

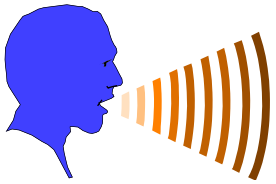
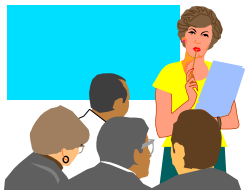



Image and Video Processing

Orthonormal Transforms and Image Coding

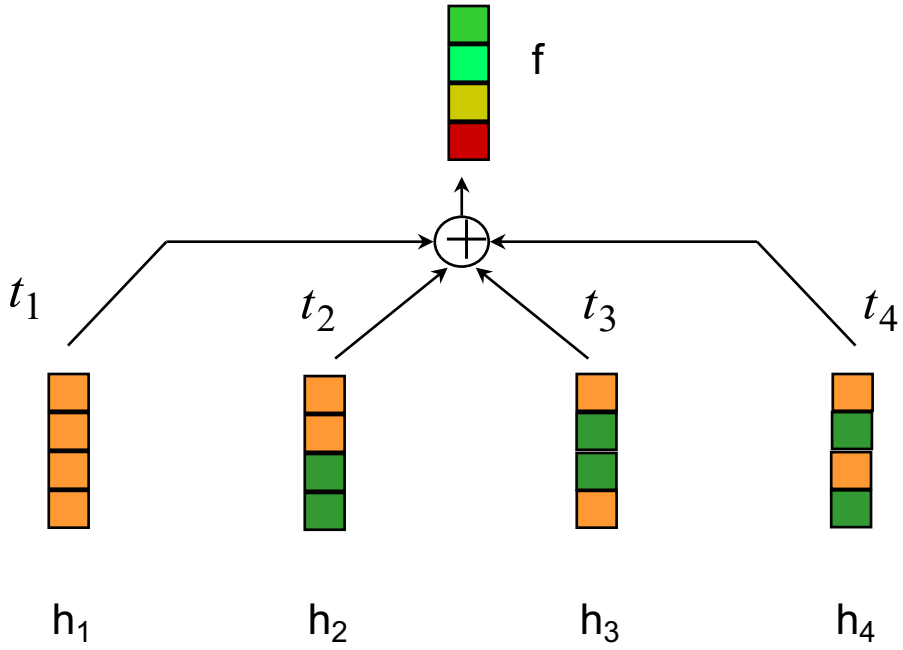
Yao Wang
Tandon School of Engineering, New York University

Outline

- Unitary transform: from 1D to 2D
- Optimal transform: KLT=PCA
- Transform coding framework
- JPEG image coding standard

Transform Representation (1D)

- Represent a vector f as the weighted combination of some basis vectors h_n . Weights t_n are called transform coefficients
- An N-dimensional vector needs N non-linearly dependent bases



One Dimensional Linear Transform

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

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

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

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

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 $$<\mathbf{f}_1,\mathbf{f}_2>=0$$
- 2-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

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

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

$A^H = (A^*)^T$

$A^H = A^T$ if A is real

Example: 4-pt Hadamard Transform

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

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

1D DFT as a Unitary Transform

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

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

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

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

Forward transform:

$$F(k) = \boldsymbol{h}_k^H \mathbf{f}$$

Inverse transform:

$$\mathbf{f} = \sum_{\boldsymbol{k}} \boldsymbol{h_k} F(k) = \boldsymbol{HF}$$

$$\boldsymbol{H} = [\boldsymbol{h_0} \quad \ldots \quad \boldsymbol{h_{N-1}}]$$

Example: 1D DFT, N=2

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

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

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

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

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

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

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

1D Discrete Cosine Transform (DCT)

Basis Vectors:

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

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

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

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

Vector Representation $N = 4$ case:

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

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

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

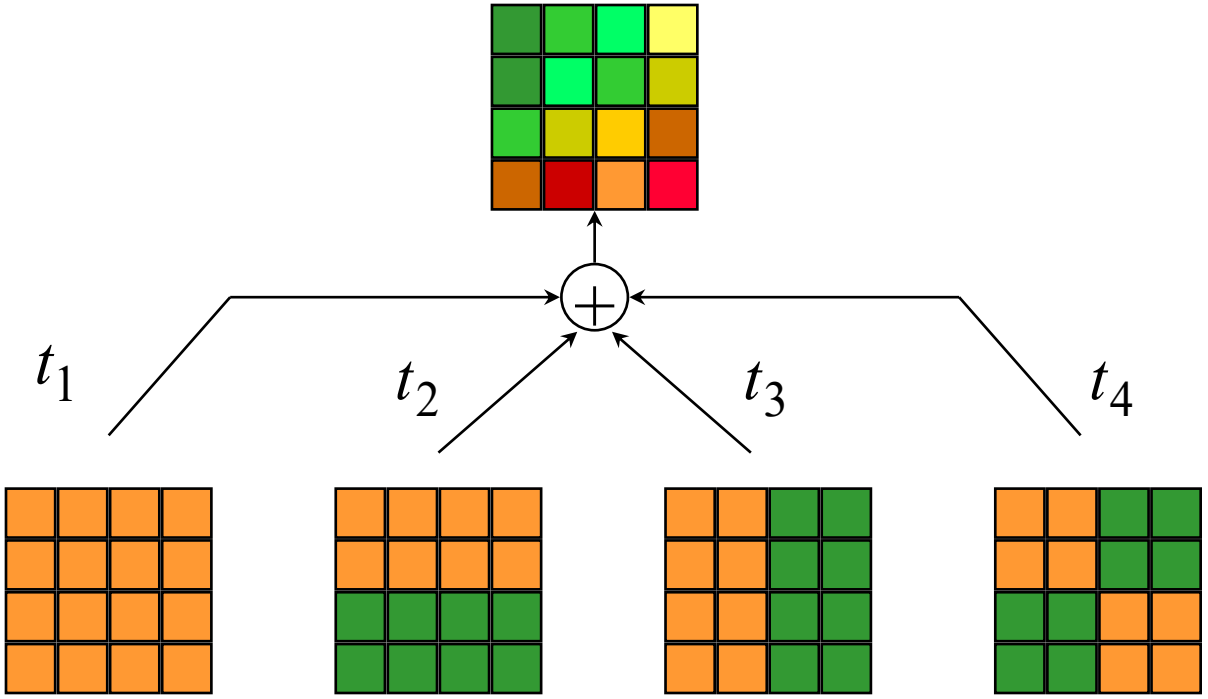
Two Dimensional Transform

- 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:

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

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

Transform Representation of an Image Block



Inverse transform: Represent a block of image samples as the superposition of some basic block patterns (basis images)
Forward transform: Determine the coefficients associated with each basis image

2D Transform Can be Treated as 1D Transform

- Arrange each image block into a vector
- Arrange each basis image as a vector using the same order
- But this does not take advantage of special properties of 2D transforms that are separable

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,
- Let $\mathbf{H}_{k,l}=\mathbf{h}_k\mathbf{g}_l^T$, 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}.
- We often use the same set of 1D basis of dimension N, $\mathbf{h}_k$ to construct a 2D basis of dimension MxM $\mathbf{H}_{k,l}$
  - $\mathbf{H}_{k,l}=\mathbf{h}_k\mathbf{h}_l^T$ k,l=0, 1, …, M-1

Example of Separable Unitary Transform

- Example 1

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

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

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

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:

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

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

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

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

Example: 4x4 DFT

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

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

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

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

Example: 8x8 DCT

Basis Vectors:

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

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

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

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

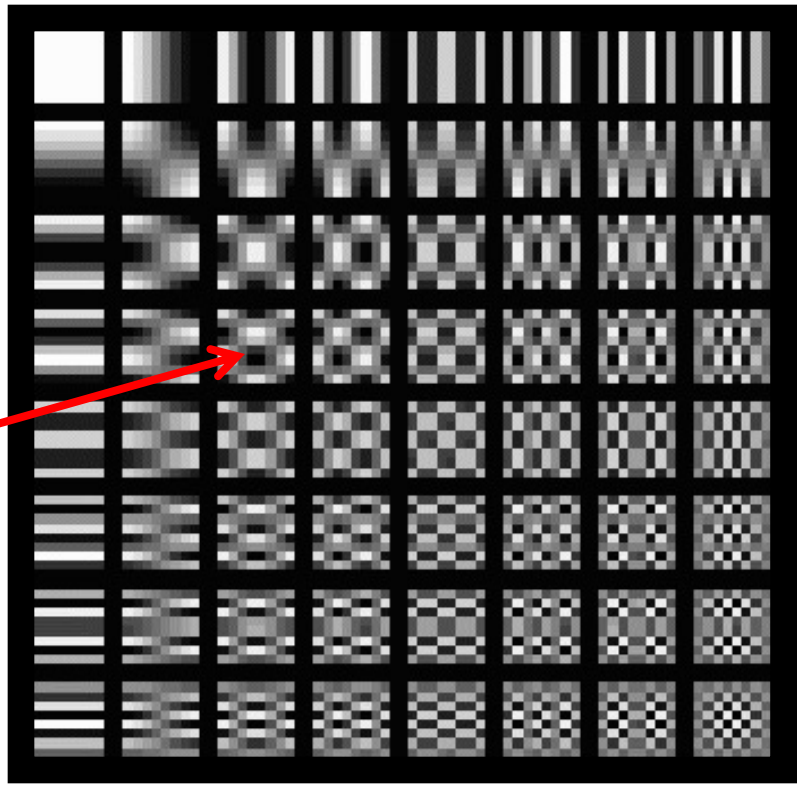
$\mathbf{H}_{k,l}=\mathbf{h}_k\mathbf{h}_l^T$

Low-Low

High-Low

Basis Images:

D=dctmtx(8);
Basis43=D(:,4)*D(3,:)';



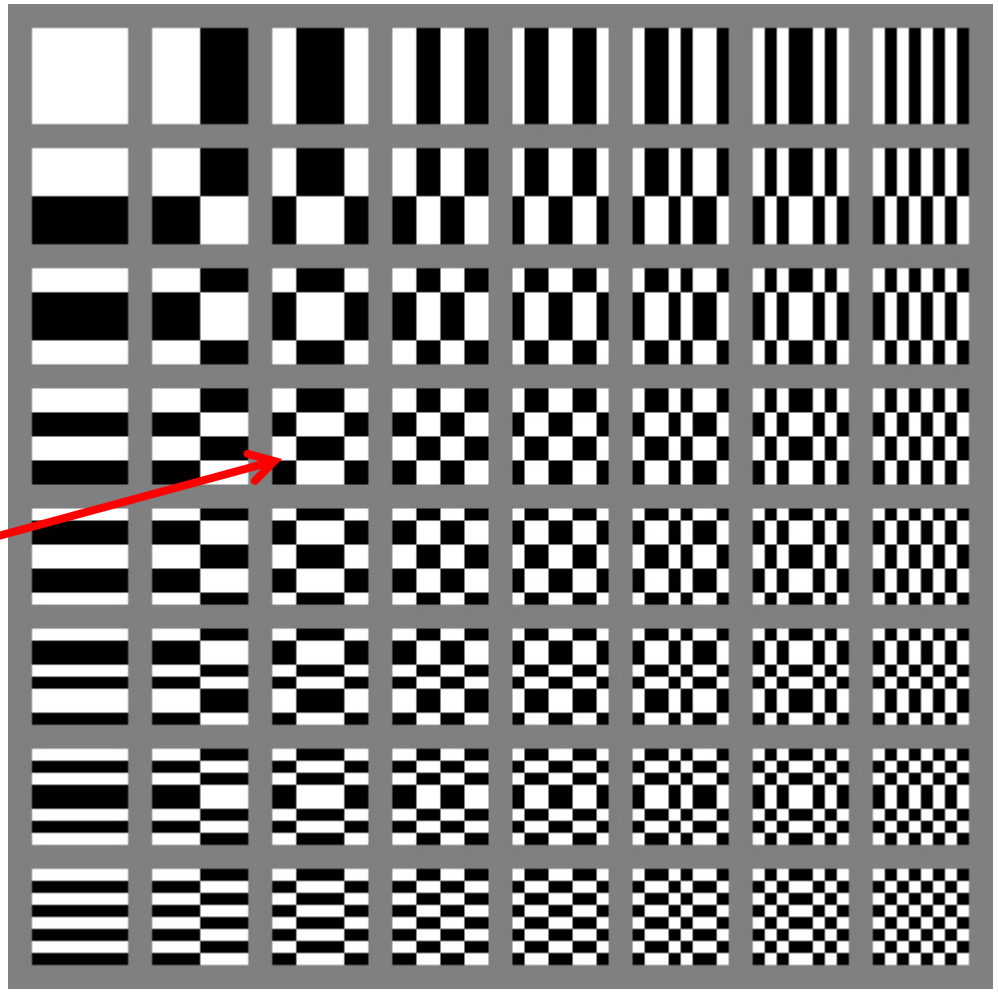
Low-High

High-High

Hadamard Transform: Basis images

Example:

```
D=hadamard(8);
reindex=[1,8,4,5,2,7,3,6];
D(reindex,:)=D;
Basis43=D(:,4)*D(:,3)';
```



From Amy Reibman

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

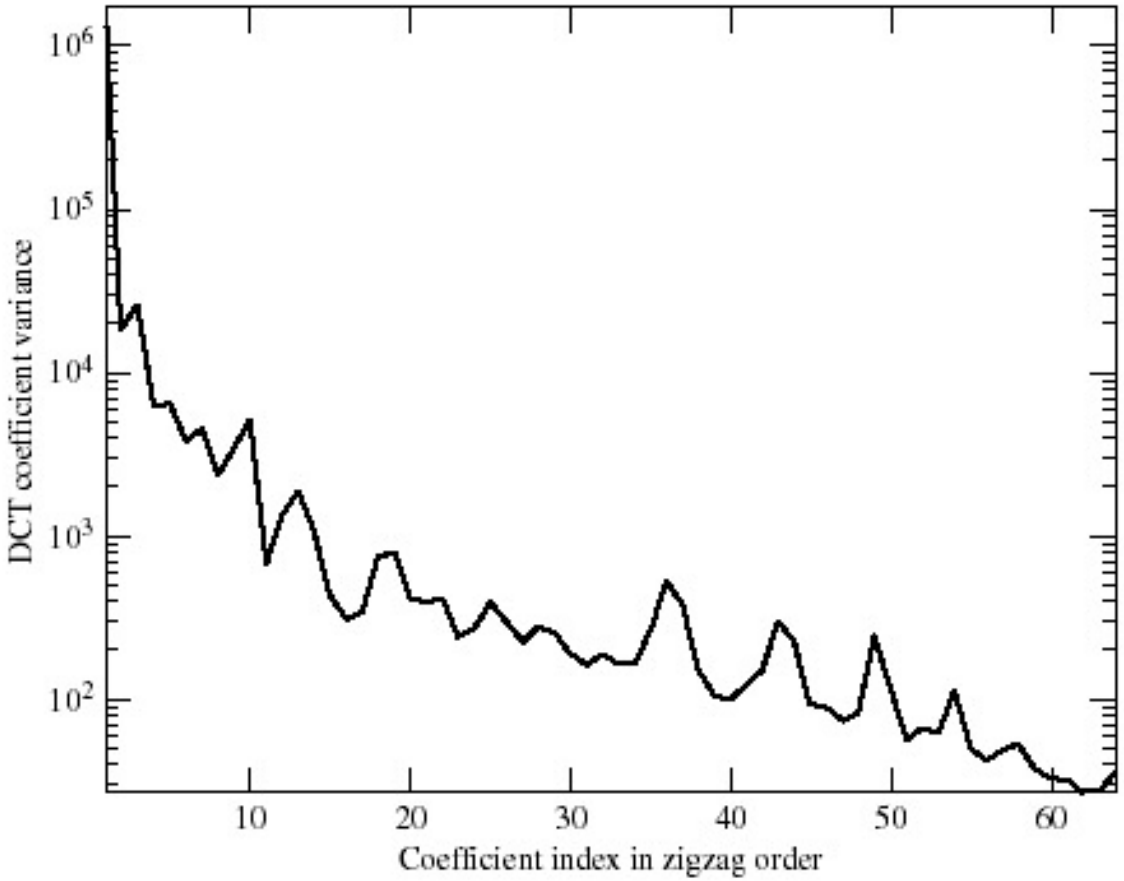
```
>>imblock = lena256(128:135,128:135)-128
imblock=
 54   68   71   73   75   73   71   45
 47   52   48   14   20   24   20   -8
 20  -10   -5  -13  -14  -21  -20  -21
-13  -18  -18  -16  -23  -19  -27  -28
-24  -22  -22  -26  -24  -33  -30  -23
-29  -13    3  -24  -10  -42  -41    5
-16   26   26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

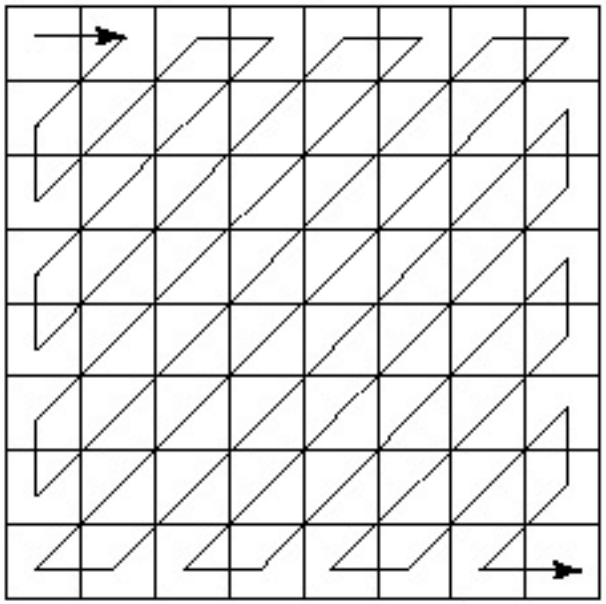
We subtract 128 from original image block to shift the mean to zero

Energy Distribution of DCT Coefficients in Typical Image Blocks



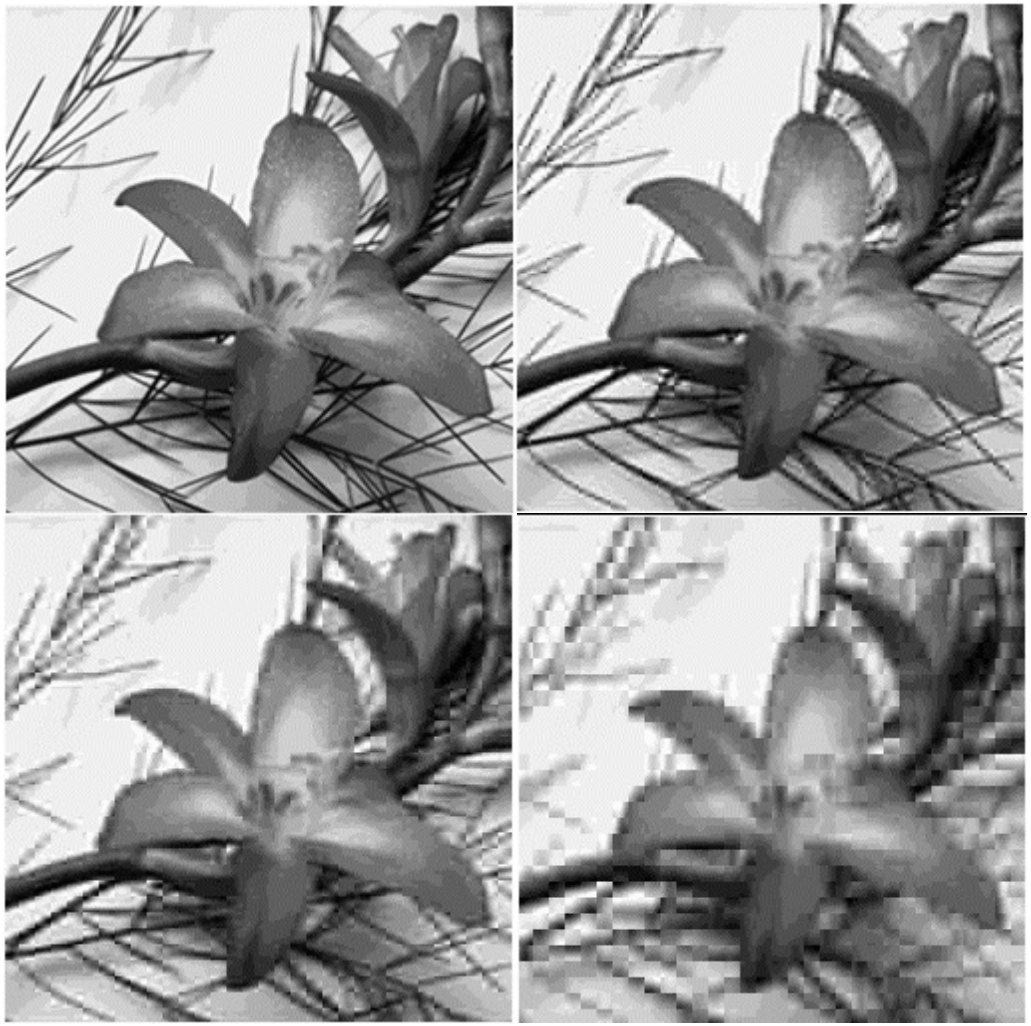
Variance of each coefficient is determined by the average of the square of this coefficient in all blocks of an image

Zig-zag ordering



Images Approximated by Different Number of DCT Coefficients per Block

Original



With 16/64 Coefficients

With 8/64 Coefficients

With 4/64 Coefficients

Properties of Unitary Transforms

$\mathbf{s}=[s_n]$: signal vector; $\mathbf{t}=[t_k]$: transform coefficient vector; $\mathbf{u}_k$: Basis vectors

forward transform: $t_k = \langle \mathbf{u}_k, \mathbf{s} \rangle$ or $\mathbf{t} = [\mathbf{U}]^H \mathbf{s} = [\mathbf{V}]\mathbf{s}$

inverse transform: $\mathbf{s} = \sum_{k \in \mathcal{N}} t_k \mathbf{u}_k = [\mathbf{U}]\mathbf{t} = [\mathbf{V}]^H \mathbf{t}.$

If **s** is a random vector, than **t** is also a random vector

1. The mean vectors $\eta_s = E\{\mathcal{S}\}$ and $\eta_t = E\{\mathcal{T}\}$ are related by

$$\eta_t = [\mathbf{V}]\eta_s, \qquad \eta_s = [\mathbf{V}]^H \eta_t. \tag{9.1.11}$$

The covariance matrices $[\mathbf{C}]_s = E\{(\mathcal{S} - \eta_s)(\mathcal{S} - \eta_s)^H\}$ and $[\mathbf{C}]_t = E\{(\mathcal{T} - \eta_t)(\mathcal{T} - \eta_t)^H\}$ are related by

$$[\mathbf{C}]_t = [\mathbf{V}][\mathbf{C}]_s[\mathbf{V}]^H, \qquad [\mathbf{C}]_s = [\mathbf{V}]^H[\mathbf{C}]_t[\mathbf{V}]. \tag{9.1.12}$$

From Wang, et al, Digital video processing and communications, 2002.

Properties of Unitary Transforms

2. The total energy of the transformed vector equals that of the sample vector. This is true both for a given realization and the ensemble average; that is,

$$\sum_{n\in\mathcal{N}} s_n^2 = \sum_{k\in\mathcal{N}} t_k^2, \tag{9.1.13}$$

$$\sum_{n\in\mathcal{N}} \sigma_{s,n}^2 = \sum_{k\in\mathcal{N}} \sigma_{t,k}^2. \tag{9.1.14}$$

where $\sigma_{s,n}^2 = E\{(\mathcal{S}_n - \eta_{s,n})^2\}$ and $\sigma_{t,k}^2 = E\{(\mathcal{T}_k - \eta_{t,k})^2\}$ are the variances of $\mathcal{S}_n$ and $\mathcal{T}_k$, respectively. (This property is equivalent to Parseval's theorem for the Fourier transform.)

From Wang, et al, Digital video processing and communications, 2002.

Properties of Unitary Transforms

3. Suppose that we use only the first $K < N$ coefficients to approximate $\mathbf{s}$, with the approximated vector being $\hat{\mathbf{s}}_K = \sum_{k=1}^{K} t_k \mathbf{u}_k$; then the approximation error signal is $\mathbf{e}_K = \mathbf{s} - \hat{\mathbf{s}}_K = \sum_{k=K+1}^{N} t_k \mathbf{u}_k$. The approximation error energy for a given $\mathbf{s}$ is

$$\|\mathbf{e}_K\|^2 = \sum_{n \in \mathcal{N}} e_n^2 = \sum_{k=K+1}^{N} t_k^2. \tag{9.1.15}$$

The variance of $\boldsymbol{\mathcal{E}}_K = \boldsymbol{\mathcal{S}} - \hat{\boldsymbol{\mathcal{S}}}_K$ over the ensemble of $\boldsymbol{\mathcal{S}}$ is

$$E\{\|\boldsymbol{\mathcal{E}}_K\|^2\} = \sum_{n \in \mathcal{N}} \sigma_{e,n}^2 = \sum_{k=K+1}^{N} \sigma_{t,k}^2. \tag{9.1.16}$$

Because the sum of all coefficient squares or variances is a constant (Equation (9.1.13) or (9.1.14)), the approximation error for a particular signal vector is minimized if one chooses K coefficients that have the largest values to approximate the original signal. Similarly, the mean approximation error is minimized by choosing K coefficients with the largest variances.

From Wang, et al, Digital video processing and communications, 2002.

Example: 4-pt Hadamard Transform

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

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

Energy preservation: $\|f\|_2^2 = 1^2 + 2^2 + 3^2 + 4^2 = 30, \|t\|_2^2 = 5^2 + (-2)^2 + 0^3 + (-1)^2 = 30$

Suppose we use only the two basis with largest (in magnitude) coefficients to reconstruct the signal:

$$\hat{f} = t_0 h_0 + t_1 h_1 = \frac{5}{2}\begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix} - \frac{2}{2}\begin{bmatrix} 1 \\ 1 \\ -1 \\ -1 \end{bmatrix} = \frac{1}{2}\begin{bmatrix} 3 \\ 3 \\ 7 \\ 7 \end{bmatrix}, \mathrm{e} = f - \hat{f} = \frac{1}{2}\begin{bmatrix} 2 \\ 4 \\ 6 \\ 8 \end{bmatrix} - \frac{1}{2}\begin{bmatrix} 3 \\ 3 \\ 7 \\ 7 \end{bmatrix} = \frac{1}{2}\begin{bmatrix} -1 \\ 1 \\ -1 \\ 1 \end{bmatrix}$$

$$\|e\|_2^2 = 1, t_2^2 + t_3^2 = 1$$

Pop Quiz

- What do we mean by orthonormal transforms
- What properties do the orthogonal transform bases satisfy
- How do you obtain the transform coefficients
- How do you form 2D transform bases from 1D bases?
- How do you perform 2D separable transform?
- What will be the sum of squared errors of the reconstructed signal from K largest coefficients?

Pop Quiz (w/ answers)

- What do we mean by orthonormal transforms
  - Representing a signal by a weighted average of transform bases
- What properties do the orthonormal transform bases satisfy
  - Different bases are orthogonal to each other. Each has a unit norm
- How do you obtain the transform coefficients
  - Taking the inner product of the signal with each transform basis
- How do you form 2D transform bases from 1D bases?
  - Using outproduct of every two 1D bases
- How do you perform 2D separable transform?
  - Perform 1D transform of every row using the 1D row transform bases, then 1D transform of every column of the intermediate image using the 1D column transform bases.
- What will be sum of squared errors of the reconstructed signal from K largest coefficients?
  - The sum of squares of the remaining N-K coefficients

Outline

- Unitary transform: from 1D to 2D
- Optimal transform: KLT=PCA
- Transform coding framework
- JPEG image coding standard

Why do we want to use transform?

- **Compression:** With a well designed set of basis vectors, only a few coefficients of typical image blocks are large, and coefficients are uncorrelated. Each image block can be represented by its quantized transform coefficients!
- **Feature dimension reduction:** Using a few large coefficients rather than original samples.
- **Denoising:** Noise typically contribute to small coefficients at all frequencies, but real data typically have significant coefficients only at low frequencies. We can suppress noise by setting small high frequency coefficients to zero.
- Just as DFT, we could attenuate different DCT coefficients to achieve different filtering effect (but the convolution property does not hold for DCT)

Transform design

- What are desirable properties of a transform for image and video?
  - Decorrelating
    - Good for compression: one can entropy code each coefficient independently without losing efficiency;
    - Good for feature reduction: each feature reveal independent info.
  - High energy compaction – Only a few large coefficients, rest are zero or negligible
  - Easy to compute (few operations)
  - Separable – compute 1-D transform first on rows, then on columns. So we only consider design of 1-D transform bases.
- What size transform should we use?
  - Entire image? Small blocks?
  - 2-D (on an image) or 3-D (incorporating time also)?

Karhunen Loève Transform (KLT) = Principle Component Analysis (PCA)

- Basis vectors in directions with largest variance = eigenvectors (principle components) of the signal covariance matrix.
- Coefficients are completely uncorrelated ☺
- Best energy compaction ☺
  - Sort coefficients from largest to smallest in expected squared magnitude; then the sum of the energies of the first M coefficients is as large as possible
- No computationally efficient algorithm ☹
- Requires the knowledge of the mean and covariance matrix of the signal vector. ☹

Karhunen Loève Transform (KLT)

- KLT: Using eigenvectors of signal covariance matrix $[C]_s$ as transform bases, ordered based on the eigenvalues

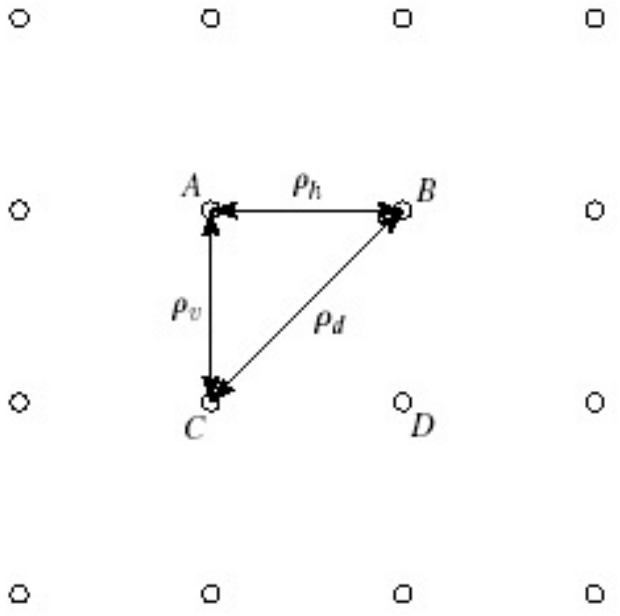
$$[\mathbf{C}]_s \phi_k = \lambda_k \phi_k, \quad \text{with} \quad \langle \phi_k, \phi_l \rangle = \delta_{k,l}.$$

$$\sigma_k^2 = \lambda_k. \qquad [\mathbf{C}]_t = \begin{bmatrix} \lambda_1 & & \\ & \ldots & \\ & & \lambda_N \end{bmatrix}$$

  - Resulting transform coefficients are uncorrelated
  - The coefficient variances are the eigenvalues
  - K-term approximation error = The expected square error of representing a vector with first K coefficients = sum of last N-K eigen-values
  - It can be shown that among all possible unitary transforms, KLT yields the least K-term approximation error, for all K=1,2,…,N.

Example

- Consider 2x2 image blocks with inter-sample correlation as shown below.



$$\rho_h = \rho_v = \rho, \rho_v = \rho^2$$

Example Continued (Convert 2x2 into 4x1)

- Correlation matrix

$$[\mathbf{C}]_s = E\left\{ \begin{bmatrix} A \\ B \\ C \\ D \end{bmatrix} [A \quad B \quad C \quad D] \right\} = \begin{bmatrix} C_{AA} & C_{AB} & C_{AC} & C_{AD} \\ C_{BA} & C_{BB} & C_{BC} & C_{BD} \\ C_{CA} & C_{CB} & C_{CC} & C_{CD} \\ C_{DA} & C_{DB} & C_{DC} & C_{DD} \end{bmatrix}$$

$$= \sigma_s^2 \begin{bmatrix} 1 & \rho_h & \rho_v & \rho_d \\ \rho_h & 1 & \rho_d & \rho_v \\ \rho_v & \rho_d & 1 & \rho_h \\ \rho_d & \rho_v & \rho_h & 1 \end{bmatrix}.$$

- DCT basis images

$$\frac{1}{2}\begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}, \quad \frac{1}{2}\begin{bmatrix} 1 & -1 \\ 1 & -1 \end{bmatrix}, \quad \frac{1}{2}\begin{bmatrix} 1 & 1 \\ -1 & -1 \end{bmatrix}, \quad \frac{1}{2}\begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix}.$$

- Equivalent 1D transform matrix

$$[\mathbf{U}] = \frac{1}{2}\begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{bmatrix}$$

- Covariance matrix after DCT

$$[C_t] = [V][C_s][V]^T = diag\left((1+\rho)^2, 1-\rho^2, 1-\rho^2, (1-\rho)^2\right)$$

Example

- Determine the KLT for the 2x2 image block in the previous example

$$[\mathbf{C}]_s = \sigma_s^2 \begin{bmatrix} 1 & \rho_h & \rho_v & \rho_d \\ \rho_h & 1 & \rho_d & \rho_v \\ \rho_v & \rho_d & 1 & \rho_h \\ \rho_d & \rho_v & \rho_h & 1 \end{bmatrix} \qquad \rho_h = \rho_v = \rho,\ \rho_v = \rho^2$$

Determine the eigenvalues by solving: $\det([\mathbf{C}]_s - \lambda[\mathbf{I}]) = 0$

$$\lambda_k = \{(1+\rho)^2, (1-\rho^2), (1-\rho^2), (1-\rho)^2\}\sigma_s^2.$$

(same as the coefficient variances with DCT)

Determine the eigenvectors by solving $([\mathbf{C}]_s - \lambda[\mathbf{I}])\phi_k = \mathbf{0}$

Resulting transform is the same as the DCT !
In general, DCT is a good transform for images, with energy compaction close to KLT!
DCT is a fixed transform (does not depend on the signal statistics) and can be computed fast.

N Ahmed, T Natarajan, KR Rao, "Discrete Cosine Transform", IEEE Transactions on Computers 23, 90-93

What if I don't know the covariance matrix?

- Using samples to form the covariance matrix

Given samples $f_k, k = 1,2,\ldots,K$

Mean $\mu_f = \frac{1}{K}\sum_{k=1}^{K} f_k$

Covariance matrix $C_f = \frac{1}{K}\sum_{k=1}^{K} (f_k - \mu_f)(f_k - \mu_f)^T$

Let $\boldsymbol{F}$ be the sample matrix, consisting all the mean removed samples in its columns,

$$\boldsymbol{F} = [f_k - \mu; k = 1,2,\ldots,K]$$

Empirical Covariance matrix

$$\boldsymbol{C}_f = \frac{1}{K}\boldsymbol{F}\,\boldsymbol{F}^T$$

Feature Dimension Reduction by PCA

- Given a set of samples in a dataset, each represented by a raw signal vector of dimension *N*
- Want to find a reduced feature representation of dimension *K*
- Principle component analysis (PCA) = KLT using the sample covariance matrix
  - The eigenvector corresponding to the largest eigen value corresponds to the direction with largest variance.
- Features = first *K* transform coefficients
- Feature dimension reduction is an important problem in machine learning.
- PCA is a linear feature reduction method.
- There are other more powerful non-linear transformations.

Signal Independent Transform Bases

- Suboptimal transforms – many available!
  - Discrete Fourier Transform (DFT): complex values;
    - convolution in space = multiplication in coefficients;
    - not best in terms of energy compaction.
  - Discrete Cosine transform (DCT): real values only
    - nearly as good as KLT for common image signals
  - Hadamard and Haar: basis functions contain only +1,0,-1
    - Very fast computation
    - Has blocky artifacts with K-term approximation

Pop Quiz

- How do you determine the optimal transform (KLT)?
- What are its properties?
- What are the difficulty of using KLT in practice?
- Why do we use DCT for images?

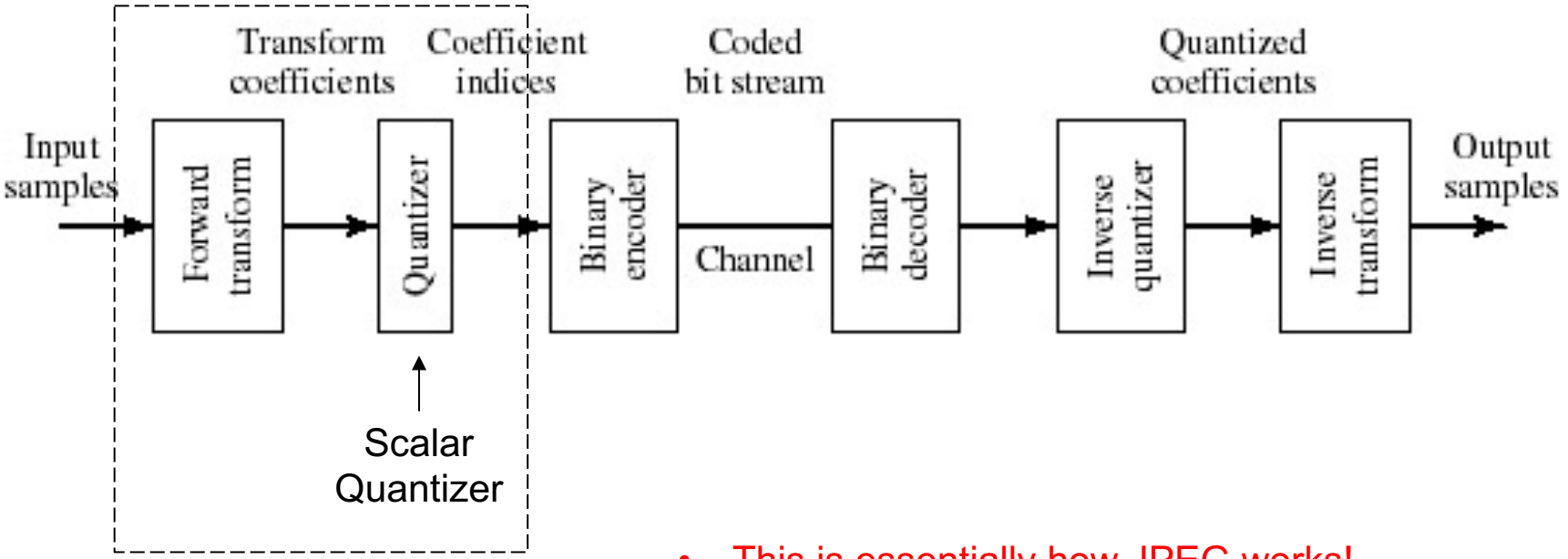
Pop Quiz (w/ Answers)

- How do you determine the optimal transform (KLT)?
  - Find the covariance matrix
  - Find the eigen vectors of the covariance matrix
- What are its properties?
  - Decorrelate the original signal components (i.e. transformed coefficients are uncorrelated, or the covariance matrix of the transformed vector is diagonal
  - Maximize the energy minimization: having the minimal K-term approximation error among all transforms, for K=1,2,..,N
- What are the difficulty of using KLT in practice?
  - Signal dependent, no fast algorithm to compute the transform
- Why do we use DCT for images?
  - Very close to the KLT corresponding to the covariance matrix of common image signals.

Outline

- Unitary transform: from 1D to 2D
- Optimal transform: KLT=PCA
- Transform coding framework
- JPEG image coding standard

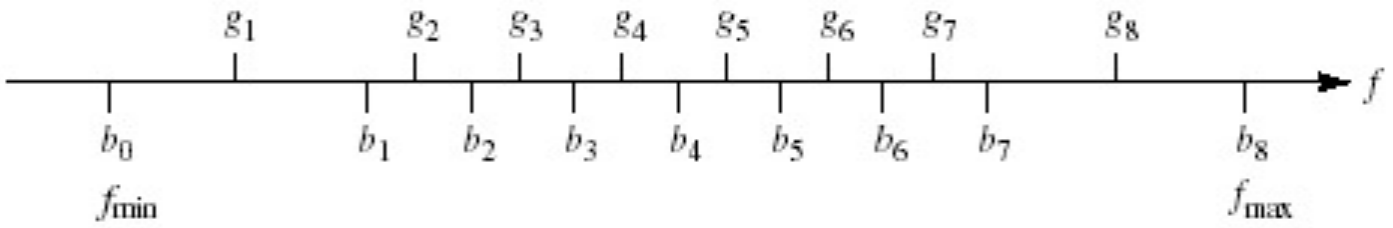
Block Diagram of Transform Coding



- This is essentially how JPEG works!
- JPEG uses 8x8 DCT, and Runlength+Huffman coding for binary encoding.

Scalar Quantization

- Purpose of quantization: to convert the continuous range (requiring infinite number of bits) to a finite number of quantized bins, each bin represented by a single quantized/reconstruction value



Quantization levels: L

Boundary values: b_l

Partition regions: $B_l = [b_{l-1}, b_l)$

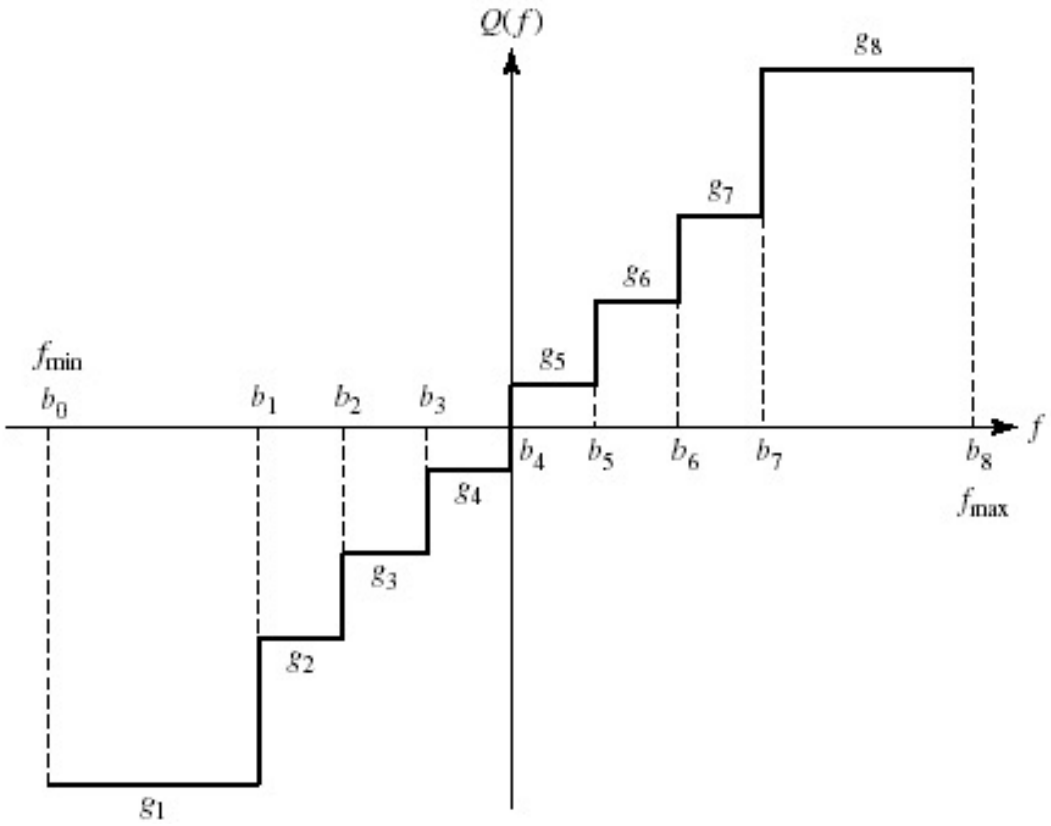
Reconstruction values: g_l

Quantizer mapping: $Q(f) = g_l$, if $f \in B_l$

Quantization index: g_l is indexed by l

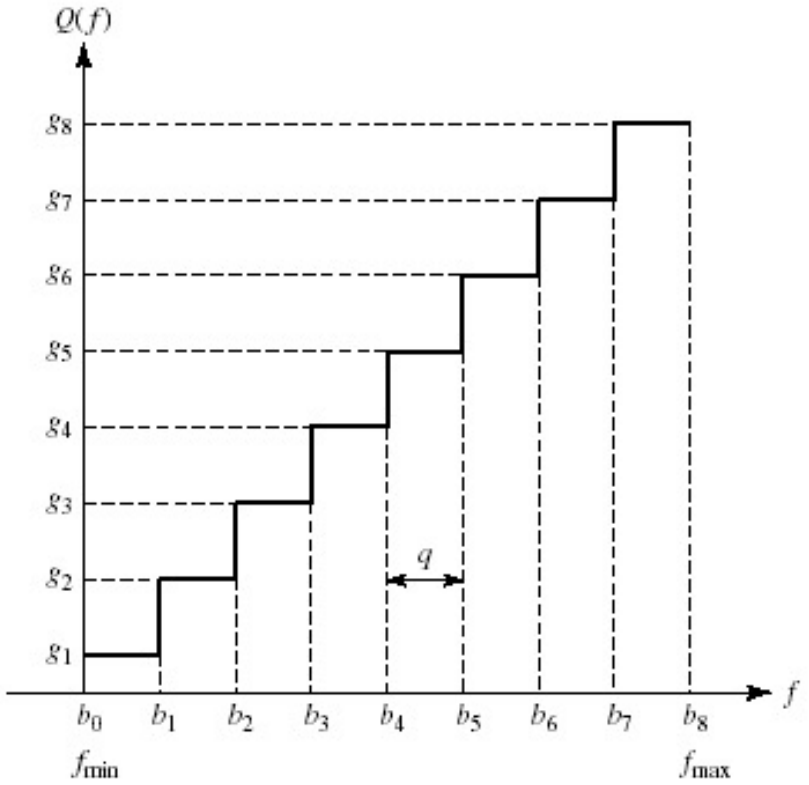
L possible indices can be described by $log_2(L)$ bits

Function Representation of Quantization



$$Q(f) = g_l, \text{ if } f \in B_l$$

Uniform Quantization



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

Alternatively, centered at the mean value:

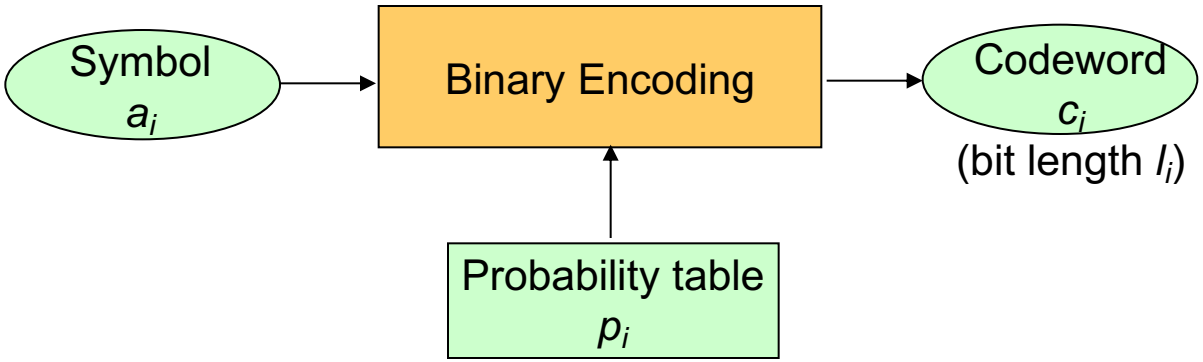
$$Q(f) = \left\lfloor \frac{f - f_{mean} + \frac{q}{2}}{q} \right\rfloor * \mathrm{q} + f_{mean}$$

$(f_{mean} - \frac{q}{2}, f_{mean} + \frac{q}{2})$ is quantized to f_{mean}

All bin sizes are equal. q= quantization interval or quantization step size
Quantization error is in the range (-q/2, q/2).

Lossless Coding (Binary Encoding)

- Binary encoding is a necessary step in any coding system
  - Applies to
    - original symbols (e.g. image pixels) in a discrete source,
    - or converted symbols (e.g. quantized transformed coefficients) from a continuous or discrete source
- Binary encoding process (scalar coding)



Bit rate (bit/symbol): $$R = \sum_{a_i \in \mathcal{A}} p(a_i) l(a_i).$$

Binary Encoding Methods

- Fixed length coding
  - *N* levels represented by *(int)* $log_2(N)$ bits.
  - Ex: simple binary codes
- Variable length coding
  - more frequently appearing symbols represented by shorter codewords (Huffman, arithmetic, LZW=zip).
- The minimum average number of bits required to represent a symbol is bounded by its entropy.

$$R(F) \geq H(F) = lower\ bound$$

Entropy of a RV

- Consider RV $F=\{f_1,f_2,\ldots,f_K\}$, with probability $p_k=Prob.\{F=f_K\}$
- Self-Information of one realization f_k : $H_k= -\log(p_k)$
  - p_k=1: always happen, no information
  - P_k ~0: seldom happen, its realization carries a lot of information
- Entropy = average information: $$H(\mathcal{F}) = -\sum_{f\in\mathcal{A}} p_{\mathcal{F}}(f)\log_2 p_{\mathcal{F}}(f).$$
  - Entropy is a measure of uncertainty or information content, unit=bits
  - Very uncertain -> high information content

Huffman Coding

- Idea: more frequent symbols -> shorter codewords

Step 1: Arrange the symbol probabilities $p(a_l), l = 1, 2, \ldots, L$, in a decreasing order and consider them as leaf nodes of a tree.

Step 2: While there is more than one node:

(a) Find the two nodes with the smallest probability and arbitrarily assign 1 and 0 to these two nodes.

(b) Merge the two nodes to form a new node whose probability is the sum of the two merged nodes. Go back to Step 1.

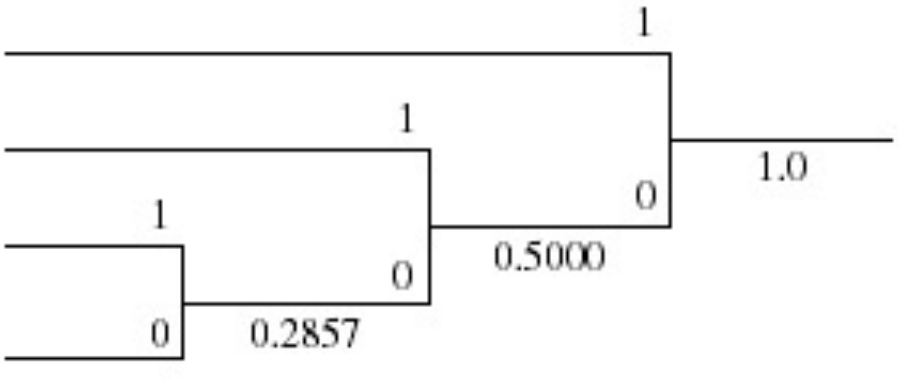
Step 3: For each symbol, determine its codeword by tracing the assigned bits from the corresponding leaf node to the top of the tree. The bit at the leaf node is the last bit of the codeword.

- Can be applied to one symbol at a time (scalar coding), or a group of symbols (vector coding), or one symbol conditioned on previous symbols (conditional coding)
- The resulting bit rate is away from the entropy bound by 1 bit at most!

$$H_1(\mathcal{F}) \le \bar{R}_1(\mathcal{F}) \le H_1(\mathcal{F}) + 1.$$

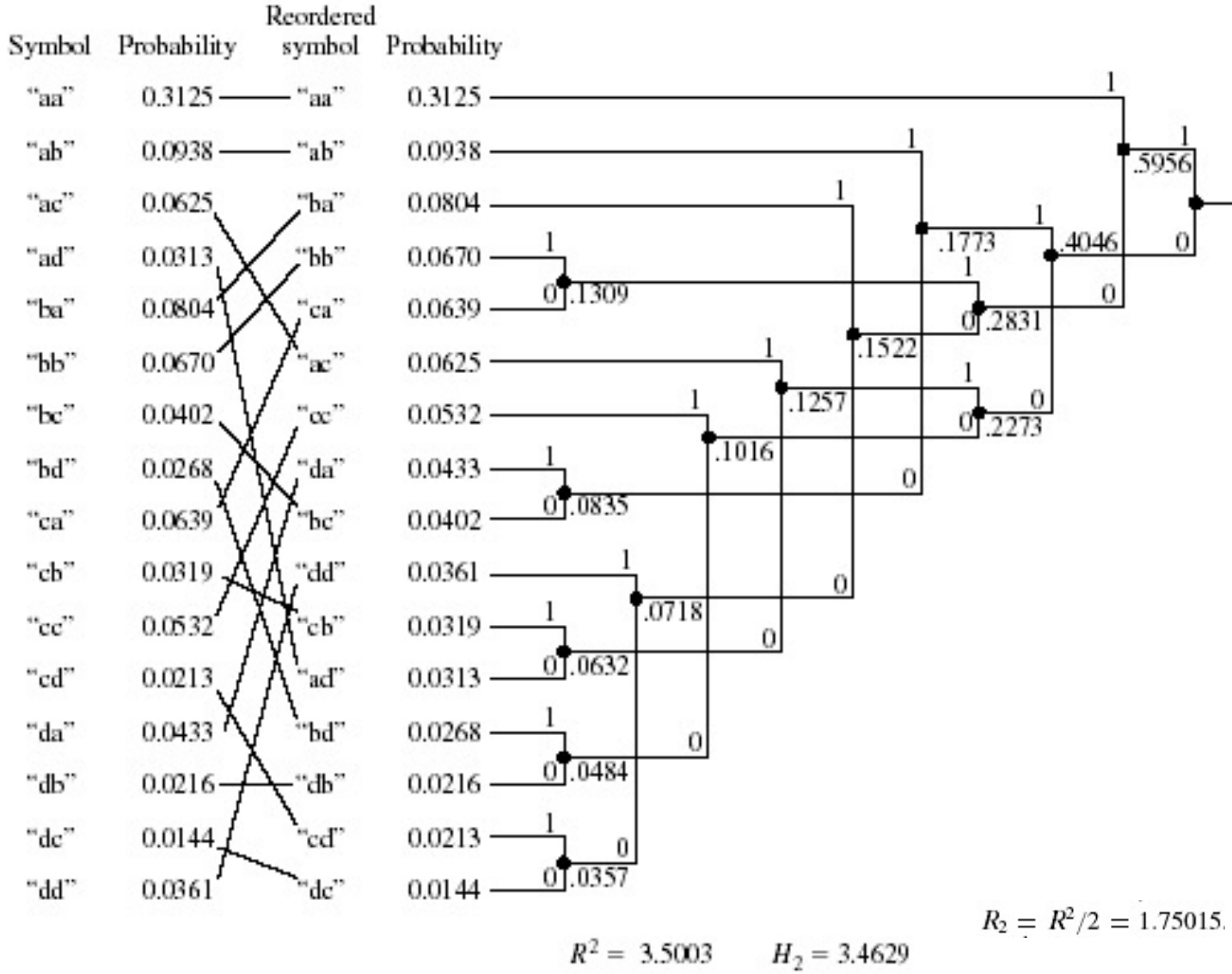
Huffman Coding Example: Scalar Coding

Symbol	Probability	Codeword	Codeword length
"a"	0.5000	"1"	1
"b"	0.2143	"01"	2
"c"	0.1703	"001"	3
"d"	0.1154	"000"	3



Bit rate $R = 1.7857$ Entropy $H_1 = 1.7707$

Huffman Coding Example: Vector Coding



Symbol	Probability	Reordered symbol	Probability	Codeword	Length
"aa"	0.3125	"aa"	0.3125	"11"	2
"ab"	0.0938	"ab"	0.0938	"011"	3
"ac"	0.0625	"ba"	0.0804	"1001"	4
"ad"	0.0313	"bb"	0.0670	"1011"	4
"ba"	0.0804	"ca"	0.0639	"1010"	4
"bb"	0.0670	"ac"	0.0625	"0011"	4
"bc"	0.0402	"cc"	0.0532	"0001"	4
"bd"	0.0268	"da"	0.0433	"0101"	4
"ca"	0.0639	"bc"	0.0402	"0100"	4
"cb"	0.0319	"dd"	0.0361	"10001"	5
"cc"	0.0532	"cb"	0.0319	"00101"	5
"cd"	0.0213	"ad"	0.0313	"00100"	5
"da"	0.0433	"bd"	0.0268	"00001"	5
"db"	0.0216	"db"	0.0216	"00000"	5
"dc"	0.0144	"cd"	0.0213	"100001"	6
"dd"	0.0361	"dc"	0.0144	"100000"	6

$R_2 = R^2/2 = 1.75015$

$R^2 = 3.5003 \qquad H_2 = 3.4629$

Outline

- Unitary transform: from 1D to 2D
- Optimal transform: KLT=PCA
- Transform coding framework
- JPEG image coding standard

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

Quantization of DCT Coefficients

- DC coefficient is predicted from the DC of the previous block, and the prediction error is quantized.
- Use uniform quantizer on the DC prediction error and each AC 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
- JPEG bit allocation does not intend to minimize MSE!

Default Normalization Matrix in JPEG

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)

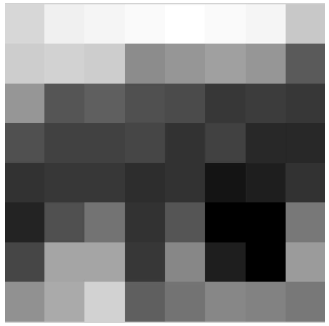
Note that the stepsize for the DC coefficient is for the prediction error for DC, not the original DC value. High-High coefficients are quantized about 10 times of low-low coefficients!

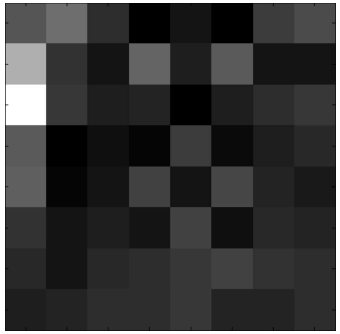
Example: DCT on a Real Image Block

>>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)

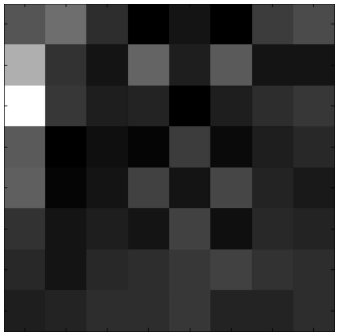
dctblock=

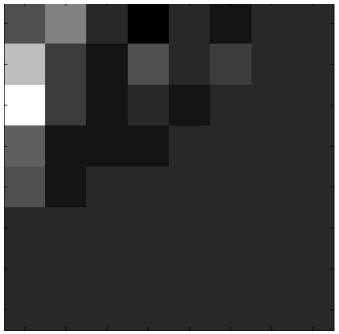
31.0000	51.7034	1.1673	-24.5837	-12.0000	-25.7508	11.9640	23.2873
113.5766	6.9743	-13.9045	43.2054	-6.0959	35.5931	-13.3692	-13.0005
195.5804	10.1395	-8.6657	-2.9380	-28.9833	-7.9396	0.8750	9.5585
35.8733	-24.3038	-15.5776	-20.7924	11.6485	-19.1072	-8.5366	0.5125
40.7500	-20.5573	-13.6629	17.0615	-14.2500	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

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

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

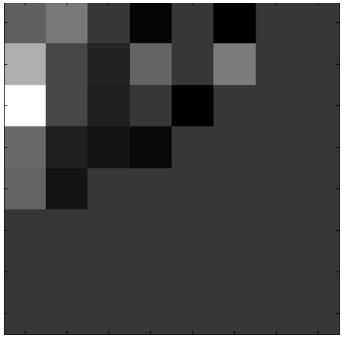
Example: Quantized Coefficients

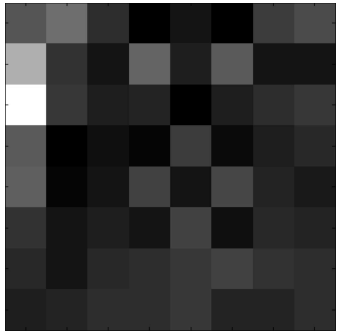
%dequantized DCT block

>> 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

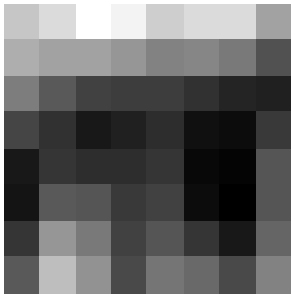
Example: Reconstructed Image

%reconstructed image block

>> qimblock=round(idct2(iqdct))

qimblock=

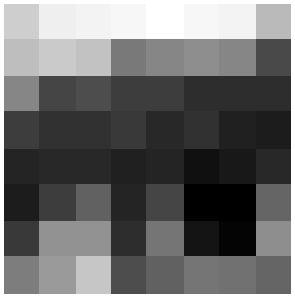
58	68	85	79	61	68	67	38
45	38	39	33	22	24	19	-2
21	2	-11	-12	-13	-19	-24	-27
-8	-19	-31	-26	-20	-35	-37	-15
-31	-17	-21	-20	-16	-39	-41	0
-33	3	-1	-14	-11	-37	-44	1
-16	32	18	-10	1	-16	-30	8
3	54	30	-6	16	11	-7	23



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 Quantized DCT Coefficients

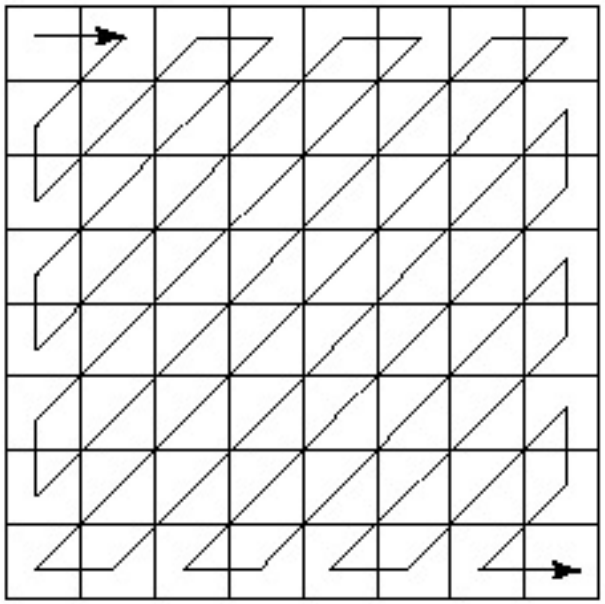
- DC coefficient: Predictive coding
  - The DC value of the current block is predicted from that of the previous block, and the error is coded using Huffman coding
- AC Coefficients: Runlength coding
  - Many high frequency AC coefficients are zero after first few low-frequency coefficients
  - Runlength 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
    - 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.

Example: Run-length Coding

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

Zig-zag ordering of DCT coefficients:



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

Coding of DC Symbols (not required)

- 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 (not required)

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 (not required)

- 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 (not required)

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

Blocking artifacts!

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
  - 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 (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

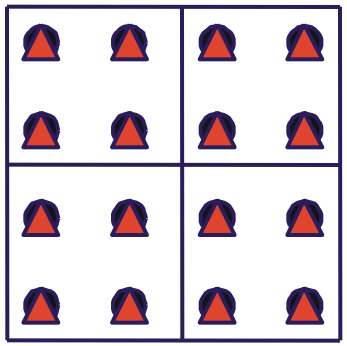
RGB <-> YCbCr Conversion

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

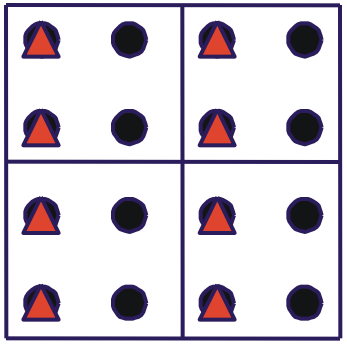
$$\begin{bmatrix} R \\ G \\ B \end{bmatrix} = \begin{bmatrix} 1.000 & -0.001 & 1.402 \\ 1.000 & -0.344 & -0.714 \\ 1.000 & 1.772 & 0.001 \end{bmatrix} \begin{bmatrix} Y \\ C_b - 128 \\ C_r - 128 \end{bmatrix}$$

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

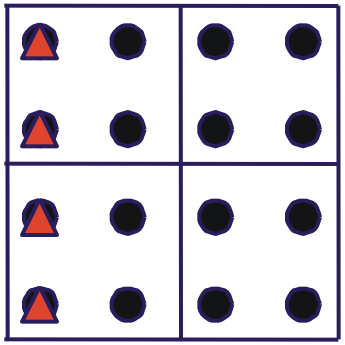
Chrominance Subsampling



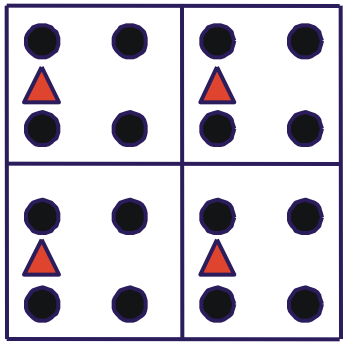
4:4:4
For every 2x2 Y Pixels
4 Cb & 4 Cr Pixel
(No subsampling)



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)



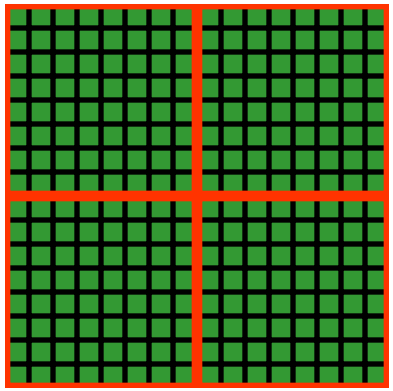
4:2:0
For every 2x2 Y Pixels
1 Cb & 1 Cr Pixel
(Subsampling by 2:1 both horizontally and vertically)

Y Pixel

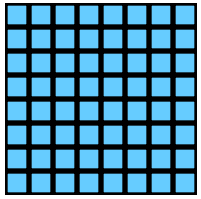
Cb and Cr Pixel

4:2:0 is the most common format

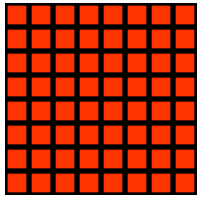
Coding Unit in JPEG



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

Pop Quiz

- How does JPEG work? Describe the main steps
- How were the quantization stepsizes determined?
- How are the quantized coefficients coded?
- How do you achieve different compression ratios? (file sizes)?

Pop Quiz (w/ Answers)

- How does JPEG work? Describe the main steps
  - Divide an image into blocks, code each block using transform coding (DCT transform, quantize coefficients, code the coefficients using runlength+Huffman coding)
- How were the quantization stepsizes determined?
  - Based on the human visual system sensitivity to different spatial frequency
- How are the quantized coefficients coded?
  - First convert the coefficients to a runlength representation and then code each pair of (runlength, non-zero value) using a predetermined Huffman codeword
- How do you achieve different compression ratios? (file sizes)?
  - By scaling the default quantization matrix by a scalar

Summary

- Orthonormal transform
  - Properties of orthonormal transform
  - 2D separable transform from 1D transform
  - Example transforms, DCT
- KL transform
  - Decorrelating signal components
  - Maximizing energy compaction
  - How to determine the KLT transform bases?
- Transform coding: general principles
  - Transform -> quantization -> binary encoding
- JPEG standard: block wise DCT
  - Blockwise DCT, perceptual quantization, runlength+Huffman coding
  - Simple, but has blocking artifacts at lower bit rates

Reading Assignment

- Reading assignment:
  - [Wang2002]] Wang, et al, Digital video processing and communications. Sec. 9.1 (Transform coding) (9.1.5 optional), Sec. 8.2-8.3 (Review of probability and information theory) (Sec. 8.3.2,8.3.3 optional), 8.4 (Binary encoding) (Sec 8.4.2 optional), 8.5 (scalar quantization)

- Optional readings
  - G. K. Wallace: The JPEG Still picture compression standard, *Communications of ACM*, April 1991.

Written Homework (2)

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. Assuming the mean value of the DC coefficient is 5, and the mean value for the AC coefficients is 0. Also, determine the reconstructed image from the quantized coefficients.

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

Written Homework (3)

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. For the DC coefficient, first predict it from the DC value of the previous block (which you can assume = 1200), and then quantize the prediction error. 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).

$$dctblock = 1.0e+003 * \begin{bmatrix} 1.3676 & -0.0500 & -0.0466 & 0.0912 \\ 0.0134 & -0.0033 & 0.0877 & 0.0071 \\ -0.0086 & -0.0207 & 0.0036 & 0.0019 \\ -0.0046 & 0.0086 & 0.0044 & 0.0085 \end{bmatrix},$$

$$Q = \begin{bmatrix} 16 & 10 & 24 & 51 \\ 14 & 16 & 40 & 69 \\ 18 & 37 & 68 & 103 \\ 49 & 78 & 103 & 120 \end{bmatrix},$$