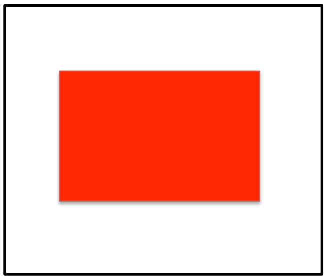
Lecture 5: Compression I

Reading: book chapter 6, section 3 &5
chapter 7, section 1, 2, 3, 4, 8

This Week's Schedule

- Today:
  - The concept behind compression
  - Rate distortion theory
  - Image compression via DCT
- Wed.:
  - Speech compression via Prediction
  - Video compression via IPB and motion estimation/compensation

Motivation

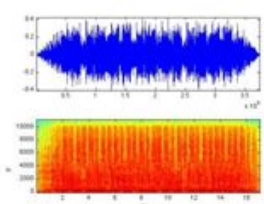


- A simple example: a 10x10 colored image, how many bits are required to represent this image:
  - Without compression
  - With compression

The concept behind compression is to extract and remove redundancy that naturally exists in media data;
In many cases, we can remove information without affecting the visual/ audio effect, because human eyes/ears have limited sensitivity

Redundancy in Media Data

- Medias (speech, audio, image, video) are not random collection of signals, but exhibit a similar structure in local neighborhood
  - **Temporal redundancy**: current and next signals are very similar (smooth media: speech, audio, video)
  - **Spatial redundancy**: the pixels' intensities and colors in local regions are very similar (image, video)
  - **Spectral redundancy**: When the data is mapped into the frequency domain, a few frequencies dominate over the others



Lossless Compression

- Lossless compression
  - Compress the signal but can reproduce **the exact** original signal
  - Used for archival purposes and often medical imaging, technical drawings
  - Example 1: Run Length Encoding (BMP, PCX)
    BBBBEEEEEEEECCCCDAAAAA → 4B8E4C1D5A
  - Example 2: Lempel-Ziv-Welch (LZW): adaptive dictionary, dynamically create a dictionary of strings to efficiently represent messages, used in GIF & TIFF
  - Example 3: Huffman coding: the length of the codeword to present a symbol (or a value) scales inversely with the probability of the symbol's appearance, used in PNG, MNG, TIFF

Huffman Coding

Fixed-length coding

Symbol	Probability	Binary Code	Code length
A	0.28	000	3
B	0.2	001	3
C	0.17	010	3
D	0.17	011	3
E	0.1	100	3
F	0.05	101	3
G	0.02	110	3
H	0.01	111	3

Average symbol length=3

Symbol	Probability	Binary Code	Code length
A	0.28	00	1
B	0.2	10	3
C	0.17	010	3
D	0.17	011	3
E	0.1	110	3
F	0.05	1110	4
G	0.02	11110	5
H	0.01	11111	5

Average symbol length=2.63

The length of the codeword to present a symbol (or a value) scales inversely with the probability of the symbol's appearance.

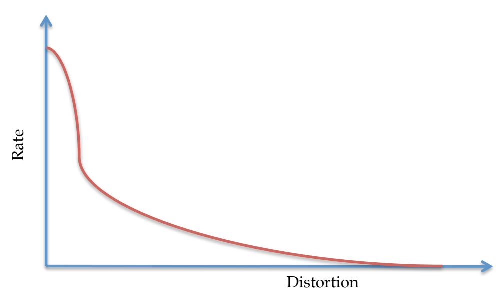
Lossy Compression

- The compressed signal after de-compressed, does not match the original signal
  - Compression leads to some signal **distortion**
  - Suitable for natural images such as photos in applications where minor (sometimes imperceptible) loss of fidelity is acceptable to achieve a **substantial reduction** in bit rate.
- Types
  - **Color space reduction**: reduce 24→8bits via color lookup table
  - **Chrominance subsampling**: from 4:4:4 to 4:2:2, 4:1:1, 4:2:0
    - eye perceives spatial changes of brightness more sharply than those of color, by averaging or dropping some of the chrominance information
  - **Transform coding** (or perceptual coding): Fourier transform (DCT, wavelet) followed by quantization and entropy coding

Today's focus

Rate Distortion Theory

- As the degree of compression increases, the number of bits used to represent the image reduces, and this increases the distortion



Distortion Measures

The three most commonly used distortion measures in image are:

- *mean square error* (MSE) σ^2,

$$\sigma^2 = \frac{1}{N}\sum_{n=1}^{N}(x_n - y_n)^2$$

where x_n, y_n, and N are the input data sequence, reconstructed data sequence, and length of the data sequence respectively.

- *signal to noise ratio* (SNR), in decibel units (dB),

$$SNR = 10 \cdot \log_{10}\left(\frac{\frac{1}{N}\sum_{n=1}^{N} x_n^2}{\sigma^2}\right)$$

- *peak signal to noise ratio* (PSNR), in decibel units (dB),

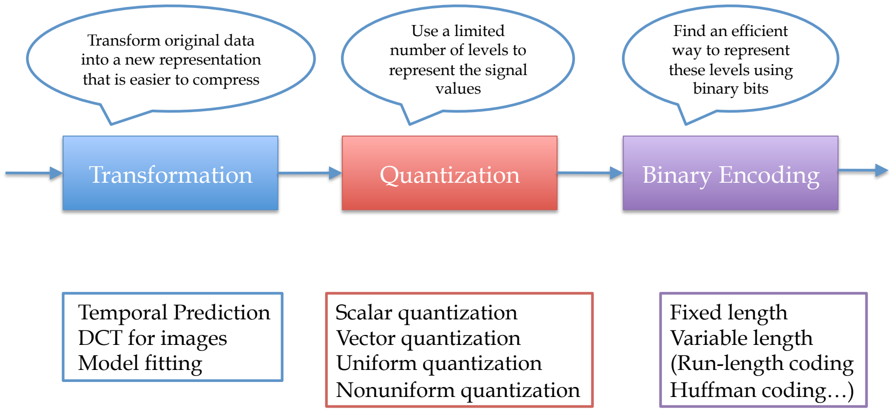
$$PSNR = 10 \cdot \log_{10}\left(\frac{255^2}{\sigma^2}\right) = 20 \cdot \log_{10}\left(\frac{255}{\sigma}\right)$$

Lossy Compression

- The compressed signal after de-compressed, does not match the original signal
  - Compression leads to some signal distortion
  - Suitable for natural images such as photos in applications where minor (sometimes imperceptible) loss of fidelity is acceptable to achieve a **substantial reduction** in bit rate.
- Types
  - **Color space reduction**: reduce 24→8bits via color lookup table
  - **Chrominance subsampling**: from 4:4:4 to 4:2:2, 4:1:1, 4:2:0, eye perceives spatial changes of brightness more sharply than those of color, by averaging or dropping some of the chrominance information
  - **Transform coding** (or perceptual coding): Fourier transform (DCT, wavelet) followed by quantization and entropy coding

Today's focus

A Typical Compression System



Why Transformation?

- Goal of transformation:
  - To yield a more efficient representation of the original samples.
  - The transformed parameters should require fewer bits to code.
- Types of transformation used:
  - For speech coding: prediction
    - Code predictor and prediction error sample
  - For audio coding: subband decomposition
    - Code subband samples
  - **For image coding: DCT and wavelet transforms**
    - Code DCT/wavelet coefficients

Image: Transformation

- Represent an image as the linear combination of some basis images and specify the linear coefficients
- **Instead of storing the original image, now store the linear coefficients $\{t_k\}$**

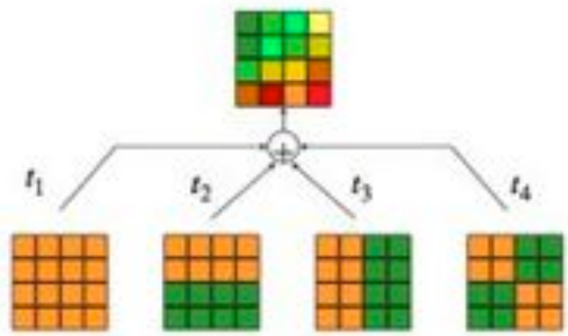


Image: Transformation II

- Optimality Criteria:
  - **Energy compaction**: a few basis images are sufficient to represent a typical image.
  - **Decorrelation**: coefficients for separate basis images are uncorrelated.
- Karhunen Loeve Transform (KLT) is the Optimal transform for a given covariance matrix of the underlying signal.
- Discrete Cosine Transform (DCT) is close to KLT for images that can be modeled by a first order Markov process (i.e., a pixel only depends on its previous pixel).

1D Transformation

$$s = \begin{bmatrix} s_0 \\ s_1 \\ \vdots \\ s_{N-1} \end{bmatrix}$$

The inverse transform says that **s can be represented as the sum of** ***N basis vectors***

$$s = t_0 u_0 + t_1 u_1 + \ldots + t_{N-1} u_{N-1}$$

$$u_k = \begin{bmatrix} u_{k,0} \\ u_{k,1} \\ \vdots \\ u_{k,N-1} \end{bmatrix}$$

where $\mathbf{u}_k$ *corresponds to the k-th transform kernel :*

$$t_k = \begin{bmatrix} s_0 & s_1 & .. & s_{N-1} \end{bmatrix} \bullet \begin{bmatrix} u_{k,0} \\ u_{k,1} \\ \vdots \\ u_{k,N-1} \end{bmatrix}^* = \sum_{n=1}^{N-1} u_{k,n}^* \cdot s_n$$

The forward transform says that the expansion coefficient t_k *can be determined by the inner product of* ***s with*** $\mathbf{u}_k$

1D Discrete Cosine Transform

$$t_k = \sum_{n=1}^{N-1} u_{k,n}^* \cdot s_n$$

$$u_{k,n} = \alpha(k)\cos\left(\frac{\pi k}{2N}(2n+1)\right)$$

$$\alpha(0) = \sqrt{\frac{1}{N}}, \quad \alpha(k) = \sqrt{\frac{2}{N}}, \quad k = 1,2,\ldots,N-1$$

$$u_k = \alpha(k)\begin{bmatrix} \cos\left(\frac{\pi k}{2N}\right) \\ \cos\left(\frac{\pi k}{2N}\cdot 3\right) \\ \vdots \\ \cos\left(\frac{\pi k}{2N}(2N+1)\right) \end{bmatrix}$$

Example 4-Point DCT

Using $u_{k,n} = \alpha(k)\cos\left(\frac{k\pi}{2*4}(2n+1)\right), \alpha(0) = \sqrt{\frac{1}{4}} = \frac{1}{2}, \alpha(k) = \sqrt{\frac{2}{4}} = \sqrt{\frac{1}{2}}, k \neq 0,$

1D DCT basis are: $\mathbf{u}_0 = \frac{1}{2}\begin{bmatrix}1\\1\\1\\1\end{bmatrix}, \mathbf{u}_1 = \frac{1}{\sqrt{2}}\begin{bmatrix}\cos\left(\frac{\pi}{8}\right)\\\cos\left(\frac{3\pi}{8}\right)\\\cos\left(\frac{5\pi}{8}\right)\\\cos\left(\frac{7\pi}{8}\right)\end{bmatrix} = \frac{1}{\sqrt{2}}\begin{bmatrix}0.9239\\0.3827\\-0.3827\\-0.9239\end{bmatrix}, \mathbf{u}_2 = \frac{1}{\sqrt{2}}\begin{bmatrix}\cos\left(\frac{\pi}{4}\right)\\\cos\left(\frac{3\pi}{4}\right)\\\cos\left(\frac{5\pi}{4}\right)\\\cos\left(\frac{7\pi}{4}\right)\end{bmatrix} = \frac{1}{2}\begin{bmatrix}1\\-1\\-1\\1\end{bmatrix}, \mathbf{u}_3 = \frac{1}{\sqrt{2}}\begin{bmatrix}\cos\left(\frac{3\pi}{8}\right)\\\cos\left(\frac{9\pi}{8}\right)\\\cos\left(\frac{15\pi}{8}\right)\\\cos\left(\frac{21\pi}{8}\right)\end{bmatrix} = \frac{1}{\sqrt{2}}\begin{bmatrix}0.3827\\-0.9239\\0.9239\\-0.3827\end{bmatrix}$

For $\mathbf{s} = \begin{bmatrix}2\\4\\5\\3\end{bmatrix}$, determine the transform coefficients t_k. Also determine the reconstructed vector from all coefficients and two largest coefficients.

Group Exercise !

2D Tranformation

$$\mathbf{S} = \begin{bmatrix} S_{0,0} & S_{0,1} & \cdots & S_{0,N-1} \\ S_{1,0} & S_{1,1} & \cdots & S_{1,N-1} \\ \cdots & \cdots & \cdots & \cdots \\ S_{M-1,0} & S_{M-1,1} & \cdots & S_{M-1,N-1} \end{bmatrix} \qquad \mathbf{S} = T_{0,0}\mathbf{U}_{0,0} + T_{0,1}\mathbf{U}_{0,1} + \ldots + T_{M-1,N-1}\mathbf{U}_{M-1,N-1}$$

$$\mathbf{U}_{k,l} = \mathbf{u}_k(\mathbf{u}_l)^T = \begin{bmatrix} u_{k,0} \\ u_{k,1} \\ \cdots \\ u_{k,N-1} \end{bmatrix} \begin{bmatrix} u^*_{l,0} & u^*_{l,1} & \cdots & u^*_{l,N-1} \end{bmatrix} = \begin{bmatrix} u_{k,0}u^*_{l,0} & u_{k,0}u^*_{l,1} & \cdots & u_{k,0}u^*_{l,N-1} \\ u_{k,1}u^*_{l,0} & u_{k,1}u^*_{l,1} & \cdots & u_{k,1}u^*_{l,N-1} \\ \cdots & \cdots & \cdots & \cdots \\ u_{k,N-1}u^*_{l,0} & u_{k,N-1}u^*_{l,1} & \cdots & u_{k,N-1}u^*_{l,N-1} \end{bmatrix}$$

$$T_{k,l} = (\mathbf{U}_{k,l}, \mathbf{S}) = \sum_{m=0}^{M-1}\sum_{n=0}^{N-1} U^*_{k,l;m,n} S_{m,n}$$

2D DCT

MATLAB ☺ dct2

$$\mathbf{a}_{k,N} = \alpha(k)\begin{bmatrix} \cos\left(\frac{k\pi}{2N}1\right) \\ \cos\left(\frac{k\pi}{2N}3\right) \\ \cdots \\ \cos\left(\frac{k\pi}{2N}(2N-1)\right) \end{bmatrix},\quad \alpha(0)=\sqrt{\frac{1}{N}},\ \alpha(k)=\sqrt{\frac{2}{N}},\ k=1,2,\ldots,N-1$$

$$\mathbf{U}_{k,l,M,N} = \mathbf{a}_{k,M}[\mathbf{a}_{l,N}]^T$$

$$= \alpha(k)\alpha(l)\begin{bmatrix} \cos\left(\frac{k\pi}{2M}1\right)\cos\left(\frac{l\pi}{2N}1\right) & \cos\left(\frac{k\pi}{2M}1\right)\cos\left(\frac{l\pi}{2N}3\right) & \cdots & \cos\left(\frac{k\pi}{2M}1\right)\cos\left(\frac{l\pi}{2N}(2N-1)\right) \\ \cos\left(\frac{k\pi}{2M}3\right)\cos\left(\frac{l\pi}{2N}1\right) & \cos\left(\frac{k\pi}{2M}3\right)\cos\left(\frac{l\pi}{2N}3\right) & \cdots & \cos\left(\frac{k\pi}{2M}3\right)\cos\left(\frac{l\pi}{2N}(2N-1)\right) \\ \cdots & \cdots & \cdots & \\ \cos\left(\frac{k\pi}{2M}(2M-1)\right)\cos\left(\frac{l\pi}{2N}1\right) & \cos\left(\frac{k\pi}{2M}(2M-1)\right)\cos\left(\frac{l\pi}{2N}3\right) & \cdots & \cos\left(\frac{k\pi}{2M}(2M-1)\right)\cos\left(\frac{l\pi}{2N}(2N-1)\right) \end{bmatrix}$$

4x4 DCT

Using $a_{k,n} = \alpha(k)\cos\left(\frac{k\pi}{2*4}(2n+1)\right), \alpha(0)=\sqrt{\frac{1}{4}}=\frac{1}{2}, \alpha(k)=\sqrt{\frac{2}{4}}=\sqrt{\frac{1}{2}}, k \neq 0,$

1D DCT basis are: $\mathbf{a}_0 = \frac{1}{2}\begin{bmatrix}1\\1\\1\\1\end{bmatrix}, \mathbf{a}_1 = \frac{1}{\sqrt{2}}\begin{bmatrix}\cos\left(\frac{\pi}{8}\right)\\ \cos\left(\frac{3\pi}{8}\right)\\ \cos\left(\frac{5\pi}{8}\right)\\ \cos\left(\frac{7\pi}{8}\right)\end{bmatrix} = \frac{1}{\sqrt{2}}\begin{bmatrix}0.9239\\0.3827\\-0.3827\\-0.9239\end{bmatrix}, \mathbf{a}_2 = \frac{1}{\sqrt{2}}\begin{bmatrix}\cos\left(\frac{\pi}{4}\right)\\ \cos\left(\frac{3\pi}{4}\right)\\ \cos\left(\frac{5\pi}{4}\right)\\ \cos\left(\frac{7\pi}{4}\right)\end{bmatrix} = \frac{1}{2}\begin{bmatrix}1\\-1\\-1\\1\end{bmatrix}, \mathbf{a}_3 = \frac{1}{\sqrt{2}}\begin{bmatrix}\cos\left(\frac{3\pi}{8}\right)\\ \cos\left(\frac{9\pi}{8}\right)\\ \cos\left(\frac{15\pi}{8}\right)\\ \cos\left(\frac{21\pi}{8}\right)\end{bmatrix} = \frac{1}{\sqrt{2}}\begin{bmatrix}0.3827\\-0.9239\\0.9239\\-0.3827\end{bmatrix}$

using $\mathbf{U}_{k,l} = \mathbf{a}_k[\mathbf{a}_l]^T$ yields:

$$\mathbf{U}_{0,0} = \frac{1}{4}\begin{bmatrix}1&1&1&1\\1&1&1&1\\1&1&1&1\\1&1&1&1\end{bmatrix}, \mathbf{U}_{0,2} = \frac{1}{4}\begin{bmatrix}1&-1&-1&1\\1&-1&-1&1\\1&-1&-1&1\\1&-1&-1&1\end{bmatrix}, \mathbf{U}_{2,0} = \frac{1}{4}\begin{bmatrix}1&1&1&1\\-1&-1&-1&-1\\-1&-1&-1&-1\\1&1&1&1\end{bmatrix}, \mathbf{U}_{2,2} = \frac{1}{4}\begin{bmatrix}1&-1&-1&1\\-1&1&1&-1\\-1&1&1&-1\\1&-1&-1&1\end{bmatrix}, \ldots$$

Matlab Demo

```
x=imread('avatar1.jpg');
y=x(:,:,1)*0.299 + x(:,:,2) *0.587 + x(:,:,3)*0.114; %convert RGB → Y
subplot(2,2,1); imshow(y);
[m,n]=size(y);
m=floor(m/8); n=floor(n/8);
for i=1:m,
  for j=1:n,
    z=y((i-1)*8+1:i*8, (j-1)*8+1:j*8);
    d(i,j,:,:)=dct2(z);
    zz((i-1)*8+1:i*8, (j-1)*8+1:j*8)=d(i,j,:,:);
     end;
end;
y(9*8+1:80,9*8+1:80)
squeeze(d(10,10,:,:))
surf(squeeze(d(10,10,:,:)))
subplot(2,2,2); imshow(zz);
```

14	12	8	8	8	9	8	8
13	12	8	7	8	8	8	7
11	10	8	7	7	8	7	7
10	8	8	7	7	7	7	7
8	8	8	7	7	7	7	7
7	7	8	8	7	7	7	8
7	7	8	9	7	7	7	8
7	7	10	9	8	7	7	10

↓

64.3750	**5.2042**	3.1427	0.4215	0.6250	-0.1329	0.7277	-0.3977
4.0089	**5.2874**	4.1819	5.1303	-0.9152	-0.8085	-1.2858	-0.4849
3.7400	0.4698	0.5366	-0.2701	0.7209	-0.3769	-0.0884	-0.5441
-0.1462	-0.3201	-0.0846	0.4658	-0.0402	0.1594	-0.5902	-0.5669
0.6250	-0.1584	0.1237	-0.3610	0.3750	-0.0028	0.6253	0.5311
-0.4242	-0.1350	-0.1379	0.2629	0.5487	0.7664	0.2623	-0.6434
0.2097	-0.5612	-0.0884	-0.5091	0.1073	-0.1914	0.7134	-0.1806
-0.2003	0.1186	-0.1791	0.1318	-0.4151	-0.1678	-0.0892	-0.5197

Recover via IDCT

```
% No compression
for i=1:m,
  for j=1:n,
      u=squeeze(d(i,j,:,:));
      w((i-1)*8+1:i*8, (j-1)*8+1:j*8)=idct2(u);
  end;
end;
subplot(2,2,3); imshow(w,[0,255]);

% Some level of compression
for i=1:m,
  for j=1:n,
     u=squeeze(d(i,j,:,:));
      f= zeros(8,8);
      k=1;
     f(1:k,1:k)=u(1:k,1:k);  % only consider the first kxk coefficient
      v((i-1)*8+1:i*8, (j-1)*8+1:j*8)=idct2(f);
      end;
end;
subplot(2,2,4); imshow(v,[0,255]);
```

Matlab Results

an original image block

ans =

14	12	8	8	8	9	8	8
13	12	8	7	8	8	8	7
11	10	8	7	7	8	7	7
10	8	8	7	7	7	7	7
8	8	8	7	7	7	7	7
7	7	8	8	7	7	7	8
7	7	8	9	7	7	7	8
7	7	10	9	8	7	7	10

its DCT coefficients

ans =

64.3750	5.2042	3.1427	0.4215	0.6250	-0.1329	0.7277	-0.3977
4.0089	5.2874	4.1819	5.1303	-0.9152	-0.8085	-1.2858	-0.4849
3.7400	0.4698	0.5366	-0.2701	0.7209	-0.3769	-0.0884	-0.5441
-0.1462	-0.3201	-0.0846	0.4658	-0.0402	0.1594	-0.5902	-0.5669
0.6250	-0.1584	0.1237	-0.3610	0.3750	-0.0028	0.6253	0.5311
-0.4242	-0.1350	-0.1379	0.2629	0.5487	0.7664	0.2623	-0.6434
0.2097	-0.5612	-0.0884	-0.5091	0.1073	-0.1914	0.7134	-0.1806
-0.2003	0.1186	-0.1791	0.1318	-0.4151	-0.1678	-0.0892	-0.5197

recovered image block

ans =

14	12	8	8	8	9	8	8
13	12	8	7	8	8	8	7
11	10	8	7	7	8	7	7
10	8	8	7	7	7	7	7
8	8	8	7	7	7	7	7
7	7	8	8	7	7	7	8
7	7	8	9	7	7	7	8
7	7	10	9	8	7	7	10

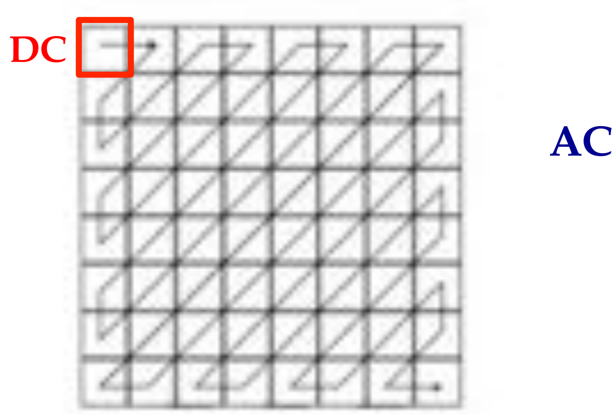
recovered with first 4x4 coefficient

ans =

14	12	9	7	8	9	8	8
13	11	8	7	8	8	8	7
11	10	8	7	7	8	7	7
10	9	8	7	7	7	7	7
8	8	8	7	7	7	7	7
7	7	8	8	7	7	7	8
6	8	9	8	7	7	8	8
6	8	9	9	8	7	8	9

Compression via DCT

Zig-Zag Ordering of DCT Coefficients



Zig-Zag ordering: converting a 2D matrix into a 1D array, so that the frequency (horizontal+vertical) increases in this order, and the coefficient variance (average of magnitude square) decreases in this order.

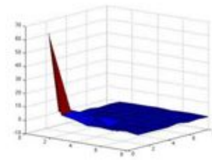
After Zig-Zag Ordering

Original block

14	12	8	8	8	9	8	8
13	12	8	7	8	8	8	7
11	10	8	7	7	8	7	7
10	8	8	7	7	7	7	7
8	8	8	7	7	7	7	7
7	7	8	8	7	7	7	8
7	7	8	9	7	7	7	8
7	7	10	9	8	7	7	10

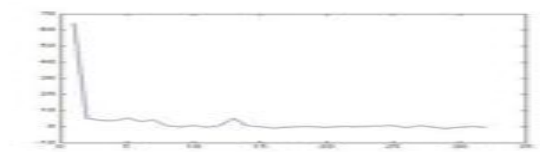
DCT results

64.3750	**5.2042**	3.1427	0.4215	0.6250	-0.1329	0.7277	-0.3977
4.0089	**5.2874**	4.1819	5.1303	-0.9152	-0.8085	-1.2858	-0.4849
3.7400	0.4698	0.5366	-0.2701	0.7209	-0.3769	-0.0884	-0.5441
-0.1462	-0.3201	-0.0846	0.4658	-0.0402	0.1594	-0.5902	-0.5669
0.6250	-0.1584	0.1237	-0.3610	0.3750	-0.0028	0.6253	0.5311
-0.4242	-0.1350	-0.1379	0.2629	0.5487	0.7664	0.2623	-0.6434
0.2097	-0.5612	-0.0884	-0.5091	0.1073	-0.1914	0.7134	-0.1806
-0.2003	0.1186	-0.1791	0.1318	-0.4151	-0.1678	-0.0892	-0.5197



Zig-Zag ordering

64.3750 5.2042 4.0089 3.7400, 5.2874, 3.1427, 4.1819, 0.4698, -0.1462, 0.6250, -0.3201, 0.5366, 5.1303, 0.6250, -0.1329, -0.9152, -0.2701, -0.0846 -0.1584, -0.4242, 0.2097, -0.1350, 0.1237, 0.4658, 0.7209, -0.8085, 0.7277, -0.3977, -1.2858, -0.3769, -0.0402, -0.3610 ………



Recap: A Typical Compression System

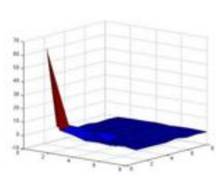
Transform original data into a new representation that is easier to compress

Use a limited number of levels to represent the signal values

Find an efficient way to represent these levels using binary bits

→ Transformation → Quantization → Binary Encoding →

DCT for images



Scalar quantization
Vector quantization
Uniform quantization
Nonuniform quantization

Fixed length
Variable length
(Run-length coding
Huffman coding…)

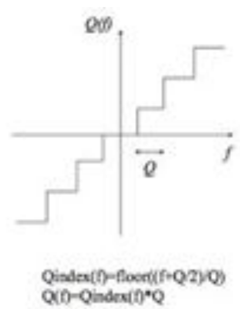
Quantization

- Reduce the number of distinct output values to a much smaller set.
- Main source of the “loss" in lossy compression.

- Three different forms of quantization.
  - **Uniform**: midrise and midtread quantizers.
  - **Nonuniform**: companded quantizer.
  - **Vector Quantization**.

Uniform Quantization

- A uniform scalar quantizer partitions the domain of input values into equally spaced intervals, except possibly at the two outer intervals.
  - The output or reconstruction value corresponding to each interval is taken to be the midpoint of the interval.
  - The length of each interval is referred to as the *step size Q*



Quantizing DCT Coefficients

- Use **uniform quantizer** on each coefficient
- Different coefficient is quantized with different step-size (Q):
  - Human eye is more sensitive to low frequency components
  - Low frequency coefficients with a smaller Q
  - High frequency coefficients with a larger Q
  - Specified in a normalization matrix
  - Normalization matrix can then be scaled by a scale factor (QP), i.e. Actual quantization = QP * Q, QP=1, 2,3, 4….

JPEG Quantization

- For Luminance component

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 =

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

Original block

14	12	8	8	8	9	8	8
13	12	8	7	8	8	8	7
11	10	8	7	7	8	7	7
10	8	8	7	7	7	7	7
8	8	8	7	7	7	7	7
7	7	8	8	7	7	7	8
7	7	8	9	7	7	7	8
7	7	10	9	8	7	7	10

DCT results

64.3750	**5.2042**	3.1427	0.4215	0.6250	-0.1329	0.7277	-0.3977
4.0089	**5.2874**	4.1819	5.1303	-0.9152	-0.8085	-1.2858	-0.4849
3.7400	0.4698	0.5366	-0.2701	0.7209	-0.3769	-0.0884	-0.5441
-0.1462	-0.3201	-0.0846	0.4658	-0.0402	0.1594	-0.5902	-0.5669
0.6250	-0.1584	0.1237	-0.3610	0.3750	-0.0028	0.6253	0.5311
-0.4242	-0.1350	-0.1379	0.2629	0.5487	0.7664	0.2623	-0.6434
0.2097	-0.5612	-0.0884	-0.5091	0.1073	-0.1914	0.7134	-0.1806
-0.2003	0.1186	-0.1791	0.1318	-0.4151	-0.1678	-0.0892	-0.5197

Quantized & dequantized DCT results

4	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
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

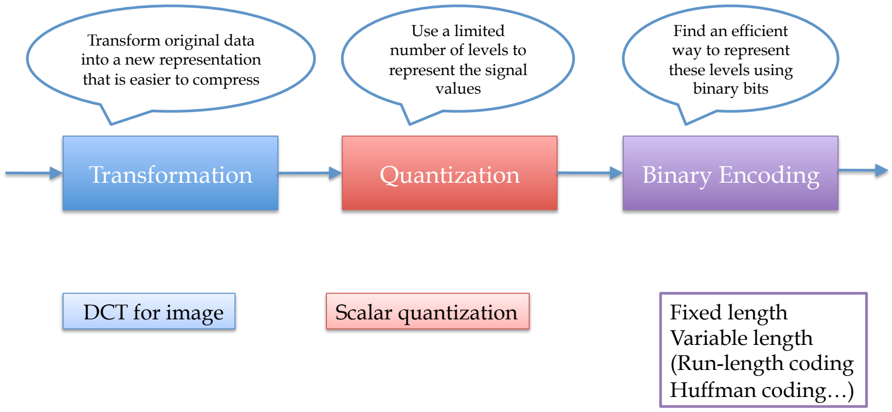
→

64	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
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

Recovered block

8	8	8	8	8	8	8	8
8	8	8	8	8	8	8	8
8	8	8	8	8	8	8	8
8	8	8	8	8	8	8	8
8	8	8	8	8	8	8	8
8	8	8	8	8	8	8	8
8	8	8	8	8	8	8	8
8	8	8	8	8	8	8	8

A Typical Compression System



Coding Quantized DCT coefficient

- **DC coefficient**: Predictive coding
  - The DC value of the current block is predicted from that of the **previous block**, and the error is coded using Huffman coding
- **AC Coefficients**: Run length coding
  - Many high frequency AC coefficients are zero after first few low-frequency coefficients
  - Runlength Representation:
    - Ordering coefficients in the **zig-zag** order
    - Specify how many zeros before a non-zero value
    - Each **symbol=(length-of-zero, non-zero-value)**
  - Code all possible symbols using **Huffman coding**
    - More frequently appearing symbols are given shorter codewords
  - One can use default Huffman tables or specify its own tables

An Illustrative Example

qdct =

DC

2	5	0	-2	0	-1	0	0
9	1	-1	2	0	1	0	0
14	1	-1	0	-1	0	0	0
3	-1	-1	-1	0	0	0	0
2	-1	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

Zig-zag + run length encoding

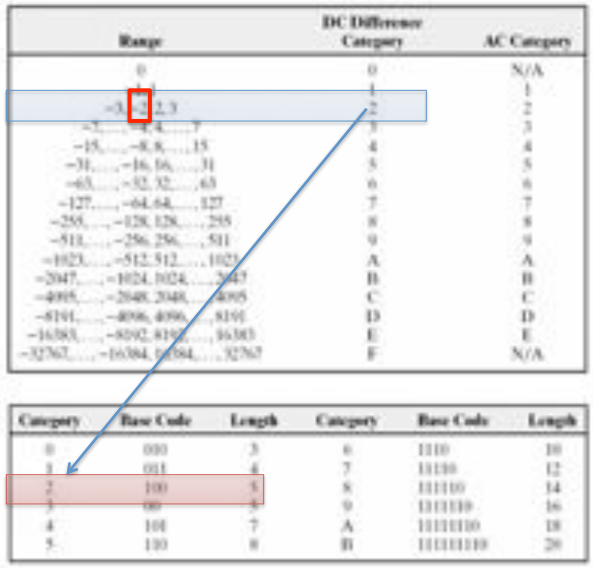
DC {2,(0,5),(0,9),(0,14),(0,1),(1,-2),(0,-1),(0,1),(0,3),(0,2),(0,-1),(0,-1),(0,2),(1,-1),(2,-1), (0,-1), (4,-1),(0,-1),(0,1),EOB}

EOB: End of block, one of the symbol that is assigned a short Huffman codeword

Coding DC component

- Current quantized DC index: 2
- Previous block DC index: 4
- Prediction error: -2
  - The prediction error is coded in two parts:
    - **Which category it belongs to** (in the Table of JPEG Coefficient Coding Categories), and code using a Huffman code (JPEG Default DC Code)
      - DC= -2 is in category "2", with a codeword "**100**"
    - **Which position it is in that category**, using a fixed length code, length=category number
      - "-2" in category 2, has a fixed length code of "**10**".
  - The overall codeword is "**10010**"

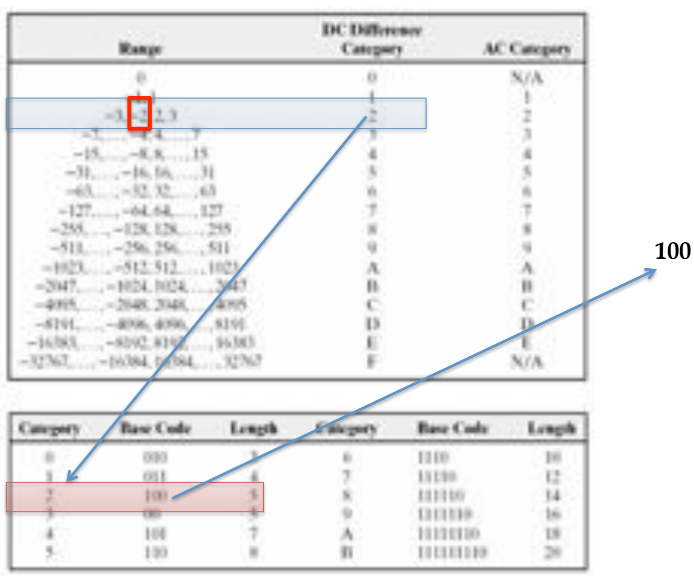
JPEG Tables for Coding DC



Range	DC Difference Category	AC Category
0	0	N/A
−1, 1	1	1
−3, −2, 2, 3	2	2
−7, ..., −4, 4, ..., 7	3	3
−15, ..., −8, 8, ..., 15	4	4
−31, ..., −16, 16, ..., 31	5	5
−63, ..., −32, 32, ..., 63	6	6
−127, ..., −64, 64, ..., 127	7	7
−255, ..., −128, 128, ..., 255	8	8
−511, ..., −256, 256, ..., 511	9	9
−1023, ..., −512, 512, ..., 1023	A	A
−2047, ..., −1024, 1024, ..., 2047	B	B
−4095, ..., −2048, 2048, ..., 4095	C	C
−8191, ..., −4096, 4096, ..., 8191	D	D
−16383, ..., −8192, 8192, ..., 16383	E	E
−32767, ..., −16384, 16384, ..., 32767	F	N/A

Category	Base Code	Length	Category	Base Code	Length
0	010	3	6	1110	10
1	011	4	7	11110	12
2	100	5	8	111110	14
3	00	5	9	1111110	16
4	101	7	A	11111110	18
5	110	8	B	111111110	20

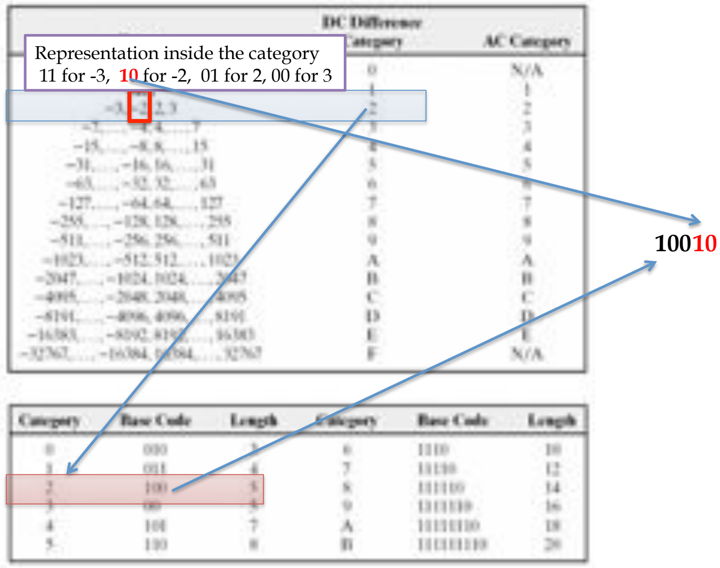
JPEG Tables for Coding DC



Range	DC Difference Category	AC Category
0	0	N/A
−1, 1	1	1
−3, −2, 2, 3	2	2
−7, ..., −4, 4, ..., 7	3	3
−15, ..., −8, 8, ..., 15	4	4
−31, ..., −16, 16, ..., 31	5	5
−63, ..., −32, 32, ..., 63	6	6
−127, ..., −64, 64, ..., 127	7	7
−255, ..., −128, 128, ..., 255	8	8
−511, ..., −256, 256, ..., 511	9	9
−1023, ..., −512, 512, ..., 1023	A	A
−2047, ..., −1024, 1024, ..., 2047	B	B
−4095, ..., −2048, 2048, ..., 4095	C	C
−8191, ..., −4096, 4096, ..., 8191	D	D
−16383, ..., −8192, 8192, ..., 16383	E	E
−32767, ..., −16384, 16384, ..., 32767	F	N/A

Category	Base Code	Length	Category	Base Code	Length
0	010	3	6	1110	10
1	011	4	7	11110	12
2	100	5	8	111110	14
3	00	5	9	1111110	16
4	101	7	A	11111110	18
5	110	8	B	111111110	20

JPEG Tables for Coding DC

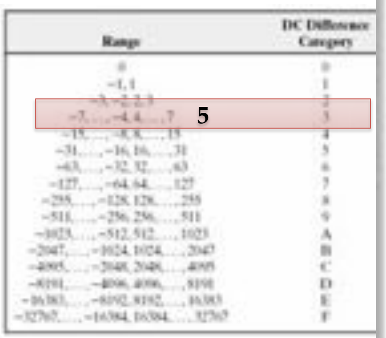


Coding AC components

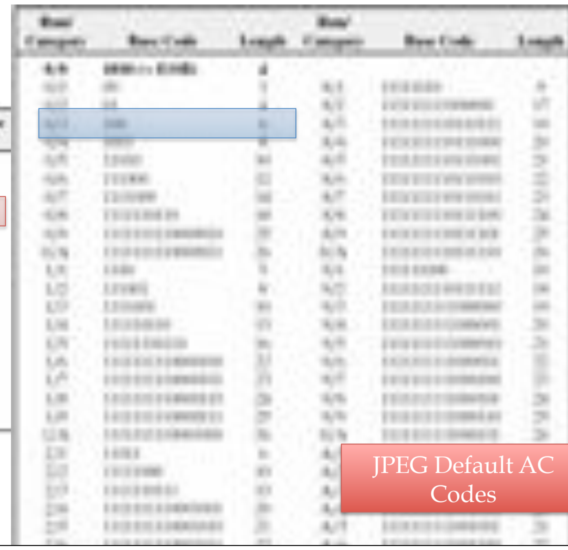
- First symbol (0,5) is represented in two parts:
  - Which category it belongs to (Table of JPEG Coefficient Coding Categories), and code the "(**runlength**, **category**)" using a Huffman code (JPEG Default AC Code)
    - AC=5 is in category "**3**",
    - Symbol (**0**,**3**) has codeword "**100**"
  - Which position it is in that category, using a fixed length code, length=category number
    - "5" is the number 5 (starting from 0) in category 3, with a fixed length code of "**101**"
    - – The overall codeword for (0,5) is "**100101**"
- Second symbol (0,9)
  - '9' in category '4', (0,4) has codeword '**1011**'
  - '9' is number 9 in category 4 with codeword '**1001**'
  - overall codeword for (0,9) is '**10111001**'

(runlength, category) Coding Table for AC symbol (0,5)

JPEG Coefficient Coding Categories



Range	DC Difference Category
0	0
−1, 1	1
−3, −2, 2, 3	2
−7, …, −4, 4, …, 7	3
−15, …, −8, 8, …, 15	4
−31, …, −16, 16, …, 31	5
−63, …, −32, 32, …, 63	6
−127, …, −64, 64, …, 127	7
−255, …, −128, 128, …, 255	8
−511, …, −256, 256, …, 511	9
−1023, …, −512, 512, …, 1023	A
−2047, …, −1024, 1024, …, 2047	B
−4095, …, −2048, 2048, …, 4095	C
−8191, …, −4096, 4096, …, 8191	D
−16383, …, −8192, 8192, …, 16383	E
−32767, …, −16384, 16384, …, 32767	F



JPEG Default AC Codes

Recap: A Typical Image Compression System

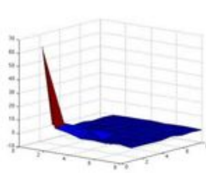
Transform original data into a new representation that is easier to compress

Use a limited number of levels to represent the signal values

Find an efficient way to represent these levels using binary bits

Transformation → Quantization → Binary Encoding

DCT for images +Zigzag ordering



Scalar 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

(Run-length coding Huffman coding…)

DC: prediction + Huffman
AC: run-length + Huffman

JPEG

- The Joint Photographic Expert Group (JPEG), under both the International Standards Organization (ISO) and the International Telecommunications Union Telecommunication Sector (ITU-T)
  - www.jpeg.org
- Has published several standards
  - JPEG: lossy coding of continuous tone still images
    - Based on DCT
  - JPEG-LS: lossless and near lossless coding of continuous tone still images
    - Based on predictive coding and entropy coding
  - JPEG2000: scalable coding of continuous tone still images (from lossy to lossless)
    - Based on wavelet transform

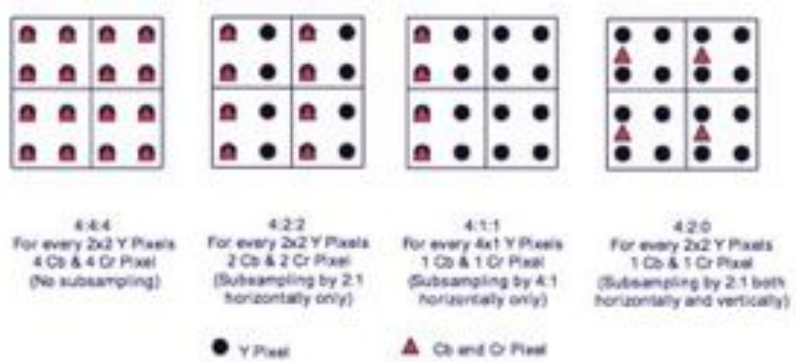
1992 JPEG

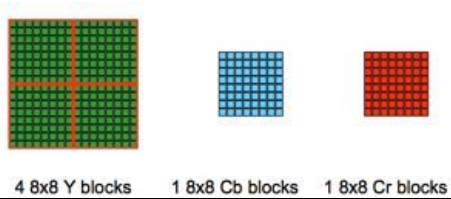
- Support several modes
  - Baseline system (what is commonly known as JPEG!): lossy
    - Can handle grayscale or colorimages, with 8bits per color component
  - Extended system:
    - can handle higher precision (12 bit) images, providing progressive streams, etc.
  - Lossless version
- Baseline version
  - Each color component is divided into 8x8 blocks
  - For each 8x8 block, three steps are involved:
    - **Block DCT**
    - **Perceptual-based quantization**
    - **Variable length coding: Run length and Huffman coding**

Coding Colored Images

- Color images are typically stored in (R,G,B) format
  - JPEG standard can be applied to each component separately
    - Does not make use of the correlation between color components
    - Does not make use of the lower sensitivity of the human eye to chrominance samples
- Alternate approach
  - Convert (R,G,B) representation to a YCbCr representation
    - Y: luminance, Cb, Cr: chrominance
    - Down-sample the two chrominance components
      - Because the peak response of the eye to the luminance component occurs at a higher frequency than to the chrominance components

Chrominance Subsampling





Quantization Tables for Y, Cr Cb

For luminance

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

For chrominance

17	18	24	47	99	99	99	99
18	21	26	66	99	99	99	99
24	26	56	99	99	99	99	99
47	66	99	99	99	99	99	99
99	99	99	99	99	99	99	99
99	99	99	99	99	99	99	99
99	99	99	99	99	99	99	99
99	99	99	99	99	99	99	99

Summary

- The concept behind compression and transformation
- How to perform 2D DCT: forward and inverse transform
  - Manual calculation for small sizes, using inner product notation
  - Using Matlab: dct2, idct2
- Why DCT is good for image coding
  - Real transform, easier than DFT
  - Most high frequency coefficients are nearly zero and can be ignored
  - Different coefficients can be quantized with different accuracy based on human sensitivity
- How to quantize & code DCT coefficients
  - Varying step sizes for different DCT coefficients based on visual sensitivity to different frequencies; A quantization matrix specifies the default quantization stepsize for each coefficient; The matrix can be scaled using a user chosen parameter (QP) to obtain different trade-offs between quality and size
  - DC: prediction + huffman; AC: run-length + huffman

Next Lecture

- Speech/Audio/Video Compression!