EL5123 --- Image Processing

Lossy Image Compression and JPEG standard

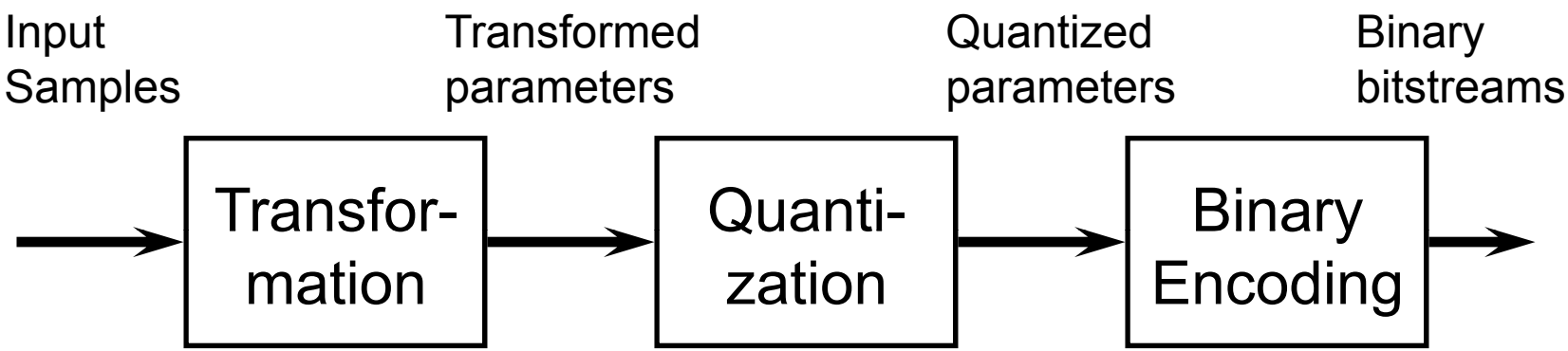
Yao Wang
Polytechnic Institute of NYU, Brooklyn, NY 11201

With contribution from Zhu Liu, and
Gonzalez/Woods, Digital Image Processing, 2ed and A. K. Jain, Fundamentals of Digital Image Processing

Lecture Outline

- Introduction
- Quantization Revisited
  - General description of quantizer
  - Uniform quantization
- Transform coding
  - Review of linear and unitary transform, 1D and 2D
  - DCT
  - Quantization of transform coefficients
  - Runlength coding of DCT coefficients
- JPEG standard overview

A Typical Compression System



Prediction
Transforms
Model fitting
……

Scalar Q
Vector Q

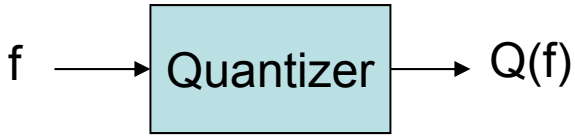
Fixed length
Variable length (Huffman, arithmetic, LZW)

- Motivation for transformation ---
  To yield a more efficient representation of the original samples.

Quantization (Review)

- General description
- Uniform quantizer

General Description of Quantization



Decision Levels $\{t_k, k = 0, \ldots, L\}$

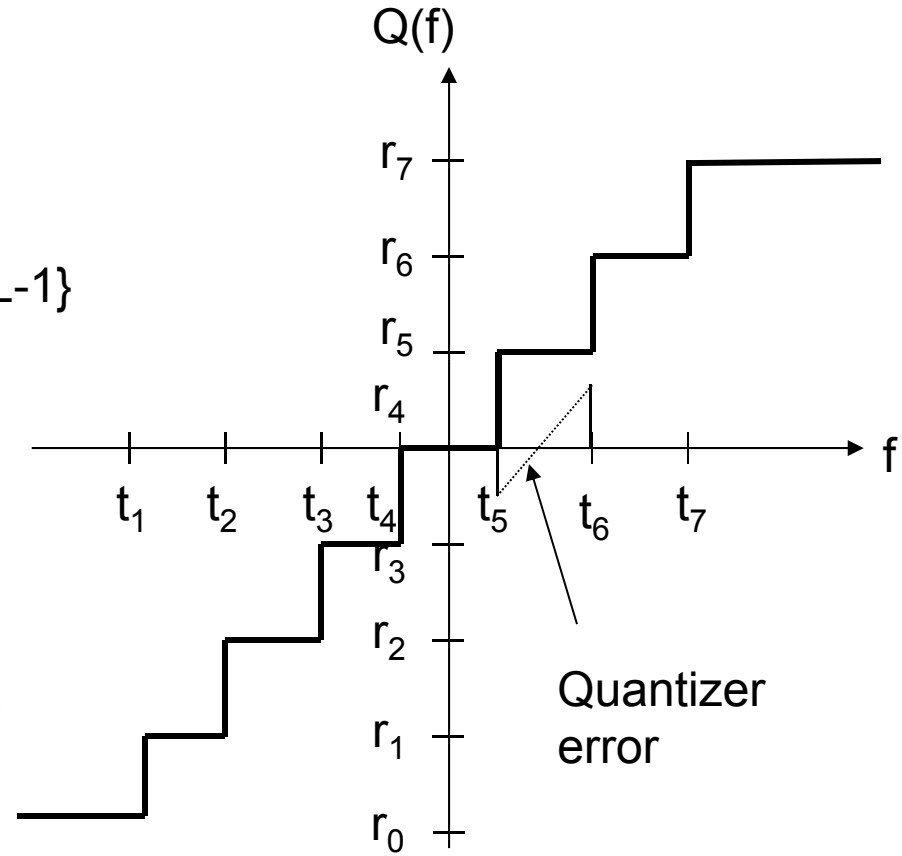
Reconstruction Levels $\{r_k, k = 0, \ldots, L-1\}$

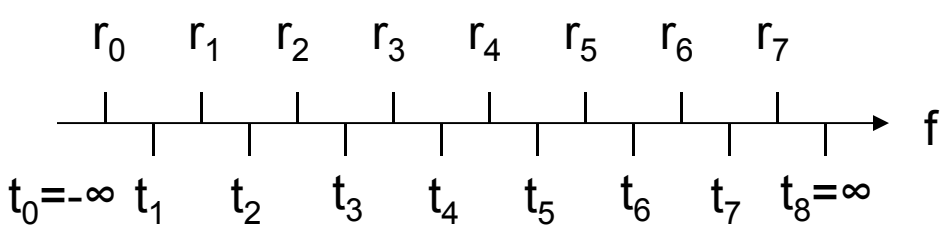
If $f \in [t_k, t_{k+1})$

Then $Q(f) = r_k$

L levels need $R = \lceil \log_2 L \rceil$ bits

$\lceil x \rceil$ returns the smallest integer that is bigger than or equal to x





Uniform Quantization

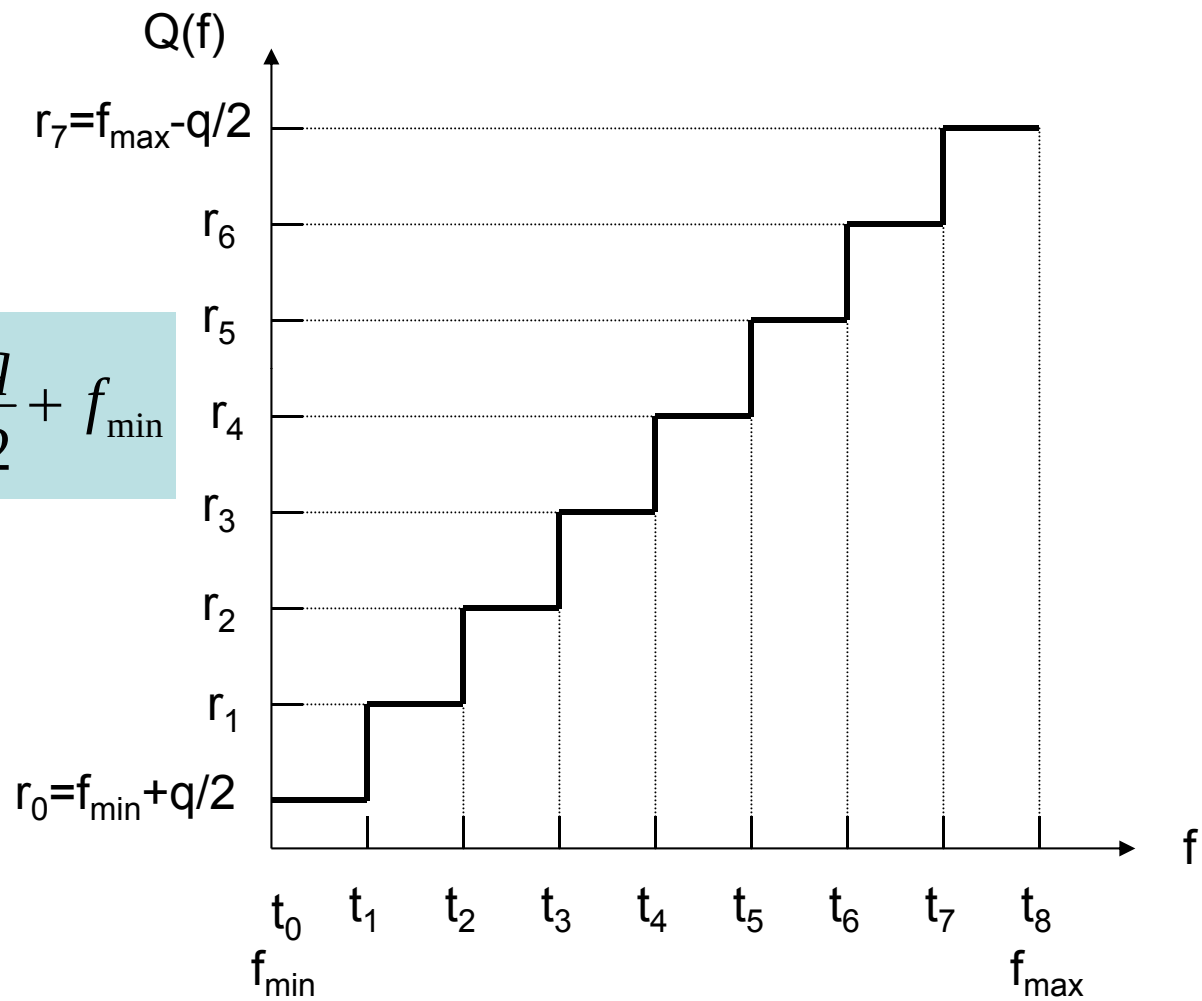
- Equal distances between adjacent decision levels and between adjacent reconstruction levels
  - $t_l - t_{l-1} = r_l - r_{l-1} = q$
- Parameters of Uniform Quantization
  - L: levels ($L = 2^R$)
  - B: dynamic range $B = f_{max} - f_{min}$
  - q: quantization interval (step size)
  - $q = B/L = B2^{-R}$

Uniform Quantization: Functional Representation

stepsize $q=(f_{max}-f_{min})/L$

$$Q(f)=\left\lfloor \frac{f-f_{\min}}{q} \right\rfloor * q+\frac{q}{2}+f_{\min}$$

$\lfloor x \rfloor$ returns the biggest integer that is smaller than or equal to x



$I(f)=\left\lfloor \frac{f-f_{\min}}{q} \right\rfloor$ is called the reconstruction level index, which indicates which reconstruction level is used for f.

Uniform Quantization on Images

Original, L=256

q=8, L=32

q=16, L=16

q=64, L=4

Implementation

- Setup the quantization function Q(f) for all possible input level into a look-up table first. Actual quantization can be implemented as a table look-up.

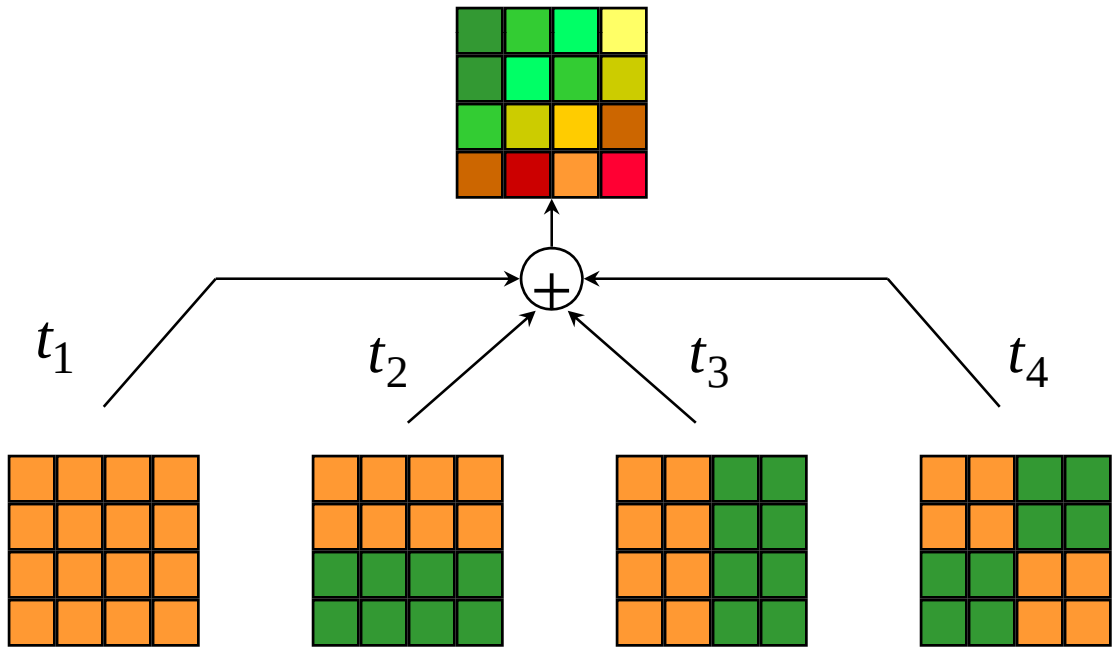
L = 16, MSE = 21.6

Matlab code:

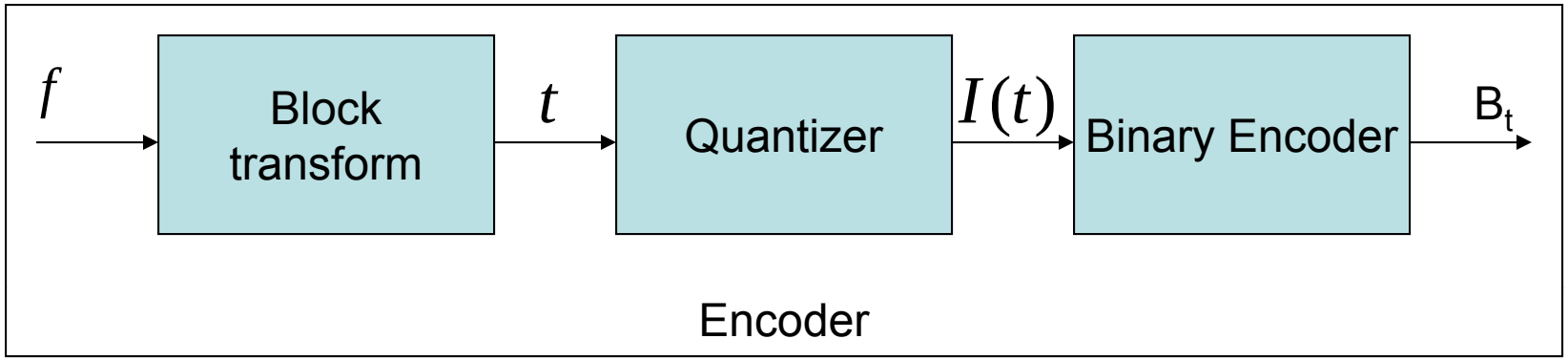
```
x = imread('lena256_8.bmp');
[height, width] = size(x);
B = 256;
x = double(x);
% This is the quantization table
Q = zeros(256, 1);
% Quantized to 64 levels
L = 16;
q = B / L;
for i = 0:255,
    Q(i+1, 1) = floor(i / q) * q + q /2;
end
y = zeros(size(x));
for i = 1:height,
    for j = 1:width,
        y(i, j) = Q(x(i,j) + 1);
    end
End
MSE = mean(mean((x-y).^2));
```

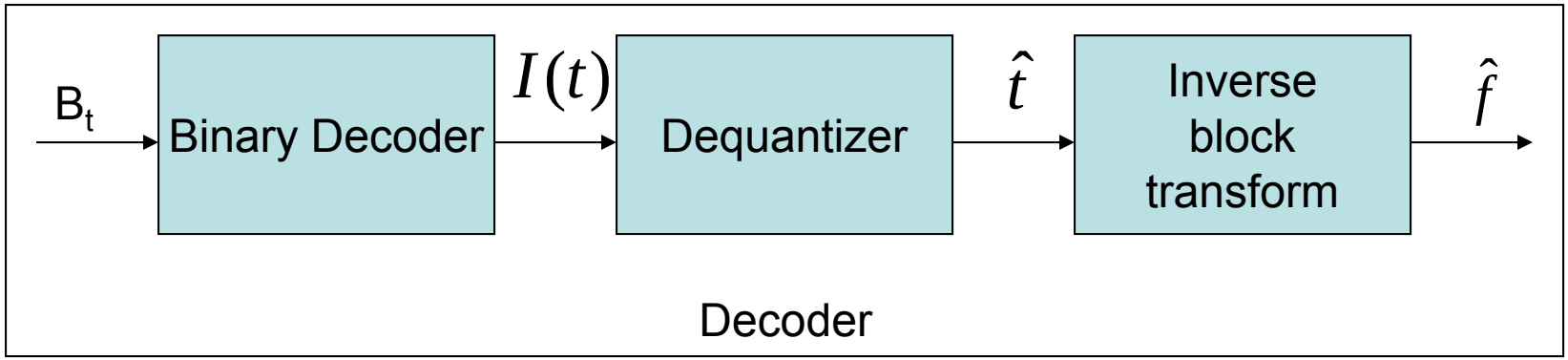
Transform Coding

- Represent an image (or an image block) as the linear combination of some basis images and specify the linear coefficients.



Components in Transform Coding





For block transform coders, we usually use *Unitary* (*othonormal*) transforms

Why Using Transform?

- When the transform basis is chosen properly
  - Many coefficients have small values and can be quantized to 0 w/o causing noticeable artifacts
  - The coefficients are uncorrelated, and hence can be coded independently w/o losing efficiency.

What Transform Basis to Use?

- The transform should
  - Minimize the correlation among resulting coefficients, so that scalar quantization can be employed without losing too much in coding efficiency compared to vector quantization
  - Compact the energy into as few coefficients as possible
- Optimal transform
  - Karhunan Loeve Transform (KLT): signal statistics dependent
- Suboptimal transform
  - Discrete Cosine transform (DCT): nearly as good as KLT for common image signals

One Dimensional Linear Transform

- Let C^N represent the N dimensional complex space.
- Let $\mathbf{h}_0$, $\mathbf{h}_1$, …, $\mathbf{h}_{N-1}$ represent N linearly independent vectors in C^N.
- Any vector $\mathbf{f} \in C^N$ can be represented as a linear combination of $\mathbf{h}_0$, $\mathbf{h}_1$, …, $\mathbf{h}_{N-1}$:

$$\mathbf{f} = \sum_{k=0}^{N-1} t(k)\mathbf{h}_k = \mathbf{B}\mathbf{t},$$

$$where \quad \mathbf{B} = [\mathbf{h}_0, \mathbf{h}_1, \ldots, \mathbf{h}_{N-1}], \mathbf{t} = \begin{bmatrix} t(0) \\ t(1) \\ \vdots \\ t(N-1) \end{bmatrix}.$$

$$\Rightarrow \quad \mathbf{t} = \mathbf{B}^{-1}\mathbf{f} = \mathbf{A}\mathbf{f}$$

f and t form a transform pair

Inner Product

- Definition of inner product

$$<\mathbf{f}_1,\mathbf{f}_2>=\mathbf{f}_1^H\mathbf{f}_2=\sum_{n=0}^{N-1} f_1^*(n)\,f_2(n)$$

- Orthogonal

$$<\mathbf{f}_1,\mathbf{f}_2>=0$$

- Norm of a vector

$$\|\mathbf{f}\|^2=<\mathbf{f},\mathbf{f}>=\mathbf{f}^H\mathbf{f}=\sum_{n=0}^{N-1}|f(n)|^2$$

- Normalized vector: unit norm $\|\mathbf{f}\|^2=1$
- Orthonomal = orthogonal + normalized

Orthonormal Basis Vectors (OBV)

- $\{\mathbf{h}_k, k=0,\ldots N-1\}$ are OBV if $<\mathbf{h}_k,\mathbf{h}_l>=\delta_{k,l}=\begin{cases}1 & k=l\\ 0 & k\neq l\end{cases}$

- With OBV

$$<\mathbf{h}_l,\mathbf{f}>=<\mathbf{h}_l,\sum_{k=0}^{N-1}t(k)\mathbf{h}_k>=\sum_{k=0}^{N-1}t(k)<\mathbf{h}_l,\mathbf{h}_k>=t(l)=\mathbf{h}_l^H\mathbf{f}$$

$$\mathbf{t}=\begin{bmatrix}\mathbf{h}_0^H\\ \mathbf{h}_1^H\\ \vdots\\ \mathbf{h}_{N-1}^H\end{bmatrix}\mathbf{f}=\mathbf{B}^H\mathbf{f}=\mathbf{A}\mathbf{f}$$

$$\mathbf{B}^{-1}=\mathbf{B}^H,\ or\ \mathbf{B}^H\mathbf{B}=\mathbf{B}\mathbf{B}^H=\mathbf{I}.$$

B is unitary

Definition of Unitary Transform

- Basis vectors are orthonormal
- Forward transform

$$t(k) = <\mathbf{h}_k, \mathbf{f}> = \sum_{n=0}^{N-1} h_k(n)^* f(n),$$

$$\mathbf{t} = \begin{bmatrix} \mathbf{h}_0^H \\ \mathbf{h}_1^H \\ \vdots \\ \mathbf{h}_{N-1}^H \end{bmatrix} \mathbf{f} = \mathbf{B}^H \mathbf{f} = \mathbf{A}\mathbf{f}$$

- Inverse transform

$$f(n) = \sum_{k=0}^{N-1} t(k) h_k(n),$$

$$\mathbf{f} = \sum_{k=0}^{N-1} t(k) \mathbf{h}_k = \begin{bmatrix} \mathbf{h}_0 & \mathbf{h}_1 & \cdots & \mathbf{h}_{N-1} \end{bmatrix} \mathbf{t} = \mathbf{B}\mathbf{t} = \mathbf{A}^H \mathbf{t}$$

Example of 1D Unitary Transform

$$\mathbf{h}_0 = \begin{bmatrix} 1/2 \\ 1/2 \\ 1/2 \\ 1/2 \end{bmatrix}, \mathbf{h}_1 = \begin{bmatrix} 1/2 \\ 1/2 \\ -1/2 \\ -1/2 \end{bmatrix}, \mathbf{h}_2 = \begin{bmatrix} 1/2 \\ -1/2 \\ -1/2 \\ 1/2 \end{bmatrix}, \mathbf{h}_3 = \begin{bmatrix} 1/2 \\ -1/2 \\ 1/2 \\ -1/2 \end{bmatrix},$$

$$\mathbf{f} = \begin{bmatrix} 1 \\ 2 \\ 3 \\ 4 \end{bmatrix} \quad \Rightarrow \quad \begin{cases} t_0 = 5 \\ t_1 = -2 \\ t_2 = 0 \\ t_3 = -1 \end{cases}$$

1D DFT as a Unitary Transform

$$F(k) = \frac{1}{\sqrt{N}} \sum_{n=0}^{N-1} f(n) e^{-j2\pi \frac{kn}{N}}, \quad k = 0, 1, \ldots, N-1;$$

$$f(n) = \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} F(k) e^{j2\pi \frac{kn}{N}}, \quad n = 0, 1, \ldots, N-1.$$

$$h_k(n) = \frac{1}{\sqrt{N}} e^{j2\pi \frac{kn}{N}}, \quad or$$

$$\mathbf{h}_k = \frac{1}{\sqrt{N}} \begin{bmatrix} 1 \\ e^{j2\pi \frac{k}{N}} \\ \vdots \\ e^{j2\pi \frac{(N-1)k}{N}} \end{bmatrix}, k = 0,1,\ldots, N-1.$$

Example: 1D DFT, N=2

$N = 2$ case : there are only two basis vectors :

$$\mathbf{h}_k = \frac{1}{\sqrt{2}}\begin{bmatrix} \exp(j2\pi\frac{k}{2}0) \\ \exp(j2\pi\frac{k}{2}1) \end{bmatrix}: \quad \mathbf{h}_0 = \frac{1}{\sqrt{2}}\begin{bmatrix} 1 \\ 1 \end{bmatrix}, \mathbf{h}_1 = \frac{1}{\sqrt{2}}\begin{bmatrix} 1 \\ -1 \end{bmatrix}$$

if $\mathbf{f} = \begin{bmatrix} 1 \\ 2 \end{bmatrix}$, determine t_0, t_1

Using $t_k = <\mathbf{h}_k, \mathbf{f}>$, we obtain

$$t_0 = \frac{1}{\sqrt{2}}\left(\begin{bmatrix} 1 \\ 1 \end{bmatrix}, \begin{bmatrix} 1 \\ 2 \end{bmatrix}\right) = \frac{1}{\sqrt{2}}(1*1+1*2) = \frac{3}{\sqrt{2}}, t_1 = \frac{1}{\sqrt{2}}\left(\begin{bmatrix} 1 \\ -1 \end{bmatrix}, \begin{bmatrix} 1 \\ 2 \end{bmatrix}\right) = \frac{1}{\sqrt{2}}(1*1-1*2) = \frac{-1}{\sqrt{2}}$$

$$\text{Verify}: t_0\mathbf{h}_0 + t_1\mathbf{h}_1 = \frac{3}{2}\begin{bmatrix} 1 \\ 1 \end{bmatrix} - \frac{1}{2}\begin{bmatrix} 1 \\ -1 \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \end{bmatrix} = \mathbf{f}$$

Another Example: 1D DFT, N=4

$$N=4\,\text{case}:\ \text{using}\ \mathbf{h}_k=\frac{1}{2}\begin{bmatrix}\exp(j2\pi\frac{k}{4}0)\\ \exp(j2\pi\frac{k}{4}1)\\ \exp(j2\pi\frac{k}{4}2)\\ \exp(j2\pi\frac{k}{4}3)\end{bmatrix}\ \text{yields}:\quad \mathbf{h}_0=\frac{1}{2}\begin{bmatrix}1\\1\\1\\1\end{bmatrix};\mathbf{h}_1=\frac{1}{2}\begin{bmatrix}1\\j\\-1\\-j\end{bmatrix};\mathbf{h}_2=\frac{1}{2}\begin{bmatrix}1\\-1\\1\\-1\end{bmatrix};\mathbf{h}_3=\frac{1}{2}\begin{bmatrix}1\\-j\\-1\\j\end{bmatrix}$$

$$\mathbf{f}=\begin{bmatrix}2\\4\\5\\3\end{bmatrix}\quad\Rightarrow\quad \begin{aligned}&t_0=\frac{1}{2}(2+4+5+3)=7;\quad t_1=\frac{1}{2}(2-4j-5+3j)=-\frac{1}{2}(3+j);\\ &t_2=\frac{1}{2}(2-4+5-3)=0;\quad t_3=\frac{1}{2}(2+4j-5-3j)=-\frac{1}{2}(3-j).\end{aligned}$$

$$\text{Verify}:\ t_0\mathbf{h}_0+t_1\mathbf{h}_1+t_2\mathbf{h}_2+t_3\mathbf{h}_3=\frac{1}{4}\begin{bmatrix}14-(3+j)-(3-j)\\14-(3+j)j+(3-j)j\\14+(3+j)+(3-j)\\14+(3+j)j-(3-j)j\end{bmatrix}=\frac{1}{4}\begin{bmatrix}8\\16\\20\\12\end{bmatrix}=\mathbf{f}$$

1D Discrete Cosine Transform (DCT)

Basis Vectors:

$$h_k(n) = \alpha(k)\cos\left[\frac{(2n+1)k\pi}{2N}\right]$$

$$\text{where } \alpha(k) = \begin{cases} \sqrt{1/N} & k = 0 \\ \sqrt{2/N} & k = 1,\ldots,N-1 \end{cases}$$

$$\text{Forward Transform: } T(k) = \sum_{n=0}^{N-1} f(n)h_k(n)$$

$$\text{Inverse Transforms: } f(n) = \sum_{u=0}^{N-1} T(k)h_k(n)$$

Vector Representation $N = 4$ case:

$$\mathbf{h}_k = \alpha(k)\begin{bmatrix} \cos\left(\frac{1}{8}k\pi\right) \\ \cos\left(\frac{3}{8}k\pi\right) \\ \cos\left(\frac{5}{8}k\pi\right) \\ \cos\left(\frac{7}{8}k\pi\right) \end{bmatrix} \text{ yields: } \mathbf{h}_0 = \frac{1}{2}\begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 0.5 \\ 0.5 \\ 0.5 \\ 0.5 \end{bmatrix}; \mathbf{h}_1 = \sqrt{\frac{1}{2}}\begin{bmatrix} \cos\left(\frac{1}{8}\pi\right) \\ \cos\left(\frac{3}{8}\pi\right) \\ \cos\left(\frac{5}{8}\pi\right) \\ \cos\left(\frac{7}{8}\pi\right) \end{bmatrix} = \begin{bmatrix} 0.6533 \\ 0.2706 \\ -0.2706 \\ -0.6533 \end{bmatrix}; \mathbf{h}_2 = \sqrt{\frac{1}{2}}\begin{bmatrix} \cos\left(\frac{2}{8}\pi\right) \\ \cos\left(\frac{6}{8}\pi\right) \\ \cos\left(\frac{10}{8}\pi\right) \\ \cos\left(\frac{14}{8}\pi\right) \end{bmatrix} = \begin{bmatrix} 0.5 \\ -0.5 \\ -0.5 \\ 0.5 \end{bmatrix}; \mathbf{h}_3 = \sqrt{\frac{1}{2}}\begin{bmatrix} \cos\left(\frac{3}{8}\pi\right) \\ \cos\left(\frac{9}{8}\pi\right) \\ \cos\left(\frac{15}{8}\pi\right) \\ \cos\left(\frac{21}{8}\pi\right) \end{bmatrix} = \begin{bmatrix} 0.2706 \\ -0.6533 \\ 0.6533 \\ -0.2706 \end{bmatrix}$$

$$\mathbf{f} = \begin{bmatrix} 2 \\ 4 \\ 5 \\ 3 \end{bmatrix} \Rightarrow \begin{array}{ll} t_0 = \frac{1}{2}(2+4+5+3) = 7; & t_1 = (2-3)*0.6533 + (4-5)*0.2706 = -0.9239; \\ t_2 = \frac{1}{2}(2-4-5+3) = -2; & t_3 = (2-3)*0.2706 + (5-4)*0.6533 = 0.3827. \end{array}$$

Verify: $t_0\mathbf{h}_0 + t_1\mathbf{h}_1 + t_2\mathbf{h}_2 + t_3\mathbf{h}_3 = \mathbf{f}$

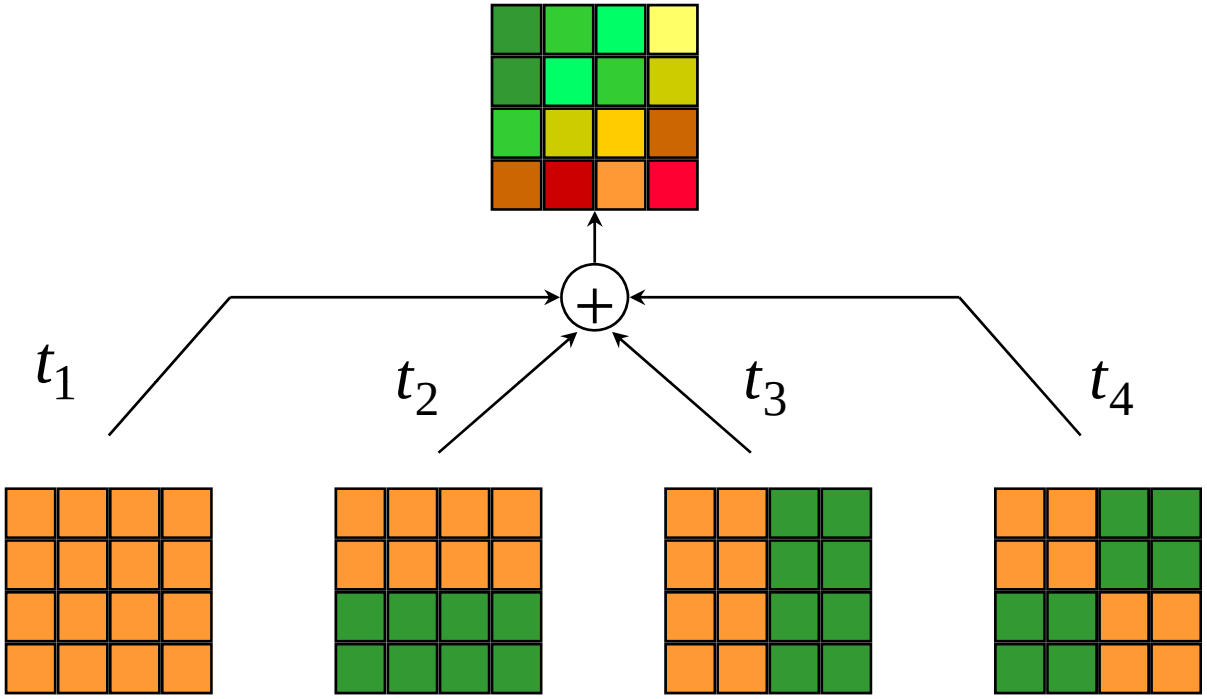
Two Dimensional Decomposition

- Decompose a MxN 2D matrix F=[F(m,n)] into a linear combination of some basic images, $H_{k,l}$=[$H_{k,l}$(m,n)], so that:

$$\mathbf{F} = \sum_{k=0}^{M-1}\sum_{l=0}^{N-1} T(k,l)\mathbf{H}_{k,l},$$

$$F(m,n) = \sum_{k=0}^{M-1}\sum_{l=0}^{N-1} T(k,l)H_{k,l}(m,n)$$

Graphical Interpretation



Inverse transform: Represent a vector (e.g. a block of image samples) as the superposition of some basis vectors (block patterns)
Forward transform: Determine the coefficients associated with each basis vector

Two Dimensional Inner Product

- Inner Product

$$<\mathbf{F}_1,\mathbf{F}_2>=\sum_{m=0}^{M-1}\sum_{n=0}^{N-1}F_1^*(m,n)F_2(m,n)$$

- Norm of a Matrix

$$\|\mathbf{F}\|=<\mathbf{F},\mathbf{F}>=\sum_{m=0}^{M-1}\sum_{n=0}^{N-1}|F(m,n)|^2$$

- A set of basis images {$\mathbf{H}_{k,l}$, k=0,1,…,M-1, l=0,1,…,N-1} is orthonormal if

$$<\mathbf{H}_{k,l},\mathbf{H}_{i,j}>=\delta_{k,i}\delta_{l,j}=\begin{cases}1, & if\ k=i,l=j\\ 0, & otherwise\end{cases}$$

Two Dimensional Unitary Transform

- $\{H_{k,l}\}$ is an orthonormal set of basis images
- Forward transform

$$T(k,l) = <\mathbf{H}_{k,l}, \mathbf{F}> = \sum_{m=0}^{M-1}\sum_{n=0}^{N-1} H^*_{k,l}(m,n)F(m,n)$$

- Inverse transform

$$F(m,n) = \sum_{k=0}^{M-1}\sum_{l=0}^{N-1} T(k,l)H_{k,l}(m,n), \quad or$$

$$\mathbf{F} = \sum_{k=0}^{M-1}\sum_{l=0}^{N-1} T(k,l)\mathbf{H}_{k,l}$$

Example of 2D Unitary Transform

$$\mathbf{H}_{00} = \begin{bmatrix} 1/2 & 1/2 \\ 1/2 & 1/2 \end{bmatrix}, \mathbf{H}_{01} = \begin{bmatrix} 1/2 & 1/2 \\ -1/2 & -1/2 \end{bmatrix}, \mathbf{H}_{10} = \begin{bmatrix} 1/2 & -1/2 \\ 1/2 & -1/2 \end{bmatrix}, \mathbf{H}_{11} = \begin{bmatrix} 1/2 & -1/2 \\ -1/2 & 1/2 \end{bmatrix},$$

$$\mathbf{F} = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \quad \Rightarrow \quad \begin{cases} T(0,0) = 5 \\ T(0,1) = -2 \\ T(1,0) = -1 \\ T(1,1) = 0 \end{cases}$$

Separable Unitary Transform

- Let $\mathbf{h}_k$, k=0, 1, …, M-1 represent orthonormal basis vectors in C^M,
- Let $\mathbf{g}_l$, l=0, 1, …, N-1 represent orthonormal basis vectors in C^N,
- $\mathbf{H}_{k,l}=\mathbf{h}_k\mathbf{g}_l^T$, or $H_{k,l}(m,n)=h_k(m)g_l(n)$.
- Then $\mathbf{H}_{k,l}$ will form an orthonormal basis set in C^{MxN}.

Example of Separable Unitary Transform

- Example 1

$$\mathbf{h}_0 = \begin{bmatrix} 1/\sqrt{2} \\ 1/\sqrt{2} \end{bmatrix}, \mathbf{h}_1 = \begin{bmatrix} 1/\sqrt{2} \\ -1/\sqrt{2} \end{bmatrix}.$$

$$\mathbf{H}_{00} = \mathbf{h}_0\mathbf{h}_0^T = \begin{bmatrix} 1/2 & 1/2 \\ 1/2 & 1/2 \end{bmatrix} \qquad \mathbf{H}_{01} = \mathbf{h}_0\mathbf{h}_1^T = \begin{bmatrix} 1/2 & -1/2 \\ 1/2 & -1/2 \end{bmatrix}$$

$$\mathbf{H}_{10} = \mathbf{h}_1\mathbf{h}_0^T = \begin{bmatrix} 1/2 & 1/2 \\ -1/2 & -1/2 \end{bmatrix} \qquad \mathbf{H}_{11} = \mathbf{h}_1\mathbf{h}_1^T = \begin{bmatrix} 1/2 & -1/2 \\ -1/2 & 1/2 \end{bmatrix}$$

Example: 4x4 DFT

Recall the 1D DFT basis are: $\mathbf{h}_0 = \frac{1}{2}\begin{bmatrix}1\\1\\1\\1\end{bmatrix}; \mathbf{h}_1 = \frac{1}{2}\begin{bmatrix}1\\j\\-1\\-j\end{bmatrix}; \mathbf{h}_2 = \frac{1}{2}\begin{bmatrix}1\\-1\\1\\-1\end{bmatrix}; \mathbf{h}_3 = \frac{1}{2}\begin{bmatrix}1\\-j\\-1\\j\end{bmatrix}$

using $\mathbf{H}_{k,l} = \mathbf{h}_k (\mathbf{h}_l)^T$ yields:

$$\mathbf{H}_{0,0} = \frac{1}{4}\begin{bmatrix}1&1&1&1\\1&1&1&1\\1&1&1&1\\1&1&1&1\end{bmatrix}, \quad \mathbf{H}_{0,1} = \frac{1}{4}\begin{bmatrix}1&j&-1&-j\\1&j&-1&-j\\1&j&-1&-j\\1&j&-1&-j\end{bmatrix}, \quad \mathbf{H}_{0,2} = \frac{1}{4}\begin{bmatrix}1&-1&1&-1\\1&-1&1&-1\\1&-1&1&-1\\1&-1&1&-1\end{bmatrix}, \quad \mathbf{H}_{0,3} = \frac{1}{4}\begin{bmatrix}1&-j&-1&j\\1&-j&-1&j\\1&-j&-1&j\\1&-j&-1&j\end{bmatrix}$$

$$\mathbf{H}_{1,0} = \frac{1}{4}\begin{bmatrix}1&1&1&1\\j&j&j&j\\-1&-1&-1&-1\\-j&-j&-j&j\end{bmatrix}, \mathbf{H}_{1,1} = \frac{1}{4}\begin{bmatrix}1&j&-1&-j\\j&-1&-j&1\\-1&-j&1&j\\-j&1&j&-1\end{bmatrix}, \mathbf{H}_{1,2} = \frac{1}{4}\begin{bmatrix}1&-1&1&-1\\j&-j&j&-j\\-1&1&-1&1\\-j&j&-j&j\end{bmatrix}, \mathbf{H}_{1,3} = \frac{1}{4}\begin{bmatrix}1&-j&-1&j\\j&1&-j&-1\\-1&j&1&-j\\-j&-1&j&1\end{bmatrix}$$

$$\mathrm{H}_{2,0} = \frac{1}{4}\begin{bmatrix}1&1&1&1\\-1&-1&-1&-1\\1&1&1&1\\-1&-1&-1&-1\end{bmatrix}, \mathbf{H}_{2,1} = \frac{1}{4}\begin{bmatrix}1&j&-1&-j\\-1&-j&1&j\\1&j&-1&-j\\-1&-j&1&j\end{bmatrix}, \mathbf{H}_{2,2} = \frac{1}{4}\begin{bmatrix}1&-1&1&-1\\-1&1&-1&1\\1&-1&1&-1\\-1&1&-1&1\end{bmatrix}, \mathbf{H}_{2,3} = \frac{1}{4}\begin{bmatrix}1&-j&-1&j\\-1&j&1&-j\\1&-j&-1&j\\-1&j&1&-j\end{bmatrix}$$

$$\mathbf{H}_{3,0} = \frac{1}{4}\begin{bmatrix}1&1&1&1\\-j&-j&-j&-j\\-1&-1&-1&-1\\j&j&j&j\end{bmatrix}, \mathbf{H}_{3,1} = \frac{1}{4}\begin{bmatrix}1&j&-1&-j\\-j&1&j&-1\\-1&-j&1&j\\j&-1&-j&1\end{bmatrix}, \mathbf{H}_{3,2} = \frac{1}{4}\begin{bmatrix}1&-1&1&-1\\-j&j&-j&j\\-1&1&-1&1\\j&-j&j&-j\end{bmatrix}, \mathbf{H}_{3,3} = \frac{1}{4}\begin{bmatrix}1&-j&-1&j\\-j&-1&j&1\\-1&j&1&-j\\j&1&-j&-1\end{bmatrix}$$

Example: 4x4 DFT

$$\text{For } \mathbf{F} = \begin{bmatrix} 1 & 2 & 2 & 0 \\ 0 & 1 & 3 & 1 \\ 0 & 1 & 2 & 1 \\ 1 & 2 & 2 & -1 \end{bmatrix}, \text{ compute } T_{k,l}$$

$$\text{Using } T_{k,l} = <\mathbf{H}_{k,l}, \mathbf{F}> \text{ yields, e.g,}$$

$$T_{0,0} = <\mathbf{H}_{0,0}, \mathbf{F}> = \frac{1}{4}\left(\begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix}, \begin{bmatrix} 1 & 2 & 2 & 0 \\ 0 & 1 & 3 & 1 \\ 0 & 1 & 2 & 1 \\ 1 & 2 & 2 & -1 \end{bmatrix}\right) = \frac{1}{4}(1+2+2+0+0+1+3+1+0+1+2+1+1+2+2-1) = \frac{18}{4}$$

$$T_{2,3} = <\mathbf{H}_{2,3}, \mathbf{F}> = \frac{1}{4}\left(\begin{bmatrix} 1 & -j & -1 & j \\ -1 & j & 1 & -j \\ 1 & -j & -1 & j \\ -1 & j & 1 & -j \end{bmatrix}^*, \begin{bmatrix} 1 & 2 & 2 & 0 \\ 0 & 1 & 3 & 1 \\ 0 & 1 & 2 & 1 \\ 1 & 2 & 2 & -1 \end{bmatrix}\right) = \frac{1}{4}(1+2j-2-j+3+j+j-2-j-1-2j+2-j) = \frac{1}{4}(1-j)$$

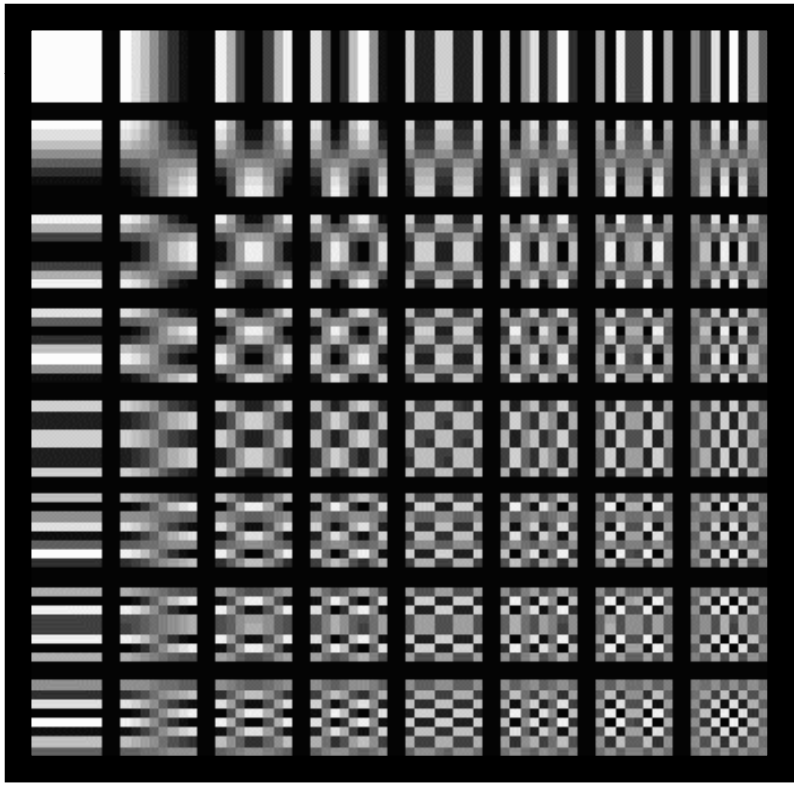
Basis Images of 8x8 DCT

$$H_{k,l}(m,n) = \alpha(k)\alpha(l)\cos\left[\frac{(2m+1)k\pi}{2N}\right]\cos\left[\frac{(2n+1)l\pi}{2N}\right]$$

$$where\ \alpha(k) = \begin{cases} \sqrt{1/N} & k = 0 \\ \sqrt{2/N} & k = 1,\ldots,N-1 \end{cases}$$

Low-Low

High-Low



Low-High

High-High

Property of Separable Transform

- When the transform is separable, we can perform the 2D transform separately.
  - First, do 1D transform for each row using basis vectors g_l,
  - Second, do 1D transform for each column of the intermediate image using basis vectors h_k.
  - Proof:

$$T(k,l) = \sum_{m=0}^{M-1}\sum_{n=0}^{N-1} H_{k,l}^*(m,n)F(m,n) = \sum_{m=0}^{M-1} h_k^*(m)\sum_{n=0}^{N-1} g_l^*(n)F(m,n) = \sum_{m=0}^{M-1} h_k^*(m)U(m,l)$$

DCT on a Real Image Block

```
>>imblock = lena256(128:135,128:135)-128
imblock=
 54   68   71   73   75   73   71   45
  47   52   48   14   20   24   20   -8
  20  -10   -5  -13  -14  -21  -20  -21
 -13  -18  -18  -16  -23  -19  -27  -28
 -24  -22  -22  -26  -24  -33  -30  -23
 -29  -13    3  -24  -10  -42  -41    5
 -16   26   26  -21   12  -31  -40   23
  17   30   50   -5    4   12   10    5
```

```
>>dctblock =dct2(imblock)
dctblock=
 31.0000   51.7034    1.1673  -24.5837  -12.0000  -25.7508   11.9640   23.2873
 113.5766    6.9743  -13.9045   43.2054   -6.0959   35.5931  -13.3692  -13.0005
 195.5804   10.1395   -8.6657   -2.9380  -28.9833   -7.9396    0.8750    9.5585
  35.8733  -24.3038  -15.5776  -20.7924   11.6485  -19.1072   -8.5366    0.5125
  40.7500  -20.5573  -13.6629   17.0615  -14.2500   22.3828   -4.8940  -11.3606
   7.1918  -13.5722   -7.5971  -11.9452   18.2597  -16.2618   -1.4197   -3.5087
  -1.4562  -13.3225   -0.8750    1.3248   10.3817   16.0762    4.4157    1.1041
  -6.7720   -2.8384    4.1187    1.1118   10.5527   -2.7348   -3.2327    1.5799
```

Note that low-low coefficients are much larger than high-high coefficients

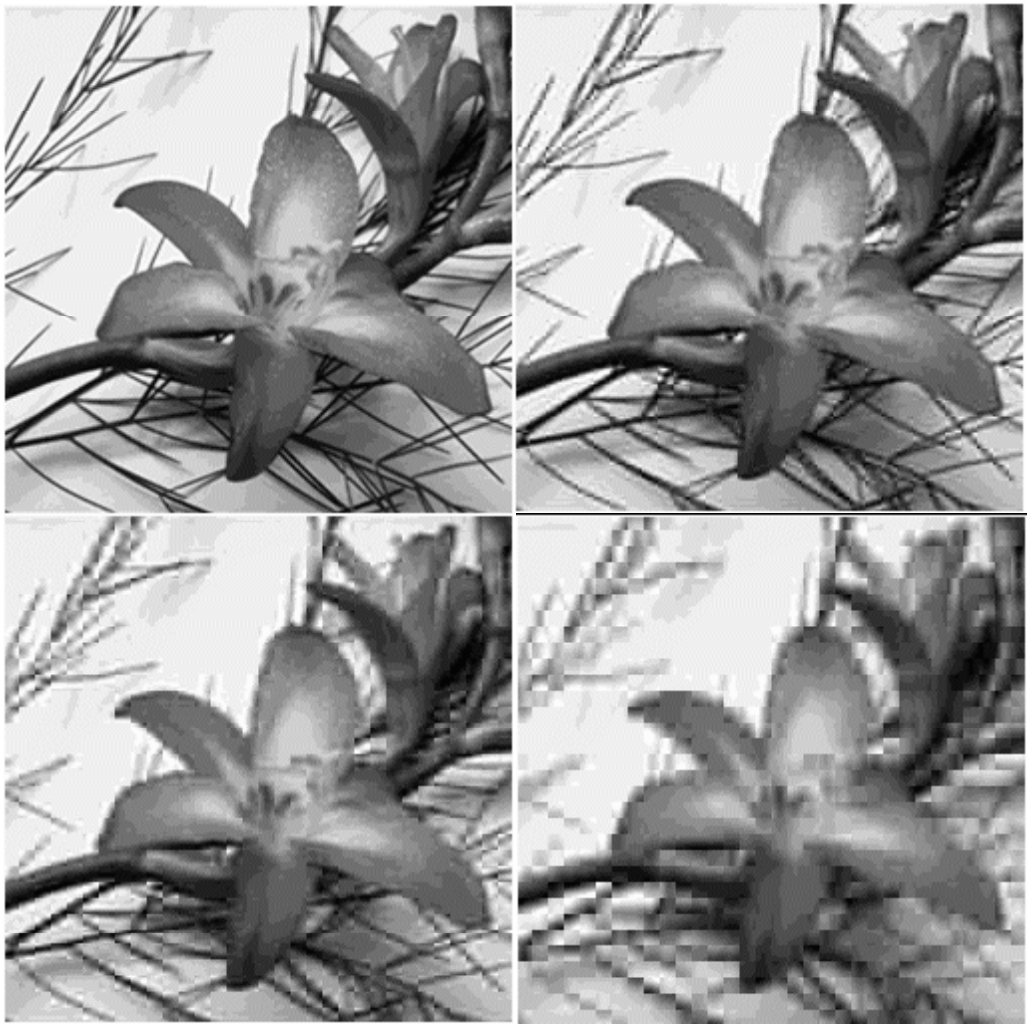
In JPEG, “imblock-128” is done before DCT to shift the mean to zero

DCT Coefficient Distribution

Approximation by DCT Basis

Original

With 16/64 Coefficients



With 8/64 Coefficients

With 4/64 Coefficients

Quantization of 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 (aka quantization matrix)
  - Normalization matrix can then be scaled by a scale factor (aka quality factor or QP)

Default Normalization Matrix in JPEG

Normalization Matrix

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

Actual step size for C(i,j): Q(i,j) = QP*M(i,j)

Example: Quantized Indices

```
>>dctblock =dct2(imblock)
dctblock=
 31.0000   51.7034    1.1673  -24.5837  -12.0000  -25.7508   11.9640   23.2873
 113.5766    6.9743  -13.9045   43.2054   -6.0959   35.5931  -13.3692  -13.0005
 195.5804   10.1395   -8.6657   -2.9380  -28.9833   -7.9396    0.8750    9.5585
  35.8733  -24.3038  -15.5776  -20.7924   11.6485  -19.1072   -8.5366  0.5125
  40.7500  -20.5573  -13.6629   17.0615  -14.2500   22.3828   -4.8940  -11.3606
   7.1918  -13.5722   -7.5971  -11.9452   18.2597  -16.2618   -1.4197   -3.5087
  -1.4562  -13.3225   -0.8750    1.3248   10.3817   16.0762    4.4157    1.1041
  -6.7720   -2.8384    4.1187    1.1118   10.5527   -2.7348   -3.2327    1.5799
```

```
>>QP=1;
>>QM=Qmatrix*QP;
>>qdct=floor((dctblock+QM/2)./(QM))
qdct =
   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
```

Only 19 coefficients are retained out of 64

Example: Quantized Coefficients

%dequantized DCT block

```
>>  iqdct=qdct.*QM
iqdct=
```

32	55	0	-32	0	-40	0	0
108	12	-14	38	0	58	0	0
196	13	-16	0	-40	0	0	0
42	-17	-22	-29	0	0	0	0
36	-22	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 DCT block

dctblock=

31.0000	51.7034	1.1673	-24.5837	-12.0000	-25.7508	11.9640	23.2873
113.5766	6.9743	-13.9045	43.2054	-6.0959	35.5931	-13.3692	-13.0005
195.5804	10.1395	-8.6657	-2.9380	-28.9833	-7.9396	0.8750	9.5585
35.8733	-24.3038	-15.5776	-20.7924	11.6485	-19.1072	-8.5366	0.5125
40.7500	-20.5573	-13.6629	17.0615	-14.2500	22.3828	-4.8940	-11.3606
7.1918	-13.5722	-7.5971	-11.9452	18.2597	-16.2618	-1.4197	-3.5087
-1.4562	-13.3225	-0.8750	1.3248	10.3817	16.0762	4.4157	1.1041
-6.7720	-2.8384	4.1187	1.1118	10.5527	-2.7348	-3.2327	1.5799

Note that many coefficients become zero after quantization!

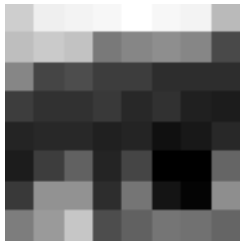
Example: Reconstructed Image

%reconstructed image block

```
>>  qimblock=round(idct2(iqdct))
qimblock=
   58   68   85   79   61   68   67   38
   45   38   39   33   22   24   19   -2
   21    2  -11  -12  -13  -19  -24  -27
   -8  -19  -31  -26  -20  -35  -37  -15
  -31  -17  -21  -20  -16  -39  -41    0
  -33    3   -1  -14  -11  -37  -44    1
  -16   32   18  -10    1  -16  -30    8
    3   54   30   -6   16   11   -7   23
```

Original image block

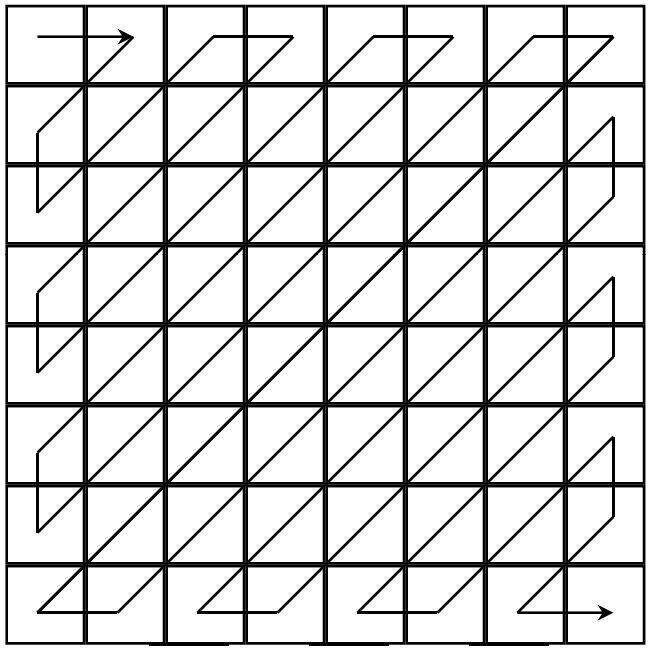
```
imblock=
   54   68   71   73   75   73   71   45
   47   52   48   14   20   24   20   -8
   20  -10   -5  -13  -14  -21  -20  -21
  -13  -18  -18  -16  -23  -19  -27  -28
  -24  -22  -22  -26  -24  -33  -30  -23
  -29  -13    3  -24  -10  -42  -41    5
  -16   26   26  -21   12  -31  -40   23
   17   30   50   -5    4   12   10    5
```



Coding of DCT Coefficients

- Many coefficients are zeros after quantization.
- Specify which coefficients are non-zero by using run-lengths of zeros.
- Run-Length Coding - Used in JPEG standard
  - Ordering coefficients in the zig-zag order
  - Each symbol=(length-of-zero, non-zero-value)
  - Code each symbol using Huffman coding

Zig-Zag Ordering of DCT Coefficients



0	1	5	6	14	15	27	28
2	4	7	13	16	26	29	42
3	8	12	17	25	30	41	43
9	11	18	24	31	40	44	53
10	19	23	32	39	45	52	54
20	22	33	38	46	51	55	60
21	34	37	47	50	56	59	61
35	36	48	49	57	58	62	63

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 decreases in this order.

Example

qdct =

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

Run-length symbol representation:
{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

What is 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

The 1992 JPEG Standard

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

Coding of Quantized DCT Coefficients

- 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: Runlength 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
    - For more details on the actual coding table, see Handout (Sec.8.5.3 in [Gonzalez02]
- One can use default Huffman tables or specify its own tables.
- Instead of Huffman coding, arithmetic coding can be used to achieve higher coding efficiency at an added complexity.

Coding of DC Symbols

- Example:
  - 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 (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” is the number 1 (starting from 0) in category 2, with a fixed length code of “01”.
      - The overall codeword is “10001”

JPEG Tables for Coding DC

TABLE 8.17
JPEG coefficient coding categories.

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

TABLE 8.18
JPEG default DC code (luminance).

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

Coding of AC Coefficients

- Example:
  - First symbol (0,5)
    - The value ‘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’
  - ETC

JPEG Tables for Coding AC (Run,Category) Symbols

Run/ Category	Base Code	Length	Run/ Category	Base Code	Length
0/0	**1010 (= EOB)**	**4**			
0/1	00	3	8/1	11111010	9
0/2	01	4	8/2	111111111000000	17
0/3	100	6	8/3	1111111110110111	19
0/4	1011	8	8/4	1111111110111000	20
0/5	11010	10	8/5	1111111110111001	21
0/6	111000	12	8/6	1111111110111010	22
0/7	1111000	14	8/7	1111111110111011	23
0/8	1111110110	18	8/8	1111111110111100	24
0/9	1111111110000010	25	8/9	1111111110111101	25
0/A	1111111110000011	26	8/A	1111111110111110	26
1/1	1100	5	9/1	111111000	10
1/2	111001	8	9/2	1111111110111111	18
1/3	1111001	10	9/3	1111111111000000	19
1/4	111110110	13	9/4	1111111111000001	20
1/5	11111110110	16	9/5	1111111111000010	21
1/6	1111111110000100	22	9/6	1111111111000011	22
1/7	1111111110000101	23	9/7	1111111111000100	23
1/8	1111111110000110	24	9/8	1111111111000101	24
1/9	1111111110000111	25	9/9	1111111111000110	25
1/A	1111111110001000	26	9/A	1111111111000111	26
2/1	11011	6	A/1	111111001	10
2/2	11111000	10	A/2	1111111111001000	18
2/3	1111110111	13	A/3	1111111111001001	19
2/4	1111111110001001	20	A/4	1111111111001010	20
2/5	1111111110001010	21	A/5	1111111111001011	21
2/6	1111111110001011	22	A/6	1111111111001100	22
2/7	1111111110001100	23	A/7	1111111111001101	23

TABLE 8.19
JPEG default AC code (luminance) *(continues on next page)*.

JPEG Performance for B/W images

65536 Bytes
8 bpp

4839 Bytes
0.59 bpp
CR=13.6

3037 Bytes
0.37 bpp
CR=21.6

1818 Bytes
0.22 bpp
CR=36.4

JPEG for Color 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
  - Convent (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 (3-10 cpd) than to the chrominance components (0.1-0.5 cpd). (Note: cpd is cycles/degree)
- JPEG standard is designed to handle an image consists of many (up to 100) components

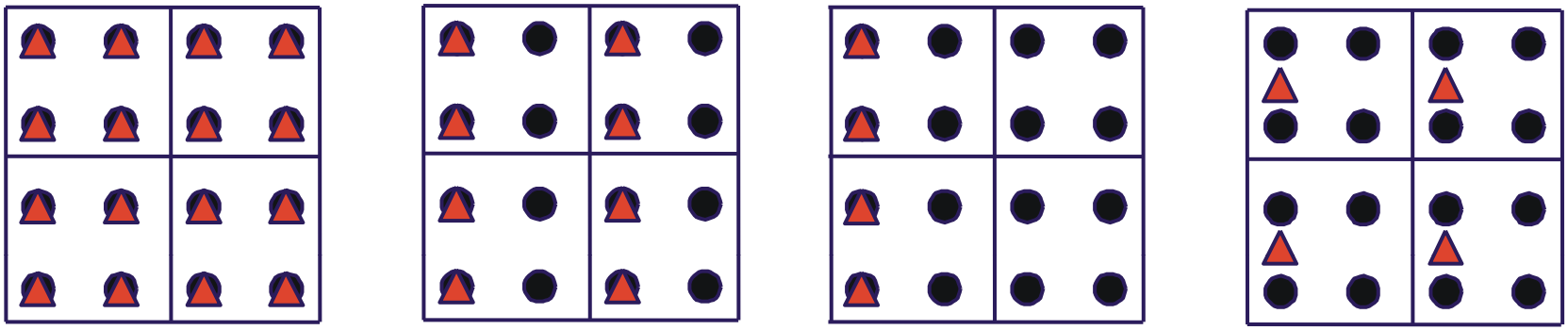
RGB <-> YCbCr Conversion

$$\begin{bmatrix} Y \\ C_b \\ C_r \end{bmatrix} = \begin{bmatrix} 0.299 & 0.587 & 0.114 \\ -0.169 & -0.331 & 0.500 \\ 0.500 & -0.419 & -0.081 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix} + \begin{bmatrix} 0 \\ 128 \\ 128 \end{bmatrix}$$

$$\begin{bmatrix} R \\ G \\ B \end{bmatrix} = \begin{bmatrix} 1.000 & -0.001 & 1.402 \\ 1.000 & -0.344 & -0.714 \\ 1.000 & 1.772 & 0.001 \end{bmatrix} \begin{bmatrix} Y \\ C_b - 128 \\ C_r - 128 \end{bmatrix}$$

Note: Cb ~ Y-B, Cr ~ Y-R, are known as color difference signals.

Chrominance Subsampling



4:4:4
For every 2x2 Y Pixels
4 Cb & 4 Cr Pixel
(No subsampling)

4:2:2
For every 2x2 Y Pixels
2 Cb & 2 Cr Pixel
(Subsampling by 2:1 horizontally only)

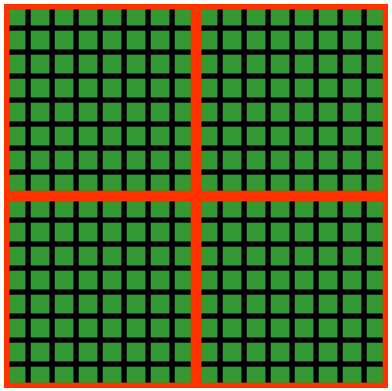
4:1:1
For every 4x1 Y Pixels
1 Cb & 1 Cr Pixel
(Subsampling by 4:1 horizontally only)

4:2:0
For every 2x2 Y Pixels
1 Cb & 1 Cr Pixel
(Subsampling by 2:1 both horizontally and vertically)

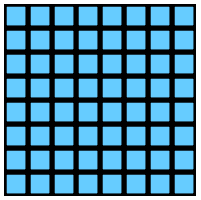
● Y Pixel ▲ Cb and Cr Pixel

4:2:0 is the most common format

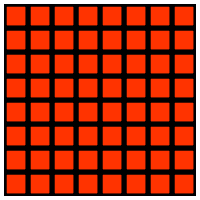
Coding Unit in JPEG



4 8x8 Y blocks



1 8x8 Cb blocks



1 8x8 Cr blocks

Each basic coding unit (called a data unit) is a 8x8 block in any color component. In the interleaved mode, 4 Y blocks and 1 Cb and 1 Cr blocks are processed as a group (called a minimum coding unit or MCU) for a 4:2:0 image.

Default Quantization Table

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

The encoder can specify the quantization tables different from the default ones as part of the header information

Performance of JPEG

- For color images at 24 bits/pixel (bpp)
  - 0.25-0.5 bpp: moderate to good
  - 0.5-0.75 bpp: good to very good
  - 0.75-1.5 bpp: excellent, sufficient for most applications
  - 1.5-2 bpp: indistinguishable from original
  - From: G. K. Wallace: The JPEG Still picture compression standard, *Communications of ACM*, April 1991.
- For grayscale images at 8 bpp
  - 0.5 bpp: excellent quality

JPEG Performance

487x414 pixels,
Uncompressed, 600471 Bytes,24 bpp

85502 Bytes, 3.39 bpp, CR=7

487x414 pixels
41174 Bytes, 1.63 bpp, CR=14.7

JPEG Pros and Cons

- Pros
  - Low complexity
  - Memory efficient
  - Reasonable coding efficiency
- Cons
  - Single resolution
  - Single quality
  - No target bit rate
  - Blocking artifacts at low bit rate
  - No lossless capability
  - Poor error resilience
  - No tiling
  - No regions of interest

Homework

1. For the 2x2 image **S** given below, compute its 2D DCT, reconstruct it by retaining different number of coefficients to evaluate the effect of different basis images.

   a) Determine the four DCT basis images.

   b) Determine the 2D-DCT coefficients for **S,** *Tk,l, k=0,1;l=0,1.*

   c) Show that the reconstructed image from the original DFT coefficients equal to the original image.

   d) Modify the DCT coefficients using the given window masks (**W**1 to **W**5) and reconstruct the image using the modified DCT coefficients. (for a given mask, "1" indicates to retain that coefficient, "0" means to set the corresponding coefficient to zero) What effect do you see with each mask and why?

$$\mathbf{S}=\begin{bmatrix}9 & 1\\ 1 & 9\end{bmatrix},\mathbf{W}_1=\begin{bmatrix}1 & 0\\ 0 & 0\end{bmatrix},\mathbf{W}_2=\begin{bmatrix}0 & 1\\ 0 & 0\end{bmatrix},\mathbf{W}_3=\begin{bmatrix}0 & 0\\ 1 & 0\end{bmatrix},\mathbf{W}_4=\begin{bmatrix}0 & 0\\ 0 & 1\end{bmatrix},\mathbf{W}_5=\begin{bmatrix}1 & 0\\ 0 & 1\end{bmatrix}$$

2. For the same image **S** as given in Prob. 1, quantize its DCT coefficients using the quantization matrix **Q**. Determine the quantized coefficient indices and quantized values. Also, determine the reconstructed image from the quantized coefficients.

$$Q=\begin{bmatrix}3 & 3\\ 3 & 5\end{bmatrix}$$

Homework (cnt'd)

3. Describe briefly how JPEG compresses an image. You may want to break down your discussion into three parts:
   a) How does JPEG compress a 8x8 image block (three steps are involved)
   b) How does JPEG compress a gray-scale image
   c) How does JPEG compress a RGB color image
4. Suppose the DCT coefficient matrix for an 4x4 image block is as shown below (*dctblock*).
   a) Quantize its DCT coefficients using the quantization matrix *Q* given below, assuming QP=1. Determine the quantized coefficient indices and quantized values.
   b) Represent the quantized indices using the run-length representation. That is, generate a series of symbols, the first being the quantized DC index, the following symbols each consisting of a length of zeros and the following non-zero index, the last symbol is EOB (end of block).
   c) Encode the DC index and the runlength symbols from (b) using the JPEG coding method, with the coding tables given in the lecture note. For this problem, assuming the quantized index for the DC coefficient of the previous block is 60.

$$dctblock = 1.0e+003 * \begin{bmatrix} 1.3676 & -0.0500 & -0.0466 & 0.0912 \\ 0.0134 & -0.0033 & 0.0877 & 0.0071 \\ -0.0086 & -0.0207 & 0.0036 & 0.0019 \\ -0.0046 & 0.0086 & 0.0044 & 0.0085 \end{bmatrix}, \qquad Q = \begin{bmatrix} 16 & 10 & 24 & 51 \\ 14 & 16 & 40 & 69 \\ 18 & 37 & 68 & 103 \\ 49 & 78 & 103 & 120 \end{bmatrix},$$

Computer Assignment

1. Computer assignment: Write a program to examine the effect of quantizing the DCT coefficients. For each 8 x 8 block in the image, your program should first calculate the DCT of the block, quantize the coefficients, and then take the inverse DCT. For quantization, using a scaled version of the JPEG quantization matrix as the stepsizes. Examine the resulting image quality with the followng values for the scaling factor: 0.5, 1, 2, 4, 8. What is the largest value of the scaling factor at which the reconstructed image quality is very close to the original image? Hint: you should make use of the ”blkproc” function in MATLAB to speed up your program.
2. Modify the program above, so that it keeps only first N coefficients in the zig-zag order. N should be a user-selectable parameter. Examine the resulting image quality when N=1, 4, 16, 32. For the image you have chosen, how many DCT coefficients should be retained to maintain a sufficient quality?

Reading

- R. Gonzalez, “Digital Image Processing,” Section 8.5 - 8.6
- A.K. Jain, “Fundamentals of Digital Image Processing,” Section 11.4 – 11.5