R_b$ is the total number of bits available for distribution.
  3. Sort the variances $\{\sigma_{\theta_k}^2\}$ Suppose $\sigma_{\theta_m}^2$ is the maximum.
  4. Increment $R_m$ by $1$, and divide $\sigma_{\theta_m}^2$ by $2$.
  5. Decrement $R_b$ by $1$. If $R_b = 0$, then stop; otherwise, go to 3.

Bit allocation map for an 8×8 transform

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 8 | 7 | 5 | 3 | 1 | 1 | 0 | 0 |
| 7 | 5 | 3 | 2 | 1 | 0 | 0 | 0 |
| 4 | 3 | 2 | 1 | 1 | 0 | 0 | 0 |
| 3 | 3 | 2 | 1 | 1 | 0 | 0 | 0 |
| 2 | 1 | 1 | 1 | 0 | 0 | 0 | 0 |
| 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

# Threshold Coding

- Another bit allocation policy is called threshold coding
  - Arrange the transform coefficients in a line
  - The first coefficient is always coded
  - For remaining coefficients
    - If the magnitude is smaller than a threshold, it is skipped
    - If the magnitude is larger than a threshold, its quantized value and the number of skipped coefficients before it is coded
- Zigzag scan is often used for 2-D to 1-D mapping



# JPEG Image Compression

- A standard defined by ISO/IEC JTC1/SC 29/WG 1 in 1992
  - The official IS number is IS 10918-1, which defines the input to the decoder (a.k.a. the elementary stream), and how the decoder reconstructs the image
  - The popular file format JFIF for JPEG elementary stream is defined in 10918-5
- There are several new image coding standards that are incompatible to the old JPEG, but still bearing the JPEG name
  - Wavelet-based JPEG-2000 (IS 15444-1)
  - High quality lossless/lossy JPEG-XR (IS 29199-2)

# JPEG Initial Processing

- Color space RGB $\rightarrow$ $YC_BC_R$ mapping
- Chroma channel 4:2:2 sub-sampling
- Level shifting: assume each pixel has $p$-bit, then each pixel $x_{i,j} = x_{i,j} - 2^{p-1}$
- Split pixels into 8×8 blocks
  - If image size is not a multiple of 8, extra rows/columns are padded to achieve multiple of 8
  - Padded data is discarded after decoding

# JPEG 8×8 DCT Transform

- ❑ Forward DCT is applied to each 8×8 block

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 124 | 125 | 122 | 120 | 122 | 119 | 117 | 118 |
| 121 | 121 | 120 | 119 | 119 | 120 | 120 | 118 |
| 126 | 124 | 123 | 122 | 121 | 121 | 120 | 120 |
| 124 | 124 | 125 | 125 | 126 | 125 | 124 | 124 |
| 127 | 127 | 128 | 129 | 130 | 128 | 127 | 125 |
| 143 | 142 | 143 | 142 | 140 | 139 | 139 | 139 |
| 150 | 148 | 152 | 152 | 152 | 152 | 150 | 151 |
| 156 | 159 | 158 | 155 | 158 | 158 | 157 | 156 |

Level-shifting

Forward DCT

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 39.88 | 6.56 | −2.24 | 1.22 | −0.37 | −1.08 | 0.79 | 1.13 |
| −102.43 | 4.56 | 2.26 | 1.12 | 0.35 | −0.63 | −1.05 | −0.48 |
| 37.77 | 1.31 | 1.77 | 0.25 | −1.50 | −2.21 | −0.10 | 0.23 |
| −5.67 | 2.24 | −1.32 | −0.81 | 1.41 | 0.22 | −0.13 | 0.17 |
| −3.37 | −0.74 | −1.75 | 0.77 | −0.62 | −2.65 | −1.30 | 0.76 |
| 5.98 | −0.13 | −0.45 | −0.77 | 1.99 | −0.26 | 1.46 | 0.00 |
| 3.97 | 5.52 | 2.39 | −0.55 | −0.051 | −0.84 | −0.52 | −0.13 |
| −3.43 | 0.51 | −1.07 | 0.87 | 0.96 | 0.09 | 0.33 | 0.01 |

# JPEG Quantization

- Midtread quantization is used; the step size for each coefficients is from an 8×8 quantization matrix $Q$, e.g.,

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 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 |

⇨ $Q_{ij}$ is the step size for $i,j$-th transform coefficients

- Quantized values are called "labels." For input coefficient $\theta_{ij}$, we have

$$l_{ij} = \left\lfloor \frac{\theta_{ij}}{Q_{ij}} + 0.5 \right\rfloor.$$

# JPEG Quantization Example

- Quantization controls the entropy of the image
  - Quantization matrices reflect image quality
  - A scalar number (quality factor) is often used as quantization matrix multiplier to control image quality

$\theta_{00}$

| | | | |
|---|---|---|---|
| 39.88 | 6.56 | –2.24 | 1.22 |
| –102.43 | 4.56 | 2.26 | 1.12 |
| 37.77 | 1.31 | 1.77 | 0.25 |
| –5.67 | 2.24 | –1.32 | -0.81 |

$Q_{00}$

| | | | |
|---|---|---|---|
| 16 | 11 | 10 | 16 |
| 12 | 12 | 14 | 19 |
| 14 | 13 | 16 | 24 |
| 14 | 17 | 22 | 29 |

$$l_{00} = \left\lfloor \theta_{00} / Q_{00} + 0.5 \right\rfloor = \left\lfloor 39.88/16 + 0.5 \right\rfloor = \lfloor 2.99 \rfloor = 2$$

$\Rightarrow$ $l_{00}$

| | | | |
|---|---|---|---|
| 2 | 1 | 0 | 0 |
| –9 | 0 | 0 | 0 |
| 3 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 |

# Entropy Coding

- DC/AC coefficients are coded differently
  - DCs are coded using
    - Differential coding + Huffman coding
    - Each DC difference is coded using a Huffman prefix plus a fixed length suffix
  - ACs are coded using
    - Run-Length coding + Huffman coding

# DC Difference Code Table

| Difference category (VLC-code as prefix) | value in each category (FLC-code as suffix) | | | | | |
|---|---|---|---|---|---|---|
| 0 | | | 0 | | | |
| 1 | | | −1 | 1 | | |
| 2 | | −3 | −2 | 2 | 3 | |
| 3 | −7 | ... | −4 | 4 | ... | 7 |
| 4 | −15 | ... | −8 | 8 | ... | 15 |
| 5 | −31 | ... | −16 | 16 | ... | 31 |
| 6 | −63 | ... | −32 | 32 | ... | 63 |
| 7 | −127 | ... | −64 | 64 | ... | 127 |
| 8 | −255 | ... | −128 | 128 | ... | 255 |
| 9 | −511 | ... | −256 | 256 | ... | 511 |
| 10 | −1,023 | ... | −512 | 512 | ... | 1,023 |
| 11 | −2,047 | ... | −1,024 | 1,024 | ... | 2,047 |
| 12 | −4,095 | ... | −2,048 | 2,048 | ... | 4,095 |
| 13 | −8,191 | ... | −4,096 | 4,096 | ... | 8,191 |
| 14 | −16,383 | ... | −8,192 | 8,192 | ... | 16,383 |
| 15 | −32,767 | ... | −16,384 | 16,384 | ... | 32,767 |
| 16 | | | 32,768 | | | |

# AC RLE Code Table

- AC is zigzag scanned into a 1-D sequence
- Each non-zero coefficient is coded using a $Z/C$ codeword plus a sign bit $S$
  - $Z$ – number of zero run before the label
  - $C$ – label magnitude
  - EOB is used to signal the end of each block
  - ZRL is used to signal 15 consecutive zeros

| $Z/C$ | Codeword | $Z/C$ | Codeword | $\cdots$ | $Z/C$ | Codeword |
|---|---|---|---|---|---|---|
| 0/0 (EOB) | 1010 | | | $\cdots$ | F/0 (ZRL) | 11111111001 |
| 0/1 | 00 | 1/1 | 1100 | $\cdots$ | F/1 | 1111111111110101 |
| 0/2 | 01 | 1/2 | 11011 | $\cdots$ | F/2 | 1111111111110110 |
| 0/3 | 100 | 1/3 | 1111001 | $\cdots$ | F/3 | 1111111111110111 |
| 0/4 | 1011 | 1/4 | 111110110 | $\cdots$ | F/4 | 1111111111111000 |
| 0/5 | 11010 | 1/5 | 11111110110 | $\cdots$ | F/5 | 1111111111111001 |
| $\vdots$ | $\vdots$ | $\vdots$ | $\vdots$ | | $\vdots$ | |

# JPEG Coding Example

- A good example from Wikipedia:


83,261 bytes
compression ratio 2.6:1


15,138 bytes
compression ratio 15:1


4,787 bytes
compression ratio 46:1