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

- 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
  \[ t_i - t_{i-1} = r_i - r_{i-1} = q \]

• Parameters of Uniform Quantization
  – L: levels (\( L = 2^R \))
  – B: dynamic range \( B = f_{\text{max}} - f_{\text{min}} \)
  – q: quantization interval (step size)
  – \( q = B/L = B2^{-R} \)
Uniform Quantization:
Functional Representation

\[ Q(f) = \left\lfloor \frac{f - f_{\text{min}}}{q} \right\rfloor \times q + \frac{q}{2} + f_{\text{min}} \]

\[ r_0 = f_{\text{min}} + q/2 \]

\[ r_7 = f_{\text{max}} - q/2 \]

\[ I(f) = \left\lfloor \frac{f - f_{\text{min}}}{q} \right\rfloor \]

The stepsize is given by \( q = (f_{\text{max}} - f_{\text{min}})/L \).

The notation \( \left\lfloor x \right\rfloor \) returns the biggest integer that is smaller than or equal to \( x \).

\( I(f) \) is called the reconstruction level index, which indicates which reconstruction level is used for \( f \).
Uniform Quantization on Images

Original, $L=256$

$q=16, L=16$

$q=8, L=32$

$q=64, L=4$
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.

Matlab code:

```matlab
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));
```

L = 16, MSE = 21.6
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 (orthonormal)* 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
  – Karhunen 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 $\mathbb{C}^N$ represent the N dimensional complex space.

• Let $\mathbf{h}_0$, $\mathbf{h}_1$, ..., $\mathbf{h}_{N-1}$ represent N linearly independent vectors in $\mathbb{C}^N$.

• Any vector $\mathbf{f} \in \mathbb{C}^N$ can be represented as a linear combination of $\mathbf{h}_0$, $\mathbf{h}_1$, ..., $\mathbf{h}_{N-1}$:

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

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

$\mathbf{t} = \mathbf{B}^{-1}\mathbf{f} = \mathbf{Af}$

f and $\mathbf{t}$ form a transform pair
Inner Product

- Definition of inner product
\[ <f_1, f_2> = f_1^H f_2 = \sum_{n=0}^{N-1} f_1^*(n)f_2(n) \]

- Orthogonal
\[ <f_1, f_2> = 0 \]

- Norm of a vector
\[ \|f\|^2 = <f, f> = f^H f = \sum_{n=0}^{N-1} |f(n)|^2 \]

- Normalized vector: unit norm
\[ \|f\|^2 = 1 \]

- Orthonormal = orthogonal + normalized
Orthonormal Basis Vectors (OBV)

• \{h_k, k=0,…N-1\} are OBV if

\[
< h_k, h_l > = \delta_{k,l} = \begin{cases} 
1 & k = l \\
0 & k \neq l 
\end{cases}
\]

• With OBV

\[
< h_l, f > = < h_l, \sum_{k=0}^{N-1} t(k)h_k > = \sum_{k=0}^{N-1} t(k) < h_l, h_k > = t(l) = h_l^H f
\]

\[
t = \begin{bmatrix}
    h_0^H \\
h_1^H \\
    \vdots \\
h_{N-1}^H
\end{bmatrix}
\]

\[
f = B^H f = Af
\]

\[
B^{-1} = B^H, \text{ or } B^H B = BB^H = I.
\]

B is unitary
Definition of Unitary Transform

- Basis vectors are orthonormal
- Forward transform
  \[ t(k) = \langle h_k, f \rangle = \sum_{n=0}^{N-1} h_k(n)^* f(n), \]
  \[ t = \begin{bmatrix} h_0^H \\ h_1^H \\ \vdots \\ h_{N-1}^H \end{bmatrix} f = B^H f = Af \]
- Inverse transform
  \[ f(n) = \sum_{k=0}^{N-1} t(k) h_k(n), \]
  \[ f = \sum_{k=0}^{N-1} t(k) h_k = [h_0 \ h_1 \ \cdots \ h_{N-1}] t = B t = A^H t \]
Example of 1D Unitary Transform

\[
\begin{bmatrix}
1/2 \\
1/2 \\
1/2 \\
1/2
\end{bmatrix}, \quad
\begin{bmatrix}
1/2 \\
1/2 \\
-1/2 \\
-1/2
\end{bmatrix}, \quad
\begin{bmatrix}
1/2 \\
-1/2 \\
-1/2 \\
1/2
\end{bmatrix}, \quad
\begin{bmatrix}
1/2 \\
-1/2 \\
1/2 \\
-1/2
\end{bmatrix},
\]

\[
f = \begin{bmatrix} 1 \\ 2 \\ 3 \\ 4 \end{bmatrix} \quad \Rightarrow \quad \begin{cases} t_0 = 5 \\ t_1 = -2 \\ t_2 = 0 \\ t_3 = -1 \end{cases}
\]
1D DFT as a Unitary Transform

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

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

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

\[ h_k = \frac{1}{\sqrt{N}} \begin{bmatrix}
1 \\
\vdots \\
1 - 2k \\
e^{j2\pi \frac{(N-1)k}{N}} \end{bmatrix}, \quad k = 0, 1, \ldots, N - 1. \]
Example: 1D DFT, N=2

\[ N = 2 \text{ case: there are only two basis vectors:} \]

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

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

Using \( t_k = \langle \mathbf{h}_k, \mathbf{f} \rangle \), we obtain

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

\[
t_1 = \frac{1}{\sqrt{2}} \left( \begin{bmatrix} 1 \\ -1 \end{bmatrix}, \begin{bmatrix} 1 \\ 2 \end{bmatrix} \right) = \frac{1}{\sqrt{2}} (1*1 - 1*2) = -\frac{1}{\sqrt{2}}
\]

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

\[
N = 4 \text{ case: using } h_k = \frac{1}{2} \begin{bmatrix}
\exp(j2\pi \frac{k}{4} 0) \\
\exp(j2\pi \frac{k}{4} 1) \\
\exp(j2\pi \frac{k}{4} 2) \\
\exp(j2\pi \frac{k}{4} 3)
\end{bmatrix}
\]

yields:
\[
h_0 = \frac{1}{2} \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}; h_1 = \frac{1}{2} \begin{bmatrix} 1 \\ j \\ -1 \\ -j \end{bmatrix}; h_2 = \frac{1}{2} \begin{bmatrix} 1 \\ -1 \\ 1 \\ -1 \end{bmatrix}; h_3 = \frac{1}{2} \begin{bmatrix} 1 \\ -j \\ -1 \\ j \end{bmatrix}
\]

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

Verify:
\[
t_0h_0 + t_1h_1 + t_2h_2 + t_3h_3 = \frac{1}{4} \begin{bmatrix} 14 - (3 + j) - (3 - j) \\ 14 - (3 + j)j + (3 - j)j \\ 14 + (3 + j) + (3 - j) \\ 14 + (3 + j)j - (3 - j)j \end{bmatrix} = \frac{1}{4} \begin{bmatrix} 8 \\ 16 \\ 20 \\ 12 \end{bmatrix} = f
\]
1D Discrete Cosine Transform (DCT)

Basis Vectors:

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

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

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

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

Vector Representation \( N = 4 \) case:

\[
\mathbf{h}_k = \alpha(k) \begin{bmatrix}
\cos \left( \frac{1}{8} k\pi \right) \\
\cos \left( \frac{3}{8} k\pi \right) \\
\cos \left( \frac{5}{8} k\pi \right) \\
\cos \left( \frac{7}{8} k\pi \right)
\end{bmatrix}
\]

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} \]

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

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

Yao Wang, NYU-Poly
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:

\[
F = \sum_{k=0}^{M-1} \sum_{l=0}^{N-1} T(k,l) H_{k,l},
\]

\[
F(m,n) = \sum_{k=0}^{M-1} \sum_{l=0}^{N-1} T(k,l) H_{k,l}(m,n)
\]
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
  \[ < F_1, F_2 > = \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} F_1^*(m,n)F_2(m,n) \]

- Norm of a Matrix
  \[ \|F\| = < F, F > = \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} |F(m,n)|^2 \]

- A set of basis images \( \{ H_{k,l}, k=0,1,\ldots,M-1, \ l=0,1,\ldots,N-1 \} \) is orthonormal if
  \[ < H_{k,l}, H_{i,j} > = \delta_{k,i} \delta_{l,j} = \begin{cases} 1, & \text{if } k = i, l = j \\ 0, & \text{otherwise} \end{cases} \]
Two Dimensional Unitary Transform

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

\[
T(k,l) = \langle H_{k,l}, F \rangle = \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} H_{k,l}^*(m,n) F(m,n)
\]

- Inverse transform

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

\[
F = \sum_{k=0}^{M-1} \sum_{l=0}^{N-1} T(k,l) H_{k,l}
\]
Example of 2D Unitary Transform

\[
\mathbf{H}_{00} = \begin{bmatrix} \frac{1}{2} & \frac{1}{2} \\ \frac{1}{2} & \frac{1}{2} \end{bmatrix}, \quad \mathbf{H}_{01} = \begin{bmatrix} \frac{1}{2} & \frac{1}{2} \\ -\frac{1}{2} & -\frac{1}{2} \end{bmatrix}, \quad \mathbf{H}_{10} = \begin{bmatrix} \frac{1}{2} & -\frac{1}{2} \\ \frac{1}{2} & -\frac{1}{2} \end{bmatrix}, \quad \mathbf{H}_{11} = \begin{bmatrix} \frac{1}{2} & -\frac{1}{2} \\ -\frac{1}{2} & \frac{1}{2} \end{bmatrix},
\]

\[
\mathbf{F} = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \Rightarrow \begin{cases} T(0,0) = 5 \\ T(0,1) = -2 \\ T(1,0) = -1 \\ T(1,1) = 0 \end{cases}
\]
Separable Unitary Transform

• Let $h_k$, $k=0, 1, \ldots, M-1$ represent orthonormal basis vectors in $\mathbb{C}^M$,
• Let $g_l$, $l=0, 1, \ldots, N-1$ represent orthonormal basis vectors in $\mathbb{C}^N$,
• $H_{k,l} = h_k g_l^T$, or $H_{k,l}(m,n) = h_k(m)g_l(n)$.
• Then $H_{k,l}$ will form an orthonormal basis set in $\mathbb{C}^{M \times N}$.
Example of Separable Unitary Transform

- Example 1

\[ h_0 = \begin{bmatrix} 1/\sqrt{2} \\ 1/\sqrt{2} \end{bmatrix}, \quad h_1 = \begin{bmatrix} 1/\sqrt{2} \\ -1/\sqrt{2} \end{bmatrix}. \]

\[ H_{00} = h_0 h_0^T = \begin{bmatrix} 1/2 & 1/2 \\ 1/2 & 1/2 \end{bmatrix}, \quad H_{01} = h_0 h_1^T = \begin{bmatrix} 1/2 & -1/2 \\ 1/2 & -1/2 \end{bmatrix}. \]

\[ H_{10} = h_1 h_0^T = \begin{bmatrix} 1/2 & 1/2 \\ -1/2 & -1/2 \end{bmatrix}, \quad H_{11} = h_1 h_1^T = \begin{bmatrix} 1/2 & -1/2 \\ -1/2 & 1/2 \end{bmatrix}. \]
Example: 4x4 DFT

Recall the 1D DFT basis are: 

\[ h_0 = \frac{1}{2} \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}; h_1 = \frac{1}{2} \begin{bmatrix} 1 \\ j \\ -1 \\ -j \end{bmatrix}; h_2 = \frac{1}{2} \begin{bmatrix} 1 \\ -1 \\ 1 \\ -1 \end{bmatrix}; h_3 = \frac{1}{2} \begin{bmatrix} 1 \\ -j \\ 1 \\ j \end{bmatrix} \]

using \( H_{k,l} = h_k(h_l)^T \) yields:

\[
\begin{align*}
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}, \\
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}, \\
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}, \\
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}, \\
H_{1,0} &= \frac{1}{4} \begin{bmatrix} 1 & 1 & 1 & 1 \\ j & j & j & j \\ -j & -j & -j & -j \\ -j & -j & -j & -j \end{bmatrix}, \\
H_{1,1} &= \frac{1}{4} \begin{bmatrix} 1 & j & -1 & -j \\ j & -1 & -j & 1 \\ -j & 1 & j & -1 \\ -j & 1 & j & -1 \end{bmatrix}, \\
H_{1,2} &= \frac{1}{4} \begin{bmatrix} 1 & -1 & 1 & -1 \\ j & -j & j & -j \\ -j & j & -j & j \\ -j & j & -j & j \end{bmatrix}, \\
H_{1,3} &= \frac{1}{4} \begin{bmatrix} 1 & -j & 1 & j \\ j & 1 & -j & -1 \\ j & 1 & -j & -1 \\ j & 1 & -j & -1 \end{bmatrix}, \\
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}, \\
H_{2,1} &= \frac{1}{4} \begin{bmatrix} 1 & j & -1 & -j \\ -1 & 1 & -j & 1 \\ 1 & j & -1 & -j \\ -1 & 1 & -j & 1 \end{bmatrix}, \\
H_{2,2} &= \frac{1}{4} \begin{bmatrix} 1 & -1 & 1 & -1 \\ -j & j & -j & j \\ -j & j & -j & j \\ -j & j & -j & j \end{bmatrix}, \\
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}, \\
H_{3,0} &= \frac{1}{4} \begin{bmatrix} 1 & 1 & 1 & 1 \\ -j & -j & -j & -j \\ -j & -j & -j & -j \\ j & j & j & j \end{bmatrix}, \\
H_{3,1} &= \frac{1}{4} \begin{bmatrix} 1 & j & -1 & -j \\ -j & 1 & j & -1 \\ -j & 1 & j & -1 \\ j & j & j & j \end{bmatrix}, \\
H_{3,2} &= \frac{1}{4} \begin{bmatrix} 1 & -1 & 1 & -1 \\ -j & j & -j & j \\ -j & j & -j & j \\ j & j & j & j \end{bmatrix}, \\
H_{3,3} &= \frac{1}{4} \begin{bmatrix} 1 & j & -1 & -j \\ j & 1 & -j & -1 \\ j & 1 & -j & -1 \\ j & 1 & -j & -1 \end{bmatrix}.
\]
Example: 4x4 DFT

For \( F = \begin{bmatrix} 1 & 2 & 2 & 0 \\ 0 & 1 & 3 & 1 \\ 0 & 1 & 2 & 1 \\ 1 & 2 & 2 & -1 \end{bmatrix} \)

compute \( T_{k,l} \)

Using \( T_{k,l} = \langle H_{k,l}, F \rangle \) yields, e.g.,

\[
T_{0,0} = \langle H_{0,0}, F \rangle = \frac{1}{4} \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} 1 & 2 & 2 & 0 \\ 1 & 3 & 1 & 1 \\ 0 & 1 & 2 & 1 \\ 1 & 2 & 2 & -1 \end{bmatrix} = \frac{1}{4} (1 + 2 + 2 + 0 + 0 + 1 + 3 + 1 + 0 + 1 + 2 + 1 + 2 + 1 + 2 - 1) = \frac{18}{4}
\]

\[
T_{2,3} = \langle H_{2,3}, F \rangle = \frac{1}{4} \begin{bmatrix} 1 & -j & -1 & j \\ -1 & j & 1 & -j \\ 1 & -j & -1 & j \\ -1 & j & 1 & -j \end{bmatrix} \begin{bmatrix} 1 & 2 & 2 & 0 \\ 0 & 1 & 3 & 1 \\ 0 & 1 & 2 & 1 \\ 1 & 2 & 2 & -1 \end{bmatrix} = \frac{1}{4} (1 + 2j - 2 - j + 3 + j + 2 - j - 1 - 2j + 2 - j) = \frac{1}{4} (1 - j)
\]
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 \( \alpha(k) = \begin{cases} \sqrt{\frac{1}{N}} & k = 0 \\ \sqrt{\frac{1}{N}} & k = 1, \ldots, N - 1 \end{cases} \)
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
 -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
 -1.4562  -13.3225   -0.8750    1.3248    10.3817   16.0762    4.4157    1.1041

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

Coefficient Index (Zig-Zag Order)

Variance

Coefficient Index (Zig-Zag Order)
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

<table>
<thead>
<tr>
<th>16</th>
<th>11</th>
<th>10</th>
<th>16</th>
<th>24</th>
<th>40</th>
<th>51</th>
<th>61</th>
</tr>
</thead>
<tbody>
<tr>
<td>12</td>
<td>12</td>
<td>14</td>
<td>19</td>
<td>26</td>
<td>58</td>
<td>60</td>
<td>55</td>
</tr>
<tr>
<td>14</td>
<td>13</td>
<td>16</td>
<td>24</td>
<td>40</td>
<td>57</td>
<td>69</td>
<td>56</td>
</tr>
<tr>
<td>14</td>
<td>17</td>
<td>22</td>
<td>29</td>
<td>51</td>
<td>87</td>
<td>80</td>
<td>62</td>
</tr>
<tr>
<td>18</td>
<td>22</td>
<td>37</td>
<td>56</td>
<td>68</td>
<td>109</td>
<td>103</td>
<td>77</td>
</tr>
<tr>
<td>24</td>
<td>35</td>
<td>55</td>
<td>64</td>
<td>81</td>
<td>104</td>
<td>113</td>
<td>92</td>
</tr>
<tr>
<td>49</td>
<td>64</td>
<td>78</td>
<td>87</td>
<td>103</td>
<td>121</td>
<td>120</td>
<td>101</td>
</tr>
<tr>
<td>72</td>
<td>92</td>
<td>95</td>
<td>98</td>
<td>112</td>
<td>100</td>
<td>103</td>
<td>99</td>
</tr>
</tbody>
</table>

Actual step size for C(i,j): $Q(i,j) = QP \times M(i,j)$
Example: Quantized Indices

```
>> dctblock = dct2(imblock)

dctblock =
    31.0000    51.7034    1.1673   -24.5837   -12.0000   -25.7508    11.9640    23.2873
   -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
     0     0     0     0     0     0     0     0

Only 19 coefficients are retained out of 64
```
**Example: Quantized Coefficients**

<table>
<thead>
<tr>
<th>%dequantized DCT block</th>
</tr>
</thead>
<tbody>
<tr>
<td>&gt;&gt; iqdet=qdet.*QM</td>
</tr>
<tr>
<td>iqdet=</td>
</tr>
<tr>
<td>32  55  0  -32  0  -40  0  0</td>
</tr>
<tr>
<td>108 12  -14  38  0  58  0  0</td>
</tr>
<tr>
<td>196 13  -16  0  -40  0  0  0</td>
</tr>
<tr>
<td>42  -17  -22  -29  0  0  0  0</td>
</tr>
<tr>
<td>36  -22  0  0  0  0  0  0</td>
</tr>
<tr>
<td>0  0  0  0  0  0  0  0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Original DCT block</th>
</tr>
</thead>
<tbody>
<tr>
<td>dctblock=</td>
</tr>
<tr>
<td>31.0000  51.7034  1.1673  -24.5837  -12.0000  -25.7508  11.9640  23.2873</td>
</tr>
<tr>
<td>-1.4562  -13.3225  -0.8750  1.3248  10.3817  16.0762  4.4157  1.1041</td>
</tr>
<tr>
<td>-6.7720  -2.8384  4.1187  1.1118  10.5527  -2.7348  -3.2327  1.5799</td>
</tr>
</tbody>
</table>

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
-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: 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 =

\[
\begin{bmatrix}
2 & 5 & 0 & -2 & 0 & -1 & 0 & 0 \\
9 & 1 & -1 & 2 & 0 & 1 & 0 & 0 \\
14 & 1 & -1 & 0 & -1 & 0 & 0 & 0 \\
3 & -1 & -1 & -1 & 0 & 0 & 0 & 0 \\
2 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\end{bmatrix}
\]

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”
TABLE 8.17
JPEG coefficient coding categories.

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

TABLE 8.18
JPEG default DC code (luminance).

<table>
<thead>
<tr>
<th>Category</th>
<th>Base Code</th>
<th>Length</th>
<th>Category</th>
<th>Base Code</th>
<th>Length</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>010</td>
<td>3</td>
<td>6</td>
<td>1110</td>
<td>10</td>
</tr>
<tr>
<td>1</td>
<td>011</td>
<td>4</td>
<td>7</td>
<td>11110</td>
<td>12</td>
</tr>
<tr>
<td>2</td>
<td>100</td>
<td>5</td>
<td>8</td>
<td>111110</td>
<td>14</td>
</tr>
<tr>
<td>3</td>
<td>00</td>
<td>5</td>
<td>9</td>
<td>1111110</td>
<td>16</td>
</tr>
<tr>
<td>4</td>
<td>101</td>
<td></td>
<td>A</td>
<td>11111110</td>
<td>18</td>
</tr>
<tr>
<td>5</td>
<td>110</td>
<td>8</td>
<td>B</td>
<td>111111110</td>
<td>20</td>
</tr>
</tbody>
</table>
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

<table>
<thead>
<tr>
<th>Run/Category</th>
<th>Base Code</th>
<th>Length</th>
<th>Run/Category</th>
<th>Base Code</th>
<th>Length</th>
</tr>
</thead>
<tbody>
<tr>
<td>0/0 1010 (= EOB)</td>
<td>4</td>
<td>8/1 1111010</td>
<td>9</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0/1 00</td>
<td>3</td>
<td>8/2 11111111000000</td>
<td>17</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0/2 01</td>
<td>4</td>
<td>8/3 111111110110111</td>
<td>19</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0/3 100</td>
<td>6</td>
<td>8/4 1111111101110000</td>
<td>20</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0/4 1011</td>
<td>8</td>
<td>8/5 111111110111001</td>
<td>21</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0/5 11010</td>
<td>10</td>
<td>8/6 111111110111010</td>
<td>22</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0/6 111000</td>
<td>12</td>
<td>8/7 111111110111011</td>
<td>23</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0/7 1111000</td>
<td>14</td>
<td>8/8 111111110111100</td>
<td>24</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0/8 111110110</td>
<td>18</td>
<td>8/9 111111110111101</td>
<td>25</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0/9 111111111000010</td>
<td>25</td>
<td>8/A 111111110111110</td>
<td>26</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0/A 111111111100011</td>
<td>26</td>
<td>9/1 1111111000</td>
<td>10</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1/1 1100</td>
<td>5</td>
<td>9/2 11111110111111</td>
<td>18</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1/2 111001</td>
<td>8</td>
<td>9/3 111111111000000</td>
<td>19</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1/3 1111001</td>
<td>10</td>
<td>9/4 111111111000001</td>
<td>20</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1/4 111110110</td>
<td>13</td>
<td>9/5 111111111000010</td>
<td>21</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1/5 1111110110</td>
<td>16</td>
<td>9/6 111111111000011</td>
<td>22</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1/6 111111111000100</td>
<td>22</td>
<td>9/7 111111111000010</td>
<td>23</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1/7 111111111100010</td>
<td>23</td>
<td>9/8 11111111100001</td>
<td>24</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1/8 111111111100011</td>
<td>24</td>
<td>9/9 111111111000010</td>
<td>25</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1/9 111111111100011</td>
<td>25</td>
<td>9/A 111111111000011</td>
<td>26</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1/A 1111111111000100</td>
<td>26</td>
<td>A/1 1111111001</td>
<td>10</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2/1 11011</td>
<td>6</td>
<td>A/2 111111111000000</td>
<td>18</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2/2 1111000</td>
<td>10</td>
<td>A/3 11111111000001</td>
<td>19</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2/3 111110111</td>
<td>13</td>
<td>A/4 111111111000100</td>
<td>20</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2/4 111111111000100</td>
<td>20</td>
<td>A/5 111111111000110</td>
<td>21</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2/5 111111111000110</td>
<td>21</td>
<td>A/6 111111111001100</td>
<td>22</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2/6 111111111001100</td>
<td>22</td>
<td>A/7 111111111001101</td>
<td>23</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2/7 111111111001100</td>
<td>23</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**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
  – 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

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

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

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

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

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.
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

• 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 $S$, $T_{k,l}$, $k=0,1; l=0,1$.
   
   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?

   
   $S = \begin{bmatrix} 9 & 1 \\ 1 & 9 \end{bmatrix}, W_1 = \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix}, W_2 = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix}, W_3 = \begin{bmatrix} 0 & 0 \\ 1 & 0 \end{bmatrix}, W_4 = \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix}, W_5 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$

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.

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

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

\[
Q = \begin{bmatrix}
16 & 10 & 24 & 51 \\
14 & 16 & 40 & 69 \\
18 & 37 & 68 & 103 \\
49 & 78 & 103 & 120 \\
\end{bmatrix}.
\]
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 following 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