Ch. 13 Transform Coding

Coding Gain & Classic Transforms

Coding Gain for Transform Coding

This is one way (and an effective one!) to compare Transform Coding to Direct Quantization in the Time Domain…

It is also a good way to compare the performance between various Transforms…

Define "***Transform Coding Gain***" as $G_{TC} \triangleq \dfrac{D_{DSQ}}{D_{TC}}$

(Large G_{TC} is what we want!!!)

Dist. for *Direct SQ*

Dist. for *TC*

Let's look at this assuming:

- Gaussian WSS signal w/ σ_x^2
- High-Rate Approximate Distortion Function (for both DSQ & TC)

For DSQ using $\bar{R}$ bits for each of the N samples we have total distortion:

$$D_{DSQ} = NC_G 2^{-2\bar{R}} \sigma_x^2$$

For Gaussian

For TC we saw earlier that the total distortion is

$$D_{TC} = NC_G \gamma 2^{-2\bar{R}} \qquad \gamma = \left[\prod_{i=0}^{N-1} \sigma_i^2 \right]^{1/N}$$

Linear Trans of Gaussian is Gaussian… standard result!

Forming the ratio and canceling common terms gives:

$$G_{TC} = \frac{\sigma_x^2}{\left[\prod_{i=0}^{N-1} \sigma_i^2\right]^{1/N}}$$

An alternate (equivalent) form of G_{TC} (for an ON transform) uses: $\sigma_x^2 = \frac{1}{N}\sum_{i=0}^{N-1} \sigma_i^2$

<u>Proof</u>: First we have $E\left\{\mathbf{x}^T\mathbf{x}\right\} = E\left\{\sum_{i=0}^{N-1} x_i^2\right\} = \sum_{i=0}^{N-1} \sigma_x^2 = N\sigma_x^2$

Then, by ON properties

$$N\sigma_x^2 = E\left\{\mathbf{x}^T\mathbf{x}\right\} = E\left\{\mathbf{y}^T \underbrace{\mathbf{A}\mathbf{A}^T}_{=\mathbf{I}} \mathbf{y}\right\} = E\left\{\mathbf{y}^T\mathbf{y}\right\} = \sum_{i=0}^{N-1} \sigma_i^2$$

$$\boxed{G_{TC} = \frac{\frac{1}{N}\sum_{i=0}^{N-1} \sigma_i^2}{\left[\prod_{i=0}^{N-1} \sigma_i^2\right]^{1/N}}} = \frac{\text{"Arithmetic Avg"}}{\text{"Geometric Avg"}}$$

For TC to outperform DSQ…
We need:
(Geom. Avg) < (Arith Avg)

So… for a given signal scenario…

…we want to choose our transform to make G_{TC} as large as possible

… that is equivalent to saying that we want a transform that gives σ_i^2 that have a larger arithmetic avg than geometric avg

So… for example, for images we might try to come up with a reasonable fairly general model…

… then see if we can identify a transform that gives for that model σ_i^2 that have a larger arithmetic avg than geometric avg

Classical Transforms

Q: What transform maximizes G_{TC}?

A: The Karhunen-Loeve (K-L) Transform

Let $\mathbf{x}$ be the signal vector drawn from a zero mean WSS process

Then
$$\mathbf{R}_{\mathbf{x}} = E\left\{\mathbf{x}\mathbf{x}^T\right\} = \begin{bmatrix} E\{x_0x_0\} & E\{x_0x_1\} & E\{x_0x_2\} & \cdots & E\{x_0x_{N-1}\} \\ E\{x_1x_0\} & E\{x_1x_1\} & & \cdots & E\{x_1x_{N-1}\} \\ E\{x_2x_0\} & & E\{x_2x_2\} & & \\ \vdots & & & \ddots & \vdots \\ E\{x_{N-1}x_0\} & E\{x_{N-1}x_1\} & & \cdots & E\{x_{N-1}x_{N-1}\} \end{bmatrix}$$

Autocorrelation Matrix

σ_x^2 on diagonal (assuming WSS)

Let $\mathbf{v}_i$ be the i^{th} eigenvector of $\mathbf{R}_{\mathbf{x}}$ with eigenvalue λ_i $\boxed{\mathbf{R}_{\mathbf{x}}\mathbf{v}_i = \lambda_i \mathbf{v}_i}$

$$\Rightarrow \quad \mathbf{R}_{\mathbf{x}} \underbrace{\begin{bmatrix} \mathbf{v}_0 & \mathbf{v}_1 & \cdots & \mathbf{v}_{N-1} \end{bmatrix}}_{\mathbf{V}} = \begin{bmatrix} \lambda_0 \mathbf{v}_0 & \lambda_1 \mathbf{v}_1 & \cdots & \lambda_{N-1} \mathbf{v}_{N-1} \end{bmatrix}$$

$$\Rightarrow \quad \boxed{\mathbf{R}_{\mathbf{x}}\mathbf{V} = \mathbf{\Lambda}\mathbf{V}} \qquad \mathbf{\Lambda} = diag\left\{\lambda_0, \lambda_1, \cdots, \lambda_{N-1}\right\} \quad (\bigstar)$$

"**Fact**": Since $\mathbf{R}_\mathbf{x}$ is symmetric, its eigenvectors form a complete set of ON vectors

We use these ON eigenvectors obtained from the AC Matrix to form a transform matrix **A**:

$$\mathbf{A} = \begin{bmatrix} \mathbf{v}_0 & \mathbf{v}_1 & \cdots & \mathbf{v}_{N-1} \end{bmatrix}^T$$

i^{th} row of **A** is $\mathbf{v}_i^T$

$$\mathbf{A} = \mathbf{V}^T$$

This transform is called the Karhunen-Loeve Transform…

Note that there is not one K-L transform but rather one for each WSS process

Applying this transform to the signal vector **x** gives the transform coefficients:

$$\mathbf{y} = \mathbf{A}\mathbf{x}$$

Q: What is the AC matrix of these transform coefficients?

A:
$$\mathbf{R}_\mathbf{y} = E\left\{\mathbf{y}\mathbf{y}^T\right\} = E\left\{\mathbf{A}\mathbf{x}(\mathbf{A}\mathbf{x})^T\right\} = E\left\{\mathbf{A}\mathbf{x}\mathbf{x}^T\mathbf{A}^T\right\}$$

$$= \mathbf{A}E\left\{\mathbf{x}\mathbf{x}^T\right\}\mathbf{A}^T = \mathbf{A}\mathbf{R}_\mathbf{x}\mathbf{A}^T = \mathbf{V}^T\mathbf{R}_\mathbf{x}\mathbf{V} = \mathbf{V}^T\mathbf{\Lambda}\mathbf{V} = \underbrace{\mathbf{V}^T\mathbf{V}}_{=\mathbf{I}}\mathbf{\Lambda}$$

$$\Rightarrow \quad \boxed{\mathbf{R}_\mathbf{y} = \mathbf{\Lambda}}$$ **K-L diagonalizes!!!**

Big Picture Result for K-L Transform

- The K-L Transform Matrix is made from the eigenvectors of the AC Matrix of the signal vector
- The AC Matrix of the K-L coefficients is diagonal (& the values on the diagonal are the eigenvalues of the AC Matrix)
  - "The K-L Diagonalizes the AC Matrix"
  - The coefficients after the K-L transform are uncorrelated!

See Next Slide

- The K-L is the optimal transform… it maximizes the TC gain
- But there are some drawbacks to using the K-L transform!!!
  - "The" transform is ***data*** dependent
    - Must send info to describe the transform matrix… Wasteful!!!
    - No efficient implementation
  - The AC Matrix must be estimated from the data
    - Adds complexity
    - Makes the algorithm sub-optimal ("only as good as the estimate of the AC")

So… the K-L is mostly of Theoretical & Historical Interest

Optimal TC Characteristics of K-L Transform

<u>"Fact" #1</u>: For <u>any</u> ON transform **A** with **y** = **Ax** we have det($\mathbf{R_y}$) = det($\mathbf{R_x}$)

<u>"Fact" #2</u>: For <u>any</u> AC Matrix **R** whose diagonal elements are σ_i^2

$$\det(\mathbf{R}) \leq \prod_{i=1}^{N} \sigma_i^2 \qquad \text{with equality iff } \mathbf{R} \text{ is a diagonal matrix}$$

Now, let **A** be any ON transform **y = Ax**.

Let $\mathbf{R_y}$ be the AC matrix of the transform coefficients… w/ diagonal elements σ_i^2

Recall Coding Gain: $$G_{TC} = \frac{\frac{1}{N}\sum_{i=0}^{N-1}\sigma_i^2}{\left[\prod_{i=0}^{N-1}\sigma_i^2\right]^{1/N}}$$

Facts #1 & #2 state that:

$$\det(\mathbf{R_x}) = \underbrace{\det(\mathbf{R_y}) \leq \prod_{i=1}^{N}\sigma_i^2}$$

Equality when $\mathbf{R_y}$ is diagonal… which is given by the K-L

$$G_{TC} \leq \frac{\frac{1}{N}\sum_{i=0}^{N-1}\sigma_i^2}{\det(\mathbf{R_x})}$$

This is an upper bound on TC Gain

$$G_{TC(K-L)} = \frac{\frac{1}{N}\sum_{i=0}^{N-1}\sigma_i^2}{\det(\mathbf{R_x})}$$

K-L Gives Largest!

Q: What transform is used in JPEG?

A: <u>The Discrete Cosine Transform (DCT)</u>

$\mathbf{c}_i$ is i^{th} row of $\mathbf{C}$

The **1-*D*** DCT has a transform matrix **C** with elements given by:

$$\mathbf{y} = \mathbf{C}\mathbf{x} \quad \Rightarrow \quad y_i = \mathbf{c}_i^T \mathbf{x}$$

$$C_{ij} = \begin{cases} \sqrt{\dfrac{1}{N}} \cos\left(\dfrac{(2j+1)i\pi}{2N}\right), & i = 0, \quad j = 0, 1, 2, \ldots, N-1 \\ \sqrt{\dfrac{2}{N}} \cos\left(\dfrac{(2j+1)i\pi}{2N}\right), & \begin{cases} i = 1, 2, \ldots, N-1 \\ j = 0, 1, 2, \ldots, N-1 \end{cases} \end{cases}$$

Each *i* (each row) is a "cos of *j*" at a different frequency

Note: The DCT is related to the DFT.

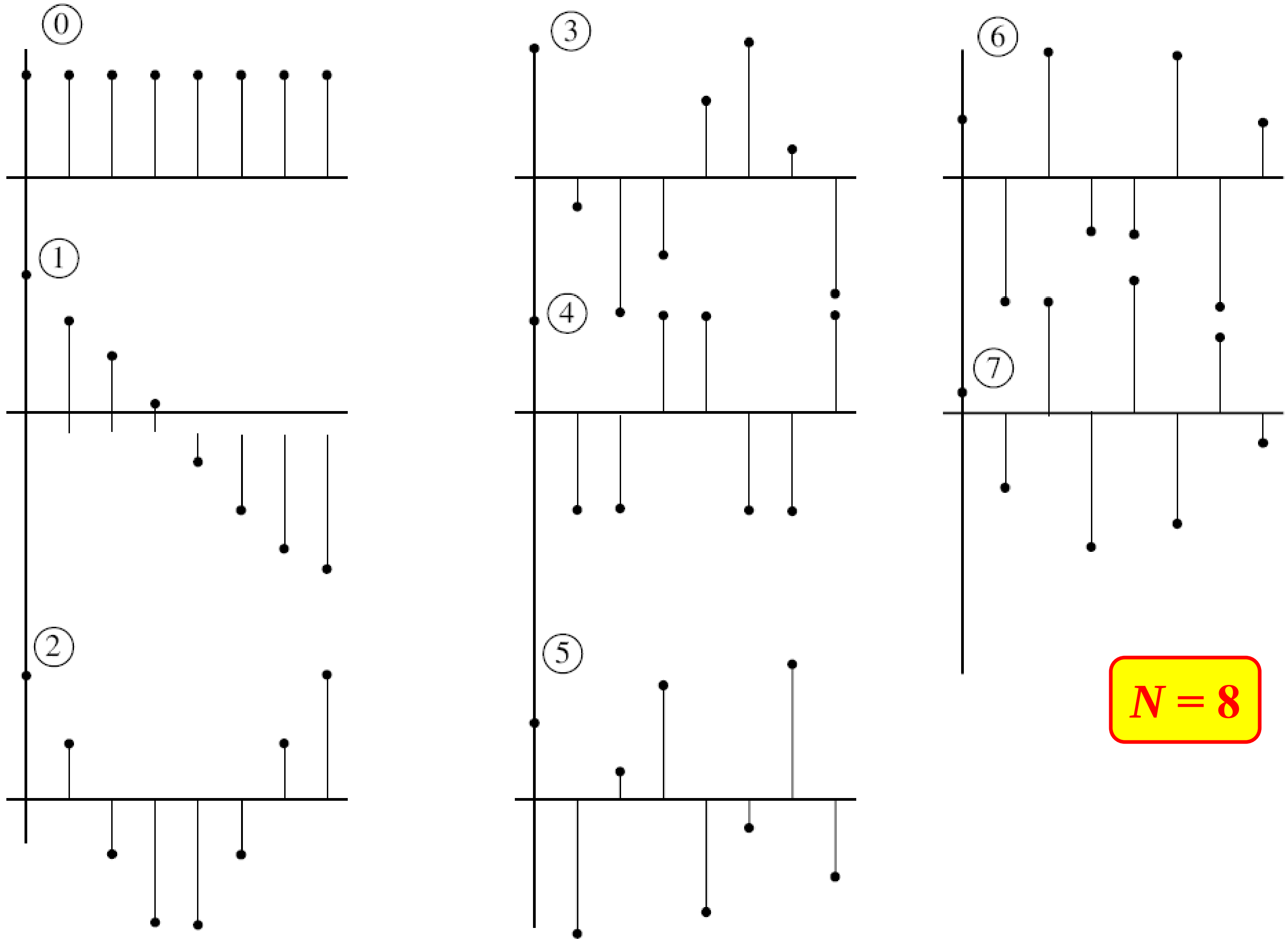
However, the DFT is less commonly used in compression… partly because it maps real-valued signals into complex-valued coefficients… which complicates the coding part of the compression algorithm.

Note that the DFT takes *N* real-valued samples into *N* complex-valued coefficients so that is really 2*N* real-valued numbers…

Q: Does this mean that the DFT double the amount of information?

$$C_{ij} = K_i \cos\left(\frac{(2j+1)i\pi}{2N}\right), \quad i, j = 0, 1, 2, \dots, N-1, \quad K_i = \begin{cases} 1, i = 0 \\ 2, i \neq 0 \end{cases}$$

1-*D* DCT



$N = 8$

FIGURE 13. 3 **Basis set for the discrete cosine transform. The numbers in the circles correspond to the row of the transform matrix.**

Let's see why is the DCT commonly used… Recall the 1st Order AR Model

$$x[n] = a_1 x[n-1] + \varepsilon[n] \qquad \text{ACF:} \quad R(k) = \left[\frac{\sigma_\varepsilon^2}{1-a_1^2}\right] a_1^k$$

with $|a_1| < 1$ and with $\varepsilon[n]$ a zero-mean white Gaussian Noise (WGN) process.

Such a process is called a 1st-Order Gauss-Markov (GM) Process

For here we'll notationally let $a_1 = \rho$ and we'll set σ_ε^2 so that $R(0) = 1$

$\Rightarrow$ $R(k) = \rho^k$ where ρ controls the "correlation decay" of the process

For a vector **x** taken from this process the AC matrix is then

$$\mathbf{R}_\mathbf{x} = \begin{bmatrix} 1 & \rho & \rho^2 & \cdots & \rho^{N-1} \\ \rho & 1 & \rho & \ddots & \vdots \\ \rho^2 & \rho & 1 & \ddots & \rho^2 \\ \vdots & \ddots & \ddots & \ddots & \rho \\ \rho^{N-1} & \cdots & \rho^2 & \rho & 1 \end{bmatrix}$$

"<u>**Fact**</u>": As $\rho \to 1$…. the DCT approximately diagonalizes this $\mathbf{R}_\mathbf{x}$

The DCT is approximately the KL transform for a 1st Order GM Process

A decent model for Images: DCT in JPEG

Q: How is the DCT used for Images (e.g., in JPEG)?

When we looked at <u>ON</u> transforms as <u>ON</u> matrices that operate on vectors we were really focused on the 1-D signal case (e.g., time signal)…

1-*D* signal → vector x

$$\mathbf{y} = \mathbf{A}\mathbf{x} \quad \Rightarrow \quad y_i = \sum_{j=1}^{N} A_{ij} x_j$$

j^{th} column of **A**

For DCT these are 1-D cosines

$$\mathbf{x} = \mathbf{A}^T \mathbf{y} \quad \Rightarrow \quad \mathbf{x} = \sum_{j=1}^{N} y_j \mathbf{a}_j$$

x is linear combo of basis vectors

But images are 2-D signals… so they are best viewed as matrices:

2-*D* signal → matrix X

Apply **A** to columns

Apply **A** to rows

$$\mathbf{Y} = \mathbf{A}\mathbf{X}\mathbf{A}^T \quad \Rightarrow \mathbf{Y}_{columns} = \mathbf{A}\mathbf{X} \quad \Rightarrow \mathbf{Y} = (\mathbf{A}\mathbf{X})\mathbf{A}^T$$

A "separable" 2-D transform

Apply $\mathbf{A}^T$ to columns

Apply $\mathbf{A}^T$ to rows

$$\mathbf{X} = \mathbf{A}^T \mathbf{Y}\mathbf{A} \quad \Rightarrow \quad \mathbf{X}_{columns} = \mathbf{A}^T \mathbf{Y} \quad \Rightarrow \quad \mathbf{X} = (\mathbf{A}^T \mathbf{Y})\mathbf{A}$$

So… in 1-D case the DCT coefficients come from "comparing" the signal vector to each vector in the basis vector set

If we work through the math for the 2-D case… and write it out for the DCT… we see a similar thing for the 2-D DCT:

$$Y_{lk} = \frac{K(l)K(k)}{4}\sum_{i=1}^{N}\sum_{j=1}^{N} \underbrace{\cos\left(\frac{(2i+1)l\pi}{2N}\right)\cos\left(\frac{(2j+1)k\pi}{2N}\right)}_{C_{ij}(l,k)} X_{ij}$$

$$= \frac{K(l)K(k)}{4}\sum_{i=1}^{N}\sum_{j=1}^{N} C_{ij}(l,k)X_{ij}$$

$$K(l) = \begin{cases} 1/\sqrt{2}, & l = 0 \\ 1, & l = otherwise \end{cases}$$

lkth DCT coefficient is found by "comparing" it to the *lk*th matrix **C**(*l*,*k*)

This is similar to the case for 1-D DCT… where "comparisons" were made to 1-D cosines of different frequencies

For 2-D DCT… the "comparisons" are made to <u>2-D cosines</u> of different <u>"mixed" frequencies</u> (<u>horizontal frequency</u> & <u>vertical frequency</u>)

2-D DCT Basis Matrices (8x8 case for JPEG)

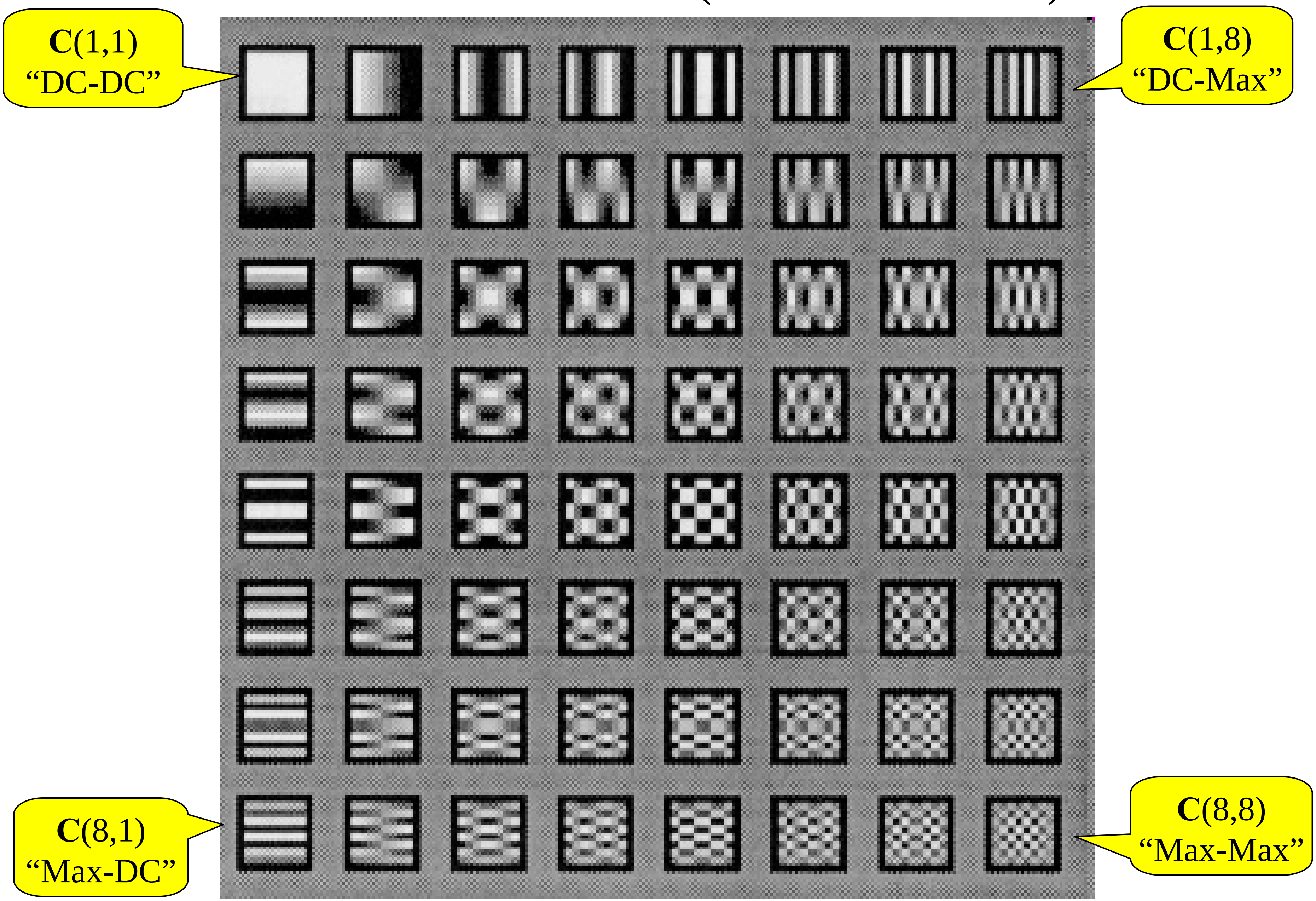
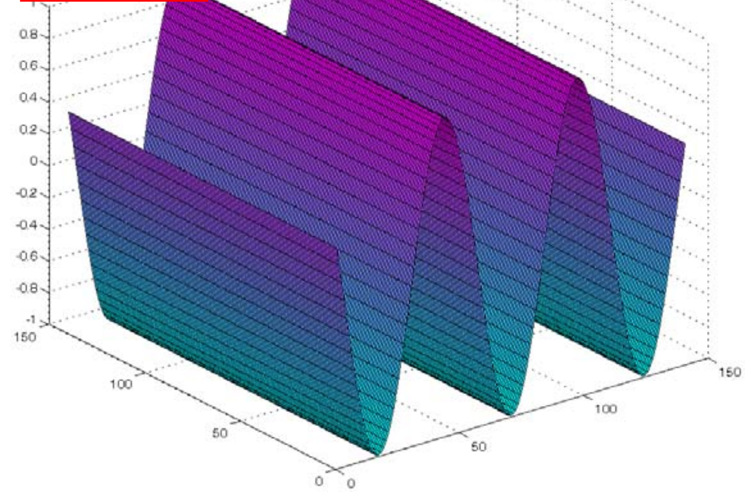
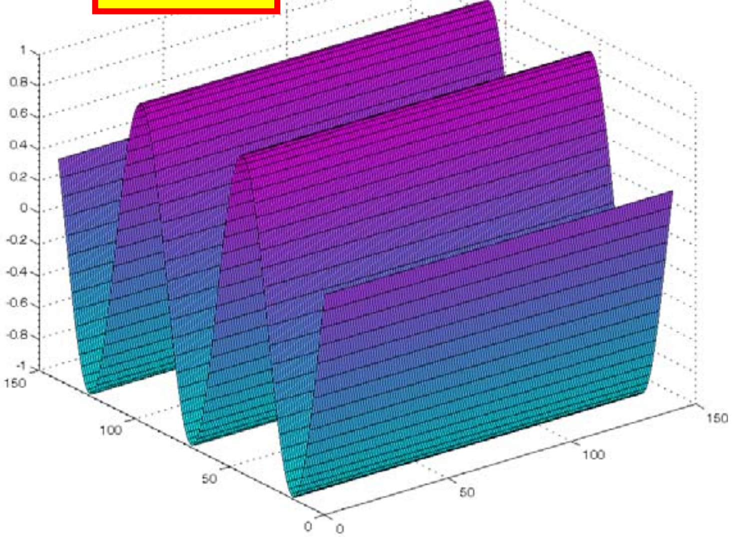


Image from http://www.cs.cf.ac.uk/Dave/Multimedia/node231.html

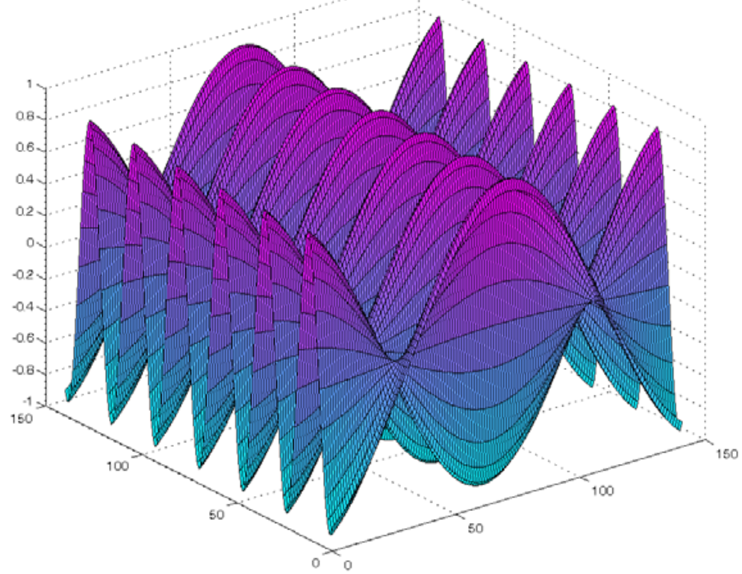
C(1,4)



C(4,1)



C(8,2)



Note: These were computed on a denser grid so it is easier to see their behavior

4.2.1. Overview of JPEG

What is JPEG?

- "Joint Photographic Expert Group". Voted as international standard in 1992.
- Works with color and grayscale images, e.g., satellite, medical, ...

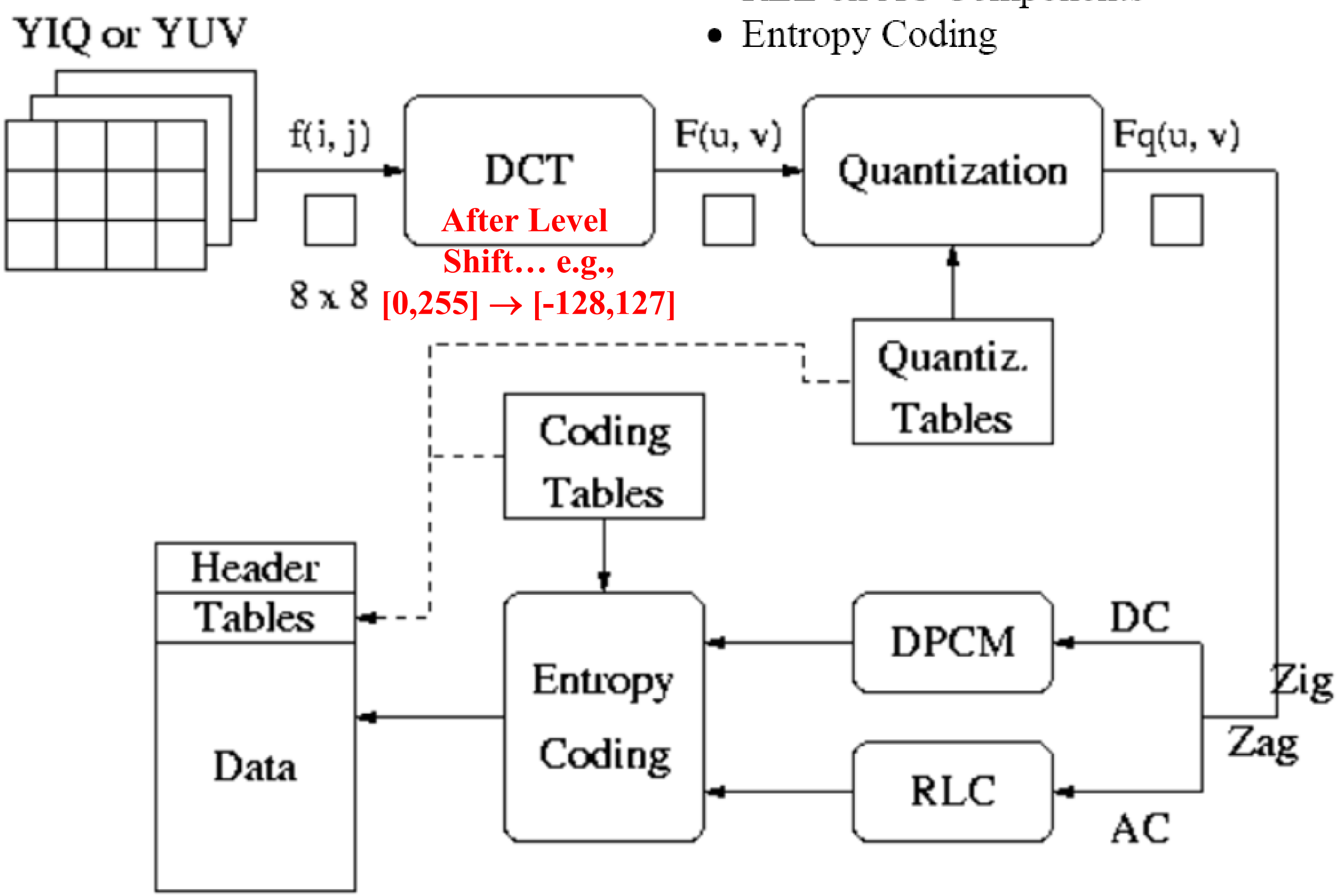
Motivation

- The *compression ratio* of lossless methods (e.g., Huffman, Arithmetic, LZW) is not high enough for image and video compression, especially when the distribution of pixel values is relatively flat.
- JPEG uses *transform coding*, it is largely based on the following observations:
  - Observation 1: A large majority of useful image contents change relatively slowly across images, i.e., it is unusual for intensity values to alter up and down several times in a small area, for example, within an 8 x 8 image block. Translate this into the spatial frequency domain, it says that, generally, lower spatial frequency components contain more information than the high frequency components which often correspond to less useful details and noises.
  - Observation 2: Pshchophysical experiments suggest that humans are more receptive to the loss of higher spatial frequency components than the loss of lower frequency components.

From http://www.cs.cf.ac.uk/Dave/Multimedia/node231.html

JPEG Coding Structure

- DCT (Discrete Cosine Transformation)
- Quantization
- Zigzag Scan
- DPCM on DC component
- RLE on AC Components
- Entropy Coding



From http://www.cs.cf.ac.uk/Dave/Multimedia/node231.html

Computing the 8x8 DCT

2-*D* signal $\rightarrow$ matrix X

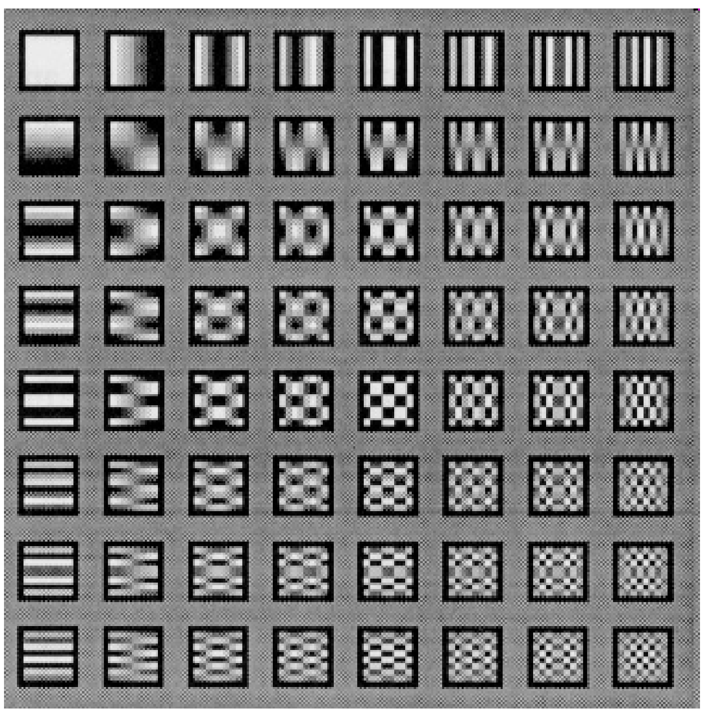
Apply 1-D **DCT** to columns

Apply 1-D **DCT** to columns

$$\mathbf{Y} = \mathbf{AXA}^T \quad \Rightarrow \mathbf{Y}_{columns} = \mathbf{AX} \quad \Rightarrow \mathbf{Y} = (\mathbf{AX})\mathbf{A}^T$$

How we ***DO*** it…

How we ***THINK*** about it…



$$Y_{lk} = \frac{K(l)K(k)}{4} \sum_{i=1}^{N} \sum_{j=1}^{N} \cos\left(\frac{(2i+1)l\pi}{2N}\right) \cos\left(\frac{(2j+1)k\pi}{2N}\right) X_{ij}$$

Image from
http://www.cs.cf.ac.uk/Dave/Multimedia/node231.html

Quantizing the 8x8 DCT Coefficients

Has zero as RL

Each of the 64 DCT coefficients in an 8x8 block are quantized using uniform mid-tread quantizers…

Each quantizer can have a different step size… step sizes are in “Quantization Table”

Each quantizer creates a “label” from a DCT coefficient:

$\lfloor x \rfloor =$ largest integer smaller than x

$$l_{ij} = \left\lfloor \frac{Y_{ij}}{Q_{ij}} + 0.5 \right\rfloor = \text{round}\left(\frac{Y_{ij}}{Q_{ij}} \right)$$

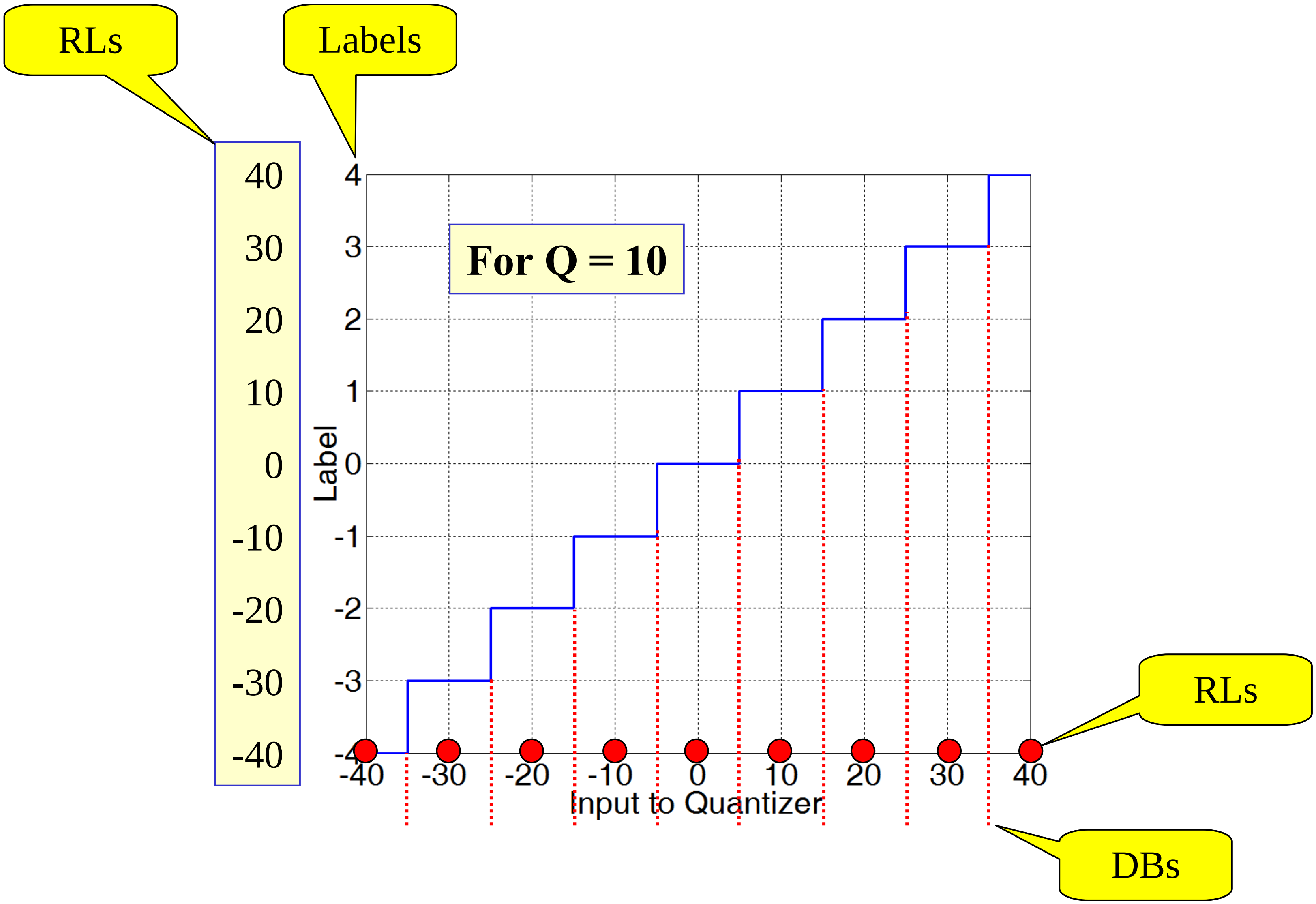
l_{ij} = label
Y_{ij} = ij^{th} DCT coefficient
Q_{ij} = ij^{th} Quant. Table Entry

Reconstruction Levels

$$\hat{Y}_{ij} = l_{ij} Q_{ij}$$

At Decoder

What does such a quantizer look like?



Rationale Behind the Quantization Tables

Table values are part of JPEG standard… but can also be user specified.

Choice of table controls quality… usually just scale standard table up/down

Example Quantization Table Values

Small… Fine Quantization

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

Large… Coarse Quantization

Tables have larger values for High Frequency coefficients…

- High freq coeffs tend to be small… quantizing to zero causes small contribution to MSE
- Also… human visual perception not as sensitive to errors in high freq components

For color… different tables for “luminence” and “chrominance” components

- Exploit difference in human perception of errors in these components

Original DCT Values

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

Quantizer Table Values

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

Quantized Labels (Mostly Zeros!!!)

2	1	0	0	0	0	0	0
−9	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0		
0	0	0	0	0	0		

Reconstructed DCT Values

32	11	0	0	0	0	0	0
−108	0	0	0	0	0	0	0
42	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

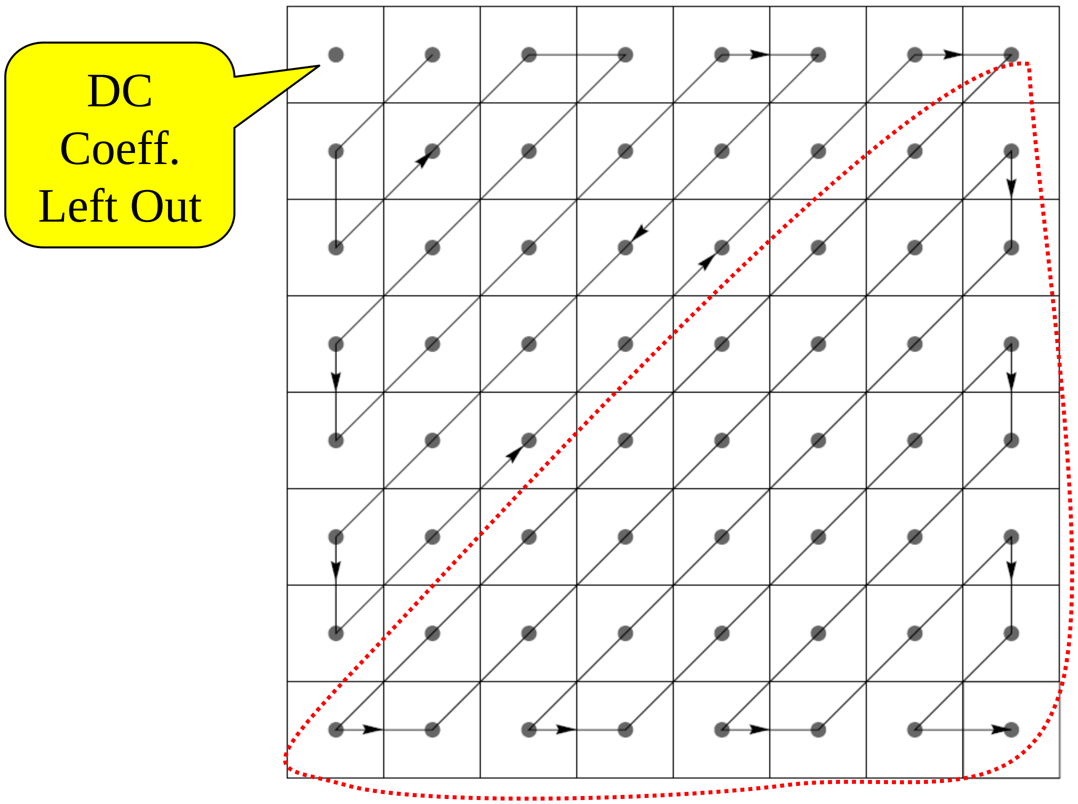
Original 8x8 Block

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

Reconstructed 8x8 Block from Previous Example

123	122	122	121	120	120	119	119
121	121	121	120	119	118	118	118
121	121	120	119	119	118	117	117
124	124	123	122	122	121	120	120
130	130	129	129	128	128	128	127
141	141	140	140	139	138	138	137
152	152	151	151	150	149	149	148
159	159	158	157	157	156	155	155

Zig Zag Scan of DCT Coefficients of 8x8 Block



Rationale: Small Coeffs w/ large Q values… Lots of zeros… ZigZag increases long strings of zeros…

Enables "Run Length Enoding"

Labels for Quantized DCT

2	1	0	0	0	0	0	0
−9	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

1
-9
3
0
0
0
.
.
.
0

RLE… sends (Skip, Value) using a Huffman Code

Coding DC Coefficients

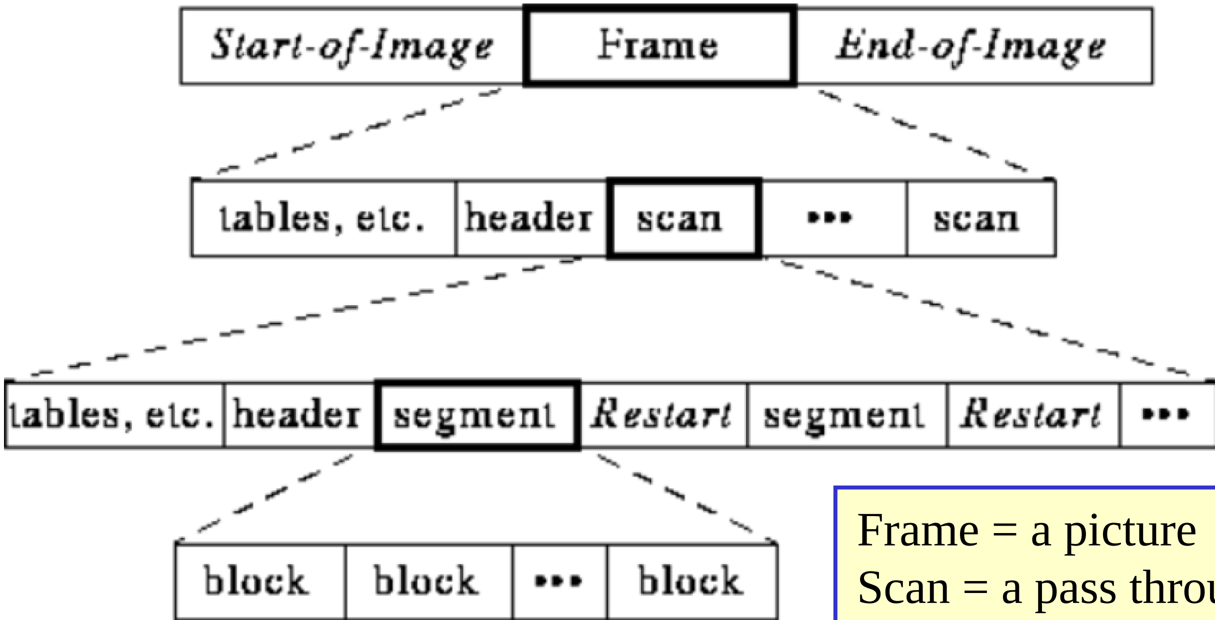
DC Coeff is essentially the average value of the 8x8 block

Expect this to vary slowly from block to block…

Code differences between successive blocks…. Use a form of Huffman

JPEG File Structure

From http://www.cs.cf.ac.uk/Dave/Multimedia/node231.html



Frame = a picture
Scan = a pass through the pixels (e.g., red comp.)
Segment = a group of 8x8 blocks

- Frame header:
  sample precision
  (width, height) of image
  number of components
  unique ID (for each component)
  horizontal/vertical sampling factors (for each component)
  quantization table to use (for each component)
- Scan header
  Number of components in scan
  component ID (for each component)
  Huffman table for each component (for each component)
- Misc. (can occur between headers)
  Quantization tables
  Huffman Tables
  Arithmetic Coding Tables
  Comments
  Application Data